MC68HC908AS32A

Data Sheet

M68HC08
Microcontrollers

MC68HC908AS32A
Rev. 2.0
05/2006

freescale.com

freescale™
semiconductor

MC68HC908AS32A

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/

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision History

Date	Revision Level	Description	Page Number(s)
September, 2005	1.0	Reformatted to meet new publications guidelines. Modules updated with additional data	Throughout
		1.4.16 BDLC Receive Pin (BDRxD) — Corrected name of BDLC receive pin to BDRxD.	25
		Removed Keyboard Interface Module	N/A
		Removed Timer Interface Module B Removed all references to TIMB and TBCLK	N/A
May, 2006	2.0	Figure 1-3. 64-Pin QFP Assignments (Top View) — Added pin assignment diagram for the 64-pin QFP.	23
		Figure 2-2. I/O Data, Status and Control Registers — Corrected two register entries: SPI Status and Control Register (SPSCR) FLASH Control Register (FLCR)	 32 38
		Chapter 20 Ordering Information and Mechanical Specifications — Added information pertaining to the 64-pin quad flag pack (QFP).	271

List of Chapters

Table of Contents

Chapter 1
General Description

Chapter 2
Memory

Chapter 3
Analog-to-Digital Converter (ADC)

Chapter 4
Byte Data Link Controller (BDLC)

Chapter 5
Clock Generator Module (CGM)

Chapter 6
Configuration Register (CONFIG1)

Chapter 7
Configuration Register (CONFIG2)

Chapter 8
Computer Operating Properly (COP)

Chapter 9
Central Processor Unit (CPU)

Chapter 10
External Interrupt Module (IRQ)

Chapter 11
Low-Voltage Inhibit (LVI)

Chapter 12
Programmable Interrupt Timer (PIT)

Chapter 13
Input/Output Ports

Chapter 14
Serial Communications Interface (SCI)

Chapter 15
System Integration Module (SIM)

Chapter 16
Serial Peripheral Interface (SPI)

Chapter 17
Timer Interface Module (TIM)

Chapter 18
Development Support

Chapter 19
Electrical Specifications

Chapter 20
Ordering Information and Mechanical Specifications

Chapter 1
General Description

1.1 Introduction

The MC68HC908AS32A 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:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- 8.4 MHz internal bus frequency
- 32,256 bytes of FLASH electrically erasable read-only memory (FLASH)
- FLASH data security[1]
- 512 bytes of on-chip electrically erasable programmable read-only memory with security option (EEPROM)[1]
- 1 Kbyte of on-chip RAM
- Clock generator module (CGM)
- Serial peripheral interface module (SPI)
- Serial communications interface module (SCI)
- 8-bit, 15-channel analog-to-digital converter (ADC)
- 16-bit, 6-channel timer interface module (TIM)
- Programmable interrupt timer (PIT)
- 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
- SAE J1850 byte data link controller digital module

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

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

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908AS32A.

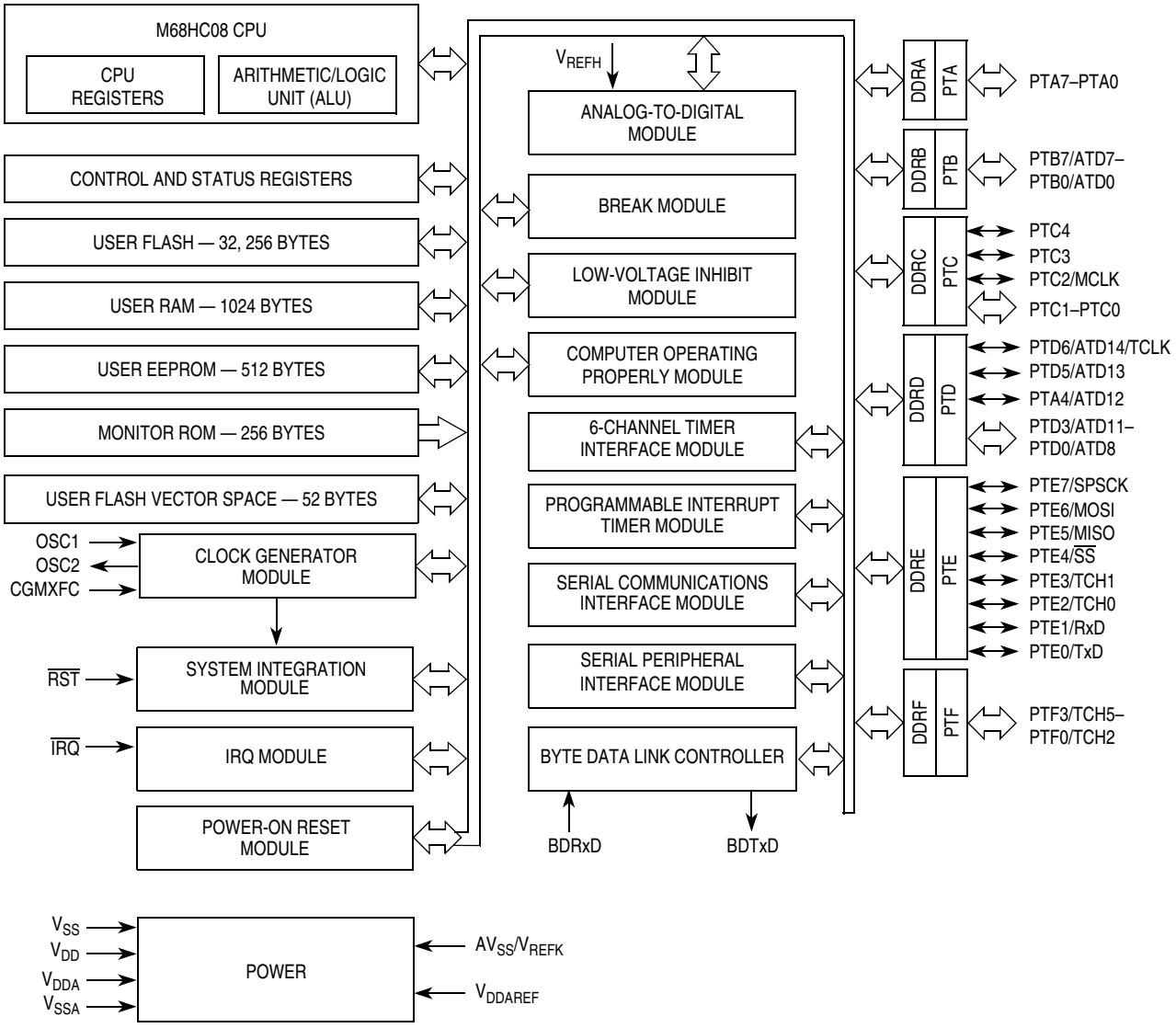


Figure 1-1. MCU Block Diagram for the MC68HC908AS32A

1.4 Pin Assignments

The MC68HC908AS32A is available in a 52-pin plastic leaded chip carrier (PLCC) and a 64-pin quad flat pack (QFP). Figure 1-2 and Figure 1-3 show the pin assignments for these packages.

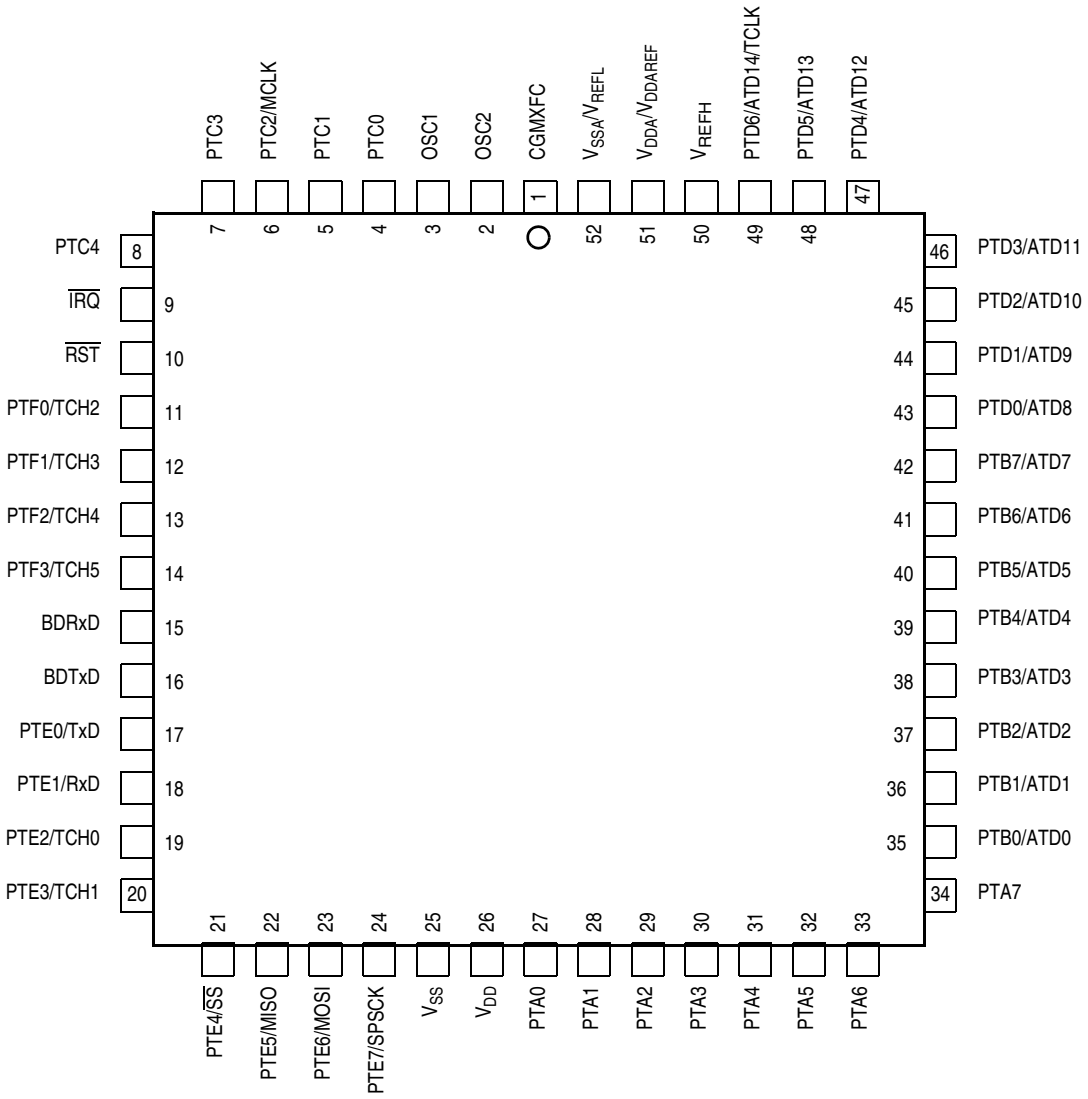


Figure 1-2. 52-Pin PLCC Assignments (Top View)

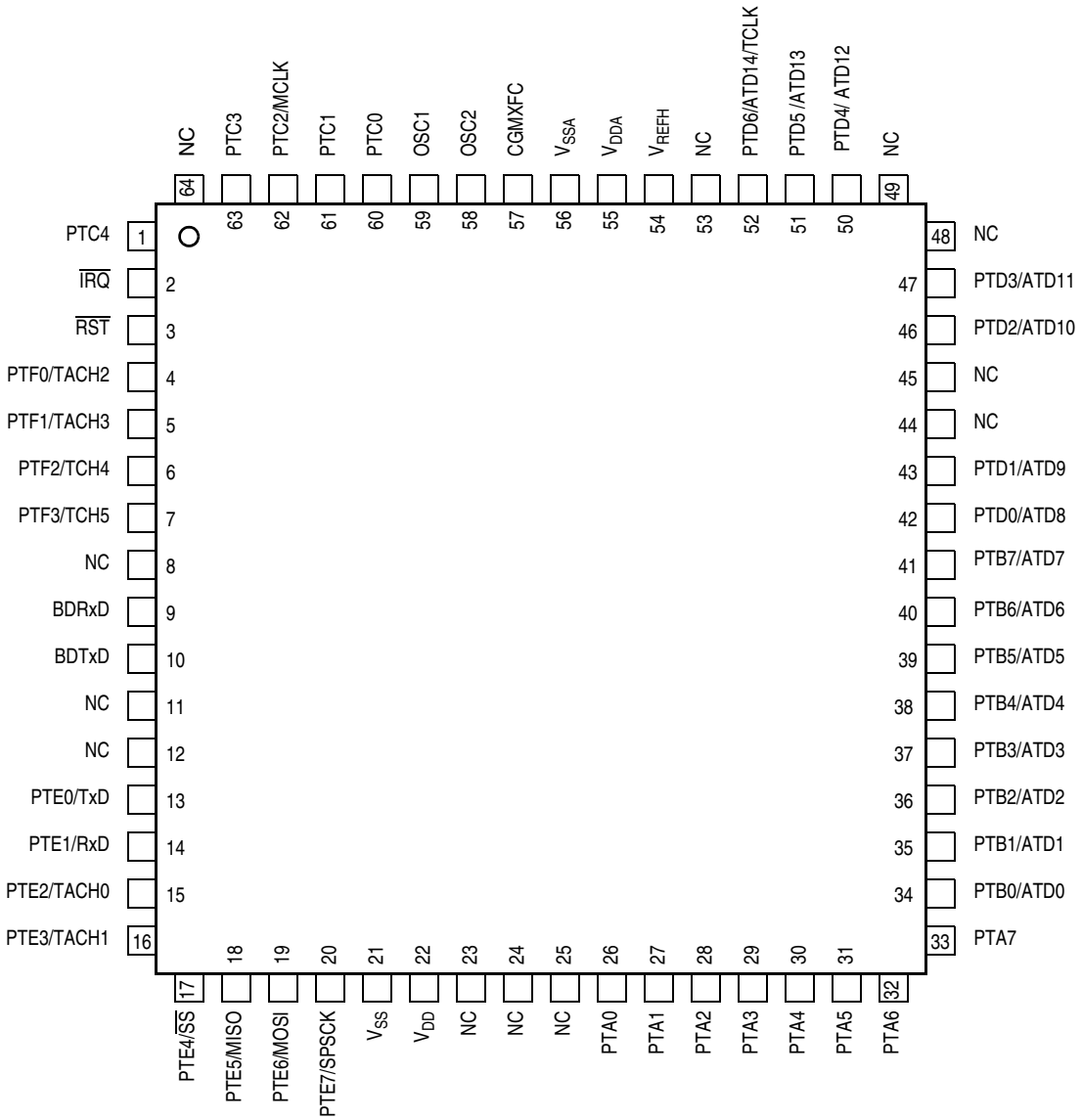


Figure 1-3. 64-Pin QFP Assignments (Top View)

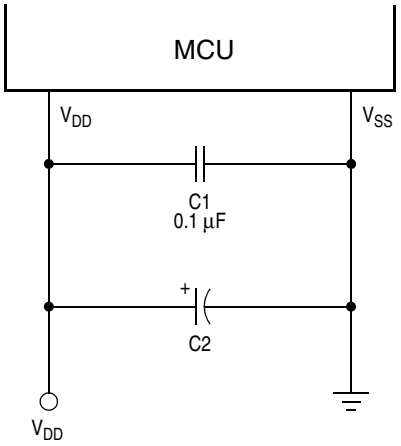
NOTE

The following pin descriptions are just a quick reference. For a more detailed representation, see Chapter 13 Input/Output Ports.

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

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

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



NOTE: Component values shown represent typical applications.

Figure 1-4. Power Supply Bypassing

V_{SS} is also the ground for the port output buffers and the ground return for the serial clock in the SPI. See Chapter 16 Serial Peripheral Interface (SPI) for more information.

NOTE

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

1.4.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. See Chapter 5 Clock Generator Module (CGM) for more information.

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

A 0 on the $\overline{RST}$ pin forces the MCU to a known startup 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) for more information.

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

$\overline{IRQ}$ is an asynchronous external interrupt pin. See Chapter 10 External Interrupt Module (IRQ) for more information.

1.4.5 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the clock generator module (CGM). See Chapter 5 Clock Generator Module (CGM) for more information.

1.4.6 Analog Power Supply Pin (V_{DDA}/V_{DDAREF})

V_{DDA}/V_{DDAREF} is the power supply pin for the analog portion of the ADC and the CGM. See Chapter 3 Analog-to-Digital Converter (ADC) and Chapter 5 Clock Generator Module (CGM) for more information.

1.4.7 Analog Ground Pin (V_{SSA}/V_{REFL})

The V_{SSA}/V_{REFL} pin provides both the analog ground connection and the reference low voltage for the ADC as well as the ground connection for the CGM. See Chapter 3 Analog-to-Digital Converter (ADC) and Chapter 5 Clock Generator Module (CGM) for more information.

1.4.8 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) for more information.

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

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

1.4.10 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) and Chapter 13 Input/Output Ports for more information.

1.4.11 Port C I/O Pins (PTC4–PTC0)

PTC4–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 which has a frequency equivalent to the system clock. See Chapter 13 Input/Output Ports for more information.

1.4.12 Port D I/O Pins (PTD6–PTD0/ATD8)

Port D is an 7-bit special-function port that shares seven of its pins with the ADC and one of its pins with the TIM. See Chapter 17 Timer Interface Module (TIM), Chapter 3 Analog-to-Digital Converter (ADC), and Chapter 13 Input/Output Ports for more information.

1.4.13 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 the TIM, 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 (TIM), and Chapter 13 Input/Output Ports for more information.

1.4.14 Port F I/O Pins (PTF3–PTF0/TCH2)

Port F is a 4-bit special function port that shares four of its pins with the TIM. See Chapter 17 Timer Interface Module (TIM) and Chapter 13 Input/Output Ports for more information.

1.4.15 BDLC Transmit Pin (BDTxD)

This pin is the digital output from the BDLC module (BDTxD). See Chapter 19 Electrical Specifications for more information.

1.4.16 BDLC Receive Pin (BDRxD)

This pin is the digital input to the BDLC module (BDRxD). See Chapter 19 Electrical Specifications for more information.

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
PTC4–PTC0	General-purpose I/O	Dual state	No	Input Hi-Z
PTD6/ATD14/TCLK ADC channel	General-purpose I/O ADC channel/timer external input clock	Dual state	No	Input Hi-Z
PTD5/ATD13 ADC channel	General-purpose I/O ADC channel	Dual state	No	Input Hi-Z
PTD4/ATD12 ADC channel	General-purpose I/O ADC channel	Dual state	No	Input Hi-Z
PTD3/ATD11–PTD0/ATD8 ADC channels	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	Input Hi-Z
PTE6/MOSI	General-purpose I/O SPI data path	Dual state Open drain	Yes	Input Hi-Z
PTE5/MISO	General-purpose I/O SPI data path	Dual state Open drain	Yes	Input Hi-Z
PTE4/$\overline{SS}$	General-purpose I/O SPI slave select	Dual state	Yes	Input Hi-Z
PTE3/TCH1	General-purpose I/O TIM channel 1	Dual state	Yes	Input Hi-Z
PTE2/TCH0	General-purpose I/O TIM channel 0	Dual state	Yes	Input Hi-Z
PTE1/RxD	General-purpose I/O SCI receive data	Dual state	Yes	Input Hi-Z
PTE0/TxD	General-purpose I/O SCI transmit data	Dual state	No	Input Hi-Z
PTF3/TCH5	General-purpose I/O TIM channel 5	Dual state	Yes	Input Hi-Z
PTF2/TCH4	General-purpose I/O TIM channel 4	Dual state	Yes	Input Hi-Z
PTF1/TCH3	General-purpose I/O TIM channel 3	Dual state	Yes	Input Hi-Z
PTF0/TCH2	General-purpose I/O TIM channel 2	Dual state	Yes	Input Hi-Z
V_{DD}	Chip power supply	N/A	N/A	N/A
V_{SS}	Chip ground	N/A	N/A	N/A
V_{DDA}/V_{DDAREF}	ADC analog power supply CGM analog power supply	N/A	N/A	N/A
V_{SSA}/V_{REFL}	ADC ground/ADC reference low voltage CGM analog ground	N/A	N/A	N/A
V_{REFH}	A/D reference high voltage	N/A	N/A	N/A
OSC1	External clock in	N/A	N/A	Input Hi-Z
OSC2	External clock out	N/A	N/A	Output
CGMXFC	PLL loop filter cap	N/A	N/A	N/A
$\overline{IRQ}$	External interrupt request	N/A	N/A	Input Hi-Z
$\overline{RST}$	Reset	N/A	N/A	Output low
BDRxD	BDLC serial input	N/A	Yes	Input Hi-Z
BDTxD	BDLC serial output	Output	No	Output

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

Table 1-2. Clock Signal Naming Conventions

Clock 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
TCLK	External clock input for TIM

Table 1-3. Clock Source Summary

Module	Clock Source
ADC	CGMXCLK or bus clock
CAN	CGMXCLK or CGMOUT
COP	CGMXCLK
CPU	Bus clock
FLASH	Bus clock
EEPROM	CGMXCLK or bus clock
RAM	Bus clock
SPI	Bus clock/SPSCK
SCI	CGMXCLK
TIM	Bus clock or PTD6/ATD14/TCLK
PIT	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, shown in Figure 2-1, includes:

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

2.2 Unimplemented Memory Locations

Accessing an unimplemented location can have unpredictable effects on MCU operation. In Figure 2-1 and in register figures in this document, unimplemented locations are shaded.

2.3 Reserved Memory Locations

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

2.4 Input/Output (I/O) Section

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

2.5 Additional Status and Control Registers

Selected addresses in the range $FE00–$FF88 contain additional status and control registers as shown in Figure 2-3. A noted exception is the computer operating properly (COP) control register (COPCTL) at address $FFFF.

2.6 Vector Addresses and Priority

Addresses in the range $FFDA–$FFFF contain the user-specified vector locations. The vector addresses are shown in Table 2-1. It is recommended that all vector addresses are defined.

$0000 ↓ $004F	I/O REGISTERS (80 BYTES)
$0050 ↓ $044F	RAM (1024 BYTES)
$0450 ↓ $07FF	UNIMPLEMENTED (944 BYTES)
$0800 ↓ $09FF	EEPROM (512 BYTES)
$0A00 ↓ $7FFF	UNIMPLEMENTED (30,208 BYTES)
$8000 ↓ $FDFF	FLASH (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	CONFIGURATION WRITE-ONCE REGISER 2 (CONFIG2)
$FE0A	RESERVED
$FE0B	RESERVED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10	EEPROM EEDIVH NONVOLATILE REGISTER (EEDIVHNVR)
$FE11	EEPROM EEDIVL NONVOLATILE REGISTER (EEDIVLNVR)
$FE12 ↓ $FE19	RESERVED
$FE1A	EEPROM EE DIVIDER HIGH REGISTER (EEDIVH)
$FE1B	EEPROM EE DIVIDER LOW REGISTER (EEDIVL)
$FE1C	EEPROM NONVOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION REGISTER (EEACR)

Figure 2-1. Memory Map

$FE20 ↓ $FF1F	MONITOR ROM (256 BYTES)
$FF20 ↓ $FF7F	UNIMPLEMENTED (80 BYTES)
$FF80	FLASH BLOCK PROTECT REGISTER (FLBPR)
$FF81 ↓ $FF87	RESERVED (7 BYTES)
$FF88	FLASH CONTROL REGISTER (FLCR)
$FF89 ↓ $FF8F	RESERVED (6 BYTES)
$FF90 ↓ $FFBF	UNIMPLEMENTED (48 BYTES)
$FFC0 ↓ $FFD9	RESERVED (26 BYTES)
$FFDA ↓ $FFFF	VECTORS (38 BYTES)

Figure 2-1. Memory Map (Continued)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA) See page 151.	Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB) See page 153.	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC) See page 155.	Read:	0	0	0	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD) See page 157.	Read:	0	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Write:								
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA) See page 151.	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0005	Data Direction Register B (DDRB) See page 153.	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 155.	Read:	MCLKEN	0	0	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD) See page 158.	Read:	0	DDRD6	DDRD5	DDRD4	DDRD3	DDR2	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE) See page 159.	Read:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Write:								
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF) See page 161.	Read:	0	0	0	0	PTF3	PTF2	PTF1	PTF0
		Write:								
		Reset:	Unaffected by reset							
$000A	Reserved		R	R	R	R	R	R	R	R
$000B	Reserved		R	R	R	R	R	R	R	R
$000C	Data Direction Register E (DDRE) See page 160.	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 162.	Read:	0	0	0	0	DDRF3	DDRF2	DDRF1	DDRF0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000E	Reserved		R	R	R	R	R	R	R	R
$000F	Reserved		R	R	R	R	R	R	R	R
$0010	SPI Control Register (SPCR) See page 221.	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 223.	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 225.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0013	SCI Control Register 1 (SCC1) See page 178.	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 180.	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 182.	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 183.	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 185.	Read:	0	0	0	0	0	0	BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR) See page 186.	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 186.	Read:	0	0	SCP1	SCP0	0	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001A	IRQ Status/Control Register (ISCR) See page 139.	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Reserved		R	R	R	R	R	R	R	R
$001C	PLL Control Register (PCTL) See page 106.	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 107.	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 108.	Read:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		Write:								
		Reset:	0	1	1	0	0	1	1	0
$001F	Configuration Write-Once Register (CONFIG1) See page 115.	Read:	LVISTOP	R	LVIRST	LVIPWR	SSREC	COPL	STOP	COPD
		Write:								
		Reset:	0	1	1	1	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	TIM Status and Control Register (TSC) See page 237.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021	Reserved		R	R	R	R	R	R	R	R
$0022	TIM Counter Register High (TCNTH) See page 238.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	TIM Counter Register Low (TCNTL) See page 238.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0024	TIM Modulo Register High (TMODH) See page 239.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	TIM Modulo Register Low (TMODL) See page 239.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0026	TIM Channel 0 Status and Control Register (TSC0) See page 240.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0027	TIM Channel 0 Register High (TCH0H) See page 243.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	TIM Channel 0 Register Low (TCH0L) See page 243.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:								
$0029	TIM Channel 1 Status and Control Register (TSC1) See page 240.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$002A	TIM Channel 1 Register High (TCH1H) See page 243.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002B	TIM Channel 1 Register Low (TCH1L) See page 243.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$002C	TIM Channel 2 Status and Control Register (TSC2) See page 240.	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002D	TIM Channel 2 Register High (TCH2H) See page 243.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$002E	TIM Channel 2 Register Low (TCH2L) See page 243.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$002F	TIM Channel 3 Status and Control Register (TSC3) See page 240.	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0030	TIM Channel 3 Register High (TCH3H) See page 243.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0031	TIM Channel 3 Register Low (TCH3L) See page 243.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0032	TIM Channel 4 Status and Control Register (TSC4) See page 240.	Read:	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0033	TIM Channel 4 Register High (TCH4H) See page 243.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0034	TIM Channel 4 Register Low (TCH4L) See page 243.	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0035	TIM Channel 5 Status and Control Register (TSC5) See page 240.	Read:	CH5F	CH5IE	0	MS5A	ELS5B	ELS5A	TOV5	CH5MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0036	TIM Channel 5 Register High (TCH5H) See page 243.	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0037	TIM Channel 5 Register Low (TCH5L) See page 243.	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 63.	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:	R							
		Reset:	0	0	0	0	0	0	0	0
$0039	ADC Data Register (ADR) See page 65.	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	R	R	R	R	R	R	R	R
		Reset:	Indeterminate after reset							

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$003A	ADC Input Clock Register (ADICLK) See page 65.	Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$003B	BDLC Analog and Roundtrip Delay Register (BARD) See page 85.	Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
		Write:			R	R				
		Reset:	1	1	0	0	0	1	1	1
$003C	BDLC Control Register 1 (BCR1) See page 86.	Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
		Write:					R	R		
		Reset:	1	1	1	0	0	0	0	0
$003D	BDLC Control Register 2 (BCR2) See page 88.	Read:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$003E	BDLC State Vector Register (BSVR) See page 92.	Read:	0	0	I3	I2	I1	I0	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$003F	BDLC Data Register (BDR) See page 94.	Read:	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
		Write:								
		Reset:	Unaffected by reset							
$0040 ↓ $004A	Reserved		R	R	R	R	R	R	R	R
$004B	PIT Status and Control Register (PSC) See page 147.	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 148.	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 148.	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 149.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$004F	PIT Counter Modulo Register Low (PMODL) See page 149.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1

= Unimplemented R = Reserved U = Unaffected

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR) See page 202.	Read:	R	R	R	R	R	R	BW	R
		Write:							See note	
		Reset:							0	
		Note: Writing a 0 clears BW								
$FE01	SIM Reset Status Register (SRSR) See page 202.	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$FE02	Reserved		R	R	R	R	R	R	R	R
$FE03	SIM Break Flag Control Register (SBFCR) See page 203.	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE09	Configuration Write-Once Register (CONFIG2) See page 117.	Read:	EEDIVCLK	R	R	R	AS32A	R	R	R
		Write:								
		Reset:	0	0	0	1	1	0	0	0
$FE0C	Break Address Register High (BRKH) See page 248.	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 248.	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 248.	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 142.	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 48.	Read:	EEDIV SECD					EEDIV10	EEDIV9	EEDIV8
		Write:								
		Reset:	Unaffected by reset; $FF when blank							
$FE11	EEDIV Low Nonvolatile Register (EEDIVLNVR) See page 48.	Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
		Write:								
		Reset:	Unaffected by reset; $FF when blank							
$FE1A	EEDIV Timebase Divider High Register (EEDIVH) See page 48.	Read:	EEDIV SECD					EEDIV10	EEDIV9	EEDIV8
		Write:								
		Reset:	Contents of EEDIVHNVR ($FE10), Bits [6:3] = 0							
$FE1B	EEDIV Timebase Divider Low Register (EEDIVL) See page 49.	Read:	EEDIV7	EEDIV6	EEDIV5	EEDIV4	EEDIV3	EEDIV2	EEDIV1	EEDIV0
		Write:								
		Reset:	Contents of EEDIVLNVR ($FE11)							

= Unimplemented R = Reserved

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE1C	EEPROM Nonvolatile Register (EENVR) See page 48.	Read:				EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
		Write:								
		Reset:	Programmed value or 1 in the erased state							
$FE1D	EEPROM Control Register (EECR) See page 45.	Read:		0	EEOFF	EERAS1	EERAS0	EELAT	AUTO	EEPGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE1F	EEPROM Array Configuration Register (EEACR) See page 46.	Read:				EEPRTCT	EEBP3	EEBP2	EEBP1	EEBP0
		Write:								
		Reset:	Contents of EENVR ($FE1C)							
$FF80	FLASH Block Protect Register (FLBPR) See page 52.	Read:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		Write:								
		Reset:	Unaffected by reset							
$FF88	FLASH Control Register (FLCR) See page 51.	Read:	0	0	0	0	HVEN	MASS	ERASE	PGM
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FFFF	COP Control Register (COPCTL) See page 121.	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)

Table 2-1. Vector Addresses

Vector Priority	Address	Vector
Lowest	$FFDA	PIT Vector (High)
	$FFDB	PIT Vector (Low)
	$FFDC	BDLC Vector (High)
	$FFDD	BDLC Vector (Low)
	$FFDE	ADC Vector (High)
	$FFDF	ADC Vector (Low)
	$FFE0	SCI Transmit Vector (High)
	$FFE1	SCI Transmit Vector (Low)
	$FFE2	SCI Receive Vector (High)
	$FFE3	SCI Receive Vector (Low)
	$FFE4	SCI Error Vector (High)
	$FFE5	SCI Error Vector (Low)
	$FFE6	SPI Transmit Vector (High)
	$FFE7	SPI Transmit Vector (Low)
	$FFE8	SPI Receive Vector (High)
	$FFE9	SPI Receive Vector (Low)
	$FFEA	TIM Overflow Vector (High)
	$FFEB	TIM Overflow Vector (Low)
	$FFEC	TIM Channel 5 Vector (High)
	$FFED	TIM Channel 5 Vector (Low)
	$FFEE	TIM Channel 4 Vector (High)
	$FFEF	TIM Channel 4 Vector (Low)
	$FFF0	TIM Channel 3 Vector (High)
	$FFF1	TIM Channel 3 Vector (Low)
	$FFF2	TIM Channel 2 Vector (High)
	$FFF3	TIM Channel 2 Vector (Low)
	$FFF4	TIM Channel 1 Vector (High)
	$FFF5	TIM Channel 1 Vector (Low)
	$FFF6	TIM Channel 0 Vector (High)
	$FFF7	TIM Channel 0 Vector (Low)
	$FFF8	PLL Vector (High)
	$FFF9	PLL Vector (Low)
	$FFFA	IRQ1 Vector (High)
	$FFFB	IRQ1 Vector (Low)
	$FFFC	SWI Vector (High)
	$FFFD	SWI Vector (Low)
	$FFFE	Reset Vector (High)
Highest	$FFFF	Reset Vector (Low)

2.7 Random-Access Memory (RAM)

The 1024 bytes of random-access memory (RAM) are located at address $0050–$044F is the RAM location. The 16-bit stack pointer allows the stack RAM to be anywhere in the 64 Kbyte memory space.

NOTE

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

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

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

NOTE

For M68HC05, M6805, and M146805 compatibility, the H register is not stacked.

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

NOTE

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

2.8 Electrically Erasable Programmable Read-Only Memory (EEPROM)

This subsection describes the 512 bytes of electrically erasable programmable read-only memory (EEPROM) residing at address range $0800–$09FF.

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
- Security option[1]
- AUTO bit driven programming/erasing time feature

2.8.1 Functional Description

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.

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

2.8.1.1 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 security option
- EEPROM block protection

EENVR and EEDIVNVR are nonvolatile EEPROM registers that 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 EEDIVNCR (two 8-bit registers: EEDIVHNVR and EEDIVLNVR), the corresponding volatile register is the EEPROM divider register (EEDIV: EEDIVH and EEDIVL).

2.8.1.2 EEPROM Timebase Requirements

A 35 µs timebase is required by the EEPROM control circuit for program and erase of EEPROM content. This timebase is derived from dividing the CGMXCLK or bus clock (selected by EEDIVCLK bit in CONFIG2 register) using a timebase divider circuit controlled by the 16-bit EEPROM timebase divider 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 (see 2.8.1.1 EEPROM Configuration and 2.8.1.2 EEPROM Timebase Requirements).

2.8.1.3 EEPROM Program/Erase Protection

The EEPROM has a special feature that designates the 16 bytes of addresses from $08F0–$08FF to be permanently secured. This program/erase protect option is enabled by programming the EEPRTCT bit in the EEPROM nonvolatile register to a 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 and $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 2.8.1.4 EEPROM Block Protection and 2.8.2.2 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.

2.8.1.4 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. See 2.8.2.2 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 0s 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 on programming a value into the EEDIVHNVR.

2.8.1.5 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 a 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 a 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.

2.8.1.6 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. See 2.8.1.7 EEPROM Programming, 2.8.1.8 EEPROM Erasing, and 2.8.2.1 EEPROM Control Register for more information.

2.8.1.7 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.*

2.8.1.8 EEPROM Erasing

The programmed state of an EEPROM bit is a 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.*

2.8.2 EEPROM Register Descriptions

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

2.8.2.1 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 Bit

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. See note D for 2.8.1.7 EEPROM Programming, 2.8.1.8 EEPROM Erasing, and 19.11.2 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.

2.8.2.2 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 Bits

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 2.8.1.3 EEPROM Program/Erase Protection).

1 = EEPROM security disabled
0 = EEPROM security enabled

NOTE

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.

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

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

Table 2-5. 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 erasing available	Byte programming available only byte erasing available
	EEBP3 = 1	Protected	Protected

2.8.2.3 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 2.8.2.1 EEPROM Control Register for individual bit descriptions).

Address: $FE1C

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

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.

2.8.2.4 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 nonvolatile registers (EEDIVHNVR and EEDIVLNVR) after a reset.

Address: $FE1A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	EEDIV SECD	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 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 CONFIG2 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.

2.8.2.5 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 EEDIVSECD is set to 1 in EEDIVH (see 2.8.2.4 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 0s 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.

2.8.3 Low-Power Modes

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

2.8.3.1 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.

2.8.3.2 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.

2.9 FLASH Memory (FLASH)

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

The FLASH memory is an array of 32,256 bytes with one byte of block protection and an additional 38 bytes of user vectors. An erased bit reads as a 1 and a programmed bit reads as a 0.

Memory in the FLASH array is organized into rows within pages. There are two rows of memory per page with 64 bytes per row. The minimum erase block size is a single page,128 bytes. Programming is performed on a per-row basis, 64 bytes at a time. Program and erase operations are facilitated through control bits in the FLASH control register (FLCR). Details for these operations appear later. The FLASH memory map consists of:

- $8000–$FDFF — user memory (32,256 bytes)
- $FF80 — FLASH block protect register (FLBPR)
- $FF88 — FLASH control register (FLCR)
- $FFCC–$FFFF — these locations are reserved for user-defined interrupt and reset vectors

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

NOTE

A security feature prevents viewing of the FLASH contents.[1]

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

2.9.1 FLASH Control and Block Protect Registers

The FLASH array has two registers that control its operation, the FLASH control register (FLCR) and the FLASH bock protect register (FLBPR).

2.9.1.1 FLASH Control Register

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

Address: $FF88

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

= Unimplemented

Figure 2-11. FLASH Control Register (FLCR)

HVEN — High-Voltage Enable Bit

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

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

MASS — Mass Erase Control Bit

Setting this read/write bit configures the FLASH array for mass or page erase operation.

1 = Mass erase operation selected
0 = Page erase operation selected

ERASE — Erase Control Bit

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

1 = Erase operation selected
0 = Erase operation unselected

PGM — Program Control Bit

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

1 = Program operation selected
0 = Program operation unselected

2.9.1.2 FLASH Block Protect Register

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

Address: $FF80

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

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

Figure 2-12. FLASH Block Protect Register (FLBPR)

FLBPR[7:0] — Block Protect Register Bits [7:0]

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

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

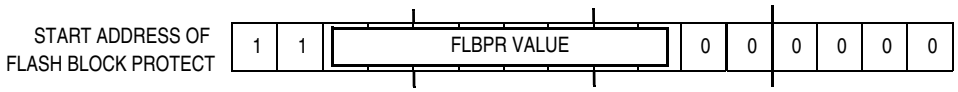


Figure 2-13. FLASH Block Protect Start Address

Decreasing the value in FLBPR by one increases the protected range by one page (128 bytes). However, programming the block protect register with $FE protects a range twice that size, 256 bytes, in the corresponding array. $FE means that locations $FF00–$FFFF are protected in FLASH. See Table 2-6.

The FLASH memory does not exist at some locations. The block protection range configuration is unaffected if FLASH memory does not exist in that range. Refer to the memory map (Figure 2-1) and make sure that the desired locations are protected.

Table 2-6. FLASH Protected Ranges

FLBPR[7:0]	Protected Range
$FF	No Protection
$FE	$FF00 – $FFFF
$FD	$FE80 – $FFFF
↓	↓
$0B	$8580 – $FFFF
$0A	$8500 – $FFFF
$09	$8480 – $FFFF
$08	$8400 – $FFFF
↓	↓
$04	$8200 – $FFFF
$03	$8180 – $FFFF
$02	$8100 – $FFFF
$01	$8080 – $FFFF
$00	$8000 – $FFFF

2.9.2 FLASH Block Protection

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

NOTE

In performing a program or erase operation, FLBPR must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.

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

When bits within FLBPR are programmed (0s), they lock a block of memory address ranges as shown in 2.9.1.2 FLASH Block Protect Register. If FLBPR is programmed with any value other than $FF, the protected block of FLASH memory can not be erased or programmed.

NOTE

The vector locations and the FLBPR are located in the same page. FLBPR is not protected with special hardware or software; therefore, if this page is not protected by FLBPR and the vector locations are erased by either a page or a mass erase operation, FLBPR will also be erased.

2.9.3 FLASH Page Erase Operation

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

1. Set the ERASE bit and clear the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH location within the address range of the block to be erased.
4. Wait for a time, t_{NVS} (minimum 10 μs).
5. Set the HVEN bit.
6. Wait for a time, t_{Erase} (minimum 1 ms or 4 ms).
7. Clear the ERASE bit.
8. Wait for a time, t_{NVH} (minimum 5 μs).
9. Clear the HVEN bit.
10. After time, t_{RCV} (typical 1 μs), the memory can be accessed in read mode again.

NOTE

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

CAUTION

A page erase of the vector page will erase the internal oscillator trim value at $FFC0.

In applications that require more than 1000 program/erase cycles, use the 4 ms page erase specification to get improved long-term reliability. Any application can use this 4 ms page erase specification. However, in applications where a FLASH location will be erased and reprogrammed less than 1000 times, and speed is important, use the 1 ms page erase specification to get a shorter cycle time.

2.9.4 FLASH Mass Erase Operation

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

1. Set both the ERASE bit and the MASS bit in the FLASH control register.
2. Read the FLASH block protect register.
3. Write any data to any FLASH address[1] within the FLASH memory address range.
4. Wait for a time, t_{NVS} (minimum 10 µs).
5. Set the HVEN bit.
6. Wait for a time, t_{MErase} (minimum 4 ms).
7. Clear the ERASE and MASS bits.

NOTE

Mass erase is disabled whenever any block is protected (FLBPR does not equal $FF).

8. Wait for a time, t_{NVHL} (minimum 100 µs).
9. Clear the HVEN bit.
10. After time, t_{RCV} (typical 1 µs), the memory can be accessed in read mode again.

NOTE

A. *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

B. *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register at $FFFF.*

C. *It is highly recommended that interrupts be disabled during program/erase operations.*

2.9.5 FLASH Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 64 consecutive bytes with address ranges as follows:

- $XX00 to $XX3F
- $XX40 to $XX7F
- $XX80 to $XXBF
- $XXC0 to $XXFF

During the programming cycle, make sure that all addresses being written to fit within one of the ranges specified above. Attempts to program addresses in different row ranges in one programming cycle will fail. Use this step-by-step procedure to program a row of FLASH memory.

NOTE

1. When in monitor mode, with security sequence failed (see 18.3.2 Security), write to the FLASH block protect register instead of any FLASH address.

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

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

The FLASH programming algorithm flowchart is shown in Figure 2-14.

NOTE

A. *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

B. *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. However, care must be taken to ensure that these operations do not access any address within the FLASH array memory space such as the COP control register at $FFFF.*

C. *It is highly recommended that interrupts be disabled during program/erase operations.*

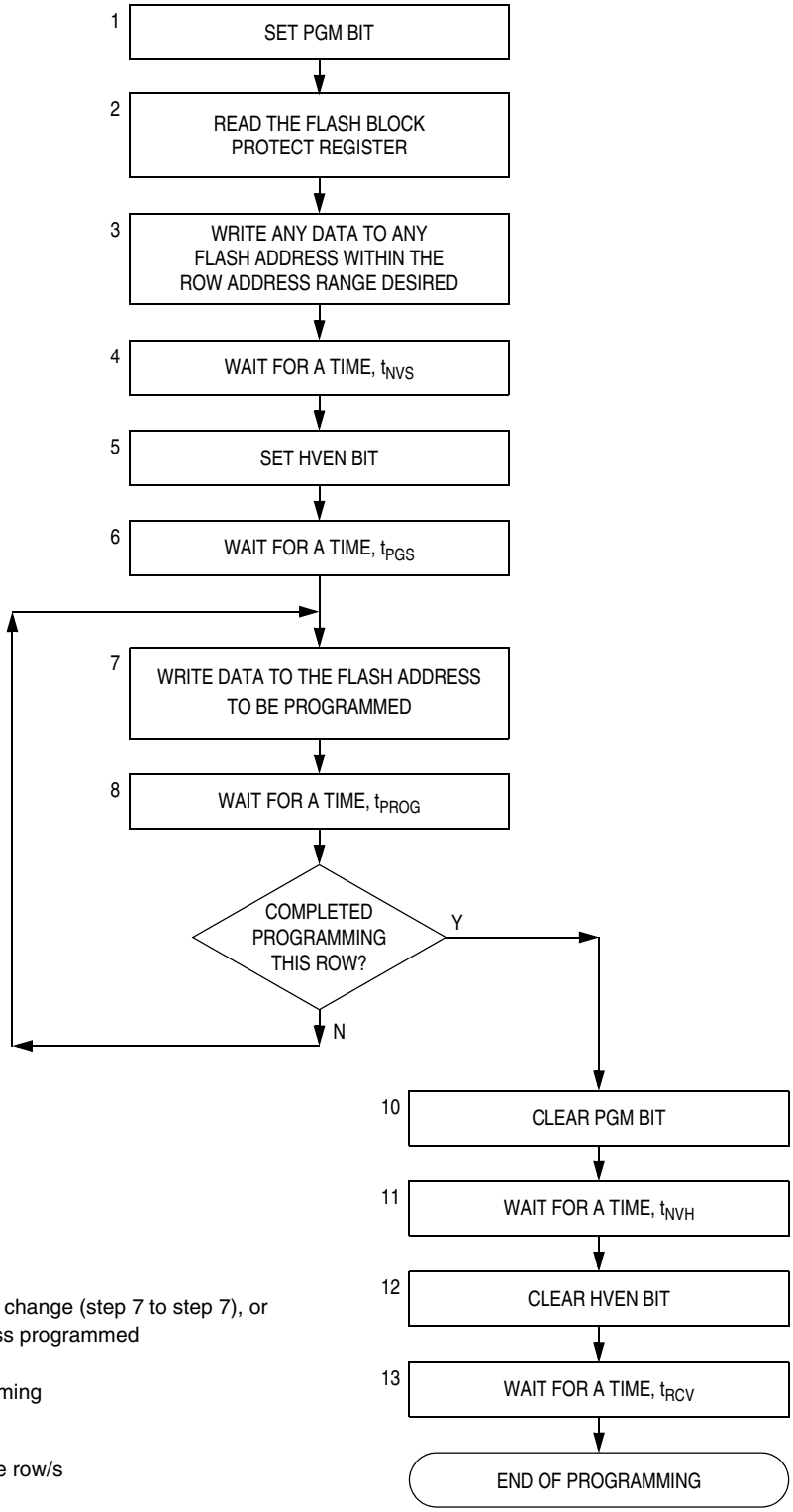
D. *Do not exceed t_{PROG} maximum or t_{HV} maximum. t_{HV} is defined as the cumulative high voltage programming time to the same row before next erase. t_{HV} must satisfy this condition: t_{NVS}+ t_{NVH} + t_{PGS} + (t_{PROG}X 64) ≤ t_{HV} max. Please also see 19.11.3 FLASH Memory Characteristics.*

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

F. *Be cautious when programming the FLASH array to ensure that non-FLASH locations are not used as the address that is written to when selecting either the desired row address range in step 3 of the algorithm or the byte to be programmed in step 7 of the algorithm. This applies particularly to $FFDA–$FFFF (38 bytes)*

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

Algorithm for Programming
a Row (64 Bytes) of FLASH Memory



Notes:

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

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

Figure 2-14. FLASH Programming Algorithm Flowchart

2.9.6 Low-Power Modes

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

2.9.6.1 Wait Mode

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. Wait mode will suspend any FLASH program/erase operations and leave the memory in a standby mode.

2.9.6.2 Stop Mode

Putting the MCU into stop mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. Stop mode will suspend any FLASH program/erase operations and leave the memory in a standby mode.

NOTE

Standby mode is the power saving mode of the FLASH module, in which all internal control signals to the FLASH are inactive and the current consumption of the FLASH is minimum.

Chapter 3
Analog-to-Digital Converter (ADC)

3.1 Introduction

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

For further information regarding analog-to-digital converters on Freescale microcontrollers, please consult the *HC08 ADC Reference Manual*, Freescale document order number ADCRM/AD.

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 15 ADC channels as ADC voltage in (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 share with the ADC channels.

The channel select bits define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general-purpose I/O. Writes to the port register or DDR will not have any affect on the port pin that is selected by the ADC. Read of a port pin 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.

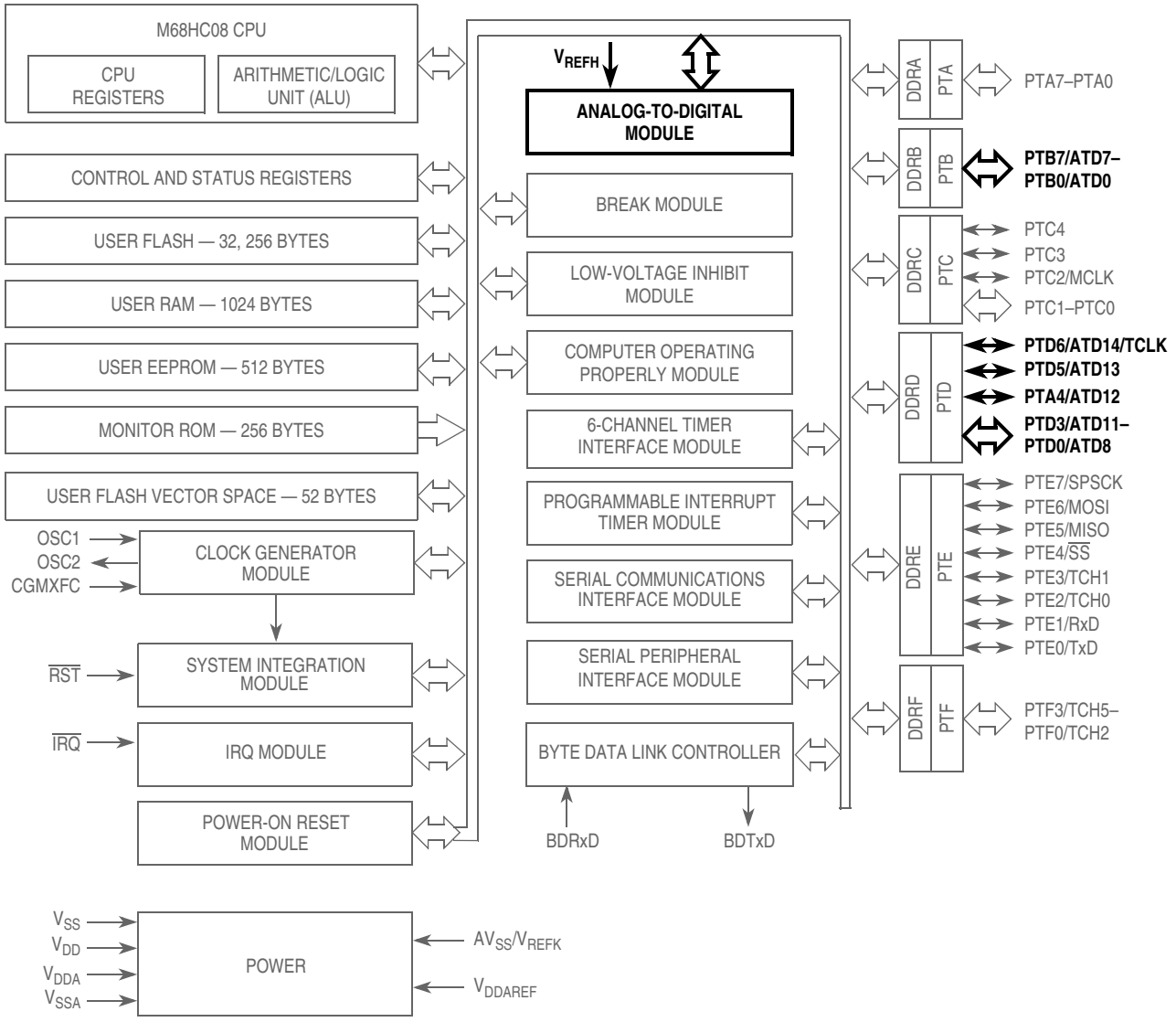


Figure 3-1. Block Diagram Highlighting ADC Block and Pins

3.3.2 Voltage Conversion

When the input voltage to the ADC equals V_{REFH} (see 19.7 Analog-to-Digital Converter (ADC) Characteristics), the ADC converts the signal to $FF (full scale). If the input voltage equals V_{SSA}, the ADC converts it to $00. Input voltages between V_{REFH} and V_{SSA} 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.

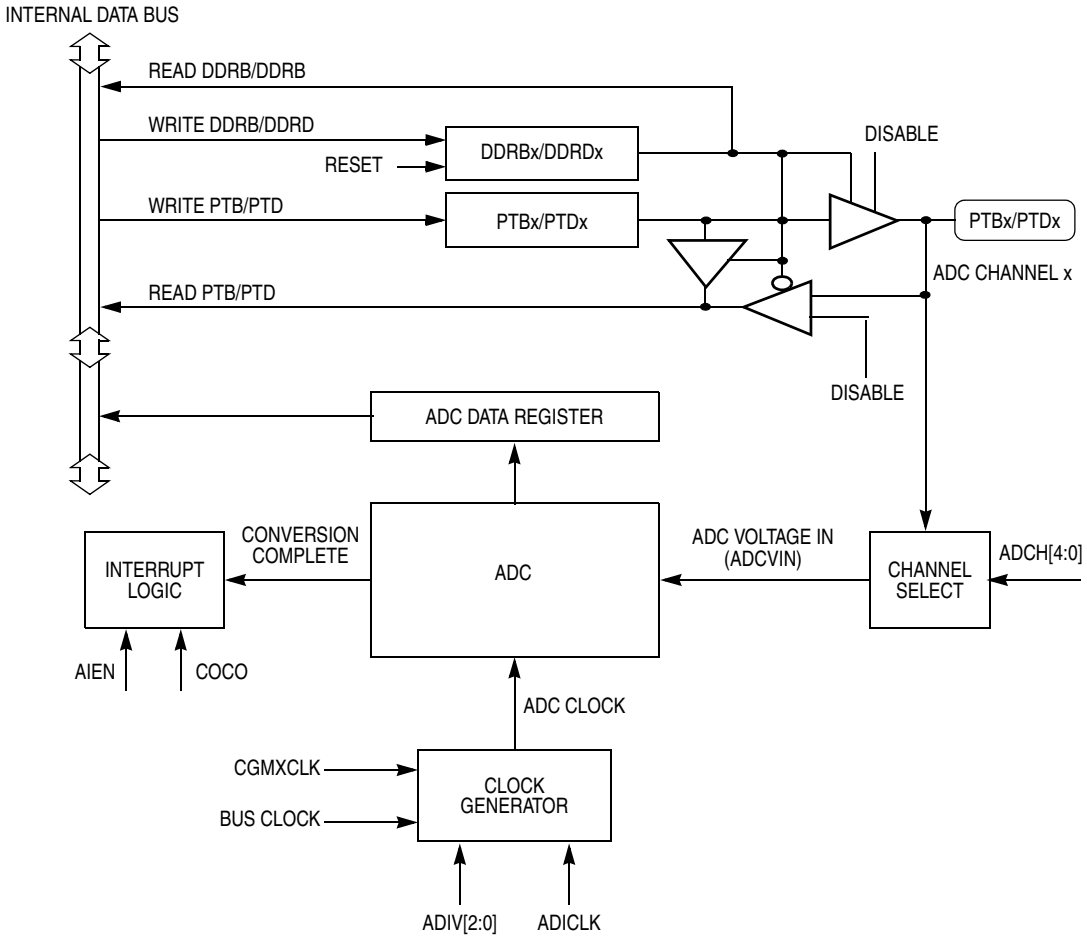


Figure 3-2. ADC Block Diagram

3.3.3 Conversion Time

Conversion starts after a write to the ADSCR (ADC status 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 19.7 Analog-to-Digital Converter (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 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 19.7 Analog-to-Digital Converter (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. If the COCO bit is set, an interrupt is generated. 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 19.7 Analog-to-Digital Converter (ADC) Characteristics for voltages referenced below.

3.6.1 ADC Analog Power Pin (V_{DDAREF})/ADC Voltage Reference Pin (V_{REFH})

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

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

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 Pin (V_{SSA})/ADC Voltage Reference Low Pin (V_{REFL})

The ADC analog portion uses V_{SSA} as its ground pin. Connect the V_{SSA} pin to the same voltage potential as V_{SS}. V_{REFL} is the lower reference supply for the ADC.

3.6.3 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	CH4	CH3	CH2	CH1	CH0
Write:	R							
Reset:	0	0	0	1	1	1	1	1

R = Reserved

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

COCO — Conversions Complete Bit

When the AIEN bit is a 0, the COCO is a read-only bit which is set each time a conversion is completed. This bit is cleared whenever the ADC status and control register is written or whenever the ADC data register is read.

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

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

AIEN — ADC Interrupt Enable Bit

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

1 = ADC interrupt enabled
0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion Bit

When set, the ADC will convert samples continuously and update the ADR register at the end of each conversion. Only one conversion is 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 the following table. 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. See Table 3-1.

The ADC subsystem is turned off when the channel select bits are all set to 1. This feature allows for reduced power consumption for the MCU when the ADC is not 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/ATD8
0	1	0	0	1	PTD1/ATD9/ATD9
0	1	0	1	0	PTD2/ATD10/ATD10
0	1	0	1	1	PTD3/ATD11/ATD11
0	1	1	0	0	PTD4/ATD12/TBCLK/ATD12
0	1	1	0	1	PTD5/ATD13/ATD13
0	1	1	1	0	PTD6/ATD14/TACLK/ATD14
Range 01111 ($0F) to 11010 ($1A)					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	V_{SSA}/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 19.7 Analog-to-Digital Converter (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
Byte Data Link Controller (BDLC)

4.1 Introduction

The byte data link controller (BDLC) provides access to an external serial communication multiplex bus, operating according to the Society of Automotive Engineers (SAE) J1850 protocol.

4.2 Features

Features include:

- SAE J1850 class B data communications network interface compatible and ISO compatible for low speed (≤125 kbps) serial data communications in automotive applications
- 10.4 kbps variable pulse width (VPW) bit format
- Digital noise filter
- Collision detection
- Hardware cyclical redundancy check (CRC) generation and checking
- Two power-saving modes with automatic wakeup on network activity
- Polling and CPU interrupts available
- Block mode receive and transmit supported
- Supports 4X receive mode, 41.6 kbps
- Digital loopback mode
- Analog loopback mode
- In-frame response (IFR) types 0, 1, 2, and 3 supported

4.3 Functional Description

Figure 4-2 shows the organization of the BDLC module. The CPU interface contains the software addressable registers and provides the link between the CPU and the buffers. The buffers provide storage for data received and data to be transmitted onto the J1850 bus. The protocol handler is responsible for the encoding and decoding of data bits and special message symbols during transmission and reception. The MUX interface provides the link between the BDLC digital section and the analog physical interface. The wave shaping, driving, and digitizing of data is performed by the physical interface.

Use of the BDLC module in message networking fully implements the *SAE Standard J1850 Class B Data Communication Network Interface* specification.

NOTE

It is recommended that the reader be familiar with the SAE J1850 document and ISO Serial Communication document prior to proceeding with this section.

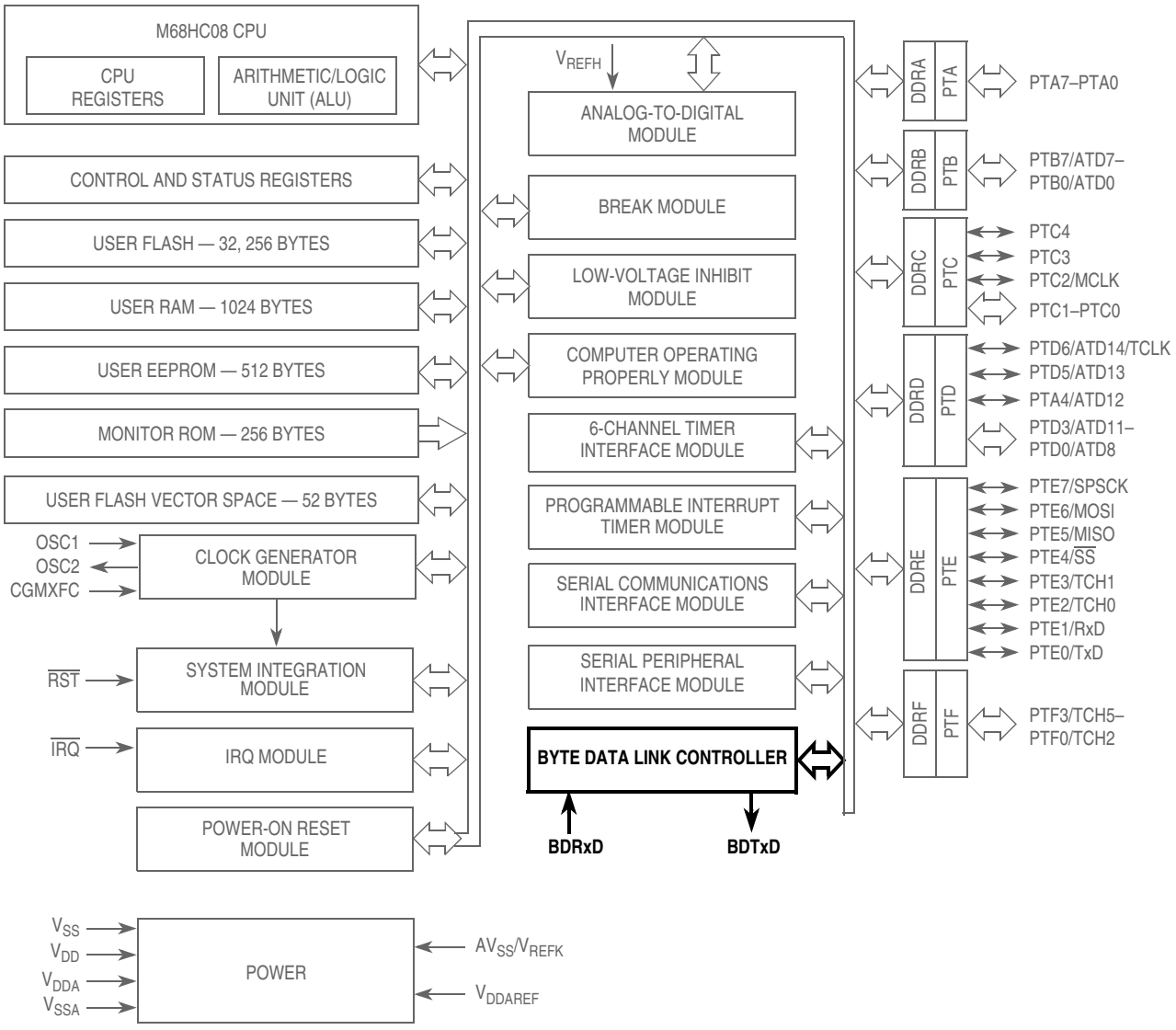


Figure 4-1. Block Diagram Highlighting BDLC Block and Pins

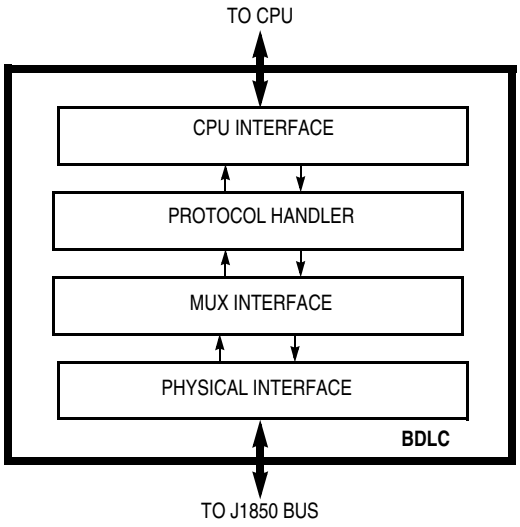


Figure 4-2. BDLC Block Diagram

4.3.1 BDLC Operating Modes

The BDLC has five main modes of operation which interact with the power supplies, pins, and the remainder of the MCU as shown in Figure 4-3.

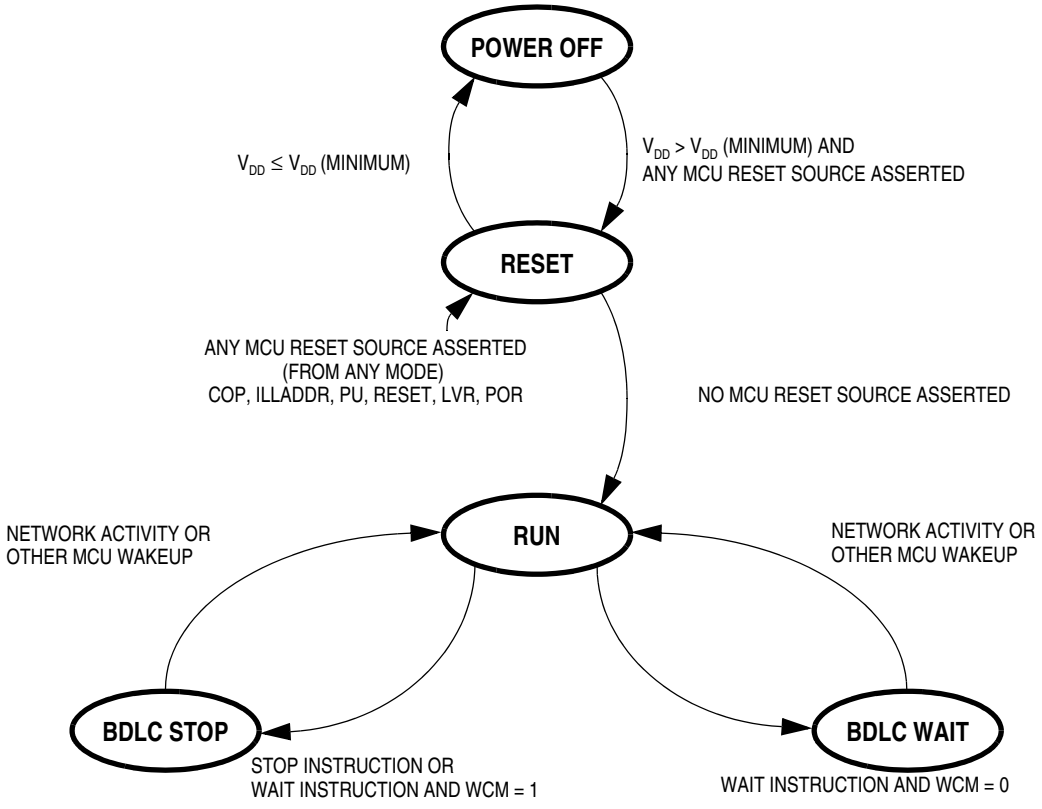


Figure 4-3. BDLC Operating Modes State Diagram

4.3.1.1 Power Off Mode

This mode is entered from reset mode whenever the BDLC supply voltage, V_{DD}, drops below its minimum specified value for the BDLC to guarantee operation. The BDLC will be placed in reset mode by low-voltage reset (LVR) before being powered down. In this mode, the pin input and output specifications are not guaranteed.

4.3.1.2 Reset Mode

This mode is entered from the power off mode whenever the BDLC supply voltage, V_{DD}, rises above its minimum specified value (V_{DD} –10%) and some MCU reset source is asserted. The internal MCU reset must be asserted while powering up the BDLC or an unknown state will be entered and correct operation cannot be guaranteed. Reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (such as LVR, POR, COP watchdog, and reset pin, etc.) is asserted.

In reset mode, the internal BDLC voltage references are operative; V_{DD} is supplied to the internal circuits which are held in their reset state; and the internal BDLC system clock is running. Registers will assume their reset condition. Outputs are held in their programmed reset state. Therefore, inputs and network activity are ignored.

4.3.1.3 Run Mode

This mode is entered from the reset mode after all MCU reset sources are no longer asserted. Run mode is entered from the BDLC wait mode whenever activity is sensed on the J1850 bus.

Run mode is entered from the BDLC stop mode whenever network activity is sensed, although messages will not be received properly until the clocks have stabilized and the CPU is in run mode also.

In this mode, normal network operation takes place. The user should ensure that all BDLC transmissions have ceased before exiting this mode.

4.3.1.4 BDLC Wait Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a WAIT instruction and if the WCM bit in the BCR1 register is cleared previously.

In this mode, the BDLC internal clocks continue to run. The first passive-to-active transition of the bus generates a CPU interrupt request from the BDLC which wakes up the BDLC and the CPU. In addition, if the BDLC receives a valid EOF symbol while operating in wait mode, then the BDLC also will generate a CPU interrupt request which wakes up the BDLC and the CPU. See 4.7.1 Wait Mode.

4.3.1.5 BDLC Stop Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a STOP instruction or if the CPU executes a WAIT instruction and the WCM bit in the BCR1 register is set previously.

In this mode, the BDLC internal clocks are stopped but the physical interface circuitry is placed in a low-power mode and awaits network activity. If network activity is sensed, then a CPU interrupt request will be generated, restarting the BDLC internal clocks. See 4.7.2 Stop Mode.

4.3.1.6 Digital Loopback Mode

When a bus fault has been detected, the digital loopback mode is used to determine if the fault condition is caused by failure in the node's internal circuits or elsewhere in the network, including the node's analog physical interface. In this mode, the transmit digital output pin (BDTxD) and the receive digital input pin (BDRxD) of the digital interface are disconnected from the analog physical interface and tied together to allow the digital portion of the BDLC to transmit and receive its own messages without driving the J1850 bus.

4.3.1.7 Analog Loopback Mode

Analog loopback is used to determine if a bus fault has been caused by a failure in the node's off-chip analog transceiver or elsewhere in the network. The BCLD analog loopback mode does not modify the digital transmit or receive functions of the BDLC. It does, however, ensure that once analog loopback mode is exited, the BDLC will wait for an idle bus condition before participation in network communication resumes. If the off-chip analog transceiver has a loopback mode, it usually causes the input to the output drive stage to be looped back into the receiver, allowing the node to receive messages it has transmitted without driving the J1850 bus. In this mode, the output to the J1850 bus is typically high impedance. This allows the communication path through the analog transceiver to be tested without interfering with network activity. Using the BDLC analog loopback mode in conjunction with the analog transceiver's loopback mode ensures that, once the off-chip analog transceiver has exited loopback mode, the BCLD will not begin communicating before a known condition exists on the J1850 bus.

4.4 BDLC MUX Interface

The MUX interface is responsible for bit encoding/decoding and digital noise filtering between the protocol handler and the physical interface.

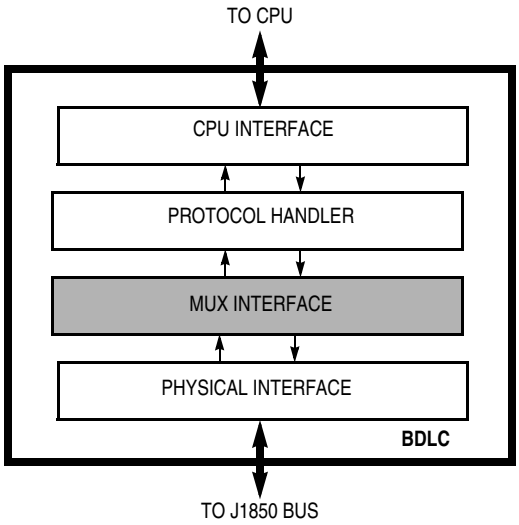


Figure 4-4. BDLC Block Diagram

4.4.1 Rx Digital Filter

The receiver section of the BDLC includes a digital low-pass filter to remove narrow noise pulses from the incoming message. An outline of the digital filter is shown in Figure 4-5.

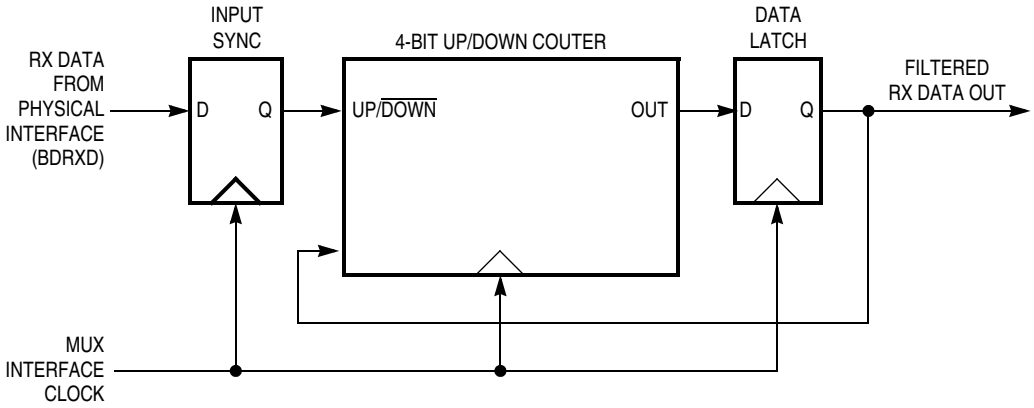


Figure 4-5. BDLC Rx Digital Filter Block Diagram

4.4.1.1 Operation

The clock for the digital filter is provided by the MUX interface clock (see f_{BDLC} parameter in Table 4-3). At each positive edge of the clock signal, the current state of the receiver physical interface (BDRxD) signal is sampled. The BDRxD signal state is used to determine whether the counter should increment or decrement at the next negative edge of the clock signal.

The counter will increment if the input data sample is high but decrement if the input sample is low. Therefore, the counter will thus progress either up toward 15 if, on average, the BDRxD signal remains high or progress down toward 0 if, on average, the BDRxD signal remains low.

When the counter eventually reaches the value 15, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 1 and the data latch is set, causing the filtered Rx data signal to become a logic level 1. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

Alternatively, should the counter eventually reach the value 0, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 0 and the data latch is reset, causing the filtered Rx data signal to become a logic level 0. Furthermore, the counter is prevented from underflowing and can only be incremented from this state.

The data latch will retain its value until the counter next reaches the opposite end point, signifying a definite transition of the signal.

4.4.1.2 Performance

The performance of the digital filter is best described in the time domain rather than the frequency domain.

If the signal on the BDRxD signal transitions, then there will be a delay before that transition appears at the filtered Rx data output signal. This delay will be between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This filter delay must be taken into account when performing message arbitration.

For example, if the frequency of the MUX interface clock (f_{BDLC}) is 1.0486 MHz, then the period (t_{BDLC}) is 954 ns and the maximum filter delay in the absence of noise will be 15.259 µs.

The effect of random noise on the BDRxD signal depends on the characteristics of the noise itself. Narrow noise pulses on the BDRxD signal will be ignored completely if they are shorter than the filter delay. This provides a degree of low-pass filtering.

If noise occurs during a symbol transition, the detection of that transition can be delayed by an amount equal to the length of the noise burst. This is just a reflection of the uncertainty of where the transition is truly occurring within the noise.

Noise pulses that are wider than the filter delay, but narrower than the shortest allowable symbol length, will be detected by the next stage of the BDLC's receiver as an invalid symbol.

Noise pulses that are longer than the shortest allowable symbol length will be detected normally as an invalid symbol or as invalid data when the frame's CRC is checked.

4.4.2 J1850 Frame Format

All messages transmitted on the J1850 bus are structured using the format shown in Figure 4-6. J1850 states that each message has a maximum length of 101 PWM bit times or 12 VPW bytes, excluding SOF, EOD, NB, and EOF, with each byte transmitted MSB first.

All VPW symbol lengths in the following descriptions are typical values at a 10.4 kbps bit rate.

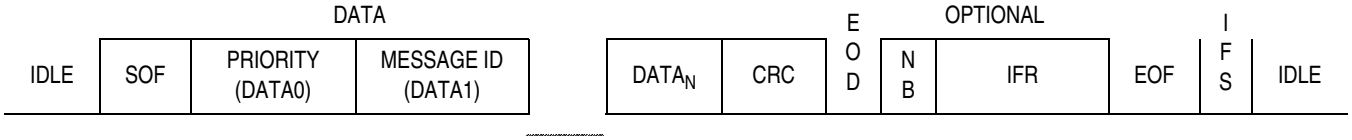


Figure 4-6. J1850 Bus Message Format (VPW)

SOF — Start-of-Frame Symbol

All messages transmitted onto the J1850 bus must begin with a long-active 200-µs period SOF symbol. This indicates the start of a new message transmission. The SOF symbol is not used in the CRC calculation.

Data — In-Message Data Bytes

The data bytes contained in the message include the message priority/type, message ID byte (typically the physical address of the responder), and any actual data being transmitted to the receiving node. The message format used by the BDLC is similar to the 3-byte consolidated header message format outlined by the SAE J1850 document. See *SAE J1850 — Class B Data Communications Network Interface* for more information about 1- and 3-byte headers.

Messages transmitted by the BDLC onto the J1850 bus must contain at least one data byte and, therefore, can be as short as one data byte and one CRC byte. Each data byte in the message is eight bits in length and is transmitted MSB to LSB.

CRC — Cyclical Redundancy Check Byte

This byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. The BDLC calculates the CRC byte and appends it onto any messages transmitted onto the J1850 bus. It also performs CRC detection on any messages it receives from the J1850 bus.

CRC generation uses the divisor polynomial $X^8 + X^4 + X^3 + X^2 + 1$. The remainder polynomial initially is set to all ones. Each byte in the message after the start of frame (SOF) symbol is processed serially through the CRC generation circuitry. The one's complement of the remainder then becomes the 8-bit CRC byte, which is appended to the message after the data bytes in MSB-to-LSB order.

When receiving a message, the BDLC uses the same divisor polynomial. All data bytes, excluding the SOF and end of data symbols (EOD) but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial will equal $X^7 + X^6 + X^2$ = $C4, regardless of the data contained in the message. If the calculated CRC does not equal $C4, the BDLC will recognize this as a CRC error and set the CRC error flag in the BSVR.

EOD — End-of-Data Symbol

The EOD symbol is a long 200-µs passive period on the J1850 bus used to signify to any recipients of a message that the transmission by the originator has completed. No flag is set upon reception of the EOD symbol.

IFR — In-Frame Response Bytes

The IFR section of the J1850 message format is optional. Users desiring further definition of in-frame response should review the *SAE J1850 — Class B Data Communications Network Interface* specification.

EOF — End-of-Frame Symbol

This symbol is a long 280-µs passive period on the J1850 bus and is longer than an end-of-data (EOD) symbol, which signifies the end of a message. Since an EOF symbol is longer than a 200-µs EOD symbol, if no response is transmitted after an EOD symbol, it becomes an EOF, and the message is assumed to be completed. The EOF flag is set upon receiving the EOF symbol.

IFS — Inter-Frame Separation Symbol

The IFS symbol is a 20-µs passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node after the completion of the end-of-frame (EOF) period and, therefore, is seen as a 300-µs passive period.

When the last byte of a message has been transmitted onto the J1850 bus and the EOF symbol time has expired, all nodes then must wait for the IFS symbol time to expire before transmitting a start-of-frame (SOF) symbol, marking the beginning of another message.

However, if the BDLC is waiting for the IFS period to expire before beginning a transmission and a rising edge is detected before the IFS time has expired, it will synchronize internally to that edge. If a write to the BDR register (for instance, to initiate transmission) occurred on or before 104 • t_{BDLC} from the received rising edge, then the BDLC will transmit and arbitrate for the bus. If a CPU write to the BDR register occurred after 104 • t_{BDLC} from the detection of the rising edge, then the BDLC will not transmit, but will wait for the next IFS period to expire before attempting to transmit the byte.

A rising edge may occur during the IFS period because of varying clock tolerances and loading of the J1850 bus, causing different nodes to observe the completion of the IFS period at different times. To allow for individual clock tolerances, receivers must synchronize to any SOF occurring during an IFS period.

NOTE

If two messages are received with a 300 µs *(± 1 µs) interframe separation (IFS) as measured at the RX pin, the start-of-frame (SOF) symbol of the second message will generate an invalid symbol interrupt. This interrupt results in the second message being lost and will therefore be unavailable*

to the application software. Implementations of this BDLC design on silicon have not been exposed to interframe separation rates faster than 320 µs in practical application and have therefore previously not exhibited this behavior. Ensuring that no nodes on the J1850 network transmit messages at 300 µs (± 1 µs) IFS will avoid this missed message frame. In addition, developing application software to robustly handle lost messages will minimize application impact.

BREAK — Break

The BDLC cannot transmit a BREAK symbol.

If the BDLC is transmitting at the time a BREAK is detected, it treats the BREAK as if a transmission error had occurred and halts transmission.

If the BDLC detects a BREAK symbol while receiving a message, it treats the BREAK as a reception error and sets the invalid symbol flag in the BSVR, also ignoring the frame it was receiving. If while receiving a message in 4X mode, the BDLC detects a BREAK symbol, it treats the BREAK as a reception error, sets the invalid symbol flag, and exits 4X mode (for example, the RX4XE bit in BCR2 is cleared automatically). If bus control is required after the BREAK symbol is received and the IFS time has elapsed, the programmer must resend the transmission byte using highest priority.

NOTE

The J1850 protocol BREAK symbol is not related to the HC08 break module. Refer to 18.2 Break Module (BRK).

IDLE — Idle Bus

An idle condition exists on the bus during any passive period after expiration of the IFS period (for instance, ≥ 300 µs). Any node sensing an idle bus condition can begin transmission immediately.

4.4.3 J1850 VPW Symbols

Huntsinger's variable pulse width modulation (VPW) is an encoding technique in which each bit is defined by the time between successive transitions and by the level of the bus between transitions (for instance, active or passive). Active and passive bits are used alternately. This encoding technique is used to reduce the number of bus transitions for a given bit rate.

Each logic 1 or logic 0 contains a single transition and can be at either the active or passive level and one of two lengths, either 64 µs or 128 µs (t_{NOM} at 10.4 kbps baud rate), depending upon the encoding of the previous bit. The start-of-frame (SOF), end-of-data (EOD), end-of-frame (EOF), and inter-frame separation (IFS) symbols always will be encoded at an assigned level and length. See Figure 4-7.

Each message will begin with an SOF symbol an active symbol and, therefore, each data byte (including the CRC byte) will begin with a passive bit, regardless of whether it is a 1 or a 0.

All VPW bit lengths stated in the following descriptions are typical values at a 10.4 kbps bit rate.

Logic 0

A logic 0 is defined as either:

- An active-to-passive transition followed by a passive period 64 µs in length, or
- A passive-to-active transition followed by an active period 128 µs in length

See Figure 4-7(a).

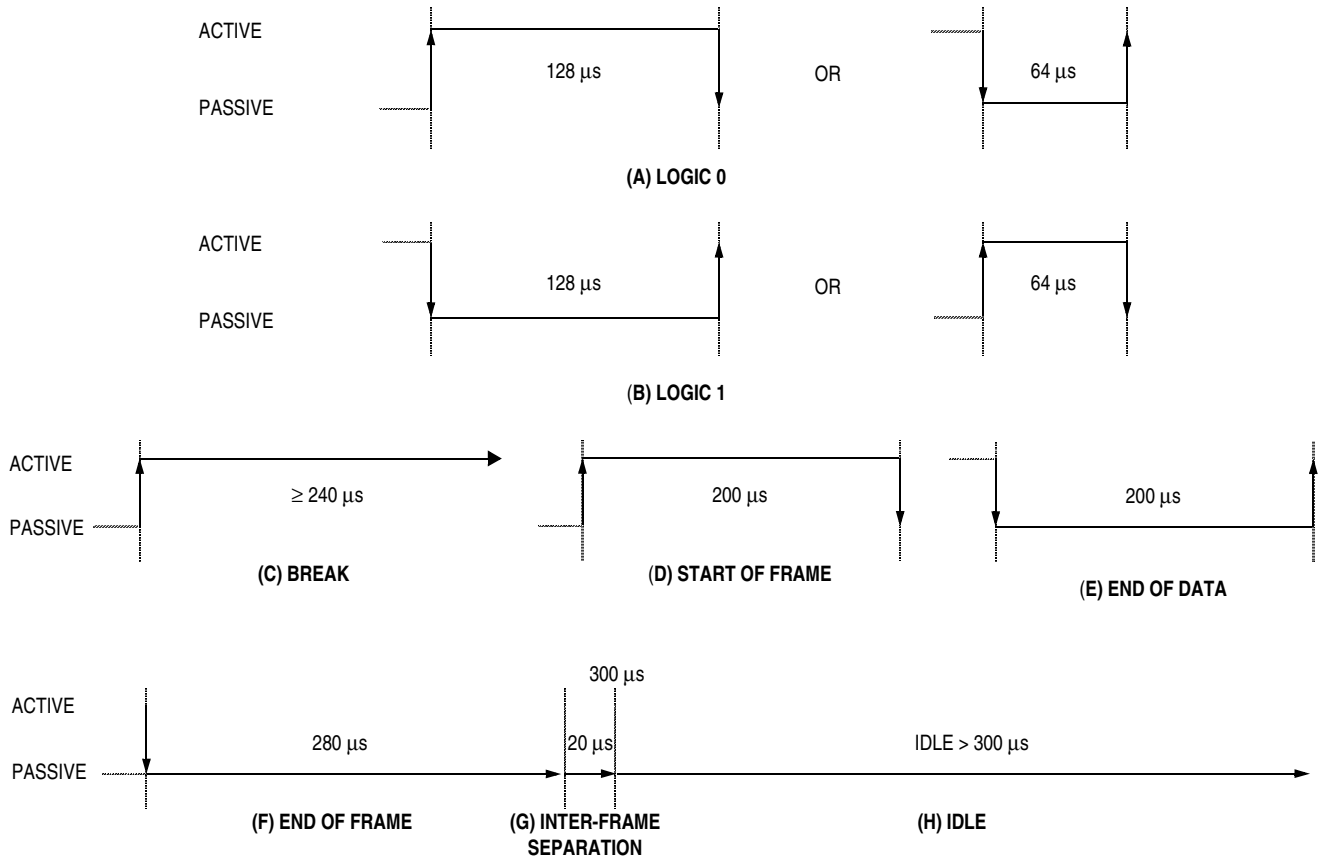


Figure 4-7. J1850 VPW Symbols with Nominal Symbol Times

Logic 1

A logic 1 is defined as either:

a. An active-to-passive transition followed by a passive period 128 µs in length, or

b. A passive-to-active transition followed by an active period 64 µs in length

See Figure 4-7(b).

Normalization Bit (NB)

The NB symbol has the same property as a logic 1 or a logic 0. It is only used in IFR message responses.

Break Signal (BREAK)

The BREAK signal is defined as a passive-to-active transition followed by an active period of at least 240 µs. See Figure 4-7(c).

Start-of-Frame Symbol (SOF)

The SOF symbol is defined as passive-to-active transition followed by an active period 200 µs in length (see Figure 4-7(d)). This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic 1 or a logic 0.

End-of-Data Symbol (EOD)

The EOD symbol is defined as an active-to-passive transition followed by a passive period 200 µs in length (see Figure 4-7(e)).

End-of-Frame Symbol (EOF)

The EOF symbol is defined as an active-to-passive transition followed by a passive period 280 µs in length (see Figure 4-7**(f)**). If no IFR byte is transmitted after an EOD symbol is transmitted, after another 80 µs the EOD becomes an EOF, indicating completion of the message.

Inter-Frame Separation Symbol (IFS)

The IFS symbol is defined as a passive period 300 µs in length. The 20-µs IFS symbol contains no transition, since when used it always appends to an EOF symbol (see Figure 4-7(g)).

Idle

An idle is defined as a passive period greater than 300 µs in length.

4.4.4 J1850 VPW Valid/Invalid Bits and Symbols

The timing tolerances for **receiving** data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies. In many cases the maximum time allowed to define a data bit or symbol is equal to the minimum time allowed to define another data bit or symbol.

Since the minimum resolution of the BDLC for determining what symbol is being received is equal to a single period of the MUX interface clock (t_{BDLC}), an apparent separation in these maximum time/minimum time concurrences equal to one cycle of t_{BDLC} occurs.

This one clock resolution allows the BDLC to differentiate properly between the different bits and symbols. This is done without reducing the valid window for receiving bits and symbols from transmitters onto the J1850 bus which have varying oscillator frequencies.

In Huntsinger's' variable pulse width (VPW) modulation bit encoding, the tolerances for both the passive and active data bits received and the symbols received are defined with no gaps between definitions. For example, the maximum length of a passive logic 0 is equal to the minimum length of a passive logic 1, and the maximum length of an active logic 0 is equal to the minimum length of a valid SOF symbol. See Figure 4-8.

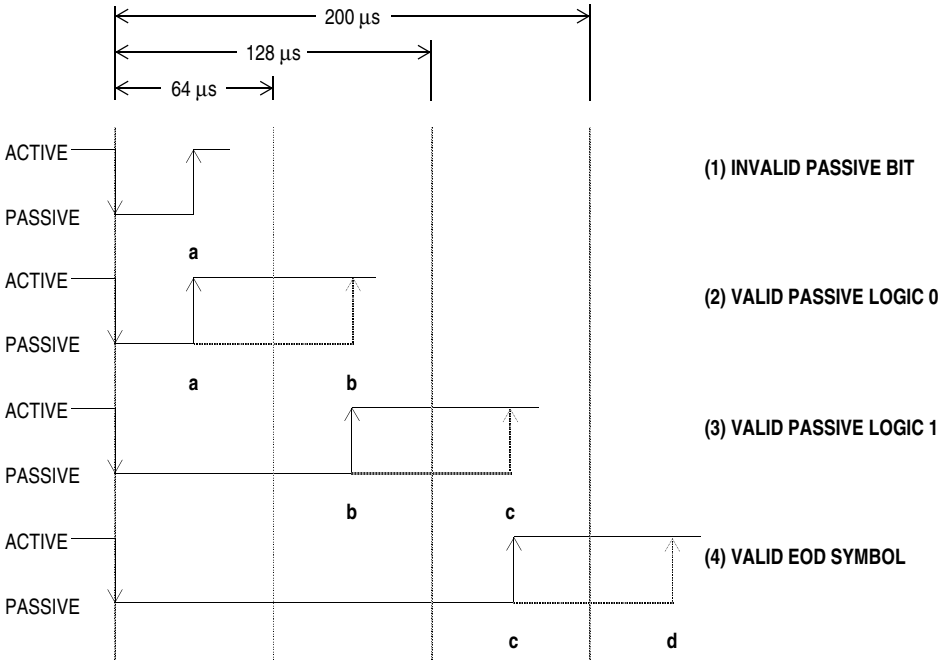


Figure 4-8. J1850 VPW Received Passive Symbol Times

Invalid Passive Bit

See Figure 4-8(1). If the passive-to-active received transition beginning the next data bit or symbol occurs between the active-to-passive transition beginning the current data bit (or symbol) and **a**, the current bit would be invalid.

Valid Passive Logic 0

See Figure 4-8(2). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit would be considered a logic 0.

Valid Passive Logic 1

See Figure 4-8(3). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit would be considered a logic 1.

Valid EOD Symbol

See Figure 4-8(4). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol would be considered a valid end-of-data symbol (EOD).

Valid EOF and IFS Symbol

See Figure 4-9.

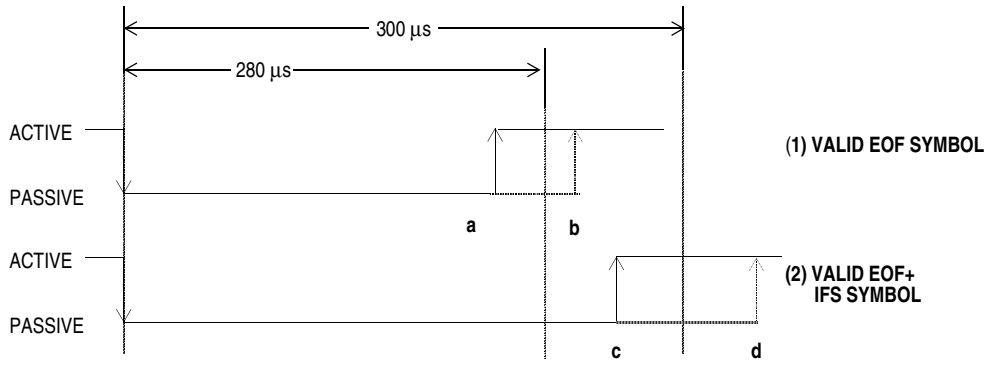


Figure 4-9. J1850 VPW Received Passive EOF and IFS Symbol Times

In Figure 4-9(1), if the passive-to-active received transition beginning the SOF symbol of the next message occurs between **a** and **b**, the current symbol will be considered a valid end-of-frame (EOF) symbol.

See Figure 4-9(2). If the passive-to-active received transition beginning the SOF symbol of the next message occurs between **c** and **d**, the current symbol will be considered a valid EOF symbol followed by a valid inter-frame separation symbol (IFS). All nodes must wait until a valid IFS symbol time has expired before beginning transmission. However, due to variations in clock frequencies and bus loading, some nodes may recognize a valid IFS symbol before others and immediately begin transmitting. Therefore, any time a node waiting to transmit detects a passive-to-active transition once a valid EOF has been detected, it should immediately begin transmission, initiating the arbitration process.

Idle Bus

In Figure 4-9(2), if the passive-to-active received transition beginning the start-of-frame (SOF) symbol of the next message does not occur before **d,** the bus is considered to be idle, and any node wishing to transmit a message may do so immediately.

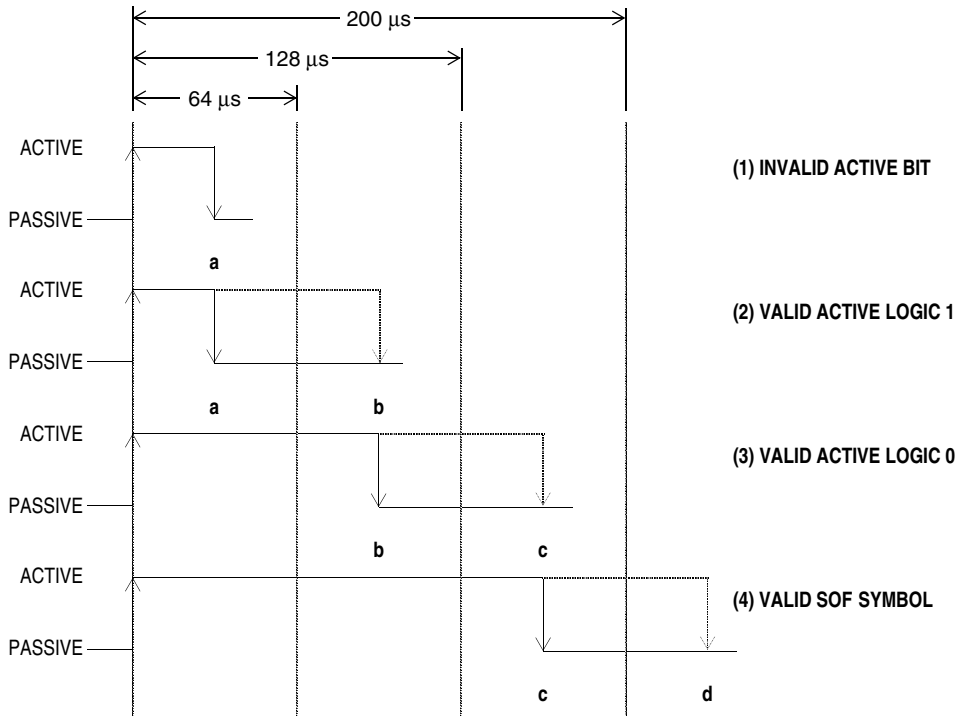


Figure 4-10. J1850 VPW Received Active Symbol Times

Invalid Active Bit

In Figure 4-10(1), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between the passive-to-active transition beginning the current data bit (or symbol) and **a**, the current bit would be invalid.

Valid Active Logic 1

In Figure 4-10(2), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit would be considered a logic 1.

Valid Active Logic 0

In Figure 4-10(3), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit would be considered a logic 0.

Valid SOF Symbol

In Figure 4-10(4), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol would be considered a valid SOF symbol.

Valid BREAK Symbol

In Figure 4-11, if the next active-to-passive received transition does not occur until after **e**, the current symbol will be considered a valid BREAK symbol. A BREAK symbol should be followed by a start-of-frame (SOF) symbol beginning the next message to be transmitted onto the J1850 bus. See 4.4.2 J1850 Frame Format for BDLC response to BREAK symbols.

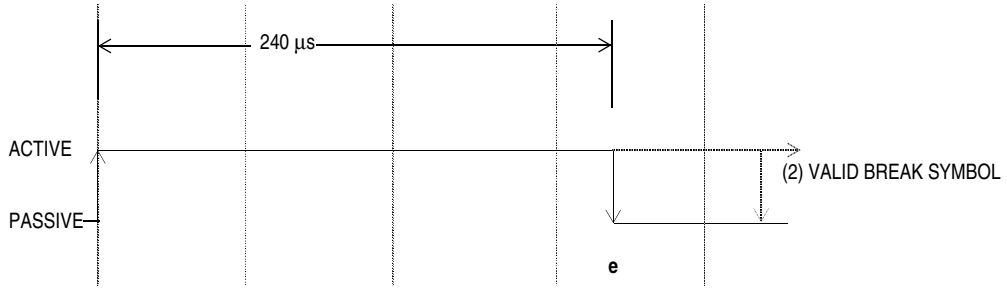


Figure 4-11. J1850 VPW Received BREAK Symbol Times

4.4.5 Message Arbitration

Message arbitration on the J1850 bus is accomplished in a non-destructive manner, allowing the message with the highest priority to be transmitted, while any transmitters which lose arbitration simply stop transmitting and wait for an idle bus to begin transmitting again.

If the BDLC wants to transmit onto the J1850 bus, but detects that another message is in progress, it waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, message arbitration will occur beginning with the first bit after the SOF symbol and will continue with each bit thereafter.

The variable pulse width modulation (VPW) symbols and J1850 bus electrical characteristics are chosen carefully so that a logic 0 (active or passive type) will always dominate over a logic 1 (active or passive type) that is simultaneously transmitted. Hence, logic 0s are said to be dominant and logic 1s are said to be recessive.

Whenever a node detects a dominant bit on BDRxD when it transmitted a recessive bit, the node loses arbitration and immediately stops transmitting. This is known as bitwise arbitration.

Since a logic 0 dominates a logic 1, the message with the lowest value will have the highest priority and will always win arbitration. For instance, a message with priority 000 will win arbitration over a message with priority 011.

This method of arbitration will work no matter how many bits of priority encoding are contained in the message.

During arbitration, or even throughout the transmitting message, when an opposite bit is detected, transmission is stopped immediately unless it occurs on the 8th bit of a byte. In this case, the BDLC automatically will append up to two extra logic 1 bits and then stop transmitting. These two extra bits will be arbitrated normally and thus will not interfere with another message. The second logic 1 bit will not be sent if the first loses arbitration. If the BDLC has lost arbitration to another valid message, then the two extra logic 1s will not corrupt the current message. However, if the BDLC has lost arbitration due to noise on the bus, then the two extra logic 1s will ensure that the current message will be detected and ignored as a noise-corrupted message.

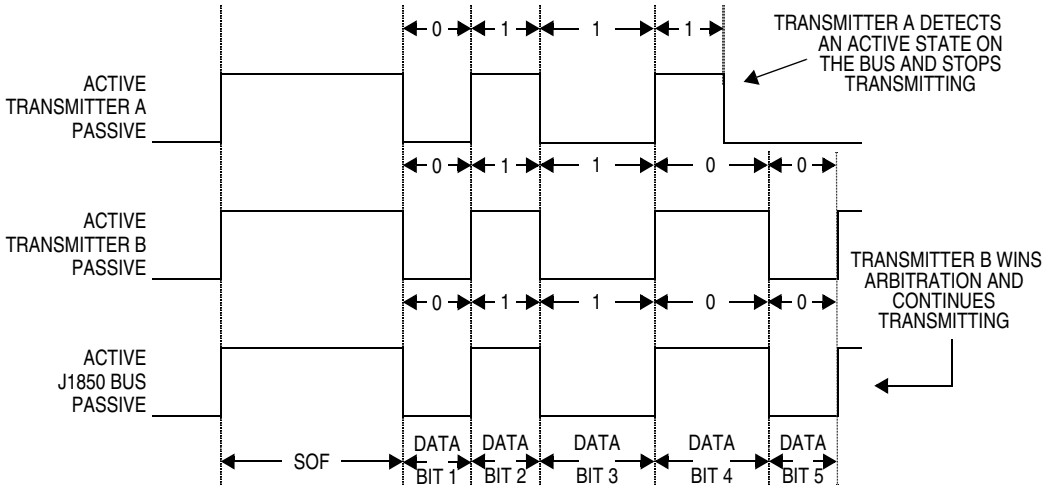


Figure 4-12. J1850 VPW Bitwise Arbitrations

4.5 BDLC Protocol Handler

The protocol handler is responsible for framing, arbitration, CRC generation/checking, and error detection. The protocol handler conforms to *SAE J1850 — Class B Data Communications Network Interface*.

NOTE

Freescale assumes that the reader is familiar with the J1850 specification before this protocol handler description is read.

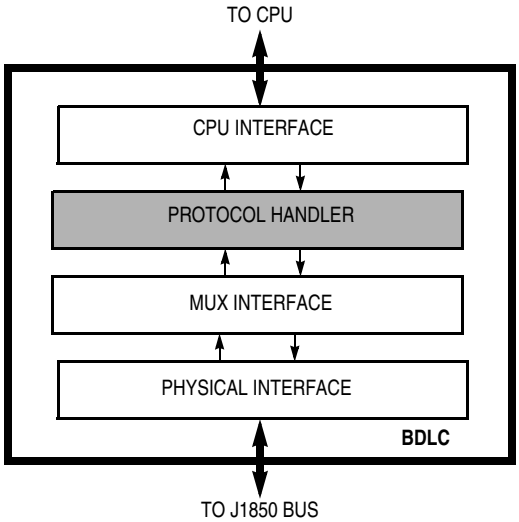


Figure 4-13. BDLC Block Diagram

4.5.1 Protocol Architecture

The protocol handler contains the state machine, Rx shadow register, Tx shadow register, Rx shift register, Tx shift register, and loopback multiplexer as shown in Figure 4-14.

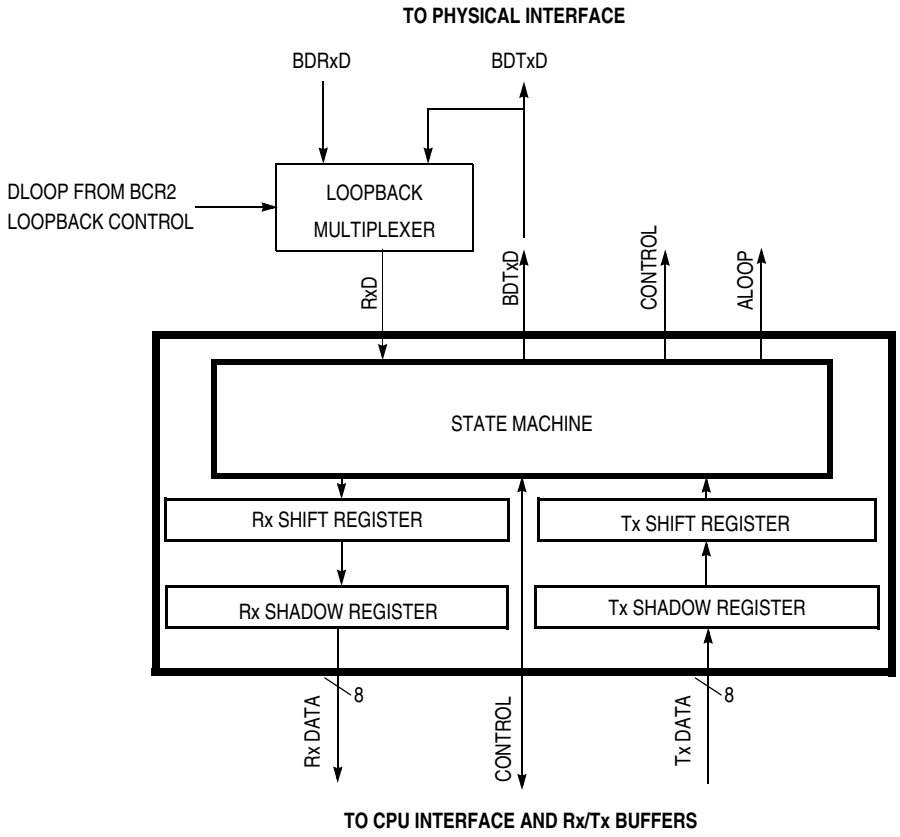


Figure 4-14. BDLC Protocol Handler Outline

4.5.2 Rx and Tx Shift Registers

The Rx shift register gathers received serial data bits from the J1850 bus and makes them available in parallel form to the Rx shadow register. The Tx shift register takes data, in parallel form, from the Tx shadow register and presents it serially to the state machine so that it can be transmitted onto the J1850 bus.

4.5.3 Rx and Tx Shadow Registers

Immediately after the Rx shift register has completed shifting in a byte of data, this data is transferred to the Rx shadow register and RDRF or RXIFR is set (see 4.6.4 BDLC State Vector Register) and an interrupt is generated if the interrupt enable bit (IE) in BCR1 is set. After the transfer takes place, this new data byte in the Rx shadow register is available to the CPU interface, and the Rx shift register is ready to shift in the next byte of data. Data in the Rx shadow register must be retrieved by the CPU before it is overwritten by new data from the Rx shift register.

Once the Tx shift register has completed its shifting operation for the current byte, the data byte in the Tx shadow register is loaded into the Tx shift register. After this transfer takes place, the Tx shadow register is ready to accept new data from the CPU when TDRE flag in BSVR is set.

4.5.4 Digital Loopback Multiplexer

The digital loopback multiplexer connects RxD to either BDTxD or BDRxD, depending on the state of the DLOOP bit in the BCR2 register (see 4.6.3 BDLC Control Register 2).

4.5.5 State Machine

All of the functions associated with performing the protocol are executed or controlled by the state machine. The state machine is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The following subsections describe the BDLC's actions in a variety of situations.

4.5.5.1 4X Mode

The BDLC can exist on the same J1850 bus as modules which use a special 4X (41.6 kbps) mode of J1850 variable pulse width modulation (VPW) operation. The BDLC cannot transmit in 4X mode, but can receive messages in 4X mode, if the RX4X bit is set in BCR2 register. If the RX4X bit is not set in the BCR2 register, any 4X message on the J1850 bus is treated as noise by the BDLC and is ignored.

4.5.5.2 Receiving a Message in Block Mode

Although not a part of the SAE J1850 protocol, the BDLC does allow for a special block mode of operation of the receiver. As far as the BDLC is concerned, a block mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All of the other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Another node wishing to send a block mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

4.5.5.3 Transmitting a Message in Block Mode

A block mode message is transmitted inherently by simply loading the bytes one by one into the BDR register until the message is complete. The programmer should wait until the TDRE flag (see 4.6.4 BDLC State Vector Register) is set prior to writing a new byte of data into the BDR register. The BDLC does not contain any predefined maximum J1850 message length requirement.

4.5.5.4 J1850 Bus Errors

The BDLC detects several types of transmit and receive errors which can occur during the transmission of a message onto the J1850 bus.

Transmission Error

If the message transmitted by the BDLC contains invalid bits or framing symbols on non-byte boundaries, this constitutes a transmission error. When a transmission error is detected, the BDLC immediately will cease transmitting. The error condition ($1C) is reflected in the BSVR register (see Table 4-5). If the interrupt enable bit (IE in BCR1) is set, a CPU interrupt request from the BDLC is generated.

CRC Error

A cyclical redundancy check (CRC) error is detected when the data bytes and CRC byte of a received message are processed and the CRC calculation result is not equal to $C4. The CRC code will detect any single and 2-bit errors, as well as all 8-bit burst errors and almost all other types of errors. The CRC error flag ($18 in BSVR) is set when a CRC error is detected. See 4.6.4 BDLC State Vector Register.

Symbol Error

A symbol error is detected when an abnormal (invalid) symbol is detected in a message being received from the J1850 bus. However, if the BDLC is transmitting when this happens, it will be treated as a loss of arbitration ($14 in BSVR) rather than a transmitter error. The ($1C) symbol invalid or the out-of-range flag is set when a symbol error is detected. Therefore, ($1C) symbol invalid flag is stacked behind the ($14) LOA flag during a transmission error process. See 4.6.4 BDLC State Vector Register.

Framing Error

A framing error is detected if an EOD or EOF symbol is detected on a non-byte boundary from the J1850 bus. A framing error also is detected if the BDLC is transmitting the EOD and instead receives an active symbol. The ($1C) symbol invalid or the out-of-range flag is set when a framing error is detected. See 4.6.4 BDLC State Vector Register.

Bus Fault

If a bus fault occurs, the response of the BDLC will depend upon the type of bus fault.

If the bus is shorted to battery, the BDLC will wait for the bus to fall to a passive state before it will attempt to transmit a message. As long as the short remains, the BDLC will never attempt to transmit a message onto the J1850 bus.

If the bus is shorted to ground, the BDLC will see an idle bus, begin to transmit the message, and then detect a transmission error ($1C in BSVR), since the short to ground would not allow the bus to be driven to the active (dominant) SOF state. The BDLC will abort that transmission and wait for the next CPU command to transmit.

In any case, if the bus fault is temporary, as soon as the fault is cleared, the BDLC will resume normal operation. If the bus fault is permanent, it may result in permanent loss of communication on the J1850 bus. See 4.6.4 BDLC State Vector Register.

BREAK — Break

If a BREAK symbol is received while the BDLC is transmitting or receiving, an invalid symbol ($1C in BSVR) interrupt will be generated. Reading the BSVR register (see 4.6.4 BDLC State Vector Register) will clear this interrupt condition. The BDLC will wait for the bus to idle, then wait for a start-of-frame (SOF) symbol.

The BDLC cannot transmit a BREAK symbol. It can only receive a BREAK symbol from the J1850 bus.

4.5.5.5 Summary

Table 4-1. BDLC J1850 Bus Error Summary

Error Condition	BDLC Function
Transmission error	For invalid bits or framing symbols on non-byte boundaries, invalid symbol interrupt will be generated. BDLC stops transmission.
Cyclical redundancy check (CRC) error	CRC error interrupt will be generated. The BDLC will wait for SOF.
Invalid symbol: BDLC receives invalid bits (noise)	The BDLC will abort transmission immediately. Invalid symbol interrupt will be generated.
Framing error	Invalid symbol interrupt will be generated. The BDLC will wait for start-of-frame (SOF).
Bus short to V_{DD}	The BDLC will not transmit until the bus is idle.
Bus short to GND	Thermal overload will shut down physical interface. Fault condition is reflected in BSVR as an invalid symbol.
BDLC receives BREAK symbol.	The BDLC will wait for the next valid SOF. Invalid symbol interrupt will be generated.

4.6 BDLC CPU Interface

The CPU interface provides the interface between the CPU and the BDLC and consists of five user registers.

- BDLC analog and roundtrip delay register (BARD)
- BDLC control register 1 (BCR1)
- BDLC control register 2 (BCR2)
- BDLC state vector register (BSVR)
- BDLC data register (BDR)

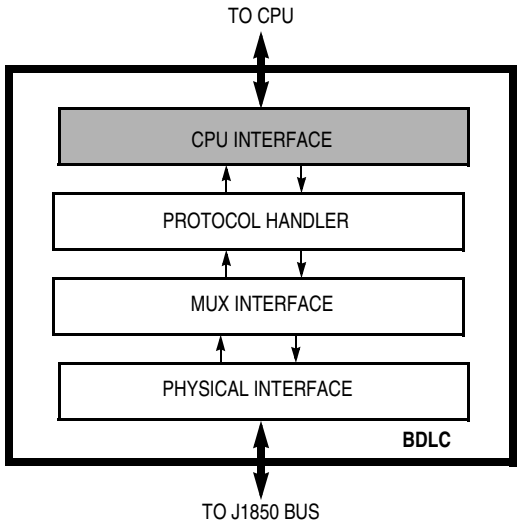


Figure 4-15. BDLC Block Diagram

4.6.1 BDLC Analog and Roundtrip Delay Register

This register programs the BDLC to compensate for various delays of different external transceivers. The default delay value is16 μs. Timing adjustments from 9 μs to 24 μs in steps of 1 μs are available. The BARD register can be written only once after each reset, after which they become read-only bits. The register may be read at any time.

Address: $003B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
Write:			R	R				
Reset:	1	1	0	0	0	1	1	1

R = Reserved

Figure 4-16. BDLC Analog and Roundtrip Delay Register (BARD)

ATE — Analog Transceiver Enable Bit

The analog transceiver enable (ATE) bit is used to select either the on-board or an off-chip analog transceiver.

1 = Select on-board analog transceiver
0 = Select off-chip analog transceiver

NOTE

This device does not contain an on-board transceiver. This bit should be programmed to a logic 0 for proper operation.

RXPOL — Receive Pin Polarity Bit

The receive pin polarity (RXPOL) bit is used to select the polarity of an incoming signal on the receive pin. Some external analog transceivers invert the receive signal from the J1850 bus before feeding it back to the digital receive pin.

1 = Select normal/true polarity; true non-inverted signal from the J1850 bus; for example, the external transceiver does not invert the receive signal

0 = Select inverted polarity, where an external transceiver inverts the receive signal from the J1850 bus

B03–B00 — BARD Offset Bits

Table 4-2 shows the expected transceiver delay with respect to BARD offset values.

Table 4-2. BDLC Transceiver Delay

BARD Offset Bits B0[3:0]	Corresponding Expected Transceiver's Delays (μs)
0000	9
0001	10
0010	11
0011	12
0100	13
0101	14
0110	15
0111	16
1000	17
1001	18
1010	19
1011	20
1100	21
1101	22
1110	23
1111	24

4.6.2 BDLC Control Register 1

This register is used to configure and control the BDLC.

Address: $003C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
Write:					R	R		
Reset:	1	1	1	0	0	0	0	0

R = Reserved

Figure 4-17. BDLC Control Register 1 (BCR1)

IMSG — Ignore Message Bit

This bit is used to disable the receiver until a new start-of-frame (SOF) is detected.

1 = Disable receiver. When set, all BDLC interrupt requests will be masked and the status bits will be held in their reset state. If this bit is set while the BDLC is receiving a message, the rest of the incoming message will be ignored.

0 = Enable receiver. This bit is cleared automatically by the reception of an SOF symbol or a BREAK symbol. It will then generate interrupt requests and will allow changes of the status register to occur. However, these interrupts may still be masked by the interrupt enable (IE) bit.

CLKS — Clock Bit

The nominal BDLC operating frequency (f_{BDLC}) must always be 1.048576 MHz or 1 MHz for J1850 bus communications to take place. The CLKS register bit allows the user to select the frequency (1.048576 MHz or 1 MHz) used to adjust symbol timing automatically.

1 = Binary frequency (1.048576 MHz) selected for f_{BDLC}

0 = Integer frequency (1 MHz) selected for f_{BDLC}

R1 and R0 — Rate Select Bits

These bits determine the amount by which the frequency of the MCU CGMXCLK signal is divided to form the MUX interface clock (f_{BDLC}) which defines the basic timing resolution of the MUX interface. They may be written only once after reset, after which they become read-only bits.

The nominal frequency of f_{BDLC} must always be 1.048576 MHz or 1.0 MHz for J1850 bus communications to take place. Hence, the value programmed into these bits is dependent on the chosen MCU system clock frequency per Table 4-3.

Table 4-3. BDLC Rate Selection

f_{XCLK} Frequency	R1	R0	Division	f_{BDLC}
1.049 MHz	0	0	1	1.049 MHz
2.097 MHz	0	1	2	1.049 MHz
4.194 MHz	1	0	4	1.049 MHz
8.389 MHz	1	1	8	1.049 MHz
1.000 MHz	0	0	1	1.00 MHz
2.000 MHz	0	1	2	1.00 MHz
4.000 MHz	1	0	4	1.00 MHz
8.000 MHz	1	1	8	1.00 MHz

IE— Interrupt Enable Bit

This bit determines whether the BDLC will generate CPU interrupt requests in run mode. It does not affect CPU interrupt requests when exiting the BDLC stop or BDLC wait modes. Interrupt requests will be maintained until all of the interrupt request sources are cleared by performing the specified actions upon the BDLC's registers. Interrupts that were pending at the time that this bit is cleared may be lost.

1 = Enable interrupt requests from BDLC

0 = Disable interrupt requests from BDLC

If the programmer does not wish to use the interrupt capability of the BDLC, the BDLC state vector register (BSVR) can be polled periodically by the programmer to determine BDLC states. See 4.6.4 BDLC State Vector Register for a description of the BSVR.

WCM — Wait Clock Mode Bit

This bit determines the operation of the BDLC during CPU wait mode. See 4.7.2 Stop Mode and 4.7.1 Wait Mode for more details on its use.

1 = Stop BDLC internal clocks during CPU wait mode
0 = Run BDLC internal clocks during CPU wait mode

4.6.3 BDLC Control Register 2

This register controls transmitter operations of the BDLC. It is recommended that BSET and BCLR instructions be used to manipulate data in this register to ensure that the register's content does not change inadvertently.

Address: $003D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
Reset:	1	1	0	0	0	0	0	0

Figure 4-18. BDLC Control Register 2 (BCR2)

ALOOP — Analog Loopback Mode Bit

This bit determines whether the J1850 bus will be driven by the analog physical interface's final drive stage. The programmer can use this bit to reset the BDLC state machine to a known state after the off-chip analog transceiver is placed in loopback mode. When the user clears ALOOP, to indicate that the off-chip analog transceiver is no longer in loopback mode, the BDLC waits for an EOF symbol before attempting to transmit.

1 = Input to the analog physical interface's final drive stage is looped back to the BDLC receiver. The J1850 bus is not driven.
0 = The J1850 bus will be driven by the BDLC. After the bit is cleared, the BDLC requires the bus to be idle for a minimum of end-of-frame symbol time (t_{TRV4}) before message reception or a minimum of inter-frame symbol time (t_{TRV6}) before message transmission. See 19.12.1 BDLC Transmitter VPW Symbol Timings.

DLOOP — Digital Loopback Mode Bit

This bit determines the source to which the digital receive input (BDRxD) is connected and can be used to isolate bus fault conditions (see Figure 4-14). If a fault condition has been detected on the bus, this control bit allows the programmer to connect the digital transmit output to the digital receive input. In this configuration, data sent from the transmit buffer will be reflected back into the receive buffer. If no faults exist in the BDLC, the fault is in the physical interface block or elsewhere on the J1850 bus.

1 = When set, BDRxD is connected to BDTxD. The BDLC is now in digital loopback mode.
0 = When cleared, BDTxD is not connected to BDRxD. The BDLC is taken out of digital loopback mode and can now drive the J1850 bus normally.

RX4XE — Receive 4X Enable Bit

This bit determines if the BDLC operates at normal transmit and receive speed (10.4 kbps) or receive only at 41.6 kbps. This feature is useful for fast download of data into a J1850 node for diagnostic or factory programming of the node.

1 = When set, the BDLC is put in 4X receive-only operation.
0 = When cleared, the BDLC transmits and receives at 10.4 kbps.

NBFS — Normalization Bit Format Select Bit

This bit controls the format of the normalization bit (NB) (see Figure 4-19). SAE J1850 strongly encourages using an active long (logic 0) for in-frame responses containing cyclical redundancy check (CRC) and an active short (logic 1) for in-frame responses without CRC.

1 = NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) does not end with a CRC byte.

0 = NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) does not end with a CRC byte.

TEOD — Transmit End of Data Bit

This bit is set by the programmer to indicate the end of a message is being sent by the BDLC. It will append an 8-bit CRC after completing transmission of the current byte. This bit also is used to end an in-frame response (IFR). If the transmit shadow register is full when TEOD is set, the CRC byte will be transmitted after the current byte in the Tx shift register and the byte in the Tx shadow register have been transmitted. (See 4.5.3 Rx and Tx Shadow Registers for a description of the transmit shadow register.) Once TEOD is set, the transmit data register empty flag (TDRE) in the BDLC state vector register (BSVR) is cleared to allow lower priority interrupts to occur. See 4.6.4 BDLC State Vector Register.

1 = Transmit end-of-data (EOD) symbol

0 = The TEOD bit will be cleared automatically at the rising edge of the first CRC bit that is sent or if an error is detected. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC receives back a valid EOD symbol or an error condition occurs.

TSIFR, TMIFR1, and TMIFR0 — Transmit In-Frame Response Control Bits

These three bits control the type of in-frame response being sent. The programmer should not set more than one of these control bits to a 1 at any given time. However, if more than one of these three control bits are set to 1, the priority encoding logic will force these register bits to a known value as shown in Table 4-4. For example, if 011 is written to TSIFR, TMIFR1, and TMIFR0, then internally they will be encoded as 010. However, when these bits are read back, they will read 011.

Table 4-4. BDLC Transmit In-Frame Response Control Bit Priority Encoding

Write/Read TSIFR	Write/Read TMIFR1	Write/Read TMIFR0	Actual TSIFR	Actual TMIFR1	Actual TMIFR0
0	0	0	0	0	0
1	X	X	1	0	0
0	1	X	0	1	0
0	0	1	0	0	1

The BDLC supports the in-frame response (IFR) feature of J1850 by setting these bits correctly. The four types of J1850 IFR are shown below. The purpose of the in-frame response modes is to allow multiple nodes to acknowledge receipt of the data by responding with their personal ID or physical address in a concatenated manner after they have seen the EOD symbol. If transmission arbitration is lost by a node while sending its response, it continues to transmit its ID/address until observing its unique byte in the response stream. For VPW modulation, because the first bit of the IFR is always

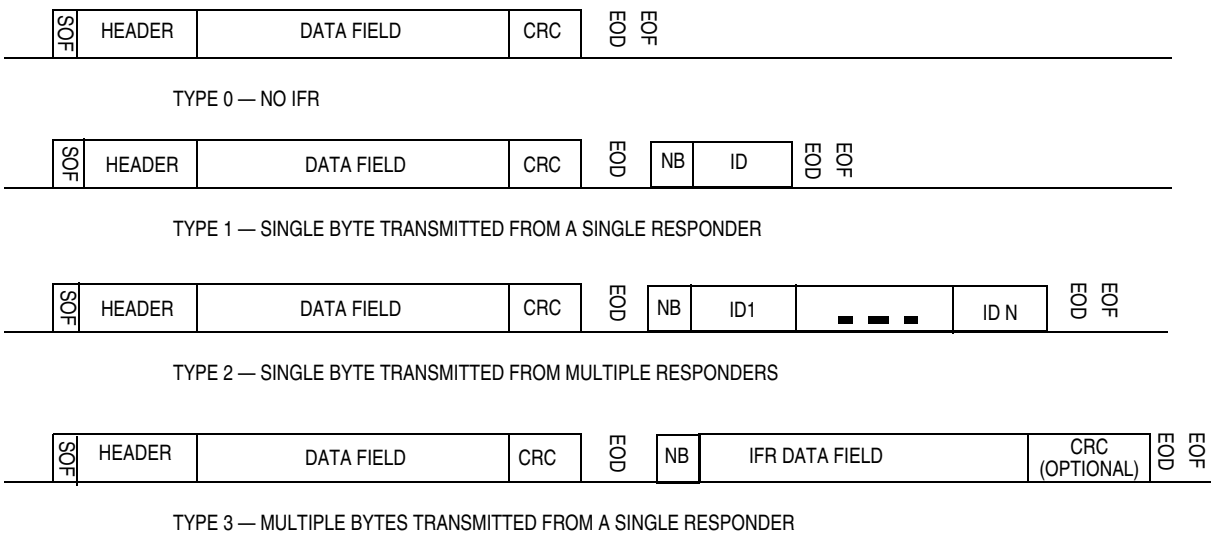
passive, a normalization bit (active) must be generated by the responder and sent prior to its ID/address byte. When there are multiple responders on the J1850 bus, only one normalization bit is sent which assists all other transmitting nodes to sync up their response.

TSIFR — Transmit Single Byte IFR with No CRC (Type 1 or 2) Bit

The TSIFR bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR, $003F) as a single byte IFR with no CRC. Typically, the byte transmitted is a unique identifier or address of the transmitting (responding) node. See Figure 4-19.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC will attempt to transmit the appropriate normalization bit followed by the byte in the BDR.

0 = The TSIFR bit will be cleared automatically, once the BDLC has successfully transmitted the byte in the BDR onto the bus, or TEOD is set, or an error is detected on the bus.



NB = Normalization Bit
ID = Identifier (usually the physical address of the responder(s))
HEADER = Specifies one of three frame lengths

Figure 4-19. Types of In-Frame Response (IFR)

If the programmer attempts to set the TSIFR bit immediately after the EOD symbol has been received from the bus, the TSIFR bit will remain in the reset state and no attempt will be made to transmit the IFR byte.

If a loss of arbitration occurs when the BDLC attempts to transmit and after the IFR byte winning arbitration completes transmission, the BDLC will again attempt to transmit the BDR (with no normalization bit). The BDLC will continue transmission attempts until an error is detected on the bus, or TEOD is set, or the BDLC transmission is successful.

If loss or arbitration occurs in the last two bits of the IFR byte, two additional 1 bits **will not** be sent out because the BDLC will attempt to retransmit the byte in the transmit shift register after the IRF byte winning arbitration completes transmission.

TMIFR1 — Transmit Multiple Byte IFR with CRC (Type 3) Bit

The TMIFR1 bit requests the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR with CRC or as a single byte IFR with CRC. Response IFR bytes are still subject to J1850 message length maximums. See 4.4.2 J1850 Frame Format and Figure 4-19.

If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR register. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.

The TMIFR1 bit will be cleared automatically – once the BDLC has successfully transmitted the CRC byte and EOD symbol – by the detection of an error on the multiplex bus or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

If the TMIFR1 bit is set, the BDLC will attempt to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see 4.6.4 BDLC State Vector Register) will occur similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BDLC control register 2 (BCR2). This will instruct the BDLC to transmit a CRC byte once the byte in the BDR is transmitted and then transmit an EOD symbol, indicating the end of the IFR portion of the message frame.

However, if the programmer wishes to transmit a single byte followed by a CRC byte, the programmer should load the byte into the BDR before the EOD symbol has been received, and then set the TMIFR1 bit. Once the TDRE interrupt occurs, the programmer should then set the TEOD bit in the BCR2. This will result in the byte in the BDR being the only byte transmitted before the IFR CRC byte, and no TDRE interrupt will be generated.

If the programmer attempts to set the TMIFR1 bit immediately after the EOD symbol has been received from the bus, the TMIFR1 bit will remain in the reset state, and no attempt will be made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting any byte of a multiple byte IFR, the BDLC will go to the loss of arbitration state, set the appropriate flag, and cease transmission.

If the BDLC loses arbitration during the IFR, the TMIFR1 bit will be cleared and no attempt will be made to retransmit the byte in the BDR. If loss of arbitration occurs in the last two bits of the IFR byte, two additional 1 bits will be sent out.

NOTE

The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise from going onto the J1850 bus from a corrupted message.

TMIFR0 — Transmit Multiple Byte IFR without CRC (Type 3) Bit

The TMIFR0 bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR without CRC. Response IFR bytes are still subject to J1850 message length maximums. See 4.4.2 J1850 Frame Format and Figure 4-19.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR register. After TEOD has been set, the last IFR byte to be transmitted will be the last byte which was written into the BDR register.

0 = The TMIFR0 bit will be cleared automatically; once the BDLC has successfully transmitted the EOD symbol; by the detection of an error on the multiplex bus; or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

If the TMIFR0 bit is set, the BDLC will attempt to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see 4.6.4 BDLC State Vector Register) will occur similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BCR2. This will instruct the BDLC to transmit an EOD symbol once the byte in the BDR is transmitted, indicating the end of the IFR portion of the message frame. The BDLC will not append a CRC when the TMIFR0 is set.

If the programmer attempts to set the TMIFR0 bit after the EOD symbol has been received from the bus, the TMIFR0 bit will remain in the reset state, and no attempt will be made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting, the TMIFR0 bit will be cleared and no attempt will be made to retransmit the byte in the BDR. If loss of arbitration occurs in the last two bits of the IFR byte, two additional 1 bits (active short bits) will be sent out.

NOTE

The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise from going onto the J1850 bus from a corrupted message.

4.6.4 BDLC State Vector Register

This register is provided to substantially decrease the CPU overhead associated with servicing interrupts while under operation of a multiplex protocol. It provides an index offset that is directly related to the BDLC's current state, which can be used with a user-supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

Address: $003E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	I3	I2	I1	I0	0	0
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 4-20. BDLC State Vector Register (BSVR)

I0, I1, I2, and I3 — Interrupt Source Bits

These bits indicate the source of the interrupt request that currently is pending. The encoding of these bits are listed in Table 4-5.

Bits I0, I1, I2, and I3 are cleared by a read of the BSVR except when the BDLC data register needs servicing (RDRF, RXIFR, or TDRE conditions). RXIFR and RDRF can be cleared only by a read of the BSVR followed by a read of the BDLC data register (BDR). TDRE can either be cleared by a read of the BSVR followed by a write to the BDLC BDR or by setting the TEOD bit in BCR2.

Table 4-5. BDLC Interrupt Sources

BSVR	I3	I2	I1	I0	Interrupt Source	Priority
$00	0	0	0	0	No interrupts pending	0 (lowest)
$04	0	0	0	1	Received EOF	1
$08	0	0	1	0	Received IFR byte (RXIFR)	2
$0C	0	0	1	1	BDLC Rx data register full (RDRF)	3
$10	0	1	0	0	BDLC Tx data register empty (TDRE)	4
$14	0	1	0	1	Loss of arbitration	5
$18	0	1	1	0	Cyclical redundancy check (CRC) error	6
$1C	0	1	1	1	Symbol invalid or out of range	7
$20	1	0	0	0	wakeup	8 (highest)

Upon receiving a BDLC interrupt, the user can read the value within the BSVR, transferring it to the CPU's index register. The value can then be used to index into a jump table, with entries four bytes apart, to quickly enter the appropriate service routine. For example:

```
Service     LDX    BSVR          Fetch State Vector Number
            JMP    JMPTAB,X      Enter service routine,
*                                (must end in RTI)
*
JMPTAB      JMP    SERVE0        Service condition #0
            NOP
            JMP    SERVE1        Service condition #1
            NOP
            JMP    SERVE2        Service condition #2
            NOP
*
            JMP    SERVE8        Service condition #8
            END
```

NOTE

The NOPs are used only to align the JMPs onto 4-byte boundaries so that the value in the BSVR can be used intact. Each of the service routines must end with an RTI instruction to guarantee correct continued operation of the device. Note also that the first entry can be omitted since it corresponds to no interrupt occurring.

The service routines should clear all of the sources that are causing the pending interrupts. Note that the clearing of a high priority interrupt may still leave a lower priority interrupt pending, in which case bits I0, I1, and I2 of the BSVR will then reflect the source of the remaining interrupt request.

If fewer states are used or if a different software approach is taken, the jump table can be made smaller or omitted altogether.

4.6.5 BDLC Data Register

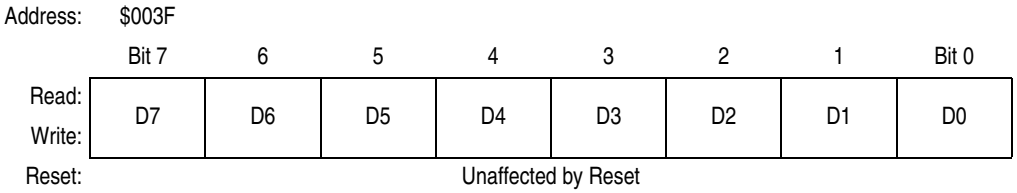


Figure 4-21. BDLC Data Register (BDR)

This register is used to pass the data to be transmitted to the J1850 bus from the CPU to the BDLC. It is also used to pass data received from the J1850 bus to the CPU. Each data byte (after the first one) should be written only after a Tx data register empty (TDRE) state is indicated in the BSVR.

Data read from this register will be the last data byte received from the J1850 bus. This received data should only be read after an Rx data register full (RDRF) interrupt has occurred. See 4.6.4 BDLC State Vector Register.

The BDR is double buffered via a transmit shadow register and a receive shadow register. After the byte in the transmit shift register has been transmitted, the byte currently stored in the transmit shadow register is loaded into the transmit shift register. Once the transmit shift register has shifted the first bit out, the TDRE flag is set, and the shadow register is ready to accept the next data byte. The receive shadow register works similarly. Once a complete byte has been received, the receive shift register stores the newly received byte into the receive shadow register. The RDRF flag is set to indicate that a new byte of data has been received. The programmer has one BDLC byte reception time to read the shadow register and clear the RDRF flag before the shadow register is overwritten by the newly received byte.

To abort an in-progress transmission, the programmer should stop loading data into the BDR. This will cause a transmitter underrun error and the BDLC automatically will disable the transmitter on the next non-byte boundary. This means that the earliest a transmission can be halted is after at least one byte plus two extra logic 1s have been transmitted. The receiver will pick this up as an error and relay it in the state vector register as an invalid symbol error.

NOTE

The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise from going onto the J1850 bus from a corrupted message.

4.7 Low-Power Modes

The following information concerns wait mode and stop mode.

4.7.1 Wait Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a WAIT instruction and the WCM bit in BDLC control register 1 (BCR1) is previously clear. In BDLC wait mode, the BDLC cannot drive any data.

A subsequent successfully received message, including one that is in progress at the time that this mode is entered, will cause the BDLC to wake up and generate a CPU interrupt request if the interrupt enable (IE) bit in the BDLC control register 1 (BCR1) is previously set. (See 4.6.2 BDLC Control Register 1 for a

better understanding of IE.) This results in less of a power saving, but the BDLC is guaranteed to receive correctly the message which woke it up, since the BDLC internal operating clocks are kept running.

NOTE

Ensuring that all transmissions are complete or aborted before putting the BDLC into wait mode is important.

4.7.2 Stop Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a STOP instruction or if the CPU executes a WAIT instruction and the WCM bit in the BDLC control register 1 (BCR1) is previously set. This is the lowest power mode that the BDLC can enter.

A subsequent passive-to-active transition on the J1850 bus will cause the BDLC to wake up and generate a non-maskable CPU interrupt request. When a STOP instruction is used to put the BDLC in stop mode, the BDLC is not guaranteed to correctly receive the message which woke it up, since it may take some time for the BDLC internal operating clocks to restart and stabilize. If a WAIT instruction is used to put the BDLC in stop mode, the BDLC is guaranteed to correctly receive the byte which woke it up, if and only if an end-of-frame (EOF) has been detected prior to issuing the WAIT instruction by the CPU. Otherwise, the BDLC will not correctly receive the byte that woke it up.

If this mode is entered while the BDLC is receiving a message, the first subsequent received edge will cause the BDLC to wake up immediately, generate a CPU interrupt request, and wait for the BDLC internal operating clocks to restart and stabilize before normal communications can resume. Therefore, the BDLC is not guaranteed to receive that message correctly.

NOTE

It is important to ensure all transmissions are complete or aborted prior to putting the BDLC into stop mode.

Chapter 5
Clock Generator Module (CGM)

5.1 Introduction

The CGM generates the crystal clock signal, CGMXCLK, which operates at the frequency of the crystal. The CGM also generates the base clock signal, CGMOUT, from which the system 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.

5.2 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

5.3 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 5-2 shows the structure of the CGM.

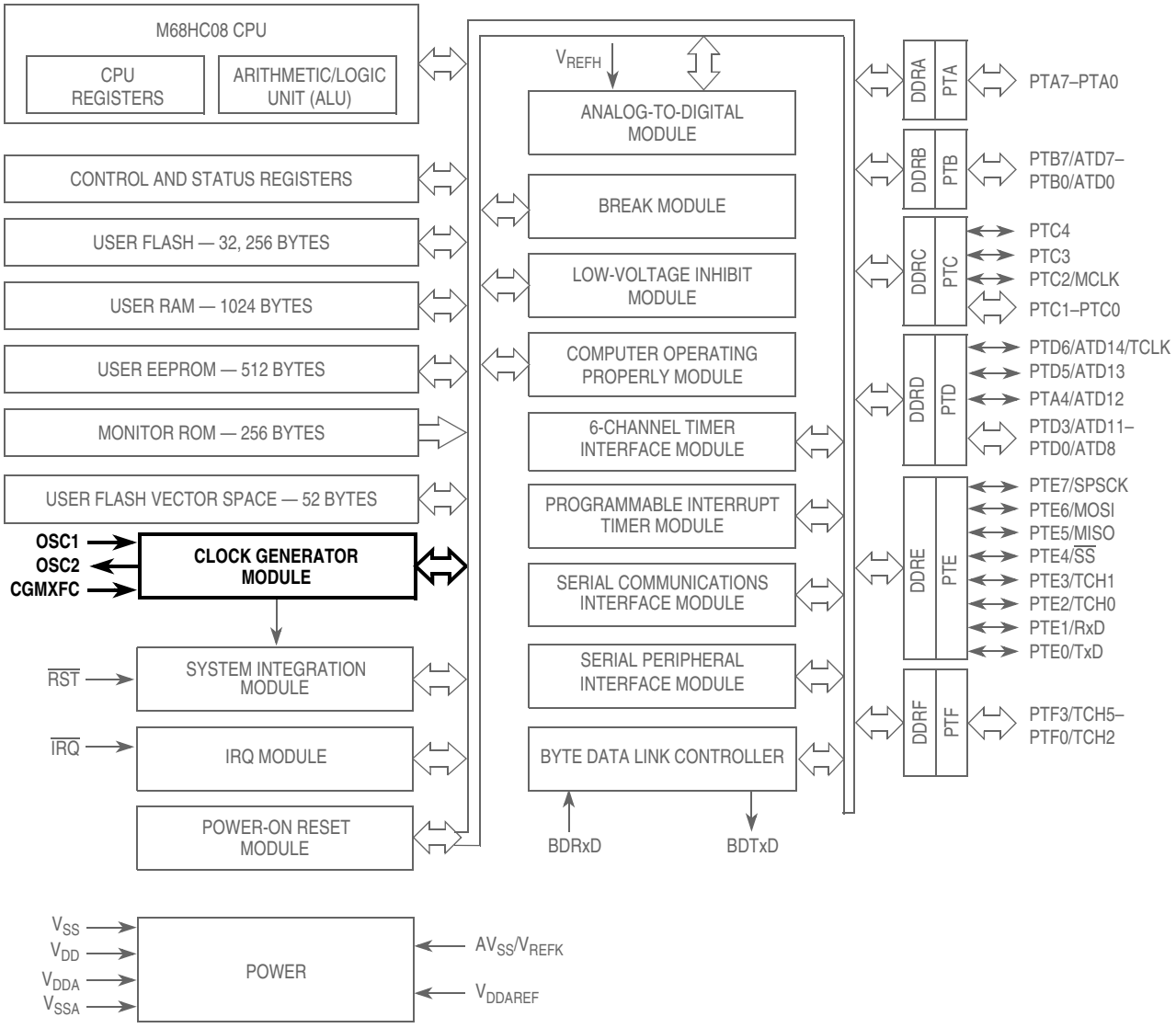


Figure 5-1. Block Diagram Highlighting CGM Block and Pins

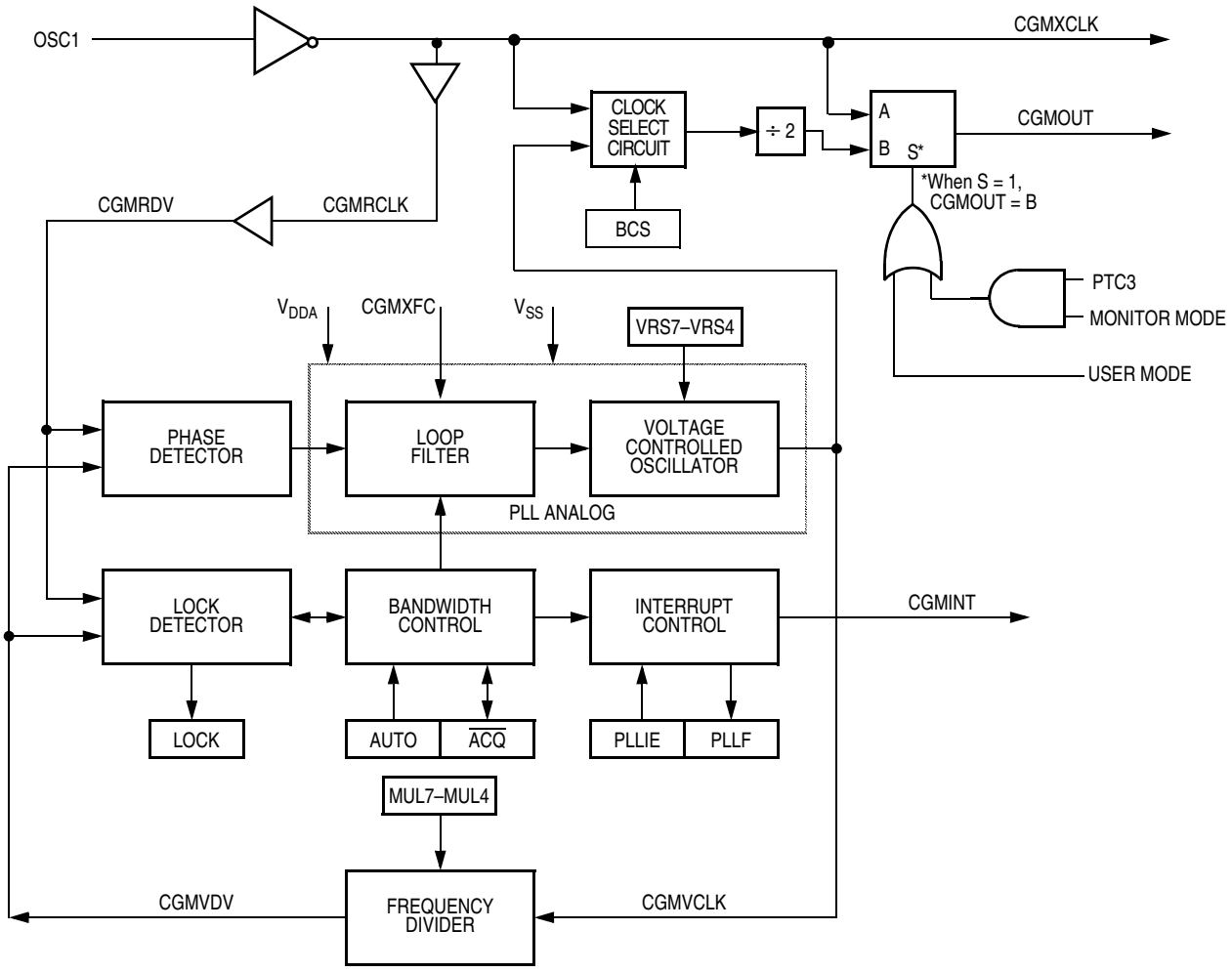


Figure 5-2. CGM Block Diagram

5.3.1 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.

5.3.2 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.

5.3.2.1 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 $f_{CGMVDV} = f_{CGMVCLK}/N$. See 5.3.2.4 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 5.3.2.2 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.

5.3.2.2 Acquisition and Tracking Modes

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

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL 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 5.5.2 PLL Bandwidth Control Register for more information.

- 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 5.3.3 Base Clock Selector Circuit for more information. The PLL is automatically in tracking mode when it's not in acquisition mode or when the $\overline{ACQ}$ bit is set.

5.3.2.3 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 5.5.2 PLL Bandwidth Control Register for more information. 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 5.3.3 Base Clock Selector Circuit for more information. 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 5.6 Interrupts for more information.

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

- The $\overline{ACQ}$ bit (see 5.5.2 PLL Bandwidth Control Register) is a read-only indicator of the mode of the filter. See 5.3.2.2 Acquisition and Tracking Modes for more information.
- 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 Chapter 19 Electrical Specifications for more information.
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance, Δ_{Lock}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{UNL}. See Chapter 19 Electrical Specifications for more information.
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. See 5.5.1 PLL Control Register for more 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 Chapter 19 Electrical Specifications for more 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.

5.3.2.4 Programming the PLL

Use this 9-step procedure to program the PLL. Table 5-1 lists the variables used and their meaning. Please also reference Figure 5-2

Table 5-1. 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: $f_{VCLKDES} = 4 \times 8$ MHz = 32 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.

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

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

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

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

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

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

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

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

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.

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

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

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.

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

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.

5.3.2.5 Special Programming Exceptions

The programming method described in 5.3.2.4 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 5.3.3 Base Clock Selector Circuit for more information.

5.3.3 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.

5.3.4 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 5-3. Figure 5-3 shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

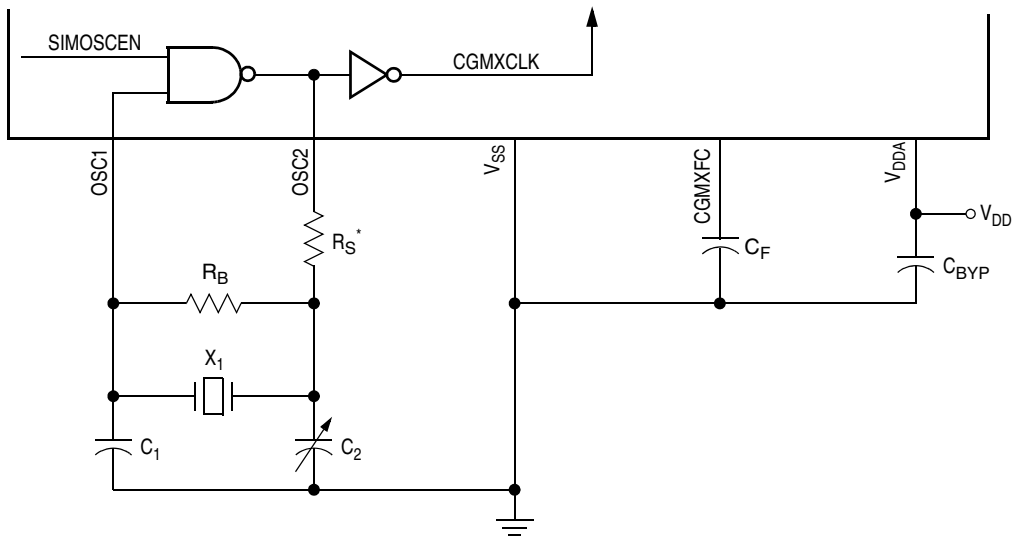
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S (optional)

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 5-3 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 5.9 Acquisition/Lock Time Specifications for routing information and more information on the filter capacitor's value and its effects on PLL performance).



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

Figure 5-3. CGM External Connections

5.4 I/O Signals

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

5.4.1 Crystal Amplifier Input Pin (OSC1)

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

5.4.2 Crystal Amplifier Output Pin (OSC2)

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

5.4.3 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.

5.4.4 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.

5.4.5 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal enables the oscillator and PLL.

5.4.6 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 5-3 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.

5.4.7 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.

5.4.8 CGM CPU Interrupt (CGMINT)

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

5.5 CGM Registers

Three registers control and monitor operation of the CGM:

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

5.5.1 PLL Control Register

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

Address: $001C

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

= Unimplemented

Figure 5-4. 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 a 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 5.3.3 Base Clock Selector Circuit for more information. 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 5.3.3 Base Clock Selector Circuit for more information. 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 5.3.3 Base Clock Selector Circuit for more information.

PCTL3–PCTL0 — Unimplemented

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

5.5.2 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	$\overline{ACQ}$	XLD	0	0	0	0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 5-5. 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 $\overline{ACQ}$ 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.

5.5.3 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 5-6. 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 5.3.2.1 Circuits and 5.3.2.4 Programming the PLL). A value of $0 in the multiplier select bits configures the modulo feedback divider the same as a value of $1. Reset initializes these bits to $6 to give a default multiply value of 6.

Table 5-2. VCO Frequency Multiplier (N) Selection

MUL7:MUL6:MUL5:MUL4	VCO Frequency Multiplier (N)
0000	1
0001	1
0010	2
0011	3
↓	↓

Table 5-2. VCO Frequency Multiplier (N) Selection (Continued)

MUL7:MUL6:MUL5:MUL4	VCO Frequency Multiplier (N)
1101	13
1110	14
1111	15

NOTE

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

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 5.3.2.1 Circuits, 5.3.2.4 Programming the PLL, and 5.5.1 PLL Control Register for more information.) VRS7–VRS4 cannot be written when the PLLON bit in the PLL control register (PCTL) is set. See 5.3.2.5 Special Programming Exceptions for more information. A value of $0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See 5.3.3 Base Clock Selector Circuit and 5.3.2.5 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.

5.6 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.

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 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.

5.7.2 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.

5.8 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 18.2 Break Module (BRK) for more information.

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.

5.9 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.

5.9.1 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 5.3.2.3 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 5.3.2.3 Manual and Automatic PLL Bandwidth Modes for more information.

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.

5.9.2 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 5-2). 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 5.9.3 Choosing a Filter Capacitor for more information.

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.

5.9.3 Choosing a Filter Capacitor

As described in 5.9.2 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}, (refer to 19.1 Introduction). 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.

5.9.4 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 5.9.3 Choosing a Filter Capacitor for more information).
- 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 5.3.2.2 Acquisition and Tracking Modes for more information.

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

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

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

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. (Refer to 5.3.2.3 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.

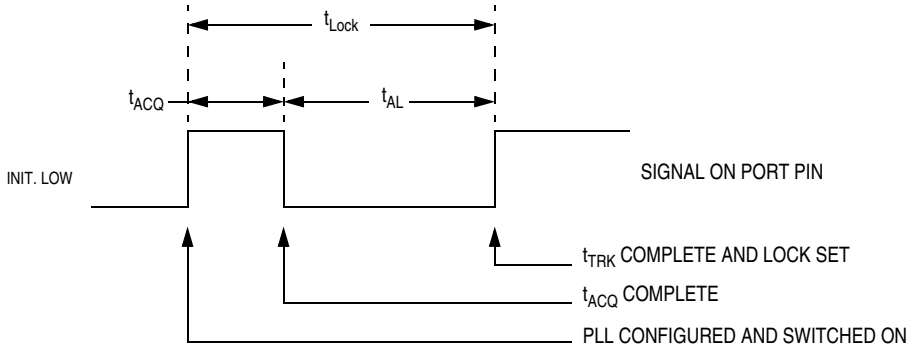
In manual mode, it is usually necessary to wait considerably longer than t_{Lock} before selecting the PLL clock (refer to 5.3.3 Base Clock Selector Circuit), because the factors described in 5.9.2 Parametric Influences on Reaction Time, may slow the lock time considerably.

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:



NOTE

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

Chapter 6
Configuration Register (CONFIG1)

6.1 Introduction

This section describes the configuration register (CONFIG1), which contains bits that configure these options:

- Resets caused by the low-voltage inhibit (LVI) module
- Power to the LVI module
- LVI enabled during stop mode
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- Computer operating properly (COP) module
- STOP instruction enable/disable

6.2 Functional Description

The configuration register is a write-once register. Out of reset, the configuration register will read the default value. Once the register is written, further writes will have no effect until a reset occurs.

NOTE

If the LVI module and the LVI reset signal are enabled, a reset occurs when V_{DD} falls to a voltage, LVI_{TRIPF}. Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises to a voltage, LVI_{TRIPR}.

Address: $001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	LVISTOP	R	LVIRST	LVIPWR	SSREC	COPL	STOP	COPD
Reset:	0	1	1	1	0	0	0	0

R = Reserved

Figure 6-1. Configuration Register (CONFIG1)

LVISTOP — LVI Stop Mode Enable Bit

LVISTOP enables the LVI module in stop mode. Refer to Chapter 11 Low-Voltage Inhibit (LVI).

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

NOTE

To have the LVI enabled in stop mode, the LVIPWR must be at a 1 and the LVISTOP bit must be at a 1. Take note that by enabling the LVI in stop mode, the stop I_{DD} current will be higher.

LVIRST — LVI Reset Enable Bit

LVIRST enables the reset signal from the LVI module. Refer to Chapter 11 Low-Voltage Inhibit (LVI).

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

LVIPWR — LVI Power Enable Bit

LVIPWR enables the LVI module. Refer to Chapter 11 Low-Voltage Inhibit (LVI).

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. Refer to Chapter 15 System Integration Module (SIM).

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

NOTE

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

COPL — COP Long Timeout

COPL enables the shorter COP timeout period. Refer to Chapter 8 Computer Operating Properly (COP).

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

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. Refer to Chapter 8 Computer Operating Properly (COP).

1 = COP module disabled
0 = COP module enabled

Chapter 7
Configuration Register (CONFIG2)

7.1 Introduction

This section describes the configuration register (CONFIG2). This register contains bits that configures the device to either the MC68HC08AZxx emulator or the MC68HC08ASxx emulator

7.2 Functional Description

The configuration register is a write-once register. Out of reset, the configuration register will read the default. Once the register is written, further writes will have no effect until a reset occurs.

Address: $FE09

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	EEDIVCLK	R	R	R	AS32A	R	R	R
Reset:	0	0	0	1	1	0	0	0

R = Reserved

Figure 7-1. Configuration Register (CONFIG2)

EEDIVCLK — EEPROM Timebase Divider Clock Select Bit

This bit selects the reference clock source for the EEPROM timebase divider module.

1 = EExDIV clock input is driven by internal bus clock
0 = EExDIV clock input is driven by CGMXCLK

AS32A— Device Indicator Bit

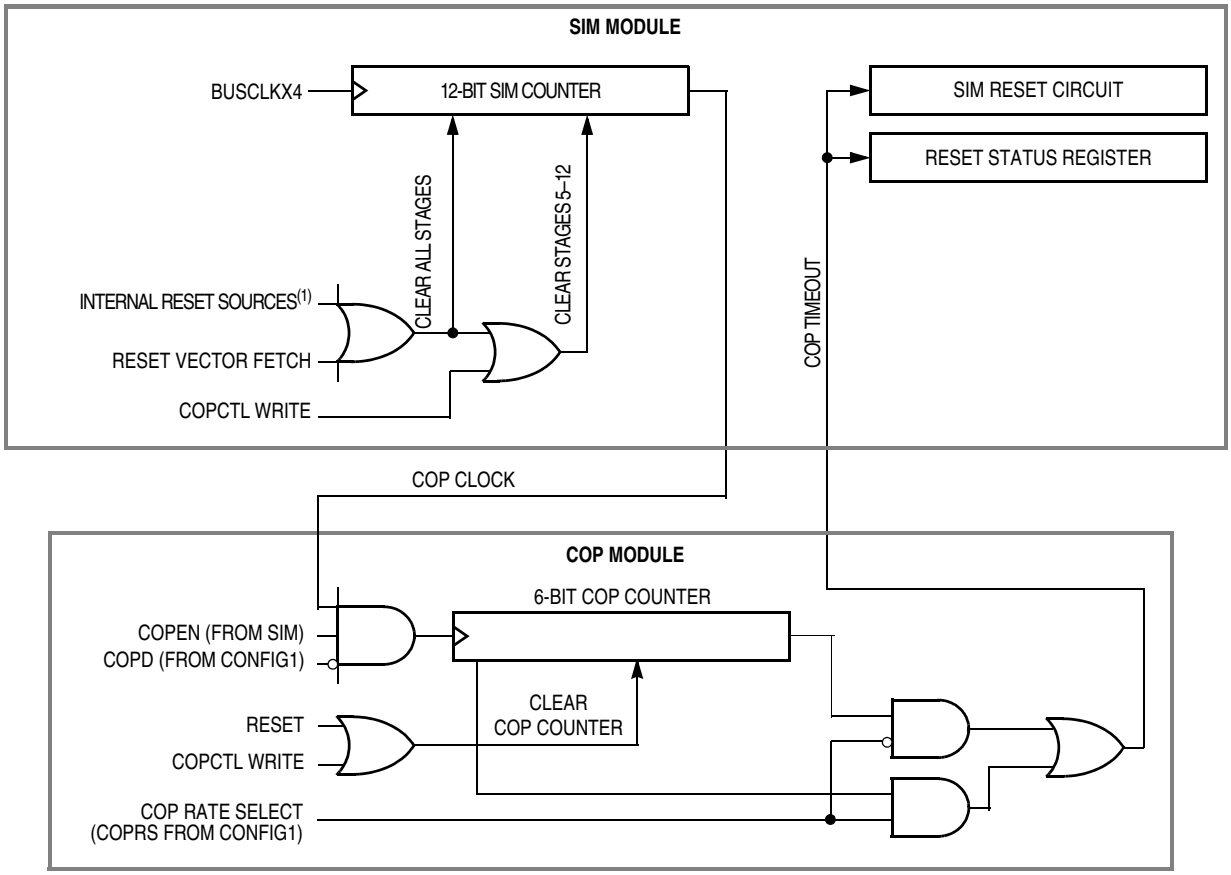
This bit is used to distinguish MC68HC908AS32A from older non-A suffix versions.

1 = A version
0 = Non-A version

Chapter 8
Computer Operating Properly (COP)

8.1 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the configuration 1 (CONFIG1) register.



1. See Chapter 15 System Integration Module (SIM) for more details.

Figure 8-1. COP Block Diagram

8.2 Functional Description

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler. If not cleared by software, the COP counter overflows and generates an asynchronous reset after 8176 or 262,128 CGMXCLK cycles, depending on the state of the COP long timeout bit, COPL, in the CONFIG1. When COPL = 0, a 4.9152-MHz crystal gives a COP timeout period of 53.3 ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 4 through 12 of the SIM counter.

NOTE

Service the COP immediately after reset and before entering or after exiting stop mode to guarantee the maximum time before the first COP counter overflow.

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles and sets the COP bit in the reset status register (RSR).

In monitor mode, the COP is disabled if the $\overline{\text{RST}}$ pin or the $\overline{\text{IRQ}}$ pin is held at V_{TST}. During the break state, V_{TST} on the $\overline{\text{RST}}$ pin disables the COP.

NOTE

Place COP clearing instructions in the main program and not in an interrupt subroutine. Such an interrupt subroutine could keep the COP from generating a reset even while the main program is not working properly.

8.3 I/O Signals

The following paragraphs describe the signals shown in Figure 8-1.

8.3.1 CGMXCLK

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

8.3.2 STOP Instruction

The STOP instruction clears the COP prescaler.

8.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) clears the COP counter and clears stages 12 through 4 of the COP prescaler (refer to 8.4 COP Control Register). Reading the COP control register returns the reset vector.

8.3.4 Power-On Reset

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

8.3.5 Internal Reset

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

8.3.6 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.

8.3.7 COPD

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register. See Chapter 6 Configuration Register (CONFIG1) for more information.

8.3.8 COPL

The COPL signal reflects the state of the COP rate select bit. (COPL) in the configuration register. See Chapter 6 Configuration Register (CONFIG1) for more information.

8.4 COP Control Register

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

Address: $FFFF

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LOW BYTE OF RESET VECTOR							
Write:	CLEAR COP COUNTER							
Reset:	Unaffected by reset							

Figure 8-2. COP Control Register (COPCTL)

8.5 Interrupts

The COP does not generate CPU interrupt requests.

8.6 Monitor Mode

The COP is disabled in monitor mode when V_{TST} is present on the $\overline{IRQ}$ pin or on the $\overline{RST}$ pin.

8.7 Low-Power Modes

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

8.7.1 Wait Mode

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

8.7.2 Stop Mode

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

The STOP bit in the configuration register (CONFIG1) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by clearing the STOP bit.

8.8 COP Module During Break Interrupts

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

Chapter 9
Central Processor Unit (CPU)

9.1 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.

9.2 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

9.3 CPU Registers

Figure 9-1 shows the five CPU registers. CPU registers are not part of the memory map.

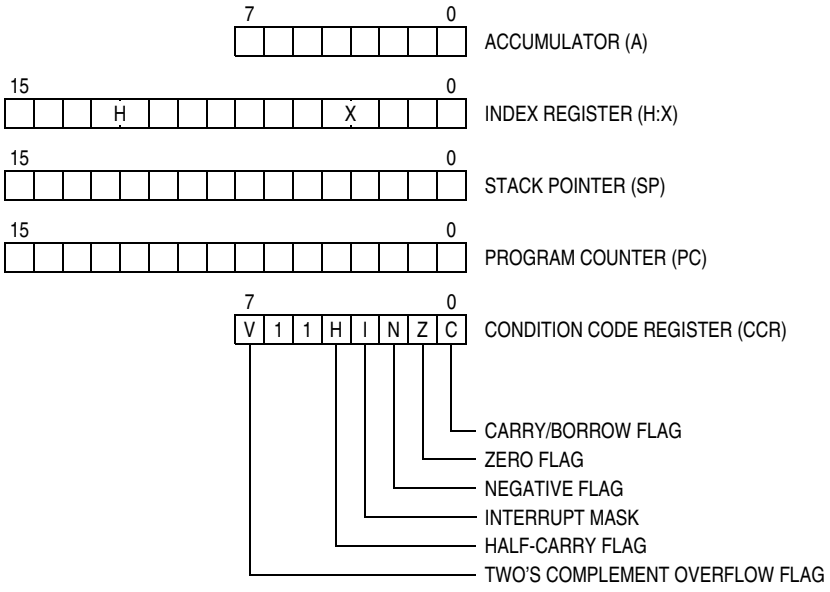


Figure 9-1. CPU Registers

9.3.1 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.

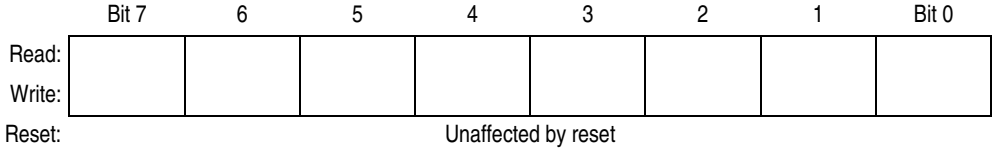


Figure 9-2. Accumulator (A)

9.3.2 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.

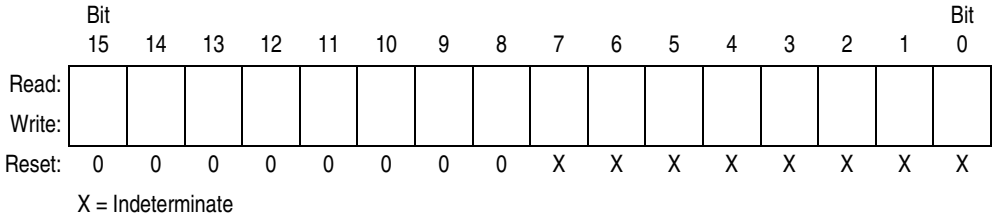


Figure 9-3. Index Register (H:X)

9.3.3 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.

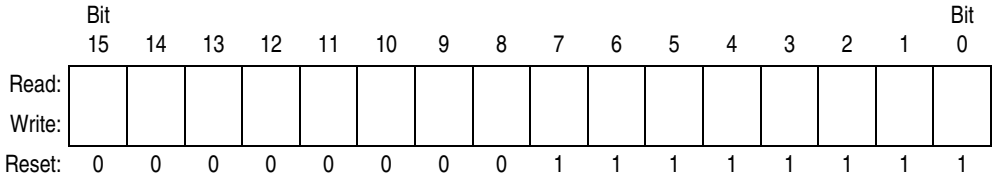


Figure 9-4. Stack Pointer (SP)

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.

9.3.4 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.

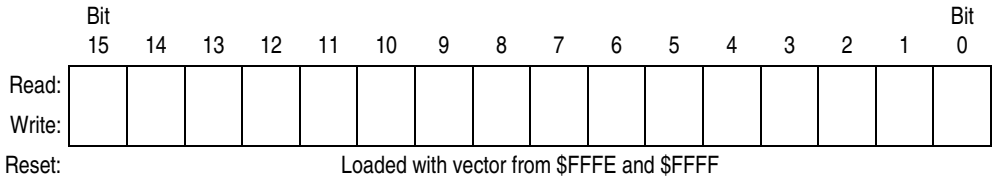


Figure 9-5. Program Counter (PC)

9.3.5 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 9-6. 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

9.4 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.

9.5 Low-Power Modes

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

9.5.1 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

9.5.2 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.

9.6 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.

9.7 Instruction Set Summary

Table 9-1 provides a summary of the M68HC08 instruction set.

Table 9-1. Instruction Set Summary (Sheet 1 of 6)

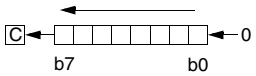
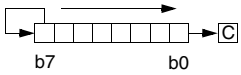
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9	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 9-1. 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 9-1. 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 9-1. Instruction Set Summary (Sheet 4 of 6)

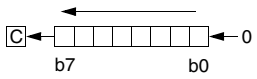
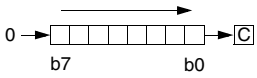
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,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)$_{Destination}$ ← (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 9-1. Instruction Set Summary (Sheet 5 of 6)

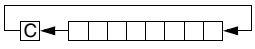
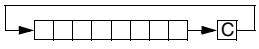
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	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 9-1. 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	*n*	Any bit
C	Carry/borrow bit	*opr*	Operand (one or two bytes)
CCR	Condition code register	PC	Program counter
dd	Direct address of operand	PCH	Program counter high byte
dd rr	Direct address of operand and relative offset of branch instruction	PCL	Program counter low byte
DD	Direct to direct addressing mode	REL	Relative addressing mode
DIR	Direct addressing mode	*rel*	Relative program counter offset byte
DIX+	Direct to indexed with post increment addressing mode	rr	Relative program counter offset byte
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing	SP1	Stack pointer, 8-bit offset addressing mode
EXT	Extended addressing mode	SP2	Stack pointer 16-bit offset addressing mode
ff	Offset byte in indexed, 8-bit offset addressing	SP	Stack pointer
H	Half-carry bit	U	Undefined
H	Index register high byte	V	Overflow bit
hh ll	High and low bytes of operand address in extended addressing	X	Index register low byte
I	Interrupt mask	Z	Zero bit
ii	Immediate operand byte	&	Logical AND
IMD	Immediate source to direct destination addressing mode	|	Logical OR
IMM	Immediate addressing mode	⊕	Logical EXCLUSIVE OR
INH	Inherent addressing mode	( )	Contents of
IX	Indexed, no offset addressing mode	–( )	Negation (two's complement)
IX+	Indexed, no offset, post increment addressing mode	#	Immediate value
IX+D	Indexed with post increment to direct addressing mode	«	Sign extend
IX1	Indexed, 8-bit offset addressing mode	←	Loaded with
IX1+	Indexed, 8-bit offset, post increment addressing mode	?	If
IX2	Indexed, 16-bit offset addressing mode	:	Concatenated with
M	Memory location	↕	Set or cleared
N	Negative bit	—	Not affected

9.8 Opcode Map

See Table 9-2.

Table 9-2. 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
Low Byte of Opcode in Hexadecimal	0

High Byte of Opcode in Hexadecimal

5 Cycles
BRSET0 Opcode Mnemonic
3 DIR Number of Bytes / Addressing Mode

Chapter 10
External Interrupt Module (IRQ)

10.1 Introduction

This section describes the non-maskable external interrupt (IRQ) input.

10.2 Features

Features include:

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

10.3 Functional Description

A falling edge applied to the external interrupt pin can latch a CPU interrupt request. Figure 10-2 shows the structure of the IRQ module.

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

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the 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 both interrupt latches.

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 MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

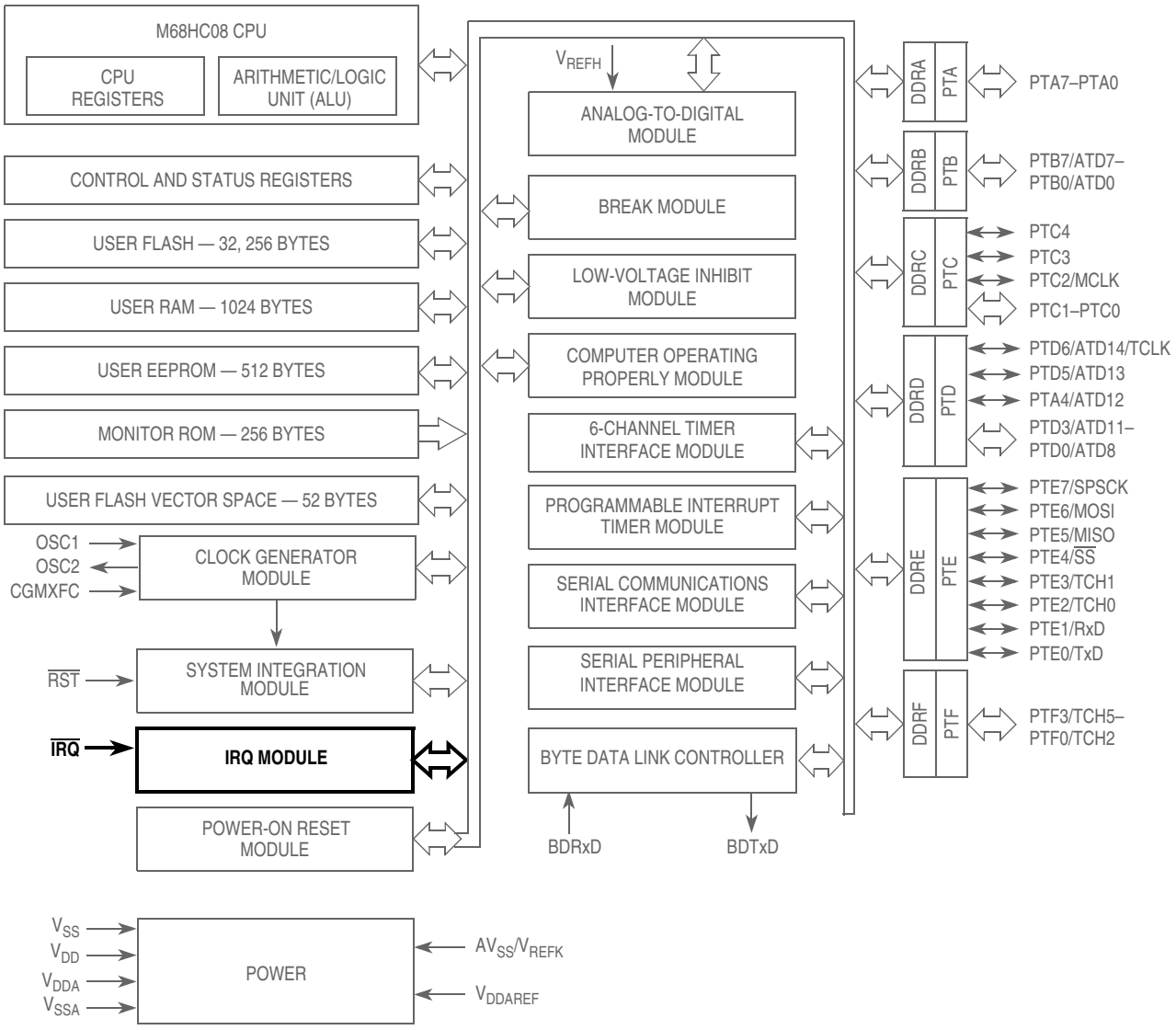


Figure 10-1. Block Diagram Highlighting IRQ Block and Pins

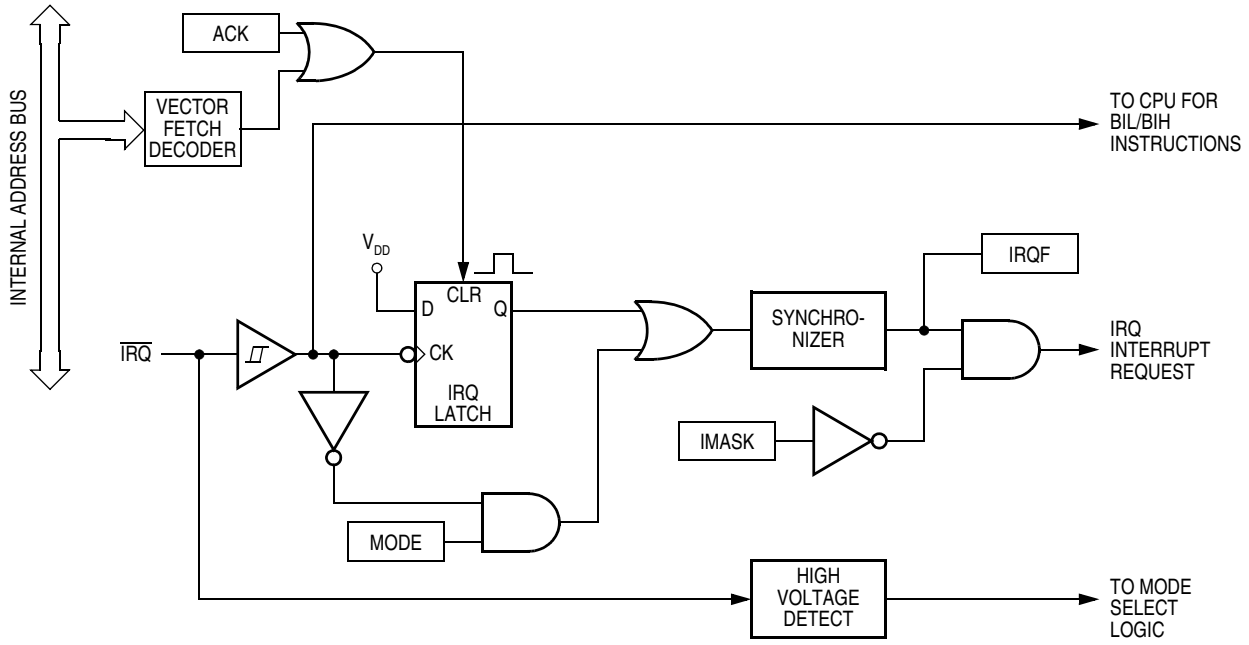


Figure 10-2. IRQ Block Diagram

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. Refer to Figure 10-3.

10.4 $\overline{\text{IRQ}}$ 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 IRQ1 latch remains set.

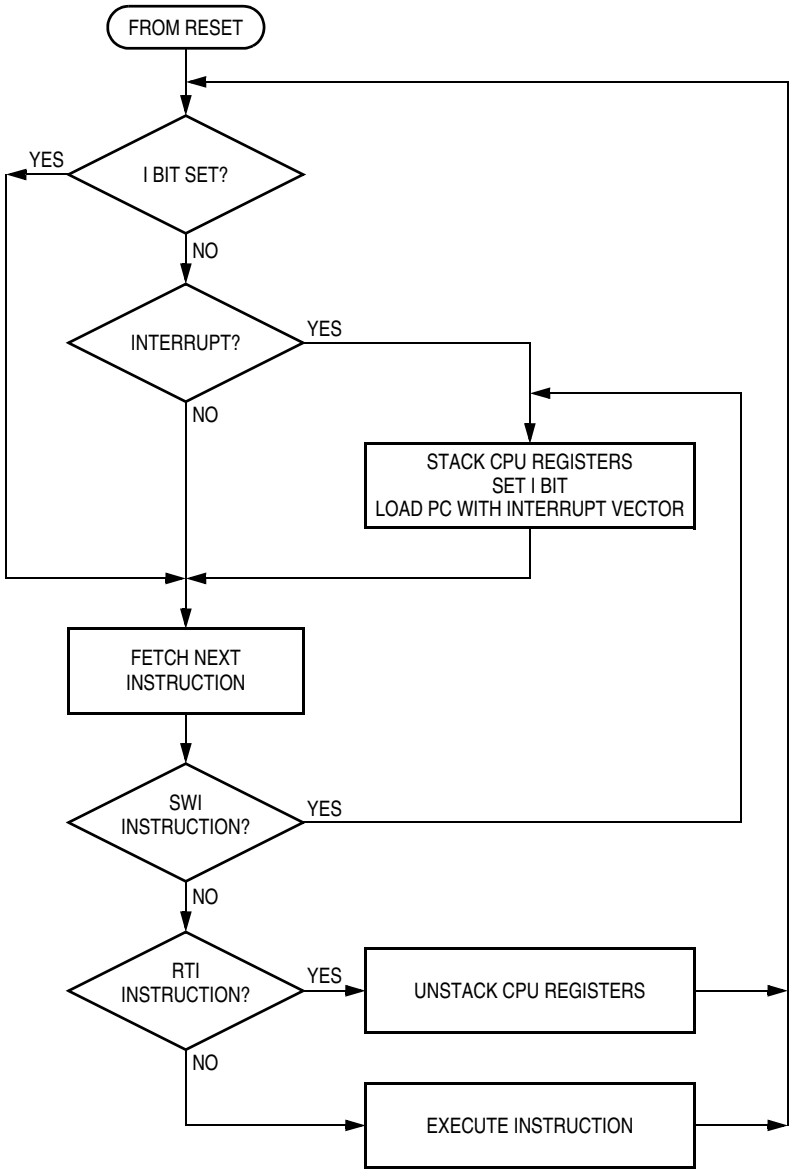


Figure 10-3. IRQ Interrupt Flowchart

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

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.

Use the BIH or BIL instruction 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.

10.5 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 Chapter 15 System Integration Module (SIM) for additional information.

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 latch 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.

10.6 IRQ Status and Control Register

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

- Shows the state of the IRQ interrupt flag
- Clears the IRQ interrupt latch
- Masks IRQ interrupt request
- Controls triggering sensitivity of the $\overline{\text{IRQ}}$ interrupt pin

Address: $001A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	IRQF	0	IMASK	MODE
Write:						ACK		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 10-4. IRQ Status and Control Register (ISCR)

IRQF — $\overline{\text{IRQ}}$ Flag Bit

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

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

ACK — $\overline{\text{IRQ}}$ 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 11
Low-Voltage Inhibit (LVI)

11.1 Introduction

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

11.2 Features

Features of the LVI module include:

- Programmable LVI reset
- Programmable power consumption

NOTE

If a low-voltage interrupt (LVI) occurs during programming of EEPROM or FLASH 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.

11.3 Functional Description

Figure 11-1 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, 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

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, LVISTOP, and LVIRST are in the configuration register, CONFIG1 (see Chapter 6 Configuration Register (CONFIG1)).

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 (see 11.3.2 Forced Reset Operation). 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.

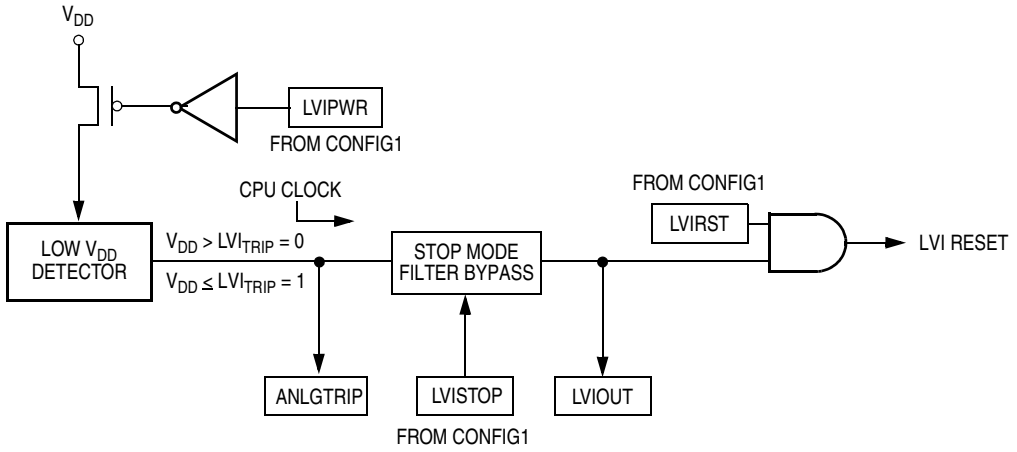


Figure 11-1. LVI Module Block Diagram

11.3.1 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 configuration register, the LVIPWR bit must be at 1 to enable the LVI module, and the LVIRST bit must be at 0 to disable LVI resets.

11.3.2 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 configuration register, the LVIPWR and LVIRST bits must be at 1 to enable the LVI module and to enable LVI resets.

11.3.3 False Reset Protection

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.

11.4 LVI Status Register

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 11-2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the LVI_{TRIPF} voltage (see Table 11-1). Reset clears the LVIOUT bit.

Table 11-1. LVIOUT Bit Indication

V_{DD} At Level:	LVIOUT
$V_{DD} > {}^{LVI}TRIPR$	0
$V_{DD} < {}^{LVI}TRIPF$	1
${}^{LVI}TRIPF < V_{DD} < {}^{LVI}TRIPR$	Previous value

11.5 LVI Interrupts

The LVI module does not generate interrupt requests.

11.6 Low-Power Modes

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

11.6.1 Wait Mode

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

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

11.6.2 Stop Mode

With the LVISTOP and LVIPWR bits in the configuration register programmed to a 1, the LVI module will be active after a STOP instruction. Because CPU clocks are disabled during stop mode, the LVI trip will generate a reset and bring the MCU out of stop.

With the LVIPWR bit in the configuration register programmed to a 1 and the LVISTOP bit at 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. It is not intended that users operate the microcontroller at lower than the specified operating voltage (V_{DD}).

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.

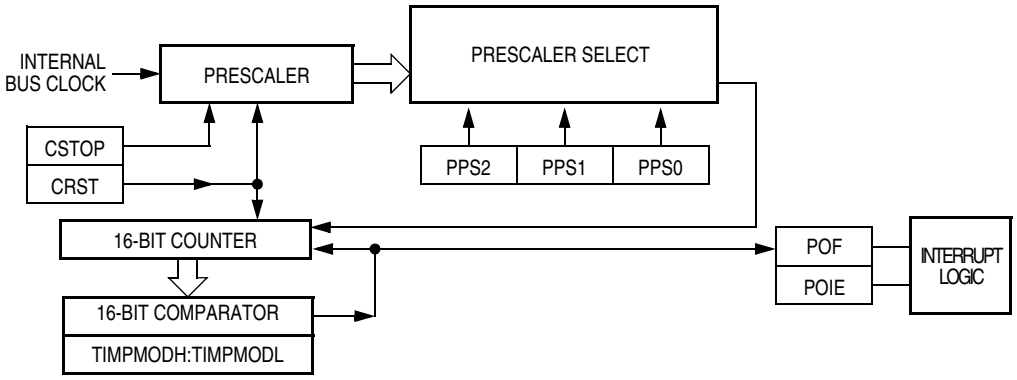


Figure 12-1. PIT Block Diagram

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 15-18. SIM Reset Status Register (SRSR)).

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 12-2. 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 a 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 0. 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

Figure 12-3. PIT Counter Registers (PCNTH–PCNTL)

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 12-3. PIT Counter Registers (PCNTH–PCNTL) (Continued)

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 12-4. PIT Counter Modulo Registers (PMODH–PMODL)

NOTE

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

Chapter 13
Input/Output Ports

13.1 Introduction

Forty bidirectional input/output (I/O) pins form six 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.

13.2 Port A

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

13.2.1 Port A Data Register

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

Address: $0000

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

Figure 13-1. Port A Data Register (PTA)

PTA[7:0] — Port A Data Bits

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

13.2.2 Data Direction Register A

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.

Address: $0004

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

Figure 13-2. Data Direction Register A (DDRA)

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

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

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

NOTE

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

Figure 13-3 shows the port A I/O logic.

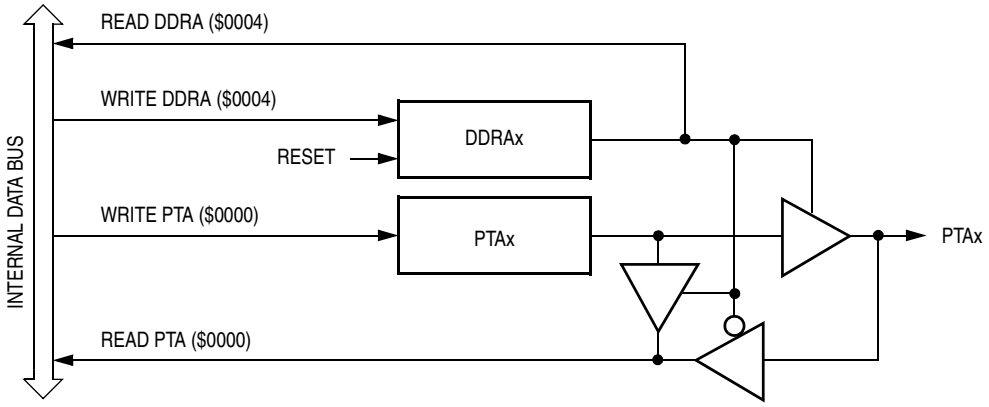


Figure 13-3. Port A I/O Circuit

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	Input, Hi-Z	DDRA[7:0]	Pin	PTA[7:0](1)
1	X	Output	DDRA[7:0]	PTA[7:0]	PTA[7:0](1)

X = don't care
Hi-Z = high impedance
1. 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

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 13-4. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

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

ATD[7:0] — ADC Channels

PTB7/ATD7–PTB0/ATD0 are eight of the 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 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. See Chapter 3 Analog-to-Digital Converter (ADC).

Data direction register B (DDRB) does not affect the data direction of port B pins that are being used by the ADC. However, the DDRB bits always determine whether reading port B returns to the states of the latches or a 0.

13.3.2 Data Direction Register B

Data direction register B determines whether each port B pin is an input or an output. Writing a 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 13-5. Data Direction Register B (DDRB)

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

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

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

NOTE

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

Figure 13-6 shows the port B I/O logic.

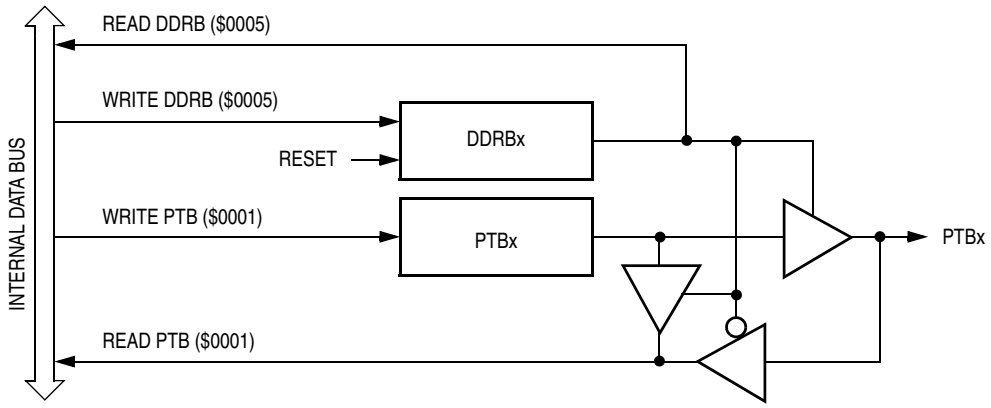


Figure 13-6. Port B I/O Circuit

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	Input, Hi-Z	DDRB[7:0]	Pin	PTB[7:0](1)
1	X	Output	DDRB[7:0]	PTB[7:0]	PTB[7:0](1)

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

13.4 Port C

Port C is an 5-bit general-purpose bidirectional I/O port.

13.4.1 Port C Data Register

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

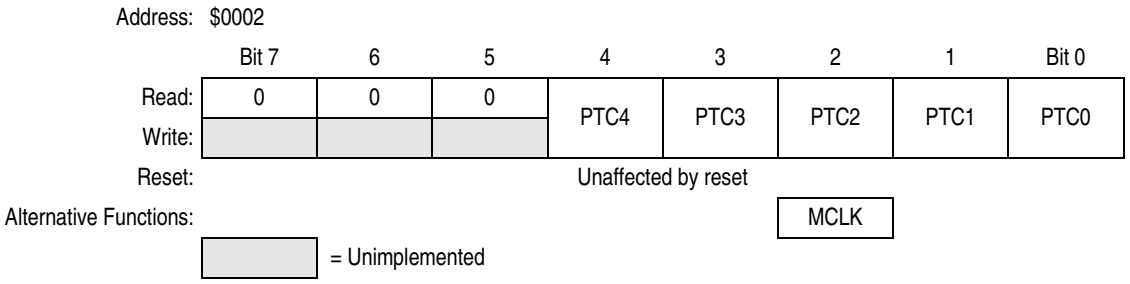


Figure 13-7. Port C Data Register (PTC)

PTC[4:0] — 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 PTC[4:0].

MCLK — System Clock Bit

The system clock is driven out of PTC2 when enabled by MCLKEN bit in PTCDDR7.

13.4.2 Data Direction Register C

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.

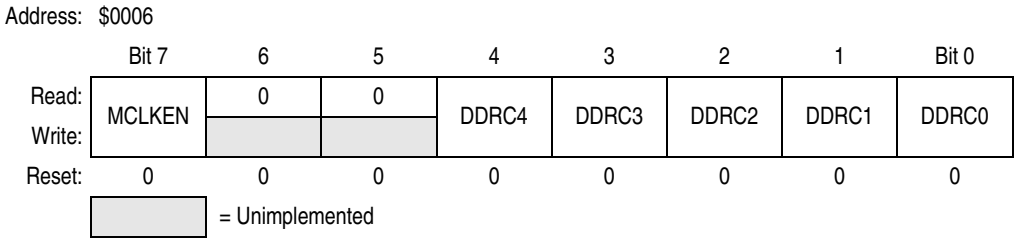


Figure 13-8. Data Direction Register C (DDRC)

MCLKEN — MCLK Enable Bit

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

1 = MCLK output enabled
0 = MCLK output disabled

DDRC[4:0] — Data Direction Register C Bits

These read/write bits control port C data direction. Reset clears DDRC[4:0], configuring all port C pins as inputs.

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

NOTE

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

Figure 13-9 shows the port C I/O logic.

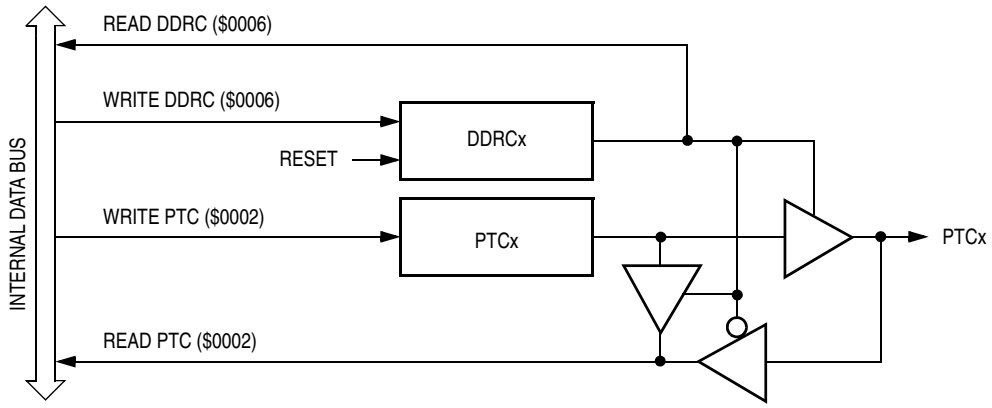


Figure 13-9. Port C I/O Circuit

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

Bit Value	PTC Bit	I/O Pin Mode	Accesses to DDRC	Accesses to PTC	
			Read/Write	Read	Write
0	2	Input, Hi-Z	DDRC[2]	Pin	PTC2
1	2	Output	DDRC[2]	0	—
0	X	Input, Hi-Z	DDRC[4:0]	Pin	PTC[4:0](1)
1	X	Output	DDRC[4:0]	PTC[4:0]	PTC[4:0](1)

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

13.5 Port D

Port D is an 7-bit general-purpose I/O port.

13.5.1 Port D Data Register

Port D is a 7-bit special function port that shares seven of its pins with the analog to digital converter and two with the timer interface modules.

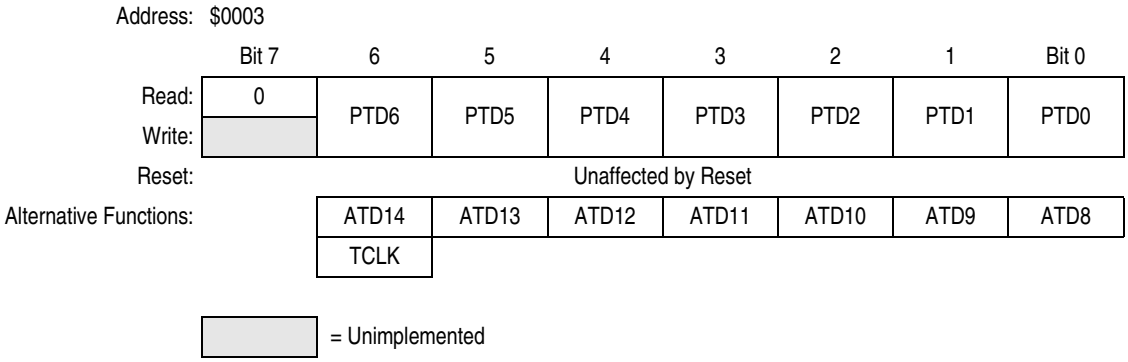


Figure 13-10. Port D Data Register (PTD)

PTD[6:0] — Port D Data Bits

PTD[6:0] are read/write, software programmable bits. Data direction of PTD[6:0] pins are under the control of the corresponding bit in data direction register D.

ATD[14:8] — ADC Channel Status Bits

PTD6/ATD14/TCLK–PTD0/ATD8 are seven of the 15 ADC channels. The ADC channel select bits, CH[4:0], determine whether the PTD6/ATD14/TCLK–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. See Chapter 3 Analog-to-Digital Converter (ADC).

Data direction register D (DDRD) does not affect the data direction of port D pins that are being used by the TIM. However, the DDRD bits always determine whether reading port D returns the states of the latches or a 0.

TCLK — Timer Clock Input Bit

The PTD6/ATD14/TACLK pin is the external clock input for the timer interface module (TIM). The prescaler select bits, PS[2:0], select PTD6/ATD14/TACLK as the TIM clock input. See Chapter 17 Timer Interface Module (TIM).

When not selected as the TIM clock, PTD6/ATD14/TACLK is available for general-purpose I/O. While TCLK is selected corresponding DDRD bits have no effect.

13.5.2 Data Direction Register D

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:	0	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Write:								
Reset:	0	0	0	0	0	0	0	0
	= Unimplemented							

Figure 13-11. Data Direction Register D (DDRD)

DDRD[6:0] — Data Direction Register D Bits

These read/write bits control port D data direction. Reset clears DDRD[6:0], 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 13-12 shows the port D I/O logic.

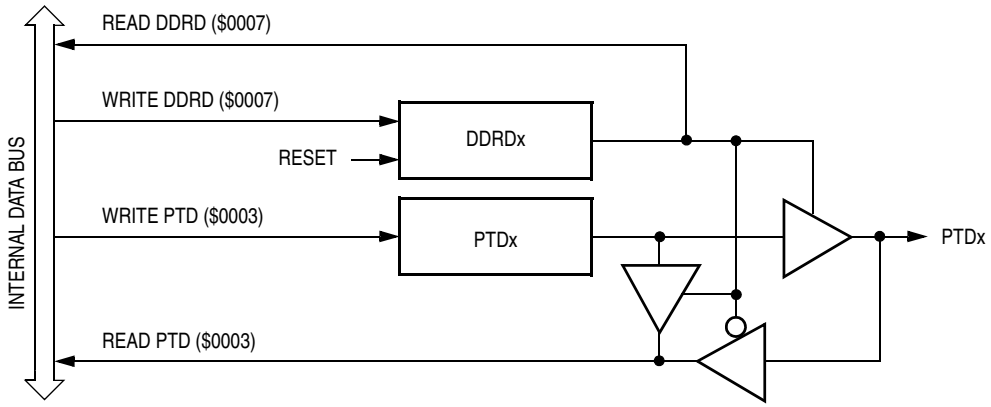


Figure 13-12. Port D I/O Circuit

When bit DDRDx is a 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	Input, Hi-Z	DDRD[6:0]	Pin	PTD[6:0](1)
1	X	Output	DDRD[6:0]	PTD[6:0]	PTD[6:0](1)

X = don't care
Hi-Z = high impedance
1. 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 TIM, 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

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

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
Reset:	Unaffected by reset							
Alternative Function:	SPSCK	MOSI	MISO	$\overline{SS}$	TCH1	TCH0	RxD	TxD

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

PTE[7:0] — Port E Data Bits

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

SPSCK — SPI Serial Clock Bit

The PTE7/SPSCK pin is the serial clock input of an SPI slave module and serial clock output of an SPI master module. When the SPE bit is clear, the PTE7/SPSCK pin is available for general-purpose I/O. See Chapter 16 Serial Peripheral Interface (SPI).

MOSI — Master Out/Slave In Bit

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.

MISO — Master In/Slave Out Bit

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 Chapter 16 Serial Peripheral Interface (SPI).

$\overline{SS}$ — Slave Select Bit

The PTE4/$\overline{SS}$ pin is the slave select input of the SPI module. When the SPE bit is clear, or when the SPI master bit (SPMSTR) is set and MODFEN bit is low, the PTE4/$\overline{SS}$ pin is available for general-purpose I/O. See 16.12.4 SS (Slave Select). When the SPI is enabled as a slave, the DDRF0 bit in data direction register E (DDRE) has no effect on the PTE4/$\overline{SS}$ 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.

TCH[1:0] — Timer Channel I/O Bits

The PTE3/TACH1–PTE2/TACH0 pins are the TIM 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 Chapter 17 Timer Interface Module (TIM).

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIM. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. See Table 13-5.

RxD — SCI Receive Data Input Bit

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 14.8.1 SCI Control Register 1.

NOTE

Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the 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

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 13-14. Data Direction Register E (DDRE)

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

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

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

NOTE

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

Figure 13-15 shows the port E I/O logic.

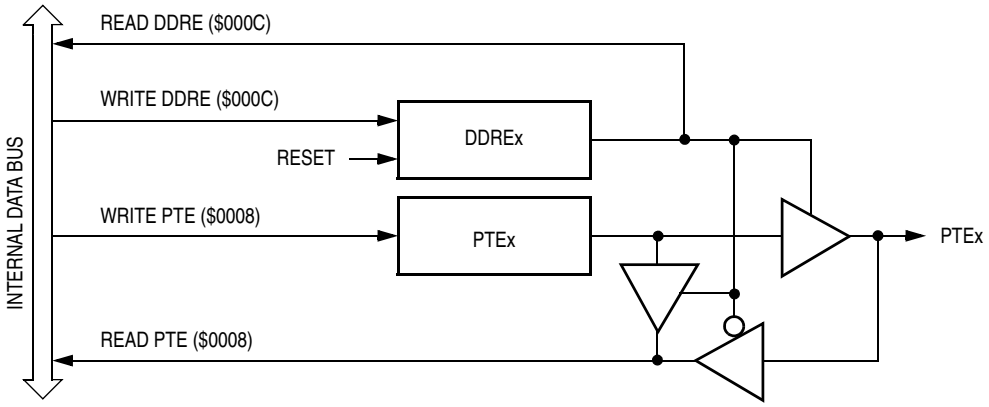


Figure 13-15. Port E I/O Circuit

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	Input, Hi-Z	DDRE[7:0]	Pin	PTE[7:0](1)
1	X	Output	DDRE[7:0]	PTE[7:0]	PTE[7:0](1)

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

13.7 Port F

Port F is a 4-bit special function port that shares four of its pins with the TIM.

13.7.1 Port F Data Register

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

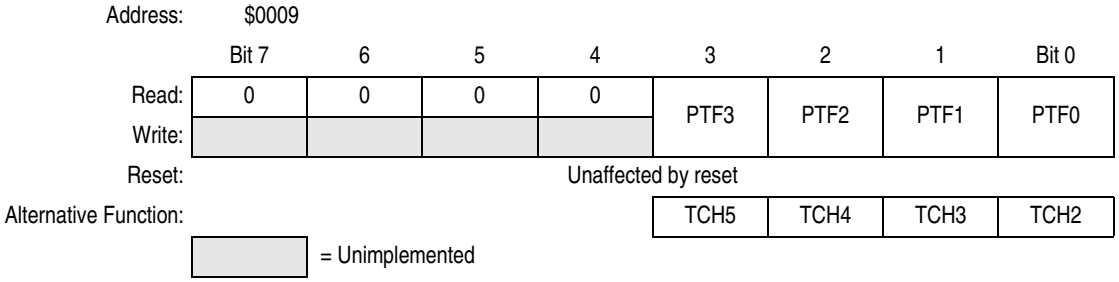


Figure 13-16. Port F Data Register (PTF)

PTF[3:0] — 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[3:0].

TCH[5:2] — Timer A Channel I/O Bits

The PTF3–PTF0/TCH2 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF3–PTF0/TCH2 pins are timer channel I/O pins or general-purpose I/O pins. See 17.8.1 TIM 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 the TIM. 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

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	0	0	0	DDRF3	DDRF2	DDRF1	DDRF0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-17. Data Direction Register F (DDRF)

DDRF[3:0] — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF[3:0], 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 13-18 shows the port F I/O logic.

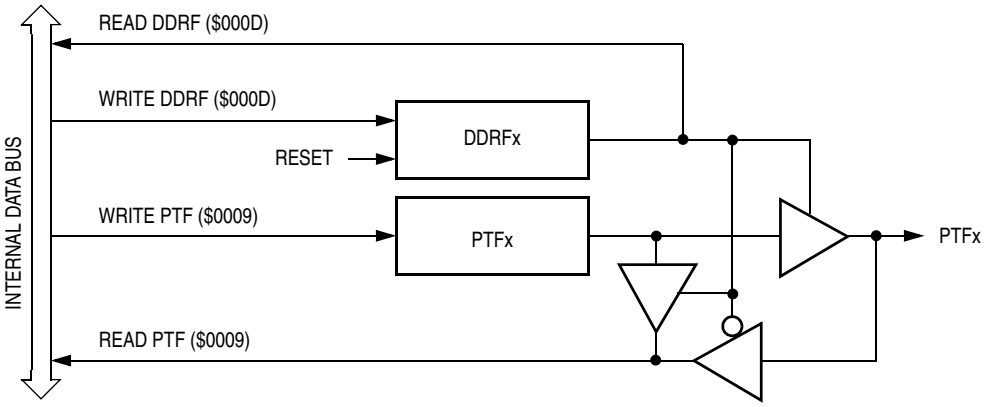


Figure 13-18. Port F I/O Circuit

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	Input, Hi-Z	DDRF[3:0]	Pin	PTF[3:0](1)
1	X	Output	DDRF[3:0]	PTF[3:0]	PTF[3:0](1)

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

Chapter 14
Serial Communications Interface (SCI)

14.1 Introduction

The serial communications interface (SCI) allows 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

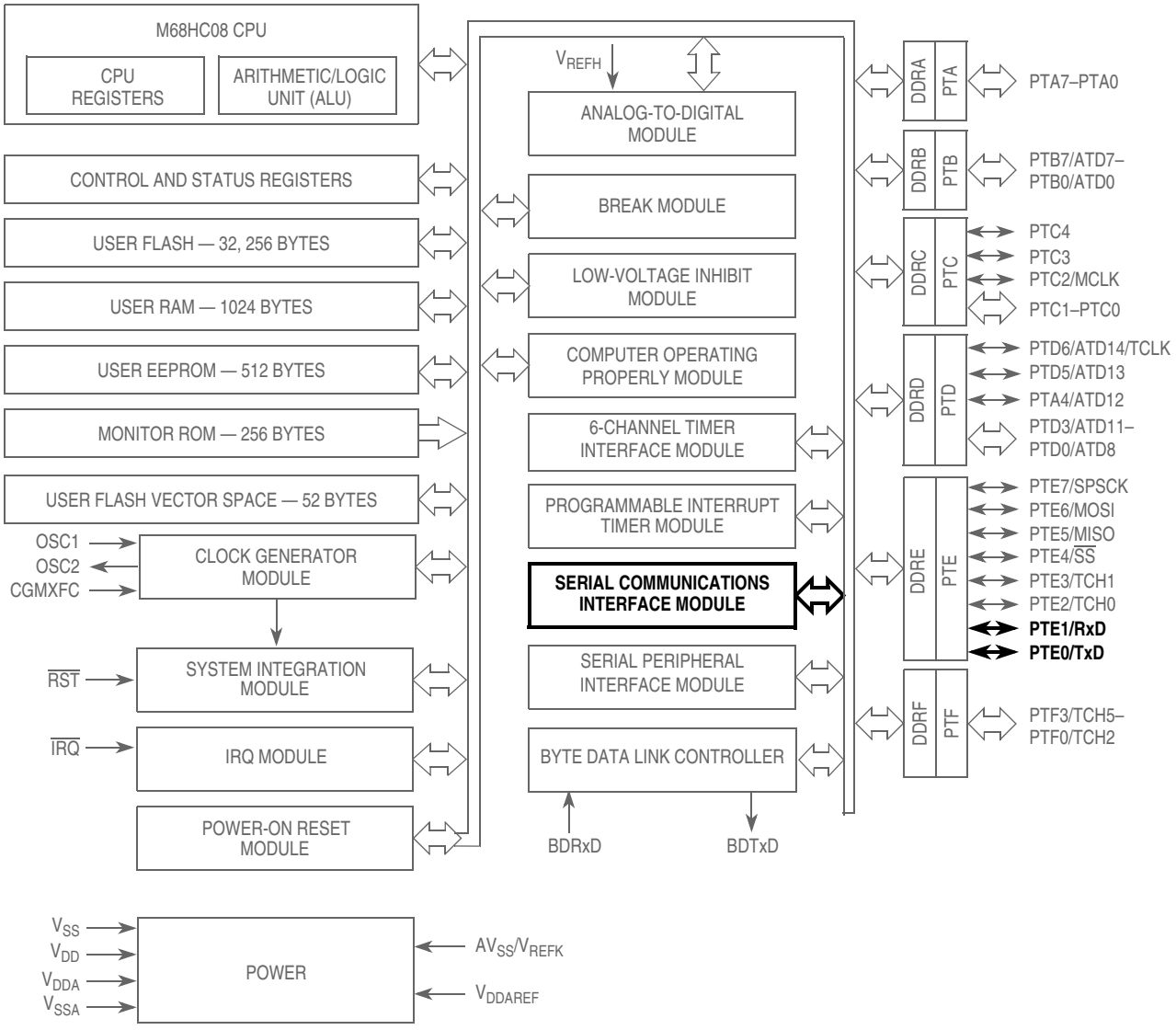


Figure 14-1. Block Diagram Highlighting SCI Block and Pins

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/SCRxD	PTE0/SCTxD

14.4 Functional Description

Figure 14-2 shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

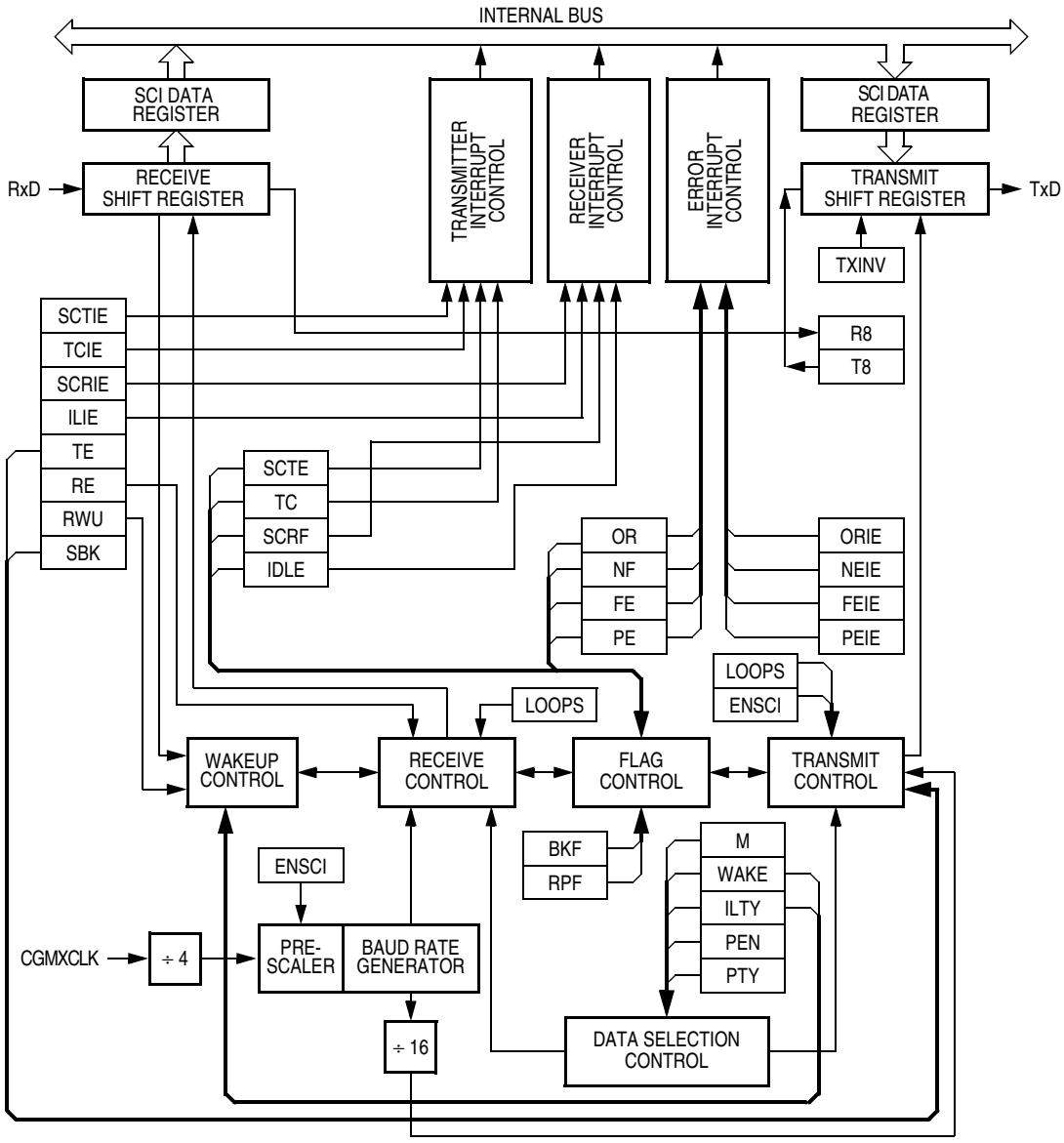


Figure 14-2. SCI Module Block Diagram

14.4.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 14-3.

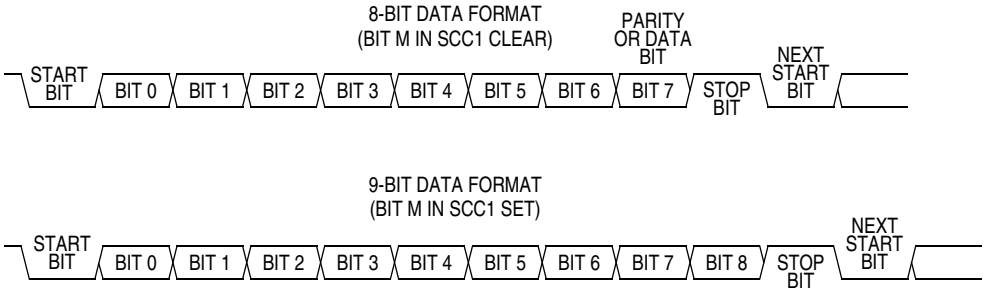


Figure 14-3. SCI Data Formats

14.4.2 Transmitter

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

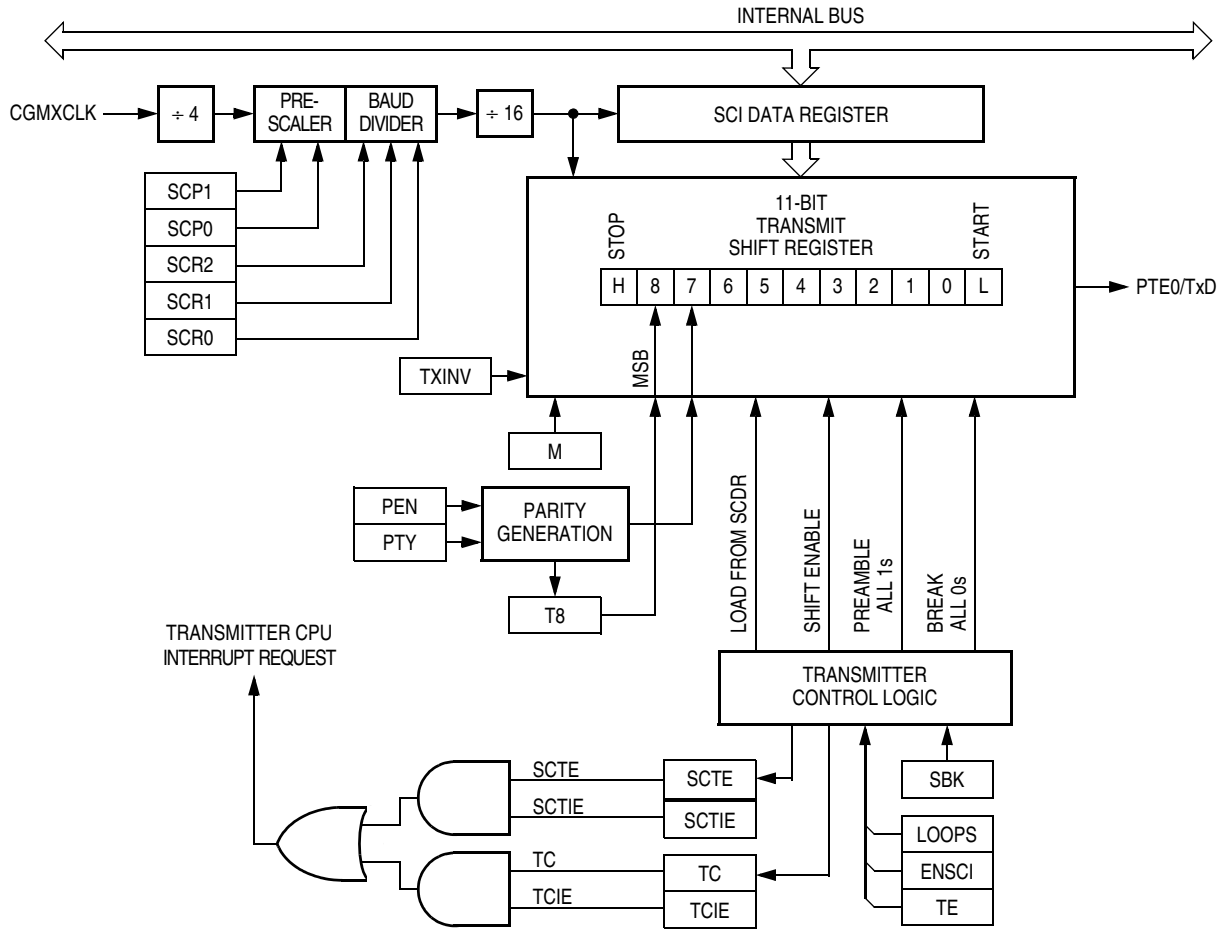


Figure 14-4. SCI Transmitter

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).

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

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 a 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 14-5 shows the structure of the SCI receiver.

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).

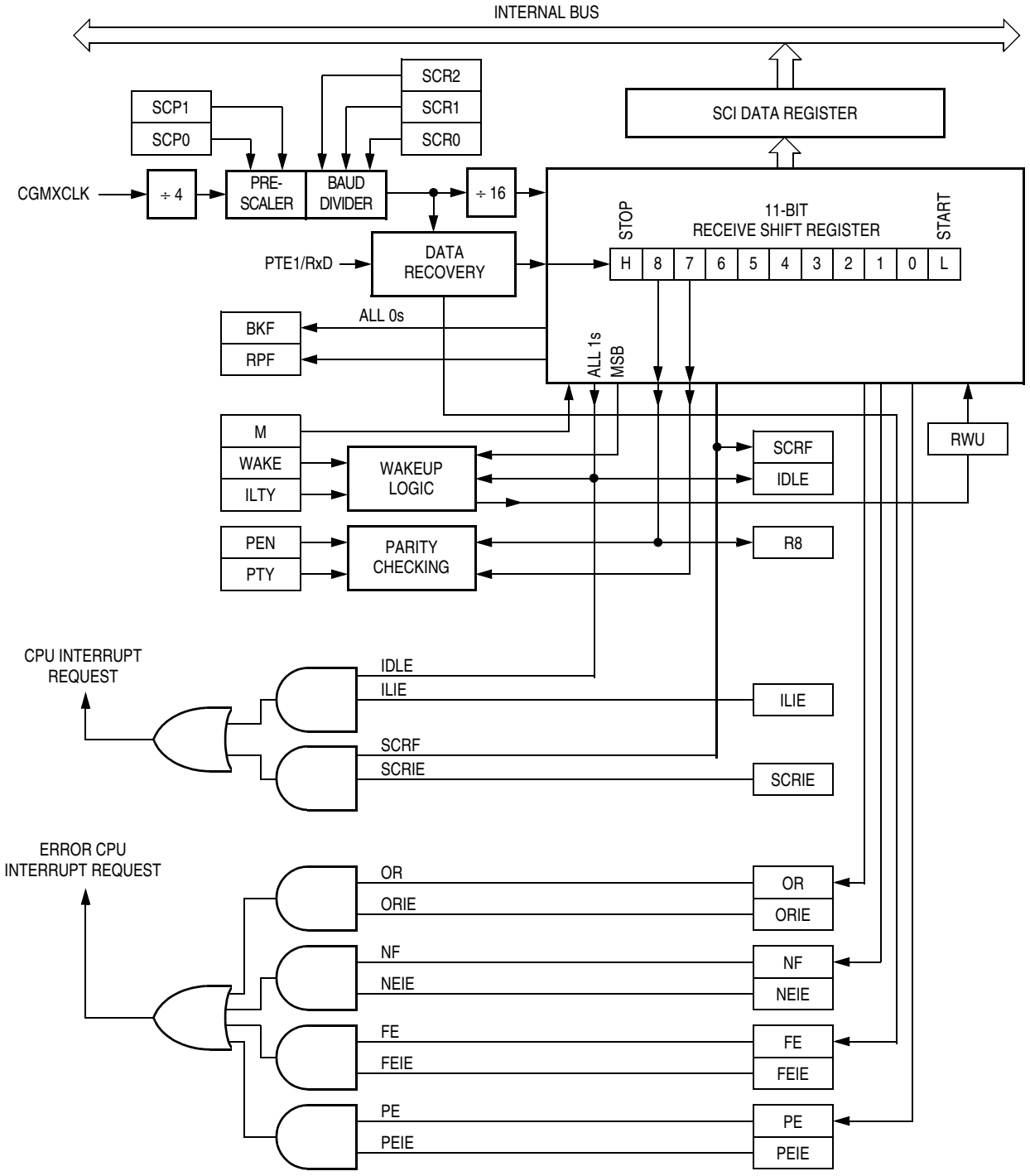


Figure 14-5. SCI Receiver Block Diagram

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 14-6):

- 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.

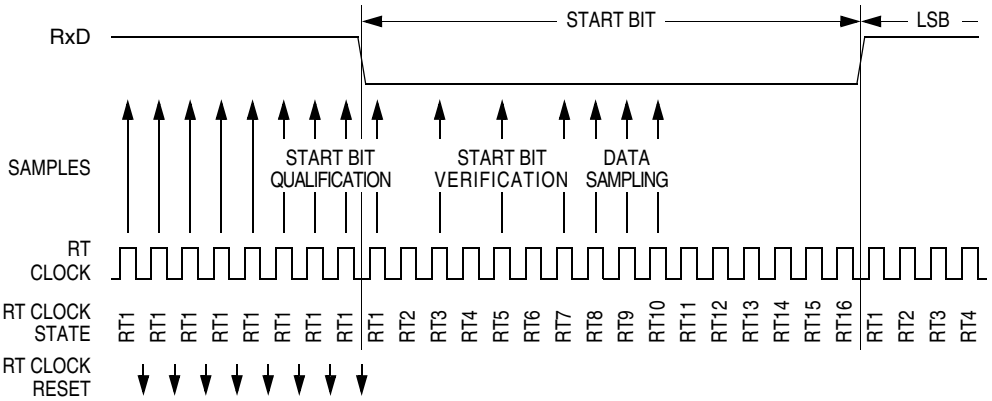


Figure 14-6. Receiver Data Sampling

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

Start bit verification is not successful if any two of the three verification samples are 1s. If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 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 14-7 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

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 14-7, the receiver counts 154 RT cycles at the point when the count of the transmitting device is
9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

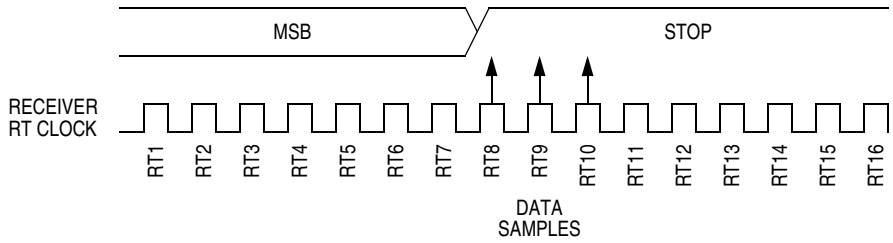


Figure 14-7. Slow Data

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 14-7, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

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

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

Fast Data Tolerance

Figure 14-8 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

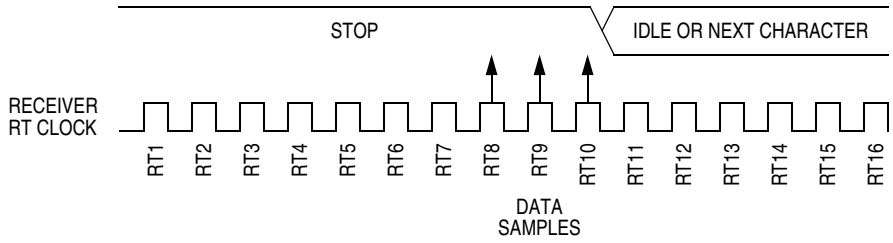


Figure 14-8. Fast Data

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

With the misaligned character shown in Figure 14-8, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

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

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.90\dot{}\%$$

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

With the misaligned character shown in Figure 14-8, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

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

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

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

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

14.5.2 Stop Mode

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

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

To allow software to clear status bits during a break interrupt, write a 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/SCTxD (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/SCRxD (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 14-9. 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 and PTY	Start Bits	Data Bits	Parity	Stop Bits	Character Length
0	0X	1	8	None	1	10 bits
1	0X	1	9	None	1	11 bits
0	10	1	7	Even	1	10 bits
0	11	1	7	Odd	1	10 bits
1	10	1	8	Even	1	11 bits
1	11	1	8	Odd	1	11 bits

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 14-10. SCI Control Register 2 (SCC2)

SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests. 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

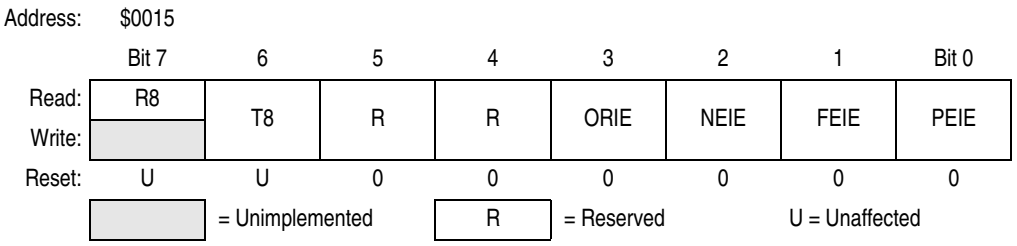


Figure 14-11. SCI Control Register 3 (SCC3)

R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.

When the SCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

T8 — Transmitted Bit 8

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

ORIE — Receiver Overrun Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the receiver overrun bit, OR.

1 = SCI error CPU interrupt requests from OR bit enabled
0 = SCI error CPU interrupt requests from OR bit disabled

NEIE — Receiver Noise Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the noise error bit, NE. Reset clears NEIE.

1 = SCI error CPU interrupt requests from NE bit enabled
0 = SCI error CPU interrupt requests from NE bit disabled

FEIE — Receiver Framing Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the framing error bit, FE. Reset clears FEIE.

1 = SCI error CPU interrupt requests from FE bit enabled
0 = SCI error CPU interrupt requests from FE bit disabled

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables SCI receiver CPU interrupt requests generated by the parity error bit, PE. 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 14-12. SCI Status Register 1 (SCS1)

SCTE — SCI Transmitter Empty Bit

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an SCI transmitter CPU interrupt request. When the SCTIE bit in SCC2 is set, SCTE generates an SCI transmitter CPU interrupt request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR. Reset sets the SCTE bit.

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

TC — Transmission Complete Bit

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an SCI transmitter CPU interrupt request if the TCIE bit in SCC2 is also set. TC is 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 14-13 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

In applications that are subject to software latency or in which it is important to know which byte is lost due to an overrun, the flag-clearing routine can check the OR bit in a second read of SCS1 after reading the data register.

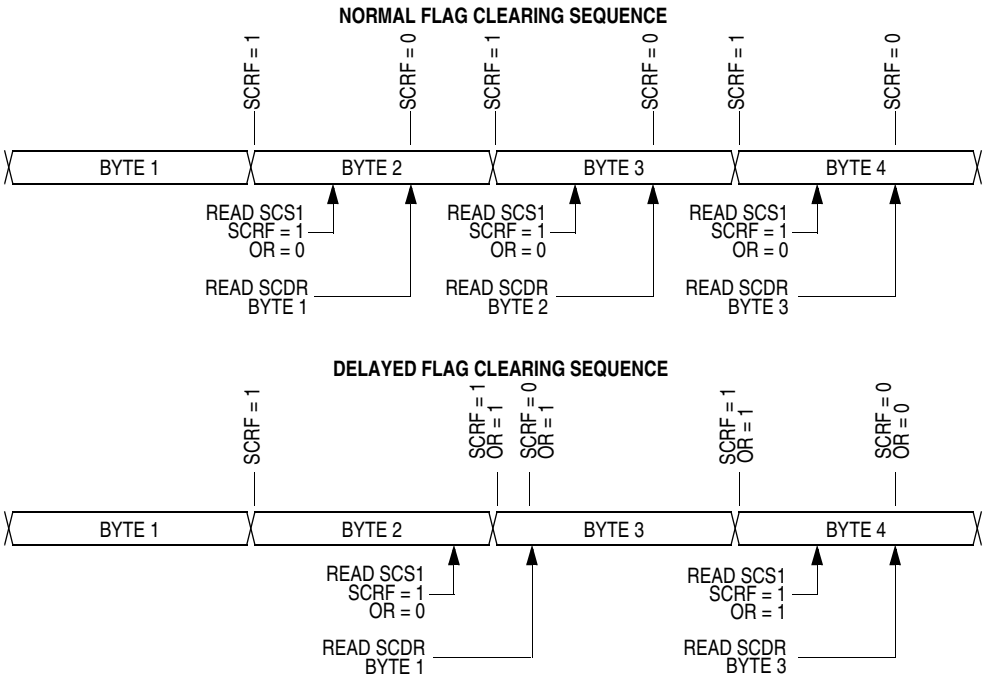


Figure 14-13. Flag Clearing Sequence

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 14-14. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the 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 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 14-15. SCI Data Register (SCDR)

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

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

NOTE

Do not use read-modify-write instructions on the SCI data register.

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	0	SCR2	SCR1	SCR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-16. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in Table 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 (SIM), which supports up to 32 external and/or internal interrupts. Together with the central processor unit (CPU), the SIM controls all MCU activities.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.

Table 15-1 shows the internal signal names used in this section.

Table 15-1. Signal Name Conventions

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

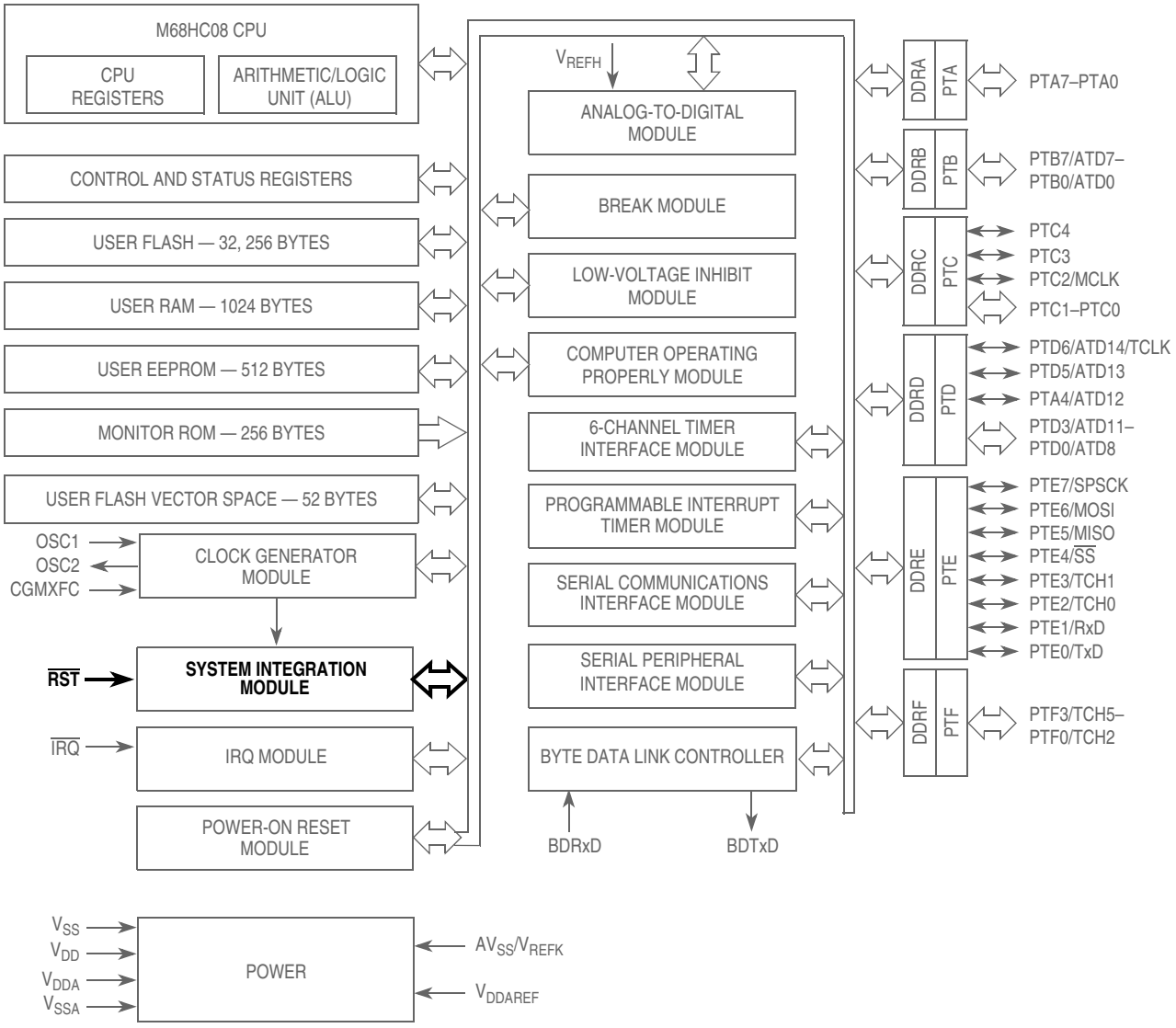


Figure 15-1. Block Diagram Highlighting SIM Block and Pins

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 15-3. This clock can come from either an external oscillator or from the on-chip phase-lock loop (PLL). See Chapter 5 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. See Chapter 5 Clock Generator Module (CGM).

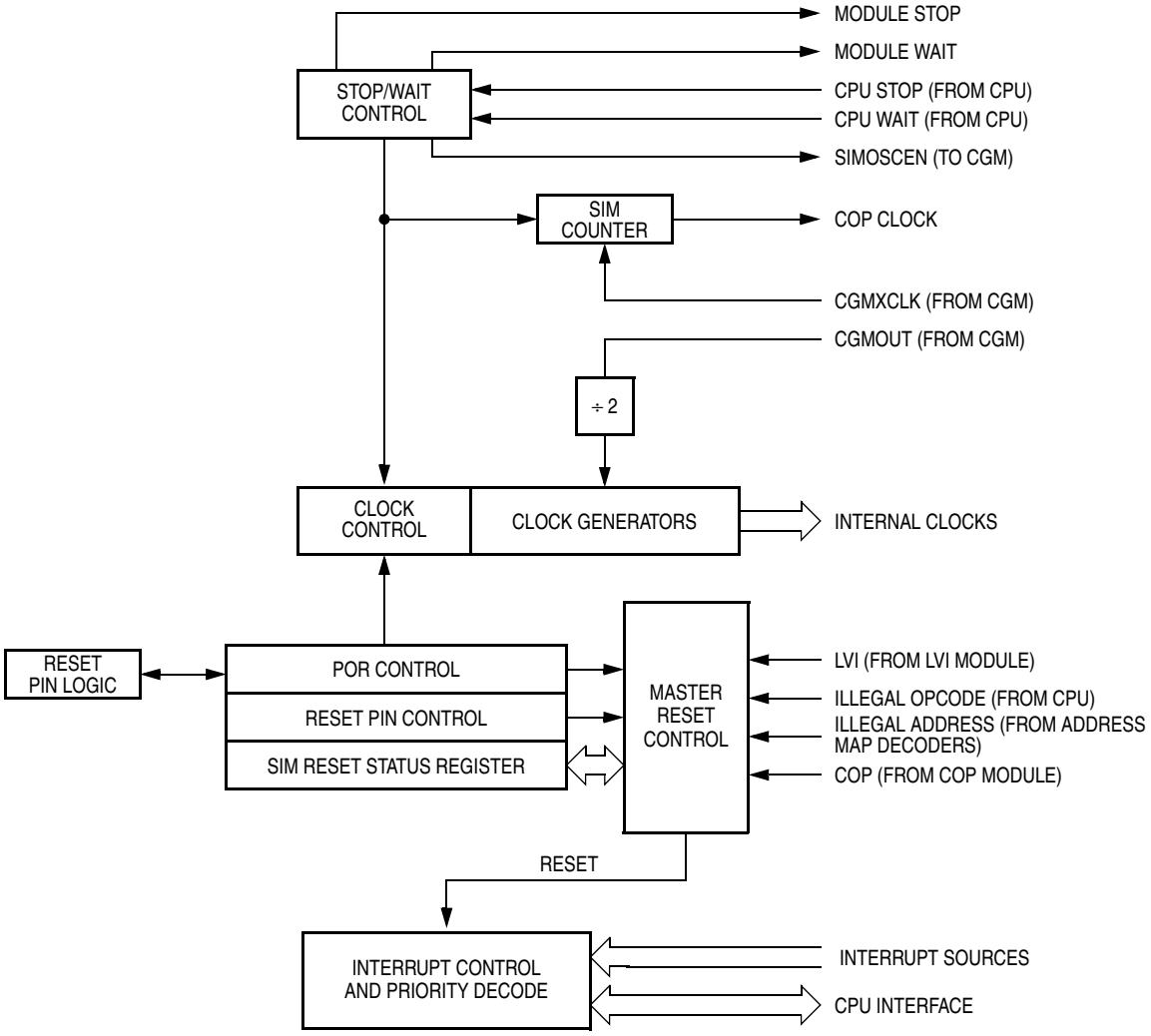


Figure 15-2. SIM Block Diagram

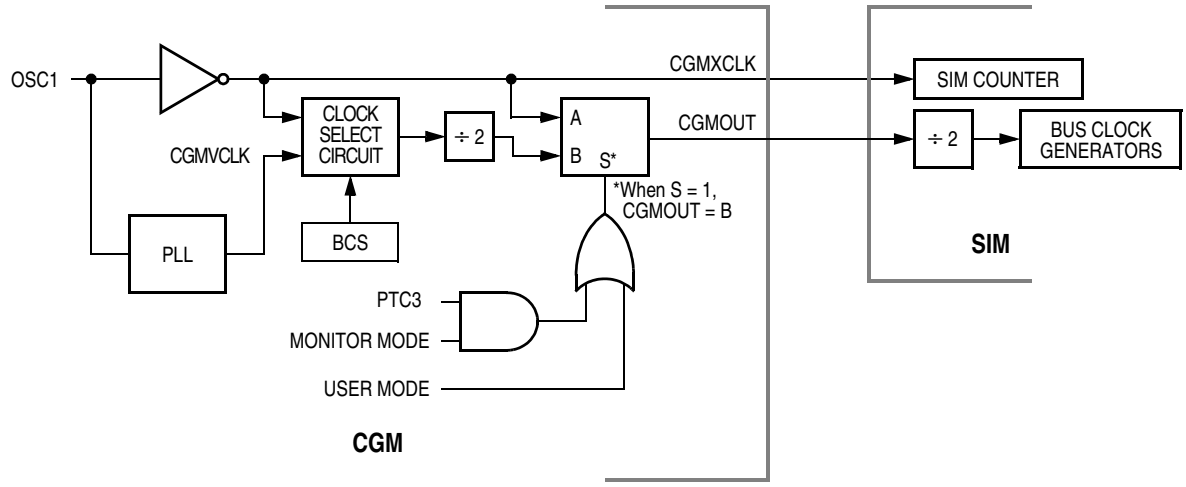


Figure 15-3. CGM Clock Signals

15.2.2 Clock Startup 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 4096 CGMXCLK cycles. The $\overline{RST}$ 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 Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. See 15.6.2 Stop Mode.

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.

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 (see 15.4 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR) (see 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 15-4 shows the relative timing.

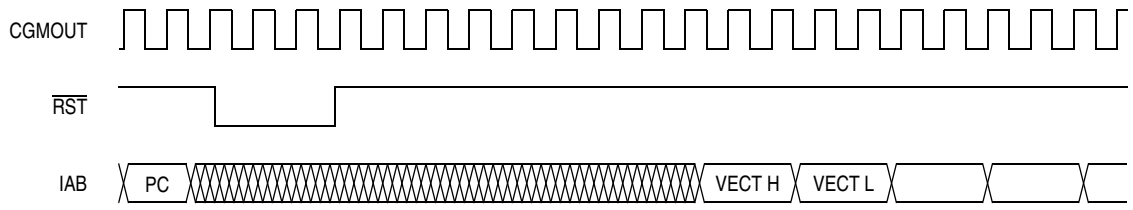


Figure 15-4. External Reset Timing

15.3.2 Active Resets from Internal Sources

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see Figure 15-5). An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR (see Figure 15-6). Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the $\overline{\text{RST}}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ shown in Figure 15-5.

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.

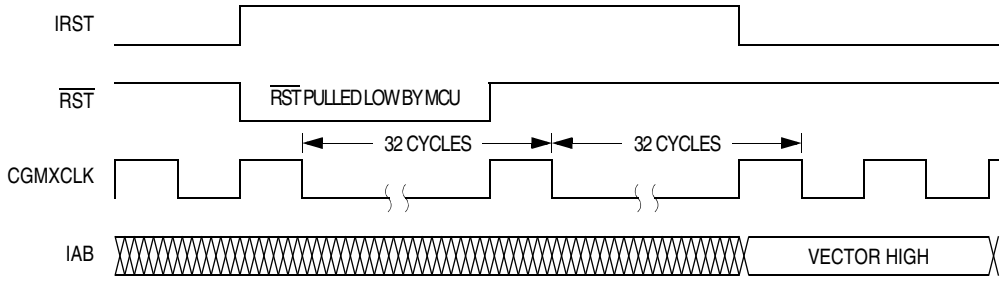


Figure 15-5. Internal Reset Timing

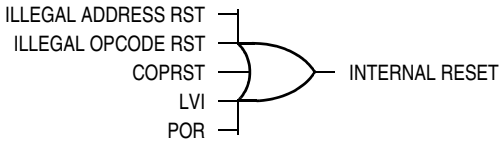


Figure 15-6. Sources of Internal Reset

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 sixty-four CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur. See Figure 15-7.

At power-on, the following events occur:

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

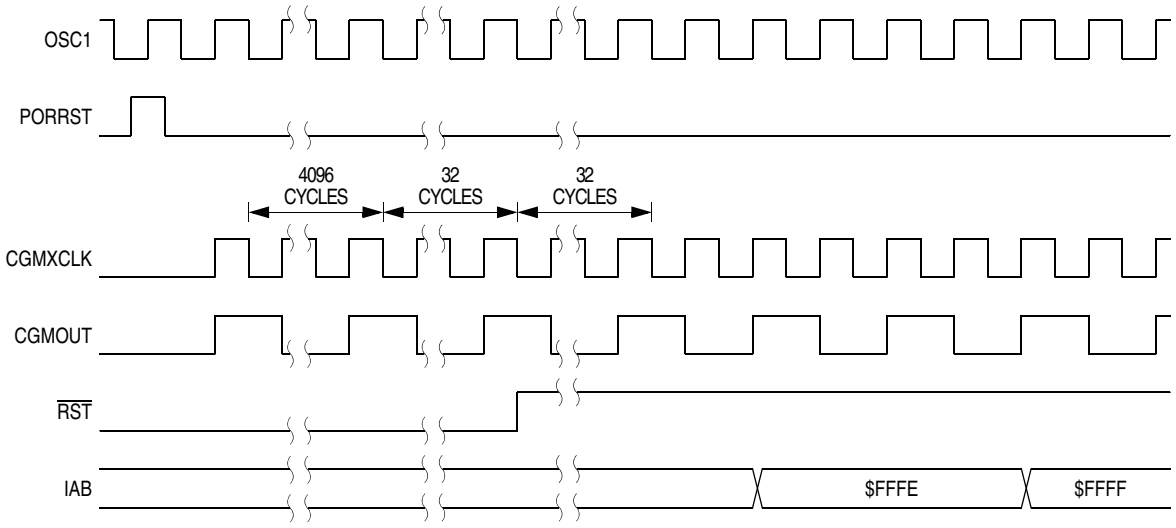


Figure 15-7. POR Recovery

15.3.2.2 Computer Operating Properly (COP) Reset

The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR) if the COPD bit in the CONFIG1 register is at 0. See Chapter 8 Computer Operating Properly (COP).

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 CONFIG1 register is 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset.

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 $\overline{RST}$ pin for all internal reset sources.

WARNING

Extra care should be exercised if code in this part has been migrated from older HC08 devices since the illegal address reset specification may be different. Also, extra care should be exercised when using this emulation part for development of code to be run in ROM AZ, AB or AS Family parts with a smaller memory size since some legal addresses will become illegal addresses on the smaller ROM memory map device and may as a result generate unwanted resets.

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 V_{LVII} voltage. The LVI bit in the SIM reset status register (SRSR) is set and a chip reset is asserted if the LVIPWRD and LVIRSTD bits in the CONFIG1 register are at 0. The $\overline{RST}$ pin will be held low until the SIM counts 4096 CGMXCLK cycles after V_{DD} rises above V_{LVIR}. Another sixty-four CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. See Chapter 11 Low-Voltage Inhibit (LVI).

15.4 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly module (COP). The SIM counter overflow supplies the clock for the COP module. The SIM counter is 12 bits long and is clocked by the falling edge of CGMXCLK.

15.4.1 SIM Counter During Power-On Reset

The power-on reset module (POR) detects power applied to the MCU. At power-on, the POR circuit asserts the signal PORRST. Once the SIM is initialized, it enables the clock generation module (CGM) to drive the bus clock state machine.

15.4.2 SIM Counter During Stop Mode Recovery

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the CONFIG1 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 start-up times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

15.4.3 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.

15.5 Program Exception Control

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

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

15.5.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 15-8 shows interrupt entry timing. 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 can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared), see Figure 15-9.

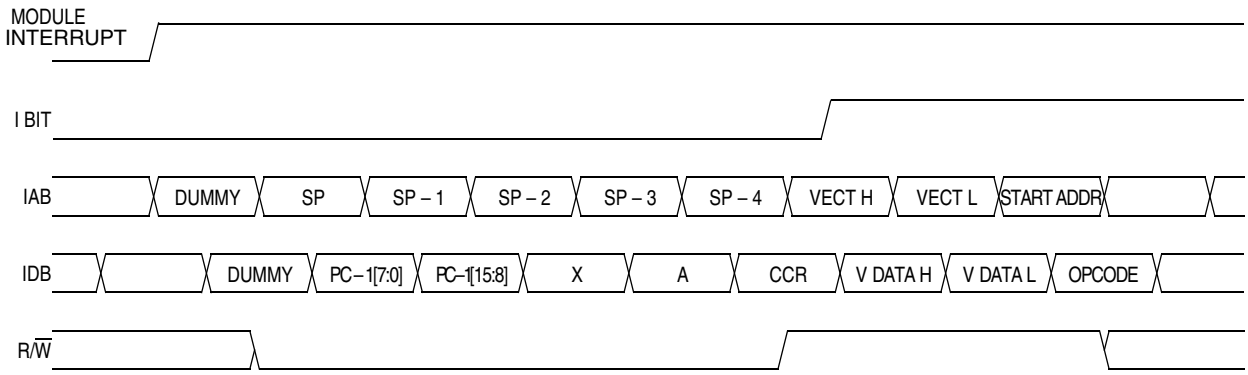


Figure 15-8. Interrupt Entry

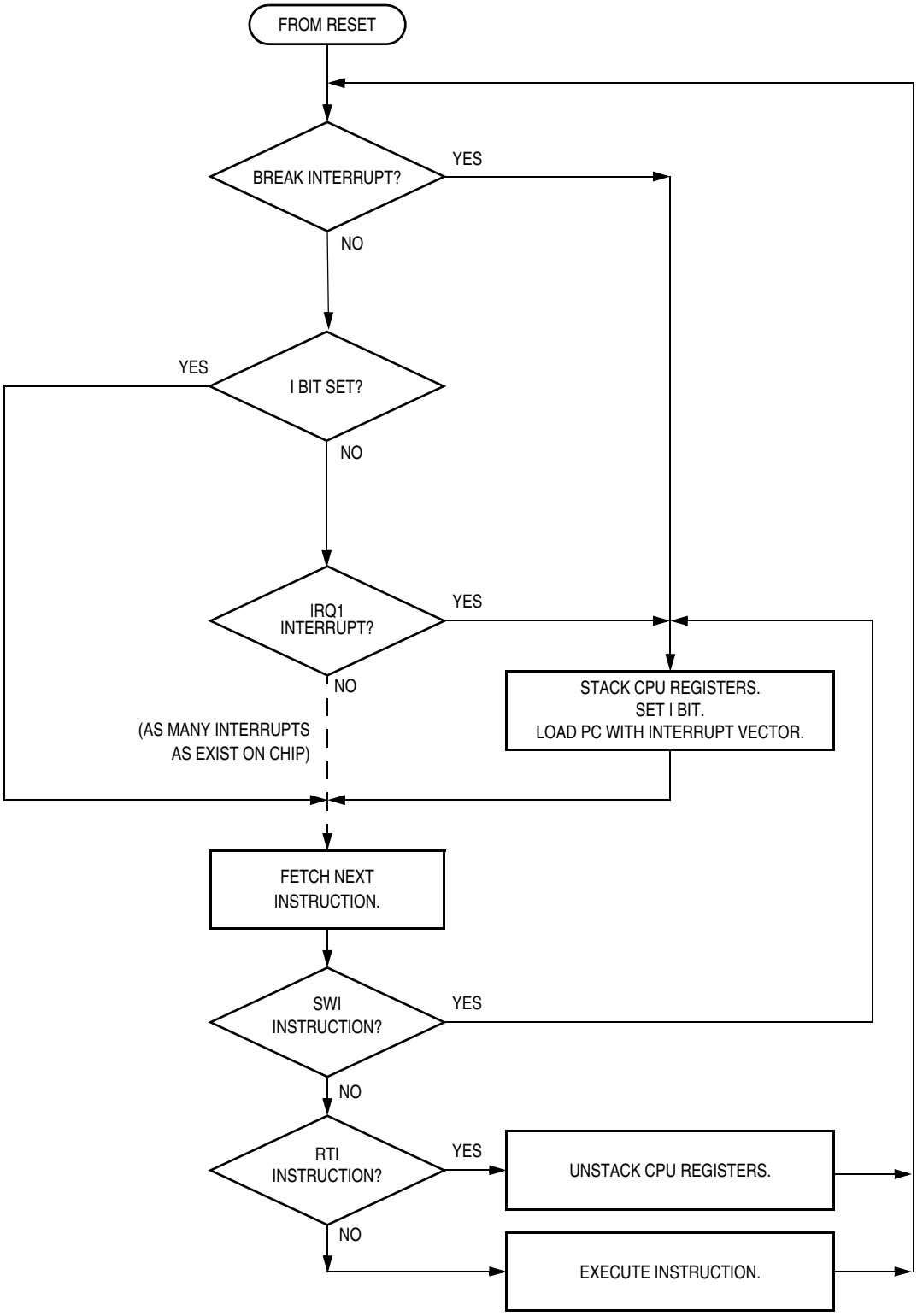


Figure 15-9. Interrupt Processing

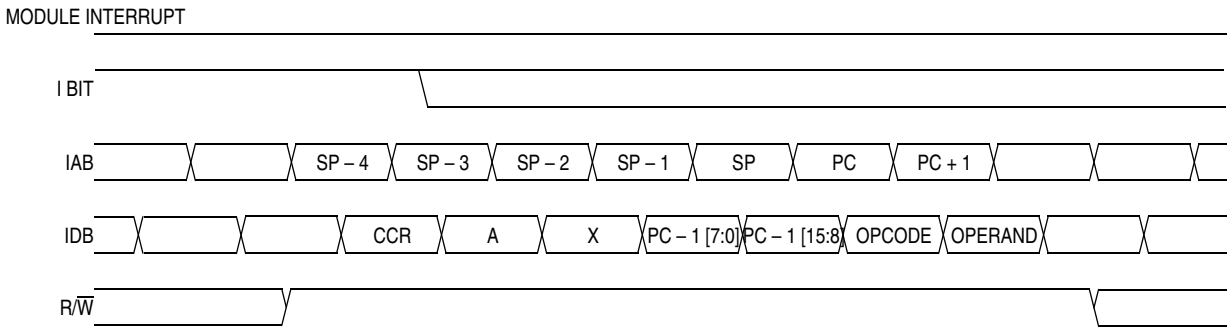


Figure 15-10. Interrupt Recovery

Hardware Interrupts

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

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 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.

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 M68HC05, M6805 and M146805 Families 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.

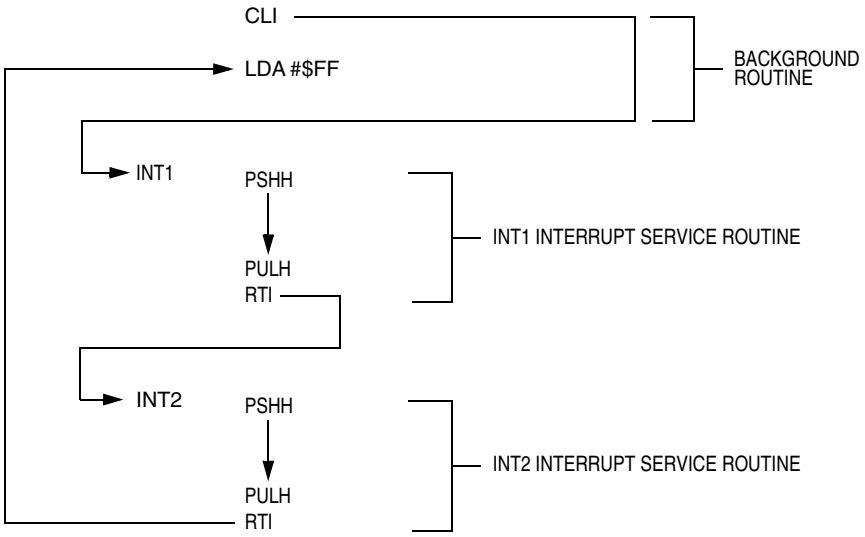


Figure 15-11. Interrupt Recognition Example

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.*

15.5.2 Reset

All reset sources always have higher priority than interrupts and cannot be arbitrated.

15.5.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See 18.2 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.

15.5.4 Status Flag Protection in Break Mode

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

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

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

15.6 Low-Power Modes

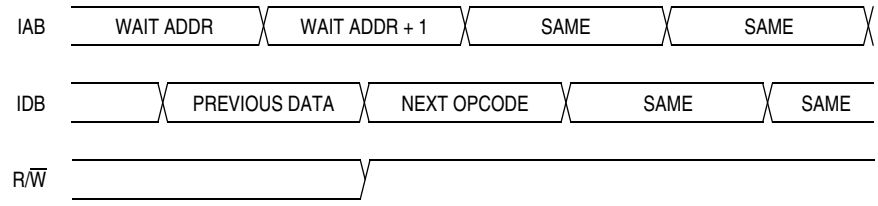
Executing the WAIT or STOP instruction puts the MCU in a low power- consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing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

15.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while one set of peripheral clocks continue to run. Figure 15-12 shows the timing for wait mode entry.

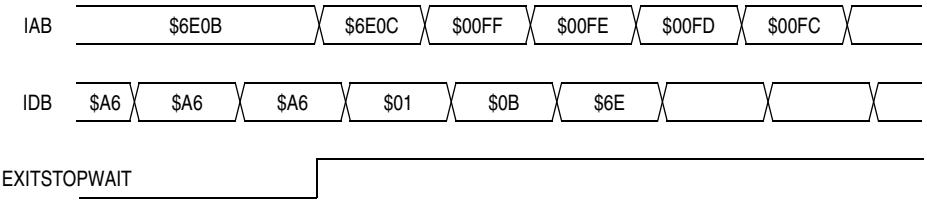
A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. Refer to the wait mode subsection of each module to see if the module is active or inactive 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 configuration register is 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.



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

Figure 15-12. Wait Mode Entry Timing



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

Figure 15-13. Wait Recovery from Interrupt or Break

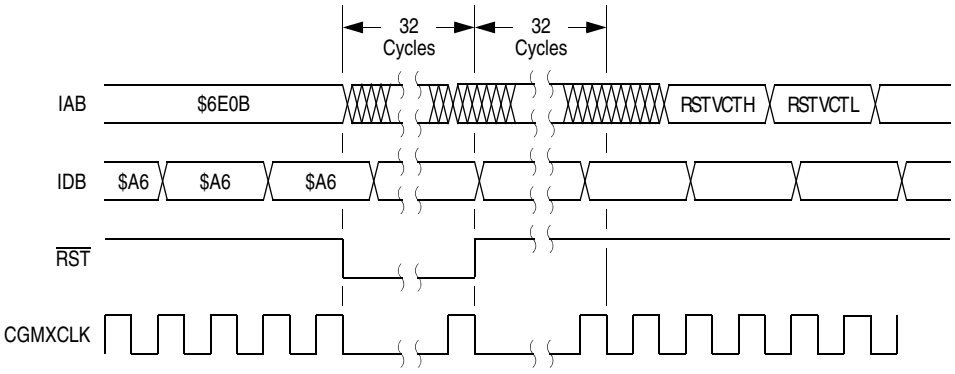


Figure 15-14. Wait Recovery from Internal Reset

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 configuration register (CONFIG1). If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long startup times from stop mode.

NOTE

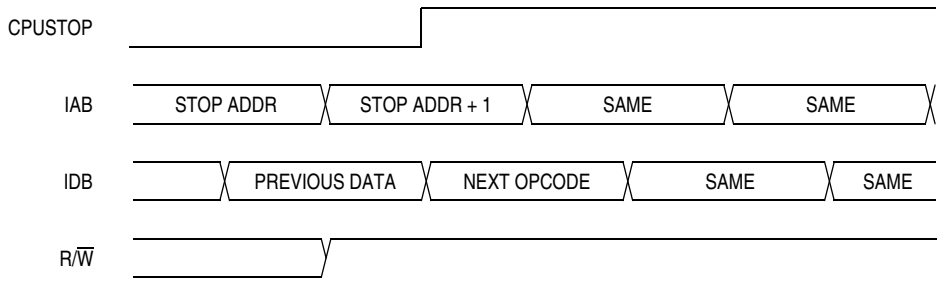
External crystal applications should use the full stop recovery time by clearing the SSREC bit.

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

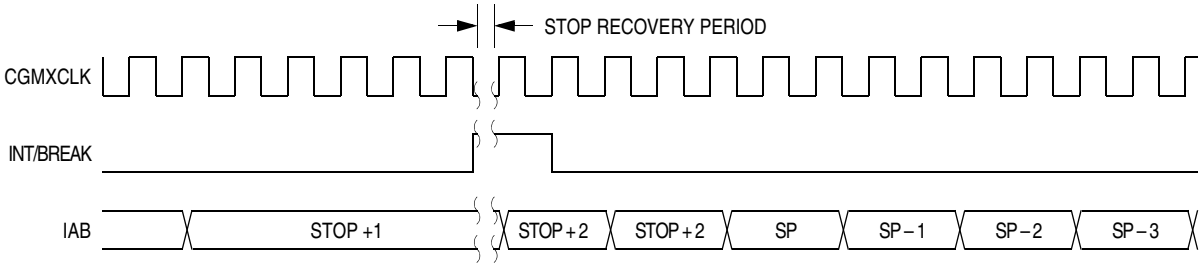


Figure 15-16. Stop Mode Recovery from Interrupt or Break

15.7 SIM Registers

The SIM has three memory mapped registers.

15.7.1 SIM Break Status Register

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

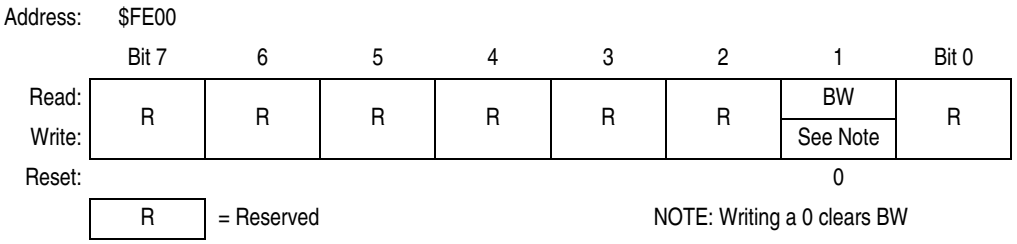


Figure 15-17. SIM Break Status Register (SBSR)

BW — SIM Break Wait

This status bit is useful in applications requiring a return to wait mode after exiting from a break interrupt. Clear BW by writing a 0 to it. Reset clears BW.

1 = Wait mode was exited by break interrupt
0 = Wait mode was not exited by break interrupt

15.7.2 SIM Reset Status Register

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.

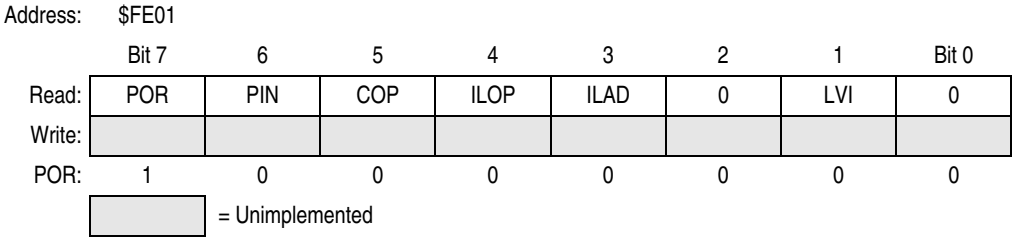


Figure 15-18. SIM Reset Status Register (SRSR)

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 ($\overline{RST}$)
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

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 (SPI) module, which allows full-duplex, synchronous, serial communications with peripheral devices.

16.2 Features

Features of the SPI module include:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency
- Serial clock with programmable polarity and phase
- Two separately enabled 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

16.3 Pin Name and Register Name Conventions

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 Name	MISO	MOSI	$\overline{SS}$	SPSCK
Full SPI Pin Name	PTE5/MISO	PTE6/MOSI	PTE4/$\overline{SS}$	PTE7/SPSCK

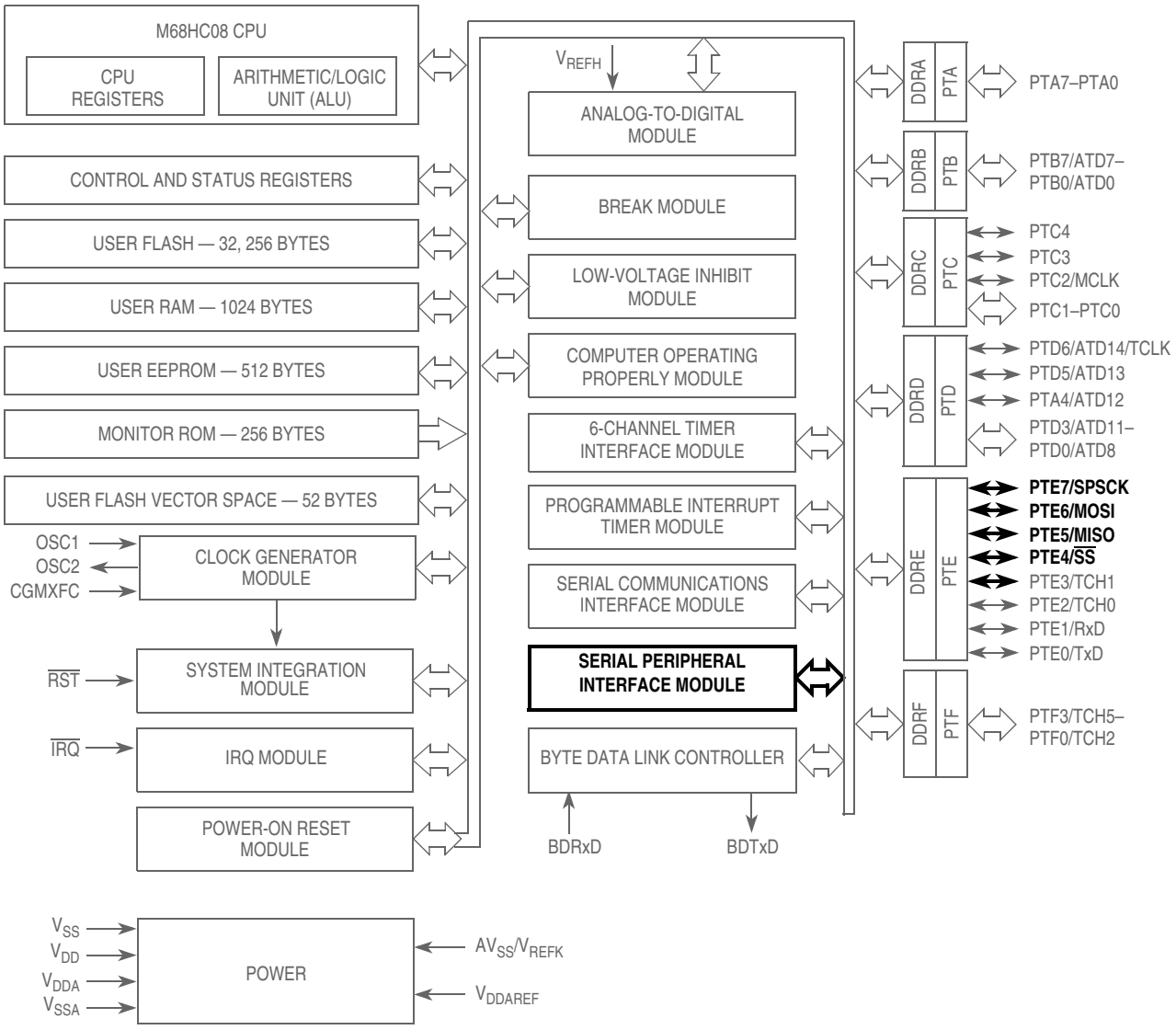


Figure 16-1. Block Diagram Highlighting SPI Block and Pins

The generic names of the SPI I/O registers are:

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

16.4 Functional Description

Figure 16-2 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.

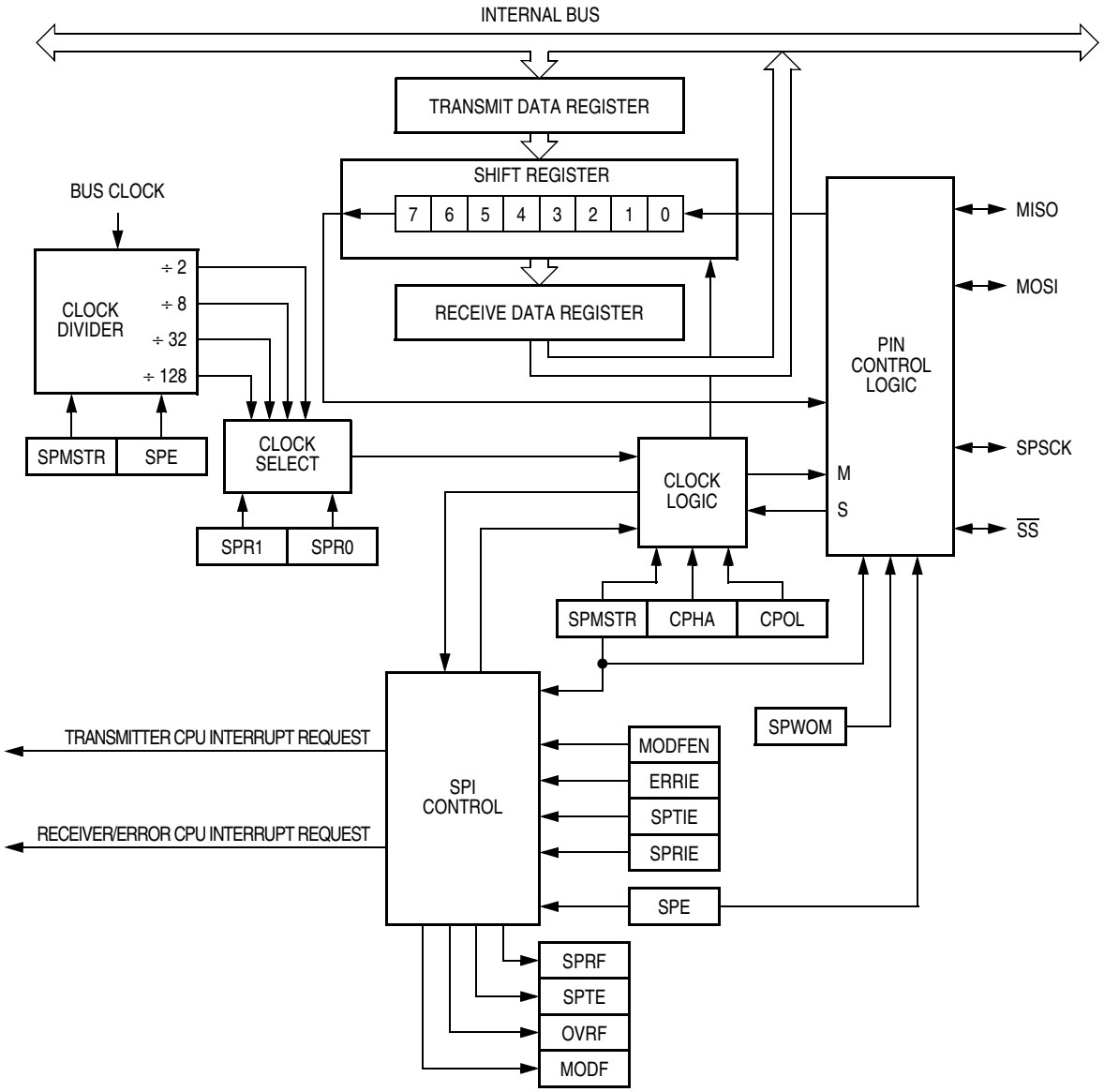


Figure 16-2. SPI Module Block Diagram

16.4.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR (SPCR $0010), 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 16.13.1 SPI Control Register.

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE (SPSCR $0011). The byte begins shifting out on the MOSI pin under the control of the serial clock. See Table 16-2.

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register (see 16.13.2 SPI Status and Control Register). Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

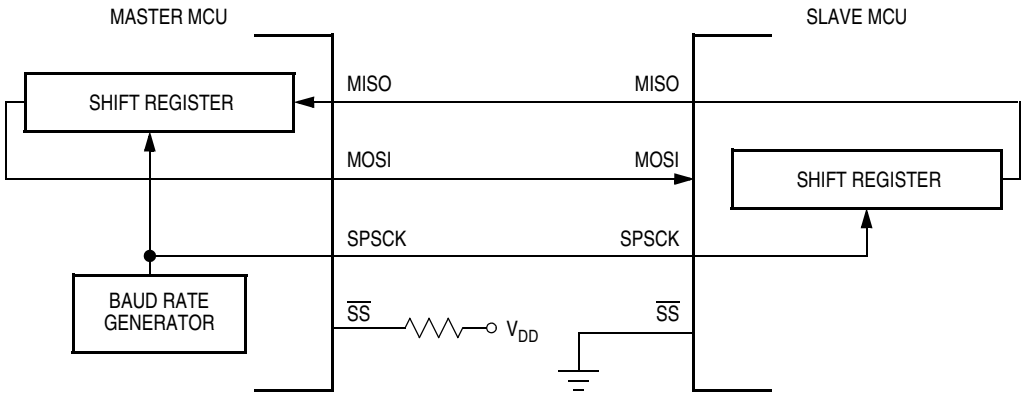


Figure 16-3. Full-Duplex Master-Slave Connections

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 (SPSCR), becomes set. At the same time that SPRF becomes set, the byte from the slave transfers to the receive data register. In normal operation, SPRF signals the end of a transmission. Software clears SPRF by reading the SPI status and control register and then reading the SPI data register. Writing to the SPI data register clears the SPTIE bit.

16.4.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit (SPCR, $0010) 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 low. $\overline{SS}$ must remain low until the transmission is complete. See 16.6.2 Mode Fault Error.

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

To prevent SPSCK from appearing as a clock edge, SPSCK must be in the proper idle state before the slave is enabled.

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 be used optionally 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 (SPCR) 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 (SPCR), disable the SPI by clearing the SPI enable bit (SPE).

16.5.2 Transmission Format When CPHA = 0

Figure 16-4 shows an SPI transmission in which CPHA (SPCR) 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 16.6.2 Mode Fault Error). When CPHA = 0, the first SPSCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the transmission. The $\overline{SS}$ pin must be toggled high and then low again between each byte transmitted.

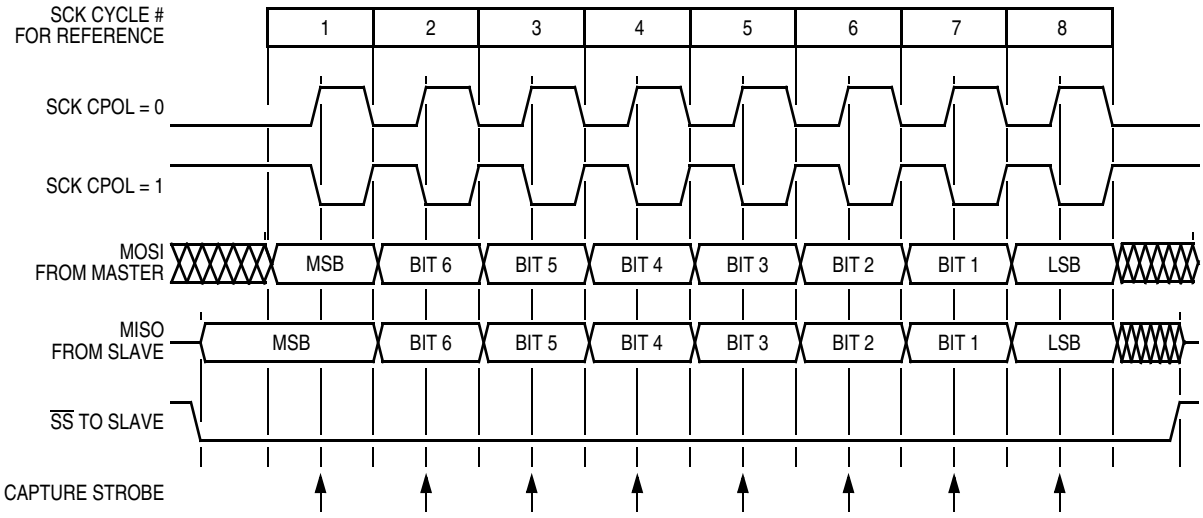


Figure 16-4. Transmission Format (CPHA = 0)

16.5.3 Transmission Format When CPHA = 1

Figure 16-5 shows an SPI transmission in which CPHA (SPCR) 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 16.6.2 Mode Fault Error). When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

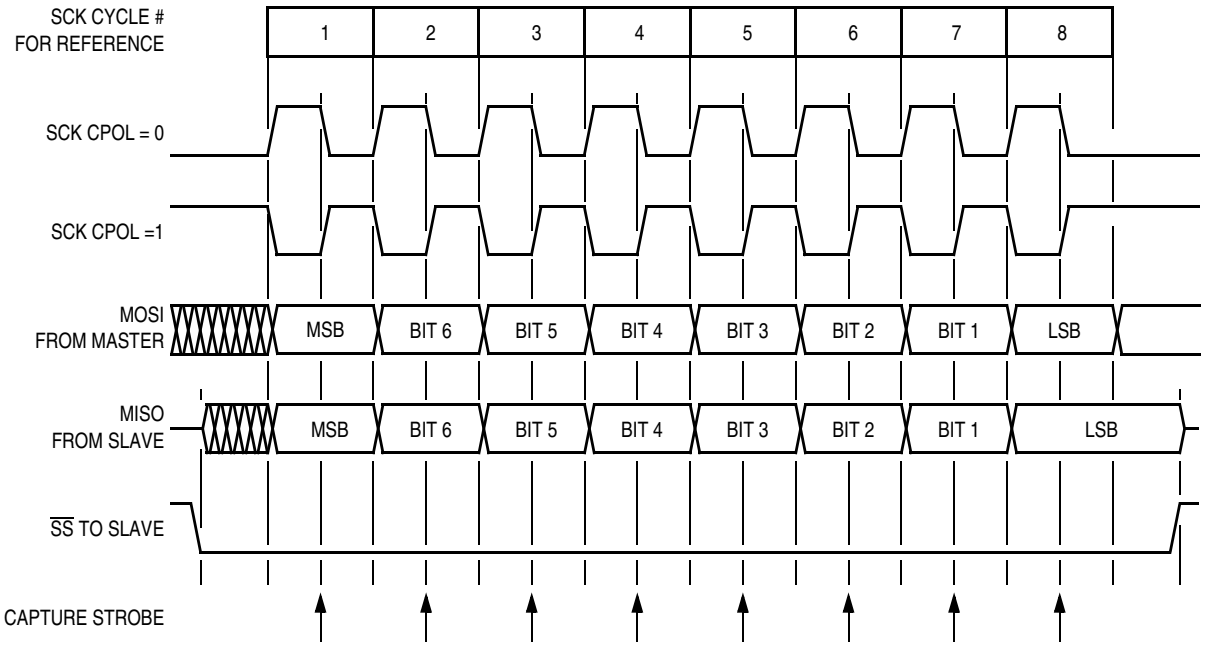


Figure 16-5. Transmission Format (CPHA = 1)

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 ($0012). 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 16-6). 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 (SPCR) are set to conserve power. SCK edges occur half way 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 16-6. This

delay will be no longer than a single SPI bit time. That is, the maximum delay between the write to SPDR and the start of the SPI transmission is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

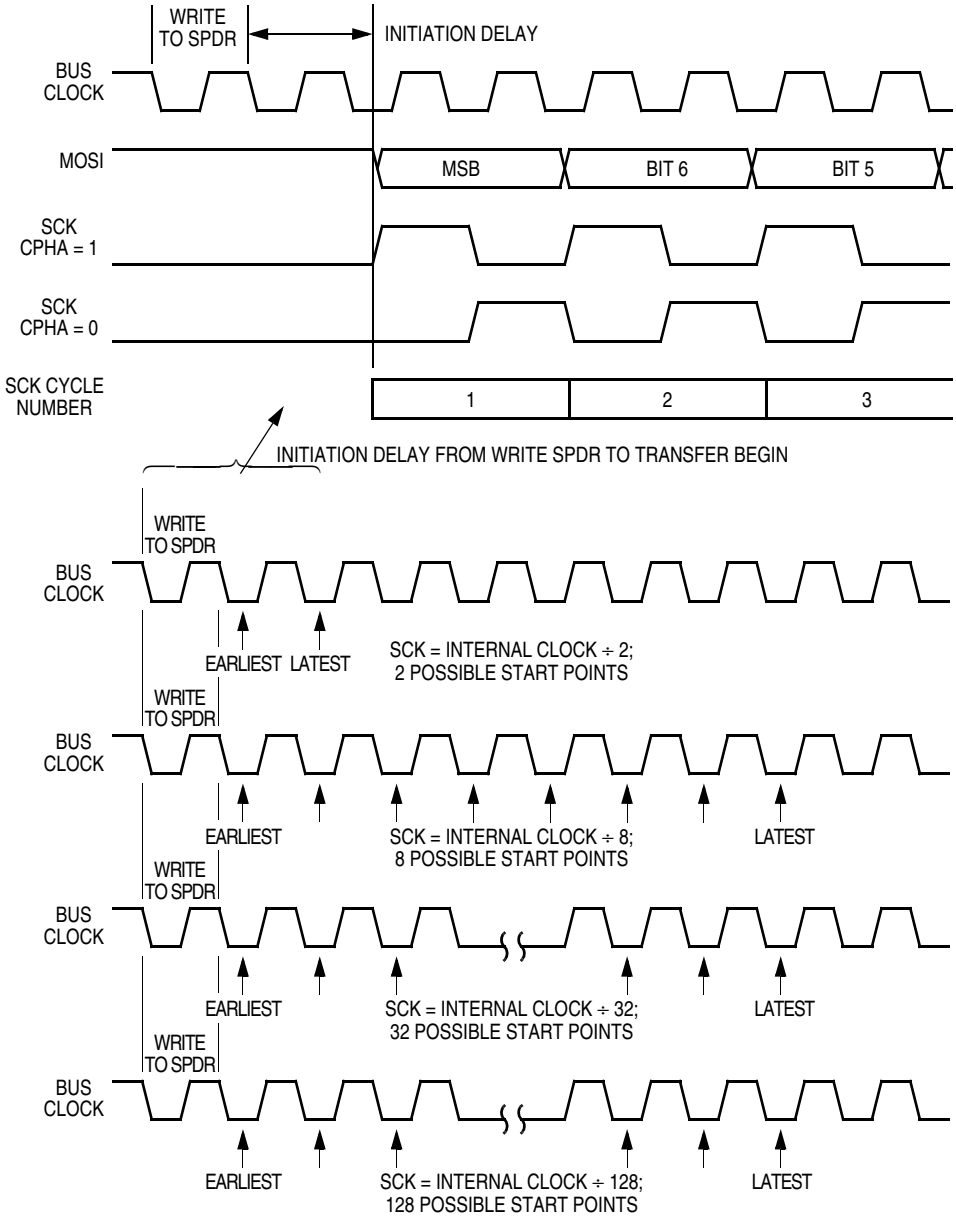


Figure 16-6. Transmission Start Delay (Master)

16.6 Error Conditions

Two flags signal SPI error conditions:

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

16.6.1 Overflow Error

The overflow flag (OVRF in SPSCR) 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 16-4 and Figure 16-5.) 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 in SPSCR) is also set. MODF and OVRF can generate a receiver/error CPU interrupt request (see Figure 16-9). 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 16-7 shows how it is possible to miss an overflow.

The first part of Figure 16-7 shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in between the time that SPSCR and SPDR are read.

In this case, an overflow can be easily missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR after the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will complete with an SPRF interrupt. Figure 16-8 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit (SPSCR).

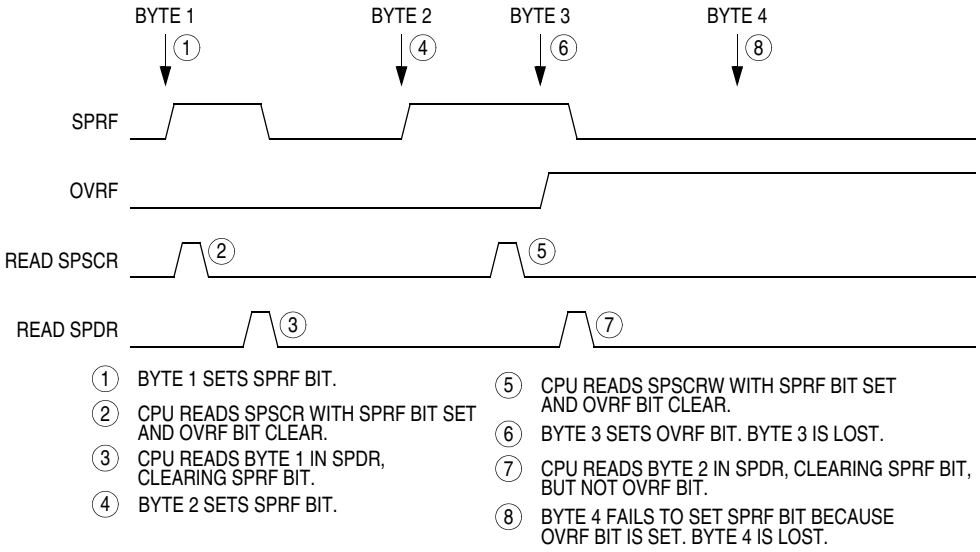


Figure 16-7. Missed Read of Overflow Condition

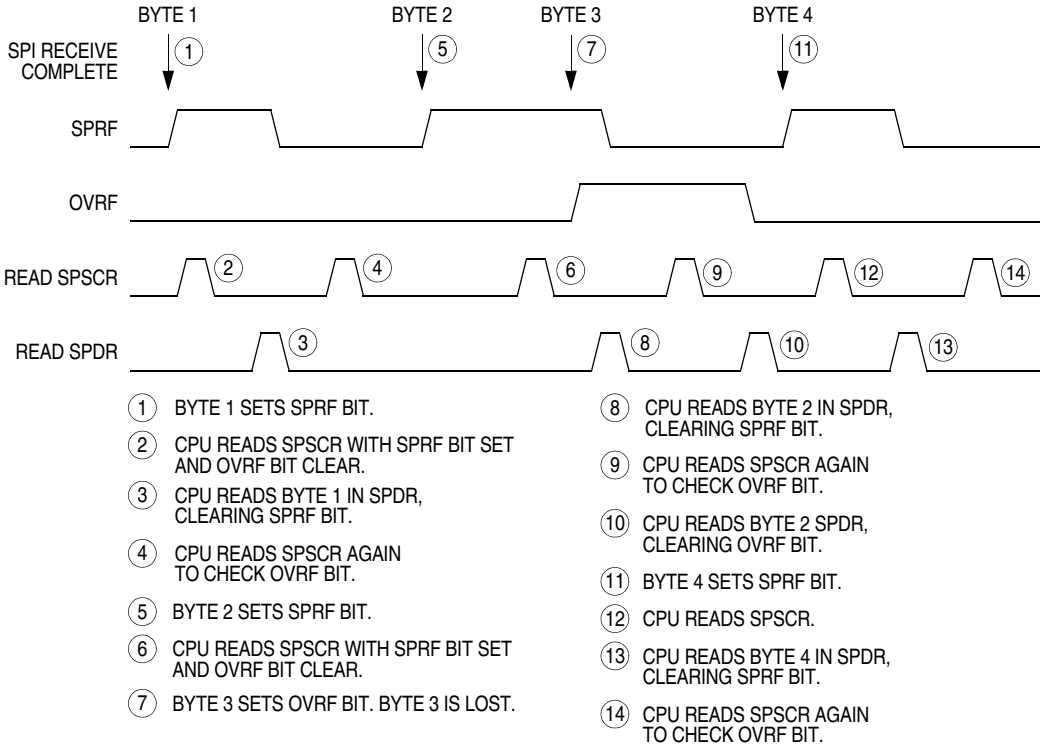


Figure 16-8. Clearing SPRF When OVRF Interrupt Is Not Enabled

16.6.2 Mode Fault Error

For the MODF flag (in SPSCR) to be set, the mode fault error enable bit (MODFEN in SPSCR) 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 in SPSCR) 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 16-9). 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 $\overline{SS}$ is 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 returns to its idle level after 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 after 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 $\overline{SS}$ is low 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 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if 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.

16.7 Interrupts

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

Table 16-2. SPI Interrupts

Flag	Request
SPTE (transmitter empty)	SPI transmitter CPU interrupt request (SPTIE = 1)
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, 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.

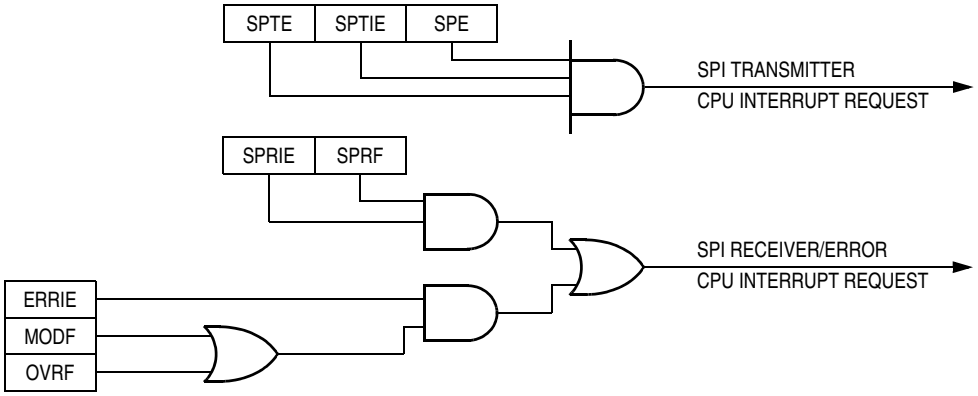


Figure 16-9. SPI Interrupt Request Generation

Two sources in the SPI status and control register can generate CPU interrupt requests:

1. 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.
2. 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.

16.8 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 in SPSCR) 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 16-10 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA:CPOL = 1:0).

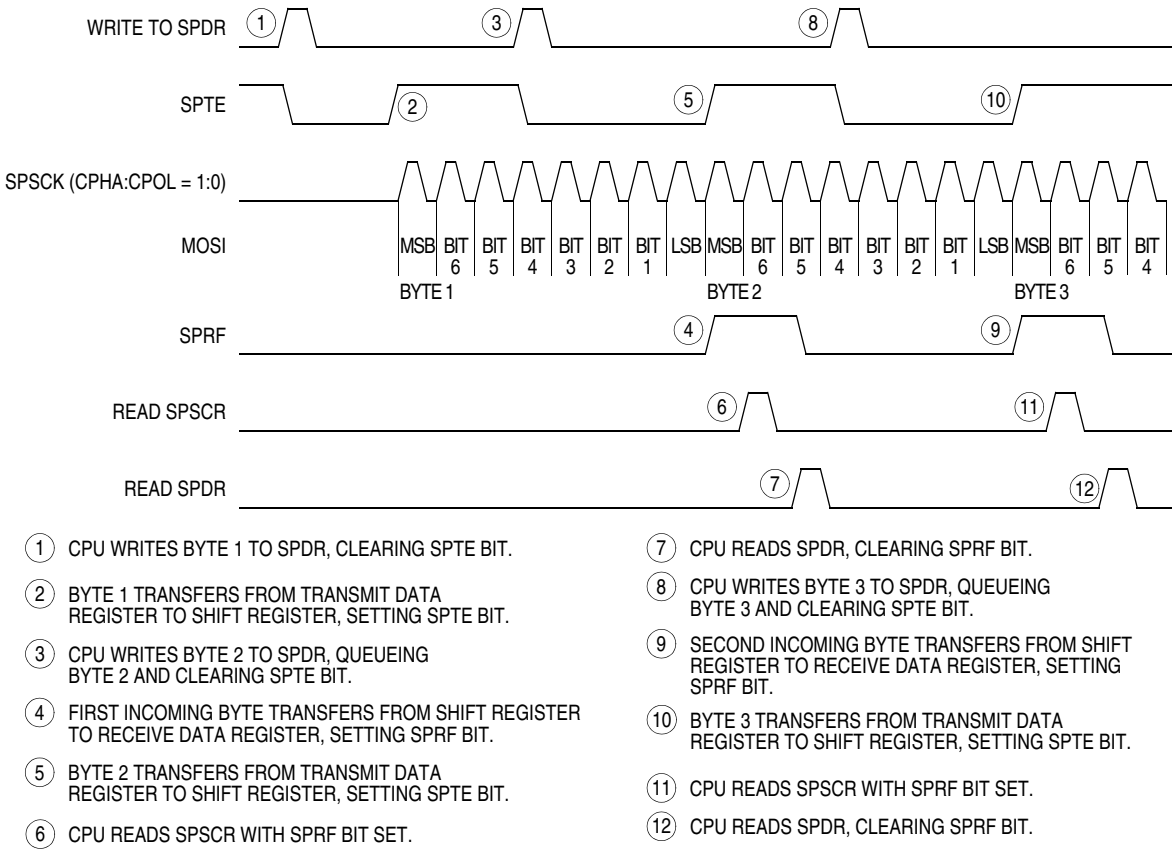


Figure 16-10. SPRF/SPTE CPU Interrupt Timing

For a slave, the transmit data buffer allows back-to-back transmissions to occur without the slave having to 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.

16.9 Resetting the SPI

Any system reset completely resets the SPI. Partial reset occurs 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 additional 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 reset all control bits when SPE is set back to 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 also can be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

16.10 Low-Power Modes

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

16.10.1 Wait Mode

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

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

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). See 16.7 Interrupts.

16.10.2 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 the MCU exits stop mode. If stop mode is exited by reset, any transfer in progress is aborted and the SPI is reset.

16.11 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, $FE03) enables software to clear status bits during the break state. See 15.7.3 SIM Break Flag Control Register.

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.

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.

16.12 I/O Signals

The SPI module has four I/O pins and shares three 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}.

16.12.1 MISO (Master In/Slave Out)

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

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is 0 and its $\overline{SS}$ pin is low. 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.

16.12.2 MOSI (Master Out/Slave In)

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

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.

16.12.3 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.

16.12.4 $\overline{SS}$ (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 16-11.

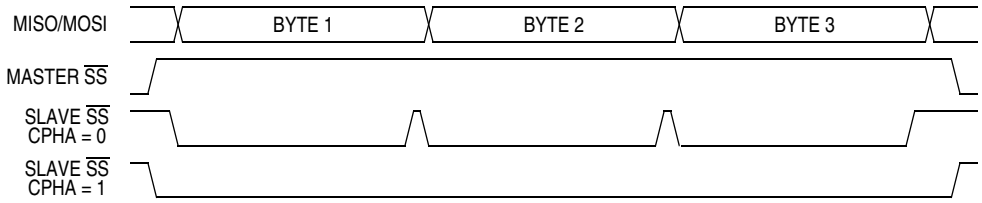


Figure 16-11. CPHA/$\overline{SS}$ Timing

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{SS}$ from creating a MODF error. See 16.13.2 SPI Status and Control Register.

NOTE

A high voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if a transmission already has 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 16.6.2 Mode Fault Error). For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the 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

16.12.5 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.

16.13 I/O Registers

Three registers control and monitor SPI operation:

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

16.13.1 SPI Control Register

The SPI control register:

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

Address: $0010

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

R = Reserved

Figure 16-12. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

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

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

SPMSTR — SPI Master Bit

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

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See Figure 16-4 and Figure 16-5.) To transmit data between SPI modules, the SPI modules must have identical CPOL bits. Reset clears the CPOL bit.

CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See Figure 16-4 and Figure 16-5.) 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 1 between bytes. (See Figure 16-11). 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 16.6.2 Mode Fault Error). A 1 on the $\overline{SS}$ pin does not in any way affect the state of the SPI state machine.

SPWOM — SPI Wired-OR Mode Bit

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

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

SPE — SPI Enable Bit

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI (see 16.9 Resetting the SPI). Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE — SPI Transmit Interrupt Enable Bit

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

16.13.2 SPI Status and Control Register

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 these 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 16-13. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

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

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 Bit

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 Bit

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

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

This clearable, read-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 Bit

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

NOTE

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

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

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

MODFEN — Mode Fault Enable Bit

This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the $\overline{SS}$ pin is available as a general-purpose I/O.

If the MODFEN bit is set, then this pin is not available as a general purpose I/O. When the SPI is enabled as a slave, the $\overline{SS}$ pin is not available as a general-purpose I/O regardless of the value of MODFEN. See 16.12.4 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 16.6.2 Mode Fault Error.

SPR1 and SPR0 — SPI Baud Rate Select Bits

In master mode, these read/write bits select one of four baud rates as shown in Table 16-4. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 16-4. SPI Master Baud Rate Selection

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

Use this formula to calculate the SPI baud rate:

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

where:

CGMOUT = base clock output of the clock generator module (CGM), see Chapter 5 Clock Generator Module (CGM).
BD = baud rate divisor

16.13.3 SPI Data Register

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 16-2.

Address: $0012

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

Figure 16-14. SPI Data Register (SPDR)

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

NOTE

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

Chapter 17
Timer Interface Module (TIM)

17.1 Introduction

This section describes the timer interface module (TIM). The TIM 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 TIM.

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

17.2 Features

Features 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 TIM clock input
  - Seven frequency internal bus clock prescaler selection
  - External TIM clock input (4 MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIM counter stop and reset bits

17.3 Functional Description

Figure 17-2 shows the TIM structure. The central component of the TIM is the 16-bit TIM counter that can operate as a free-running counter or a modulo up-counter. The TIM counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH–TMODL, control the modulo value of the TIM counter. Software can read the TIM counter value at any time without affecting the counting sequence.

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

17.3.1 TIM Counter Prescaler

The TIM clock source can be one of the seven prescaler outputs or the TIM clock pin, PTD6/ATD14/TCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIM status and control register select the TIM clock source.

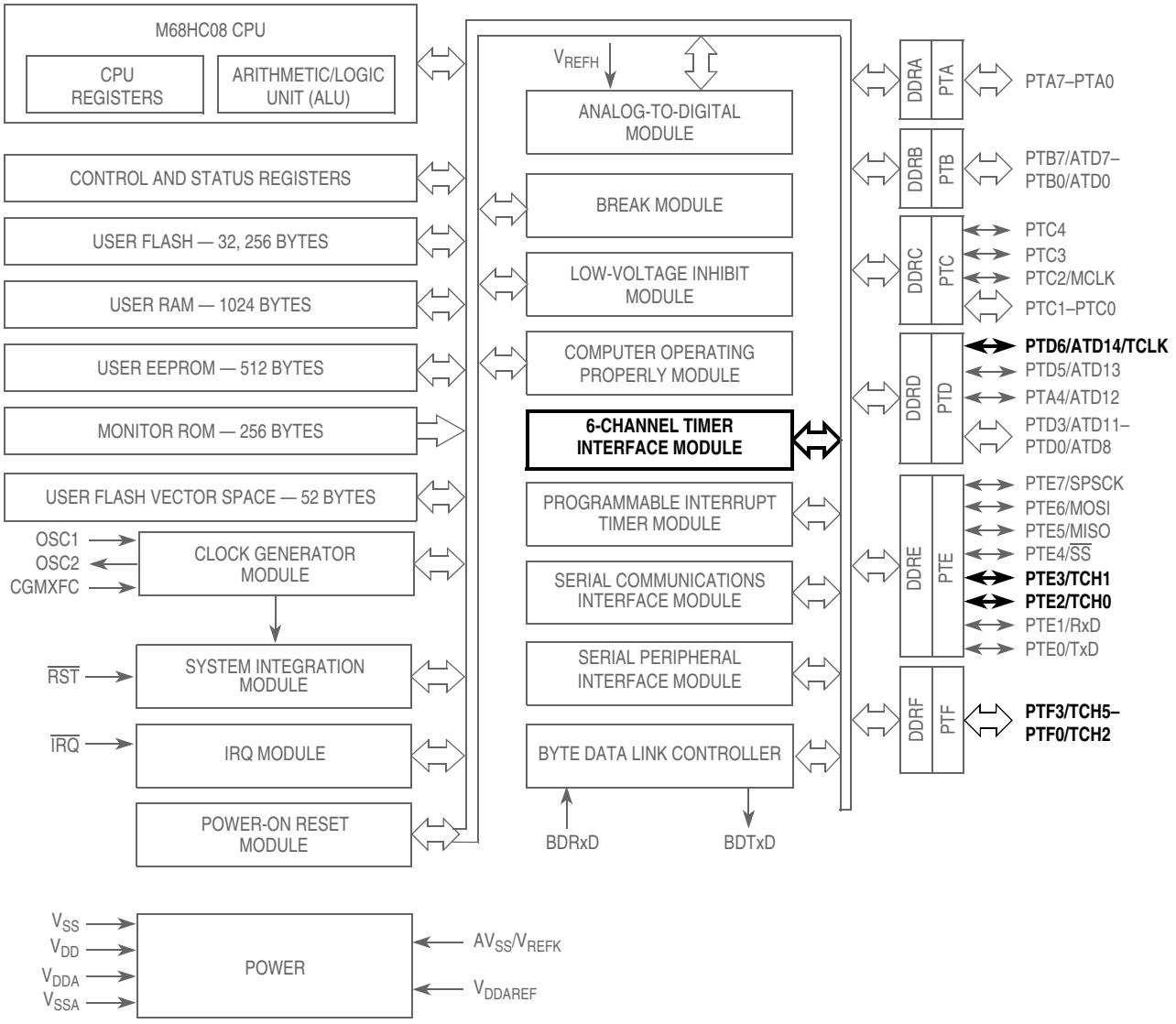


Figure 17-1. Block Diagram Highlighting TIM Block and Pins

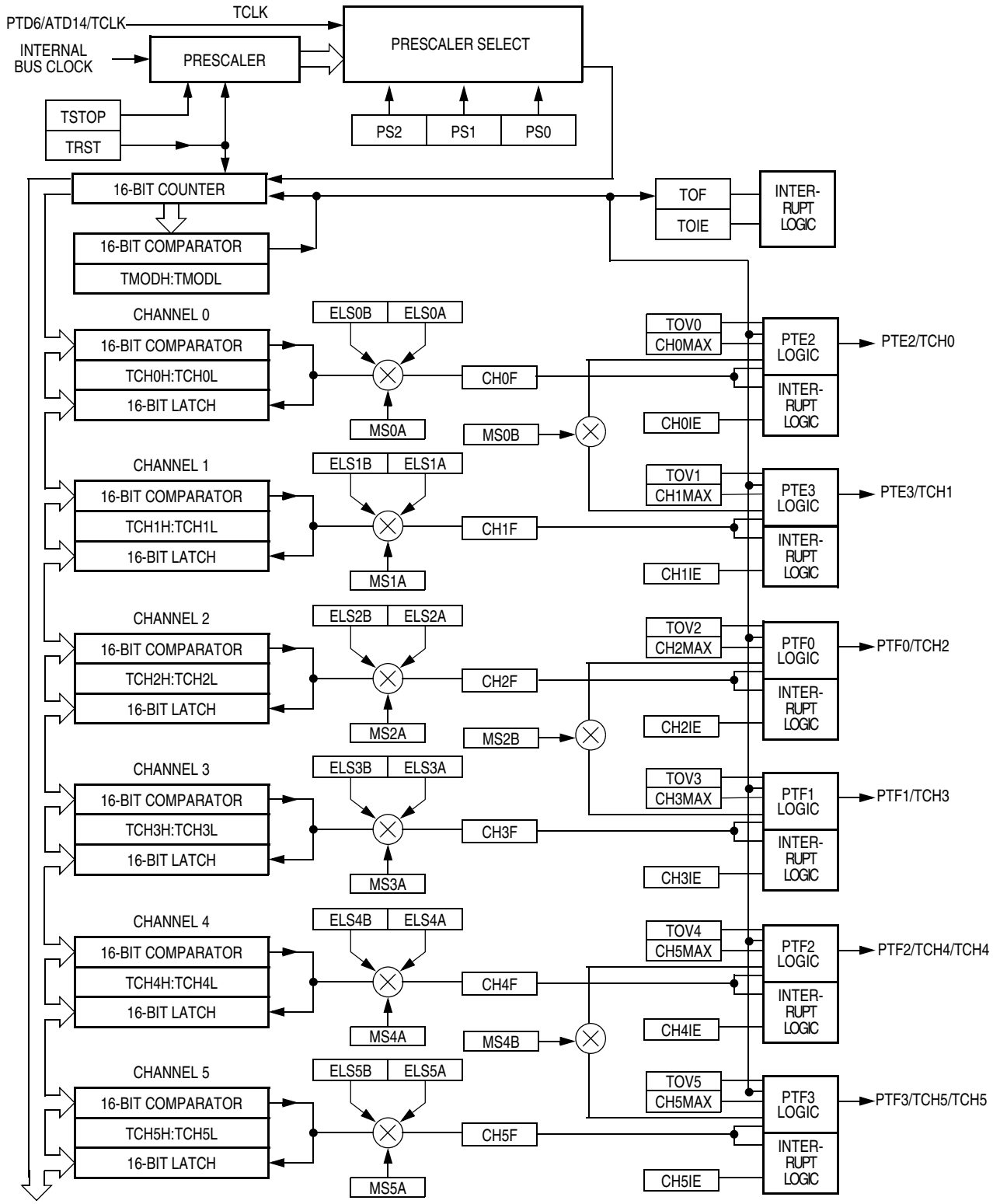


Figure 17-2. TIM Block Diagram

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 TSC0–TSC5 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIM latches the contents of the TIM counter into the TIM channel registers, TCHxH–TCHxL. Input captures can generate TIM 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 TIM channel register (TCHxH–TCHxL see 17.8.5 TIM Channel Registers) on each proper signal transition regardless of whether the TIM channel flag (CH0F–CH5F in TSC0–TSC5 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 TIM 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 TIM channel registers.

17.3.3 Output Compare

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

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 TIM channel registers.

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

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

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

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/TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The output compare value in the TIM channel 0 registers initially controls the output on the PTE2/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TCH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered output compare channel whose output appears on the PTF0/TCH2 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The output compare value in the TIM channel 2 registers initially controls the output on the PTF0/TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (2 or 3) that control the output are the ones written to last. TSC2 controls and monitors the buffered output compare function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TCH3, 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/TCH4 pin. The TIM channel registers of the linked pair alternately control the output.

Setting the MS4B bit in TIM channel 4 status and control register (TSC4) links channel 4 and channel 5. The output compare value in the TIM channel 4 registers initially controls the output on the PTF2/TCH4 pin. Writing to the TIM channel 5 registers enables the TIM channel 5 registers to synchronously control the output after the TIM overflows. At each subsequent overflow, the TIM channel registers (4 or 5) that control the output are the ones written to last. TSC4 controls and monitors the buffered output compare function and TIM channel 5 status and control register (TSC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3/TCH5, 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 TIM can generate a PWM signal. The value in the TIM counter modulo registers determines the period of the PWM signal. The channel pin toggles when the counter reaches the value in the TIM counter modulo registers. The time between overflows is the period of the PWM signal.

As Figure 17-3 shows, the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM to clear the channel pin on output compare if the state of the PWM pulse is 1. Program the TIM to set the pin if the state of the PWM pulse is 0.

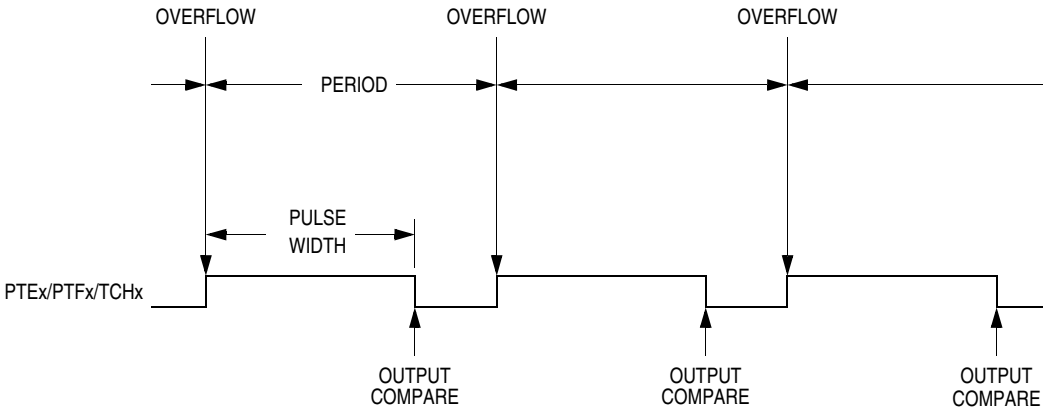


Figure 17-3. PWM Period and Pulse Width

The value in the TIM counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see 17.8.1 TIM Status and Control Register).

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

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 TIM channel registers.

An unsynchronized write to the TIM channel registers to change a pulse width value could cause incorrect operation for up to two PWM periods. For example, writing a new value before the counter reaches the old value but after the counter reaches the new value prevents any compare during that PWM period. Also, using a TIM overflow interrupt routine to write a new, smaller pulse width value may cause the compare to be missed. The TIM may pass the new value before it is written to the TIM 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 TIM overflow interrupts and write the new value in the TIM overflow interrupt routine. The TIM overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

NOTE

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

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/TCH0 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the PTE2/TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TCH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered PWM channel whose output appears on the PTF0/TCH2 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The TIM channel 2 registers initially control the pulse width on the PTF0/TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (2 or 3) that control the pulse width are the ones written to last. TSC2 controls and monitors the buffered PWM function and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TCH3, 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/TCH4 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS4B bit in TIM channel 4 status and control register (TSC4) links channel 4 and channel 5. The TIM channel 4 registers initially control the pulse width on the PTF2/TCH4 pin. Writing to the TIM channel 5 registers enables the TIM channel 5 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (4 or 5) that control the pulse width are the ones written to last. TSC4 controls and monitors the buffered PWM function and TIM channel 5 status and control register (TSC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3/TCH5, 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 TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter and prescaler by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH–TMODL) write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH–TCHxL) write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA (see Table 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 TIM status control register (TSC) clear the TIM stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIM channel 0 registers (TCH0H–TCH0L) initially control the buffered PWM output. TIM status control register 0 (TSC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIM channel 2 registers (TCH2H–TCH2L) initially control the buffered PWM output. TIM status control register 2 (TSC2) 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 TIM channel 4 registers (TCH4H–TCH4L) initially control the buffered PWM output. TIM status control register 4 (TSC4) 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 TIM overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output (see 17.8.4 TIM Channel Status and Control Registers).

17.4 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter reaches the modulo value programmed in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH5F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

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

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

17.5.2 Stop Mode

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

17.6 TIM During Break Interrupts

A break interrupt stops the TIM 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 Figure 15-19. 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 TIM. Port E shares two of its pins with the TIM and port F shares four of its pins with the TIM. PTD6/ATD14/TCLK is an external clock input to the TIM prescaler. The six TIM channel I/O pins are PTE2/TCH0, PTE3/TCH1, PTF0/TCH2, PTF1/TCH3, PTF2/TCH4, and PTF3/TCH5.

17.7.1 TIM Clock Pin (PTD6/ATD14/TCLK)

PTD6/ATD14/TCLK is an external clock input that can be the clock source for the TIM counter instead of the prescaled internal bus clock. Select the PTD6/ATD14/TCLK input by writing 1s to the three prescaler select bits, PS[2:0] (see 17.8.1 TIM 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/TCLK is available as a general-purpose I/O pin or ADC channel when not used as the TIM clock input. When the PTD6/ATD14/TCLK pin is the TIM clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

17.7.2 TIM Channel I/O Pins (PTF3/TCH5–PTF0/TCH2 and PTE3/TCH1–PTE2/TCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE2/TCH0, PTF0/TCH2, and PTF2/TCH4 can be configured as buffered output compare or buffered PWM pins.

17.8 I/O Registers

These I/O registers control and monitor TIM operation:

- TIM status and control register (TSC)
- TIM control registers (TCNTH–TCNTL)
- TIM counter modulo registers (TMODH–TMODL)
- TIM channel status and control registers (TSC0–TSC5)
- TIM channel registers (TCH0H–TCH0L through TCH5H–TCH5L)

17.8.1 TIM Status and Control Register

The TIM status and control register:

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

Address: $0020

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

= Unimplemented

Figure 17-4. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

This read/write flag is set when the TIM counter reaches the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TIM status and control register when TOF is set and then writing a 0 to TOF. If another TIM 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 = TIM counter has reached modulo value.
0 = TIM counter has not reached modulo value.

TOIE — TIM Overflow Interrupt Enable Bit

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

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

TSTOP — TIM Stop Bit

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

1 = TIM counter stopped
0 = TIM counter active

NOTE

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

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

1 = Prescaler and TIM counter cleared
0 = No effect

NOTE

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD6/ATD14/TCLK pin or one of the seven prescaler outputs as the input to the TIM counter as Table 17-1 shows. Reset clears the PS[2:0] bits.

Table 17-1. Prescaler Selection

PS[2:0]	TIM Clock Source
000	Internal bus clock ÷1
001	Internal bus clock ÷ 2
010	Internal bus clock ÷ 4
011	Internal bus clock ÷ 8
100	Internal bus clock ÷ 16
101	Internal bus clock ÷ 32
110	Internal bus clock ÷ 64
111	PTD6/ATD14/TCLK

17.8.2 TIM Counter Registers

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

NOTE

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

Register Name and Address: TCNTH — $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: TCNTL — $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 17-5. TIM Counter Registers (TCNTH and TCNTL)

17.8.3 TIM Counter Modulo Registers

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

Register Name and Address: TMODH — $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: TMODL — $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 17-6. TIM Counter Modulo Registers (TMODH and TMODL)

NOTE

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

17.8.4 TIM Channel Status and Control Registers

Each of the TIM 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 TIM overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Register Name and Address		Bit 7	6	5	4	3	2	1	Bit 0
TSC0 — $0026	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC1 — $0029	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC2 — $002C	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC3 — $002F	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC4 — $0032	Read:	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TSC5 — $0035	Read:	CH5F	CH5IE	0	MS5A	ELS5B	ELS5A	TOV5	CH5MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-7. TIM Channel Status and Control Registers (TSC0–TSC5)

CHxF — Channel x Flag Bit

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

When CHxIE = 1, clear CHxF by reading TIM 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 TIM 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 TIM channel 0, TIM channel 2 and TIM channel 4 status and control registers.

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

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

Setting MS4B disables the channel 5 status and control register and reverts TCH5 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 TCHx 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 TIM status and control register (TSC).

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port E or port F and pin PTEx/TCHx or pin PTFx/TCHx 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.

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

NOTE

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

TOVx — Toggle-On-Overflow Bit

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

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

NOTE

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 17-8 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

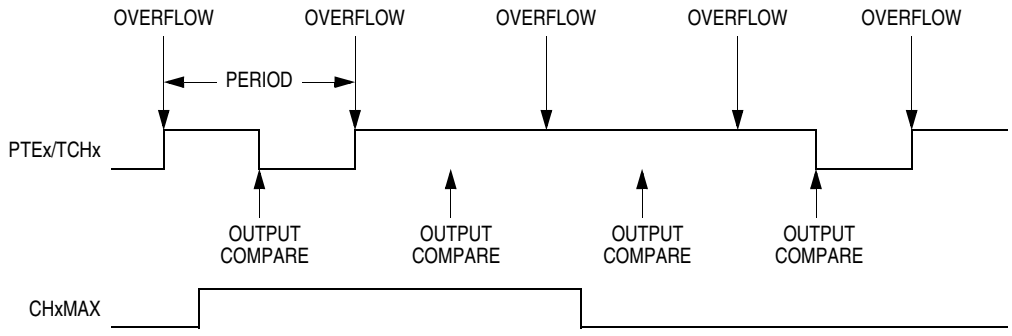


Figure 17-8. CHxMAX Latency

17.8.5 TIM Channel Registers

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

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

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

Register Name and Address: TCH0H — $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: TCH0L — $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: TCH1H — $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: TCH1L — $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: TCH2H — $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							

Register Name and Address: TCH2L — $002E

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

Figure 17-9. TIM Channel Registers (TCH0H/L–TCH5H/L)

Register Name and Address TCH3H — $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 TCH3L — $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 TCH4H — $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 TCH4L — $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 TCH5H — $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 TCH5L — $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 17-9. TIM Channel Registers (TCH0H/L–TCH5H/L) (Continued)

Chapter 18
Development Support

18.1 Introduction

This section describes the break module, the monitor read-only memory (MON), and the monitor mode entry methods.

18.2 Break Module (BRK)

The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

Features include:

- Accessible I/O registers during break interrupts
- CPU generated break interrupts
- Software generated break interrupts
- COP disabling during break interrupts

18.2.1 Functional Description

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

These events can cause a break interrupt to occur:

- A CPU generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a 1 to the BRKA bit in the break status and control register.

When a CPU generated address matches the contents of the break address registers, the break interrupt is generated. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. Figure 18-2 shows the structure of the break module.

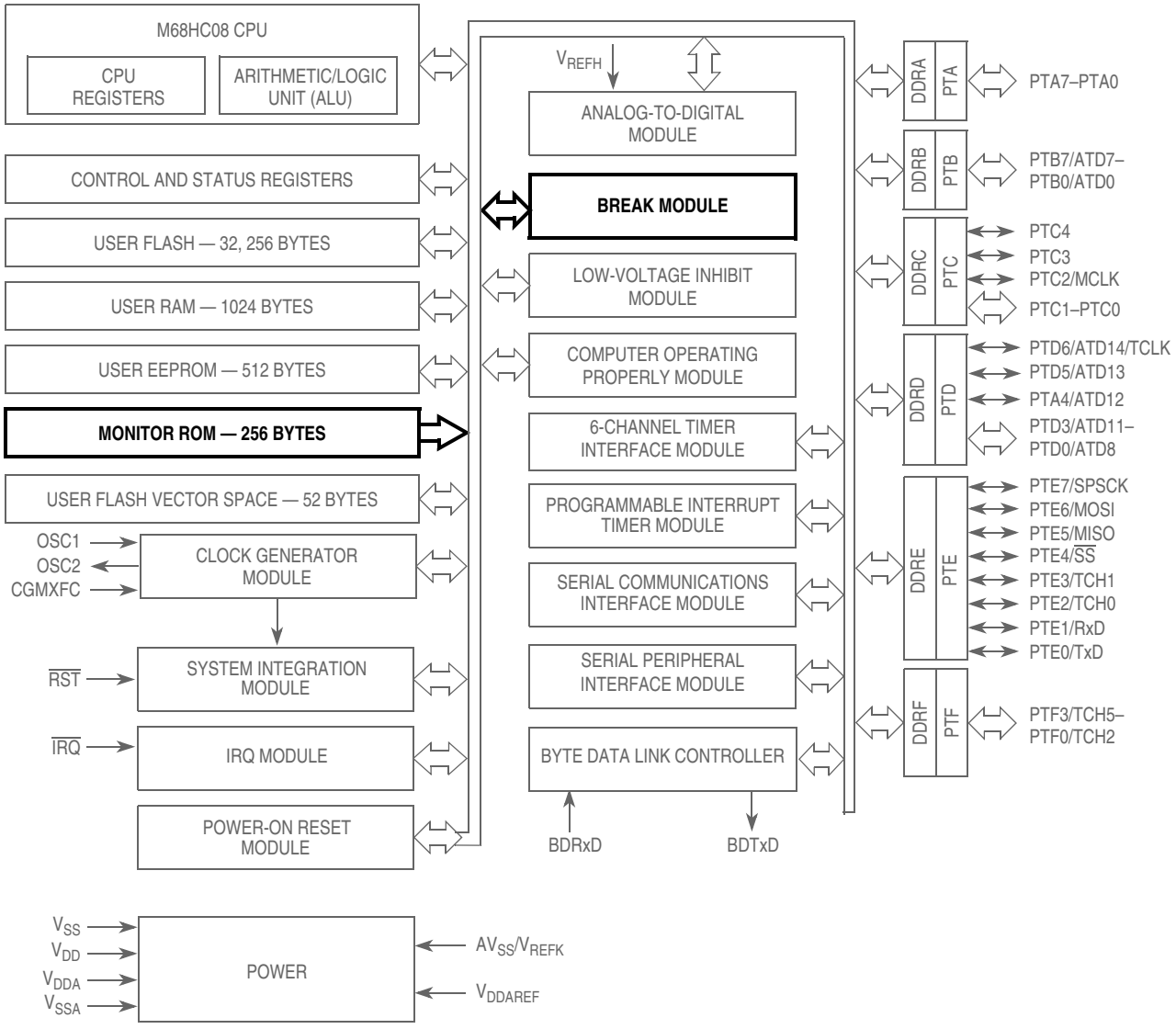


Figure 18-1. Block Diagram Highlighting BRK and MON Blocks

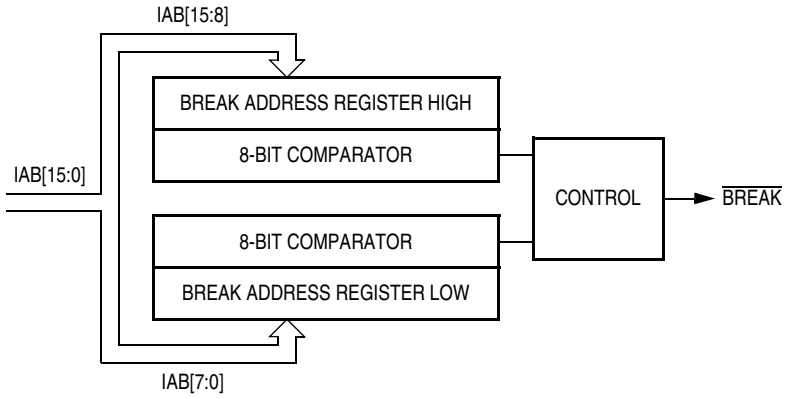


Figure 18-2. Break Module Block Diagram

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)

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.

18.2.1.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See 15.7.3 SIM Break Flag Control Register and the **Break Interrupts** subsection for each module.

18.2.1.2 TIM During Break Interrupts

A break interrupt stops the timer counter.

18.2.1.3 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 19.5 5.0 Volt DC Electrical Characteristics.

18.2.2 Break Module Registers

Three registers control and monitor operation of the break module:

- Break status and control register (BSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)

18.2.2.1 Break Status and Control Register

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

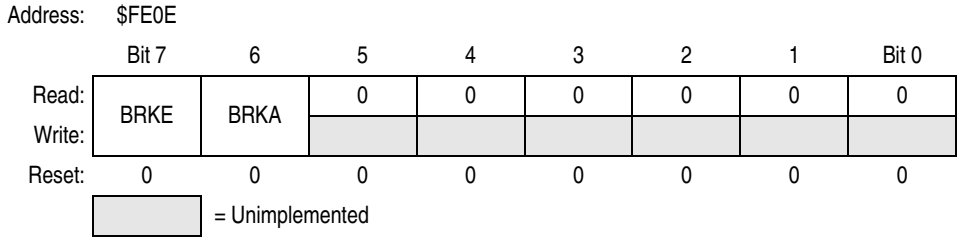


Figure 18-3. Break Status and Control Register (BSCR)

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 = When read, break address match
0 = When read, no break address match

18.2.2.2 Break Address Registers

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

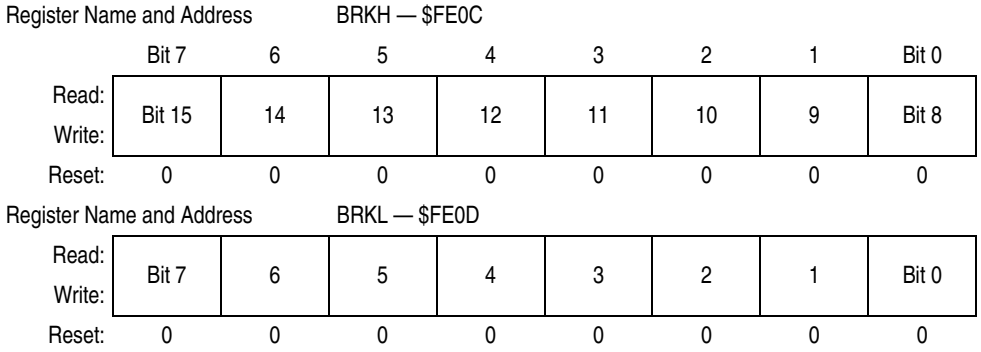


Figure 18-4. Break Address Registers (BRKH and BRKL)

18.2.3 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes. If enabled, the break module will remain enabled in wait and stop modes. However, since the internal address bus does not increment in these modes, a break interrupt will never be triggered.

18.3 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 28.8 kBaud communication with host computer
- Execution of code in RAM or FLASH
- FLASH security[1]
- FLASH programming

18.3.1 Functional Description

Figure 18-5 shows a simplified diagram of the monitor mode entry.

The monitor module receives and executes commands from a host computer.
Figure 18-6 shows an example circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

While simple monitor commands can access any memory address, the MC68HC908AS32A has a FLASH security feature to prevent external viewing of the contents of FLASH. Proper procedures must be followed to verify FLASH content. Access to the FLASH is denied to unauthorized users of customer specified software (see 18.3.2 Security).

In monitor mode, the MCU can execute code down loaded into RAM 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.

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

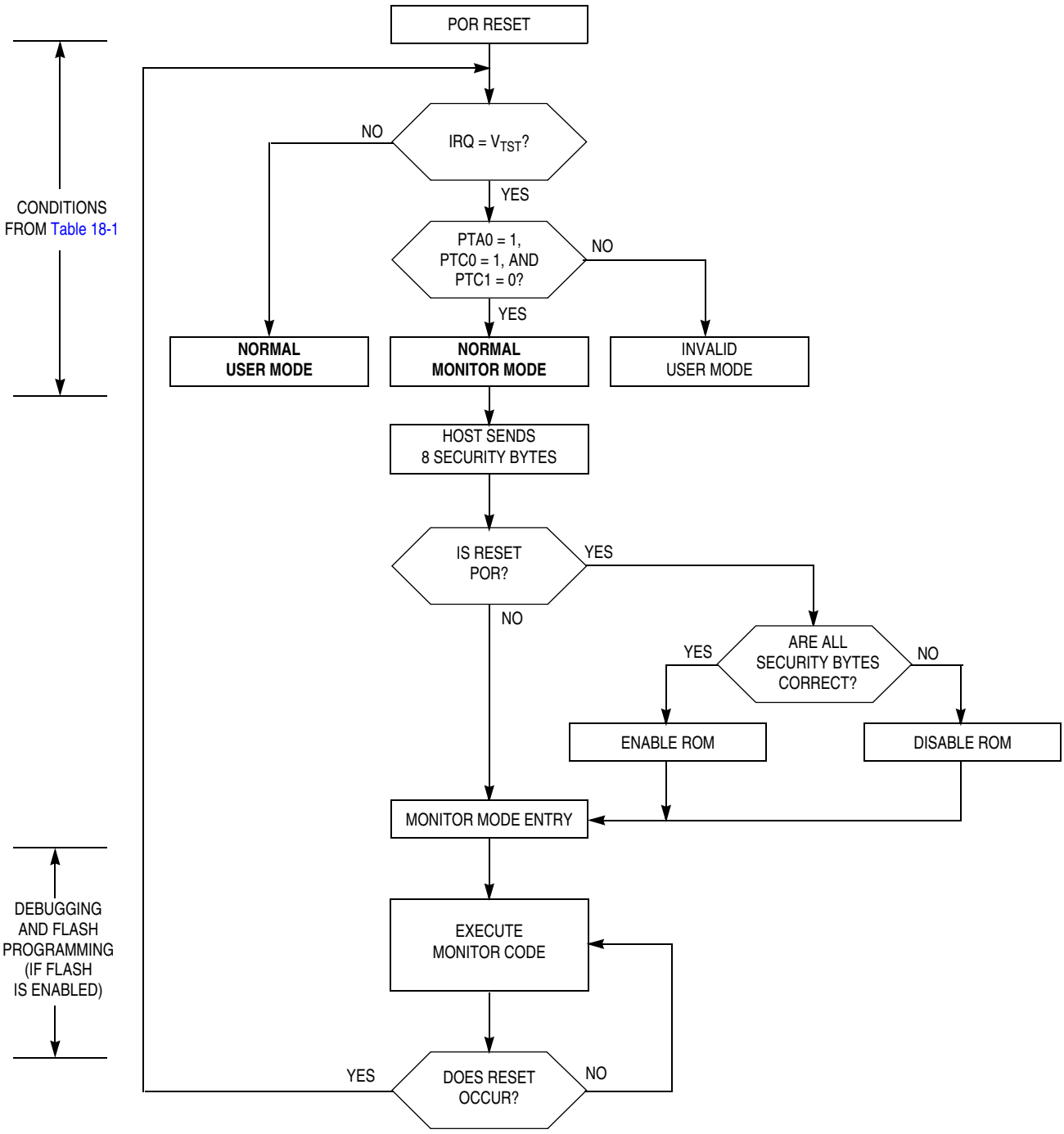


Figure 18-5. Simplified Monitor Mode Entry Flowchart

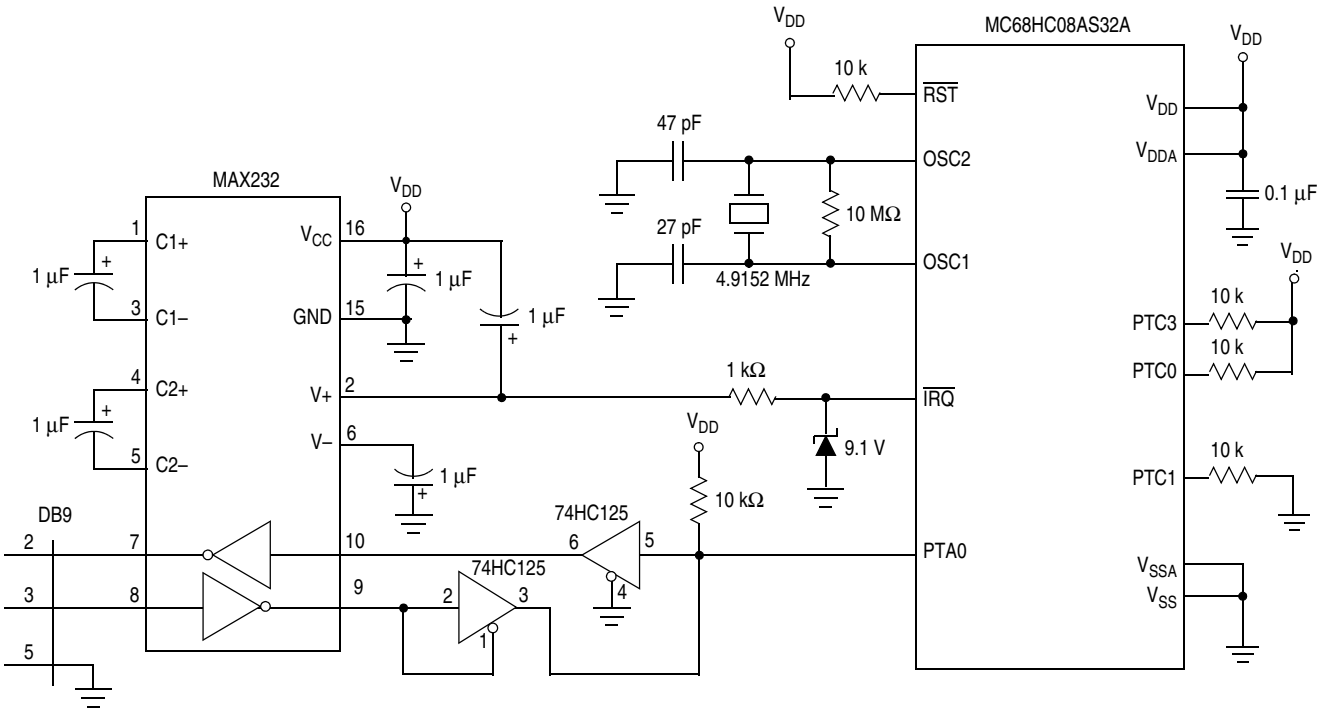


Figure 18-6. Normal Monitor Mode Circuit

18.3.1.1 Monitor Mode Entry

Table 18-1 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 18.3.2 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.

18.3.1.2 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 $\overline{\text{IRQ}}$ pin or the $\overline{\text{RST}}$ pin.

Table 18-2 summarizes the differences between user mode and monitor mode regarding vectors.

Table 18-1. Mode Selection

Mode	$\overline{IRQ}$	$\overline{RST}$	Serial Comm.	Mode Selection			PLL	COP	Communication Speed			Comments
			PTA0	PTC0	PTC1	PTC3			External Clock	Bus Frequency	Baud Rate	
Monitor	V_{TST}	V_{DD} or V_{TST}	1	1	0	0	Off	Disabled	4.9152 MHz	2.4576 MHz	9600	PTC3 determines frequency divider
	V_{TST}	V_{DD} or V_{TST}	1	1	0	1	Off	Disabled	4.9152 MHz	1.2288 MHz	4800	
User	V_{DD} or V_{SS}	V_{DD}	X	X	X	X	X	Enabled	X	X	X	
MON08 Function [Pin No.]	V_{TST} [6]	$\overline{RST}$ [4]	COM [10]	MOD0 [12]	MOD1 [14]	DIV4 [16]			OSC1 [13]			

1. PTA0 must have a pullup resistor to V_{DD} in monitor mode.
2. Communication speed in the table is an example to obtain a baud rate of 4800 or 9600.
   Baud rate using external oscillator is bus frequency / 256.
3. External clock is a 4.9152 MHz crystal on OSC1 and OSC2 or a 4.9152 or 9.8304 MHz canned oscillator on OSC1.
4. X = don't care
5. MON08 pin refers to P&E Microcomputer Systems' MON08-Cyclone 2 by 8-pin connector.

NC	1	2	GND
NC	3	4	RST
NC	5	6	IRQ
NC	7	8	NC
NC	9	10	PTA0
NC	11	12	PTC0
OSC1	13	14	PTC1
V_{DD}	15	16	PTC3

Table 18-2. Mode Differences

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

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. 19.5 5.0 Volt DC Electrical Characteristics

18.3.1.3 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See Figure 18-7 and Figure 18-8.) The data transmit and receive rate can be anywhere up to 28.8 Kbaud. Transmit and receive baud rates must be identical.

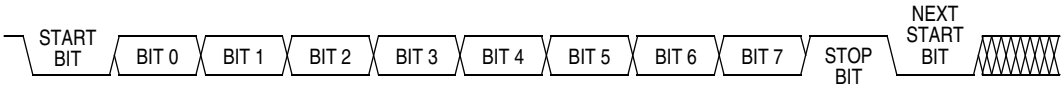


Figure 18-7. Monitor Data Format

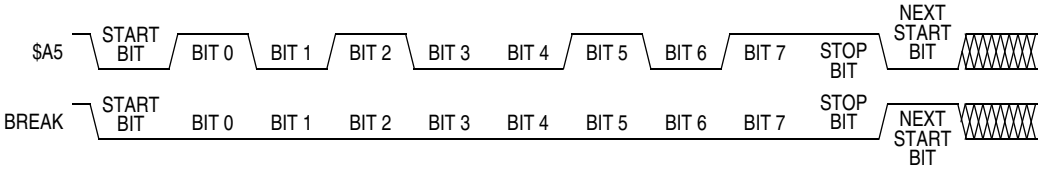


Figure 18-8. Sample Monitor Waveforms

18.3.1.4 Break Signal

A start bit followed by nine low bits is a break signal (see Figure 18-9). When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.

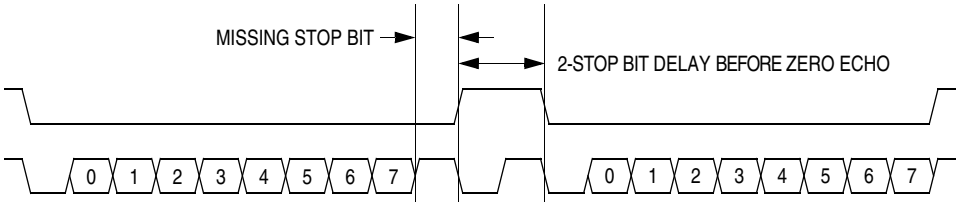


Figure 18-9. Break Transaction

18.3.1.5 Baud Rate

With a 4.9152-MHz crystal and the PTC3 pin high during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is low during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL[7:4] bits in the PLL programming register (PPG). See Chapter 5 Clock Generator Module (CGM).

Table 18-3. Monitor Baud Rate Selection

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

CAUTION

Care should be taken when setting the baud rate since incorrect baud rate setting can result in communications failure.

18.3.1.6 Commands

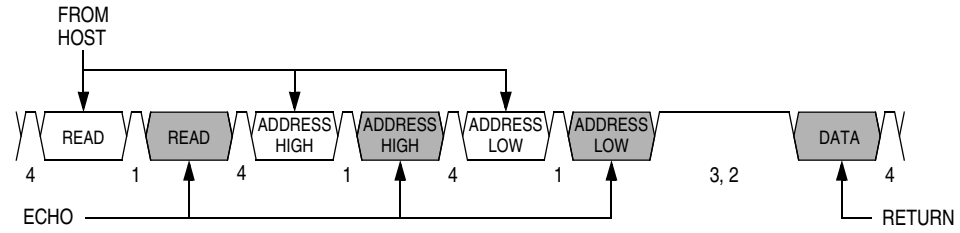
The monitor ROM firmware uses these commands:

- 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.

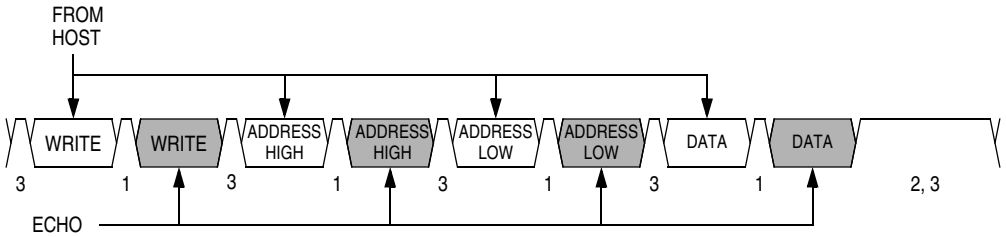
NOTE

Wait one bit time after each echo before sending the next byte.



Notes:
1 = Echo delay, 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 18-10. 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 18-11. Write Transaction

A brief description of each monitor mode command is given in Table 18-4 through Table 18-9.

Table 18-4. 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

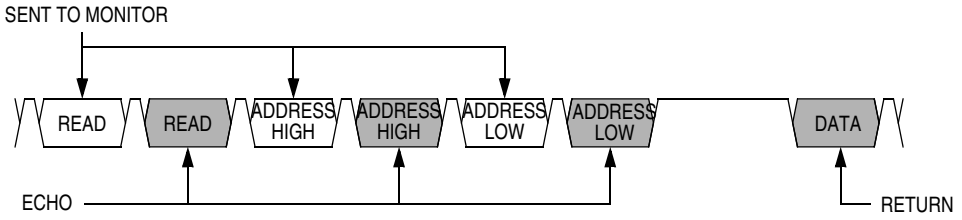


Table 18-5. WRITE (Write Memory) Command

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

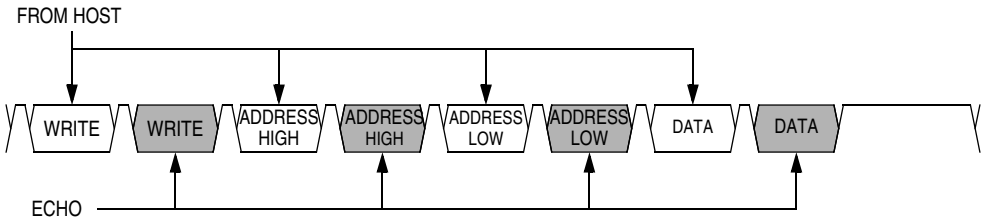


Table 18-6. IREAD (Indexed Read) Command

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

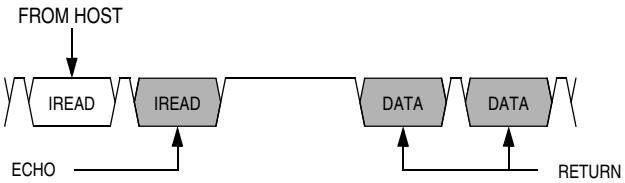
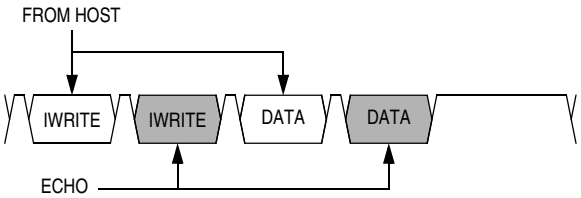


Table 18-7. IWRITE (Indexed Write) Command

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 18-8. 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	

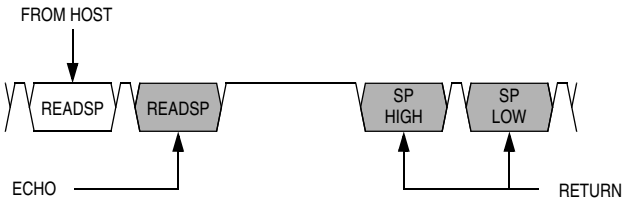
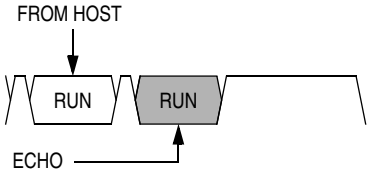


Table 18-9. RUN (Run User Program) Command

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.

	SP
HIGH BYTE OF INDEX REGISTER	SP + 1
CONDITION CODE REGISTER	SP + 2
ACCUMULATOR	SP + 3
LOW BYTE OF INDEX REGISTER	SP + 4
HIGH BYTE OF PROGRAM COUNTER	SP + 5
LOW BYTE OF PROGRAM COUNTER	SP + 6
	SP + 7

Figure 18-12. Stack Pointer at Monitor Mode Entry

18.3.2 Security

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

NOTE

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

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. See Figure 18-13.

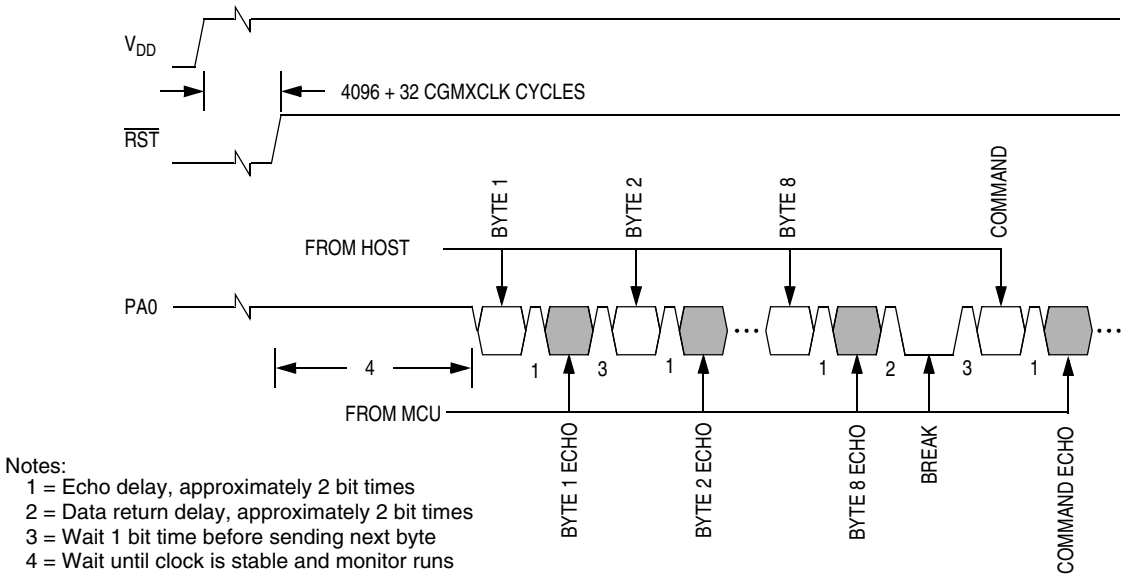


Figure 18-13. Monitor Mode Entry Timing

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

NOTE

The MCU does not transmit a break character until after the host sends the eight security bytes.

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $80 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

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. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

Chapter 19
Electrical Specifications

19.1 Introduction

This section contains electrical and timing specifications.

19.2 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 properly at the maximum ratings. Refer to 19.5 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
Reset and $\overline{IRQ}$ input voltage	V_{TST}	V_{DD} + 4.5	V

1. Voltages are referenced to V_{SS}.

NOTE

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that V_{In} and V_{Out} be constrained to the range $V_{SS} \le (V_{In}$ or $V_{Out}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}).

19.3 Functional Operating Range

Rating	Symbol	Value	Unit
Operating temperature range Part suffix MFN Part suffix VFN Part suffix CFN	T_A	 –40 to 125 –40 to 105 –40 to 85	°C
Operating voltage range	V_{DD}	5.0 ± 0.5	V

NOTE

For applications which use the LVI, Freescale guarantee the functionality of the device CPU only down to the LVI trip point (V_{LVI}) within the constraints outlined in Chapter 11 Low-Voltage Inhibit (LVI).

19.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance QFP (64 pins)	θ_{JA}	70	°C/W
Thermal resistance PLCC (52 pins)	θ_{JA}	50	°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.

19.5 5.0 Volt DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typical	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
Input high voltage All ports, $\overline{IRQ}$s, RESET, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage All ports, $\overline{IRQ}$s, RESET, OSC1	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
V_{DD} supply current Run[2] Wait[3] Stop[4] LVI enabled, T_A = 25°C LVI disabled, T_A = 25°C LVI enabled, –40°C to +125°C LVI disabled, –40°C to +125°C	I_{DD}[5]	 — — — — — —	 25 14 100 35 — —	 35 20 400 50 500 100	 mA mA µA µA µA µA
I/O ports Hi-Z leakage current	I_L	–1	—	1	µA
Input current	I_{In}	–1	—	1	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 rearm voltage[6]	V_{POR}	0	—	200	mV
POR reset voltage[7]	V_{PORRST}	0	—	800	mV
POR rise time ramp rate[8]	R_{POR}	0.02	—	—	V/ms
High COP disable voltage[9]	V_{TST}	V_{DD} + 3.0	—	V_{DD} + 4.5	V
Monitor mode entry voltage on $\overline{IRQ}$[10]	V_{TST}	V_{DD} + 3.0	—	V_{DD} + 4.5	V

1. V_{DD} = 5.0 Vdc ± 10%, 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. CL = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled. Typical values at midpoint of voltage range, 25°C only.
3. 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, CL = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait I_{DD}. Measured with all modules enabled. Typical values at midpoint of voltage range, 25°C only.
4. Stop I_{DD} measured with OSC1 = V_{SS}. Typical values at midpoint of voltage range, 25°C only.
5. Although I_{DD} is proportional to bus frequency, a current of several mA is present even at very low frequencies.
6. Maximum is highest voltage that POR is guaranteed.
7. Maximum is highest voltage that POR is possible.
8. If minimum V_{DD} is not reached before the internal POR reset is released, $\overline{RST}$ must be driven low externally until minimum V_{DD} is reached.
9. See 8.8 COP Module During Break Interrupts. V_{TST} applied to $\overline{RST}$.
10. See monitor mode description within Chapter 8 Computer Operating Properly (COP). V_{TST} applied to $\overline{IRQ}$ or $\overline{RST}$

19.6 Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Bus operating frequency (4.5–5.5 V — V_{DD} only)	f_{Bus}	—	8.4	MHz
$\overline{RST}$ 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 4	—	t_{cyc}
16-bit timer[2] Input capture pulse width[3] Input capture period	 t_{TH}, t_{TL} t_{TLTL}	 2 Note[4]	 — —	t_{cyc}

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. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
3. Refer to Table 17-2. Mode, Edge, and Level Selection.
4. 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 TBD t_{cyc}.

19.7 Analog-to-Digital Converter (ADC) Characteristics

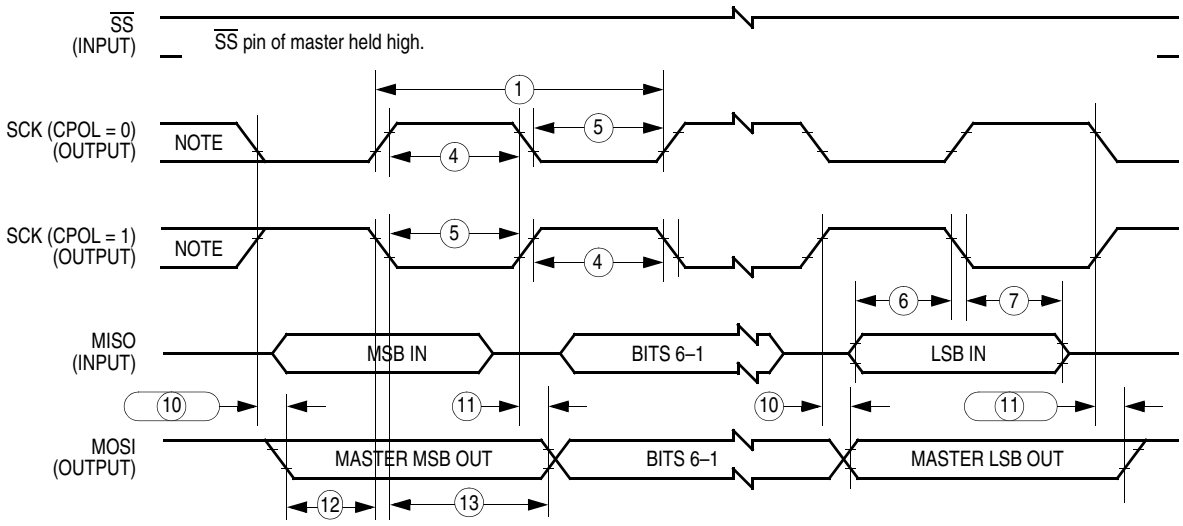
Characteristic	Min	Max	Unit	Comments
Resolution	8	8	Bits	
Absolute accuracy (V_{REFL} = 0 V, V_{DDA}/V_{DDAREF} = V_{REFH} = 5 V ± 0.5 V)	–1	+1	LSB	Includes quantization
Conversion range[1]	V_{REFL}	V_{REFH}	V	V_{REFL} = V_{SSA}
Power-up time	16	17	µs	Conversion time period
Input leakage[2] (ports B and D)	–1	1	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[3]	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. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.

19.8 5.0 Vdc ± 0.5 V Serial Peripheral Interface (SPI) Timing

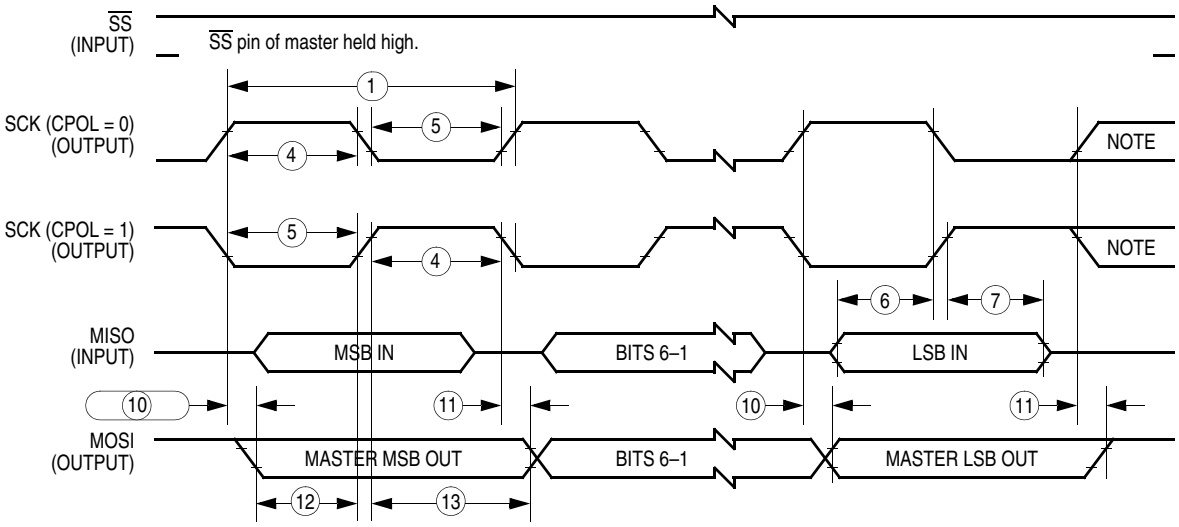
Num(1)	Characteristic(2)	Symbol	Min	Max	Unit
	Operating frequency(3) Master Slave	$f_{BUS(M)}$ $f_{BUS(S)}$	$f_{BUS}/128$ dc	$f_{BUS}/2$ f_{BUS}	MHz
1	Cycle time Master Slave	$t_{cyc(M)}$ $t_{cyc(S)}$	2 1	128 —	t_{cyc}
2	Enable lead time	t_{Lead}	15	—	ns
3	Enable lag time	t_{Lag}	15	—	ns
4	Clock (SCK) 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 lead 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 (before capture edge)	$t_{HO(M)}$	100	—	ns

1. Item numbers refer to dimensions in Figure 19-1 and Figure 19-2.
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.

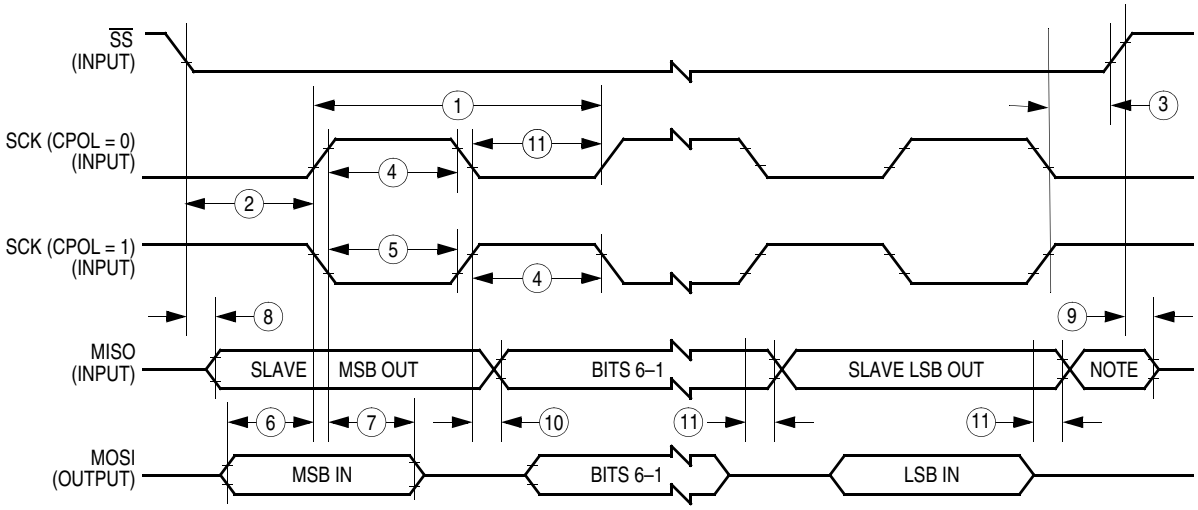
a) SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

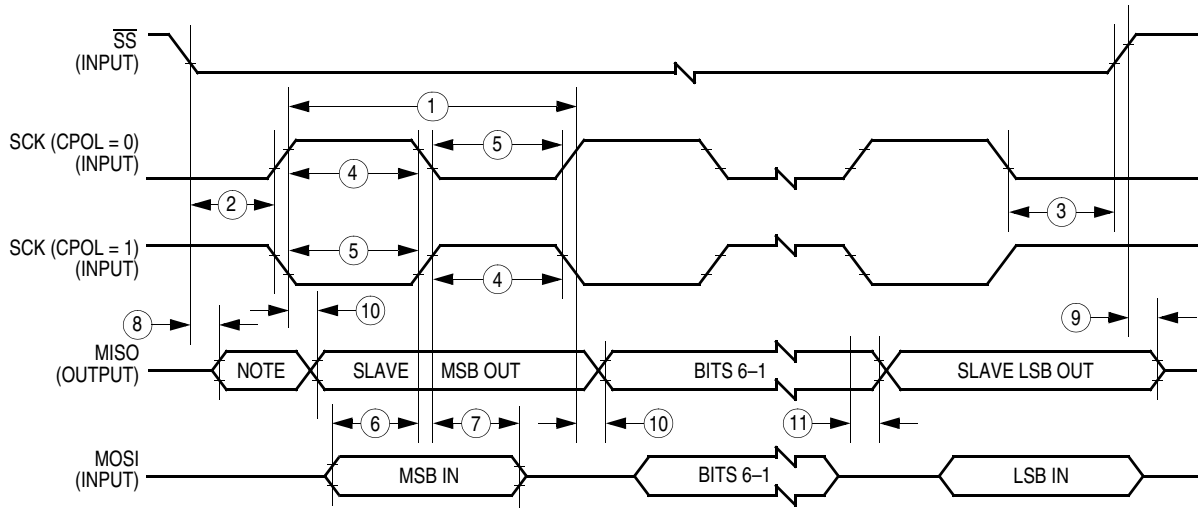
b) SPI Master Timing (CPHA = 1)

Figure 19-1. 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 19-2. SPI Slave Timing Diagram

19.9 Clock Generator Module (CGM) Characteristics

19.9.1 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	16	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 5 Clock Generator Module (CGM) for the desired VCO operating frequency, $f_{CGMVCLK}$.

19.9.2 CGM Component Information

Description	Symbol	Min	Typ	Max	Unit
Crystal load capacitance(1)	C_L	—	—	—	—
Crystal fixed capacitance(1)	C1	—	2 x C_L	—	—
Crystal tuning capacitance(1)	C2	—	2 x C_L	—	—
Filter capacitor multiply factor	C_{FACT}	—	0.0154	—	F/s V
Filter capacitor(2)	C_F	—	C_{FACT} x (V_{DDA}/f_{XCLK})	—	—
Bypass capacitor(3)	C_{BYP}	—	0.1	—	µF

1. Consult crystal manufacturer's data.
2. See 5.4.3 External Filter Capacitor Pin (CGMXFC).
3. C_{BYP} must provide low AC impedance from f = $f_{CGMXCLK}$/100 to 100 x $f_{CGMVCLK}$, so series resistance must be considered.

19.9.3 CGM Acquisition/Lock Time Information

Description[1]	Symbol	Min	Typ[2]	Max	Unit
Manual mode time to stable[3]	t_{ACQ}	—	$(8 \times V_{DDA}) / (f_{CGMXCLK} \times K_{ACQ})$	—	s
Manual stable to lock time[1]	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 frequency tolerance	D_{Lock}	0	—	± 0.9	%
LOCK exit frequency 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[1]	t_{ACQ}	n_{ACQ}/f_{XCLK}	$(8 \times V_{DDA}) / (f_{XCLK} \times K_{ACQ})$		s
Automatic stable to lock time[1]	t_{AL}	n_{TRK}/f_{XCLK}	$(4 \times V_{DDA}) / (f_{XCLK} \times K_{TRK})$	—	s
Automatic lock time	t_{Lock}	—	0.65	25	ms
PLL jitter, deviation of average bus frequency over 2 ms[4]		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 = –40C 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. N = VCO frequency multiplied. Guaranteed but not tested. Refer to Chapter 5 Clock Generator Module (CGM) for guidance on the use of the PLL.

19.10 Timer Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Input capture pulse width	t_{TIH}, t_{TIL}	125	—	ns
Input clock pulse width	t_{TCH}, t_{TCL}	$(1/f_{OP}) + 5$	—	ns

19.11 Memory Characteristics

19.11.1 RAM Memory Characteristics

Characteristic	Symbol	Min	Max	Unit
RAM data retention voltage	V_{RDR}	0.7	—	V

19.11.2 EEPROM Memory Characteristics

Characteristic	Symbol	Min	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 write/erase cycles @ 10 ms write time	—	10,000	—	Cycles
EEPROM data retention after 10,000 write/erase cycles	—	15	—	Years
EEPROM programming maximum time to AUTO bit set	—	—	500	µs
EEPROM erasing maximum time to AUTO bit set	—	—	8	ms

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.

19.11.3 FLASH Memory Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
FLASH program bus clock frequency	—	1	—	—	MHz
FLASH read bus clock frequency	f_{Read}[1]	0	—	8 M	Hz
FLASH page erase time <1 K cycles >1 K cycles	t_{Erase}	 0.9 3.6	 1 4	 1.1 5.5	ms
FLASH mass erase time	t_{MErase}	4	—	—	ms
FLASH PGM/ERASE to HVEN setup time	t_{NVS}	10	—	—	µs
FLASH high-voltage hold time	t_{NVH}	5	—	—	µs
FLASH high-voltage hold time (mass erase)	t_{NVHL}	100	—	—	µs
FLASH program hold time	t_{PGS}	5	—	—	µs
FLASH program time	t_{PROG}	30	—	40	µs
FLASH return to read time	t_{RCV}[2]	1	—	—	µs
FLASH cumulative program hv period	t_{HV}[3]	—	—	4	ms
FLASH endurance[4]	—	10 k	100 k	—	Cycles
FLASH data retention time[5]	—	15	100	—	Years

1. f_{Read} is defined as the frequency range for which the FLASH memory can be read.
2. t_{RCV} is defined as the time it needs before the FLASH can be read after turning off the high voltage charge pump, by clearing HVEN to 0.
3. t_{HV} is defined as the cumulative high voltage programming time to the same row before next erase. t_{HV} must satisfy this condition: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} \times 64) \leq t_{HV}$ maximum.
4. Typical endurance was evaluated for this product family. For additional information on how Freescale defines *Typical Endurance*, please refer to Engineering Bulletin EB619.
5. 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.

19.12 Byte Data Link Controller (BDLC) Characteristics

19.12.1 BDLC Transmitter VPW Symbol Timings

Characteristic[1], [2]	Number	Symbol	Min	Typ	Max	Unit
Passive logic 0	10	t_{TVP1}	62	64	66	µs
Passive logic 1	11	t_{TVP2}	126	128	130	µs
Active logic 0	12	t_{TVA1}	126	128	130	µs
Active logic 1	13	t_{TVA2}	62	64	66	µs
Start-of-frame (SOF)	14	t_{TVA3}	198	200	202	µs
End-of-data (EOD)	15	t_{TVP3}	198	200	202	µs
End-of-frame (EOF)	16	t_{TV4}	278	280	282	µs
Inter-frame separator (IFS)	17	t_{TV6}	298	300	302	µs

1. f_{BDLC} = 1.048576 or 1.0 MHz, V_{DD} = 5.0 V ± 10%, V_{SS} = 0 V
2. See Figure 19-3.

19.12.2 BDLC Receiver VPW Symbol Timings

Characteristic[1], [2], [3]	Number	Symbol	Min	Typ	Max	Unit
Passive logic 0	10	t_{TRVP1}	34	64	96	µs
Passive logic 1	11	t_{TRVP2}	96	128	163	µs
Active logic 0	12	t_{TRVA1}	96	128	163	µs
Active logic 1	13	t_{TRVA2}	34	64	96	µs
Start-of-frame (SOF)	14	t_{TRVA3}	163	200	239	µs
End-of-data (EOD)	15	t_{TRVP3}	163	200	239	µs
End-of-frame (EOF)	16	t_{TRV4}	239	280	320	µs
Break	18	t_{TRV6}	280	—	—	µs

1. f_{BDLC} = 1.048576 or 1.0 MHz, V_{DD} = 5.0 V ± 10%, V_{SS} = 0 V
2. The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{BDLC} µs due to sampling considerations.
3. See Figure 19-3.

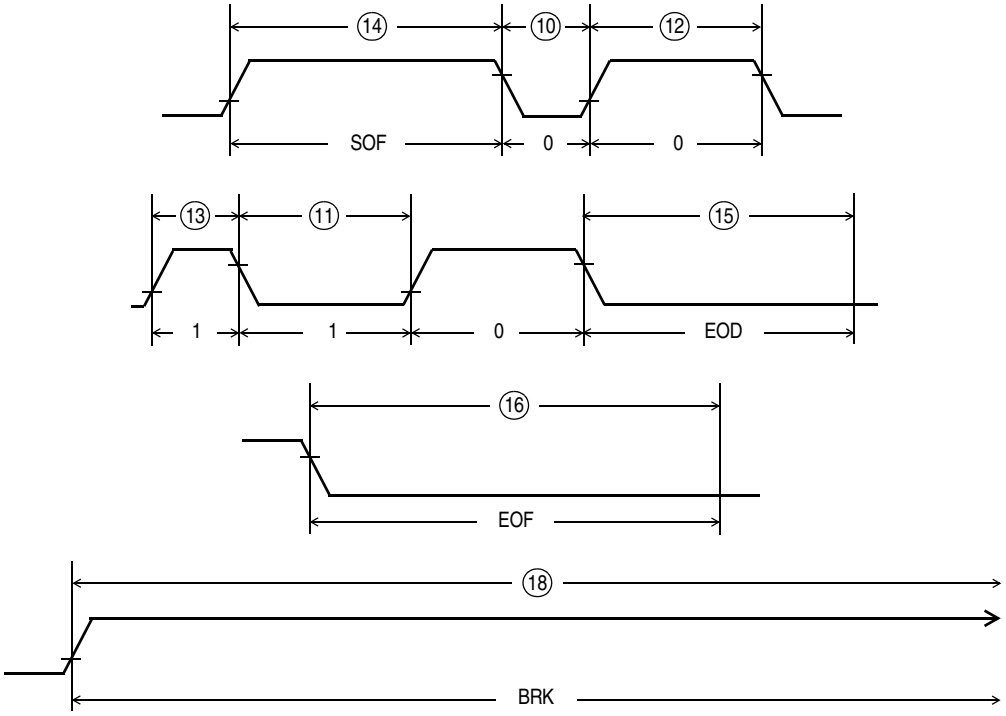


Figure 19-3. BDLC Variable Pulse Width Modulation (VPW) Symbol Timing

19.12.3 BDLC Transmitter DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Max	Unit
BDTxD output low voltage (IBDTxD = 1.6 mA)	V_{OLTX}	—	0.4	V
BDTxD output high voltage (IBDTx = –800 μA)	V_{OHTX}	V_{DD} –0.8	—	V

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40 °C to +125 °C, unless otherwise noted

19.12.4 BDLC Receiver DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Max	Unit
BDRxD input low voltage	V_{ILRX}	V_{SS}	0.3 x V_{DD}	V
BDRxD input high voltage	V_{IHRX}	0.7 x V_{DD}	V_{DD}	V
BDRxD input low current	$I_{ILBDRXI}$	–1	+1	μA
BDRxD input high current	I_{HBDRX}	–1	+1	μA

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted

Chapter 20
Ordering Information and Mechanical Specifications

20.1 Introduction

This section provides ordering information for the MC68HC908AS32A along with dimensions for:

- 52-pin plastic leaded chip carrier (PLCC)
- 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

20.2 MC Order Numbers

Table 20-1. MC Order Numbers

MC Order Number	Operating Temperature Range
MC68HC908AS32ACFN(1)	–40°C to + 85°C
MC68HC908AS32AVFN(1)	–40°C to + 105°C
MC68HC908AS32AMFN(1)	–40°C to + 125°C
MC68HC908AS32AFU(2)	–40°C to + 85°C
MC68HC908AS32ACFU	–40°C to + 105°C
MC68HC908AS32AVFU	–40°C to + 125°C

1. FN = plastic leaded chip carrier
2. FU = quad flat pack

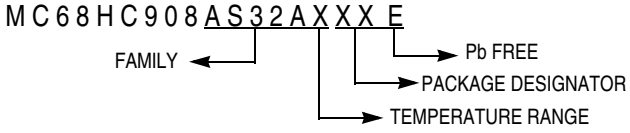
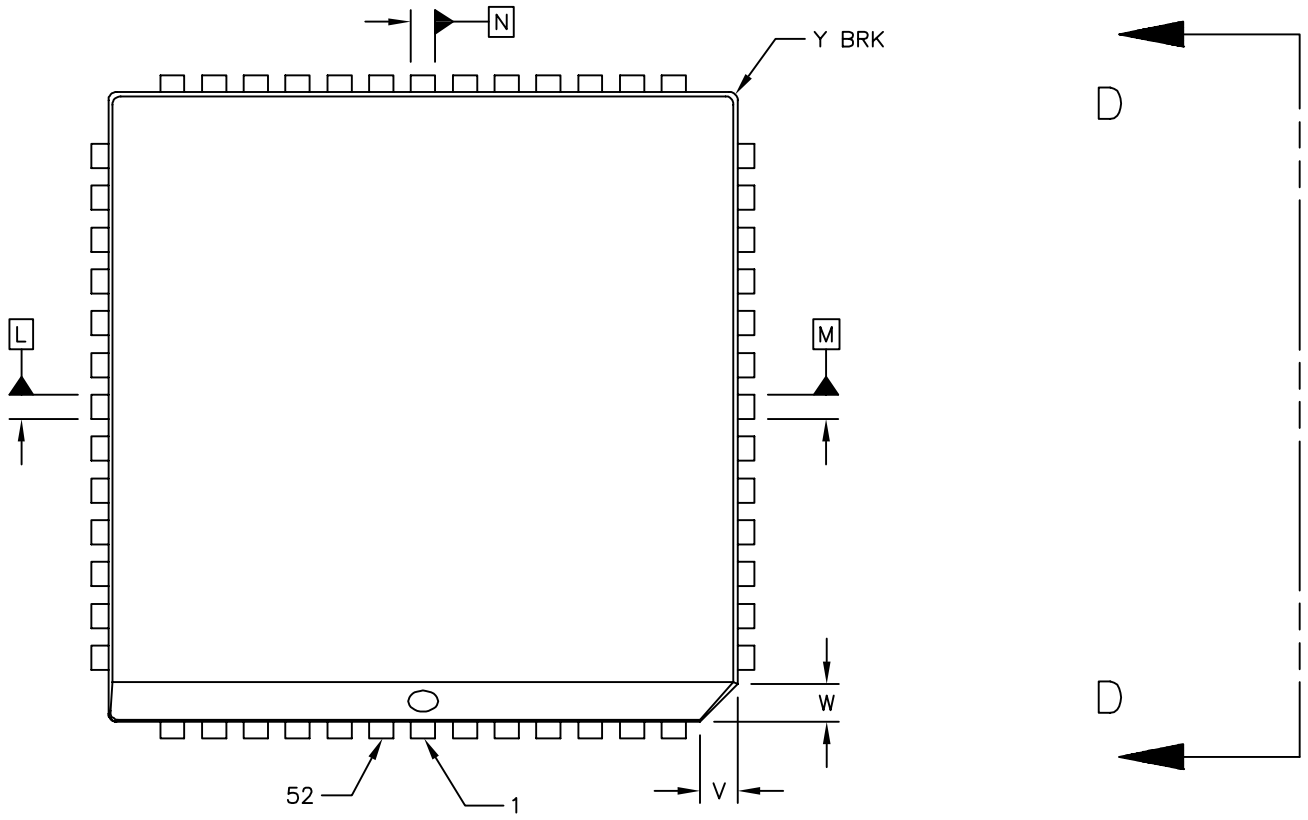
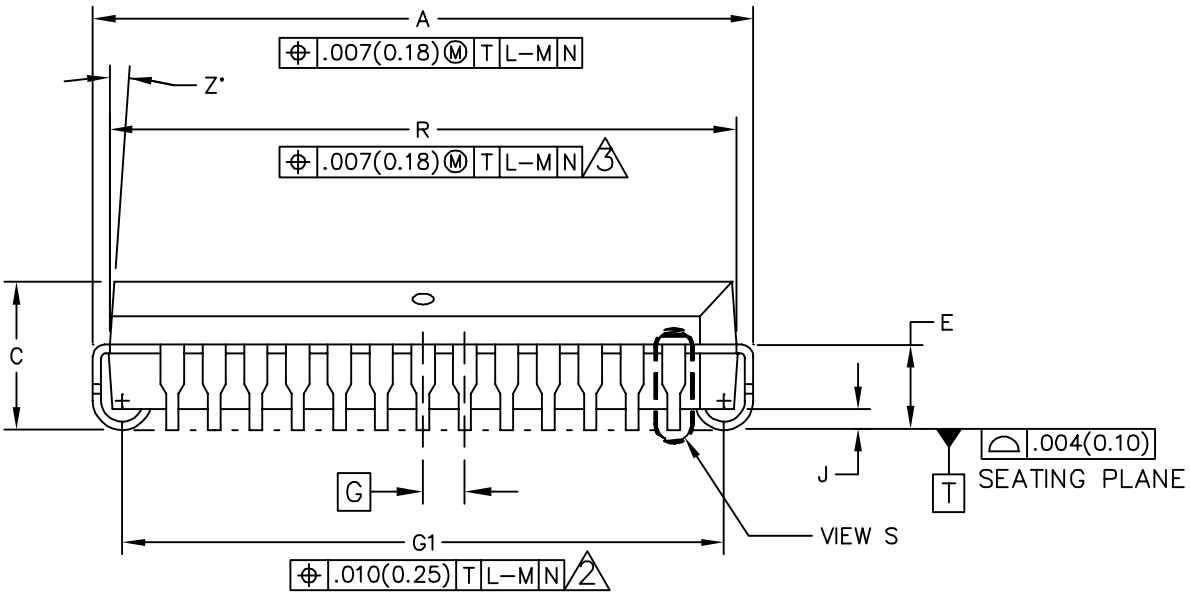


Figure 20-1. Device Numbering System

20.3 Package Dimensions

Refer to the following pages for detailed package dimensions.



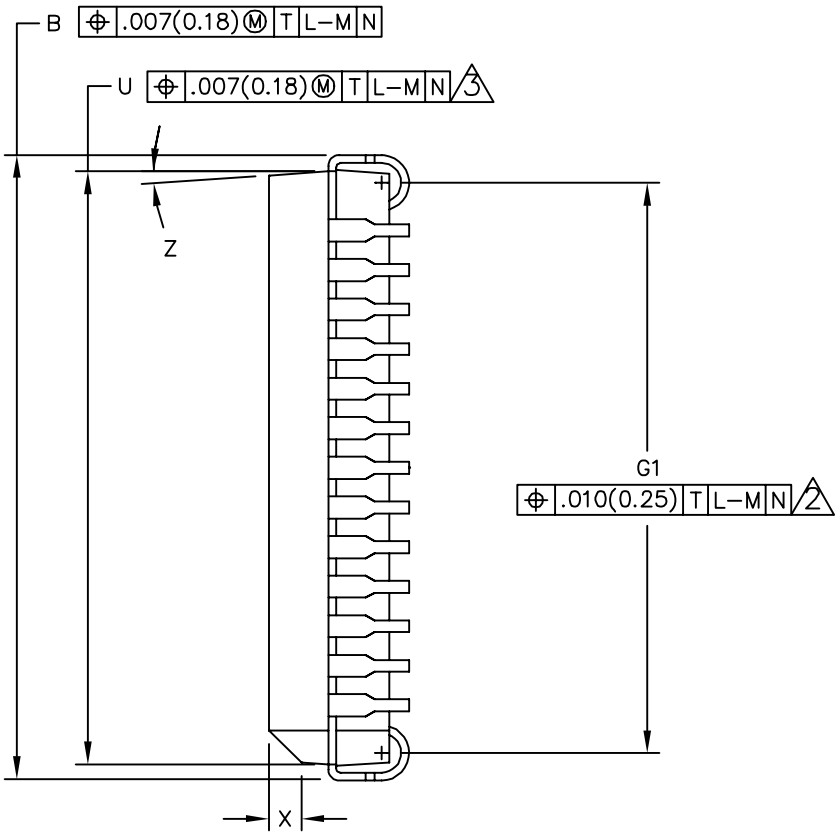


MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

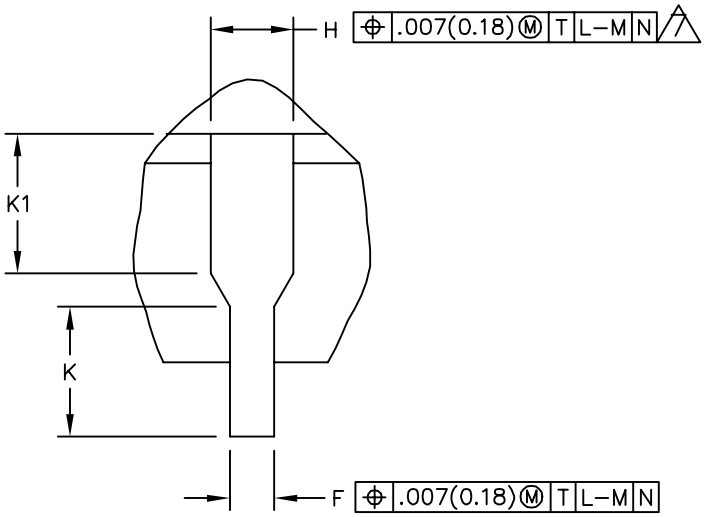
TITLE:

MOS/B1 POLAR (52 LEAD)
PLASTIC LEADED CHIP CARRIER

DOCUMENT NO: 98ASB42600B	REV: E
CASE NUMBER: 778–02	19 MAY 2005
STANDARD: NON–JEDEC	



VIEW D–D



VIEW S

MECHANICAL OUTLINE

PRINT VERSION NOT TO SCALE

TITLE:

MOS/B1 POLAR (52 LEAD)
PLASTIC LEADED CHIP CARRIER

DOCUMENT NO: 98ASB42600B

REV: E

CASE NUMBER: 778–02

19 MAY 2005

STANDARD: NON–JEDEC

NOTES:

1. DATUMS L, M AND N DETERMINED WHERE TOP OF LEAD SHOULDER EXITS PLASTIC BODY AT MOLD PARTING LINE.

2 DIMENSION TRUE POSITION TO BE MEASURED AT DATUM T, SEATING PLANE.

3 DIMENSIONS DO NOT INCLUDE MOLD FLASH. ALLOWABLE MOLD FLASH IS .010(0.25) PER SIDE.

4 DIMENSIONING AND TOLERANCING PER ASME Y14.5 – 1994.

5 CONTROLLING DIMENSION: INCH.

6 THE PACKAGE TOP MAY BE SMALLER THAN THE PACKAGE BOTTOM BY UP TO .012(0.30). DIMENSIONS R AND U ARE DETERMINED AT THE OUTERMOST EXTREMES OF THE PLASTIC BODY EXLUSIVE OF MOLD FLASH. TIE BAR BURRS, GATE BURRS AND INTERLEAD FLASH, BUT INCLUDING ANY MISMATCH BETWEEN THE TOP AND BOTTOM OF THE PLASTIC BODY.

7 DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION OR INTRUSION. THE DAMBAR PROTRUSION (S) SHALL NOT CAUSE THE DIMENSION TO BE GREATER THAN .037(0.94). THE DAM BAR INTRUSION (S) SHALL NOT CAUSE THE DIMENSION TO BE SMALLER THAN .025(0.63).

DIM	MILLIMETERS MIN	MAX	INCHES MIN	MAX	DIM	MILLIMETERS MIN	MAX	INCHES MIN	MAX
A	19.94	20.19	0.785	0.795					
B	19.94	20.19	0.785	0.795					
C	4.20	4.57	0.165	0.180					
E	2.29	2.79	0.090	0.110					
F	0.33	0.48	0.013	0.019					
G	1.27 BSC		0.050 BSC						
H	0.66	0.81	0.026	0.032					
J	0.51	---	0.020	---					
K	0.64	---	0.025	---					
R	19.05	19.20	0.750	0.756					
U	19.05	19.20	0.750	0.756					
V	1.07	1.21	0.042	0.048					
W	1.07	1.21	0.042	0.048					
X	1.07	1.42	0.042	0.056					
Y	---	0.50	---	0.020					
Z	2°	10°	2°	10°					
G1	18.04	18.54	0.710	0.730					
K1	1.02	---	0.040	---					

MECHANICAL OUTLINE

PRINT VERSION NOT TO SCALE

TITLE:

MOS/B1 POLAR (52 LEAD)
PLASTIC LEADED CHIP CARRIER

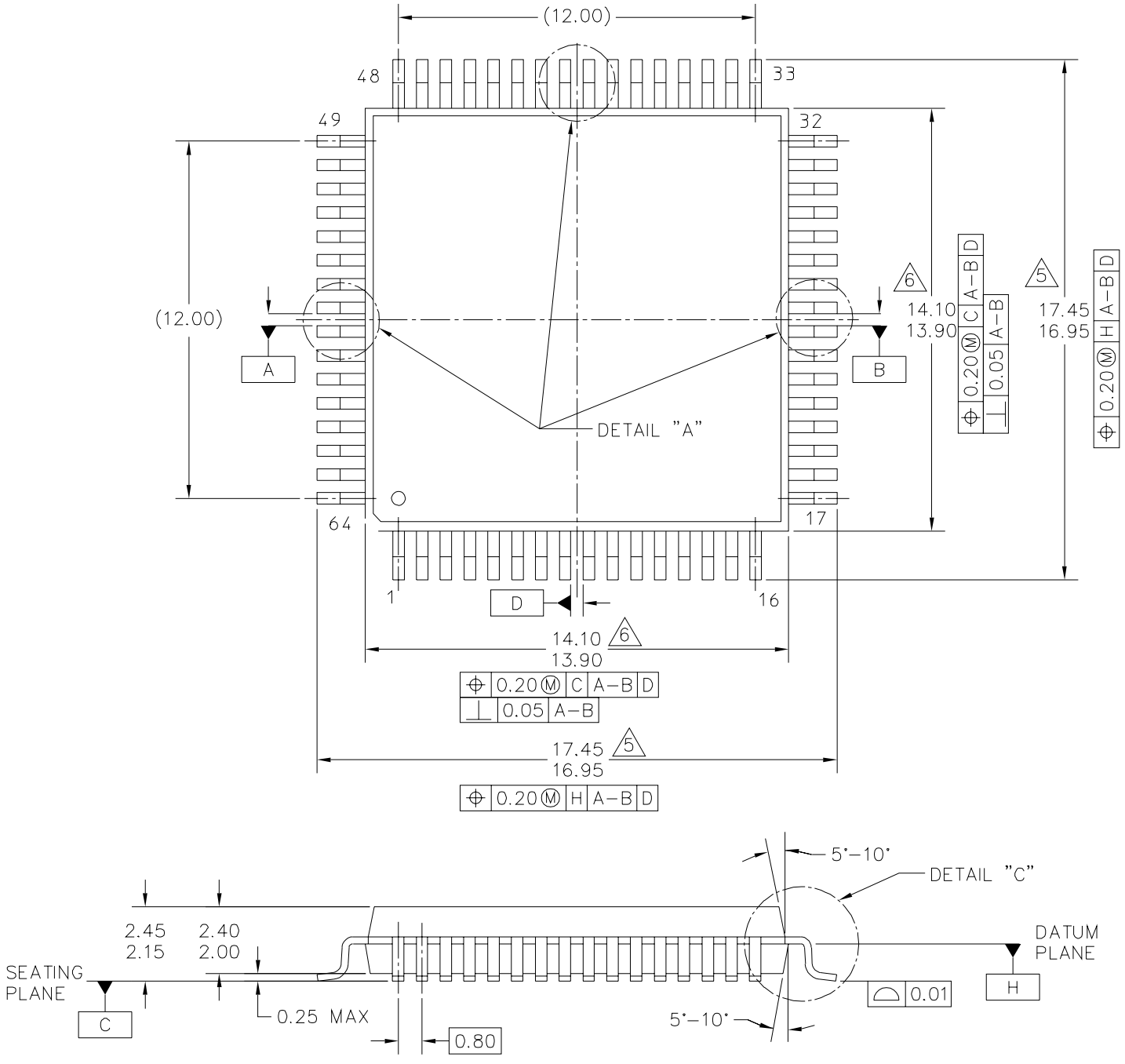
DOCUMENT NO: 98ASB42600B

REV: E

CASE NUMBER: 778–02

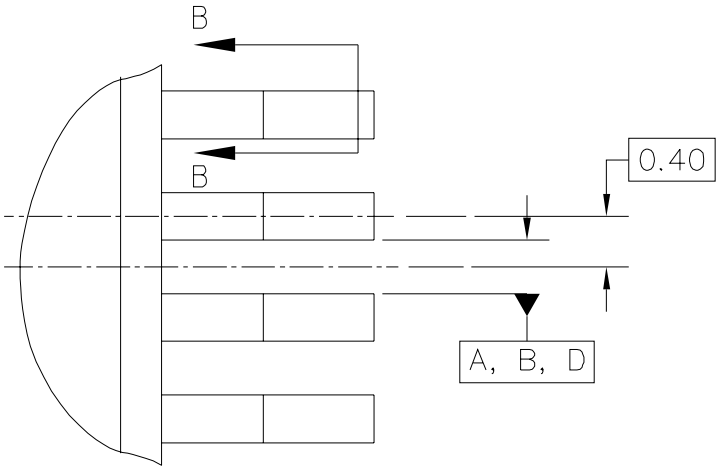
19 MAY 2005

STANDARD: NON–JEDEC

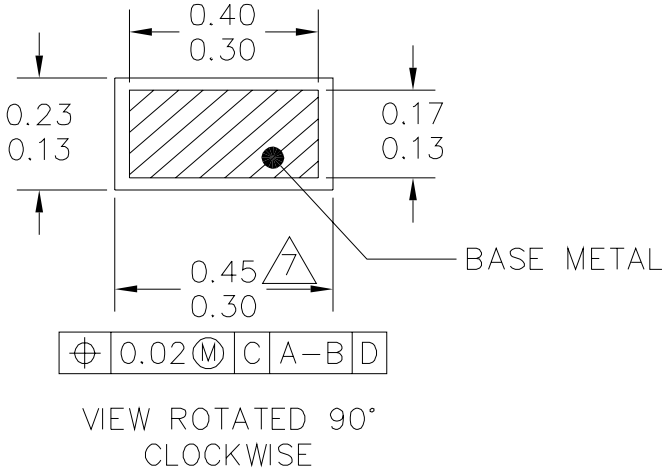


MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE:	DOCUMENT NO: 98ASB42844B	REV: A
64LD QFP (14 X 14)	CASE NUMBER: 840B-02	06 APR 2005
	STANDARD: NON-JEDEC	

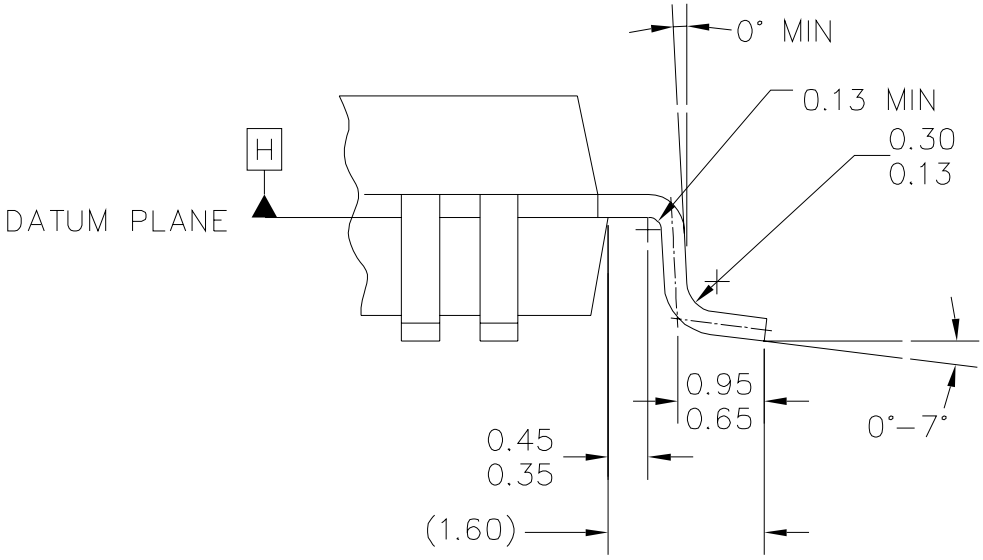


DETAIL "A"



VIEW ROTATED 90° CLOCKWISE

SECTION B–B



DETAIL "C"

	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: 64LD QFP (14 X 14)	DOCUMENT NO: 98ASB42844B	REV: A
	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	

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.

MC68HC908AS32A
Rev. 2.0, 05/2006

freescale™
semiconductor