HC708AS48GRS/D
REV. 2.0

68HC708AS48

General Release Specification

CSIC System Design Group
Austin, Texas

MOTOROLA

Motorola reserves the right to make changes without further notice to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part.

List of Sections

Table of Contents

Section 1. General Description

Section 2. Memory Map

Section 3. Random-Access Memory (RAM)

Section 4. EPROM/OTPROM

Section 5. Electrically Erasable Programmable ROM (EEPROM)

Section 6. Central Processor Unit (CPU)

Section 7. System Integration Module (SIM)

Section 8. Clock Generator Module (CGM)

Section 9. Configuration Register (CONFIG)

Section 10. Break Module (Break)

Section 11. Monitor ROM (MON08)

Section 12. Computer Operating Properly (COP)

Section 13. Low-Voltage Inhibit (LVI)

Section 14. External Interrupt (IRQ)

Section 15. Serial Communications Interface (SCI)

Section 16. Serial Peripheral Interface (SPI)

Section 17. Timer Interface (TIM)

Section 18. Analog-to-Digital Converter (ADC)

Section 19. Input/Output (I/O) Ports

Section 20. Byte Data Link Controller-Digital (BDLC-D)

Section 21. Electrical Specifications

Appendix A. CGM Electrical Information

List of Figures

List of Tables

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC708AS48 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). The M68HC08 Family is based on the customer-specified integrated circuit (CSIC) design strategy. All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

1.3 Features

Features of the MC68HC708AS48 include:

- High-Performance M68HC08 Architecture
- Fully Upward-Compatible Object Code with M6805, M146805, and M68HC05 Families
- 4-MHz Internal Bus Frequency
- 48 Kbytes of On-Chip Erasable Programmable Read-Only Memory (EPROM) or One-Time Programmable Read-Only Memory (OTPROM)
- On-Chip Programming Firmware for Use with Host Personal Computer
- EPROM/OTPROM Data Security
- 640 Bytes of On-Chip Electrically Erasable Programmable Read-Only Memory (EEPROM)
- 1.5 Kbytes of On-Chip RAM
- Serial Peripheral Interface Module (SPI)
- Serial Communications Interface Module (SCI)
- 16-Bit, 6-Channel Timer Interface Module (TIM)
- Clock Generator Module (CGM)
- 8-Bit, 15-Channel Analog-to-Digital Converter Module (ADC)
- SAE J1850 Byte Data Link Controller Digital Module (BDLC-D)

- System Protection Features
  - Computer Operating Properly (COP) Optional with 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

Additional features of the 64-pin plastic quad flat pack (QFP) include:

- PTC5 General Input/Output (I/O) Port
- PTD7 General I/O Port with A/D Function
- PTF4 General I/O Port
- Three PTG General I/O Ports

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

1.4 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC708AS48.

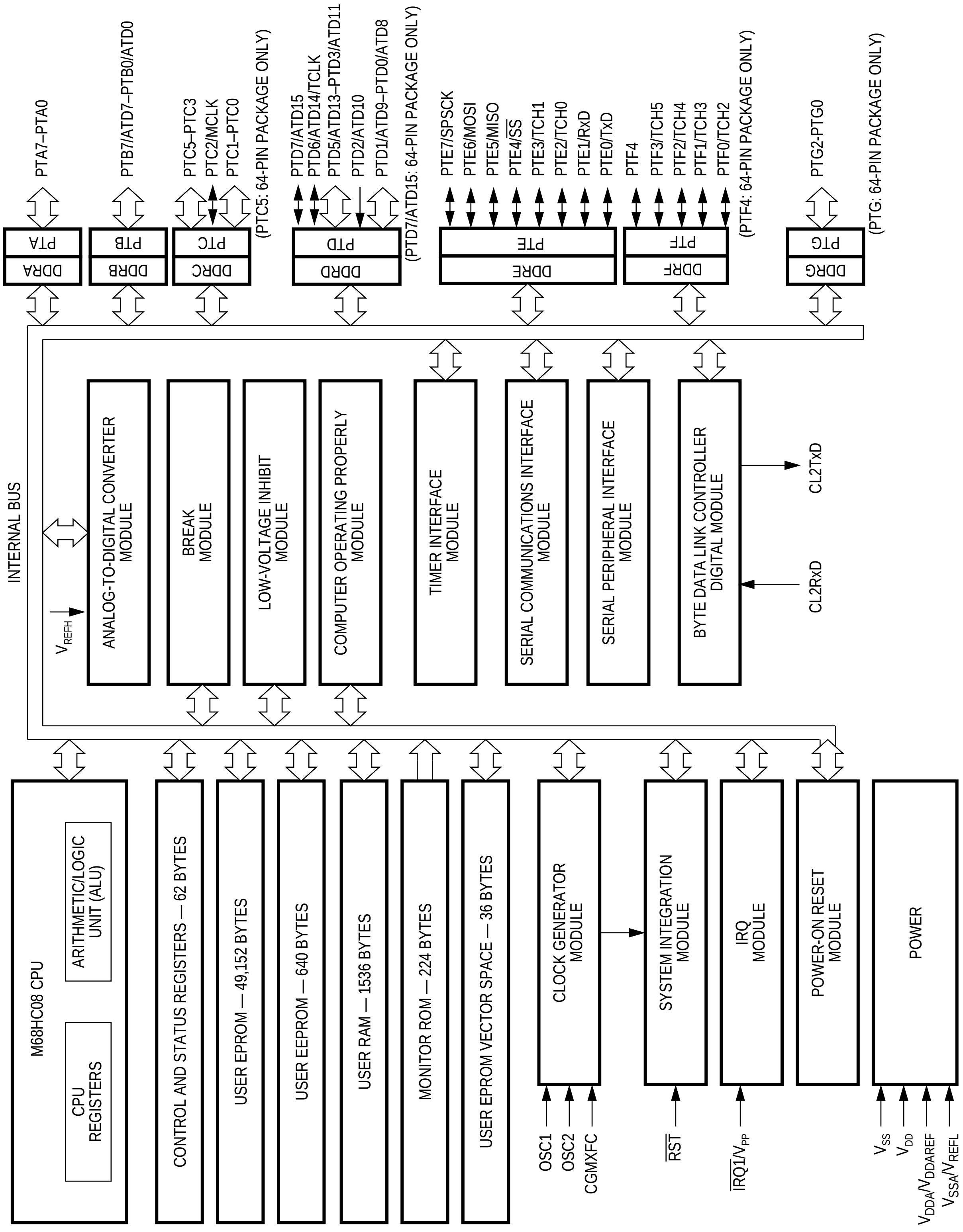


Figure 1-1. MCU Block Diagram

1.5 Pin Assignments

Figure 1-2 shows the QFP pin assignments, and **Figure 1-3** shows PLCC pin assignments.

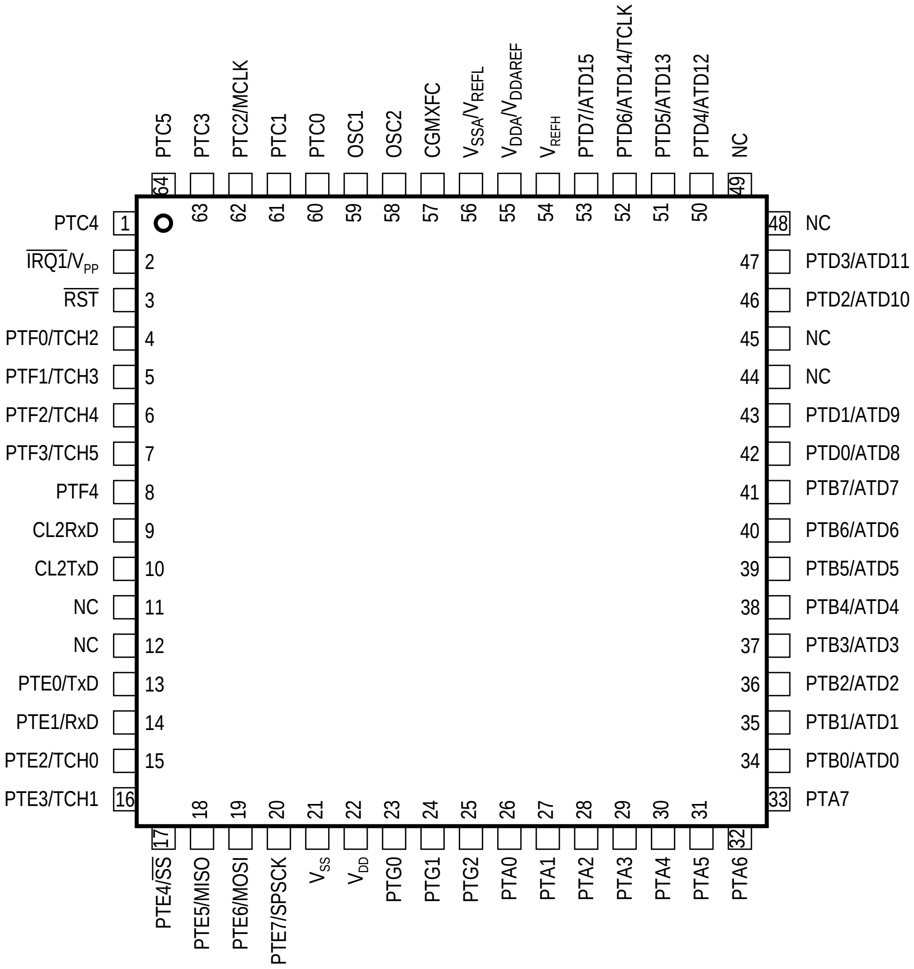


Figure 1-2. 64-Pin QFP Assignments (Top View)

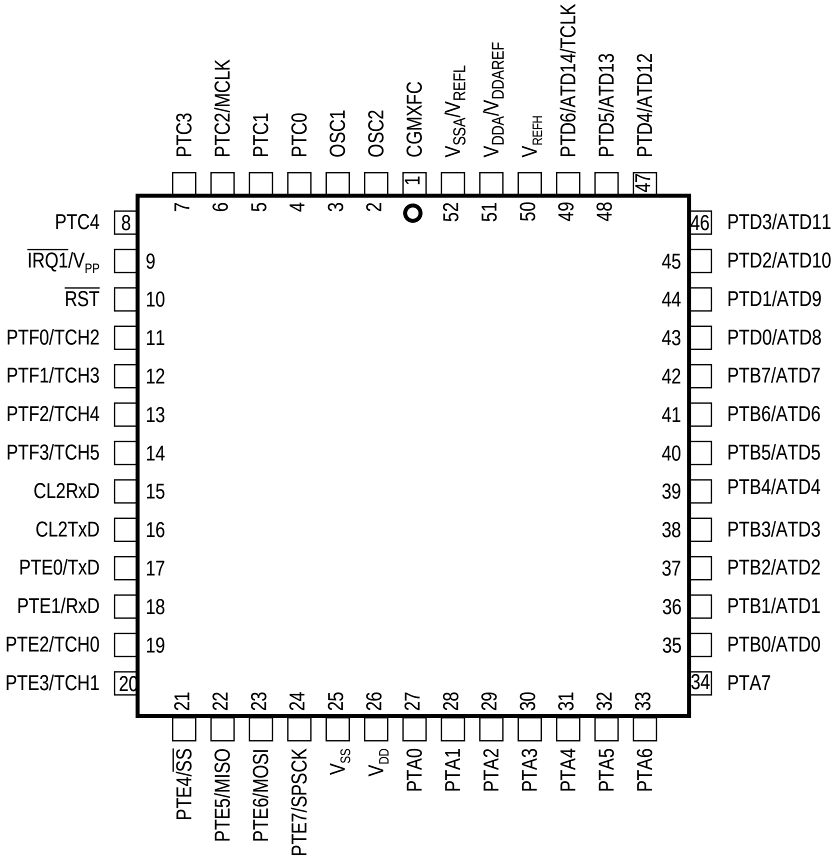
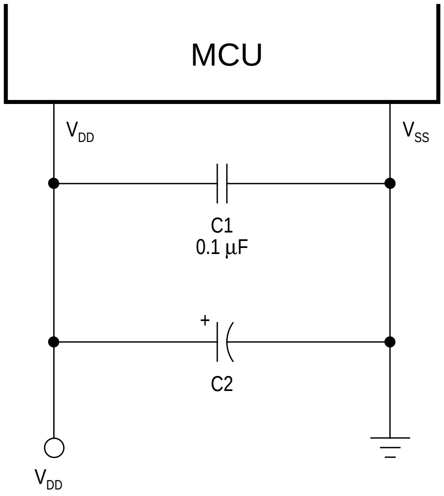


Figure 1-3. 52-Pin PLCC Assignments (Top View)

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

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

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



NOTE: Component values shown represent typical applications.

Figure 1-4. Power Supply Bypassing

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

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

1.5.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. (See **Section 8. Clock Generator Module (CGM)**.)

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

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

1.5.4 External Interrupt Pin ($\overline{IRQ1}$/V_{PP})

$\overline{\text{IRQ1}}$/V_{PP} is an asynchronous external interrupt pin. $\overline{\text{IRQ1}}$/V_{PP} is also the EPROM/OTPROM programming power pin. (See **Section 4. EPROM/OTPROM** and **Section 14. External Interrupt (IRQ)**.)

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

V_{DDA}/V_{DDAREF} is the power supply pin for the analog portion of the chip. These modules are the analog-to-digital converter and the clock generator module (CGM). (See **Section 8. Clock Generator Module (CGM)** and **Section 18. Analog-to-Digital Converter (ADC)**.)

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

The V_{SSA}/V_{REFL} analog ground pin is used only for the ground connections for the analog sections of the circuit and should be decoupled as per the V_{SS} digital ground pin. The analog sections consist of a clock generator module (CGM), and an analog to digital converter (ADC). (See **Section 8. Clock Generator Module (CGM)** and **Section 18. Analog-to-Digital Converter (ADC)**.)

1.5.7 V_{SS} External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. (See **Section 8. Clock Generator Module (CGM)**.)

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

PTA7–PTA0 are general-purpose bidirectional I/O port pins. (See **Section 19. Input/Output (I/O) Ports**.)

1.5.9 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 analog-to-digital converter (ADC). (See **Section 18. Analog-to-Digital Converter (ADC)** and **Section 19. Input/Output (I/O) Ports**.)

1.5.10 Port C I/O Pins (PTC5–PTC0)

PTC5–PTC3 and PTC1–PTC0 are general-purpose bidirectional I/O port pins. PTC2/MCLK is a special function port that shares its pin with the system clock. (See **Section 19. Input/Output (I/O) Ports**.) PTC5 is available only with the 64-pin package.

1.5.11 Port D I/O Pins (PTD7/ATD15–PTD0/ATD8)

Port D is an 8-bit special function port that shares all of its pins with the analog-to-digital converter module (ADC) and one of its pins with the timer interface module (TIM). (See **Section 17. Timer Interface (TIM)**, **Section 18. Analog-to-Digital Converter (ADC)**, and **Section 19. Input/Output (I/O) Ports**.) PTD7/ATD15 is available only with the 64-pin package.

1.5.12 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 timer interface module (TIM), four of its pins with the serial peripheral interface module (SPI), and two of its pins with the serial communication interface module (SCI). (See **Section 15. Serial Communications Interface (SCI)**, **Section 16. Serial Peripheral Interface (SPI)**, **Section 17. Timer Interface (TIM)**, and **Section 19. Input/Output (I/O) Ports**.)

1.5.13 Port F I/O Pins (PTF4–PTF0/TCH2)

Port F is a 5-bit special function port that shares four of its pins with the timer interface module (TIM). (See **Section 17. Timer Interface (TIM)** and **Section 19. Input/Output (I/O) Ports**.) Port F4 is only available with the 64-pin package.

1.5.14 Port G I/O Pins (PTG2–PTG0)

PTG2–PTG0 are general-purpose bidirectional I/O pins.
(See **Section 19. Input/Output (I/O) Ports**.) Port G is available only with the 64-pin package.

1.5.15 Class II Transmit Pin Digital (CL2TxD)

CL2TxD is a serial digital output data from the J1850 physical interface. (See **Section 20. Byte Data Link Controller-Digital (BDLC-D)**).

1.5.16 Class II Receive Pin Digital (CL2RxD)

CL2RxD is a serial digital input data from the J1850 physical interface. (See **Section 20. Byte Data Link Controller-Digital (BDLC-D)**).

Table 1-1. External Pins Summary

PIN NAME	FUNCTION	DRIVER TYPE	HYSTERESIS	RESET STATE
PTA7–PTA0	General-Purpose I/O	Dual State	No	Input (Hi-Z)
PTB7/ATD7–PTB0/ATD0	General-Purpose I/O ADC Channel	Dual State	No	Input (Hi-Z)
PTC5–PTC0	General-Purpose I/O	Dual State	No	Input (Hi-Z)
PTD7/ATD15	General-Purpose I/O ADC Channel	Dual State	No	Input (Hi-Z)
PTD6/ATD14/TCLK	General-Purpose I/O ADC Channel/Timer External Input Clock	Dual State	No	Input (Hi-Z)
PTD5/ATD13–PTD3/ATD11	General-PurposeI/O ADC Channel	Dual State	No	Input (Hi-Z)
PTD2/ATD10	General-Purpose Input ADC Channel	NA	No	Input (Hi-Z)
PTD1/ATD9–PTD0/ATD8	General-Purpose I/O ADC Channel	Dual State	No	Input (Hi-Z)
PTE7/SPSCK	General-Purpose I/O SPI Clock	Dual State (Open Drain)	Yes	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/$\overline{\text{SS}}$	General-Purpose I/O SPI Slave Select	Dual State	Yes	Input (Hi-Z)
PTE3/TCH1	General-Purpose I/O Timer Channel 1	Dual State	Yes	Input (Hi-Z)

Table 1-1. External Pins Summary (Continued)

PIN NAME	FUNCTION	DRIVER TYPE	HYSTERESIS	RESET STATE
PTE2/TCH0	General-Purpose I/O Timer Channel 0	Dual State	Yes	Input (Hi-Z)
PTE1/RxD	General-Purpose I/O SCI Receive Data	Dual State	Yes	Input (Hi-Z)
PTE0/TxD	General-Purpose I/O SCI Transmit Data	Dual State	Yes	Input (Hi-Z)
PTF4	General-Purpose I/O	Dual State	No	Input (Hi-Z)
PTF3/TCH5	General-Purpose I/O Timer Channel 5	Dual State	Yes	Input (Hi-Z)
PTF2/TCH4	General-Purpose I/O Timer Channel 4	Dual State	Yes	Input (Hi-Z)
PTF1/TCH3	General-Purpose I/O Timer Channel 3	Dual State	Yes	Input (Hi-Z)
PTF0/TCH2	General-Purpose I/O Timer Channel 2	Dual State	Yes	Input (Hi-Z)
PTG2–PTG0	General-Purpose I/O	Dual State	Yes	Input (Hi-Z)
V_{DD}	Chip Power Supply	NA	NA	NA
V_{SS}	Chip Ground	NA	NA	NA
V_{DDA}/V_{DDAREF}	Analog Power Supply (CGM and ADC)	NA	NA	NA
V_{SSA}/V_{REFL}	Analog Ground	NA	NA	NA
V_{REFH}	A/D Reference Voltage	NA	NA	NA
OSC1	External Clock In	NA	NA	Input (Hi-Z)
OSC2	External Clock Out	NA	NA	Output
CGMXFC	PLL Loop Filter Cap	NA	NA	NA
$\overline{\text{IRQ1}}/V_{PP}$	External Interrupt Request EPROM Programming Voltage	NA	NA	Input (Hi-Z)
$\overline{\text{RST}}$	Reset	NA	NA	Output Low
CL2RxD	BDLC-D Serial Input	NA	No	Input (Hi-Z)
CL2TxD	BDLC-D Serial Output	Output	No	Output Low

Table 1-2. Clock Source Summary

Module	Clock Source
ADC	CGMXCLK or Bus Clock
BDLC	CGMXCLK
COP	CGMXCLK
CPU	Bus Clock
EEPROM	CGMXCLK or Bus Clock
SPI	Bus Clock/SPSCK
SCI	CGMXCLK
TIM	Bus Clock or PTD6/ATD14/TCLK

Section 2. Memory Map

2.1 Contents

2.2 Introduction

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

- 48 Kbytes of EPROM or OTPROM
- 1.5 Kbytes of RAM
- 36 bytes of user-defined vectors
- 224 bytes of monitor ROM

$0000 ↓ $003F	I/O REGISTERS (62 BYTES) ($000B & $000F ARE NOT IMPLEMENTED)
$0040 ↓ $004F	UNIMPLEMENTED (16 BYTES)
$0050 ↓ $064F	RAM (1536 BYTES)
$0650 ↓ $07FF	UNIMPLEMENTED (432 BYTES)
$0800 ↓ $0A7F	EEPROM (640 BYTES)
$0A80 ↓ $3DFF	UNIMPLEMENTED (13,184 BYTES)
$3E00 ↓ $FDFF	EPROM (49,152 BYTES) ($AE00–$FDFF FOR 20,480 BYTES)
$FE00	SIM BREAK STATUS REGISTER (SBSR)
$FE01	SIM RESET STATUS REGISTER (SRSR)
$FE02	RESERVED
$FE03	SIM BREAK FLAG CONTROL REGISTER (SBFCR)
$FE04	RESERVED
$FE05	RESERVED
$FE06	UNIMPLEMENTED
$FE07	EPROM CONTROL REGISTER (EPMCR)

Figure 2-1. Memory Map

$FE08	RESERVED
$FE09	RESERVED
$FE0A	RESERVED
$FE0B	UNIMPLEMENTED
$FE0C	BREAK ADDRESS REGISTER HIGH (BRKH)
$FE0D	BREAK ADDRESS REGISTER LOW (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1B	UNIMPLEMENTED (12 BYTES)
$FE1C	EEPROM NON-VOLATILE REGISTER (EENVR)
$FE1D	EEPROM CONTROL REGISTER (EECR)
$FE1E	RESERVED
$FE1F	EEPROM ARRAY CONFIGURATION REGISTER (EEACR)
$FE20 ↓ $FEFF	MONITOR ROM (224 BYTES)
$FF00 ↓ $FFBF	UNIMPLEMENTED (192 BYTES)
$FFC0 ↓ $FFDB	RESERVED (28 BYTES)
$FFDC ↓ $FFFF	VECTORS (36 BYTES)

Figure 2-1. Memory Map (Continued)

2.3 I/O Section

Addresses $0000–$003F, shown in **Figure 2-2**, contain most of the control, status, and data registers. Additional I/O registers have these addresses:

- $FE00 (SIM break status register, SBSR)
- $FE01 (SIM reset status register, SRSR)
- $FE03 (SIM break flag control register, SBFCR)
- $FE07 (EPROM control register, EPMCR)
- $FE0C and $FE0D (break address registers, BRKH and BRKL)
- $FE0E (break status and control register, BRKSCR)
- $FE0F (LVI status register, LVISR)
- $FE1C (EEPROM non-volatile register, EENVR)
- $FE1D (EEPROM control register, EECR)
- $FE1F (EEPROM array configuration register, EEACR)
- $FFFF (COP control register, COPCTL)

NOTE: *DMA section and associated functions are valid only if the MCU has a DMA module.*

Table 2-1 is a list of vector locations.

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	R:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		W:								
$0001	Port B Data Register (PTB)	R:	PTB7	PTB6	PTB25	PTB4	PTB3	PTB2	PTB1	PTB0
		W:								
$0002	Port C Data Register	R:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		W:	▨	▨						
$0003	Port D Data Register (PTD)	R:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		W:								
$0004	Data Direction Register A (DDRA)	R:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		W:								
$0005	Data Direction Register B (DDRB)	R:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		W:								
$0006	Data Direction Register C (DDRC)	R:	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		W:		▨						
$0007	Data Direction Register D (DDRD)	R:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	0	DDRD1	DDRD0
		W:						▨		
$0008	Port E Data Register (PTE)	R:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		W:								
$0009	Port F Data Register (PTF)	R:	0	0	0	PTF4	PTF3	PTF2	PTF1	PTF0
		W:	▨	▨	▨					
$000A	Port G Data Register (PTG)	R:	0	0	0	0	0	PTG2	PTG1	PTG0
		W:	▨	▨	▨	▨	▨			
$000B	Unimplemented	R:	▨	▨	▨	▨	▨	▨	▨	▨
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$000C	Data Direction Register E (DDRE)	R:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		W:								
$000D	Data Direction Register F (DDRF)	R:	0	0	0	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		W:	▨	▨	▨					
$000E	Data Direction Register G (DDRG)	R:	0	0	0	0	0	DDRG2	DDRG1	DDRG0
		W:	▨	▨	▨	▨	▨			
$000F	Unimplemented	R:	▨	▨	▨	▨	▨	▨	▨	▨
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$0010	SPI Control Register (SPCR)	R:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		W:								
$0011	SPI Status and Control Register (SPSCR)	R:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		W:	▨		▨	▨	▨			
$0012	SPI Data Register (SPDR)	R:	R7	R6	R5	R4	R3	R2	R1	R0
		W:	T7	T6	T5	T4	T3	T2	T1	T0
$0013	SCI Control Register 1 (SCC1)	R:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		W:								
$0014	SCI Control Register 2 (SCC2)	R:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		W:								

▨ = Unimplemented   R = Reserved

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

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$0015	SCI Control Register 3 (SCC3)	R:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		W:	▨							
$0016	SCI Status Register 1 (SCS1)	R:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$0017	SCI Status Register 2 (SCS2)	R:	0	0	0	0	0	0	BKF	RPF
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$0018	SCI Data Register (SCDR)	R:	R7	R6	R5	R4	R3	R2	R1	R0
		W:	T7	T6	T5	T4	T3	T2	T1	T0
$0019	SCI Baud Rate Register (SCBR)	R:	0	0	SCP1	SCP0	▨	SCR2	SCR1	SCR0
		W:	▨	▨						
$001A	IRQ Status and Control Register (ISCR)	R:	0	0	0	0	IRQF1	0	IMASK1	MODE1
		W:	▨	▨	▨	▨	▨	ACK1		
$001B	Reserved	R:	▨	▨	▨	▨	▨	▨	▨	▨
		W:								
$001C	PLL Control Register (PCTL)	R:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
		W:		▨			▨	▨	▨	▨
$001D	PLL Bandwidth Control Register (PBWC)	R:	AUTO	LOCK	$\overline{\text{ACQ}}$	XLD	0	0	0	0
		W:		▨			▨	▨	▨	▨
$001E	PLL Programming Register (PPG)	R:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		W:								
$001F	Configuration Register (CONFIG)	R:			LVIRSTD	LVIPWRD	SSREC	COPL	STOP	COPD
		W:								
$0020	Timer Status and Control Register (TSC)	R:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		W:	0			TRST	▨			
$0021	Timer DMA Select Register (TDMA)	R:	0	0	DMA5S	DMA4S	DMA3S	DMA2S	DMA1S	DMA0S
		W:	▨	▨						
$0022	Timer Counter Register High (TCNTH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$0023	Timer Counter Register Low (TCNTL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$0024	Timer Modulo Register High (TMODH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0025	Timer Modulo Register Low (TMODL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0026	Timer Channel 0 Status and Control Register (TSC0)	R:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		W:	0							
$0027	Timer Channel 0 Register High (TCH0H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0028	Timer Channel 0 Register Low (TCH0L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0029	Timer Channel 1 Status and Control Register (TSC1)	R:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		W:	0		▨					

▨ = Unimplemented R = Reserved

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

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$002A	Timer Channel 1 Register High (TCH1H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$002B	Timer Channel 1 Register Low (TCH1L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$002C	Timer Channel 2 Status and Control Register (TSC2)	R:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		W:	0							
$002D	Timer Channel 2 Register High (TCH2H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$002E	Timer Channel 2 Register Low (TCH2L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$002F	Timer Channel 3 Status and Control Register (TSC3)	R:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		W:	0		▨					
$0030	Timer Channel 3 Register High (TCH3H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0031	Timer Channel 3 Register Low (TCH3L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0032	Timer Channel 4 Status and Control Register (TSC4)	R:	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX
		W:	0							
$0033	Timer Channel 4 Register High (TCH4H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0034	Timer Channel 4 Register Low (TCH4L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0035	Timer Channel 5 Status and Control Register (TSC5)	R:	CH5F	CH5IE	0	MS5A	ELS5B	ELS5A	TOV5	CH5MAX
		W:	0		▨					
$0036	Timer Channel 5 Register High (TCH5H)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$0037	Timer Channel 5 Register Low (TCH5L)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$0038	ADSCR	R:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		W:								
$0039	ADR	R:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$003A	ADCLK	R:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		W:					▨	▨	▨	▨
$003B	BARD	R:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
		W:			▨	▨				
$003C	BCR1	R:	IMSG	CLKS	R1	R0	0	0	IE	WCM
		W:					▨	▨		
$003D	BCR2	R:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
		W:								

▨ = Unimplemented R = Reserved

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

Addr.	Name		Bit 7	6	5	4	3	2	1	Bit 0
$003E	BSVR	R:	0	0	I3	I2	I1	I0	0	0
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$003F	BDR	R:	BD7	BD6	BD5	BD4	BD3	BD2	BD1	BD0
		W:								
$FE00	SIM Break Status Register (SBSR)	R:	R	R	R	R	R	R	SBSW	R
		W:								
$FE01	SIM Reset Status Register (SRSR)	R:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$FE03	SIM Break Flag Control Register (SBFCR)	R:	BCFE	R	R	R	R	R	R	R
		W:								
$FE07	EPROM Control Register (EPMCR)	R:	0	0	0	0	0	ELAT	0	EPGM
		W:	▨	▨	▨	▨	▨		▨	
$FE0C	Break Address Register High (BRKH)	R:	Bit 15	14	13	12	11	10	9	Bit 8
		W:								
$FE0D	Break Address Register Low (BRKL)	R:	Bit 7	6	5	4	3	2	1	Bit 0
		W:								
$FE0E	Break Status and Control Register (BRKSCR)	R:	BRKE	BRKA	0	0	0	0	0	0
		W:			▨	▨	▨	▨	▨	▨
$FE0F	LVI Status Register (LVISR)	R:	LVIOUT	0	0	0	LVISTOP	LVILCK	0	0
		W:	▨	▨	▨	▨			▨	▨
$FE1C	EENVR	R:	EERA	1	1	1	EEBP3	EEBP2	EEBP1	EEBP0
		W:		▨	▨	▨				
$FE1D	EECR	R:	EEBCLK	0	EEOFF	EERAS1	EERAS0	EELAT	0	EEPGM
		W:		▨					▨	
$FE1E	RESERVED	R:	▨	▨	▨	▨	▨	▨	▨	▨
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$FE1F	EEACR	R:	EERA	1	1	1	EEBP3	EEBP2	EEBP1	EEBP0
		W:	▨	▨	▨	▨	▨	▨	▨	▨
$FFFF	COP Control Register (COPCTL)	R:	LOW BYTE OF RESET VECTOR							
		W:	WRITING TO $FFFF CLEARS COP COUNTER							

▨ = Unimplemented   R = Reserved

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

Table 2-1. Vector Addresses

Low

Priority

High

Address	Vector
$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)
$FFFF	Reset Vector (Low)

Section 3. Random-Access Memory (RAM)

3.1 Contents

3.2 Introduction

This section describes the 1536 bytes of random-access memory (RAM).

3.3 Functional Description

Addresses $0050 through $064F are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64-Kbyte memory space.

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

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

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

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

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

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

Section 4. EPROM/OTPROM

4.1 Contents

4.2 Introduction

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

4.3 Functional Description

An MCU with a quartz window has 48 Kbytes of erasable, programmable ROM (EPROM). The quartz window allows EPROM erasure by using ultraviolet light. In an MCU without the quartz window, the EPROM cannot be erased and serves as 48 Kbytes of one-time programmable ROM (OTPROM). An unprogrammed or erased location reads as $00. Hardware interlocks are provided to protect stored data corruption from accidental programming. The following addresses are user EPROM/OTPROM locations:

- $3E00–$FDFF
- $FFDC–$FFFF (These locations are reserved for user-defined interrupt and reset vectors.)

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

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

4.4 EPROM/OTPROM Control Register (EPMCR)

The EPROM control register controls EPROM/OTPROM programming.

		Bit 7	6	5	4	3	2	1	Bit 0
EPMCR $FE07	Read:	0	0	0	0	0	ELAT	0	EPGM
	Write:	▨	▨	▨	▨	▨		▨	
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

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

ELAT — EPROM/OTPROM Latch Control Bit

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

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

EPGM — EPROM/OTPROM Program Control Bit

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

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

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

4.5 EPROM/OTPROM Programming

The unprogrammed state is a zero. Programming changes the state to a one.

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

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

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

3. Write to any user EPROM/OTPROM address.

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

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

Setting the ELAT bit configures the address and data buses to latch data for programming the array. Only data written to a valid EPROM address will be latched. Attempts to read any other valid EPROM address after step 2 will read the latched data written in step 3. Further writes to valid EPROM addresses after the first write (step 3) are ignored.

The EPGM bit cannot be set if the ELAT bit is cleared. This is to ensure a proper programming sequence. If EPGM is set and a valid EPROM write occurs, V_{PP} will be applied to the user EPROM array. When the EPGM bit is cleared, the program voltage is removed from the array.

4.6 Low-Power Modes

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

4.6.1 Wait Mode

The WAIT instruction does not affect the EPROM.

4.6.2 Stop Mode

The STOP instruction reduces the EPROM power consumption to a minimum. If STOP occurs while EPROM programming is in progress, high voltage will be removed from the array and the EPGM bit will be cleared. To resume programming, clear the ELAT bit and follow the steps shown in **4.5 EPROM/OTPROM Programming**.

Section 5. Electrically Erasable Programmable ROM (EEPROM)

5.1 Contents

5.2 Introduction

This section describes the electrically erasable programmable ROM (EEPROM).

5.3 Features

EEPROM features include:

- Modular Architecture Expandable in 128 Bytes
- Byte, Block, or Bulk Erasable
- Non-Volatile Redundant Array Option
- Non-Volatile Block Protection Option
- On-Chip Charge Pump for Programming/Erasing

5.4 Functional Description

A total of 640 bytes of EEPROM can be programmed or erased without an external voltage supply. The EEPROM has a lifetime of 10,000 write-erase cycles in the non-redundant mode. Reliability (data retention) is further extended if the redundancy option is selected. EEPROM cells are protected with a non-volatile block protection option. These options are stored in the EEPROM non-volatile register (EENVR) and are loaded into the EEPROM array configuration register after reset (EEACR) or a read of EENVR. Hardware interlocks are provided to protect stored data corruption from accidental programming/erasing.

5.4.1 EEPROM Programming

The unprogrammed state is a logic one. Programming changes the state to a logic zero. Only valid EEPROM bytes in the non-protected blocks and EENVR can be programmed. When the array is configured in the redundant mode, programming the first 128 bytes will also program the last 128 bytes with the same data. Programming the EEPROM in the non-redundant mode is recommended. Program the data to both locations before entering the redundant mode.

The following procedure shows how to program a byte of EEPROM:

1. Clear EERAS1 and EERAS0 and set EELAT in the EECTL. Set value of t_{EEPGM}. (See Note a. and b.)
2. Write the desired data to any user EEPROM address.
3. Set the EEPGM bit. (See Note c.)
4. Wait for a time, t_{EEPGM}, to program the byte.
5. Clear the EEPGM bit.
6. Wait for the programming voltage time, t_{EEPGM}, to fall.
7. Clear EELAT bits. (See Note d.)
8. Repeat steps 1 through 7 for more EEPROM programming.

NOTES:

a. EERAS1 and EERAS0 must be cleared for programming. Otherwise, EEPROM will be in erase mode.

b. Setting 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 another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. Any attempts to read other EEPROM data will read the latched data. If EELAT is set, other writes to the EECR will be allowed only after a valid EEPROM write.

c. The EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. This is to ensure a proper programming sequence. When EEPGM is set, the on-board charge pump generates the program voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the program voltage is removed from the array and the internal charge pump is turned off.

d. Any attempt to clear both EEPGM and EELAT bits with a single instruction will clear EEPGM only. This allows time to remove high voltage from the EEPROM array.

5.4.2 EEPROM Erasing

The unprogrammed state is a logic one. Only the valid EEPROM bytes in the non-protected blocks and EENVR can be erased. When the array is configured in the redundant mode, erasing the first 128 bytes will also erase the last 128 bytes.

The following procedure shows how to erase EEPROM:

1. Clear/set EERAS1 and EERAS0 to select byte/block/bulk erase, and set EELAT in EECTL. Set value of t_{EEBYT} / $t_{EEBLOCK}$ / t_{EEBULK}. (See Note a.)
2. Write any data to the desired address for byte erase, to any address in the desired block for block erase, or to any array address for bulk erase.
3. Set the EEPGM bit. (See Note b.)
4. Wait for a time, t_{EEBYT}/$t_{EEBLOCK}$/t_{EEBULK}, to erase the byte.
5. Clear the EEPGM bit.
6. Wait for the erasing voltage time, t_{EEFPV}, to fall.
7. Clear the EELAT bits. (See Note c.)
8. Repeat steps 1 through 7 for more EEPROM byte/block erasing.

NOTES:

a. Setting the EELAT bit configures the address and data buses to latch data for erasing the array. Only valid EEPROM addresses with their data will be latched. If another consecutive valid EEPROM write occurs, this address and data will override the previous address and data. In block erase mode, any EEPROM address in the block may be used in step 2. All locations within this block will be erased. In bulk erase mode, any EEPROM address may be used to erase the whole EEPROM. EENVR is not affected with block or bulk erase. Any attempts to read other EEPROM data will read the latched data. If EELAT is set, other writes to the EECR will be allowed only after a valid EEPROM write.

b. To ensure proper erasing sequences, the EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. Once the EEPGM bit is set, the type of erase mode cannot be modified. If EEPGM is set, the on-board charge pump generates the erase voltage and applies it to the user EEPROM array. When the EEPGM bit is cleared, the erase voltage is removed from the array and the internal charge pump is turned off.

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

The EEBPx bit must be cleared to erase EEPROM data in the corresponding block. If any EEBPx is set, the corresponding block cannot be block nor bulk eraseable.

In general, all bits should be erased before being programmed. However, if program/erase cycling is of concern, the following procedure can be used to minimize bit cycling in each EEPROM byte. If any bit in a byte must be changed from a zero to a one, the byte needs be erased before programming. **Table 5-1** summarizes the conditions for erasing before programming

Table 5-1. EEPROM Program/Erase Cycling Reduction

EEPROM Data To Be Programmed	EEPROM Data Before Programming	Erase Before Programming?
0	0	No
0	1	No
1	0	Yes
1	1	No

5.4.3 EEPROM Block Protection

The 640 bytes of EEPROM are divided into two 256 and one128 byte blocks. Each of these blocks can be separately protected by the EEBPx bit. Any attempt to program or erase memory locations within the protected block will not allow the program/erase voltage to be applied to the array. **Table 5-2** shows the address ranges within the blocks.

Table 5-2. EEPROM Array Address Blocks

Block Number (EEBPx)	Address Range
EEBP0	$0800–$08FF
EEBP1	$0900–$09FF
EEBP2	$0A00–$0A7F

If the EEBPx bit is set, that corresponding address block is protected. These bits are effective after a reset or a read to 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.

In redundant mode, only the EEBP2 will not be valid.

5.4.4 EEPROM Redundant Mode

To extend the EEPROM data retention, the array can be placed in redundant mode. In this mode, the first 128 bytes of user EEPROM array are mapped to the last 128 bytes. Reading, programming, and erasing of the first 128 EEPROM bytes will physically affect two bytes of EEPROM. Addressing the last 128 bytes will not be recognized. Block protection still applies but EEBP2 is not valid.

NOTE: *Programming the EEPROM in the non-redundant mode and programming the data to its corresponding location before entering the redundant mode are recommended.*

5.4.5 EEPROM Configuration

The EEPROM non-volatile register (EENVR) contains configurations concerning block protection and redundancy. EENVR is physically located on the bottom of the EEPROM array. The contents are non-volatile and are not modified by reset. On reset, this special register loads the EEPROM configuration into a corresponding volatile EEPROM array configuration register (EEACR). Thereafter, all reads to the EENVR will reload EEACR.

The EEPROM configuration can be changed by programming/erasing the EENVR like a normal EEPROM byte. The new array configuration will take affect with a system reset or a read of the EENVR.

5.4.6 EEPROM Control Register (EECR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
EECR $FE1D	Read:	EEBCLK	0	EEOFF	EERAS1	EERAS0	EELAT	0	EEPGM
	Write:		▨					▨	
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 5-1. EEPROM Control Register (EECR)

EEBCLK — EEPROM Bus Clock Enable

This read/write bit determines which clock will be used to drive the internal charge pump for programming/erasing. Reset clears this bit.

1 = Bus clock drives charge pump
0 = Internal RC oscillator drives charge pump

NOTE: *Using the internal RC oscillator for applications in the 3–5 volt range is recommended.*

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

NOTE: *The EEPROM requires a recovery time, t_{EEOFF}, to stabilize after clearing the EEOFF bit.*

EERAS1 and EERAS0 — Erase Bits

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

Table 5-3. EEPROM Program/Erase Mode Select

EEBPx	EERAS1	EERA0	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
0 = Buses configured for normal read operation

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 logic zeroes to both the EELAT and EEPGM bits with a single instruction will clear only the EEPGM and will allow time to remove high voltage.*

5.4.7 EEPROM Non-Volatile Register and EEPROM Array Configuration Register

		Bit 7	6	5	4	3	2	1	Bit 0
EEACR $FE1F	Read:	EERA	1	1	1	EEBP3	EEBP2	EEBP1	EEBP0
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	EENVR	1	1	1	EENVR	EENVR	EENVR	EENVR

▨ = Unimplemented

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

		Bit 7	6	5	4	3	2	1	Bit 0
EENVR $FE1C	Read:	EERA	1	1	1	EEBP3	EEBP2	EEBP1	EEBP0
	Write:		▨	▨	▨				
	Reset:	PV	1	1	1	PV	PV	PV	PV

▨ = Unimplemented

Figure 5-3. EEPROM Non-Volatile Register (EENVR)

PV = Programmed Value or logic one in the erased state.

EERA — EEPROM Redundant Array

This programmable/erasable/readable bit in EENVR and read-only bit in EEACR configures the array in redundant mode. Reset loads EERA from EENVR to EEACR.

1 = EEPROM array is in redundant mode configuration
0 = EEPROM array is in normal mode configuration

EEBP2–EEBP0 — EEPROM Block Protection Bits.

These read/write bits select blocks of EEPROM array from being programmed or erased. Reset loads EEBP[4:0] from EENVR to EEACR.

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

5.4.8 Low-Power Modes

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

Wait Mode

The WAIT instruction does not affect the EEPROM. It is possible to program the EEPROM and put the MCU in wait mode. However, if the EEPROM is inactive, power can be reduced by setting the EEOFF bit before executing the WAIT instruction.

Stop Mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while the high voltage is turned on (EEPGM = 1).

If stop mode is entered while program/erase is in progress, high voltage will be automatically turned off. However, the EEPGM bit will remain set. When stop mode is terminated, and if EEPGM is still set, the high voltage will be automatically turned back on. Program/erase time will need to be extended if program/erase is interrupted by entering stop mode.

The module requires a recovery time, t_{EESTOP}, to stabilize after leaving stop mode. Attempts to access the array during the recovery time will result in unpredictable behavior.

Section 6. Central Processor Unit (CPU)

6.1 Contents

6.2 Introduction

This section describes the central processor unit (CPU8). The M68HC08 CPU is an enhanced and fully object-code-compatible version of the M68HC05 CPU. The *CPU08 Reference Manual* (Motorola document number CPU08RM/AD) contains a description of the CPU instruction set, addressing modes, and architecture.

6.3 Features

Features of the CPU include the following:

- Full Upward, Object-Code Compatibility with the M68HC05 Family
- 16-Bit Stack Pointer with Stack Manipulation Instructions
- 16-Bit Index Register with X-Register Manipulation Instructions
- 8-MHz CPU Internal Bus Frequency
- 64-Kbyte Program/Data Memory Space
- 16 Addressing Modes
- Memory-to-Memory Data Moves Without Using Accumulator
- Fast 8-Bit by 8-Bit Multiply and 16-Bit by 8-Bit Divide Instructions
- Enhanced Binary-Coded Decimal (BCD) Data Handling
- Modular Architecture with Expandable Internal Bus Definition for Extension of Addressing Range Beyond 64 Kbytes
- Low-Power Stop and Wait Modes

6.4 CPU Registers

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

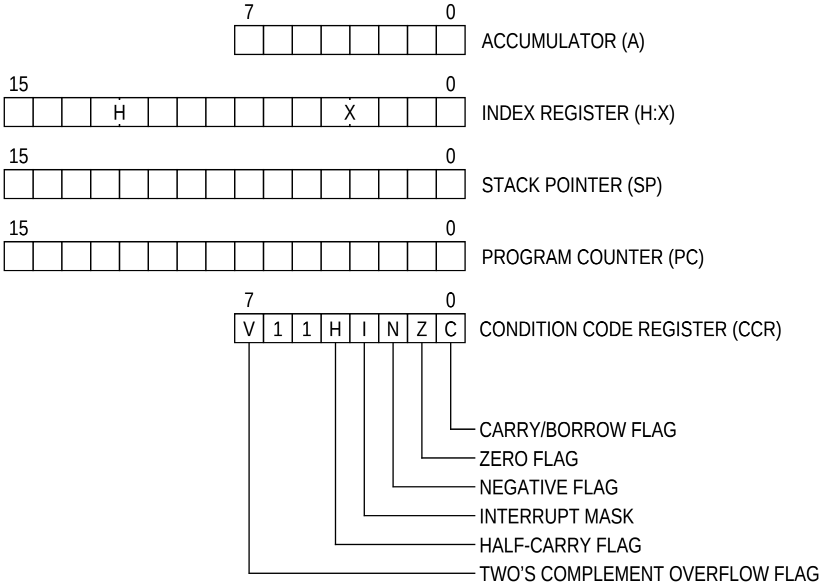


Figure 6-1. CPU Registers

6.4.1 Accumulator (A)

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

		Bit 7	6	5	4	3	2	1	Bit 0
A	Read:								
	Write:								
	Reset:	Unaffected by reset							

Figure 6-2. Accumulator (A)

6.4.2 Index Register (H:X)

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

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

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

X = Indeterminate

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

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

6.4.3 Stack Pointer (SP)

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

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

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

Figure 6-4. Stack Pointer (SP)

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

6.4.4 Program Counter (PC)

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

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

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

		Bit 15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit 0
PC	Read:																
	Write:																
	Reset:	Loaded with vector from $FFFE and $FFFF															

Figure 6-5. Program Counter (PC)

6.4.5 Condition Code Register (CCR)

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

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

X = Indeterminate

Figure 6-6. Condition Code Register (CCR)

V — Overflow Flag

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

1 = Overflow
0 = No overflow

H — Half-Carry Flag

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

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

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

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

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

N — Negative flag

The CPU sets the negative flag when an arithmetic operation, logic operation, or data manipulation produces a negative result, setting bit 7 of the result.
1 = Negative result
0 = Non-negative result

Z — Zero flag

The CPU sets the zero flag when an arithmetic operation, logic operation, or data manipulation produces a result of $00.
1 = Zero result
0 = Non-zero result

C — Carry/Borrow Flag

The CPU sets the carry/borrow flag when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some instructions — such as bit test and branch, shift, and rotate — also clear or set the carry/borrow flag.
1 = Carry out of bit 7
0 = No carry out of bit 7

6.5 Arithmetic/Logic Unit (ALU)

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

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

6.6 CPU During Break Interrupts

If the break module is enabled, a break interrupt causes the CPU to execute the software interrupt instruction (SWI) at the completion of the current CPU instruction. (See **Section 10. Break Module (Break)**.) The program counter vectors to $FFFC–$FFFD ($FEFC–$FEFD in monitor mode).

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

Section 7. System Integration Module (SIM)

7.1 Contents

7.2 Introduction

This section describes the system integration module (SIM), which supports up to 24 external and/or internal interrupts. Together with the central processor unit (CPU), the SIM controls all MCU activities. A block diagram of the SIM is shown in **Figure 7-1**. **Table 7-1** is a summary of the SIM input/output (I/O) registers. The SIM is a system state controller that coordinates CPU and exception timing. The SIM is responsible for:

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

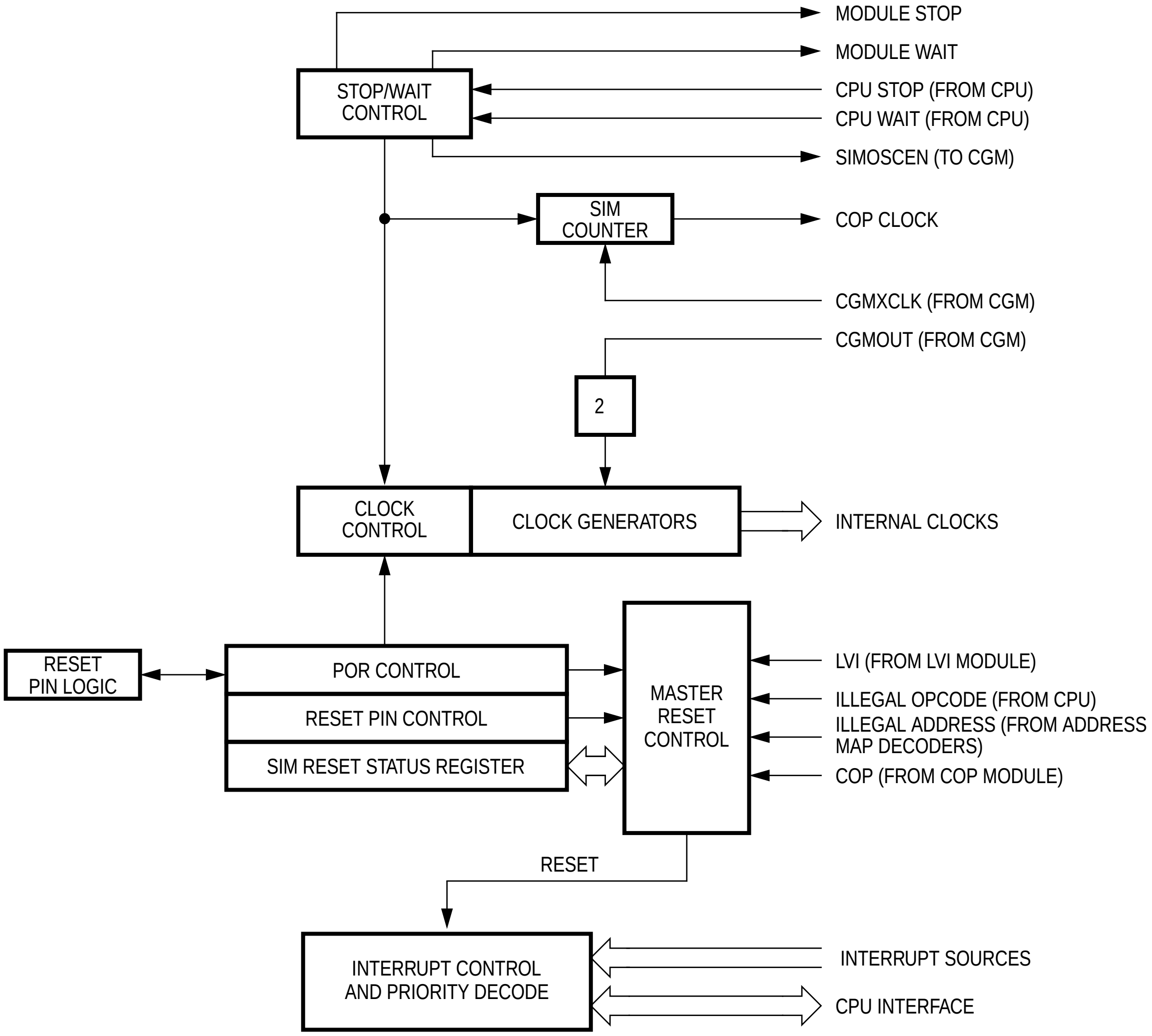


Figure 7-1. SIM Block Diagram

Table 7-1. SIM I/O Register Summary

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

R	= Reserved

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

Table 7-2. Signal Name Conventions

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

7.3 SIM Bus Clock Control and Generation

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

7.3.1 Bus Timing

In user mode**,** the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four. (See **Section 8. Clock Generator Module (CGM)**.)

7.3.2 Clock Start-Up from POR or LVI Reset

When the power-on reset module or the low-voltage inhibit module generates a reset, the clocks to the CPU and peripherals are inactive and held in an inactive phase until after 4096 CGMXCLK cycles. The $\overline{\text{RST}}$ pin is driven low by the SIM during this entire period. The bus clocks start upon completion of the timeout.

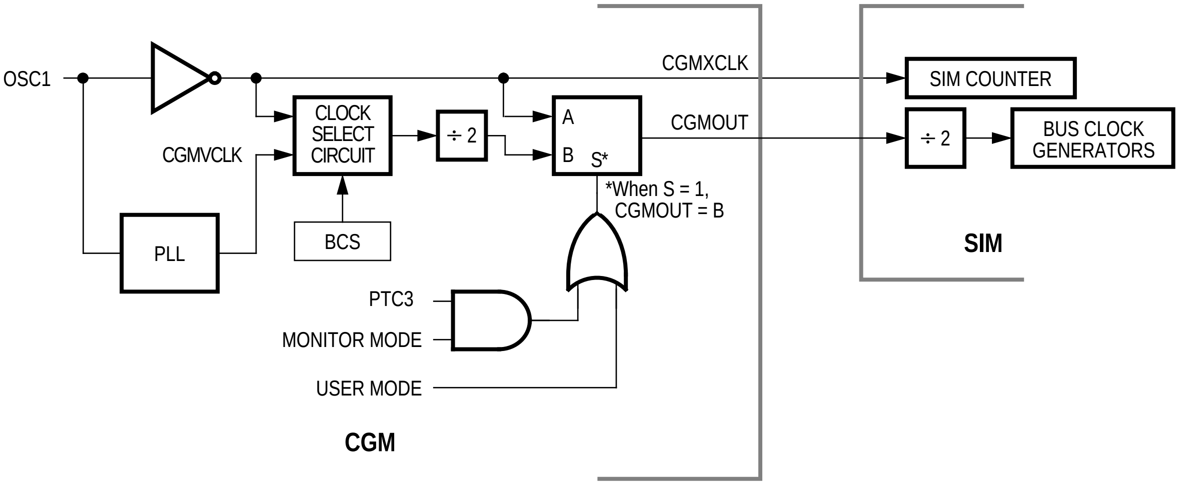


Figure 7-2. CGM Clock Signals

7.3.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. (See **7.7.2 Stop Mode**.)

In wait mode, the CPU clocks are inactive. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

7.4 Reset and System Initialization

The MCU has these reset sources:

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

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

An internal reset clears the SIM counter (see **7.5 SIM Counter**), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See **7.8 SIM Registers**.)

7.4.1 External Pin Reset

Pulling the asynchronous $\overline{\text{RST}}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{\text{RST}}$ is held low for a minimum of 67 CGMXCLK cycles, assuming that neither the POR nor the LVI was the source of the reset. See **Table 7-3** for details. **Figure 7-3** shows the relative timing.

Table 7-3. PIN Bit Set Timing

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

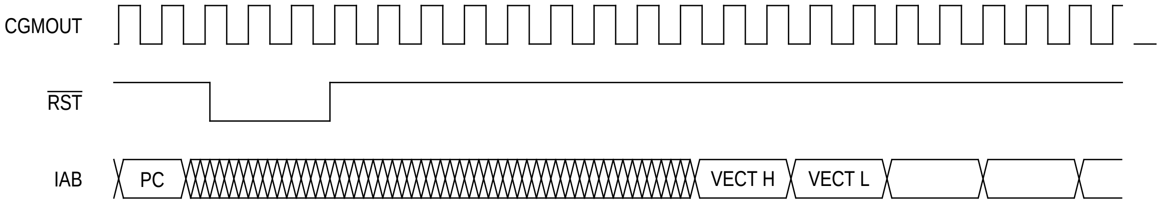


Figure 7-3. External Reset Timing

7.4.2 Active Resets from Internal Sources

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

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.

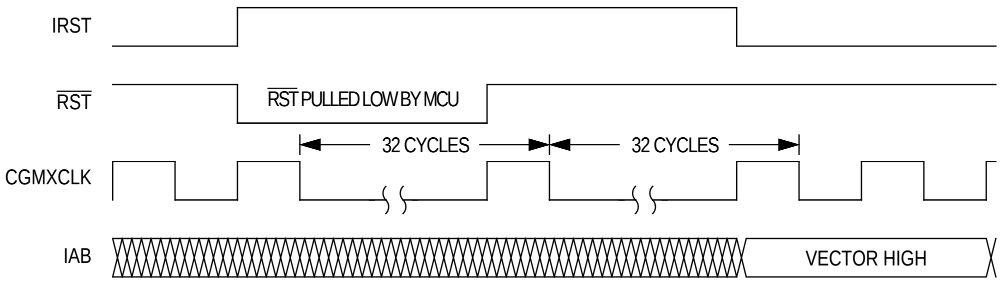


Figure 7-4. Internal Reset Timing

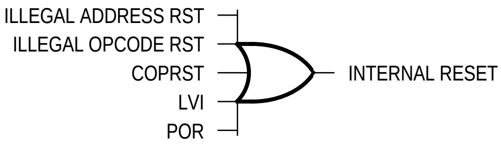


Figure 7-5. Sources of Internal Reset

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 ($\overline{\text{RST}}$) 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.

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 $\overline{\text{RST}}$ pin is driven low during the oscillator stabilization time.
- The POR bit of the SIM reset status register (SRSR) is set and all other bits in the register are cleared.

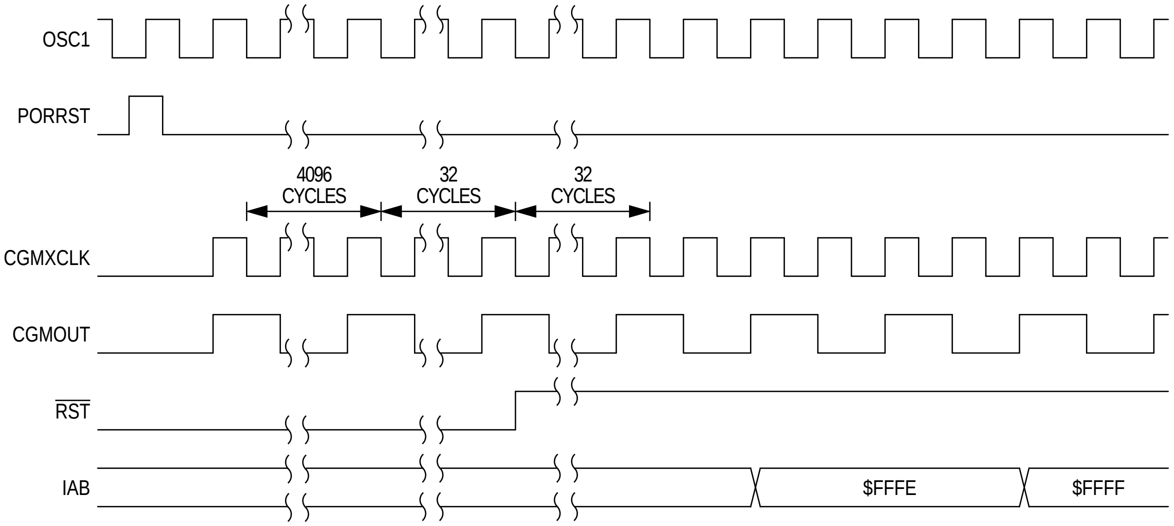


Figure 7-6. POR Recovery

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 configuration register is at logic zero. (See **Section 12. Computer Operating Properly (COP)**.)

Illegal Opcode Reset

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

If the stop enable bit, STOP, in the configuration register is logic zero, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset.

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.

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 CONFIG register are at logic zero. The $\overline{\text{RST}}$ 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 **Section 13. Low-Voltage Inhibit (LVI)**.)

7.5 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. 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.

7.5.1 SIM Counter During Power-On Reset

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

7.5.2 SIM Counter During Stop Mode Recovery

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, break, or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the configuration register. If the SSREC bit is a logic one, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

7.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See **7.7.2 Stop Mode** for details.) The SIM counter is free-running after all reset states. (See **7.4.2 Active Resets from Internal Sources** for counter control and internal reset recovery sequences.)

7.6 Program Exception Control

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

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

7.6.1 Interrupts

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

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared). (See **Figure 7-8**.)

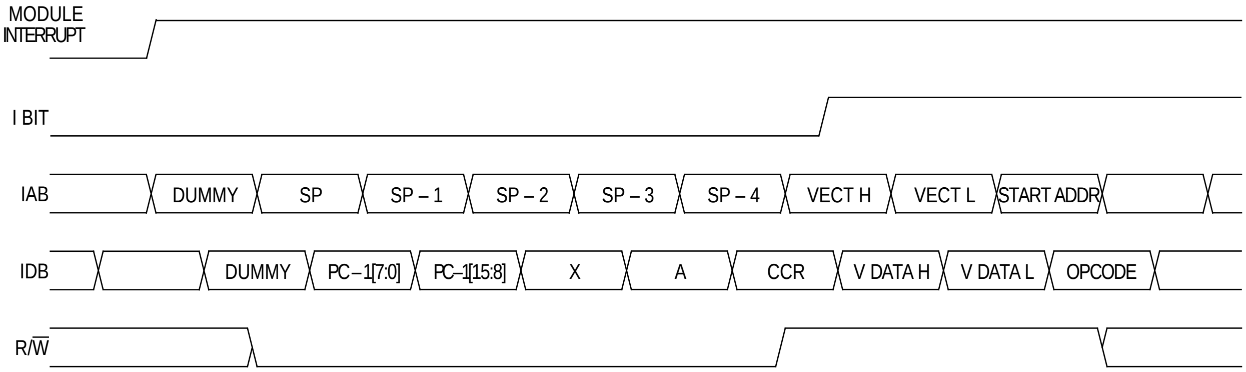


Figure 7-7. Interrupt Entry

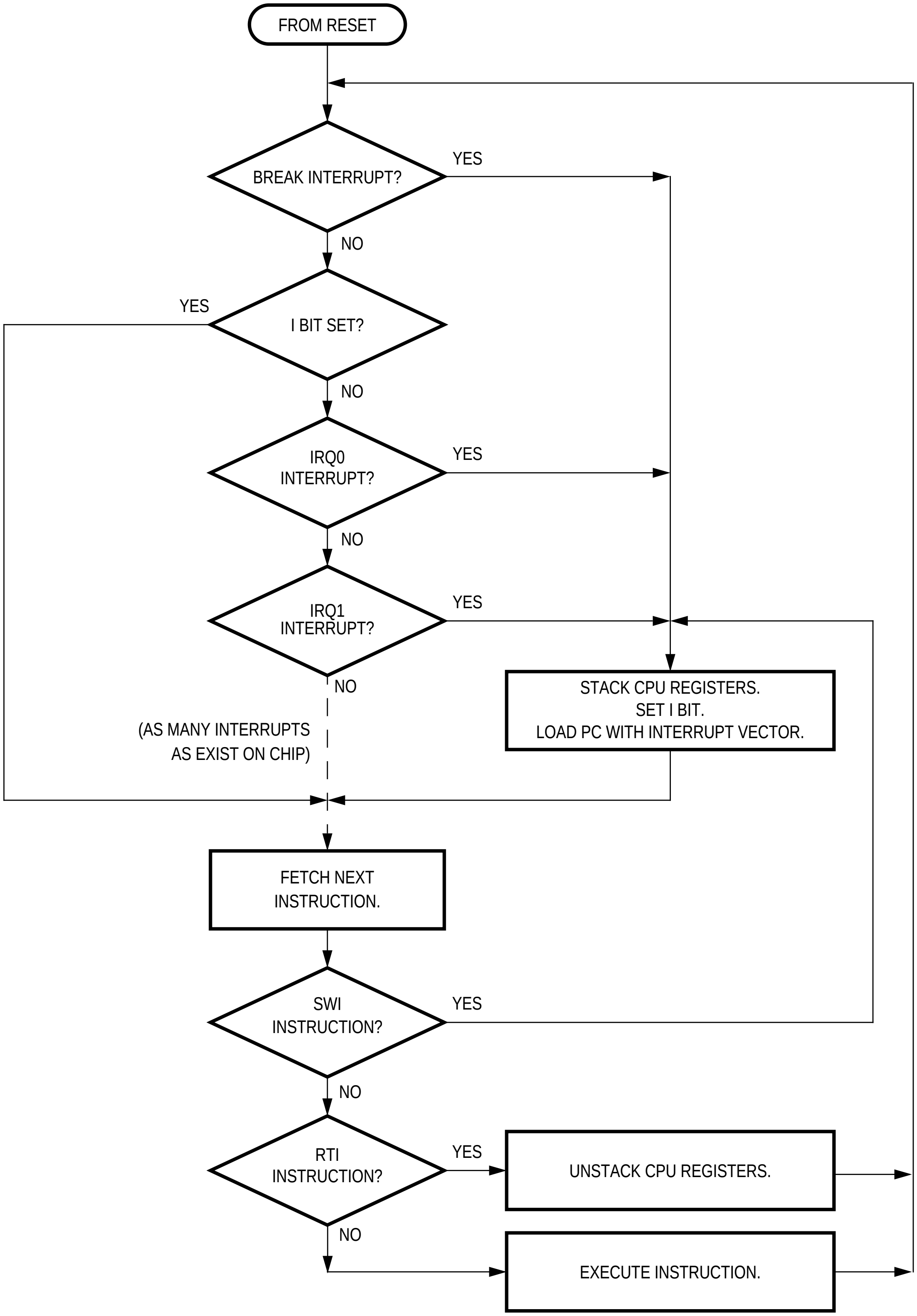


Figure 7-8. Interrupt Processing

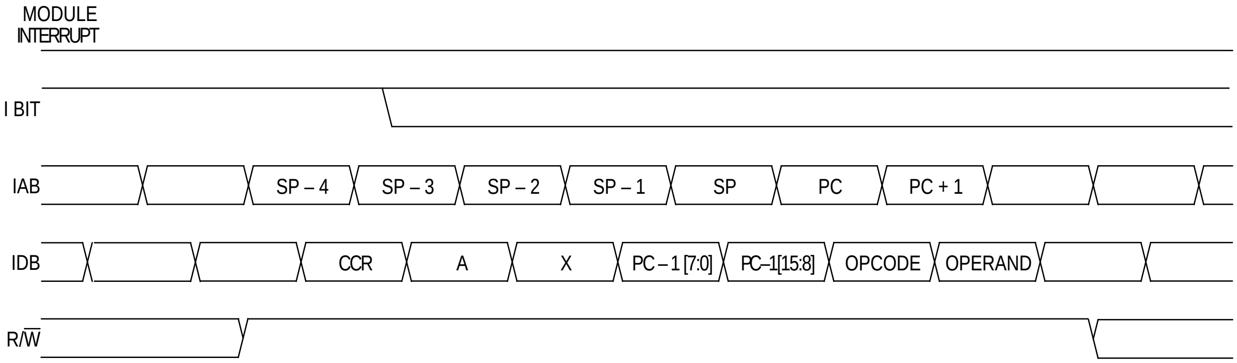


Figure 7-9. Interrupt Recovery

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 7-10** demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

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

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

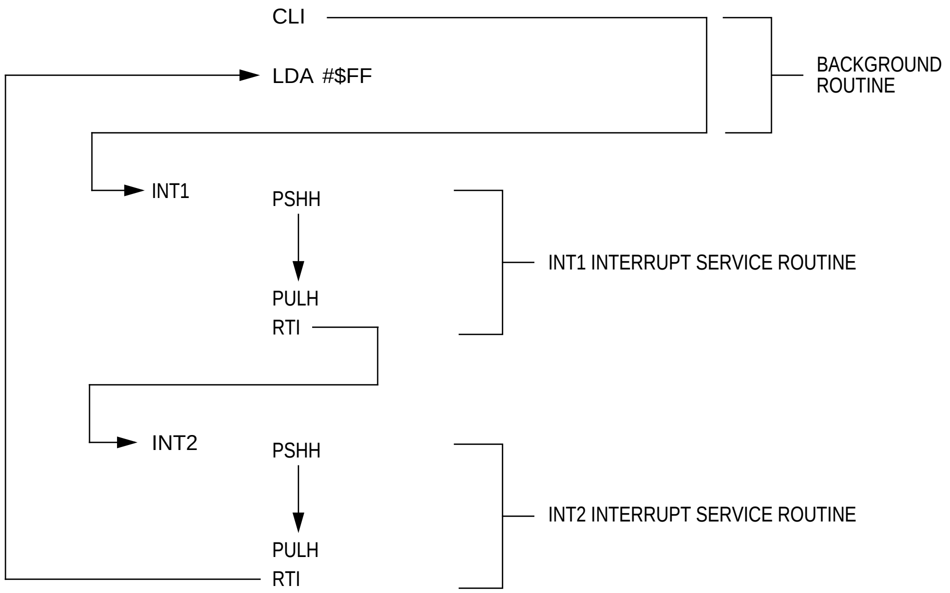

CLI
LDA #$FF
BACKGROUND ROUTINE
INT1
PSHH
PULH
RTI
INT1 INTERRUPT SERVICE ROUTINE
INT2
PSHH
PULH
RTI
INT2 INTERRUPT SERVICE ROUTINE

Figure 7-10. 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.*

7.6.2 Reset

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

7.6.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See **Section 10. Break Module (Break)**.) The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

7.6.4 Status Flag Protection in Break Mode

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

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

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

7.7 Low-Power Modes

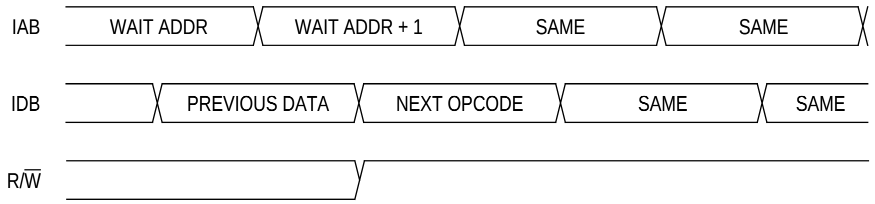
Executing the WAIT or STOP instruction puts the MCU in a low-power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts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

7.7.1 Wait Mode

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

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

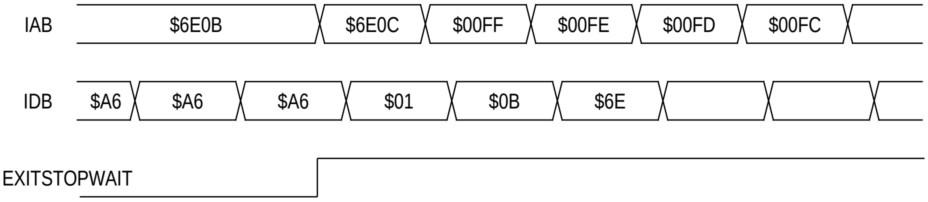
Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is logic zero, then the computer operating properly module (COP) is enabled and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



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

Figure 7-11. Wait Mode Entry Timing

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



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

Figure 7-12. Wait Recovery from Interrupt or Break

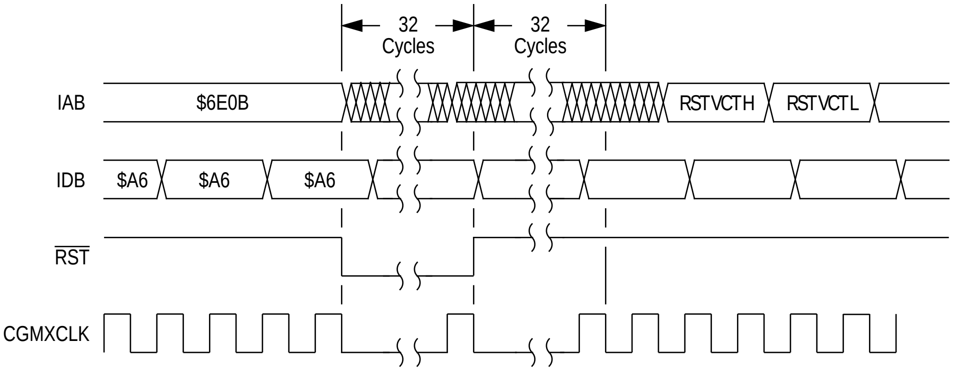


Figure 7-13. Wait Recovery from Internal Reset

7.7.2 Stop Mode

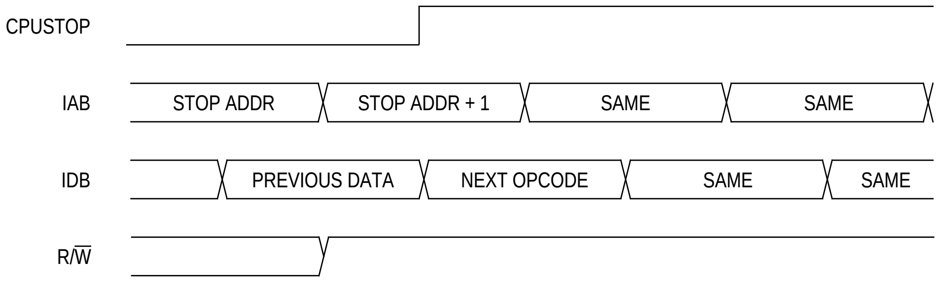
In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

The SIM disables the clock generator module outputs (CGMOUT and CGMXCLK) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the configuration register (CONFIG). 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.*

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

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



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

Figure 7-14. Stop Mode Entry Timing

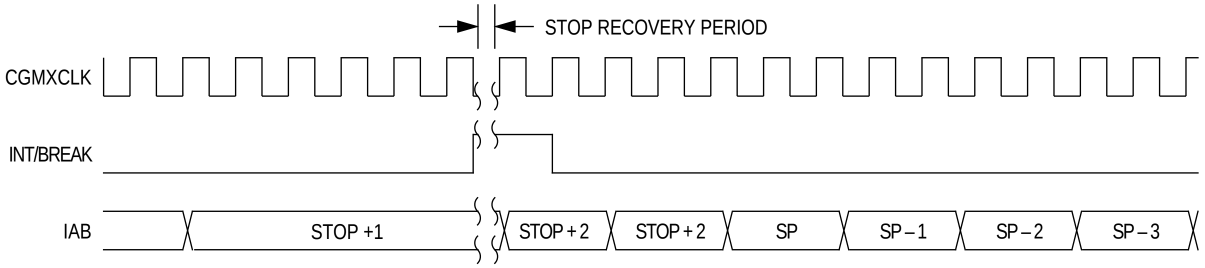


Figure 7-15. Stop Mode Recovery from Interrupt or Break

7.8 SIM Registers

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

Table 7-4. SIM Registers

Address	Register
$FE00	SBSR
$FE01	SRSR
$FE03	SBFCR

7.8.1 SIM Break Status Register (SBSR)

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

NOTE: *Writing a logic zero clears SBSW.*

		Bit 7	6	5	4	3	2	1	Bit 0
SBSR $FE00	Read:	R	R	R	R	R	R	SBSW	R
	Write:							See Note	
	Reset:							0	

R	= Reserved

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

SBSW — SIM Break Stop/Wait

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

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

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

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

  HIBYTE   EQU    5

  LOBYTE   EQU    6

;          If not SBSW, do RTI

           BRCLR  SBSW,SBSR, RETURN ; See if wait mode or stop mode was exited
                                    ; by break.

           TST    LOBYTE,SP         ; If RETURNLO is not zero,

           BNE    DOLO              ; then just decrement low byte.

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

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

  RETURN   PULH                     ; Restore H register.
           RTI
```

7.8.2 SIM Reset Status Register (SRSR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
SRSR $FE01	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	POR:	1	0	0	0	0	0	0	0

▨ = Unimplemented

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

POR — Power-On Reset Bit

1 = Last reset caused by POR circuit
0 = Read of SRSR

PIN — External Reset Bit

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

COP — Computer Operating Properly Reset Bit

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

ILOP — Illegal Opcode Reset Bit

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

ILAD — Illegal Address Reset Bit (opcode fetches only)

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

LVI — Low-Voltage Inhibit Reset Bit

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

7.8.3 SIM Break Flag Control Register (SBFCR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
SBFCR $FE03	Read: Write:	BCFE	R	R	R	R	R	R	R
	Reset:	0							

R = Reserved

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

BCFE — Break Clear Flag Enable Bit

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

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

Section 8. Clock Generator Module (CGM)

8.1 Contents

8.2 Introduction

This section describes the clock generator module (CGM). The CGM generates the crystal clock signal, CGMXCLK, which operates at the frequency of the crystal. The CGM also generates the base clock signal, CGMOUT, from which the system integration module (SIM) derives the system clocks. CGMOUT is based on either the crystal clock divided by two or the phase-locked loop (PLL) clock, CGMVCLK, divided by two. The PLL is a frequency generator designed for use with 1-MHz to 16-MHz crystals or ceramic resonators. The PLL can generate an 8-MHz bus frequency without using a 32-MHz crystal.

8.3 Features

Features of the CGM include the following:

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

8.4 Functional Description

The CGM consists of three major submodules:

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

Figure 8-1 shows the structure of the CGM.

8.4.1 Crystal Oscillator Circuit

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

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

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

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

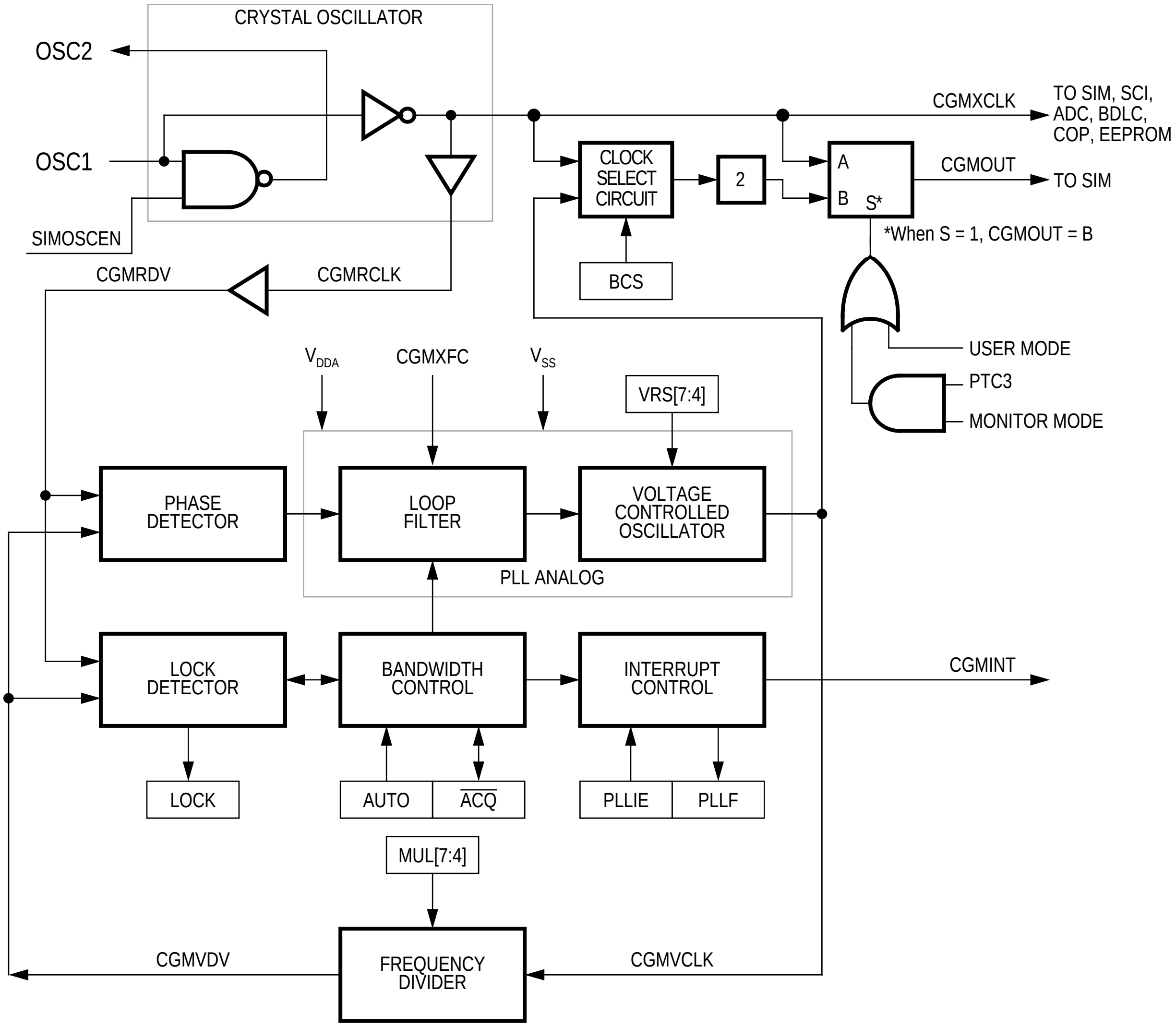


Figure 8-1. CGM Block Diagram

Table 8-1. CGM I/O Register Summary

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

▨ = Unimplemented

8.4.2 Phase-Locked Loop Circuit (PLL)

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

Circuits

The PLL consists of the following circuits:

- Voltage-controlled oscillator (VCO)
- Modulo VCO frequency divider
- Phase detector
- Loop filter
- Lock detector

The operating range of the VCO is programmable for a wide range of frequencies and for maximum immunity to external noise, including supply and CGMXFC noise. The VCO frequency is bound to a range from roughly one-half to twice the center-of-range frequency, f_{VRS}. Modulating the voltage on the CGMXFC pin changes the frequency within this range. By design, f_{VRS} is equal to the nominal center-of-range frequency, f_{NOM}, (4.9152 MHz) times a linear factor L, or $(L)f_{NOM}$.

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

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

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

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

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 $\overline{\text{ACQ}}$ bit is clear in the PLL bandwidth control register. (See **8.6.2 PLL Bandwidth Control Register (PBWC)**.)
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. (See **8.4.3 Base Clock Selector Circuit**.) The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{\text{ACQ}}$ bit is set.

Manual and Automatic PLL Bandwidth Modes

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

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. (See **8.6.2 PLL Bandwidth Control Register (PBWC)**.) If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (during PLL 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 **8.4.3 Base Clock Selector Circuit**.) If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. (See **8.7 Interrupts** for information and precautions on using interrupts.) The following conditions apply when the PLL is in automatic bandwidth control mode:

- The $\overline{\text{ACQ}}$ bit (see **8.6.2 PLL Bandwidth Control Register (PBWC)**) is a read-only indicator of the mode of the filter. (See Acquisition and Tracking Modes.)
- The $\overline{\text{ACQ}}$ bit is set when the VCO frequency is within a certain tolerance Δ_{TRK} and is cleared when the VCO frequency is out of a certain tolerance Δ_{UNT}. (See **8.10 Acquisition/Lock Time Specifications** for more information.)
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance Δ_{LOCK} and is cleared when the VCO frequency is out of a certain tolerance Δ_{UNL}. (See **8.10 Acquisition/Lock Time Specifications** for more information.)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See **8.6.1 PLL Control Register (PCTL)**.)

The PLL also may operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below f_{BUSMAX} and require fast 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 **8.10 Acquisition/Lock Time Specifications**), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time, t_{AL}, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.

Programming the PLL

The following procedure shows how to program the PLL.

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

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

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

3. Choose a practical PLL reference frequency, f_{RCLK}.

4. Select a VCO frequency multiplier, N.

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

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

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

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

6. Select a VCO linear range multiplier, L.

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

where f_{NOM} = 4.9152 MHz

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

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

8. Verify the choice of N and L by comparing f_{VCLK} to f_{VRS} and $f_{VCLKDES}$. For proper operation, f_{VCLK} must be within the application's tolerance of $f_{VCLKDES}$, and f_{VRS} must be as close as possible to f_{VCLK}.

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

9. Program the PLL registers accordingly:

   a. In the upper 4 bits of the PLL programming register (PPG), program the binary equivalent of N.

   b. In the lower 4 bits of the PLL programming register (PPG), program the binary equivalent of L.

Special Programming Exceptions

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

- A zero value for N is interpreted exactly the same as a value of one.
- A zero value for L disables the PLL and prevents its selection as the source for the base clock. (See **8.4.3 Base Clock Selector Circuit**.)

8.4.3 Base Clock Selector Circuit

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

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

8.4.4 CGM External Connections

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

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in **Figure 8-2**. **Figure 8-2** shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

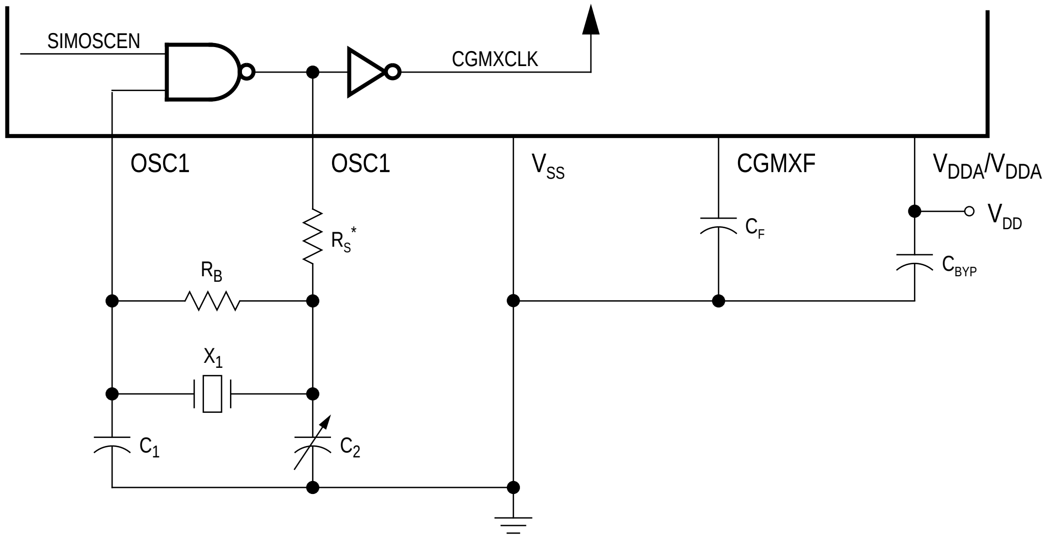
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S (optional)

The series resistor (R_S) is included in the diagram to follow strict Pierce oscillator guidelines and may not be required for all ranges of operation, especially with high frequency crystals. Refer to the crystal manufacturer's data for more information.

Figure 8-2 also shows the external components for the PLL:

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

Routing should be done with great care to minimize signal cross talk and noise. (See **8.10 Acquisition/Lock Time Specifications** for routing information and more information on the filter capacitor's value and its effects on PLL performance.)



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

Figure 8-2. CGM External Connections

8.5 I/O Signals

The following paragraphs describe the CGM I/O signals.

8.5.1 Crystal Amplifier Input Pin (OSC1)

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

8.5.2 Crystal Amplifier Output Pin (OSC2)

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

8.5.3 External Filter Capacitor Pin (CGMXFC)

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

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

8.5.4 Analog Power Pin (V_{DDA}/V_{DDAREF})

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

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

8.5.5 Oscillator Enable Signal (SIMOSCEN)

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

8.5.6 Crystal Output Frequency Signal (CGMXCLK)

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

8.5.7 CGM Base Clock Output (CGMOUT)

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

8.5.8 CGM CPU Interrupt (CGMINT)

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

8.6 CGM Registers

The following registers control and monitor operation of the CGM:

- PLL control register (PCTL) (See **8.6.1 PLL Control Register (PCTL)**.)
- PLL bandwidth control register (PBWC) (See **8.6.2 PLL Bandwidth Control Register (PBWC)**.)
- PLL programming register (PPG) (See **8.6.3 PLL Programming Register (PPG)**.)

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

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

▨ = Unimplemented

NOTES:
1. When AUTO = 0, PLLIE is forced to logic zero and is read-only.
2. When AUTO = 0, PLLF and LOCK read as logic zero.
3. When AUTO = 1, $\overline{\text{ACQ}}$ is read-only.
4. When PLLON = 0 or VRS[7:4] = $0, BCS is forced to logic zero and is read-only.
5. When PLLON = 1, the PLL programming register is read-only.
6. When BCS = 1, PLLON is forced set and is read-only.

Figure 8-3. CGM I/O Register Summary

8.6.1 PLL Control Register (PCTL)

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

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

▨ = Unimplemented

Figure 8-4. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

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

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL Interrupt Flag Bit

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

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

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

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). (See **8.4.3 Base Clock Selector Circuit**.) Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See **8.4.3 Base Clock Selector Circuit**.) Reset and the STOP instruction clear the BCS bit.

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

NOTE: *PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. (See* ***8.4.3 Base Clock Selector Circuit****.)*

PCTL[3:0] — Unimplemented bits

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

8.6.2 PLL Bandwidth Control Register (PBWC)

The PLL bandwidth control register does the following:

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

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

▨ = Unimplemented

Figure 8-6. PLL Bandwidth Control Register (PBWC)

AUTO — Automatic Bandwidth Control Bit

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

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock Indicator Bit

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

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

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

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

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

1 = Tracking mode
0 = Acquisition mode

XLD — Crystal Loss Detect Bit

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

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

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

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

PBWC[3:0] — Reserved for Test

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

8.6.3 PLL Programming Register (PPG)

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

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

Figure 8-7. PLL Programming Register (PPG)

MUL[7:4] — Multiplier Select Bits

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

Table 8-2. VCO Frequency Multiplier (N) Selection

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

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

VRS[7:4] — VCO Range Select Bits

These read/write bits control the hardware center-of-range linear multiplier L, which controls the hardware center-of-range frequency f_{VRS}. (See **Circuits**, **Programming the PLL**, and **8.6.1 PLL Control Register (PCTL)**.) VRS[7:4] cannot be written when the PLLON bit in the PLL control register (PCTL) is set. (See **Special Programming Exceptions**.) A value of $0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See **8.4.3 Base Clock Selector Circuit** and **Special Programming Exceptions** for more information.) Reset initializes the bits to $6 to give a default range multiply value of 6.

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

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

8.7 Interrupts

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

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

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

8.8 Special Modes

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

8.8.1 Wait Mode

The WAIT instruction does not affect the CGM. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL). Less power-sensitive applications can disengage the PLL without turning it off. Applications that require the PLL to wake the MCU from wait mode also can deselect the PLL output without turning off the PLL.

8.8.2 Stop Mode

When the STOP instruction executes, the SIM drives the SIMOSCEN signal low, disabling the CGM and holding low all CGM outputs (CGMXCLK, CGMOUT, and CGMINT).

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

8.9 CGM During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

8.10 Acquisition/Lock Time Specifications

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

8.10.1 Acquisition/Lock Time Definitions

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

is the time taken to return from 900 kHz to 1 MHz ±5 kHz. Five kHz = 5% of the 100 kHz step input.

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

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

- Acquisition time, t_{ACQ}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the tracking mode entry tolerance, Δ_{TRK}. Acquisition time is based on an initial frequency error, $(f_{DES} - f_{ORIG})/f_{DES}$, of not more than ±100%. In automatic bandwidth control mode (see Manual and Automatic PLL Bandwidth Modes), acquisition time expires when the $\overline{\text{ACQ}}$ 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 Manual and Automatic PLL Bandwidth Modes.)

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

8.10.2 Parametric Influences on Reaction Time

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

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

Another critical parameter is the external filter capacitor. The PLL modifies the voltage on the VCO by adding or subtracting charge from this capacitor. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. (See **8.10.3 Choosing a Filter Capacitor**.)

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

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

8.10.3 Choosing a Filter Capacitor

As described in **8.10.2 Parametric Influences on Reaction Time**, the external filter capacitor, C_F, is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to the following equation:

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

For acceptable values of C_{FACT}, see **8.10 Acquisition/Lock Time Specifications**. For the value of V_{DDA}, choose the voltage potential at which the MCU is operating. If the power supply is variable, choose a value near the middle of the range of possible supply values.

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

8.10.4 Reaction Time Calculation

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

- Correct selection of filter capacitor, C_F (See **8.10.3 Choosing a Filter Capacitor**.)
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

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

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

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

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

NOTE: *The inverse proportionality between the lock time and the reference frequency.*

In automatic bandwidth control mode the acquisition and lock times are quantized into units based on the reference frequency. (See Manual and Automatic PLL Bandwidth Modes.) A certain number of clock cycles, n_{ACQ}, is required to ascertain that the PLL is within the tracking mode entry tolerance, Δ_{TRK}, before exiting acquisition mode. A certain number of clock cycles, n_{TRK}, is required to ascertain that the PLL is within the lock mode entry tolerance, Δ_{LOCK}. Therefore, the acquisition time, t_{ACQ}, is an integer multiple of n_{ACQ}/f_{RDV}, and the acquisition to lock time, t_{AL}, is an integer multiple of n_{TRK}/f_{RDV}. Also, since the average frequency over the entire measurement period must be within the specified tolerance, the total time usually is longer than t_{LOCK} as calculated above.

In manual mode, it is usually necessary to wait considerably longer than t_{LOCK} before selecting the PLL clock (see **8.4.3 Base Clock Selector Circuit**), because the factors described in **8.10.2 Parametric Influences on Reaction Time** may slow the lock time considerably.

Section 9. Configuration Register (CONFIG)

9.1 Contents

9.2 Introduction

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

- Resets caused by the low-voltage inhibit module (LVI)
- Power to the LVI module
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- STOP instruction
- Computer operating properly module (COP) resets
- COP time-out period

9.3 Functional Description

The configuration register is used in the initialization of various options. The configuration register can only be written once after each reset. All of the configuration register bits are cleared with reset. Since the various options affect the operation of the MCU it is recommended that this register be written immediately after reset. The configuration register is located at $001F. For compatibility, a write to the ROM version at this location will have no effect. The configuration register may be read at any time.

NOTE: *Reset will clear the contents of the configuration register. The configuration register will allow only one write.*

		Bit 7	6	5	4	3	2	1	Bit 0
CONFIG $001F	Read: Write:			LVIRSTD	LVIPWRD	SSREC	COPL	STOP	COPD
	POR:	0	0	0	0	0	0	0	0

Figure 9-1. Configuration Register (CONFIG)

LVIPWRD — LVI Module Power Disable Bit

LVIPWRD disables the LVI Module. Reset resets LVIPWRD. (See **Section 13. Low-Voltage Inhibit (LVI)**.)

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

LVIRSTD — LVI Module Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module. Reset resets LVIRSTD. (See **Section 13. Low-Voltage Inhibit (LVI)**.)

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

SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 CGMXCLK cycles instead of a 4096-CGMXCLK cycle delay. Reset clears SSREC. (See **7.7.2 Stop Mode**.)
1 = Stop mode recovery after 32 CGMXCLK cycles
0 = Stop mode recovery after 4096 CGMXCLK cycles

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

COPL — COP Long Timeout Bit

COPL selects the long COP timeout period. Reset clears COPL.
1 = COP timeout period is 2^{18} -2^{4} CGMXCLK cycles
0 = COP timeout period is 2^{13} -2^{4} CGMXCLK cycles

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction. Reset clears STOP.
1 = STOP instruction enabled
0 = STOP instruction treated as illegal opcode

COPD — COP Disable Bit

COPD disables the COP module. Reset clears COPD.
(See **Section 12. Computer Operating Properly (COP)**.)
1 = COP module disabled
0 = COP module enabled

Section 10. Break Module (Break)

10.1 Contents

10.2 Introduction

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

10.3 Features

Features of the break module include the following:

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

10.4 Functional Description

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

The following events can cause a break interrupt to occur:

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

NOTE: *DMA section and associated functions are only valid if the MCU has a DMA module.*

When a CPU- or DMA-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return from interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. **Figure 10-1** shows the structure of the break module.

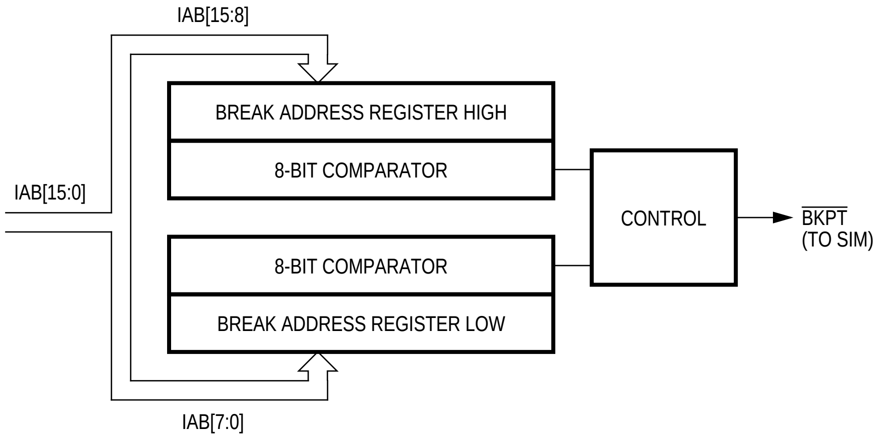


Figure 10-1. Break Module Block Diagram

Table 10-1. Break I/O Register Summary

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

▨ = Unimplemented

10.4.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)** and the **Break Interrupts** subsection for each module.)

10.4.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

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

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

10.4.3 DMA During Break Interrupts

During a break interrupt, the DMA is inactive.

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

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

10.4.4 TIM During Break Interrupts

A break interrupt stops the timer counter.

10.4.5 COP During Break Interrupts

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

10.5 Break Module Registers

Three registers control and monitor operation of the break module:

- Break status and control register (BRKSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)

10.5.1 Break Status and Control Register (BRKSCR)

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

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

▨ = Unimplemented

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

BRKE — Break Enable Bit

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a logic zero to bit 7. Reset clears the BRKE bit.
1 = Breaks enabled on 16-bit address match
0 = Breaks disabled on 16-bit address match

BRKA — Break Active Bit

This read/write status and control bit is set when a break address match occurs. Writing a logic one to BRKA generates a break interrupt. Clear BRKA by writing a logic zero to it before exiting the break routine. Reset clears the BRKA bit.
1 = Break address match
0 = No break address match

10.5.2 Break Address Registers (BRKH and BRKL)

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

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

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

10.6 Low-Power Modes

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

10.6.1 Wait Mode

If enabled, the break module and the DMA module are active in wait mode. The SIM break stop/wait bit (SBSW) in the SIM break status register (see **7.8 SIM Registers**) becomes set if a DMA-generated address matches the break address registers in wait mode. The DMA can also use the break status and control register as its destination address in order to write to the BRKA and BRKE bits during wait mode. The SBSW bit is set if the DMA writes to the break status and control register. SBSW is for applications that require a return to wait mode after exiting wait mode for a DMA-generated break interrupt. In the break routine, the user can subtract one from the return address on the stack if SBSW is set. Clear the SBSW bit by writing logic zero to it.

10.6.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the SIM break status register. (See **7.8 SIM Registers**.)

Section 11. Monitor ROM (MON08)

11.1 Contents

11.2 Introduction

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

11.3 Features

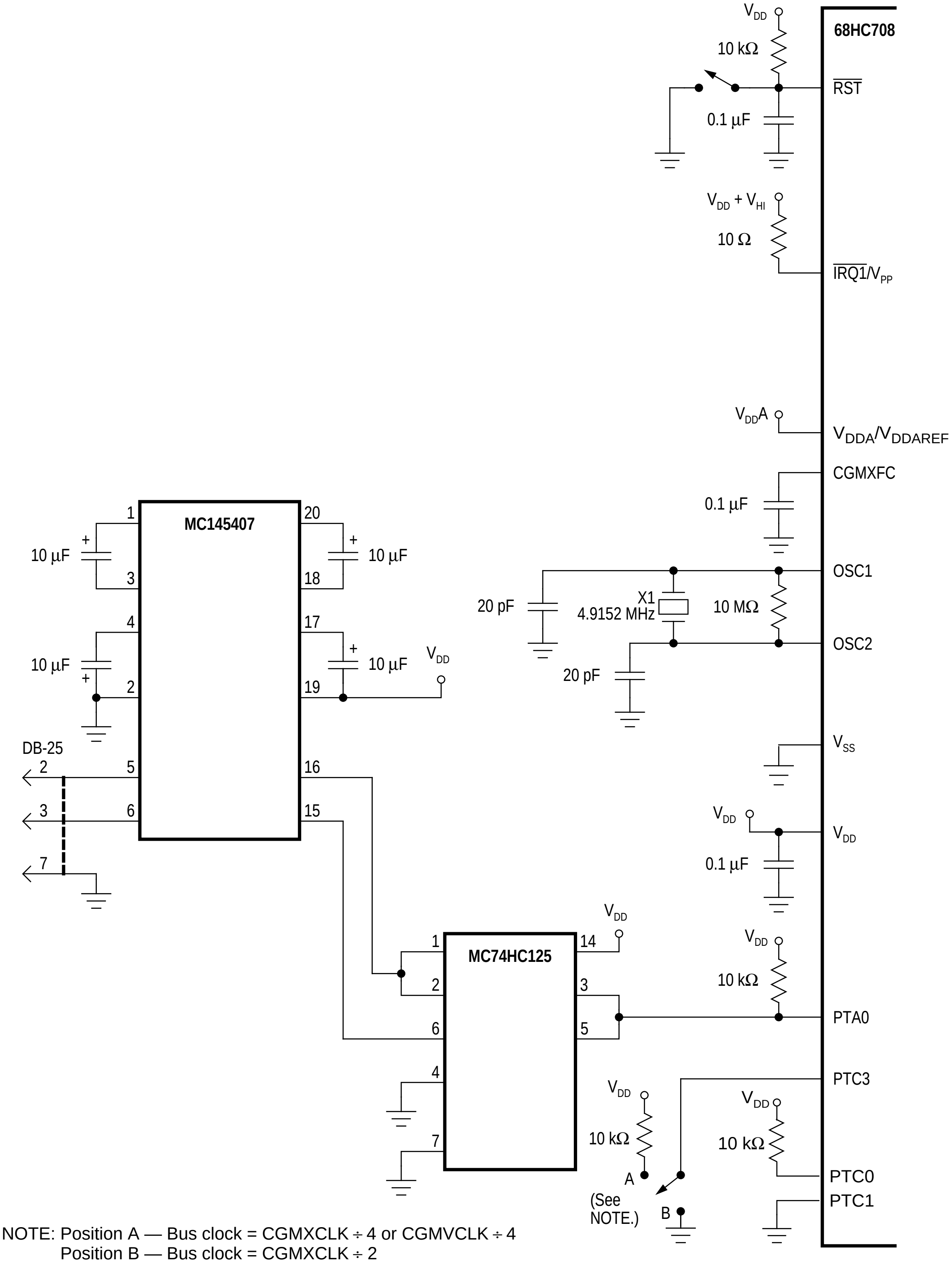
Features of the monitor ROM include the following:

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

11.4 Functional Description

The monitor ROM receives and executes commands from a host computer. **Figure 11-1** shows a sample circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

While simple monitor commands can access any memory address, the MC68HC708AS48 has an EPROM security feature that requires proper procedures to be followed before the EPROM can be accessed. In monitor mode, the MCU can execute host-computer code in RAM while all MCU pins except PTA0 retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.



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

Figure 11-1. Monitor Mode Circuit

11.4.1 Entering Monitor Mode

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

Table 11-1. Mode Selection

$\overline{IRQ1}/V_{PP}$ Pin	PTC0 Pin	PTC1 Pin	PTA0 Pin	PTC3 Pin	Mode	CGMOUT	Bus Frequency
$V_{DD} + V_{HI}$	1	0	1	1	Monitor	$\frac{CGMXCLK}{2}$ or $\frac{CGMVCLK}{2}$	$\frac{CGMOUT}{2}$
$V_{DD} + V_{HI}$	1	0	1	0	Monitor	CGMXCLK	$\frac{CGMOUT}{2}$

Enter monitor mode by either

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

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

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

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

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

Table 11-2. Mode Differences

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

[1] If the high voltage (V_{DD} + V_{HI}) is removed from the $\overline{IRQ1}/V_{PP}$ pin or the $\overline{RST}$ pin, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the configuration register.

11.4.2 Data Format

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

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

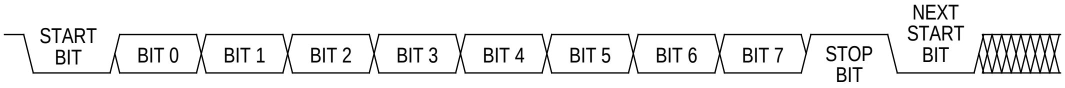


Figure 11-2. Monitor Data Format

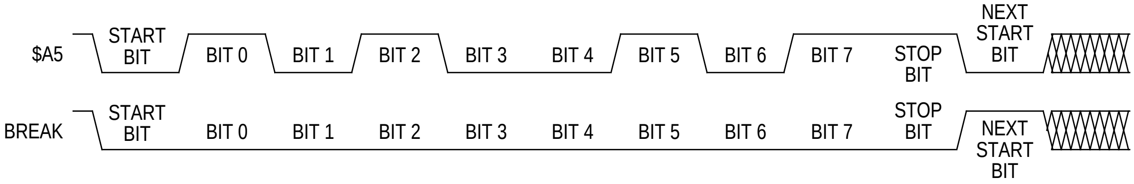


Figure 11-3. Sample Monitor Waveforms

11.4.3 Echoing

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

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

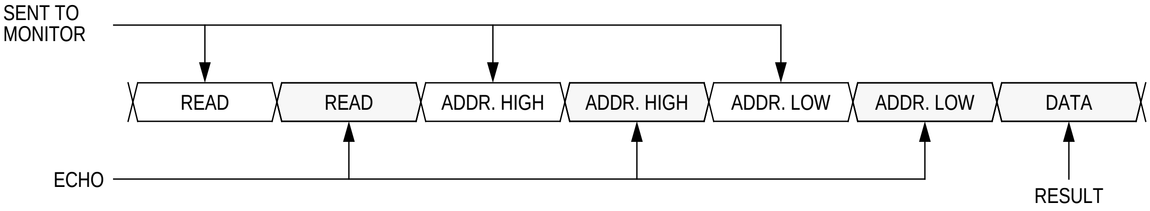


Figure 11-4. Read Transaction

11.4.4 Break Signal

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

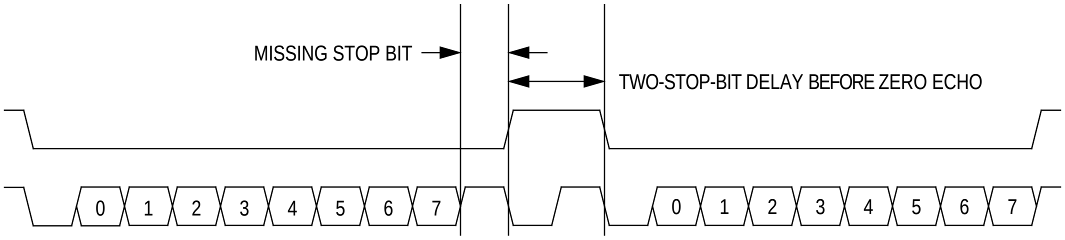


Figure 11-5. Break Transaction

11.4.5 Commands

The monitor ROM uses the following commands:

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

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

Table 11-3. READ (Read Memory) Command

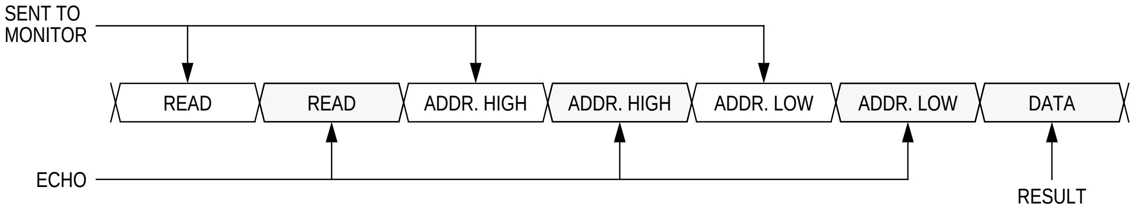
Description	Read byte from memory
Operand	Specifies 2-byte address in high byte:low byte order
Data Returned	Returns contents of specified address
Opcode	$4A
Command Sequence  SENT TO MONITOR READ READ ADDR. HIGH ADDR. HIGH ADDR. LOW ADDR. LOW DATA ECHO RESULT	

Table 11-4. WRITE (Write Memory) Command

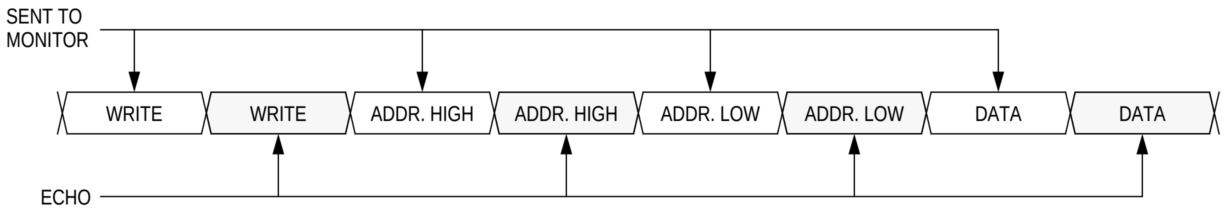
Description	Write byte to memory
Operand	Specifies 2-byte address in high byte:low byte order; low byte followed by data byte
Data Returned	None
Opcode	$49
Command Sequence	 SENT TO MONITOR WRITE WRITE ADDR. HIGH ADDR. HIGH ADDR. LOW ADDR. LOW DATA DATA ECHO

Table 11-5. IREAD (Indexed Read) Command

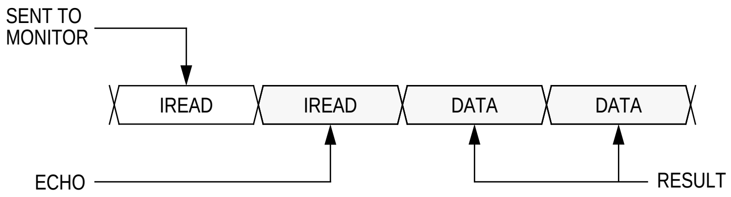
Description	Read next 2 bytes in memory from last address accessed
Operand	Specifies 2-byte address in high byte:low byte order
Data Returned	Returns contents of next two addresses
Opcode	$1A
Command Sequence	 SENT TO MONITOR IREAD IREAD DATA DATA ECHO RESULT

Table 11-6. IWRITE (Indexed Write) Command

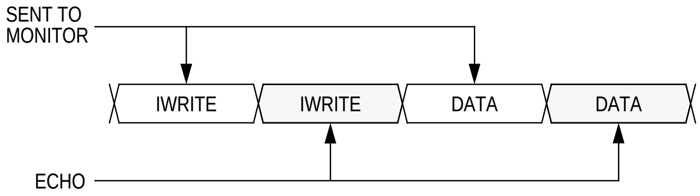
Description	Write to last address accessed + 1
Operand	Specifies single data byte
Data Returned	None
Opcode	$19
Command Sequence  	

Table 11-7. READSP (Read Stack Pointer) Command

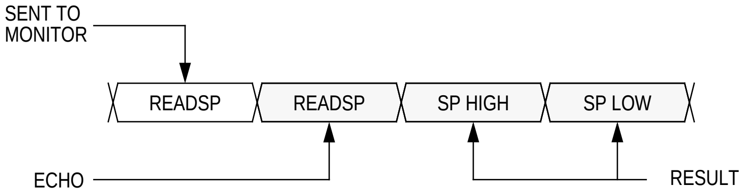
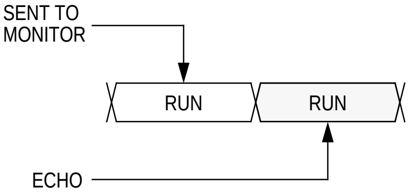
Description	Reads stack pointer
Operand	None
Data Returned	Returns stack pointer in high byte:low byte order
Opcode	$0C
Command Sequence  SENT TO MONITOR READSP READSP SP HIGH SP LOW ECHO RESULT	

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

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

11.4.6 Baud Rate

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

Table 11-9. Monitor Baud Rate Selection

	VCO Frequency Multiplier (N)					
	1	**2**	**3**	**4**	**5**	**6**
Monitor Baud Rate	4800	9600	14,400	19,200	24,000	28,800

Section 12. Computer Operating Properly (COP)

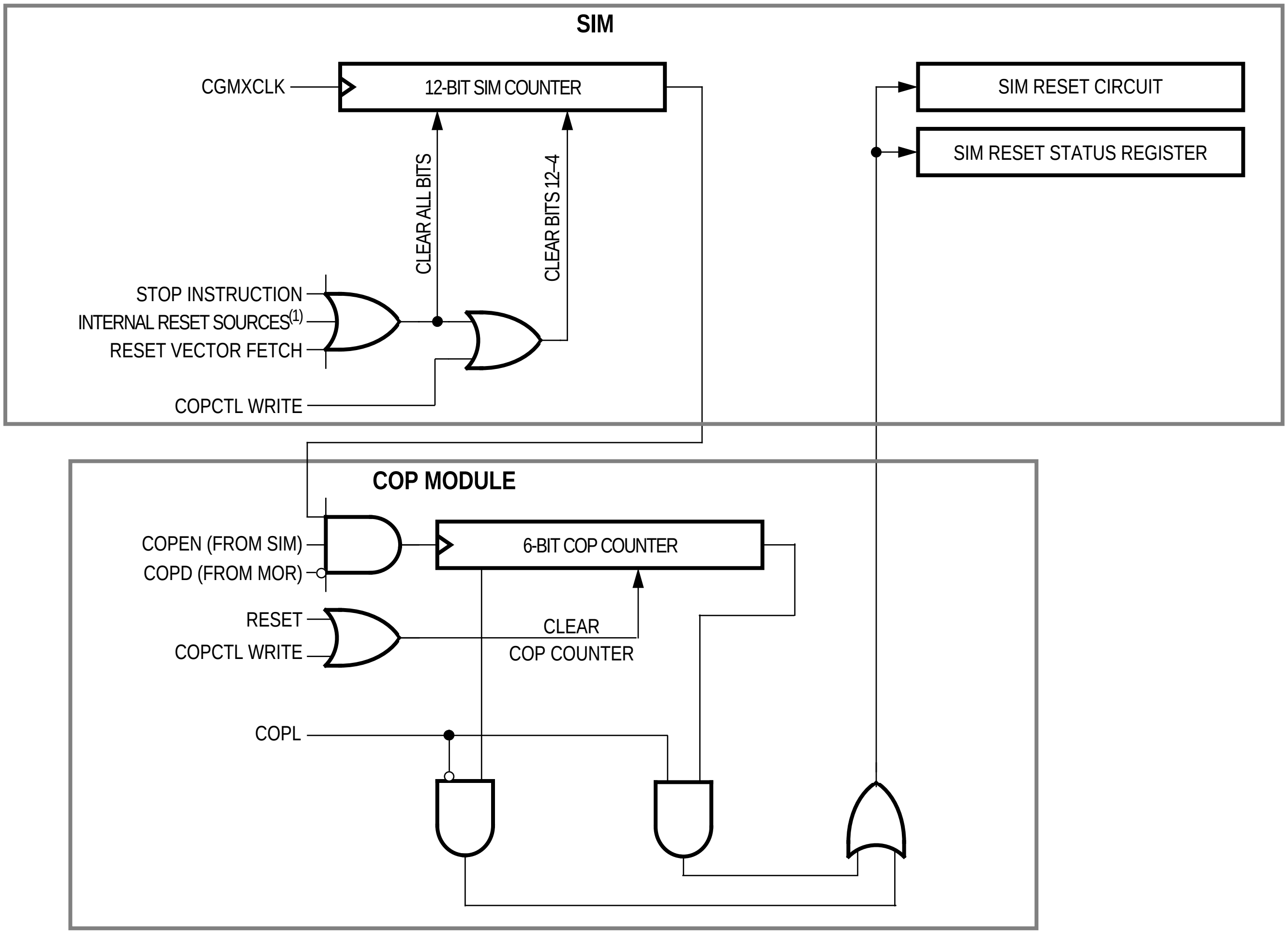
12.1 Contents

12.2 Introduction

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

12.3 Functional Description

Figure 12-1 shows the structure of the COP module.



NOTE:

1. See **7.4.2 Active Resets from Internal Sources**.

Figure 12-1. COP Block Diagram

Table 12-1. COP I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
COP Control Register (COPCTL)									$FFFF

The COP counter is a free-running 6-bit counter preceded by the 12-bit system integration module (SIM) counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after $(2^{13} – 2^4)$ or $(2^{18} – 2^4)$ CGMXCLK cycles depending upon COPL bit in the configuration register. With a 4.9152-MHz crystal and the COPL bit in the CONFIG register set to a logic one, the COP timeout period is approximately 53.3 ms. Writing any value to location $FFFF before overflow occurs clears the COP counter, clears bits 12 through 4 of the SIM counter, and prevents reset. A CPU interrupt routine or a DMA service routine can be used to clear the COP. The COP should be serviced as soon as possible out of reset and before entering or after exiting stop mode to guarantee the maximum selected amount of time before the first time out.

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles and sets the COP bit in the SIM reset status register (SRSR) (see **7.8.2 SIM Reset Status Register (SRSR))**.

The COP module is disabled if the $\overline{\text{RST}}$ pin or the $\overline{\text{IRQ1}}/V_{PP}$ pin is held at $V_{DD} + V_{HI}$ while the MCU is in monitor mode. The COP module can be disabled only through combinational logic conditioned with the high voltage signal on the $\overline{\text{RST}}$ or the $\overline{\text{IRQ1}}/V_{PP}$ pin. This prevents the COP from becoming disabled as a result of external noise. During a break state, $V_{DD} + V_{HI}$ on the $\overline{\text{RST}}$ pin disables the COP module.

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

12.4 I/O Signals

The following paragraphs describe the signals shown in **Figure 12-1**.

12.4.1 CGMXCLK

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

12.4.2 STOP Instruction

The STOP instruction clears the SIM counter.

12.4.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see **12.5 COP Control Register (COPCTL))** clears the COP counter and clears bits 12 through 4 of the SIM counter. Reading the COP control register returns the reset vector.

12.4.4 Internal Reset Resources

An internal reset clears the SIM counter and the COP counter. (See **7.4.2 Active Resets from Internal Sources**.)

12.4.5 Reset Vector Fetch

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

12.4.6 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register (CONFIG). (See **Section 9. Configuration Register (CONFIG)**.)

12.4.7 COPL (COP Long Timeout)

The COPL signal selects the state of the COP long timeout bit (COPL) in the configuration register (CONFIG). (See **Section 9. Configuration Register (CONFIG)**.)

12.5 COP Control Register (COPCTL)

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

		Bit 7	6	5	4	3	2	1	Bit 0
COPCTL $FFFF	Read:	Low byte of reset vector							
	Write:	Clear COP counter							
	Reset:	Unaffected by reset							

Figure 12-2. COP Control Register (COPCTL)

12.6 Interrupts

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

12.7 Monitor Mode

The COP is disabled in monitor mode when V_{DD} + V_{HI} is present on the $\overline{\text{IRQ1}}$/V_{PP} pin or on the $\overline{\text{RST}}$ pin.

12.8 Low-Power Modes

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

12.8.1 Wait Mode

The COP continues to operate during wait mode. To prevent a COP reset during wait mode, periodically clear the COP counter in a CPU interrupt routine or a DMA service routine.

12.8.2 Stop Mode

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

The STOP bit in the configuration register (CONFIG) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by programming the STOP bit to logic zero.

12.9 COP Module During Break Interrupts

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

Section 13. Low-Voltage Inhibit (LVI)

13.1 Contents

13.2 Introduction

This section describes the low-voltage inhibit module (LVI), 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.

13.3 Features

Features of the LVI module include the following:

- Programmable LVI Reset
- Programmable Power Consumption

13.4 Functional Description

Figure 13-1 shows the structure of the LVI module. The LVI module contains a bandgap reference circuit and comparator. The LVI power disable bit, LVIPWRD, disables the LVI from monitoring V_{DD} voltage. The LVI reset disable bit, LVIRSTD, disables the LVI module from generating a reset when V_{DD} falls below a voltage, V_{LVII}. LVIPWRD and LVIRSTD are in the configuration register (CONFIG). (See **Section 9. Configuration Register (CONFIG)**.) Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, V_{LVIR}. The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

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

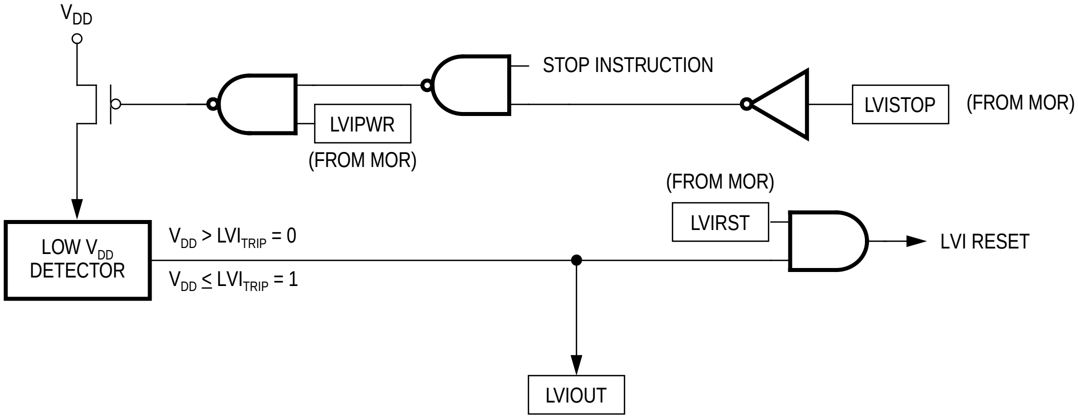


Figure 13-1. LVI Module Block Diagram

Table 13-1. LVI I/O Register Summary

	Bit 7	**6**	**5**	**4**	**3**	**2**	**1**	**Bit 0**	**Addr.**
LVI Status Register (LVISR)	LVIOUT	▨	▨	▨	LVISTOP	LVILCK	▨	▨	$FE0F

▨ = Unimplemented

13.4.1 Polled LVI Operation

In applications that can operate at V_{DD} levels below the V_{LVII} level, software can monitor V_{DD} by polling the LVIOUT bit. In the configuration register, the LVIPWRD bit must be at logic zero to enable the LVI module, and the LVIRSTD bit must be at logic one to disable LVI resets.

13.4.2 Forced Reset Operation

In applications that require V_{DD} to remain above the V_{LVII} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls to the V_{LVII} level. In the configuration register, the LVIPWRD and LVIRSTD bits must be at logic zero to enable the LVI module and to enable LVI resets.

13.5 LVI Status Register (LVISR)

The LVI status register flags V_{DD} voltages below the V_{LVII} level.

		Bit 7	6	5	4	3	2	1	Bit 0
LVISR $FE0F	Read:	LVIOUT	0	0	0	LVISTOP	LVILCK	0	0
	Write:	▨	▨	▨	▨			▨	▨
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 13-2. LVI Status Register (LVISR)

LVILCK — LVI lock

This read/write bit inhibits writing to the LVI status and control register. When LVILCK is set, writing to the LVI status and control register has no effect. The LVILCK bit can be cleared only by reset.

1 = LVISCR write-protected
0 = LVISCR not write-protected

LVISTOP — LVI Disable in Stop Mode Bit

This read/write bit turns off the low-voltage inhibit module (LVI) in stop mode when the LVISTOP bit is set.

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

NOTE: *To meet the stop mode I_{DD} specification, LVISTOP must be at logic one.*

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the V_{LVII} voltage. (See **Table 13-2**.) Reset clears the LVIOUT bit.

Table 13-2. LVIOUT Bit Indication

V_{DD}	LVIOUT
$V_{DD} > V_{LVIR}$	0
$V_{DD} < V_{LVII}$	1
$V_{LVII} < V_{DD} < V_{LVIR}$	Previous Value

13.6 LVI Interrupts

The LVI module does not generate interrupt requests.

13.7 Low-Power Modes

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

13.7.1 Wait Mode

With the LVIPWRD bit in the configuration register programmed to logic zero, the LVI module is active after a WAIT instruction.

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

13.7.2 Stop Mode

When the LVIPWRD bit in the configuration register is programmed to logic zero and the LVISTOP bit in the LVISR register is at logic zero, the LVI module remains active after a STOP instruction.

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

Section 14. External Interrupt (IRQ)

14.1 Contents

14.2 Introduction

This section describes the nonmaskable external interrupt (IRQ) input.

14.3 Features

Features include:

- Dedicated External Interrupt Pin ($\overline{\text{IRQ1}}$/V_{PP})
- Hysteresis Buffer
- Programmable Edge-only or Edge and Level Interrupt Sensitivity
- Automatic Interrupt Acknowledge

14.4 Functional Description

A logic zero applied to the external interrupt pin can latch a CPU interrupt request. **Figure 14-1** shows the structure of the IRQ module.

Interrupt signals on the $\overline{\text{IRQ1}}$/V_{PP} pin are latched into the IRQ1 latch. An interrupt latch remains set until one of the following actions occurs:

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the latch that caused the vector fetch.
- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (ISCR). Writing a logic one to the ACK1 bit clears the IRQ1 latch.
- Reset — A reset automatically clears both interrupt latches.

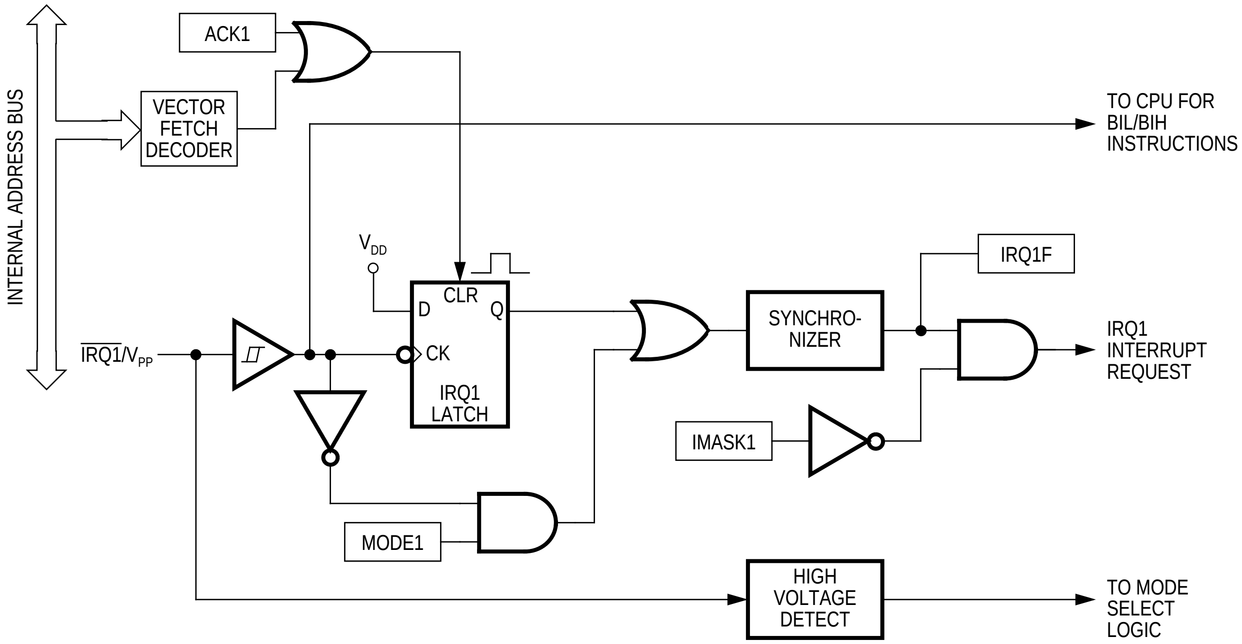


Figure 14-1. IRQ Block Diagram

Table 14-1. IRQ I/O Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
IRQ Status/Control Register (ISCR)	0	0	0	0	IRQF1	0	IMASK1	MODE1	$001A
	▨	▨	▨	▨	▨	ACK1			

▨ = Unimplemented

The external interrupt pin is falling-edge triggered and is software-configurable to be both falling-edge and low-level triggered. The MODE1 bit in the ISCR controls the triggering sensitivity of the $\overline{\text{IRQ1}}/V_{PP}$ 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 logic one

The vector fetch or software clear may occur before or after the interrupt pin returns to logic one. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

When set, the IMASK1 bit in the ISCR masks all external interrupt requests. A latched interrupt request is not presented to the interrupt priority logic unless the corresponding IMASK bit is clear.

NOTE: *The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See **Figure 14-2**.)*

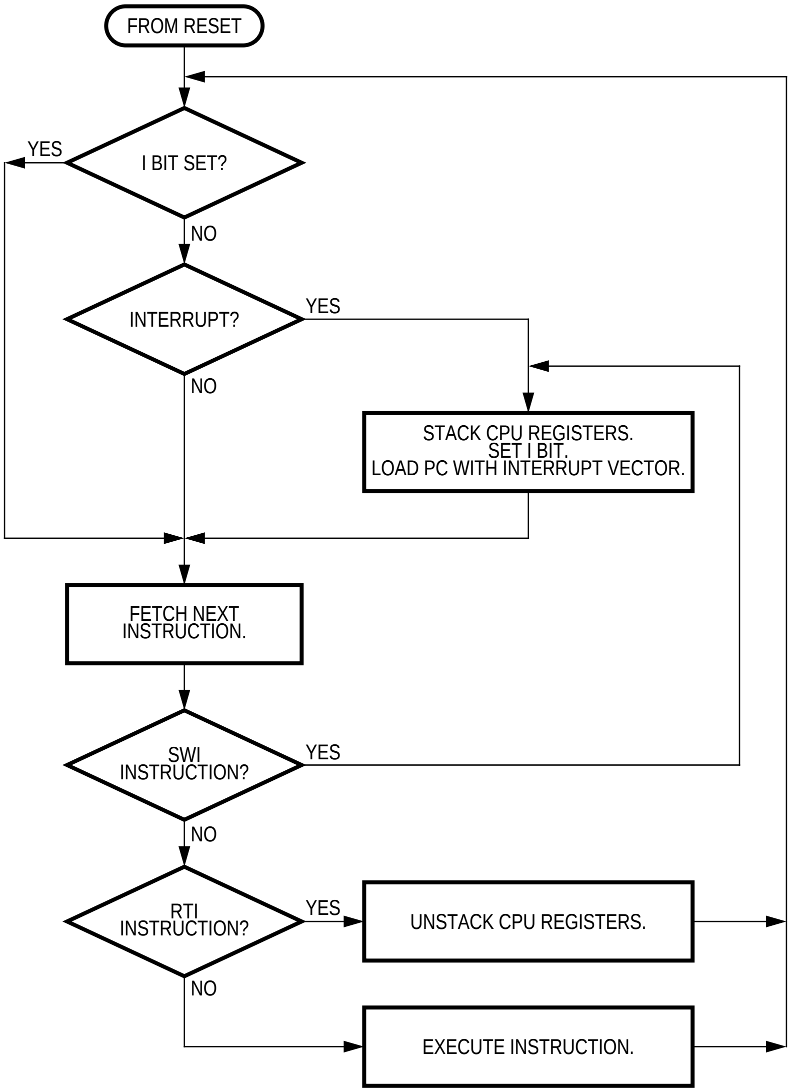


Figure 14-2. IRQ Interrupt Flowchart

14.5 $\overline{\text{IRQ1}}/V_{PP}$ Pin

A logic zero on the $\overline{\text{IRQ1}}/V_{PP}$ pin can latch an interrupt request into the IRQ1 latch. A vector fetch, software clear, or reset clears the IRQ1 latch.

If the MODE1 bit is set, the $\overline{\text{IRQ1}}/V_{PP}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE1 set, both of the following actions must occur to clear the IRQ1 latch:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic one to the ACK1 bit in the interrupt status and control register (ISCR). The ACK1 bit is useful in applications that poll the $\overline{\text{IRQ1}}/V_{PP}$ pin and require software to clear the IRQ1 latch. Writing to the ACK1 bit can also prevent spurious interrupts due to noise. Setting ACK1 does not affect subsequent transitions on the $\overline{\text{IRQ1}}/V_{PP}$ pin. A falling edge on $\overline{\text{IRQ1}}/V_{PP}$ that occurs after writing to the ACK1 bit latches another interrupt request. If the IRQ1 mask bit, IMASK1, is clear, the CPU loads the program counter with the vector address at locations \$FFFA and \$FFFB.
- Return of the $\overline{\text{IRQ1}}/V_{PP}$ pin to logic one — As long as the $\overline{\text{IRQ1}}/V_{PP}$ pin is at logic zero, the IRQ1 latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ1}}/V_{PP}$ pin to logic one may occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ1}}/V_{PP}$ pin is at logic zero. A reset will clear the latch and the MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

If the MODE1 bit is clear, the $\overline{\text{IRQ1}}/V_{PP}$ pin is falling-edge-sensitive only. With MODE1 clear, a vector fetch or software clear immediately clears the IRQ1 latch.

The IRQF1 bit in the ISCR register can be used to check for pending interrupts. The IRQF1 bit is not affected by the IMASK1 bit, which makes it useful in applications where polling is preferred.

Use the BIH or BIL instruction to read the logic level on the $\overline{\text{IRQ1}}$/V_{PP} pin.

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

14.6 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ1 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 **7.8.3 SIM Break Flag Control Register (SBFCR)**.)

To allow software to clear the IRQ1 latch during a break interrupt, write a logic one to the BCFE bit. If a latch is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect the latch during the break state, write a logic zero to the BCFE bit. With BCFE at logic zero (its default state), writing to the ACK1 bit in the IRQ status and control register during the break state has no effect on the IRQ latch.

14.7 IRQ Status and Control Register (ISCR)

The IRQ Status and Control Register (ISCR) controls and monitors operation of the IRQ module. The ISCR has the following functions:

- Shows the state of the IRQ1 interrupt flag
- Clears the IRQ1interrupt latch
- Masks IRQ1 interrupt request
- Controls triggering sensitivity of the $\overline{\text{IRQ1}}$/V_{PP} interrupt pin

		Bit 7	6	5	4	3	2	1	Bit 0
ISCR $001A	Read:	▨	▨	▨	▨	IRQF1	0	IMASK1	MODE1
	Write:	▨	▨	▨	▨	▨	ACK1		
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-3. IRQ Status and Control Register (ISCR)

IRQ1F — IRQ1 Flag

This read-only status bit is high when the IRQ1 interrupt is pending.
1 = $\overline{\text{IRQ1}}$ interrupt pending
0 = $\overline{\text{IRQ1}}$ interrupt not pending

ACK1 — IRQ1 Interrupt Request Acknowledge Bit

Writing a logic one to this write-only bit clears the IRQ1 latch. ACK1 always reads as logic zero. Reset clears ACK1.

IMASK1 — IRQ1 Interrupt Mask Bit

Writing a logic one to this read/write bit disables IRQ1 interrupt requests. Reset clears IMASK1.
1 = IRQ1 interrupt requests disabled
0 = IRQ1 interrupt requests enabled

MODE1 — IRQ1 Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ1}}$/V_{PP} pin. Reset clears MODE1.
1 = $\overline{\text{IRQ1}}$/V_{PP} interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ1}}$/V_{PP} interrupt requests on falling edges only

Section 15. Serial Communications Interface (SCI)

15.1 Contents

15.2 Introduction

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

15.3 Features

Features of the SCI module include the following:

- Full Duplex Operation
- Standard Mark/Space Non-Return-to-Zero (NRZ) Format
- 32 Programmable Baud Rates
- Programmable 8-Bit or 9-Bit Character Length
- Separately Enabled Transmitter and Receiver
- Separate Receiver and Transmitter CPU Interrupt Requests
- Separate Receiver and Transmitter DMA Service Requests
- Programmable Transmitter Output Polarity
- Two Receiver Wake-Up Methods:
  - Idle Line Wake-Up
  - Address Mark Wake-Up
- Interrupt-Driven Operation with Eight Interrupt Flags:
  - Transmitter Empty
  - Transmission Complete
  - Receiver Full
  - Idle Receiver Input
  - Receiver Overrun
  - Noise Error
  - Framing Error
  - Parity Error
- Receiver Framing Error Detection
- Hardware Parity Checking
- 1/16 Bit-Time Noise Detection

15.4 Functional Description

Figure 15-1 shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data. During DMA transfers, the DMA fetches data from memory for the SCI to transmit and/or the DMA stores received data in memory.

NOTE: *DMA section and associated functions are only valid if the MCU has a DMA module.*

Table 15-1. SCI I/O Register Summary

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

▨ = Unimplemented

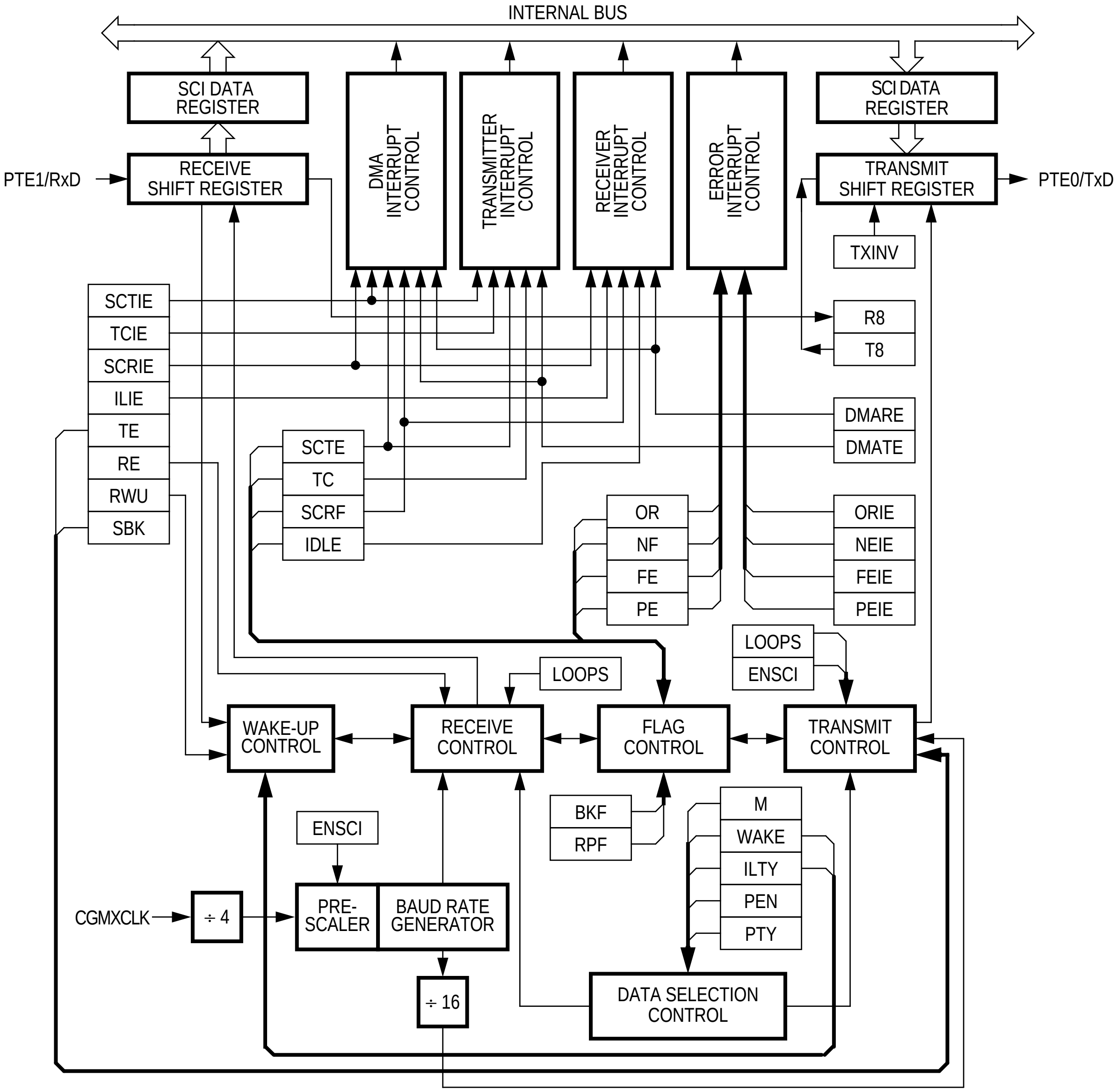


Figure 15-1. SCI Module Block Diagram

15.4.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in **Figure 15-2**.

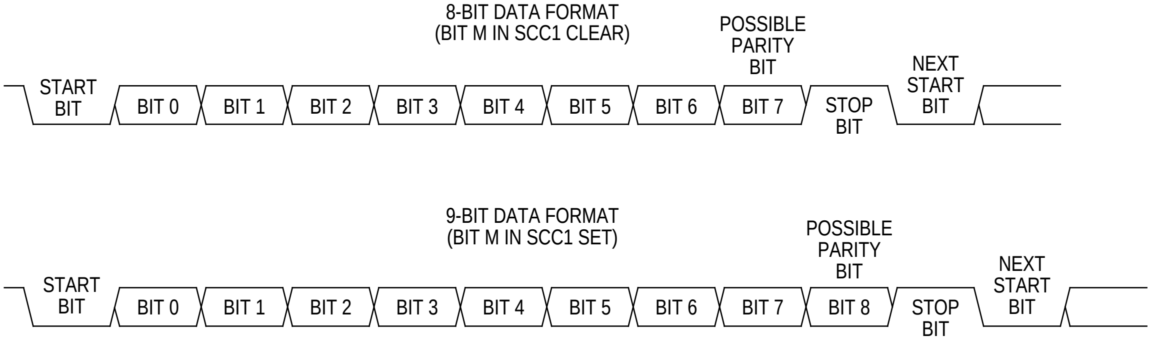


Figure 15-2. SCI Data Formats

15.4.2 Transmitter

Figure 15-3 shows the structure of the SCI transmitter.

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).

Character Transmission

During an SCI transmission, the transmit shift register shifts a character out to the PTE0/TxD pin. The SCI data register (SCDR) is the write-only buffer between the internal data bus and the transmit shift register. To initiate an SCI transmission:

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

3. Clear the SCI transmitter empty bit by first reading SCI status register 1 (SCS1) and then writing to the SCDR. In a DMA transfer, the DMA automatically clears the SCTE bit by writing to the SCDR.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of logic ones. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic zero start bit automatically goes into the least significant bit position of the transmit shift register. A logic one stop bit goes into the most significant bit position.

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

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

When the transmit shift register is not transmitting a character, the PTE0/TxD pin goes to the idle condition, logic one. If at any time software clears the ENSCI bit in SCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port E pins.

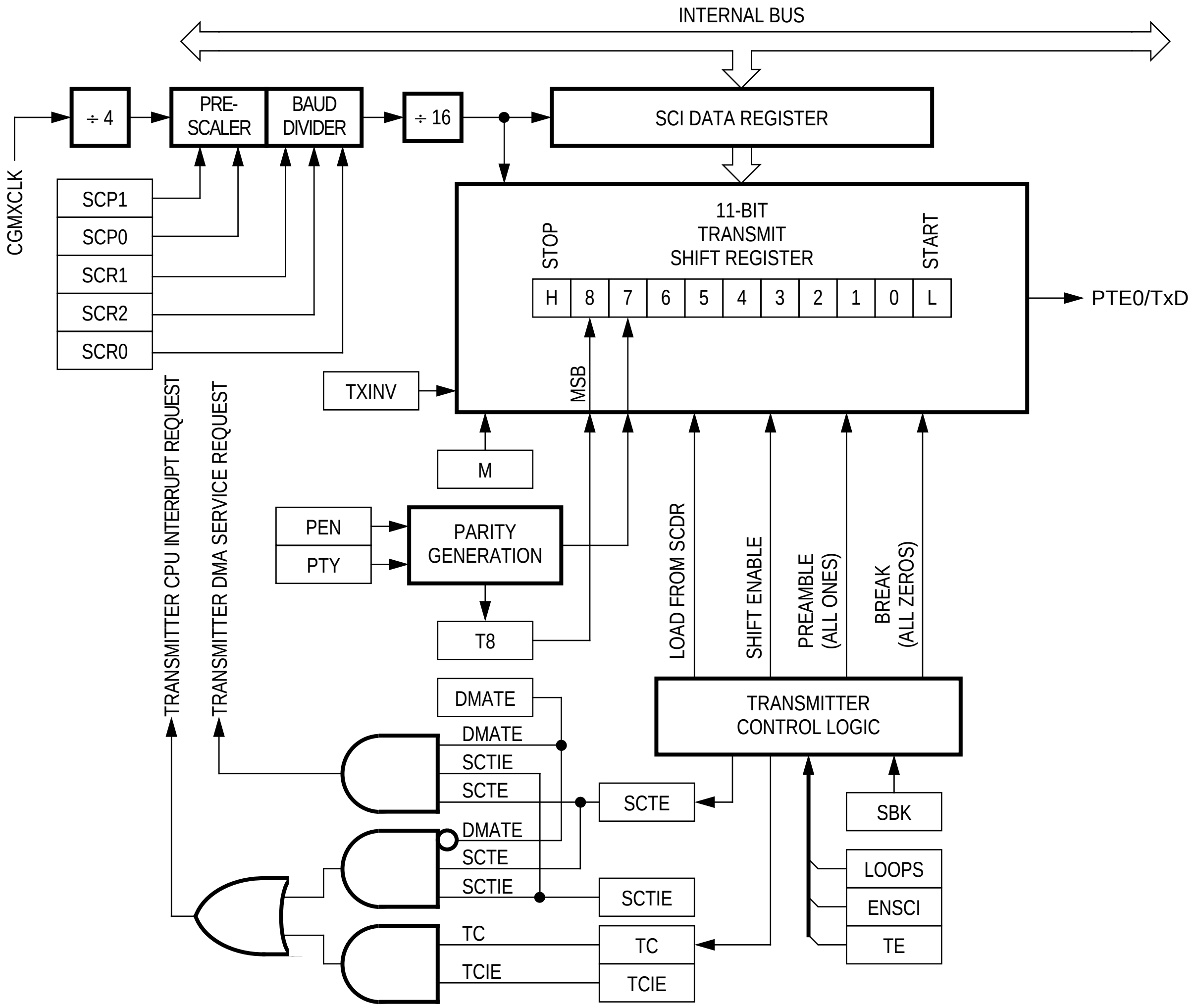


Figure 15-3. SCI Transmitter

Table 15-2. SCI Transmitter I/O Register Summary

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

▨ = Unimplemented

Break Characters

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

The SCI recognizes a break character when a start bit is followed by 8 or 9 logic zero data bits and a logic zero 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

Idle Characters

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

If the TE bit is cleared during a transmission, the PTE0/TxD pin becomes idle after completion of the transmission in progress. Clearing and then setting the TE bit during a transmission queues an idle character to be sent after the character currently being transmitted.

NOTE: *When queueing an idle character, return the TE bit to logic one before the stop bit of the current character shifts out to the PTE0/TxD pin. Setting TE after the stop bit appears on PTE0/TxD causes data previously written to the SCDR to be lost.*

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

Inversion of Transmitted Output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at logic one. (See **15.8.1 SCI Control Register 1 (SCC1)**.)

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

15.4.3 Receiver

Figure 15-4 shows the structure of the SCI receiver.

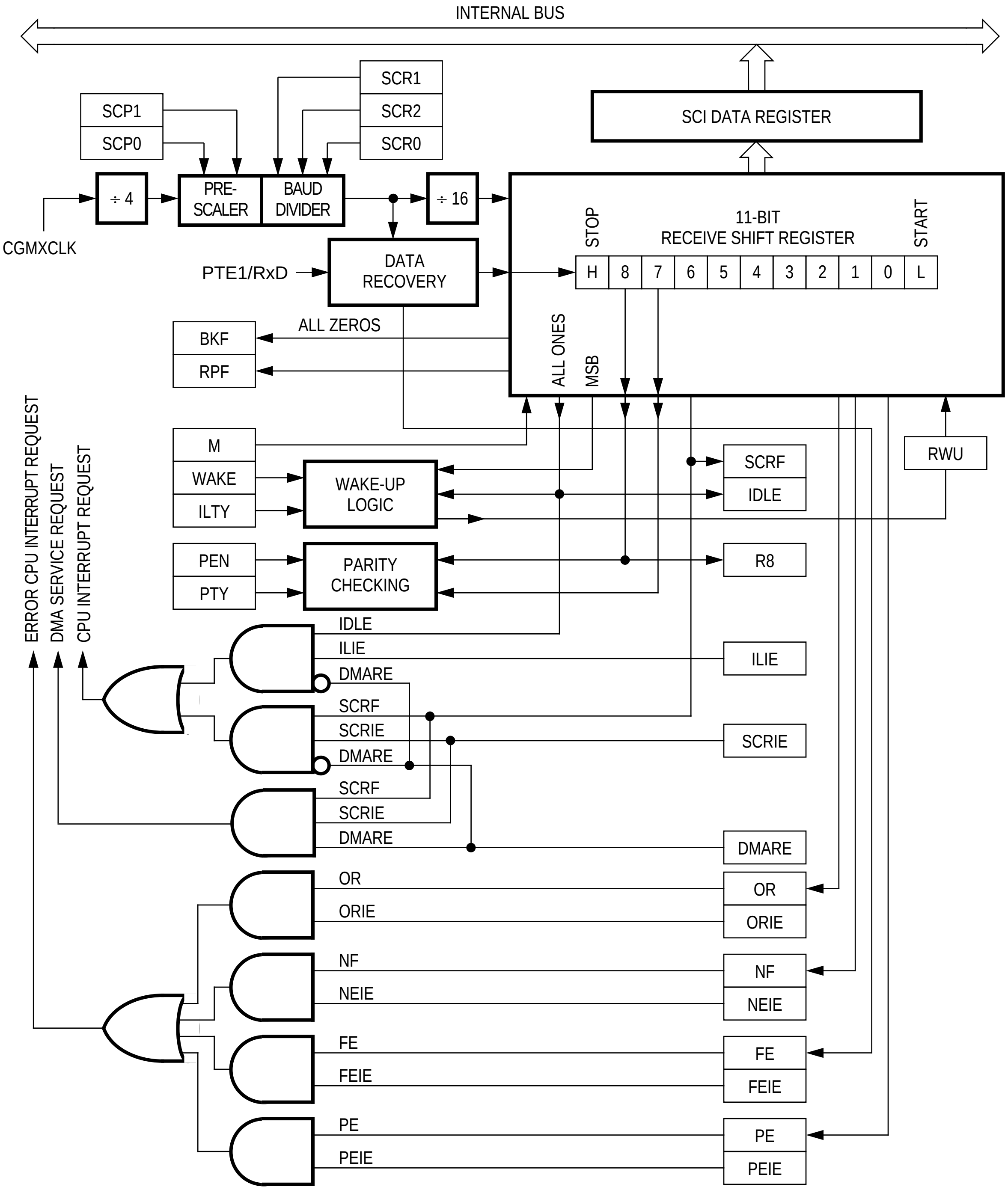


Figure 15-4. SCI Receiver Block Diagram

Table 15-3. SCI Receiver I/O Register Summary

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

▨ = Unimplemented

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).

Character Reception

During an SCI reception, the receive shift register shifts characters in from the PTE1/RxD pin. The SCI data register (SCDR) is the read-only buffer between the internal data bus and the receive shift register.

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

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

Data Sampling

The receiver samples the PTE1/RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see **Figure 14-5**):

- After every start bit
- After the receiver detects a data bit change from logic one to logic zero (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic one and the majority of the next RT8, RT9, and RT10 samples returns a valid logic zero)

To locate the start bit, data recovery logic does an asynchronous search for a logic zero preceded by three logic ones. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

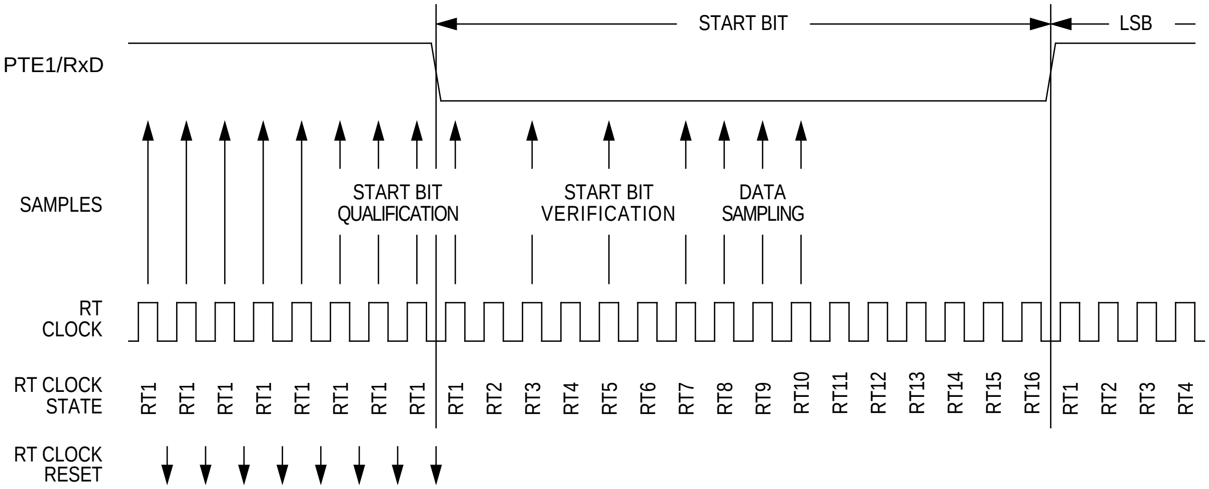


Figure 15-5. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. **Table 14-4** summarizes the results of the start bit verification samples.

Table 15-4. Start Bit Verification

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

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

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 15-5** summarizes the results of the data bit samples.

Table 15-5. Data Bit Recovery

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

NOTE: *The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic ones following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit.*

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 15-6** summarizes the results of the stop bit samples.

Table 15-6. 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

Framing Errors

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

Receiver Wake-Up

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 wake-up bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in SCC1, either of two conditions on the PTE1/RxD pin can bring the receiver out of the standby state:

- Address mark — An address mark is a logic one in the most significant bit position of a received character. When the WAKE bit is set, an address mark wakes the receiver from the standby state by clearing the RWU bit. The address mark also sets the SCI receiver full bit, SCRF. Software can then compare the character containing the address mark to the user-defined address of the receiver. If they are the same, the receiver remains awake and processes the characters that follow. If they are not the same, software can set the RWU bit and put the receiver back into the standby state.

- Idle input line condition — When the WAKE bit is clear, an idle character on the PTE1/RxD pin wakes the receiver from the standby state by clearing the RWU bit. The idle character that wakes the receiver does not set the receiver idle bit, IDLE, or the SCI receiver full bit, SCRF. The idle line type bit, ILTY, determines whether the receiver begins counting logic ones as idle character bits after the start bit or after the stop bit.

NOTE: *Clearing the WAKE bit after the PTE1/RxD pin has been idle may cause the receiver to wake up immediately.*

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 or a receiver DMA service request. Setting the SCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver CPU interrupts. Setting both the SCRIE bit and the DMA receive enable bit, DMARE, in SCC3 enables receiver DMA service requests and disables receiver CPU interrupt requests.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic ones shifted in from the PTE1/RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

NOTE: *When receiver DMA service requests are enabled (DMARE = 1), then receiver CPU interrupt requests are disabled, and the state of the ILIE bit has no effect.*

Error Interrupts

The following receiver error flags in SCS1 can generate CPU interrupt requests:

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate SCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the SCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate SCI error CPU interrupt requests.
- Framing error (FE) — The FE bit in SCS1 is set when a logic zero 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.

Error Flags During DMA Service Requests

When the DMA is servicing the SCI receiver, it clears the SCRF bit when it reads the SCI data register. The DMA does not clear the other status bits (BKF or RPF), nor does it clear error bits (OR, NF, FE, and PE). If the error bits are enabled to generate interrupt requests, the interrupt requests may accumulate during DMA servicing. To clear error bits while the DMA is servicing the receiver, enable SCI error CPU interrupts and clear the bits in an interrupt routine. Note the following latency considerations:

1. If interrupt latency is short enough for an error bit to be serviced before the next SCRF, then it can be determined which byte caused the error. If interrupt latency is long enough for a new SCRF to occur before servicing an error bit, then:
    a. It cannot be determined whether the error bit being serviced is due to the byte in the SCI data register or to a previous byte. Multiple errors can accumulate that correspond to different bytes. In a message-based system, you may have to repeat the entire message

    b. When the DMA is enabled to service the SCI receiver, merely reading the SCI data register clears the SCRF bit. The second step in clearing an error bit, reading the SCI data register, could inadvertently clear a new, unserviced SCRF that occurred during the error-servicing routine. Then the DMA would ignore the byte that set the new SCRF, and the new byte would be lost.
    To prevent clearing of an unserviced SCRF bit, clear the SCRIE bit at the beginning of the error-servicing interrupt routine and set it at the end. Clearing SCRIE disables DMA service so that both a read of SCS1 and a read of SCDR are required to clear the SCRF bit. Setting SCRIE enables DMA service so that the DMA can recognize a service request that occurred during the error-servicing interrupt routine.

    c. In the CPU interrupt routine to service error bits, do not use BRSET or BRCLR instructions. BRSET and BRCLR read the SCS1 register, which is the first step in clearing the register. Then the DMA could read the SCI data register, the second step in clearing it, thereby clearing all error bits. The next read of the data register would miss any error bits that were set.

2. DMA latency should be short enough so that an SCRF is serviced before the next SCRF occurs. If DMA latency is long enough for a new SCRF to occur before servicing an error bit, then:
   a. Overruns occur. Set the ORIE bit to enable SCI error CPU interrupt requests and service the overrun in an interrupt routine. In a message-based system, disable the DMA in the interrupt routine and manually recover. Otherwise, the byte that was lost in the overrun could prevent the DMA from reaching its byte count. If the DMA reaches it byte count in the following message, two messages may be corrupted.

   b. If the CPU does not service an overrun interrupt request, the DMA can eventually clear the SCRF bit by reading the SCI data register. The OR bit remains set. Each time a new byte sets the SCRF bit, new data transfers from the shift register to the SCI data register (provided that another overrun does not occur), even though the OR bit is set. The DMA removed the overrun condition by reading the data register, but the OR bit has not been cleared.

15.5 Low-Power Modes

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

15.5.1 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.

The DMA can service the SCI without exiting wait mode.

15.5.2 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 the MCU exits stop mode.

Because the internal clock is inactive during stop mode, entering stop mode during an SCI transmission or reception results in invalid data.

15.6 SCI During Break Module Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during interrupts generated by the break module. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

15.7 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

15.7.1 PTE0/TxD (Transmit Data)

The PTE0/TxD pin is the serial data output from the SCI transmitter. The SCI shares the PTE0/TxD pin with port E. When the SCI is enabled, the PTE0/TxD pin is an output regardless of the state of the DDRE2 bit in data direction register E (DDRE).

15.7.2 PTE1/RxD (Receive Data)

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

15.8 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)

15.8.1 SCI Control Register 1 (SCC1)

SCI control register 1 does the following:

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

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

Figure 15-6. SCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

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

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI Bit

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

1 = SCI enabled
0 = SCI disabled

TXINV — Transmit Inversion Bit

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

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

M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are 8 or 9 bits long. (See **Table 15-7**.) The ninth bit can serve as an extra stop bit, as a receiver wake-up signal, or as a parity bit. Reset clears the M bit.

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

WAKE — Wake-Up Condition Bit

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

1 = Address mark wake-up
0 = Idle line wake-up

ILTY — Idle Line Type Bit

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

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

PEN — Parity Enable Bit

This read/write bit enables the SCI parity function. (See **Table 15-7**.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See **Figure 15-2**.) Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See **Table 15-7**.) Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

NOTE: *Changing the PTY bit in the middle of a transmission or reception can generate a parity error.*

Table 15-7. Character Format Selection

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

15.8.2 SCI Control Register 2 (SCC2)

SCI control register 2 does the following:

- Enables the following CPU interrupt requests and DMA service requests:
  - Enables the SCTE bit to generate transmitter CPU interrupt requests or transmitter DMA service requests
  - Enables the TC bit to generate transmitter CPU interrupt requests
  - Enables the SCRF bit to generate receiver CPU interrupt requests or receiver DMA service requests
  - Enables the IDLE bit to generate receiver CPU interrupt requests
- Enables the transmitter
- Enables the receiver
- Enables SCI wake-up
- Transmits SCI break characters

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

Figure 15-7. 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 or DMA service requests. Setting the SCTIE bit and clearing the DMA transfer enable bit, DMATE, in SCC3 enables the SCTE bit to generate CPU interrupt requests. Setting both the SCTIE and DMATE bits enables the SCTE bit to generate DMA service requests. Reset clears the SCTIE bit.

1 = SCTE enabled to generate CPU interrupt or DMA service requests
0 = SCTE not enabled to generate CPU interrupt or DMA service requests

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 or SCI receiver DMA service requests. Setting the SCRIE bit and clearing the DMA receive enable bit, DMARE, in SCC3 enables the SCRF bit to generate CPU interrupt requests. Setting both SCRIE and DMARE enables SCRF to generate DMA service requests. Reset clears the SCRIE bit.

1 = SCRF enabled to generate CPU interrupt or DMA service requests
0 = SCRF not enabled to generate CPU interrupt or DMA service requests

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

NOTE: *When SCI receiver DMA service requests are enabled (DMARE = 1), then SCI receiver CPU interrupt requests are disabled, and the state of the ILIE bit has no effect.*

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 logic ones from the transmit shift register to the PTE0/TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the PTE0/TxD returns to the idle condition (logic one). 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 Wake-Up Bit

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

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

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

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

NOTE: *Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK too early causes the SCI to send a break character instead of a preamble.*

15.8.3 SCI Control Register 3 (SCC3)

SCI control register 3 does the following:

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

		Bit 7	6	5	4	3	2	1	Bit 0
SCC3 $0015	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
	Write:	▨							
	Reset:	U	U	0	0	0	0	0	0

▨ = Unimplemented U = Unaffected

Figure 15-8. 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.

DMARE — DMA Receive Enable Bit

This read/write bit enables the DMA to service SCI receiver DMA service requests generated by the SCRF bit. (See 14.7.4.) Setting the DMARE bit disables SCI receiver CPU interrupt requests. Reset clears the DMARE bit.

1 = DMA enabled to service SCI receiver DMA service requests generated by the SCRF bit
(SCI receiver CPU interrupt requests disabled)
0 = DMA not enabled to service SCI receiver DMA service requests generated by the SCRF bit
(SCI receiver CPU interrupt requests enabled)

DMATE — DMA Transfer Enable Bit

This read/write bit enables SCI transmitter empty (SCTE) DMA service requests. (See **15.8.4 SCI Status Register 1 (SCS1)**.) Setting the DMATE bit disables SCTE CPU interrupt requests. Reset clears DMATE.

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

ORIE — Receiver Overrun Interrupt Enable Bit

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

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

NEIE — Receiver Noise Error Interrupt Enable Bit

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

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

FEIE — Receiver Framing Error Interrupt Enable Bit

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

1 = SCI error CPU interrupt requests from FE bit enabled

0 = SCI error CPU interrupt requests from FE bit disabled

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables SCI receiver CPU interrupt requests generated by the parity error bit, PE. (See 13.7.4.) Reset clears PEIE.

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

15.8.4 SCI Status Register 1 (SCS1)

SCI status register 1 contains flags to signal the following conditions:

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

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

▨ = Unimplemented

Figure 15-9. SCI Status Register 1 (SCS1)

SCTE — SCI Transmitter Empty Bit

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an SCI transmitter CPU interrupt request or an SCI transmitter DMA service request. When the SCTIE bit in SCC2 is set and the DMATE bit in

SCC3 is clear, SCTE generates an SCI transmitter CPU interrupt request. With both the SCTIE and DMATE bits set, SCTE generates an SCI transmitter DMA service request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR. In DMA transfers, the DMA automatically clears the SCTE bit when it writes to the SCDR. Reset sets the SCTE bit.

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

NOTE: *Setting the TE bit for the first time also sets the SCTE bit. When enabling SCI transmitter DMA service requests, set the TE bit **after** setting the DMATE bit. Otherwise setting the TE and SCTIE bits generates an SCI transmitter CPU interrupt request instead of a DMA service request.*

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. When the DMA services an SCI transmitter DMA service request, the DMA clears the TC bit by writing to the SCDR. TC is automatically cleared when data, preamble or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — SCI Receiver Full Bit

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

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

IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive logic ones appear on the receiver input. IDLE generates an SCI error CPU interrupt request if the ILIE bit in SCC2 is also set and the DMARE bit in SCC3 is clear. Clear the IDLE bit by reading SCS1 with IDLE set and then reading the SCDR. After the receiver is enabled, it must receive a valid character that sets the SCRF bit before an idle condition can set the IDLE bit. Also, after the IDLE bit has been cleared, a valid character must again set the SCRF bit before an idle condition can set the IDLE bit. Reset clears the IDLE bit.

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

OR — Receiver Overrun Bit

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

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

Software latency may allow an overrun to occur between reads of SCS1 and SCDR in the flag-clearing sequence. **Figure 15-10** 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.

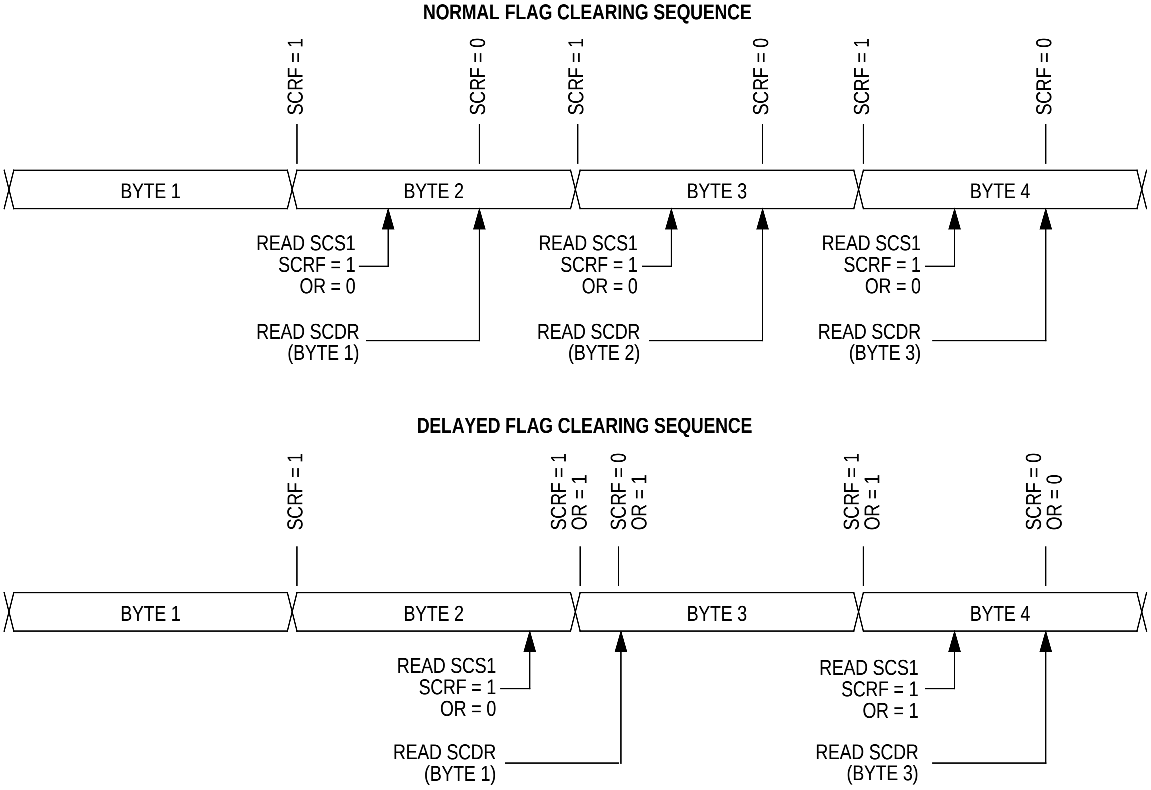


Figure 15-10. Flag Clearing Sequence

NF — Receiver Noise Flag Bit

This clearable, read-only bit is set when the SCI detects noise on the PTE1/RxD pin. NF generates an NF CPU interrupt request if the NEIE bit in SCC3 is also set. Clear the NF bit by reading SCS1 and then reading the SCDR. Reset clears the NF bit.

1 = Noise detected
0 = No noise detected

FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a logic 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

15.8.5 SCI Status Register 2 (SCS2)

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

- Break character detected
- Incoming data

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

▨ = Unimplemented

Figure 15-11. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the PTE1/RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate a CPU interrupt request or a DMA service request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after logic ones again appear on the PTE1/RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF —Reception in Progress Flag Bit

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

1 = Reception in progress
0 = No reception in progress

15.8.6 SCI Data Register (SCDR)

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

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

Figure 15-12. 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.

15.8.7 SCI Baud Rate Register (SCBR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
SCBR $0019	Read: Write:	▨	▨	SCP1	SCP0	▨	SCR2	SCR1	SCR0
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 15-13. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in **Table 15-8**. Reset clears SCP1 and SCP0.

Table 15-8. SCI Baud Rate Prescaling

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

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in **Table 15-9**. Reset clears SCR2–SCR0.

Table 15-9. SCI Baud Rate Selection

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

Use the following formula to calculate the SCI baud rate:

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

where:

f_{XCLK} = crystal frequency
PD = prescaler divisor
BD = baud rate divisor

Table 15-10 shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

Table 15-10. SCI Baud Rate Selection Examples

SCP1:0	Prescaler Divisor (PD)	SCR2:1:0	Baud Rate Divisor (BD)	Baud Rate (f_{XCLK} = 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

Table 15-10. SCI Baud Rate Selection Examples (Continued)

SCP1:0	Prescaler Divisor (PD)	SCR2:1:0	Baud Rate Divisor (BD)	Baud Rate (f_{XCLK} = 4.9152 MHz)
10	4	011	8	2400
10	4	100	16	1200
10	4	101	32	600
10	4	110	64	300
10	4	111	128	150
11	13	000	1	5908
11	13	001	2	2954
11	13	010	4	1477
11	13	011	8	739
11	13	100	16	369
11	13	101	32	185
11	13	110	64	92
11	13	111	128	46

Section 16. Serial Peripheral Interface (SPI)

16.1 Contents

16.2 Introduction

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

16.3 Features

Features of the SPI module include the following:

- Full-Duplex Operation
- Master and Slave Modes
- Double-Buffered Operation With Separate Transmit and Receive Registers
- Four Master Mode Frequencies (Maximum = Bus Frequency ÷ 2)
- Maximum Slave Mode Frequency = Bus Frequency
- Separate Clock Ground for Reduced Radio Frequency (RF) Interference
- Serial Clock with Programmable Polarity and Phase
- Two Separately Enabled Interrupts with DMA or 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^{2}C (Inter-Integrated Circuit) Compatibility

16.4 Pin Name Conventions and I/O Register Addresses

The text that follows describes the SPI I/O pin names — $\overline{SS}$ (slave select), SPSCK (SPI serial clock), MOSI (master out slave in), and MISO (master in slave out). The SPI shares three input/output (I/O) pins with one parallel I/O port.

Refer to **Table 16-1** for the I/O register addresses

Table 16-1. I/O Register Addresses

Register Name	Register Address
SPI Control Register (SPICR)	$0010
SPI Status and Control Register (SPISCR)	$0011
SPI Data Register (SPIDR)	$0012

16.5 Functional Description

Figure 16-1 summarizes the SPI I/O registers and **Figure 16-2** shows the structure of the SPI module.

Register Name	R/W	Bit 7	6	5	4	3	2	1	Bit 0
SPI Control Register (SPCR)	Read:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
	Write:								
	Reset:	0	0	1	0	0	0	0	0
SPI Status and Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
	Write:	▨		▨	▨	▨			
	Reset:	0	0	0	0	1	0	0	0
SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Unaffected by reset							

Figure 16-1. SPI I/O Register Summary

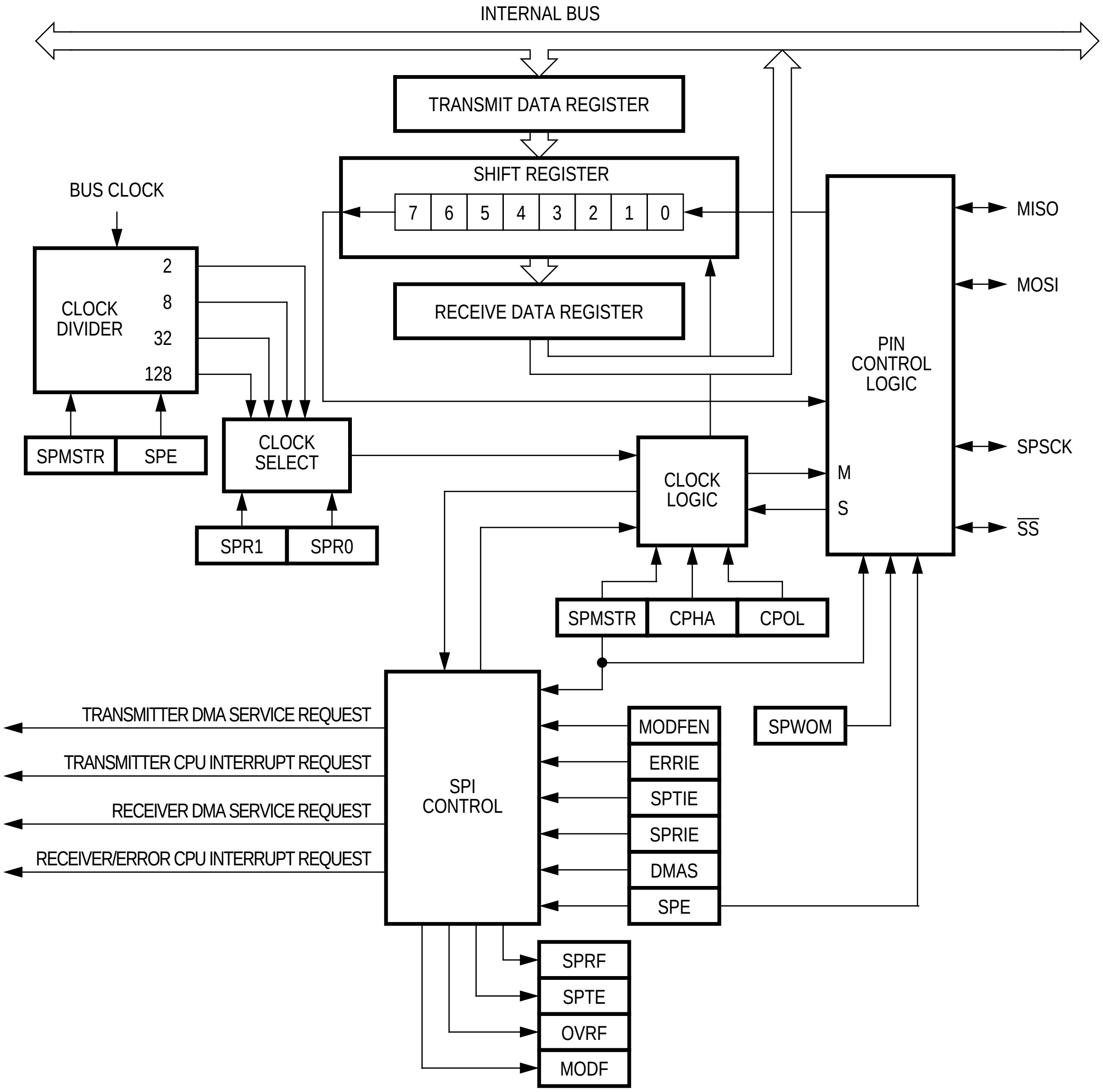


Figure 16-2. SPI Module Block Diagram

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs. Software can poll the SPI status flags or SPI operation can be interrupt-driven. All SPI interrupts can be serviced by the CPU, and the transmitter empty (SPTE) and receiver full (SPRF) flags can also be configured for DMA service.

NOTE: *DMA section and associated functions are only valid if the MCU has a DMA module.*

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

The following paragraphs describe the operation of the SPI module.

16.5.1 Master Mode

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

NOTE: *Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. (See* ***16.14.1 SPI Control Register (SPCR)****.)*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. (See **Figure 16-3**.)

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See **16.14.2 SPI Status and Control Register (SPSCR)**.) Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

As the byte shifts out on the MOSI pin of the master, another byte shifts in from the slave on the master's MISO pin. The transmission ends when the receiver full bit, SPRF, becomes set. At the same time that SPRF becomes set, the byte from the slave transfers to the receive data register. In normal operation, SPRF signals the end of a transmission. Software clears SPRF by reading the SPI status and control register and then reading the SPI data register.

When the DMAS bit is set, a read of the SPI data register by the CPU or by the DMA clears the SPRF bit. Regardless of the state of the DMAS bit, a write to the SPI data register by the CPU or by the DMA clears the SPTE bit.

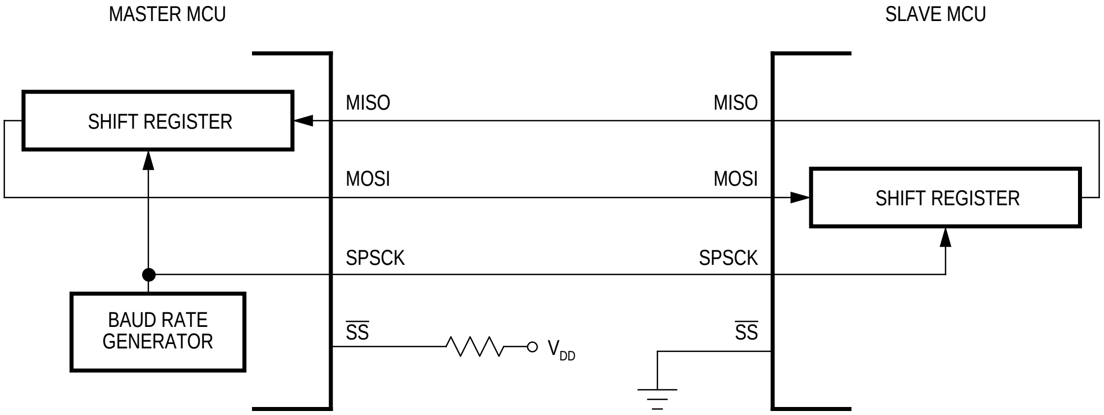


Figure 16-3. Full-Duplex Master-Slave Connections

16.5.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{\text{SS}}$ pin of the slave MCU must be at logic zero. $\overline{\text{SS}}$ must remain low until the transmission is complete. (See **16.7.2 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 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.

A slave SPI must complete the write to the data register at least one bus cycle before the master SPI starts a 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 $\overline{\text{SS}}$ starts a transmission. (See **16.6 Transmission Formats**.)

If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

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

16.6 Transmission Formats

During an SPI transmission, data is simultaneously transmitted (shifted out serially) and received (shifted in serially). A serial clock line synchronizes shifting and sampling on the two serial data lines. A slave select line allows individual selection of a slave SPI device; slave devices that are not selected do not interfere with SPI bus activities. On a master SPI device, the slave select line can optionally be used to indicate a multiple-master bus contention.

16.6.1 Clock Phase and Polarity Controls

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

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

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

16.6.2 Transmission Format When CPHA = 0

Figure 16-4 shows an SPI transmission in which CPHA is logic zero. 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 at logic zero, 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 **16.7.2 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 between each byte transmitted.

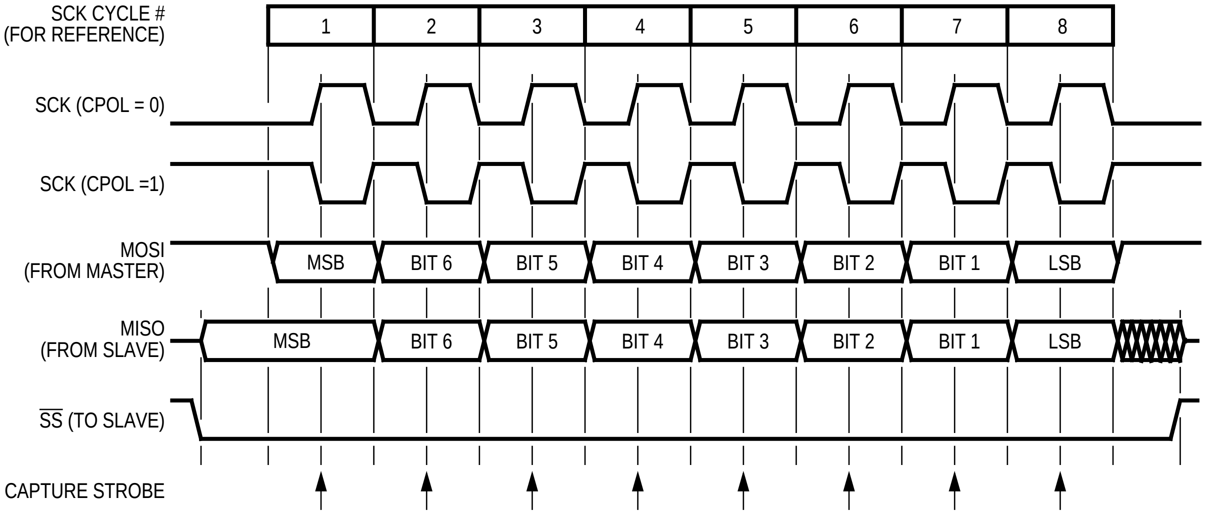


Figure 16-4. Transmission Format (CPHA = 0)

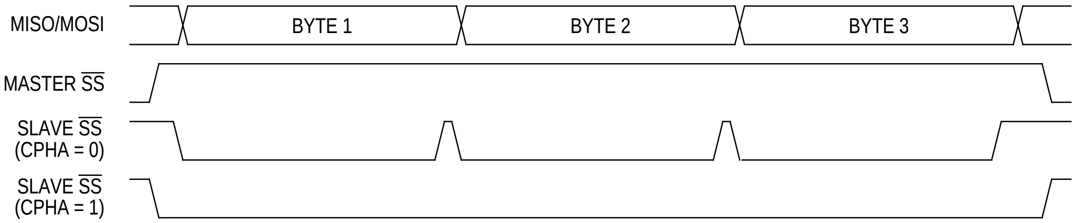


Figure 16-5. CPHA/$\overline{SS}$ Timing

16.6.3 Transmission Format When CPHA = 1

Figure 16-6 shows an SPI transmission in which CPHA is logic one. 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 $\overline{\text{SS}}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{\text{SS}}$) is at logic zero, so that only the selected slave drives to the master. The $\overline{\text{SS}}$ pin of the master is not shown but is assumed to be inactive. The $\overline{\text{SS}}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See **16.7.2 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 $\overline{\text{SS}}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

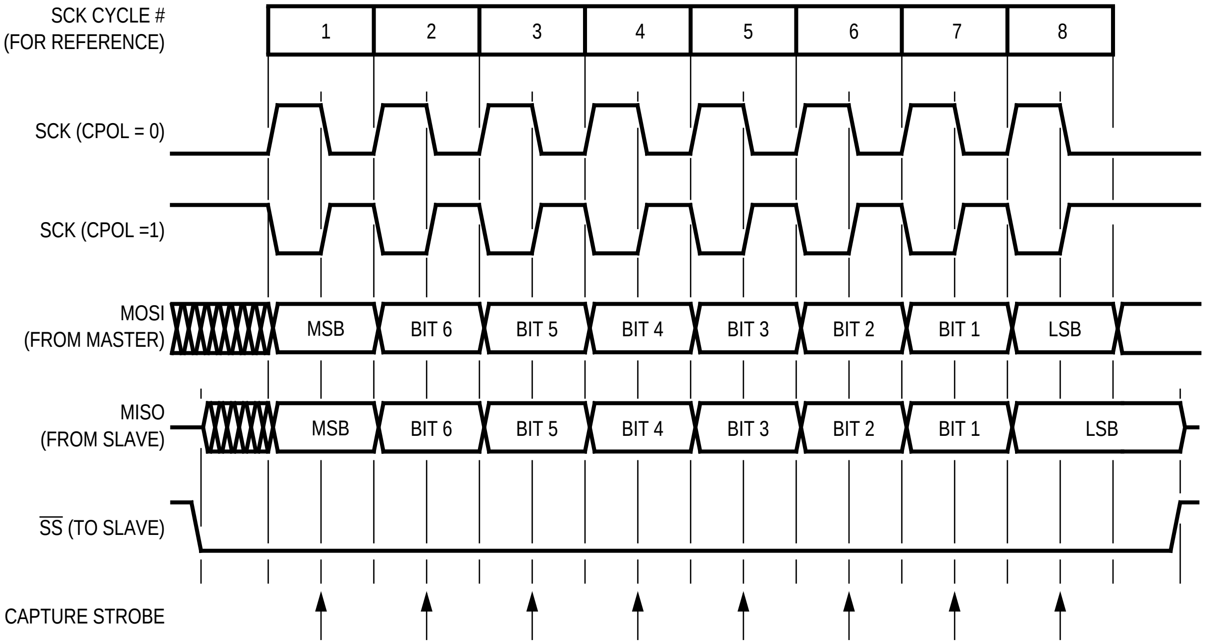


Figure 16-6. Transmission Format (CPHA = 1)

16.6.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), transmissions are started by a software write to the SPDR. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When CPHA = 0, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA = 1, the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See **Figure 16-7**.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. It is only enabled when both the SPE and SPMSTR bits are set to conserve power. SCK edges occur halfway through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in **Figure 16-7**. This delay will be no longer than a single SPI bit time. That is, the maximum delay 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.

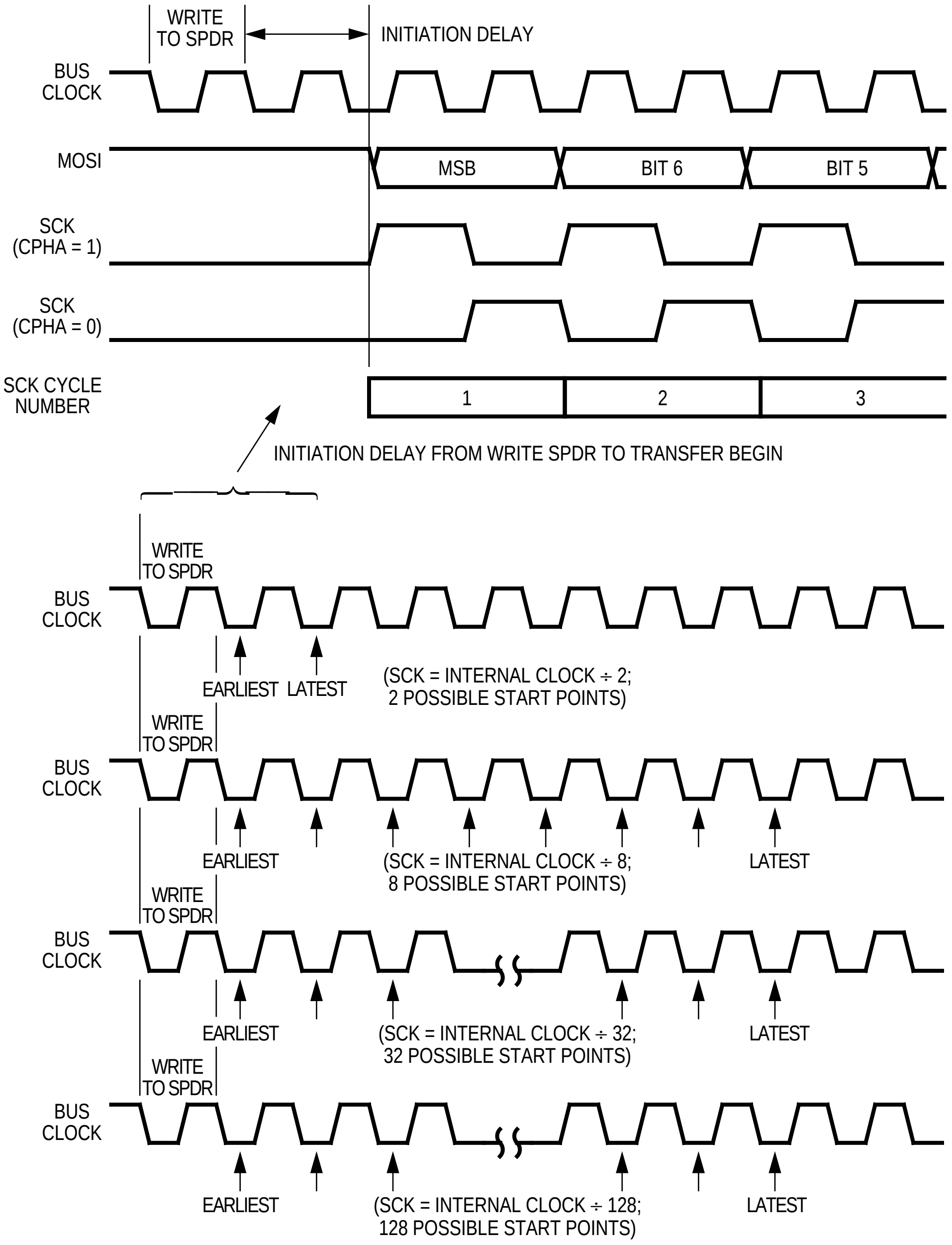

WRITE TO SPDR
INITIATION DELAY
BUS CLOCK
MOSI
MSB
BIT 6
BIT 5
SCK (CPHA = 1)
SCK (CPHA = 0)
SCK CYCLE NUMBER
1
2
3
INITIATION DELAY FROM WRITE SPDR TO TRANSFER BEGIN
WRITE TO SPDR
BUS CLOCK
EARLIEST
LATEST
(SCK = INTERNAL CLOCK ÷ 2; 2 POSSIBLE START POINTS)
WRITE TO SPDR
BUS CLOCK
EARLIEST
(SCK = INTERNAL CLOCK ÷ 8; 8 POSSIBLE START POINTS)
LATEST
WRITE TO SPDR
BUS CLOCK
EARLIEST
(SCK = INTERNAL CLOCK ÷ 32; 32 POSSIBLE START POINTS)
LATEST
WRITE TO SPDR
BUS CLOCK
EARLIEST
(SCK = INTERNAL CLOCK ÷ 128; 128 POSSIBLE START POINTS)
LATEST

Figure 16-7. Transmission Start Delay (Master)

16.7 Error Conditions

The following flags signal SPI error conditions:

- Overflow (OVRF) — Failing to read the SPI data register before the next byte enters the shift register 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.
- Mode fault error (MODF) — The MODF bit indicates that the voltage on the slave select pin ($\overline{SS}$) is inconsistent with the mode of the SPI. MODF is in the SPI status and control register.

16.7.1 Overflow Error

The overflow flag (OVRF) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. (See **Figure 16-4** and **Figure 16-6**.) If an overflow occurs, the data being received is not transferred to the receive data register so that the unread data can still be read. Therefore, an overflow error always indicates the loss of data.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request, and MODF and OVRF can generate a receiver/error CPU interrupt request. (See **Figure 16-11**.) 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.

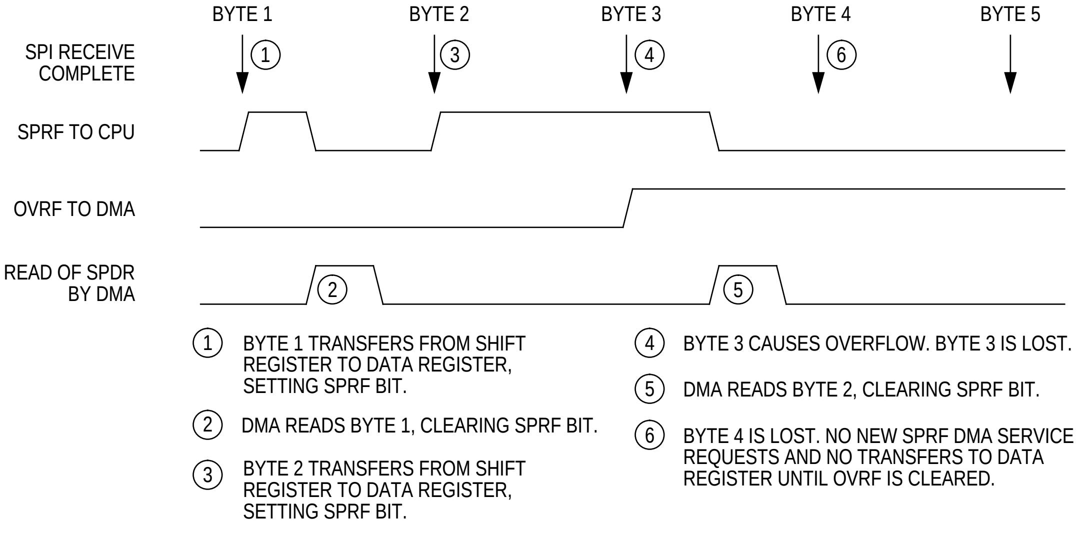
When the DMA is enabled to service the SPRF flag, it clears SPRF when it reads the SPI data register. The OVRF flag, however, still requires the two-step clearing mechanism of reading the flag when it is set and then reading the SPI data register. In this way, the DMA cannot directly clear the OVRF. However, if the CPU reads the SPI status and control register with the OVRF bit set, and then the DMA reads the SPI data register, the OVRF bit is cleared.

Conversely, if the CPU reads the data register to clear the OVRF flag, it could clear a pending SPRF service request to the DMA. Even if the DMA clears an SPRF, no new data will be transferred from the shift register to the data register with the OVRF high. This means that no new SPRF interrupts will be generated until this OVRF is cleared. For this reason, the OVRF interrupt to the CPU should be enabled when using the DMA to service the SPRF if there is any chance that the overflow condition might occur. (See **Figure 16-8**.)

The overflow service routine may need to disable the DMA and manually recover since an overflow indicates the loss of data. Loss of data may prevent the DMA from reaching its byte count.

If an end-of-block transmission interrupt from the DMA 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 16-9** shows how it is possible to miss an overflow.



1 BYTE 1 TRANSFERS FROM SHIFT REGISTER TO DATA REGISTER, SETTING SPRF BIT.

2 DMA READS BYTE 1, CLEARING SPRF BIT.

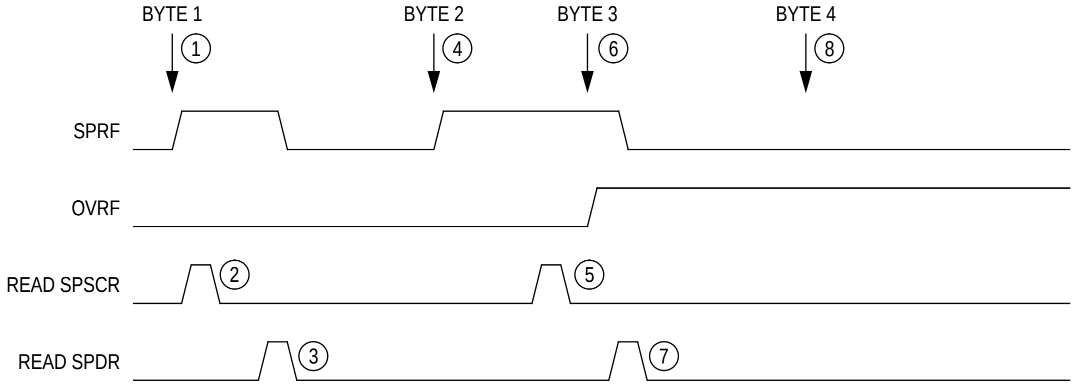
3 BYTE 2 TRANSFERS FROM SHIFT REGISTER TO DATA REGISTER, SETTING SPRF BIT.

4 BYTE 3 CAUSES OVERFLOW. BYTE 3 IS LOST.

5 DMA READS BYTE 2, CLEARING SPRF BIT.

6 BYTE 4 IS LOST. NO NEW SPRF DMA SERVICE REQUESTS AND NO TRANSFERS TO DATA REGISTER UNTIL OVRF IS CLEARED.

Figure 16-8. Overflow Condition with DMA Service of SPRF



① BYTE 1 SETS SPRF BIT.

② CPU READS SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.

③ CPU READS BYTE 1 IN SPDR, CLEARING SPRF BIT.

④ BYTE 2 SETS SPRF BIT.

⑤ CPU READS SPSCRW WITH SPRF BIT SET AND OVRF BIT CLEAR.

⑥ BYTE 3 SETS OVRF BIT. BYTE 3 IS LOST.

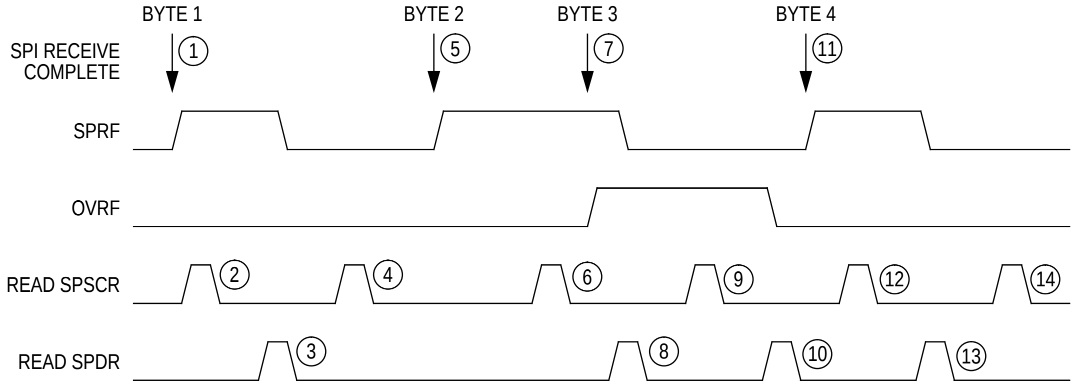
⑦ CPU READS BYTE 2 IN SPDR, CLEARING SPRF BIT, BUT NOT OVRF BIT.

⑧ BYTE 4 FAILS TO SET SPRF BIT BECAUSE OVRF BIT IS SET. BYTE 4 IS LOST.

Figure 16-9. Missed Read of Overflow Condition

The first part of **Figure 16-9** 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 easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will complete with an SPRF interrupt. **Figure 16-10** illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.



1 BYTE 1 SETS SPRF BIT.

2 CPU READS SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.

3 CPU READS BYTE 1 IN SPDR, CLEARING SPRF BIT.

4 CPU READS SPSCR AGAIN TO CHECK OVRF BIT.

5 BYTE 2 SETS SPRF BIT.

6 CPU READS SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.

7 BYTE 3 SETS OVRF BIT. BYTE 3 IS LOST.

8 CPU READS BYTE 2 IN SPDR, CLEARING SPRF BIT.

9 CPU READS SPSCR AGAIN TO CHECK OVRF BIT.

10 CPU READS BYTE 2 SPDR, CLEARING OVRF BIT.

11 BYTE 4 SETS SPRF BIT.

12 CPU READS SPSCR.

13 CPU READS BYTE 4 IN SPDR, CLEARING SPRF BIT.

14 CPU READS SPSCR AGAIN TO CHECK OVRF BIT.

Figure 16-10. Clearing SPRF When OVRF Interrupt Is Not Enabled

16.7.2 Mode Fault Error

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request instead of a CPU interrupt request, but MODF and OVRF can generate a receiver/error CPU interrupt request. (See **Figure 16-11**.) 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 $\overline{\text{SS}}$ goes to logic zero. 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.*

NOTE: *Setting the MODF flag does not clear the SPMSTR bit. Reading SPMSTR when MODF = 1 will indicate a MODE fault error occurred in either master or slave mode.*

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{\text{SS}}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{\text{SS}}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{\text{SS}}$ is already low. The transmission continues until the SPSCK returns to its IDLE level following the shift of the last data bit. (See
16.6 Transmission Formats.)

NOTE: *When CPHA = 0, a MODF occurs if a slave is selected ($\overline{\text{SS}}$ is at logic 0) and later unselected ($\overline{\text{SS}}$ is at logic 1) even if no SPSCK is sent to that slave. This happens because $\overline{\text{SS}}$ at logic 0 indicates the start of the transmission (MISO driven out with the value of MSB) for CPHA = 0. When CPHA = 1, a slave can be selected and then later unselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.*

In a slave SPI (MSTR = 0), the MODF bit generates an SPI receiver/error CPU interrupt request if the ERRIE bit is set. The MODF bit does not clear the SPE bit or reset the SPI in any way. Software can abort the SPI transmission by toggling the SPE bit of the slave.

NOTE: *A logic one voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

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

16.8 Interrupts

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

Table 16-2. SPI Interrupts

Flag	Request
SPTE (Transmitter Empty)	SPI Transmitter CPU Interrupt Request (DMAS = 0, SPTIE = 1) SPI Transmitter DMA Service Request (DMAS = 1, SPTIE = 1)
SPRF (Receiver Full)	SPI Receiver CPU Interrupt Request (DMAS = 0, SPRIE = 1) SPI Receiver DMA Service Request (DMAS = 1, SPRIE = 1)
OVRF (Overflow)	SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1)
MODF (Mode Fault)	SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1, MODFEN = 1)

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

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests or transmitter DMA service requests.

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt requests or receiver DMA service requests, provided that the SPI is enabled (SPE = 1).

The error interrupt enable bit (ERRIE) enables both the MODF and OVRF flags to generate a receiver/error CPU interrupt request.

The mode fault enable bit (MODFEN) can prevent the MODF flag from being set so that only the OVRF flag is enabled to generate receiver/error CPU interrupt requests.

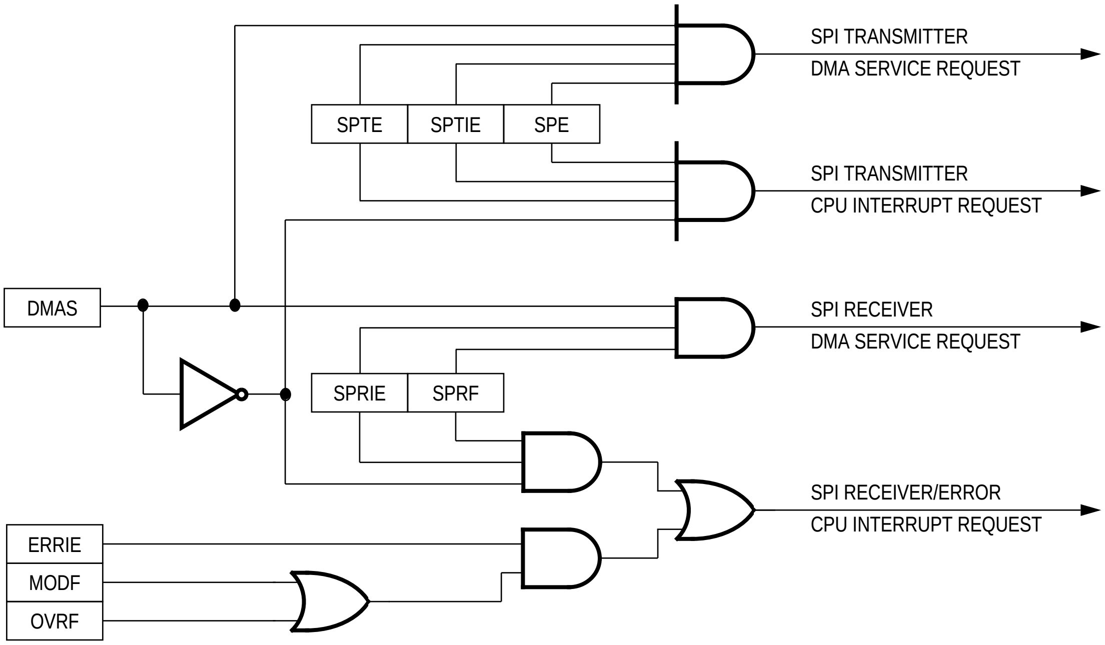


Figure 16-11. SPI Interrupt Request Generation

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

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

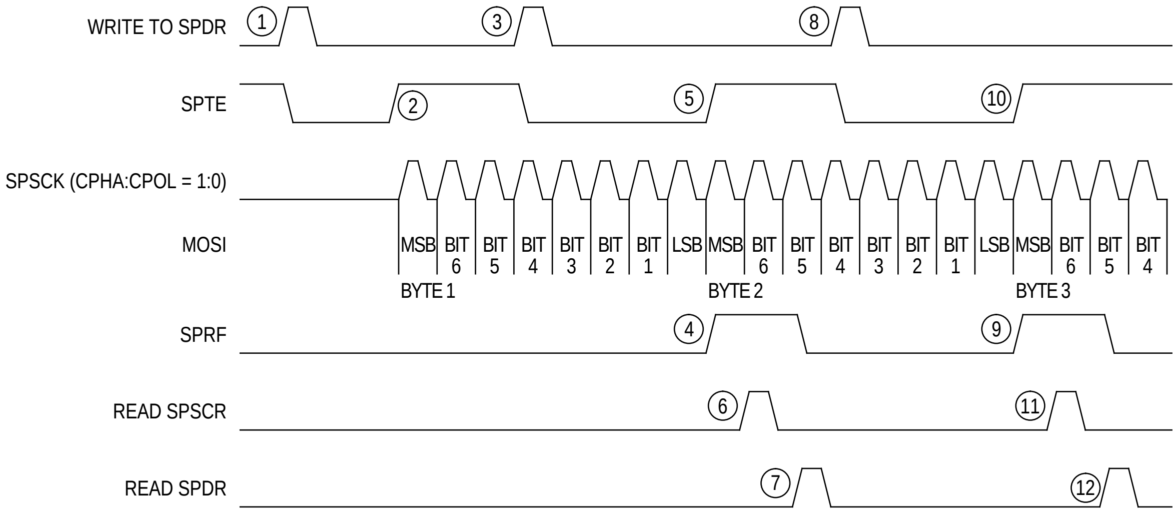
  If the DMA select bit, DMAS, is clear, SPRF generates an SPRF CPU interrupt request. If DMAS is set, SPRF generates an SPRF DMA service request.
- SPI transmitter empty (SPTE) — The SPTE bit becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE can generate either an SPTE CPU interrupt request or an SPTE DMA service request.

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

16.9 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the SPI data register only when the SPTE bit is high. **Figure 16-12** shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

Queuing Transmission Data



1. CPU WRITES BYTE 1 TO SPDR, CLEARING SPTE BIT.
2. BYTE 1 TRANSFERS FROM TRANSMIT DATA REGISTER TO SHIFT REGISTER, SETTING SPTE BIT.
3. CPU WRITES BYTE 2 TO SPDR, QUEUEING BYTE 2 AND CLEARING SPTE BIT.
4. FIRST INCOMING BYTE TRANSFERS FROM SHIFT REGISTER TO RECEIVE DATA REGISTER, SETTING SPRF BIT.
5. BYTE 2 TRANSFERS FROM TRANSMIT DATA REGISTER TO SHIFT REGISTER, SETTING SPTE BIT.
6. CPU READS SPSCR WITH SPRF BIT SET.
7. CPU READS SPDR, CLEARING SPRF BIT.
8. CPU WRITES BYTE 3 TO SPDR, QUEUEING BYTE 3 AND CLEARING SPTE BIT.
9. SECOND INCOMING BYTE TRANSFERS FROM SHIFT REGISTER TO RECEIVE DATA REGISTER, SETTING SPRF BIT.
10. BYTE 3 TRANSFERS FROM TRANSMIT DATA REGISTER TO SHIFT REGISTER, SETTING SPTE BIT.
11. CPU READS SPSCR WITH SPRF BIT SET.
12. CPU READS SPDR, CLEARING SPRF BIT.

Figure 16-12. 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.

16.10 Resetting the SPI

Any system reset completely resets the SPI. Partial resets occur whenever the SPI enable bit (SPE) is low. Whenever SPE is low, the following occurs:

- The SPTE flag is set
- Any transmission currently in progress is aborted
- The shift register is cleared
- The SPI state counter is cleared, making it ready for a new complete transmission
- All the SPI port logic is defaulted back to being general purpose I/O.

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 set all control bits again when SPE is set back high for the next transmission.

By not resetting the SPRF, OVRF, and MODF flags, the user can still service these interrupts after the SPI has been disabled. The user can disable the SPI by writing 0 to the SPE bit. The SPI can also be disabled by a mode fault occurring in an SPI that was configured as a master with the MODFEN bit set.

16.11 Low-Power Modes

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

16.11.1 Wait Mode

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

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

The DMA can service DMA service requests generated by the SPTE and SPRF flags without exiting wait mode. 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 **16.8 Interrupts**.)

16.11.2 Stop Mode

The SPI module is inactive after the execution of a STOP instruction. The STOP instruction does not affect register conditions. SPI operation resumes after the MCU exits stop mode. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

16.12 SPI During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

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.

16.13 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
- $\overline{\text{SS}}$ — 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}.

16.13.1 MISO (Master In/Slave Out)

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

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

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

16.13.2 MOSI (Master Out/Slave In)

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

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

16.13.3 SPSCK (Serial Clock)

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

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

16.13.4 $\overline{\text{SS}}$ (Slave Select)

The $\overline{\text{SS}}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{\text{SS}}$ is used to select a slave. For CPHA = 0, the $\overline{\text{SS}}$ is used to define the start of a transmission. (See **16.6 Transmission Formats**.) Since it is used to indicate the start of a transmission, the $\overline{\text{SS}}$ 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 16-13**.

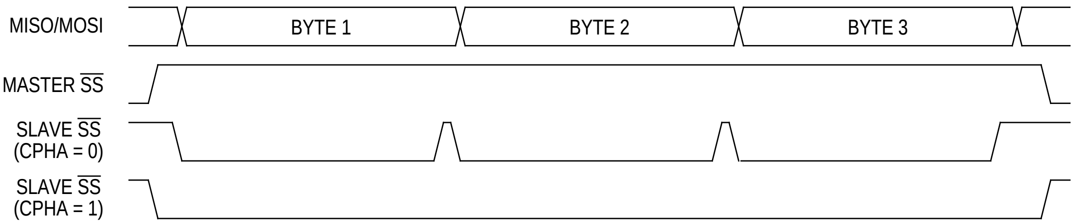

MISO/MOSI
BYTE 1
BYTE 2
BYTE 3
MASTER $\overline{\text{SS}}$
SLAVE $\overline{\text{SS}}$ (CPHA = 0)
SLAVE $\overline{\text{SS}}$ (CPHA = 1)

Figure 16-13. CPHA/$\overline{\text{SS}}$ Timing

When an SPI is configured as a slave, the $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ from creating a MODF error. (See **16.14.2 SPI Status and Control Register (SPSCR)**.)

NOTE: *A logic one voltage on the $\overline{\text{SS}}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

When an SPI is configured as a master, the $\overline{\text{SS}}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See **16.7.2 Mode Fault Error**.) For the state of the $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ LOGIC
0	X	X	Not Enabled	General-purpose I/O; $\overline{\text{SS}}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; $\overline{\text{SS}}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

X = don’t care

16.13.5 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.

16.14 I/O Registers

Three registers control and monitor SPI operation:

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

16.14.1 SPI Control Register (SPCR)

The SPI control register does the following:

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

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

Figure 16-14. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

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

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

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

DMAS —DMA Select Bit

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

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

SPMSTR — SPI Master Bit

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

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See **Figure 16-4** and **Figure 16-6**.) To transmit data between SPI modules, the SPI modules must have identical CPOL 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 16-6**.) To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the $\overline{\text{SS}}$ pin of the slave SPI module must be set to logic one between bytes. (See **Figure 16-13**.)Reset sets the CPHA bit.

When CPHA = 0 for a slave, the falling edge of $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$. 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 $\overline{\text{SS}}$ 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 **16.7.2 Mode Fault Error**.) A logic one on the $\overline{\text{SS}}$ 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 pull-up devices on pins SPSCK, MOSI, and MISO so that those pins become open-drain outputs.

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

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See **16.10 Resetting the SPI**.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

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

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

16.14.2 SPI Status and Control Register (SPSCR)

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

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

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

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

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

▨ = Unimplemented

Figure 16-15. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

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

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

NOTE: *When the DMA is configured to service the SPI (DMAS = 1), a read by the CPU of the SPI data register can inadvertently clear the SPRF bit and cause the DMA to miss a service request.*

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, ready-only flag is set in a slave SPI if the $\overline{\text{SS}}$ pin goes high during a transmission. In a master SPI, the MODF flag is set if the $\overline{\text{SS}}$ 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 = $\overline{\text{SS}}$ pin at inappropriate logic level

0 = $\overline{\text{SS}}$ 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 or an SPTE DMA service 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.*

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

NOTE: *When the DMA is configured to service the SPI (DMAS = 1), a write by the CPU of the SPI data register can inadvertently clear the SPTE bit and cause the DMA to miss a service request.*

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 $\overline{\text{SS}}$ pin is available as a general purpose I/O.

If the MODFEN bit is set, then this pin is not available as a general purpose I/O. When the SPI is enabled as a slave, the $\overline{SS}$ pin is not available as a general purpose I/O regardless of the value of MODFEN. (See **16.13.4 SS (Slave Select)**.)

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. (See **16.7.2 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 the following formula to calculate the SPI baud rate:

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

where:

CGMOUT = base clock output of the clock generator module (CGM)

BD = baud rate divisor

16.14.3 SPI Data Register (SPDR)

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate buffers that can contain different values. See **Figure 16-2**.

		Bit 7	6	5	4	3	2	1	Bit 0
SPDR $0012	Read:	R7	R6	R5	R4	R3	R2	R1	R0
	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	Reset:	Indeterminate after reset							

Figure 16-16. 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.*

Section 17. Timer Interface (TIM)

17.1 Contents

17.2 Introduction

This section describes the timer interface module (TIM6). The TIM is a six-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. **Figure 17-1** is a block diagram of the TIM.

17.3 Features

Features of the TIM include the following:

- 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
- DMA Service Request Generation
- Modular Architecture Expandable to Eight Channels

NOTE: *DMA section and associated functions are only valid if the MCU has a DMA module.*

17.4 Functional Description

Figure 17-1 shows the structure of the TIM. The central component of the TIM is the 16-bit TIM counter that can operate as a free-running counter or a modulo up-counter. The TIM counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH:TMODL, control the modulo value of the TIM counter. Software can read the TIM counter value at any time without affecting the counting sequence.

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

17.4.1 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.

17.4.2 Input Capture

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

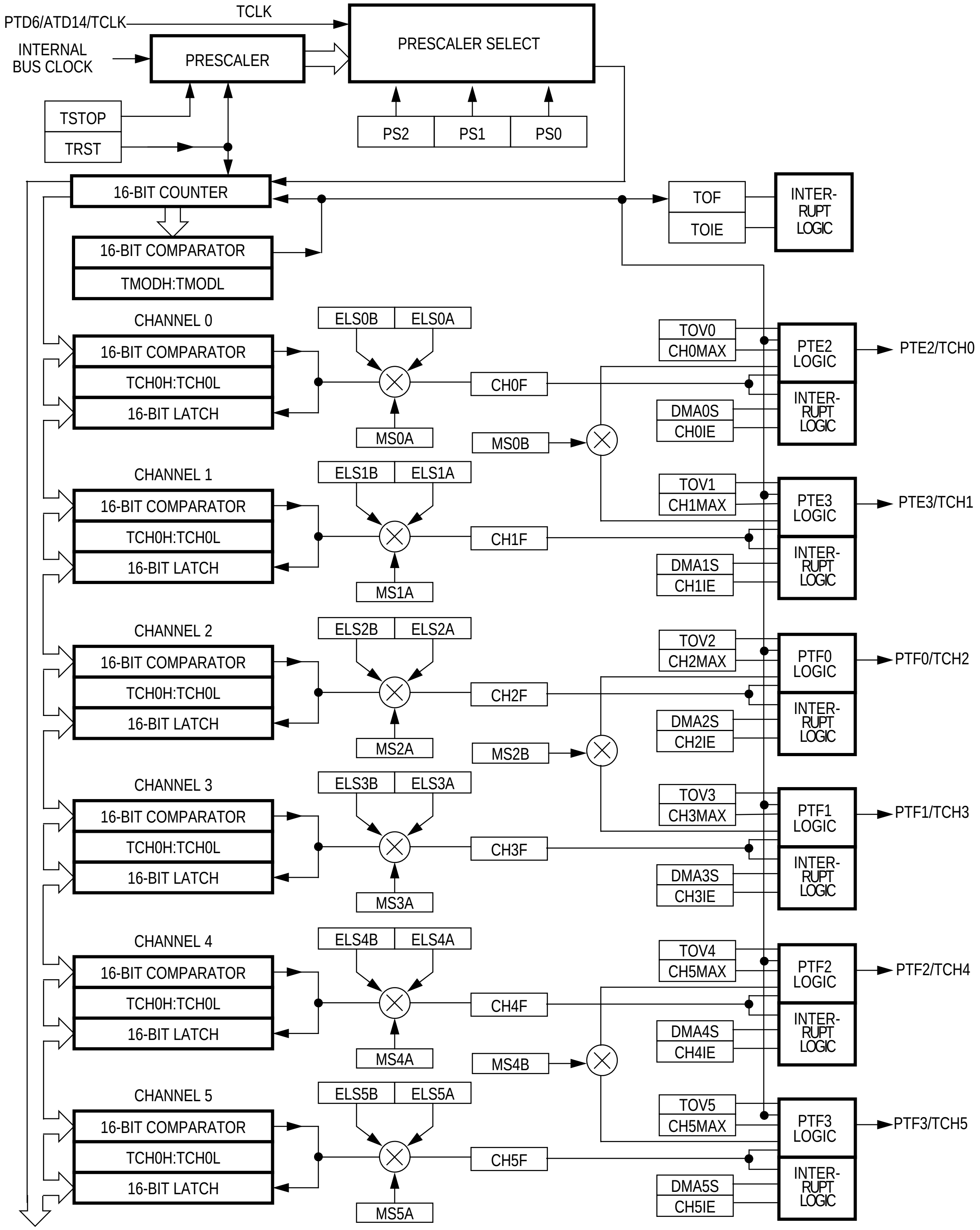


Figure 17-1. TIM Block Diagram

Table 17-1. TIM I/O Register Summary

	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
TIM Status/Control Register (TSC)	TOF	TOIE	TSTOP	TRST	0	PS2	PS1	PS0	$0020
TIM DMA Select Register (TDMA)	0	0	DMA5S	DMA4S	DMA3S	DMA2S	DMA1S	DMA0S	$0021
TIM Counter Register High (TCNTH)	Bit 15	14	13	12	11	10	9	Bit 8	$0022
TIM Counter Register Low (TCNTL)	Bit 7	6	5	4	3	2	1	Bit 0	$0023
TIM Counter Modulo Reg. High (TMODH)	Bit 15	14	13	12	11	10	9	Bit 8	$0024
TIM Counter Modulo Reg. Low (TMODL)	Bit 7	6	5	4	3	2	1	Bit 0	$0025
TIM Ch. 0 Status/Control Register (TSC0)	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX	$0026
TIM Ch. 0 Register High (TCH0H)	Bit 15	14	13	12	11	10	9	Bit 8	$0027
TIM Ch. 0 Register Low (TCH0L)	Bit 7	6	5	4	3	2	1	Bit 0	$0028
TIM Ch. 1 Status/Control Register (TSC1)	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX	$0029
TIM Ch. 1 Register High (TCH1H)	Bit 15	14	13	12	11	10	9	Bit 8	$002A
TIM Ch. 1 Register Low (TCH1L)	Bit 7	6	5	4	3	2	1	Bit 0	$002B
TIM Ch. 2 Status/Control Register (TSC2)	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX	$002C
TIM Ch. 2 Register High (TCH2H)	Bit 15	14	13	12	11	10	9	Bit 8	$002D
TIM Ch. 2 Register Low (TCH2L)	Bit 7	6	5	4	3	2	1	Bit 0	$002E
TIM Ch. 3 Status/Control Register (TSC3)	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX	$002F
TIM Ch. 3 Register High (TCH3H)	Bit 15	14	13	12	11	10	9	Bit 8	$0030
TIM Ch. 3 Register Low (TCH3L)	Bit 7	6	5	4	3	2	1	Bit 0	$0031
TIM Ch. 4 Status/Control Register (TSC4)	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX	$0032
TIM Ch. 4 Register High (TCH4H)	Bit 15	14	13	12	11	10	9	Bit 8	$0033
TIM Ch. 4 Register Low (TCH4L)	Bit 7	6	5	4	3	2	1	Bit 0	$0034
TIM Ch. 5 Status/Control Register (TSC5)	CH5F	CH5IE	0	MS5A	ELS5B	ELS5A	TOV5	CH5MAX	$0035
TIM Ch. 5 Register High (TCH5H)	Bit 15	14	13	12	11	10	9	Bit 8	$0036
TIM Ch. 5 Register Low (TCH5L)	Bit 7	6	5	4	3	2	1	Bit 0	$0037

17.4.3 Output Compare

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

Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **17.4.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

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

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

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

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, PTE2/TCH0, 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. Writing to the active channel registers is the same as generating unbuffered output compares.*

17.4.4 Pulse Width Modulation (PWM)

By using the toggle-on-overflow feature with an output compare channel, the TIM can generate a PWM signal. The value in the 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 17-2** shows, the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM to clear the channel pin on output compare if the state of the PWM pulse is logic one. Program the TIM to set the pin if the state of the PWM pulse is logic zero.

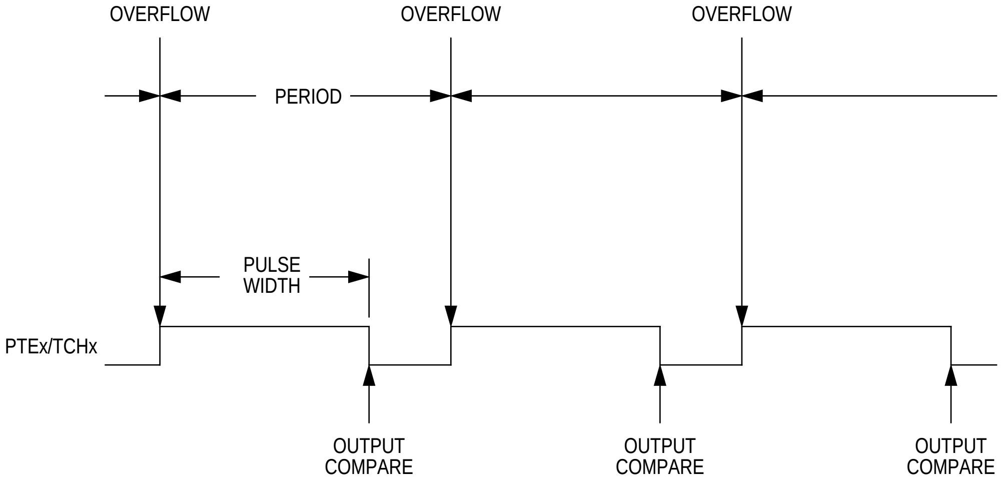


Figure 17-2. PWM Period and Pulse Width

The value in the TIM counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see **17.9.1 TIM Status and Control Register (TSC)**).

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

Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **17.4.4 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

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

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

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

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

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. Writing to the active channel registers is the same as generating unbuffered PWM signals.*

PWM Initialization

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

1. In the TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See **Table 17-3**.)
   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 17-3**.)

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

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

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

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

Setting MS4B links channels 4 and 5 and configures them for buffered PWM operation. The TIM channel 4 registers (TCH4H:TCH4L) initially control the PWM output. TIM status control register 4 (TSCR4) 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 clearing the TOVx bit generates a 100% duty cycle output. (See **17.9.5 TIM Channel Status and Control Registers (TSC0–TSC5)**.)

17.5 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter value rolls over to $0000 after matching the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH5F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests and TIM DMA service requests are controlled by the channel x interrupt enable bit, CHxIE, and the channel x DMA select bit, DMAxS. Channel x TIM CPU interrupt requests are enabled when CHxIE:DMAxS = 1:0. Channel x TIM DMA service requests are enabled when CHxIE:DMAxS = 1:1. CHxF and CHxIE are in the TIM channel x status and control register. DMAxS is in the TIM DMA select register.

17.6 Low-Power Modes

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

17.6.1 Wait Mode

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

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

The DMA can service the TIM without exiting wait mode.

17.6.2 Stop Mode

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

17.7 TIM During Break Interrupts

A break interrupt stops the TIM counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **7.8.3 SIM Break Flag Control Register (SBFCR)**.)

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

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

17.8 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.

17.8.1 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 logic ones to the three prescaler select bits, PS[2:0]. (See **17.9.1 TIM Status and Control Register (TSC)**.) The minimum TCLK pulse width, $\text{TCLK}_{\text{LMIN}}$ or $\text{TCLK}_{\text{HMIN}}$, is:

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

The maximum TCLK frequency is whichever is less of:

$$4\text{ MHz or bus frequency} \div 2$$

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.

17.8.2 TIM Channel I/O Pins (PTF3/TCH5–PTF0/TCH2, PTE3/TCH1–PTE2/TCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE2/TCH0, PTE6/TCH2, and

PTF2/TCH4 can be configured as buffered output compare or buffered PWM pins.

17.9 I/O Registers

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

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

17.9.1 TIM Status and Control Register (TSC)

The TIM status and control register does the following:

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

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

▨ = Unimplemented

Figure 17-3. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

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

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

TOIE — TIM Overflow Interrupt Enable Bit

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

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

TSTOP — TIM Stop Bit

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

1 = TIM counter stopped
0 = TIM counter active

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

TRST — TIM Reset Bit

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

1 = Prescaler and TIM counter cleared
0 = No effect

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD6/ATD14/TCLK pin or one of the seven prescaler outputs as the input to the TIM counter as **Table 17-2** shows. Reset clears the PS[2:0] bits.

Table 17-2. Prescaler Selection

PS[2:0]	TIM Clock Source
000	Internal Bus Clock ÷1
001	Internal Bus Clock ÷ 2
010	Internal Bus Clock ÷ 4
011	Internal Bus Clock ÷ 8
100	Internal Bus Clock ÷ 16
101	Internal Bus Clock ÷ 32
110	Internal Bus Clock ÷ 64
111	PTD6/ATD14/TCLK

17.9.2 TIM DMA Select Register (TDMA)

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

		Bit 7	6	5	4	3	2	1	Bit 0
TDMA $0021	Read:	0	0	DMA5S	DMA4S	DMA3S	DMA2S	DMA1S	DMA0S
	Write:	▨	▨						
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 17-4. TIM DMA Select Register (TDMA)

DMA5S — DMA Channel 5 Select Bit

This read/write bit enables TIM DMA service requests on channel 5. Reset clears the DMA5S bit.

1 = TIM DMA service requests enabled on channel 5
(TIM CPU interrupt requests disabled on channel 5)

0 = TIM DMA service requests disabled on channel 5
(TIM CPU interrupt requests enabled on channel 5)

DMA4S — DMA Channel 4 Select Bit

This read/write bit enables TIM DMA service requests on channel 4. Reset clears the DMA4S bit.

1 = TIM DMA service requests enabled on channel 4
(TIM CPU interrupt requests disabled on channel 4)

0 = TIM DMA service requests disabled on channel 4
(TIM CPU interrupt requests enabled on channel 4)

DMA3S — DMA Channel 3 Select Bit

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

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

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

DMA2S — DMA Channel 2 Select Bit

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

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

DMA1S — DMA Channel 1 Select Bit

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

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

DMA0S — DMA Channel 0 Select Bit

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

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

17.9.3 TIM Counter Registers (TCNTH:TCNTL)

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

NOTE: *If 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.*

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

▨ = Unimplemented

Figure 17-5. TIM Counter Registers (TCNTH:TCNTL)

17.9.4 TIM Counter Modulo Registers (TMODH:TMODL)

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

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

		Bit 7	6	5	4	3	2	1	Bit 0
TMODL $0025	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	1	1	1	1	1	1	1	1

Figure 17-6. TIM Counter Modulo Registers (TMODH:TMODL)

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

17.9.5 TIM Channel Status and Control Registers (TSC0–TSC5)

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

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

		Bit 7	6	5	4	3	2	1	Bit 0
TSC0 $0026	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0

		Bit 7	6	5	4	3	2	1	Bit 0
TSC1 $0029	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
	Write:	0		▨					
	Reset:	0	0	0	0	0	0	0	0

		Bit 7	6	5	4	3	2	1	Bit 0
TSC2 $002C	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 17-7. TIM Channel Status and Control Registers (TSC0–TSC3)

		Bit 7	6	5	4	3	2	1	Bit 0
TSC3 $002F	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
	Write:	0		▨					
	Reset:	0	0	0	0	0	0	0	0

		Bit 7	6	5	4	3	2	1	Bit 0
TSC4 $0032	Read:	CH4F	CH4IE	MS4B	MS4A	ELS4B	ELS4A	TOV4	CH4MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0

		Bit 7	6	5	4	3	2	1	Bit 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–TSC3) (Continued)

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

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

Reset clears the CHxF bit. Writing a logic one to CHxF has no effect.
1 = Input capture or output compare on channel x
0 = No input capture or output compare on channel x

CHxIE — Channel x Interrupt Enable Bit

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

NOTE: *TIM DMA service requests cannot be used in buffered PWM mode. In buffered PWM mode, disable TIM DMA service requests by clearing the DMAxS bit in the TIM DMA select register.*

Reset clears the CHxIE bit.

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

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

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIM channel 0, TIM channel 2, and TIM channel 4 status and control registers.

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

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

Setting MS4B disables the channel 5 status and control register and reverts TCH5 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 17-3**.)

1 = Unbuffered output compare/PWM operation

0 = Input capture operation

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. (See **Table 17-3**.). Reset clears the MSxA bit.

1 = Initial output level low
0 = Initial output level high

NOTE: *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIM status and control register (TSC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port E or port F, and pin PTEx/TCHx or PTFx/TCHx is available as a general-purpose I/O pin. **Table 17-3** shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 17-3. Mode, Edge, and Level Selection

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

NOTE: *Before enabling a TIM channel register for input capture operation, make sure that the PTEx/TCHx 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 logic zero, 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.

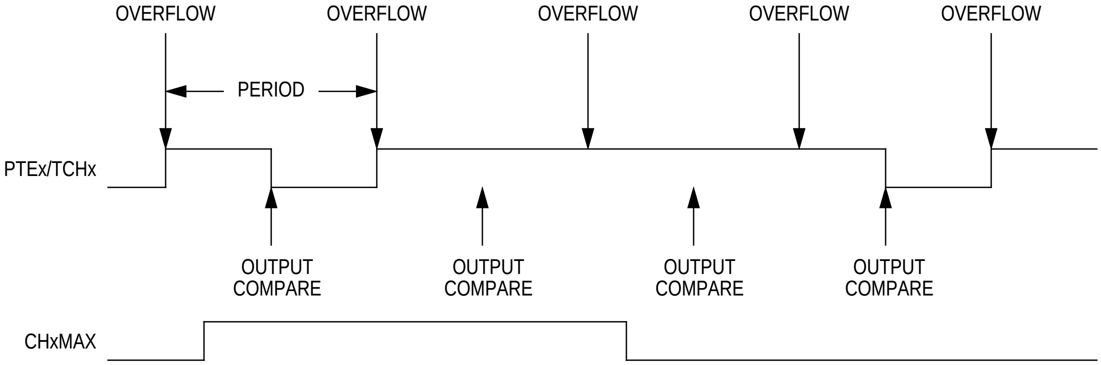


Figure 17-8. CHxMAX Latency

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

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

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

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

		Bit 7	6	5	4	3	2	1	Bit 0
TCH0H $0027	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH0L $0028	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH1H $002A	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH1L $002B	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH2H $002D	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							

Figure 17-9. TIM Channel Registers (TCH0H/L–TCH3H/L)

		Bit 7	6	5	4	3	2	1	Bit 0
TCH2L $002E	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH3H $0030	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH3L $0031	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH4H $0033	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH4L $0034	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH5H $0036	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after reset							

		Bit 7	6	5	4	3	2	1	Bit 0
TCH5L $0037	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after reset							

Figure 17-9. TIM Channel Registers (TCH0H/L–TCH3H/L) (Continued)

Section 18. Analog-to-Digital Converter (ADC)

18.1 Contents

18.2 Introduction

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

18.3 Features

Features of the ADC module include:

- 16 Channels with Multiplexed Input
- Linear Successive Approximation
- 8-bit Resolution
- Single or Continuous Conversion
- Conversion Complete Flag or Conversion Complete Interrupt
- Selectable ADC Clock

18.4 Functional Description

Sixteen ADC channels are available in the 64 QFP package for sampling external sources at pins PTD7/ATD15–PTD0/ATD8 and PTB7/ATD7–PTB0/ATD0. An analog multiplexer allows the single ADC converter to select one of 16 ADC channels as ADC voltage IN (ADCVIN). ADCVIN is converted by the successive approximation register based counter. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. (See **Figure 18-1**.)

NOTE: *DMA section and associated functions are only valid if the MCU has a DMA module.*

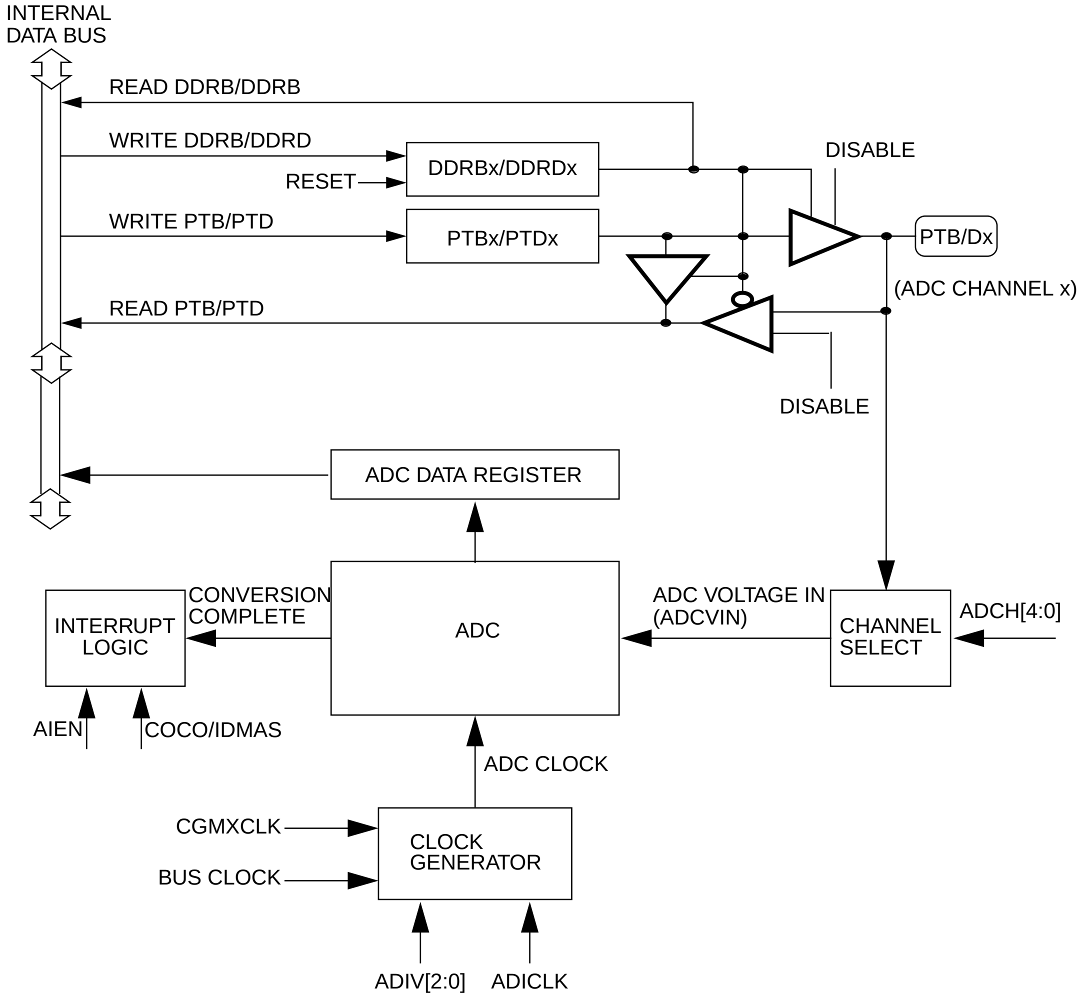


Figure 18-1. ADC Block Diagram

18.4.1 ADC Port I/O Pins

PTD7/ATD15-PTD0/ATD8 and PTB7/ATD7-PTB0/ATD0 are general-purpose I/O pins that share with the ADC channels. PTD2/ATD10 is an input-only pin that also shares with an ADC channel.

NOTE: *PTD7/ATD15 is available only on the 64-pin package.*

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 logic zero if the corresponding DDR bit is at logic zero. If the DDR bit is at logic one, the value in the port data latch is read.

NOTE: *Do not use ADC channel ATD14 when using the PTD6/ATD14/TCLK pin as the clock input for the TIM.*

18.4.2 Voltage Conversion

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

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

18.4.3 Conversion Time

Conversion starts after a write to the ADSCR 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 CGMXCLK frequency, bus frequency, and ADIV prescaler bits. For example, with an CGMXCLK frequency of 4 MHz, bus frequency of 8 MHz, and ADC clock frequency of 1 MHz, one conversion will take between 16 and 17 µsec and there will be 128 bus cycles between each conversion. Sample rate is approximately 60 kHz.

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

$$\text{\# Bus Cycles} = \text{Conversion Time x Bus Frequency}$$

18.4.4 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 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.

18.4.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes. See **21.7 ADC Converter Characteristics (see Note 1)** for accuracy information.

18.5 Interrupts

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

COCO/IDMAS bit is set, a DMA interrupt is generated. The COCO/IDMAS bit is not used as a conversion complete flag when interrupts are enabled.

18.6 Low-Power Modes

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

18.6.1 Wait Mode

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

18.6.2 Stop Mode

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

18.7 I/O Signals

The ADC module has 15 channels that are shared with I/O ports B and D and one channel with an input-only port bit on port D.

18.7.1 ADC Analog Power Pin (V_{DDAREF})

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.

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

18.7.2 ADC Analog Ground Pin (V_{SSA})

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

18.7.3 ADC Voltage Reference Pin (V_{REFH})

V_{REFH} is the high reference voltage for all analog-to-digital conversions. Connect the V_{REFH} pin to a voltage potential between 1.5 volts and V_{DDAREF}/V_{DDA} depending on the desired upper conversion boundary.

18.7.4 ADC Voltage Reference Low Pin (V_{REFL})

V_{REFL} is the lower reference supply for the ADC. Connect the V_{REFL} pin to a voltage potential between V_{SSA} and 0.5 volts depending on the desired lower conversion boundary.

18.7.5 ADC Voltage In (ADCVIN)

ADCVIN is the input voltage signal from one of the 16 ADC channels to the ADC module.

18.8 I/O Registers

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

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

18.8.1 ADC Status and Control Register (ADSCR)

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

		Bit 7	6	5	4	3	2	1	Bit 0
ADSCR $0038	Read: Write:	COCO/ IDMAS	AIEN	ADCO	CH4	CH3	CH2	CH1	CH0
	Reset:	0	0	0	1	1	1	1	1

Figure 18-2. ADC Status and Control Register

COCO/IDMAS — Conversions Complete/Interrupt DMA Select

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

1 = conversion completed (AIEN=0)
or
DMA interrupt enabled (AIEN=1)

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

AIEN — ADC Interrupt Enable

When this bit is set, an interrupt is generated at the end of an ADC conversion. CPU or DMA interrupt can be chosen with COCO/IDMAS bit. 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

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 16 ADC channels. The six channels are 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 18-1**.)

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

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

Table 18-1. Mux Channel Select

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select
0	0	0	0	0	PTB0/ATD0
0	0	0	0	1	PTB1/ATD1
0	0	0	1	0	PTB2/ATD2
0	0	0	1	1	PTB3/ATD3
0	0	1	0	0	PTB4/ATD4
0	0	1	0	1	PTB5/ATD5
0	0	1	1	0	PTB6/ATD6
0	0	1	1	1	PTB7/ATD7
0	1	0	0	0	PTD0/ATD8
0	1	0	0	1	PTD1/ATD9
0	1	0	1	0	PTD2/ATD10
0	1	0	1	1	PTD3/ATD11
0	1	1	0	0	PTD4/ATD12
0	1	1	0	1	PTD5/ATD13
0	1	1	1	0	PTD6/ATD14/TCLK
0	1	1	1	1	PTD7/ATD15
1	0	0	0	0	Unused (see Note 1)
↓	↓	↓	↓	↓	↓
1	1	0	1	0	Unused (see Note 1)
1	1	0	1	1	Reserved
1	1	1	0	0	V_{DDAREF} (see Note 2)
1	1	1	0	1	V_{REFH} (see Note 2)
1	1	1	1	0	V_{REFL} (see Note 2)
1	1	1	1	1	[ADC power off]

NOTES:

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.

18.8.2 ADC Data Register (ADR)

One 8-bit result register is provided. This register is updated each time an ADC conversion completes.

		Bit 7	6	5	4	3	2	1	Bit 0
ADR $0039	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	Indeterminate after reset							

▨ = Unimplemented

Figure 18-3. ADC Data Register

18.8.3 ADC Input Clock Register (ADICLK)

This register selects the clock frequency for the ADC.

		Bit 7	6	5	4	3	2	1	Bit 0
ADICLK $003A	Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
	Write:					▨	▨	▨	▨
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 18-4. ADC Input Clock Register

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 18-2 shows the available clock configurations. The ADC clock should be set to approximately 1 MHz.

Table 18-2. 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

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

If the external clock (cgmxclk) is equal or greater than 1 MHz, cgmxclk can be used as the clock source for the ADC. If cgmxclk is less than 1 MHz, use the PLL generated bus clock as the clock source. As long as the internal ADC clock is at approximately 1 MHz, correct operation can be guaranteed. (See **21.7 ADC Converter Characteristics (see Note 1)**.)

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.*

Section 19. Input/Output (I/O) Ports

19.1 Contents

19.2 Introduction

Forty-five bidirectional input-output (I/O) pins (or 39 in 52 PLCC package) and one input-only pin form seven parallel ports (or six in 52 PLCC package). All I/O pins are programmable as inputs or outputs.

NOTE: *Connect any unused I/O pins to an appropriate logic level, either V_{DD} or V_{SS}. Although the I/O ports do not require termination for proper operation, termination reduces excess current consumption and the possibility of electrostatic damage.*

Table 19-1. I/O Port Register Summary

Register Name	Bit 7	6	5	4	3	2	1	Bit 0	Addr.
Port A Data Register (PTA)	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0	$0000
Port B Data Register (PTB)	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0	$0001
Port C Data Register (PTC)	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0	$0002
Port D Data Register (PTD)	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0	$0003
Data Direction Register A (DDRA)	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0	$0004
Data Direction Register B (DDRB)	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0	$0005
Data Direction Register C (DDRC)	MCLKEN	0	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0	$0006
Data Direction Register D (DDRD)	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	0	DDRD1	DDRD0	$0007
Port E Data Register (PTE)	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	$0008
Port F Data Register (PTF)	0	0	0	PTF4	PTF3	PTF2	PTF1	PTF0	$0009
Port G Data Register (PTG)	0	0	0	0	0	PTG2	PTG1	PTG0	$000A
Data Direction Register E (DDRE)	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0	$000C
Data Direction Register F (DDRF)	0	0	0	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0	$000D
Data Direction Register G (DDRG)	0	0	0	0	0	DDRG2	DDRG1	DDRG0	$000E

19.3 Port A

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

19.3.1 Port A Data Register (PTA)

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

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

Figure 19-1. 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.

19.3.2 Data Direction Register A (DDRA)

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

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

Figure 19-2. 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 zero to one.*

Figure 19-3 shows the port A I/O logic.

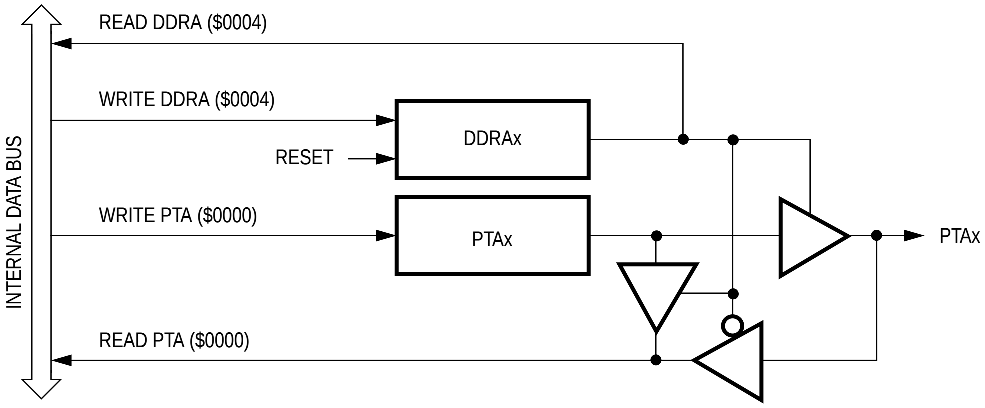


Figure 19-3. Port A I/O Circuit

When bit DDRAx is a logic one, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic zero, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 19-2** summarizes the operation of the port A pins.

Table 19-2. Port A Pin Functions

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

X = don’t care
Hi-Z = high impedance

[1] Writing affects data register, but does not affect input.

19.4 Port B

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

19.4.1 Port B Data Register (PTB)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTB $0001	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
	Reset:	Unaffected by reset							
	Alternate Functions:	ATD7	ATD6	ATD5	ATD4	ATD3	ATD2	ATD1	ATD0

Figure 19-4. 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 16 analog-to-digital converter 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 zero if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See **Section 18. Analog-to-Digital Converter (ADC)**.)

NOTE: *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 the states of the latches or logic zero.*

19.4.2 Data Direction Register B (DDRB)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRB $0005	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
	Reset:	0	0	0	0	0	0	0	0

Figure 19-5. 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 zero to one.*

Figure 19-6 shows the port B I/O logic.

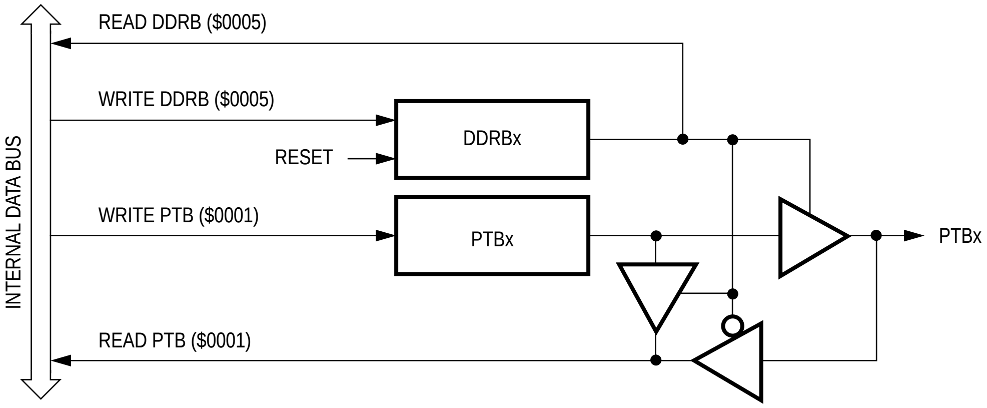


Figure 19-6. Port B I/O Circuit

When bit DDRBx is a logic one, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic zero, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 19-3** summarizes the operation of the port B pins.

Table 19-3. Port B Pin Functions

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

X = don’t care
Hi-Z = high impedance

[1] Writing affects data register, but does not affect input.

19.5 Port C

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

19.5.1 Port C Data Register (PTC)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTC $0002	Read:	0	0	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
	Write:	▨	▨						
	Reset:	Unaffected by reset							
		▨ = Unimplemented							
	Alternate Functions:						MCLK		

Figure 19-7. Port C Data Register (PTC)

PTC[5:0] — Port C Data Bits

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

NOTE: *PTC5 is available only on the 64-pin QFP.*

MCLK — T12 System Clock

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

19.5.2 Data Direction Register C (DDRC)

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

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

▨ = Unimplemented

Figure 19-8. Data Direction Register C (DDRC)

MCLKEN — MCLK Enable Bit

This read/write bit enables mclk to be an output signal on PTC2. If MCLK is enabled, PTC2 is under the control of MCLKEN. Reset clears this bit.

1 = MCLK output enabled
0 = MCLK output disabled

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

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

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

NOTE: *Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from zero to one.*

Figure 19-9 shows the port C I/O logic.

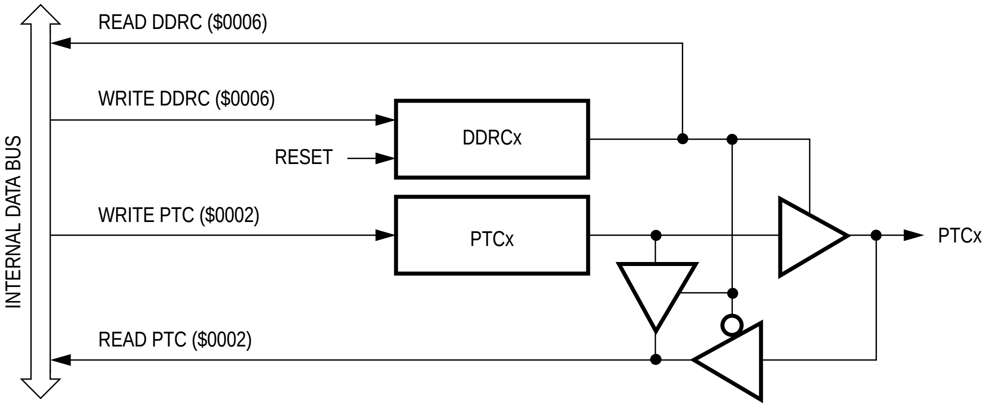


Figure 19-9. Port C I/O Circuit

When bit DDRCx is a logic one, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic zero, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 19-4** summarizes the operation of the port C pins.

Table 19-4. 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[7]	Pin	PTC2
1	2	Output	DDRC[7]	0	—
0	X	Input, Hi-Z	DDRC[5:0]	Pin	PTC[5:0][(1)]
1	X	Output	DDRC[5:0]	PTC[5:0]	PTC[5:0]

X = don’t care
Hi-Z = high impedance

1. Writing affects data register, but does not affect input.

19.6 Port D

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

19.6.1 Port D Data Register (PTD)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTD $0003	Read:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
	Write:						▨		
	Reset:	Unaffected by reset							
	Alternate Functions:	ATD15	ATD14 /TCLK	ATD13	ATD12	ATD11	ATD10	ATD9	ATD8

▨ = Unimplemented

Figure 19-10. Port D Data Register (PTD)

PTD[7:0] — Port D Data Bits

PTD[7:3,1:0] are read/write, software programmable bits. Data direction of PTD[7:3,1:0] pins are under the control of the corresponding bit in data direction register D. PTD2/ATD10 is an input-only pin.

ATD[15:8] — ADC Channels

PTD7/ATD15- PTD0/ATD8 are eight of the 16 analog-to-digital converter channels. The ADC channel select bits, CH[4:0], determine whether the PTD7/ATD15- 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 zero if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See **Section 18. Analog-to-Digital Converter (ADC)**.)

NOTE: *PTD7/ATD15 is available only on the 64-pin QFP.*

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

TCLK — Timer Clock Input

The PTD6/ATD14/TCLK pin is the external clock input for the TIM. The prescaler select bits, PS[2:0], select PTD6/ATD14/TCLK as the TIM clock input. (See **17.9.1 TIM Status and Control Register (TSC)**.) When not selected as the TIM clock, PTD6/ATD14/TCLK is available for general-purpose I/O or as an ADC channel.

NOTE: *Do not use ADC channel ATD14 when using PTD6/ATD14/TCLK pin as the clock input for the TIM.*

19.6.2 Data Direction Register D (DDRD)

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

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

▨ = Unimplemented

Figure 19-11. Data Direction Register D (DDRD)

DDRD[7:3,1:0] — Data Direction Register D Bits

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

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

NOTE: *Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from zero to one.*

Figure 19-12 shows the port D I/O logic.

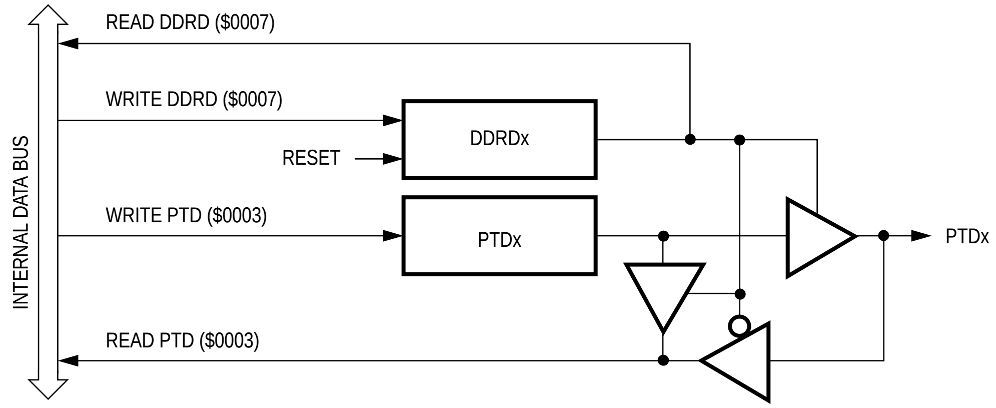


Figure 19-12. Port D I/O Circuit

When bit DDRDx is a logic one, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic zero, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 19-5** summarizes the operation of the port D pins.

Table 19-5. Port D Pin Functions

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

X = don’t care
Hi-Z = high impedance

[1] Writing affects data register, but does not affect input.

19.7 Port E

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

19.7.1 Port E Data Register (PTE)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTE $0008	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
	Reset:	Unaffected by reset							
	Alternate Function:	SPSCK	MOSI	MISO	$\overline{\text{SS}}$	TCH1	TCH0	RxD	TxD

Figure 19-13. 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

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.

MOSI — Master Out/Slave In

The PTE6/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PTE6/MOSI pin is available for general-purpose I/O. (See **16.14.1 SPI Control Register (SPCR)**.)

MISO — Master In/Slave Out

The PTE5/MISO pin is the master in/slave out terminal of the SPI module. When the SPI enable bit, SPE, is clear, the SPI module is disabled, and the PTE5/MISO pin is available for general-purpose I/O. (See **16.14.1 SPI Control Register (SPCR)**.)

$\overline{\text{SS}}$ — Slave Select

The PTE4/$\overline{\text{SS}}$ pin is the slave select input of the SPI module. When the SPE bit is clear, or when the SPI master bit, SPMSTR, is set and MODFEN bit is low, the PTE4/$\overline{\text{SS}}$ pin is available for general-purpose I/O. (See **16.13.4 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/$\overline{\text{SS}}$ pin.

NOTE: *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SPI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See **Table 19-6**.)*

TCH[1:0] — Timer Channel I/O Bits

The PTE3/TCH1–PTE2/TCH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE3/TCH1–PTE2/TCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See **17.9.5 TIM Channel Status and Control Registers (TSC0–TSC5)**.)

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 19-6**.)*

RxD — SCI Receive Data Input

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. (See **15.8.1 SCI Control Register 1 (SCC1)**.)

TxD — SCI Transmit Data Output

The PTE0/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. (See **15.8.1 SCI Control Register 1 (SCC1)**.)

NOTE: *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SCI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See* ***Table 19-6****.)*

19.7.2 Data Direction Register E (DDRE)

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

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

Figure 19-14. 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 zero to one.*

Figure 19-15 shows the port E I/O logic.

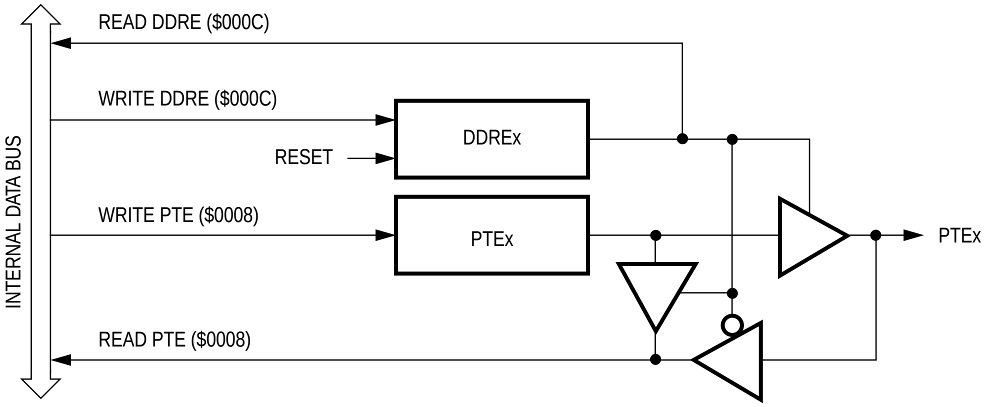


Figure 19-15. Port E I/O Circuit

When bit DDREx is a logic one, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic zero, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 19-6** summarizes the operation of the port E pins.

Table 19-6. Port E Pin Functions

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

X = don’t care
Hi-Z = high impedance

[1] Writing affects data register, but does not affect input.

19.8 Port F

Port F is a 5-bit special function port that shares four of its pins with the timer interface module (TIM).

19.8.1 Port F Data Register (PTF)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTF $0009	Read:	0	0	0	PTF4	PTF3	PTF2	PTF1	PTF0
	Write:	▨	▨	▨					
	Reset:	Unaffected by reset							
	Alternate Function:					TCH5	TCH4	TCH3	TCH2

▨ = Unimplemented

Figure 19-16. Port F Data Register (PTF)

PTF[4:0] — 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[4:0].

TCH[5:2] — Timer Channel I/O Bits

The PTF3/TCH5–PTF0/TCH2 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF3/TCH5–PTF0/TCH2 pins are timer channel I/O pins or general-purpose I/O pins. (See **17.9.5 TIM Channel Status and Control Registers (TSC0–TSC5)**.)

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 19-7**.)*

19.8.2 Data Direction Register F (DDRF)

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

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

▨ = Unimplemented

Figure 19-17. Data Direction Register F (DDRF)

DDRF[4:0] — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF[4:0], 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 zero to one.*

Figure 19-18 shows the port F I/O logic.

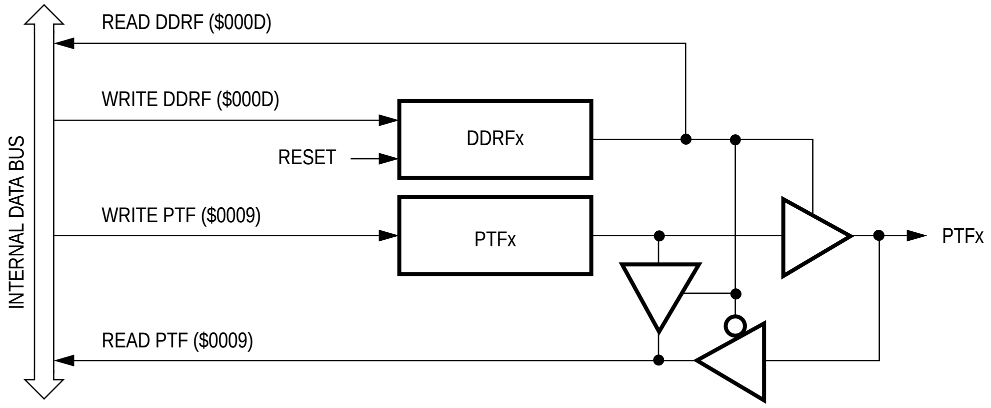


Figure 19-18. Port F I/O Circuit

When bit DDRFx is a logic one, reading address $0009 reads the PTFx data latch. When bit DDRFx is a logic zero, reading address $0009 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 19-7** summarizes the operation of the port F pins.

Table 19-7. Port F Pin Functions

DDRF Bit	PTF Bit	I/O Pin Mode	Accesses to DDRF	Accesses to PTF	
			Read/Write	Read	Write
0	X	Input, Hi-Z	DDRF[4:0]	Pin	PTF[4:0][1]
1	X	Output	DDRF[4:0]	PTF[4:0]	PTF[4:0]

X = don’t care
Hi-Z = high impedance

[1] Writing affects data register, but does not affect input.

NOTE: *Bit PTF4 is available only on the 64-pin QFP.*

19.9 Port G

Port G is a 3-bit general-purpose bidirectional I/O port.

NOTE: *Port G is available only on the 64-pin QFP.*

19.9.1 Port G Data Register (PTG)

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

		Bit 7	6	5	4	3	2	1	Bit 0
PTG $000A	Read:	0	0	0	0	0	PTG2	PTG1	PTG0
	Write:	▨	▨	▨	▨	▨			
	Reset:	Unaffected by reset							

▨ = Unimplemented

Figure 19-19. Port G Data Register (PTG)

PTG[2:0] — Port G Data Bits

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

19.9.2 Data Direction Register G (DDRG)

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

		Bit 7	6	5	4	3	2	1	Bit 0
DDRG $000E	Read:	0	0	0	0	0	DDRG2	DDRG1	DDRG0
	Write:	▨	▨	▨	▨	▨			
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 19-20. Data Direction Register G (DDRG)

DDRG[2:0] — Data Direction Register G Bits

These read/write bits control port G data direction. Reset clears DDRG[2:0], configuring all port G pins as inputs.

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

NOTE: *Avoid glitches on port G pins by writing to the port G data register before changing data direction register G bits from zero to one.*

Figure 19-21 shows the port G I/O logic.

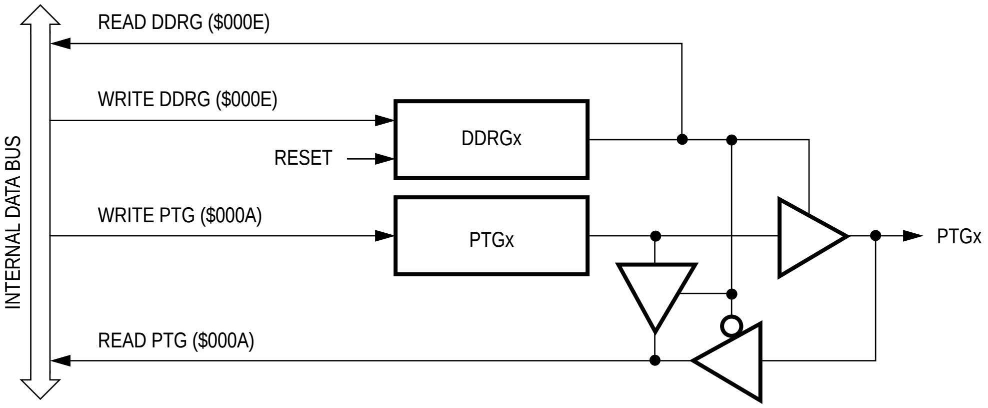


Figure 19-21. Port G I/O Circuit

When bit DDRGx is a logic one, reading address $000A reads the PTGx data latch. When bit DDRGx is a logic zero, reading address $000A reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data.

Section 20. Byte Data Link Controller-Digital (BDLC-D)

20.1 Contents

20.2 Introduction

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

20.3 Features

Features of the BDLC module include:

- SAE J1850 Compatible
- 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
- Receive and Transmit Block Mode 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

20.4 Functional Description

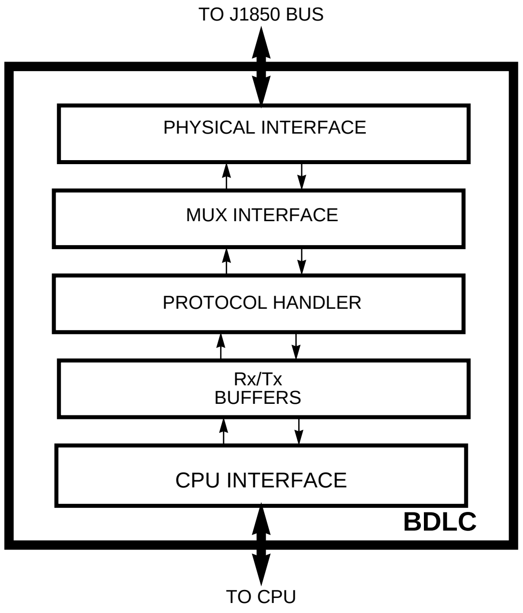


Figure 20-1. BDLC Block Diagram

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.

20.4.1 BDLC Operating Modes

The BDLC has five main modes of operation which interact with the power supplies, pins, and rest of the MCU as shown below

.

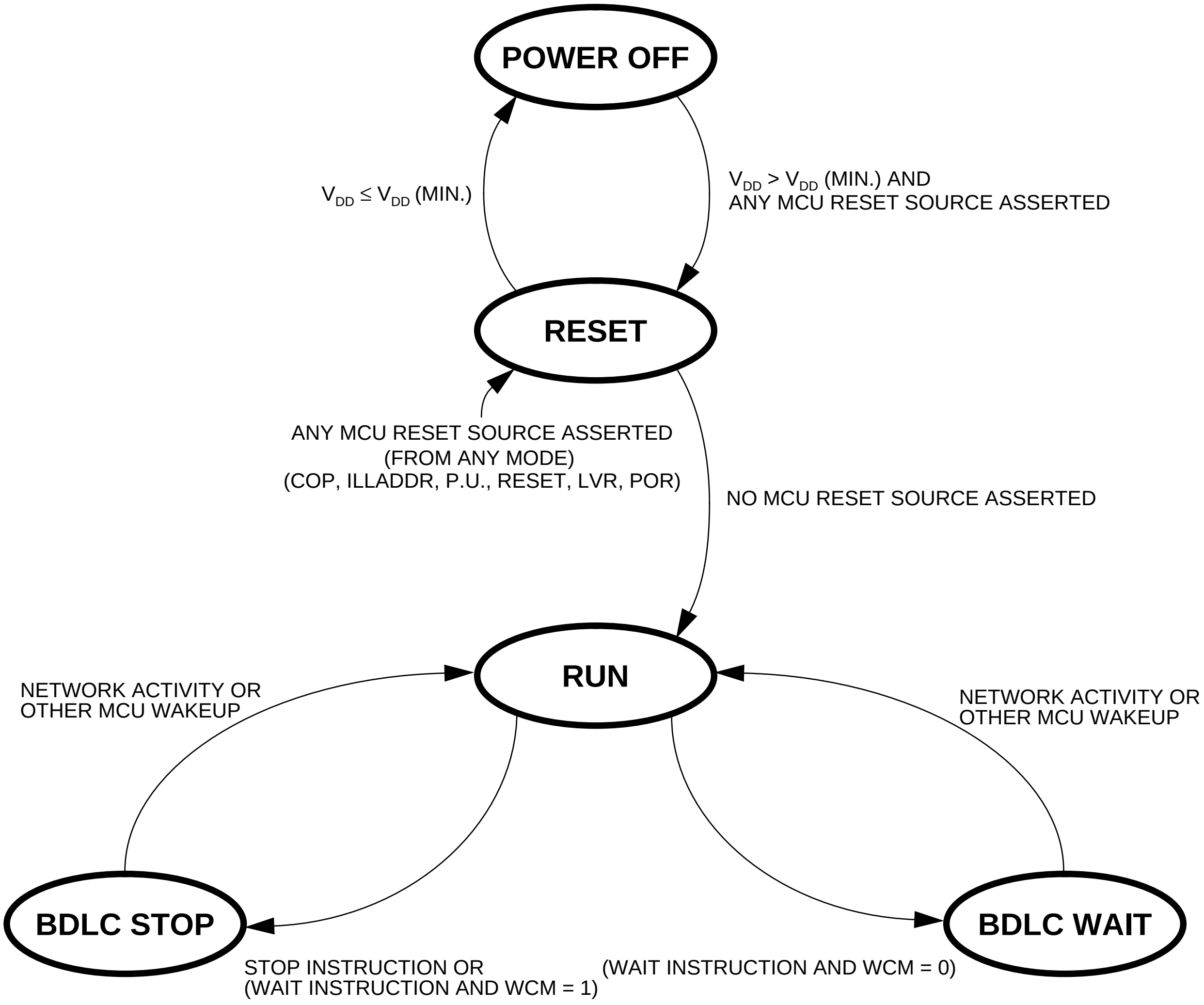

POWER OFF
$V_{DD} \leq V_{DD}$ (MIN.)
$V_{DD} > V_{DD}$ (MIN.) AND
ANY MCU RESET SOURCE ASSERTED
RESET
ANY MCU RESET SOURCE ASSERTED
(FROM ANY MODE)
(COP, ILLADDR, P.U., RESET, LVR, POR)
NO MCU RESET SOURCE ASSERTED
RUN
NETWORK ACTIVITY OR
OTHER MCU WAKEUP
NETWORK ACTIVITY OR
OTHER MCU WAKEUP
BDLC STOP
BDLC WAIT
STOP INSTRUCTION OR
(WAIT INSTRUCTION AND WCM = 1)
(WAIT INSTRUCTION AND WCM = 0)

Figure 20-2. BDLC Operating Modes State Diagram

Power Off

This mode is entered from the 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 the reset mode by low voltage reset (LVR) before being powered down. In this mode, the pin input and output specifications are not guaranteed.

Reset

This mode is entered from the power off mode whenever the BDLC supply voltage V_{DD} rises above its minimum specified value ($V_{DD(MIN)}$) 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, reset pin, etc., is asserted.

In this 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, and inputs and network activity are ignored.

Run

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 also in the run mode.

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

BDLC Wait

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

In this mode, the BDLC internal clocks continue to run but the physical interface circuitry is placed in a low power mode and wait for any activity on the bus. 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 will also generate a CPU interrupt request which wakes up the BDLC and the CPU. See **20.8.2 Wait Mode**.

BDLC Stop

This power-conserving mode is automatically entered from the run mode whenever the CPU executes a STOP instruction, or if the CPU executes a WAIT instruction and the WCM bit in the BCR 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 **20.8.1 Stop Mode**.

Digital Loopback

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 (CL2TxD) and the receive digital input pin (CL2RxD) 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.

Analog Loopback

When a bus fault has been detected, the analog 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 analog physical interface’s output drive stage for the node’s J1850 bus pin. In this mode, the input to the output drive stage is looped back into the receiver so that the BDLC is able to transmit and receive its own messages without driving the J1850 bus.

20.5 BDLC CPU Interface

The CPU interface provides the interface between the CPU and the BDLC and consists of four user registers. A full description of each register follows.

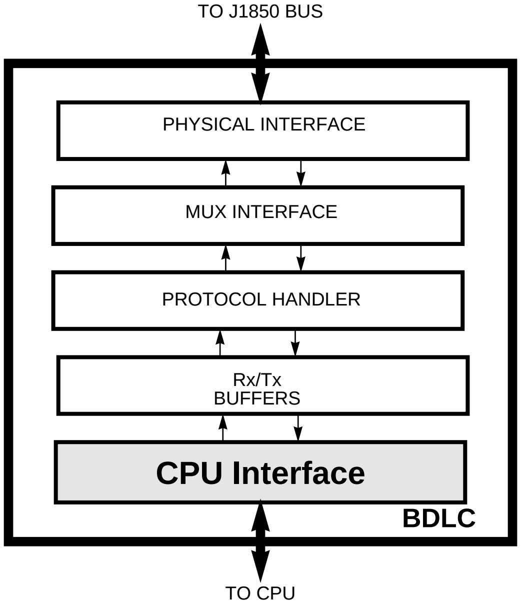


Figure 20-3. BDLC Block Diagram

Table 20-1. BDLC User Registers

Register Name	Address
BARD	$003B
BCR1	$003C
BCR2	$003D
BSVR	$003E
BDR	$003F

20.5.1 BDLC Analog and Roundtrip Delay (BARD)

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 1µs are available. The BARD register can be written once only after each reset. The register may be read at any time.

		Bit 7	6	5	4	3	2	1	Bit 0
BARD $003 B	Read:	ATE	RXPOL	0	0	BO3	BO2	BO1	BO0
	Write:			▨	▨				
	Reset:	1	1	0	0	0	1	1	1

▨ = Unimplemented

Figure 20-4. BDLC Analog and Roundtrip Delay Register

ATE — Analog Transceiver Enable

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

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

NOTE: *An on-board analog transceiver must exist for this bit to be meaningful; otherwise, the value of this bit doesn't matter.*

RXPOL — Receive Pin Polarity

The receive pin polarity (RXPOL) bit is used to select the polarity of an incoming signal on the receive pin. Some external analog transceiver inverts 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.

BO3-BO0 — BARD Offset bits

The following table shows the expected transceiver delay with respect to BARD offset values:

Table 20-2. BDLC Transceiver Delay

BARD Offset Bits (BO[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

20.5.2 BDLC Control Register 1 (BCR1)

This register is used to configure and control the BDLC.

		Bit 7	6	5	4	3	2	1	Bit 0
BCR1 $003C	Read:	IMSG	CLKS	R1	R0	0	0	IE	WCM
	Write:					▨	▨		
	Reset:	1	1	1	0	0	0	0	0

▨ = Unimplemented

Figure 20-5. BDLC Control Register 1

IMS — Ignore Message

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 Select

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 automatically adjust symbol timing.

1 = Binary frequency (1.048576 MHz) is selected for f_{bdlc}.

0 = Integer frequency (1 MHz) is selected for f_{bdlc}.

R1, R0 — Rate Select

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 1.00MHz for J1850 bus communications to take place. Hence, the value programmed into these bits is dependent on the chosen MCU system clock frequency per the following table.

Table 20-3. BDLC Rate Selection

f_{XCLK} Frequency	R1	R0	Division	f_{bdlc}
1.048576 MHz	0	0	1	1.048576 MHz
2.09715 MHz	0	1	2	1.048576 MHz
4.19430 MHz	1	0	4	1.048576 MHz
8.38861 MHz	1	1	8	1.048576 MHz
1.00000 MHz	0	0	1	1.000000 MHz
2.00000 MHz	0	1	2	1.000000 MHz
4.00000 MHz	1	0	4	1.000000 MHz
8.00000 MHz	1	1	8	1.000000 MHz

A register option is provided to allow the user to select between a BDLC operating frequency (f_{bdlc}) of 1.048576 MHz or 1.0 MHz.

IE — Interrupt Enable

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 **20.5.4 BDLC State Vector Register (BSVR)** for a description of the BSVR register.

WCM — Wait Clock Mode

This bit determines the operation of the BDLC during CPU wait mode. See **20.8.1 Stop Mode** and **20.8.2 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

20.5.3 BDLC Control Register 2 (BCR2)

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.

		Bit 7	6	5	4	3	2	1	Bit 0
BCR2 $003D	Read: Write:	ALOOP	DLOOP	RX4XE	NBFS	TEOD	TSIFR	TMIFR1	TMIFR0
	Reset:	1	1	0	0	0	0	0	0

Figure 20-6. BDLC Control Register 2

ALOOP — Analog Loopback Mode

This bit determines whether the J1850 bus will be driven by the analog physical interface's final drive stage. The bus may be used to reset the BDLC state machine after enabling the off-chip analog transceiver in loop-back mode.

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 (t_{trv4}) time before message reception or a minimum of inter-frame symbol (t_{trv6}) time before message transmission.

NOTE: *An on-board analog transceiver must exist for this bit to be meaningful; otherwise, the value of this bit doesn't matter.*

DLOOP — Digital Loopback Mode

This bit determines the source to which the digital receive input (CL2RxD) is connected and can be used to isolate bus fault conditions (see **Figure 20-11**). If a fault condition has been detected on the bus, this control bit allows the programmer to disconnect the BDLC digital block from the physical interface block and connect the digital transmit output to the digital receive input. In this configuration, data sent from the transmit buffer should be reflected back into the receive buffer. If no faults exist in the digital block, the fault is in the physical interface block or elsewhere on the J1850 bus.

1 = When set, RxD is connected to CL2TxD. The BDLC is now in digital loopback mode.

0 = When cleared, RxD is connected to CL2RxD. The BDLC is taken out of Digital Loopback Mode and can now drive the J1850 bus normally.

RX4XE — Receive 4X Enable

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

This bit controls the format of the normalization bit (NB). SAE J1850 strongly encourages using an active long, zero, for in-frame responses containing CRC and active short, 1, for In-Frame Responses without CRC.

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

TEOD — Transmit End of Data

This bit is set by the programmer to indicate the end of a message being sent by the BDLC. It will append an 8-bit CRC after completing transmission of the current byte. This bit also is also used to end an 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 has been transmitted (See **20.6.3 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.

1 = Transmit 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

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 one at any given time. However, if more than one of these three control bits are set to one, the priority encoding logic will force these register bits to a known value as shown in the following table. For example, if 011 is written to TSIFR, TMIFR1, and TMIFR0, then internally, they'll be encoded as 010. However, when these bits are read back, they will read 011.

Table 20-4. BDLC Transmit In-Frame Response Control Bit Priority Encoding

Write/Read TSIFR1	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. The three types of J1850 IFR are shown below.

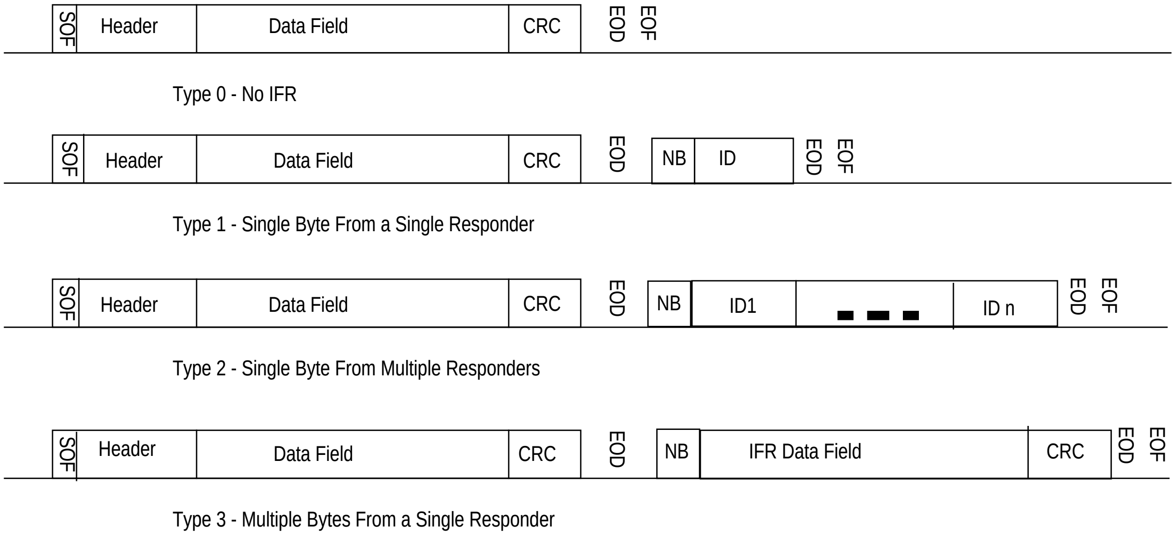

SOF
Header
Data Field
CRC
EOD
EOF
Type 0 - No IFR
NB
ID
Type 1 - Single Byte From a Single Responder
ID1
ID n
Type 2 - Single Byte From Multiple Responders
IFR Data Field
Type 3 - Multiple Bytes From a Single Responder

Figure 20-7. Types of In-Frame Response

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

This bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR) as a single byte IFR with no CRC.

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 by the CPU, or an error is detected on the bus.

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

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.

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

This 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.

If this bit is set prior to a valid EOD being received with no CRC error and once the EOD symbol has been received, the BDLC will attempt to transmit the appropriate normalization/format 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 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 BCR register. 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.

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 following the TDRE interrupt.

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

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 TIFR1 bit. Once the EOD interrupt occurs, the programmer should then set the TEOD bit in the BCR. 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 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 one bits will be sent out.

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.

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 and the last IFR byte has been transmitted, the CRC byte is transmitted.

0 = 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 following the TDRE interrupt.

TMIFR0 — Transmit Multiple Byte IFR without CRC (Type 2)

This 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.

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/format 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 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 BCR register. This will instruct the BDLC to transmit an EOD symbol, indicating the end of the IFR portion of the message frame. The BDLC will not append a CRC.

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 following the TDRE interrupt.

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 one bits (active short bits) will be sent out.

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.

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 following the TDRE interrupt.

20.5.4 BDLC State Vector Register (BSVR)

This register is provided to decrease substantially the CPU overhead associated with servicing interrupts while under operation of a MUX 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.

		Bit 7	6	5	4	3	2	1	Bit 0
BSVR $003E	Read:	0	0	I3	I2	I1	I0	0	0
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 20-8. BDLC State Vector Register

I0, I1, I2, I3 — Interrupt Source

These bits indicate the source of the interrupt request that currently is pending. The encoding of these bits follows:

Table 20-5. 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	Rx Data Register Full (RDRF)	3
$10	0	1	0	0	Tx Data Register Empty (TDRE)	4
$14	0	1	0	1	Loss of Arbitration	5
$18	0	1	1	0	CRC Error	6
$1C	0	1	1	1	Symbol Invalid or Out of Range	7
$20	1	0	0	0	Wakeup	8 (Highest)

Bits I0, I1, I2, and I3 are cleared by a read of the BSVR register 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 register followed by a read of BDR. TDRE can either be cleared only by a read of the BSVR register followed by a write to the BDLC BDR register or by setting the TEOD bit in BCR2.

Upon receiving a BDLC interrupt, the user may read the value within the BSVR, transferring it to the CPU's index register. The value may 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 just used to align the JMPs onto 4-byte boundaries so that the value in the BSVR may 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 may be made smaller or omitted altogether.

20.5.5 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) interrupt has occurred, or the BSVR register has been polled indicating this condition.

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.

The BDR register 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 byte of data.

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 will automatically 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 1-bits) has been transmitted. The receiver will pick this up as an error and relay it in the state vector register as an invalid symbol error.

		Bit 7	6	5	4	3	2	1	Bit 0
BDR $003F	Read: Write:	D7	D6	D5	D4	D3	D2	D1	D0
	Reset:	Undeterminate After Reset							

Figure 20-9. BDLC Data Register

20.6 BDLC Protocol Handler

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

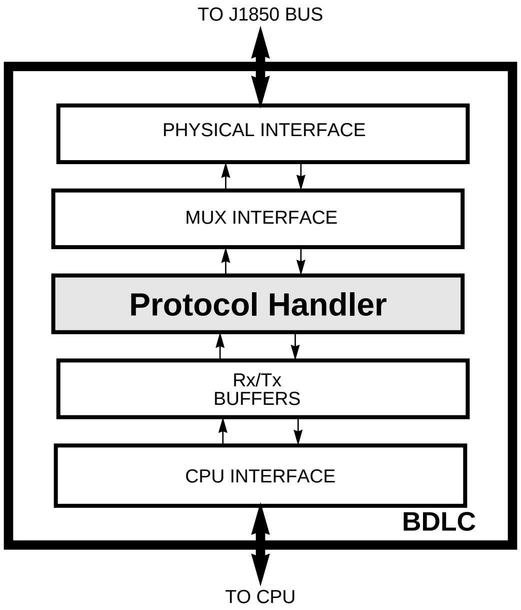


Figure 20-10. BDLC Block Diagram

20.6.1 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 below.

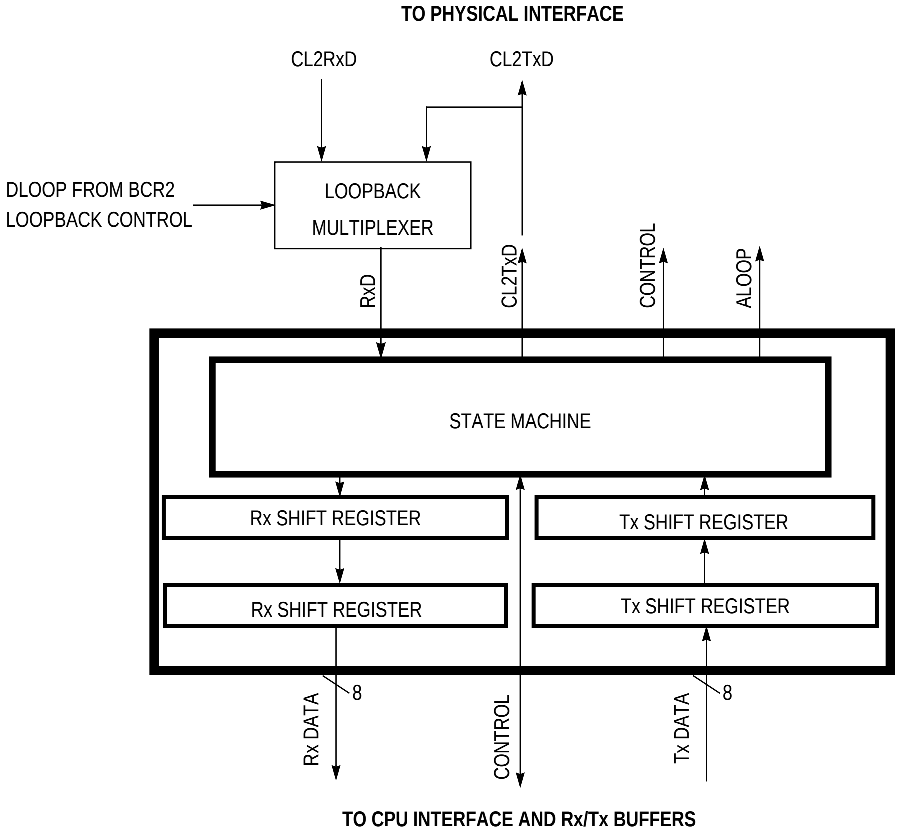


Figure 20-11. BDLC Protocol Handler Outline

20.6.2 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.

20.6.3 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 and 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.

20.6.4 Digital Loopback Multiplexer

The digital loopback multiplexer connects RxD to either CL2TxD or CL2RxD, depending on the state of the DLBE bit in the BCR2 register (See **20.5.3 BDLC Control Register 2 (BCR2)**).

20.6.5 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 sections describe the BDLC’s actions in a variety of situations.

4X Mode

The BDLC can exist on the same J1850 bus as modules which use a special 4X (41.6 kbps) mode of J1850 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.

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.

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 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.

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.

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 is reflected in the BSVR register. If the interrupt enable bit (IE) is set, an interrupt request from the BDLC is generated.

CRC Error

A 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 should 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 is set when a CRC error is detected.

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 rather than a transmitter error. Symbol invalid or the out-of-range flag is set when a symbol error is detected.

Framing Error

A framing error is detected if an EOD or EOF symbol is detected on a non-byte boundary from the J1850 bus. Symbol invalid or the out-of-range flag is set when a framing error is detected.

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 V_{batt}, 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, since the short to ground would not allow the bus to be driven to the active (dominant) 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.

BREAK — Break

Any BDLC transmitting at the time a BREAK is detected will treat the BREAK as if a transmission error had occurred and will halt transmission.

If the BDLC detects a BREAK symbol while it is receiving a message, it will treat the BREAK as a reception error.

If a BREAK symbol is received while the BDLC is transmitting or receiving, an invalid symbol interrupt will be generated. Reading the BSVR register will clear this interrupt condition. The BDLC will wait for bus to idle, then wait for SOF.

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

Summary

Table 20-6. BDLC J1850 Bus Error Summary

Error Condition	BDLC Function
Bus short to V_{batt}	The BDLC will not transmit until the bus is idle.
Bus short to Gnd	Thermal overload will shutdown physical interface. Fault condition is reflected in BSVR as invalid symbol.
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 SOF.
CRC Error	CRC error interrupt will be generated. The BDLC will wait for SOF.
BDLC receives BREAK symbol	The BDLC will wait for the next valid SOF. Invalid symbol interrupt will be generated.
Invalid symbol: BDLC sends an EOD but receives an active symbol.	Invalid symbol interrupt will be generated. The BDLC will wait for SOF.

20.7 BDLC MUX Interface

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

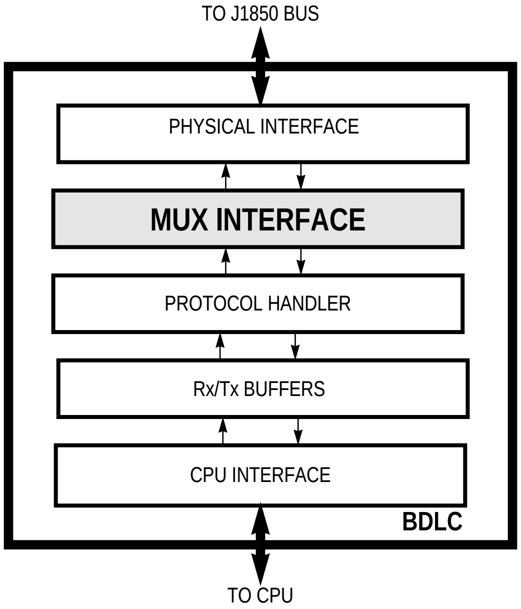


Figure 20-12. BDLC Block Diagram

20.7.1 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 20-13**.

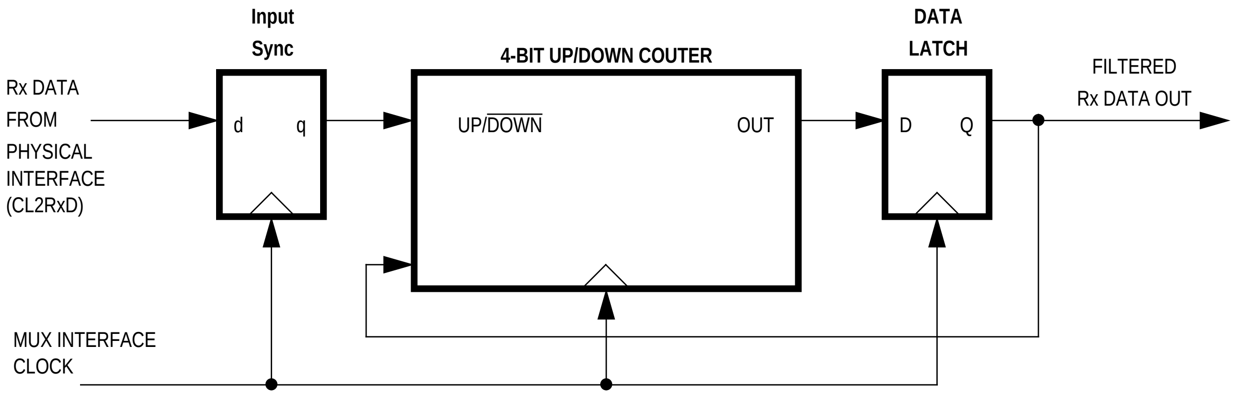


Figure 20-13. BDLC Rx Digital Filter Block Diagram

Operation

The clock for the digital filter is provided by the MUX interface clock. At each positive edge of the clock signal, the current state of the receiver physical interface (CL2RxD) signal is sampled. The CL2RxD 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. The counter will thus progress up toward 15 if, on average, the CL2RxD signal remains high or progress down toward 0 if, on average, the CL2RxD signal remains low.

When the counter eventually reaches the value 15, the digital filter decides that the condition of the CL2RxD signal is at a stable logic level one and the data latch is set, causing the filtered Rx data signal to become a logic level one. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

Alternatively, should the counter eventually reach the value zero, the digital filter decides that the condition of the CL2RxD signal is at a stable logic level zero and the data latch is reset, causing the filtered Rx data signal to become a logic level zero. 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 CL2RxD signal.

Performance

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

If the signal on the CL2RxD 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 CL2RxD signal depends on the characteristics of the noise itself. Narrow noise pulses on the CL2RxD 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 may 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.

20.7.2 J1850 Frame Format

All messages transmitted on the J1850 bus are structured using the format:

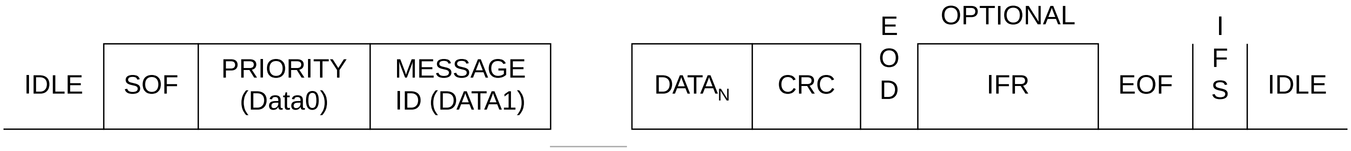


Figure 20-14. J1850 Bus Message Format (VPW)

J1850 states that each message has a maximum length of 101 bit times or 12 bytes (excluding SOF, EOD, NB, and EOF).

SOF — Start of Frame Symbol

All messages transmitted onto the J1850 bus must begin with an 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, 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 8 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 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 EOD symbols 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 register.

EOD — End of Data Symbol

The EOD symbol is a short 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.

EOF — End-of-Frame Symbol

This symbol is a passive period on the J1850 bus and is longer than an EOD symbol, which signifies the end of a message. Since an EOF symbol is longer than an 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 passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node following the completion of the EOF 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 an SOF, 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 (initiate transmission) occurred on or before $104\cdot t_{bdlc}$ 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 $104\cdot t_{bdlc}$ 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. Receivers must synchronize to any SOF occurring during an IFS period to allow for individual clock tolerances.

BREAK — Break

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. The BDLC cannot transmit a BREAK symbol. If while receiving a message the BDLC detects a BREAK symbol, it treats the BREAK as a reception error and sets the invalid symbol flag. 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. The RX4XE bit in BCR2 is cleared automatically upon reception of the BREAK symbol.

IDLE — Idle Bus

An idle condition exists on the bus during any passive period after expiration of the IFS period. Any node sensing an idle bus condition can begin transmission immediately.

20.7.3 J1850 VPW Symbols

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, active or passive. Active and passive bits are used alternately.

Each logic one or logic zero contains a single transition and can be at either the active or passive level and one of two lengths, either 64 ms or 128 ms (T_{NOM} at 10.4 kbps baud rate), depending upon the encoding of the previous bit. The SOF, EOD, EOF and IFS symbols will always be encoded at an assigned level and length. See **Figure 20-15**.

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 logic one or a logic zero.

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

Logic Zero

A logic zero is defined as either an active-to-passive transition followed by a passive period 64 ms in length, or a passive-to-active transition followed by an active period 128 ms in length (**Figure 20-15(a)**).

Logic One

A logic one is defined as either an active-to-passive transition followed by a passive period 128 ms in length or a passive-to-active transition followed by an active period 64 ms in length (**Figure 20-15(b)**).

NB — Normalization Bit

The NB symbol has the same property as a logic one or a logic zero.

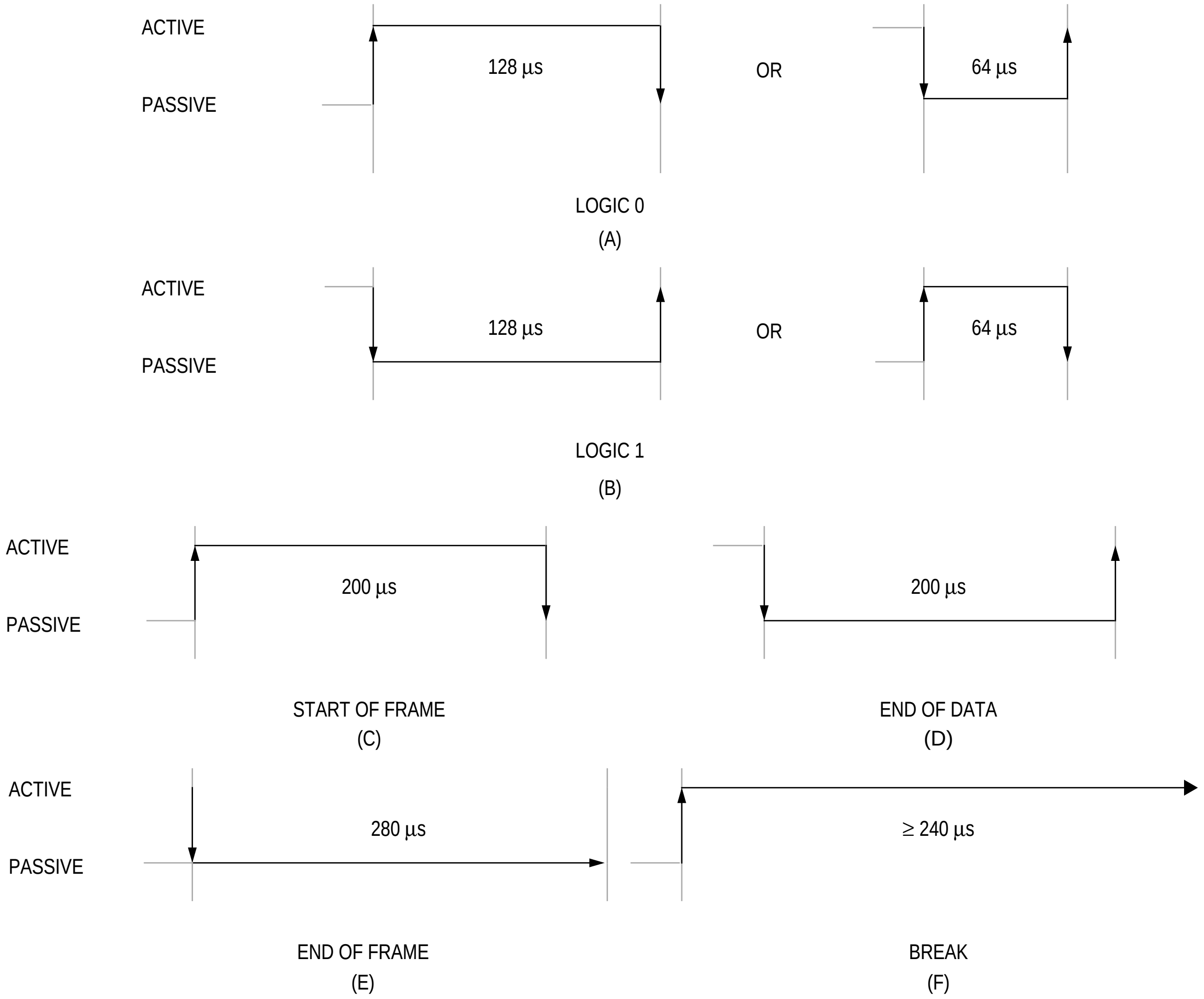


Figure 20-15. J1850 VPW Symbols

SOF — Start of Frame Symbol

The SOF symbol is defined as passive-to-active transition followed by an active period 200 ms in length (**Figure 20-15(c)**). This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic one or a logic zero.

EOD — End of Data Symbol

The EOD symbol is defined as an active-to-passive transition followed by a passive period 200 ms in length (**Figure 20-15(d)**).

EOF — End of Frame Symbol

The EOF symbol is defined as an active-to-passive transition followed by a passive period 280ms in length (**Figure 20-15(e)**). If no IFR byte is transmitted after an EOD symbol is transmitted, after another 80 ms the EOD becomes an EOF, indicating the completion of the message.

IFS — Inter-Frame Separation Symbol

The IFS symbol is defined as a passive period 300 ms in length. The IFS symbol contains no transition, since when used it always follows an EOF symbol.

BREAK — Break Signal

The BREAK signal is defined as a passive-to-active transition followed by an active period of at least 240 ms (**Figure 20-15(f)**).

20.7.4 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 has varying oscillator frequencies.

In VPW bit encoding, the tolerances for both the passive and active data bits and the symbols are defined with no gaps between definitions. For example, the maximum length of a passive logic zero is equal to the minimum length of a passive logic one, and the maximum length of an active logic zero is equal to the minimum length of a valid SOF symbol.

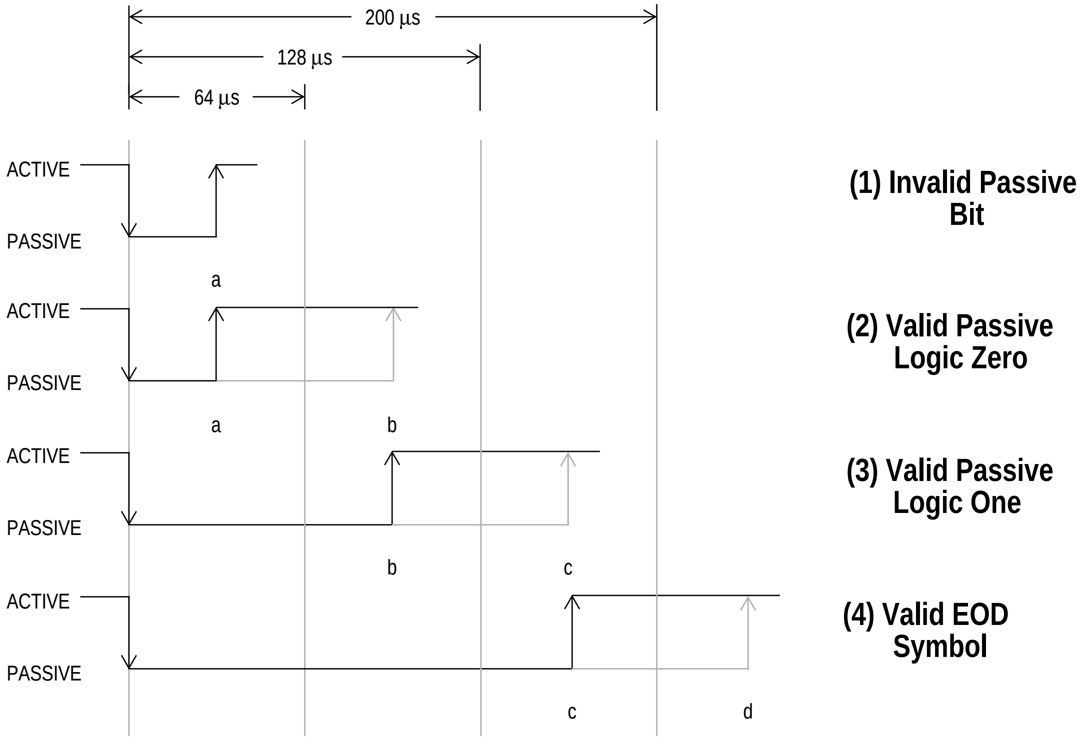


Figure 20-16. J1850 VPW Passive Symbols

Invalid Passive Bit

If the passive-to-active 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. See **Figure 20-16(1)**.

Valid Passive Logic Zero

If the passive-to-active transition beginning the next data bit or symbol occurs between **a** and **b**, the current bit would be considered a logic zero. See **Figure 20-16(2)**.

Valid Passive Logic One

If the passive-to-active transition beginning the next data bit or symbol occurs between **b** and **c**, the current bit would be considered a logic one. See **Figure 20-17(3)**.

Valid EOD Symbol

If the passive-to-active transition beginning the next data bit or symbol occurs between **c** and **d**, the current symbol would be considered a valid EOD symbol. See **Figure 20-17(4)**.

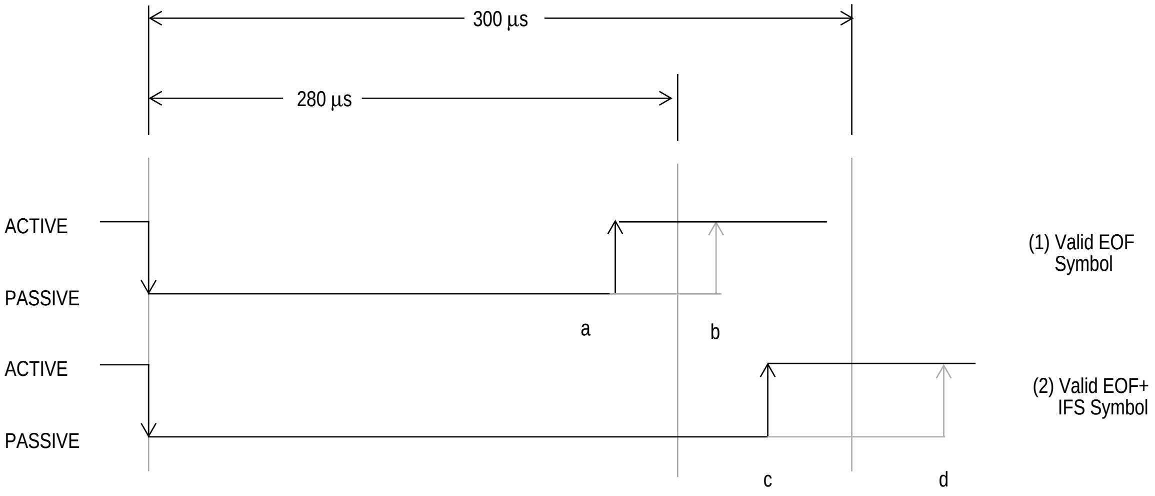


Figure 20-17. J1850 VPW EOF and IFS Symbols

Valid EOF and IFS Symbol

In **Figure 20-17(1)**, if the passive-to-active transition beginning the SOF symbol of the next message occurs between **a** and **b**, the current symbol will be considered a valid EOF symbol. If the passive-to-active 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 IFS symbol. See **Figure 20-17(2)**. 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

If the passive-to-active transition beginning the 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.

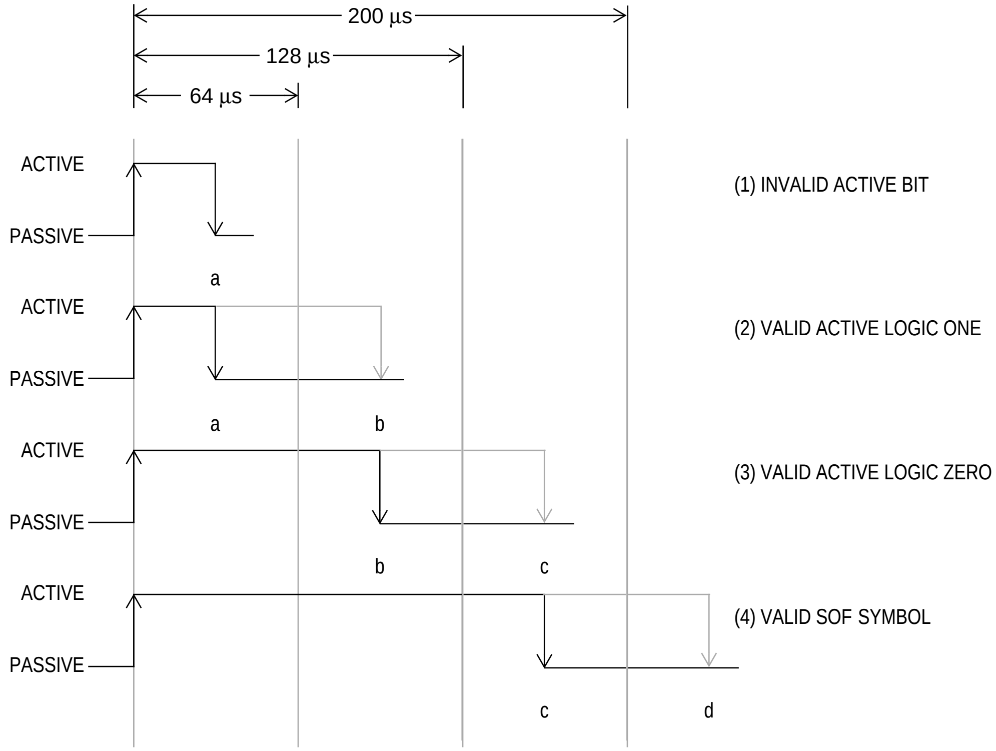


Figure 20-18. J1850 VPW Active Symbols

Invalid Active Bit

If the active-to-passive 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. See **Figure 20-18(1)**.

Valid Active Logic One

If the active-to-passive transition beginning the next data bit or symbol occurs between **a** and **b**, the current bit would be considered a logic one. See **Figure 20-18(2)**.

Valid Active Logic Zero

If the active-to-passive transition beginning the next data bit or symbol occurs between **b** and **c**, the current bit would be considered a logic zero. See **Figure 20-18(3)**.

Valid SOF Symbol

If the active-to-passive transition beginning the next data bit or symbol occurs between **c** and **d**, the current symbol would be considered a valid SOF symbol. See **Figure 20-18(4)**.

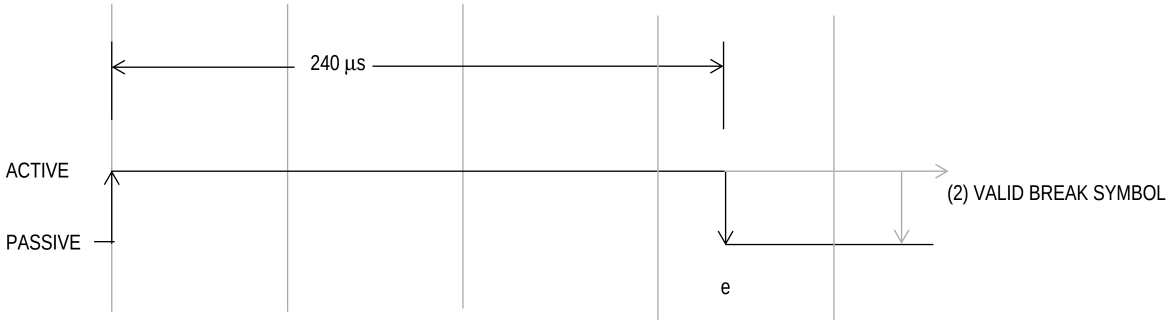


Figure 20-19. J1850 VPW BREAK Symbol

Valid BREAK Symbol

If the next active-to-passive transition does not occur until after **e**, the current symbol will be considered a valid BREAK symbol. A BREAK symbol should be followed by an SOF symbol beginning the next message to be transmitted onto the J1850 bus. See **Figure 20-19**.

20.7.5 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 continue with each bit thereafter.

The VPW symbols and J1850 bus electrical characteristics are chosen carefully so that a logic zero (active or passive type) will always dominate over a logic one (active or passive type) simultaneously transmitted. Hence logic zeroes are said to be dominant and logic ones are said to be recessive.

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

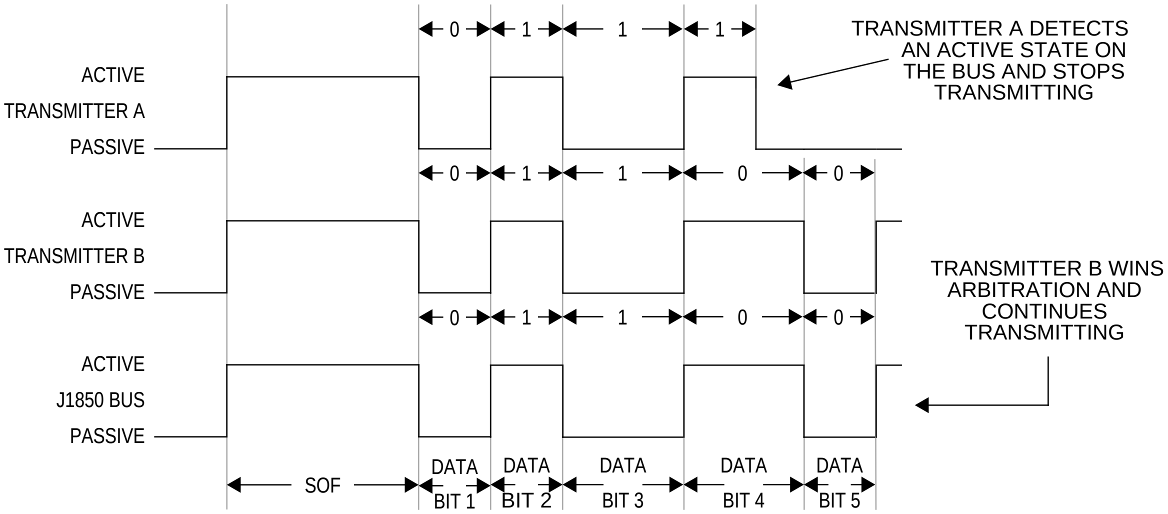


Figure 20-20. J1850 VPW Bitwise Arbitrations

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

Since a zero dominates a one, 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.

20.8 Low-Power Modes

The following application notes concern stop mode and wait mode.

20.8.1 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 BCR register 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 receive correctly the byte 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.

When BDLC stop mode is entered and the on-board analog transceiver is selected in the BARD register (assuming an on-board analog transceiver exists on the MCU), the analog circuitry within the transmitter will be put into a power conserving sleep mode. In BDLC stop mode the BDLC can neither do wave shaping nor drive any data. Therefore, ensuring that all transmissions are complete or aborted before putting the BDLC into BDLC stop mode is important when using an on-board analog transceiver.

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.

20.8.2 Wait Mode

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

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 BCR register is previously set. 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.

When BDLC wait mode is entered and the on-board analog transceiver is selected in the BARD register (assuming an on-board analog transceiver exists on the MCU), the analog circuitry within the transmitter will be put into a power-conserving sleep mode. In BDLC wait mode, the BDLC can neither do wave shaping nor drive any data. Therefore, ensuring that all transmissions are complete or aborted before putting the BDLC into wait mode is important when using an on-board analog transceiver.

Section 21. Electrical Specifications

21.1 Contents

21.2 Maximum Ratings

Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

NOTE: *This device is not guaranteed to operate properly at the maximum ratings. Refer to* ***12.5 DC Electrical Characteristics*** *for guaranteed operating conditions.*

Rating	Symbol	Value	Unit
Supply Voltage	V_{DD}	–0.3 to +7.0	V
Input Voltage	V_{IN}	V_{SS} –0.3 to V_{DD} +0.3	V
Current Drain Per Pin Excluding V_{DD} and V_{SS}	I	25	mA
Storage Temperature	T_{STG}	–55 to +150	°C
Write/Erase Cycles (@10 ms write time and –40°C, +25°C, +85°C)		10,000	Cycles
Data Retention EPROM and EEPROM (EEPROM after 10,000 cycling.)		10	Years
Programming Voltage	V_{PP}	15	V

NOTE: 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 $V_{SS} \leq (V_{IN}$ or $V_{OUT}) \leq V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}.)*

21.3 Operating Temperature Range

Rating	Symbol	Value	Unit
Operating Temperature Range MC68HC708AS48 (Standard) MC68HC708AS48 (Extended) MC68HC708AS48 (Extended) MC68HC708AS48 (Extended)	T_A	T_L to T_H 0 to +70 –40 to 85 –40 to 105 –40 to +125	°C
Operating Voltage Range	V_A	5.0 ± 10%	V

21.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal Resistance PLCC (52 Pins) QFP (64 Pins)	θ_{JA}	 50 50	°C/W

21.5 DC Electrical Characteristics (see Note 1)

Characteristic	Symbol	Min	Typ	Max	Unit
Output High Voltage (I_{LOAD} = –0.8 mA) All Ports	V_{OH}	V_{DD} –0.8	—	—	V
Output Low Voltage (I_{LOAD} = 1.6 mA) All Ports	V_{OL}	—	—	0.4	V
Input High Voltage All Ports, $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input Low Voltage All Ports, $\overline{\text{IRQ}}$, $\overline{\text{RESET}}$, OSC1	V_{IL}	V_{SS}	—	0.3 x V_{DD}	V
EPROM Programming Voltage	V_{PP}	12	13	14	V
EPROM Programming Current (1 Byte)	I_{PP}	—	5	8	mA
V_{DD} + V_{DDA} Supply Current (see Notes 2–7) Run Wait Stop 25 °C 0 °C to +70 °C (Standard) –40 °C to +85 °C (Extended) –40 °C to +125 °C	I_{DD}	 — — — — — —	 19 8 1.5 2 3 3	 38 16 3 4 6 6	mA
I/O Ports Hi-Z Leakage Current	I_L	–	–	± 10	µA
Input Current ($\overline{\text{IRQ1}}$/$V_{PP,}$ PTD2/ATD10, OSC1)	I_{IN}	–	–	± 1	µA
Capacitance (see Note 8) Ports (As Input or Output)	C_{OUT} C_{IN}	— —	— —	12 8	pF
Low-Voltage Reset Inhibit	V_{LVII}	3.7	3.9	4.1	V
Low-Voltage Reset Recover	V_{LVIR}	3.9	4.1	4.3	V
Low-Voltage Reset Inhibit/Recover Hysteresis	H_{LVI}	100	—	500	mV

NOTES:

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40 °C to +125 °C, unless otherwise noted.
2. Typical values at midpoint of voltage range, 25 °C.
3. All values shown reflect average measurements.
4. Run (Operating) I_{DD}, Wait I_{DD}: Measurement using external square wave clock source to OSC1 (f_{osc} = 4.2 MHz), all inputs 0.2 Vdc from rail; no DC loads, less than 50 pF on all outputs, C_L = 20 pF on OSC2, PLL disabled.
5. Wait, Stop I_{DD}: All ports configured as inputs, V_{IL} = 0.2 Vdc, V_{IH} = V_{DD} –0.2 Vdc, PLL disabled.
6. Stop I_{DD} measured with OSC1 = V_{SS}.
7. Wait I_{DD} is affected linearly by the OSC2 capacitance.
8. Not Tested / Guaranteed by Design

21.6 Control Timing (see Note 1)

Characteristic	Symbol	Min	Max	Unit
Bus Operating Frequency (4.5–5.5 V — V_{DD} Only)	f_{BUS}	—	4.2 M	Hz
$\overline{\text{RESET}}$ Pulse Width Low	t_{RL}	1.5	—	t_{CYC}
$\overline{\text{IRQ}}$ Interrupt Pulse Width Low (Edge-Triggered)	t_{ILHI}	1.5	—	t_{CYC}
$\overline{\text{IRQ}}$ Interrupt Pulse Period	tILIL	Note 3	—	t_{CYC}
EPROM Programming Time per Byte	t_{EPGM}	0.1	1	ms
EEPROM Programming Time per Byte	t_{EEPGM}	10	—	ms
EEPROM Erasing Time per Byte	t_{EBYTE}	10	—	ms
EEPROM Erasing Time per Block	t_{EBLOCK}	10	—	ms
EEPROM Erasing Time per Bulk	t_{EBULK}	10	—	ms
EEPROM Programming Voltage Discharge Period	t_{EEFPV}	10	—	ms
16-Bit Timer (see Note 2)				
Input Capture Pulse Width	$t_{TH,}t_{TL}$	TBD	–	ns
Input Capture Period	t_{TLTL}	Note 3	–	t_{CYC}

NOTES:

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted.
2. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
3. The minimum period t_{TLTL} or t_{ILIL} should not be less than the number of cycles it takes to execute the capture interrupt service routine plus TBD t_{CYC}.

21.7 ADC Converter Characteristics (see Note 1)

Characteristic	Min	Max	Unit	Comments
Resolution	8	8	Bits	
Absolute Accuracy (V_{REFL} = 0 V, V_{DDA} = V_{REFH} = 5 V ± 10%)	–1	+1	LSB	Includes Quantization
Conversion Range	V_{REFL}	V_{REFH}	V	$V_{REFL} = V_{SSA}$
Power-Up Time	16	17	µs	Conversion Time Period
Input Leakage (see Note 3) Ports B and D	—	± 10	µA	
Conversion Time	16	17	µs	Includes Sampling Time Based on 1 MHz ADC Clock
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 (see Note 2)	5	–	Cycles	1 MHz clock
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	

NOTES:

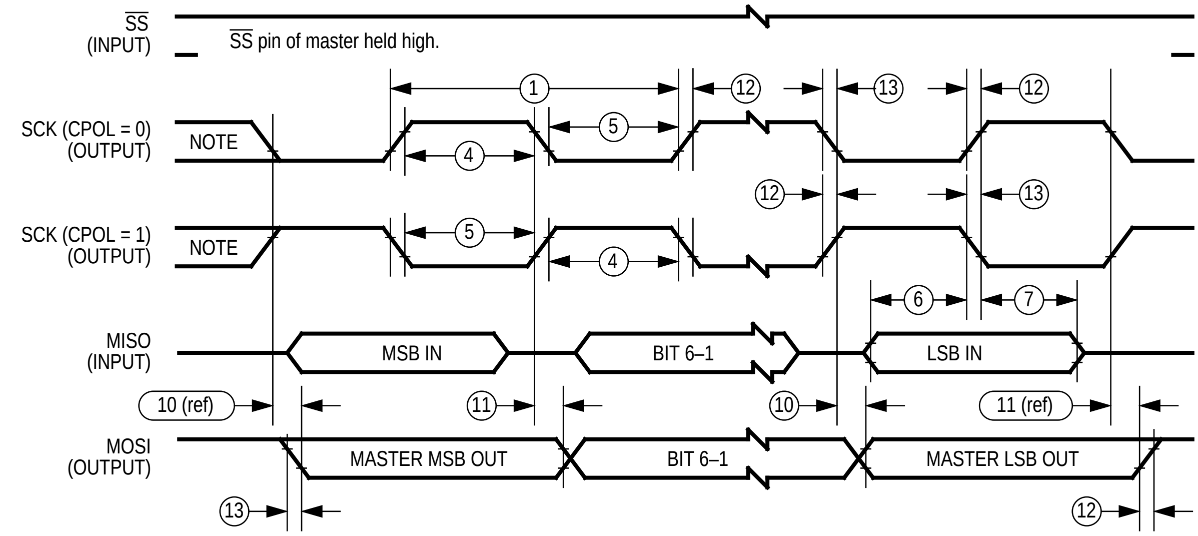
1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, V_{DDA}/V_{DDAREF} = 5.0 Vdc ± 10%, V_{SSA} = 0 Vdc, V_{REFH} = 5.0 Vdc ± 10%
2. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.
3. The external system error caused by input leakage current is approximately equal to the product of R source and input current.

21.8 Serial Peripheral Interface (SPI) Timing (see Note 1)

Num	Characteristic	Symbol	Min	Max	Unit
	Opeating Frequency Master Slave	$f_{OP(M)}$ $f_{OP(S)}$	dc dc	$f_{OP}/2$ f_{OP}	MHz
1	Cycle Time Master Slave	$t_{CYC(M)}$ $t_{CYC(S)}$	2 1	— —	t_{CYC}
2	Enable Lead Time Master Slave	$t_{LEAD(M)}$ $t_{LEAD(S)}$	Note 2 240	— —	ns
3	Enable Lag Time Master Slave	$t_{LAG(M)}$ $t_{LAG(S)}$	Note 2 240	— —	ns
4	Clock (SCK) High Time Master Slave	$t_{W(SCKH)M}$ $t_{W(SCKH)S}$	200 100	— —	ns
5	Clock (SCK) Low Time Master Slave	$t_{W(SCKL)M}$ $t_{W(SCKL)S}$	200 100	— —	ns
6	Data Setup Time (Inputs) Master Slave	$t_{SU(M)}$ $t_{SU(S)}$	50 50	— —	ns
7	Data Hold Time (Inputs) Master Slave	$t_{H(M)}$ $t_{H(S)}$	50 50	— —	ns
8	Slave Access Time (Time to Data Active from High-Impedance State)	t_A	0	120	ns
9	Slave Disable Time (Hold Time to High-Impedance State)	t_{DIS}	—	120	ns
10	Data Valid Master (Before Capture Edge) Slave (After Enable Edge) (see Note 3)	$t_{V(M)}$ $t_{V(S)}$	0.25 —	— 120	$t_{CYC(M)}$ ns
11	Data Hold Time (Outputs) Master (After Capture Edge) Slave (After Enable Edge)	$t_{HO(M)}$ $t_{HO(S)}$	0.25 0	— —	$t_{CYC(M)}$ ns
12	Rise Time (20% V_{DD} to 70% V_{DD}, C_L = 200 pF) SPI Outputs (SCK, MOSI, and MISO) SPI Inputs (SCK, MOSI, MISO, and $\overline{SS}$)	t_{RM} t_{RS}	— —	100 2	ns µs
13	Fall Time (70% V_{DD} to 20% V_{DD}, C_L = 200 pF) SPI Outputs (SCK, MOSI, and MISO) SPI Inputs (SCK, MOSI, MISO, and $\overline{SS}$)	t_{FM} t_{FS}	— —	100 2	ns µs

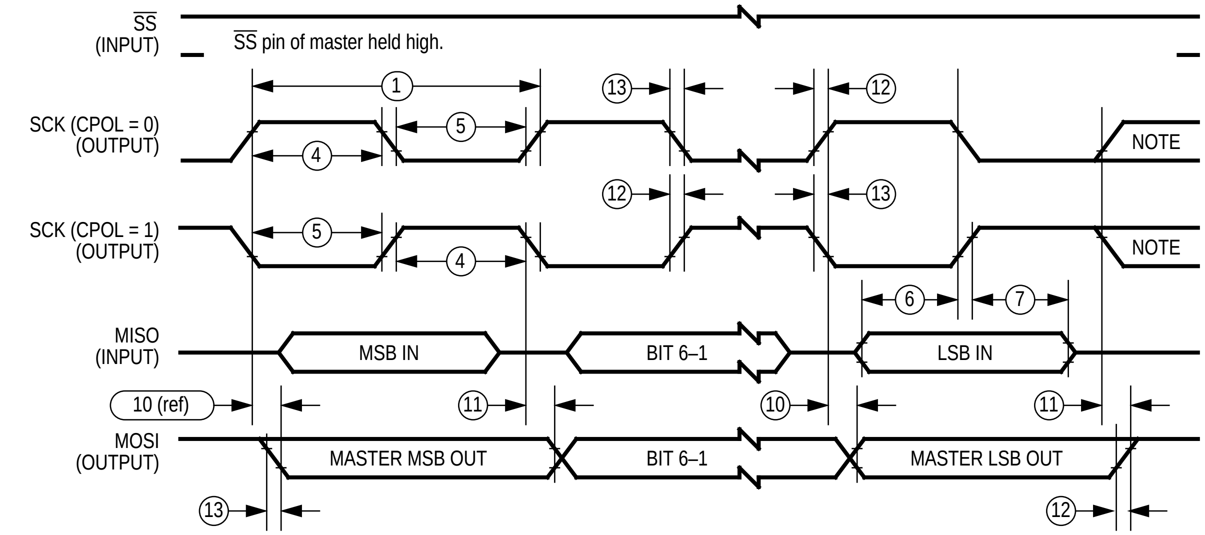
NOTES:

1. V_{DD} = 5.0 Vdc ± 10%; V_{SS} = 0 Vdc, T_A = –40 to +85 °C, unless otherwise noted. Refer to **Figure 21-1** and **Figure 21-2** for timing diagrams.
2. Signal production depends on software.
3. Assumes 200 pF load on all SPI pins



NOTE: This first clock edge is generated internally, but is not seen at the SCK pin.

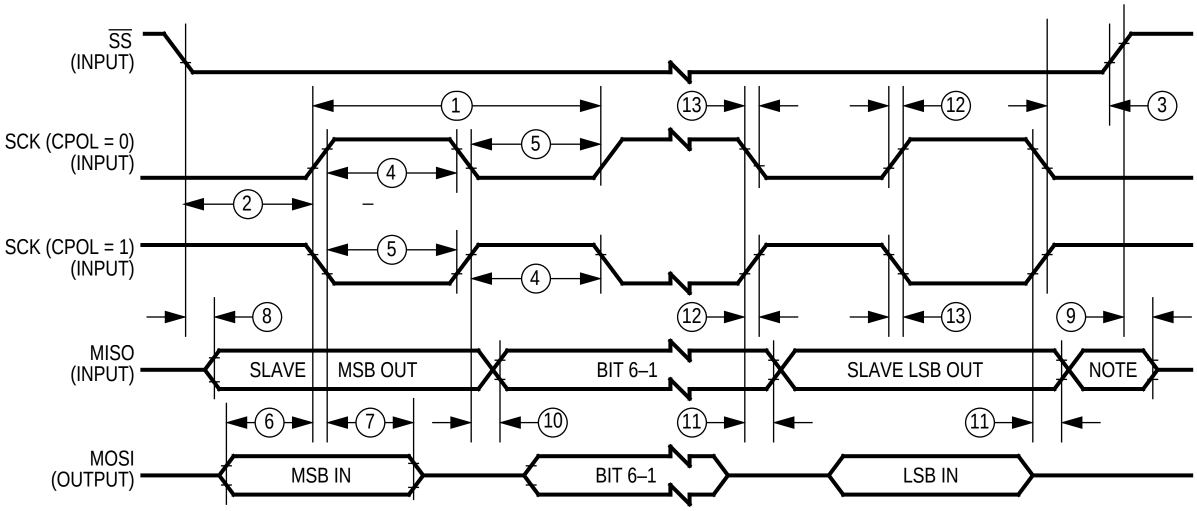
a) SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

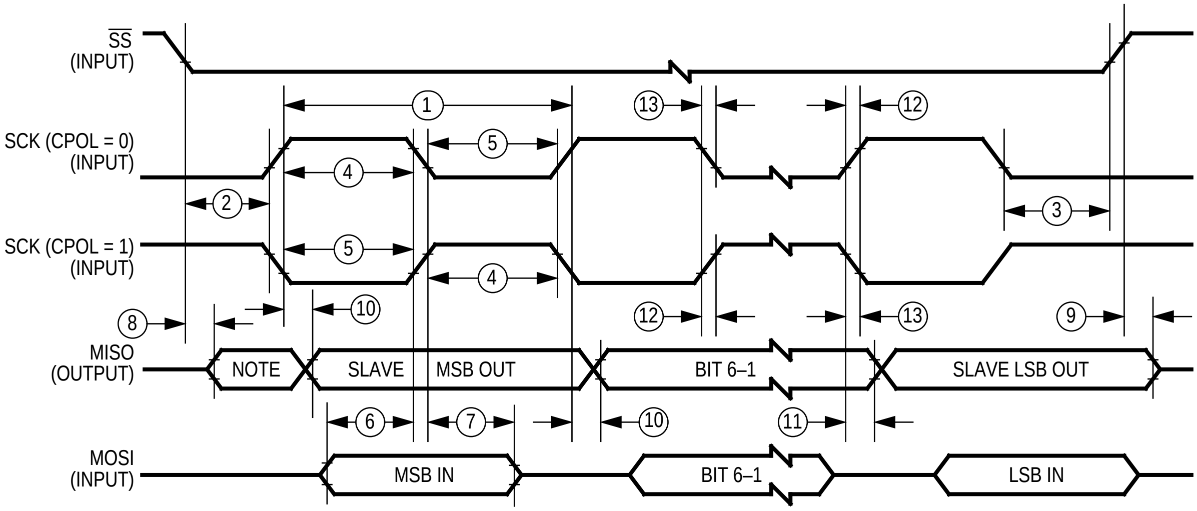
b) SPI Master Timing (CPHA = 1)

Figure 21-1. 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.

a) SPI Slave Timing (CPHA = 1)

Figure 21-2. SPI Slave Timing Diagram

21.9 BLDC Transmitter DC Electrical Characteristics (see Note 1)

Characteristic	Symbol	Min	Max	Unit
CL2TxD Output Low Voltage (I_{CL2TX} = 1.6 mA)	V_{OLTX}	—	0.4	V
CL2TxD Output High Voltage (I_{CL2TX} = –800 µA)	V_{OHTX}	V_{DD} –0.8	—	V

NOTE:
1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted.

21.10 BLDC Receiver DC Electrical Characteristics (see Note 1)

Characteristic	Symbol	Min	Max	Unit
CL2RxD Input Low Voltage	V_{ILRX}	V_{SS}	0.3 x V_{DD}	V
CL2RxD Input High Voltage	V_{IHRX}	0.7 x V_{DD}	V_{DD}	V
CL2RxD Input Low Current (CL2Rx = V_{DD})	$I_{ILCL2RX}$	–1	+1	µA
CL2RxD Input High Current (CL2Rx = 0 V)	$I_{IHCL2RX}$	–1	+1	µA

NOTE:
1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted.

21.11 BLDC Receiver VPW Symbol Timings (see Notes 1 and 2)

Characteristic	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	300	--	µs

NOTES:
1. f_{BDLC} = 1.048576 MHz, V_{DD} = 5.0 V ± 10%, V_{SS} = 0 V ± 10%.
2. The receiver symbol timing boundaries are subject to an uncertainty of 1 t_{BDLC} µs due to sampling considerations.

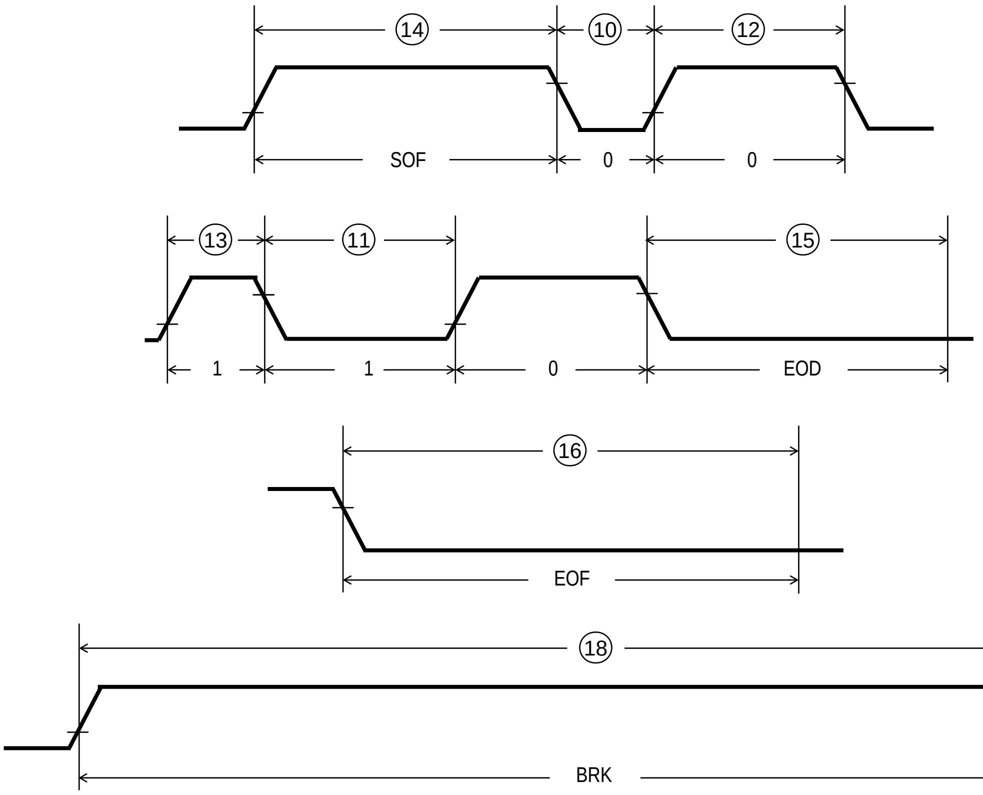


Figure 21-3. BDLC Variable Pulse Width Modulation (VPW) Symbol Timing

Appendix A. CGM Electrical Information

A.1 Contents

A.2 Introduction

This appendix provides preliminary electrical information for the clock generator module (CGM).

NOTE: *Motorola cannot guarantee the values shown in this appendix.*

A.3 CGM Operating Conditions

Characteristic	Symbol	Min	Typ	Max	Comments
Operating Voltage	V_{DD}	4.5 V	—	5.5 V	
Crystal Reference Frequency	f_{RCLK}	—	4.194 MHz	—	
Module Crystal Reference Frequency	f_{XCLK}	—	4.194 MHz	—	Same Frequency as f_{RCLK}
Crystal Oscillator Startup Time	t_{OXON}	—	TBD	t_{CYC}	Crystal Oscillator Startup Time
Stop Recovery Startup Time (Crystal Oscillator Option) Consult Crystal Manufacturer's Data	t_{ILCH}	—	TBD	t_{CYC}	Stop Recovery Startup Time (Crystal Oscillator Option)
Range Nom. Multiplier (MHz)	f_{NOM}	—	4.194	—	4.0–5.5 V V_{DD} only
VCO Center-of-Range Frequency (MHz)	f_{VRS}	4.194	—	32.0	4.0–5.5 V V_{DD} only
VCO Operating Frequency	f_{VCLK}	f_{VRSMIN}	—	f_{VRSMAX}	

A.4 CGM Component Information

Description	Symbol	Min	Typ	Max	Comments
Crystal Load Capacitance	C_L	—	—	—	Consult Crystal Manufacturer's Data
Crystal Fixed Capacitance	C1	—	2 x CL	—	Consult Crystal Manufacturer's Data
Crystal Tuning Capacitance	C2	—	2 x CL	—	Consult Crystal Manufacturer's Data
Filter Capacitor	C_F	—	C_{FACT} x (V_{DDA}/f_{XCLK})	—	See 8.5.3
Bypass Capacitor	C_{BYP}	—	0.1 µF	—	CBYP must provide low AC impedance from f = f_{XCLK}/100 to 100 x f_{VCLK}, so series resistance must be considered.

A.5 CGM Acquisition/Lock Time Information

Description	Symbol	Min	Typ	Max	Notes
Manual Mode Time to Stable	t_{ACQ}	—	$(8 \times V_{DDA})/(f_{XCLK} \times K_{ACQ})$	—	If C_F chosen correctly
Manual Stable to Lock Time	t_{AL}	—	$(4 \times V_{DDA})/(f_{XCLK} \times K_{TRK})$	—	If C_F chosen correctly
Manual Acquisition Time	t_{LOCK}	—	$t_{ACQ}+t_{AL}$	—	
Tracking Mode Entry Frequency Tolerance	D_{TRK}	0	—	± 3.6%	
Acquisition Mode Entry Frequency Tolerance	D_{UNT}	± 6.3%	—	± 7.2%	
LOCK Entry Freq. Tolerance	D_{LOCK}	0	—	± 0.9%	
LOCK Exit Freq. Tolerance	D_{UNL}	± 0.9%	—	± 1.8%	
Reference Cycles per Acquisition Mode Measurement	n_{ACQ}	—	32	—	
Reference Cycles per Tracking Mode Measurement	n_{TRK}	—	128	—	
Automatic Mode Time to Stable	t_{ACQ}	n_{ACQ}/f_{XCLK}	$(8 \times V_{DDA})/(f_{XCLK} \times K_{ACQ})$		If C_F chosen correctly
Automatic Stable to Lock Time	t_{AL}	n_{TRK}/f_{XCLK}	$(4 \times V_{DDA})/(f_{XCLK} \times K_{TRK})$	—	If C_F chosen correctly
Automatic Lock Time	t_{LOCK}	—	$t_{ACQ}+t_{AL}$	—	
PLL Jitter (Deviation of Average Bus Frequency over 2 ms)		0	—	$\pm (f_{CRYS}) \times (.025\%) \times (N/4)$	N= VCO freq. mult. (GBNT)

NOTES:

1. GBNT — guaranteed but not tested
2. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = -40°C to +125°C, unless otherwise noted.

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and Ⓜ are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

How to reach us:
USA/EUROPE: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036. 1-800-441-2447
MFAX: RMFAX0@email.sps.mot.com – TOUCHTONE (602) 244-6609
INTERNET: http://Design-NET.com
JAPAN: Nippon Motorola Ltd.; Tatsumi-SPD-JLDC, Toshikatsu Otsuki, 6F Seibu-Butsuryu-Center, 3-14-2 Tatsumi Koto-Ku, Tokyo 135, Japan. 03-3521-8315
HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park, 51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852-26629298

MOTOROLA