MC68HC08QA24 Technical Data

M68HC08
Microcontrollers

Rev. 0.1
MC68HC08QA24/D
August 5, 2005

freescale.com

Freescale was negligent regarding the design or manufacture of the part.

List of Sections

Table of Contents

Section 7. Central Processor Unit (CPU)

Section 9. Clock Generator Module (CGM)

Section 10. System Integration Module (SIM)

Section 13. Monitor ROM (MON)

Section 14. Computer Operating Properly (COP)

Section 17. Serial Communications Interface (SCI)

Section 20. Timer Interface A (TIMA)

Section 21. Timer Interface B (TIMB)

Section 22. CAN Controller

Section 24. Mechanical Data

List of Figures

List of Figures

List of Figures

List of Tables

List of Tables

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC08QA24 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 MC68HC08QA24 include:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- 8.4-MHz internal bus frequency
- 24,064 bytes of user read-only memory (ROM)
- ROM data security(1)
- 512 bytes of on-chip electrically erasable programmable read-only memory (EEPROM)
- 768 bytes of on-chip random-access memory (RAM)
- Serial peripheral interface module (SPI)
- Serial communications interface module (SCI)
- Two 16-bit, 2-channel timer interface modules (TIMA and TIMB)
- Clock generator module (CGM)
- 8-bit, 12-channel analog-to-digital converter module (ADC)
- Scalable controller area network module (MSCAN)

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

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

Features of the CPU08 include:

- Enhanced HC05 programming model
- Extensive loop control functions
- 16 addressing modes (eight more than the HC05)
- 16-bit index register and stack pointer
- Memory-to-memory data transfers
- Fast 8 $\times$ 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 MC68HC08QA24.

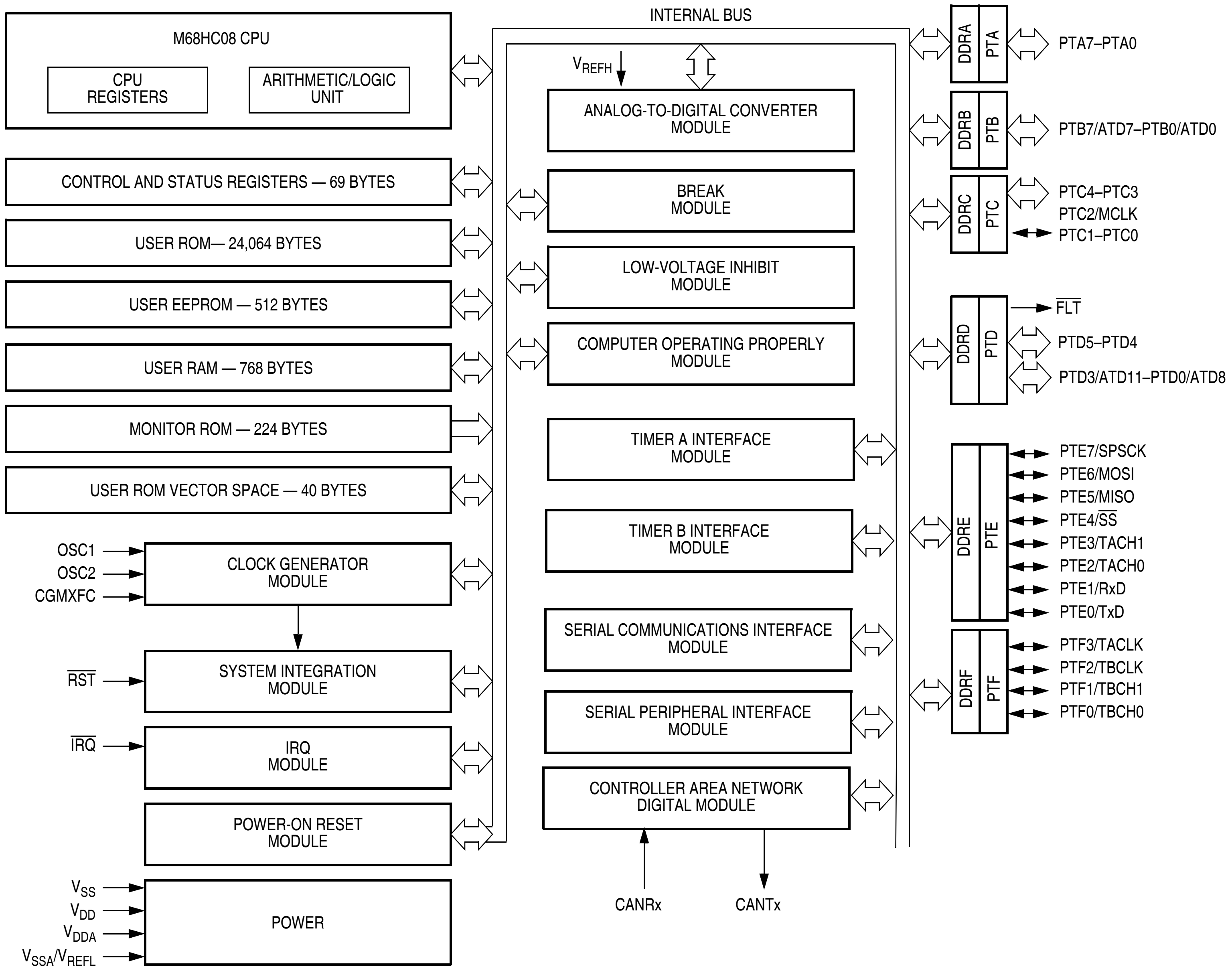


Figure 1-1. MCU Block Diagram

1.5 Pin Assignments

Figure 1-2 shows the 52-pin plastic leaded chip carrier (PLCC) assignments.

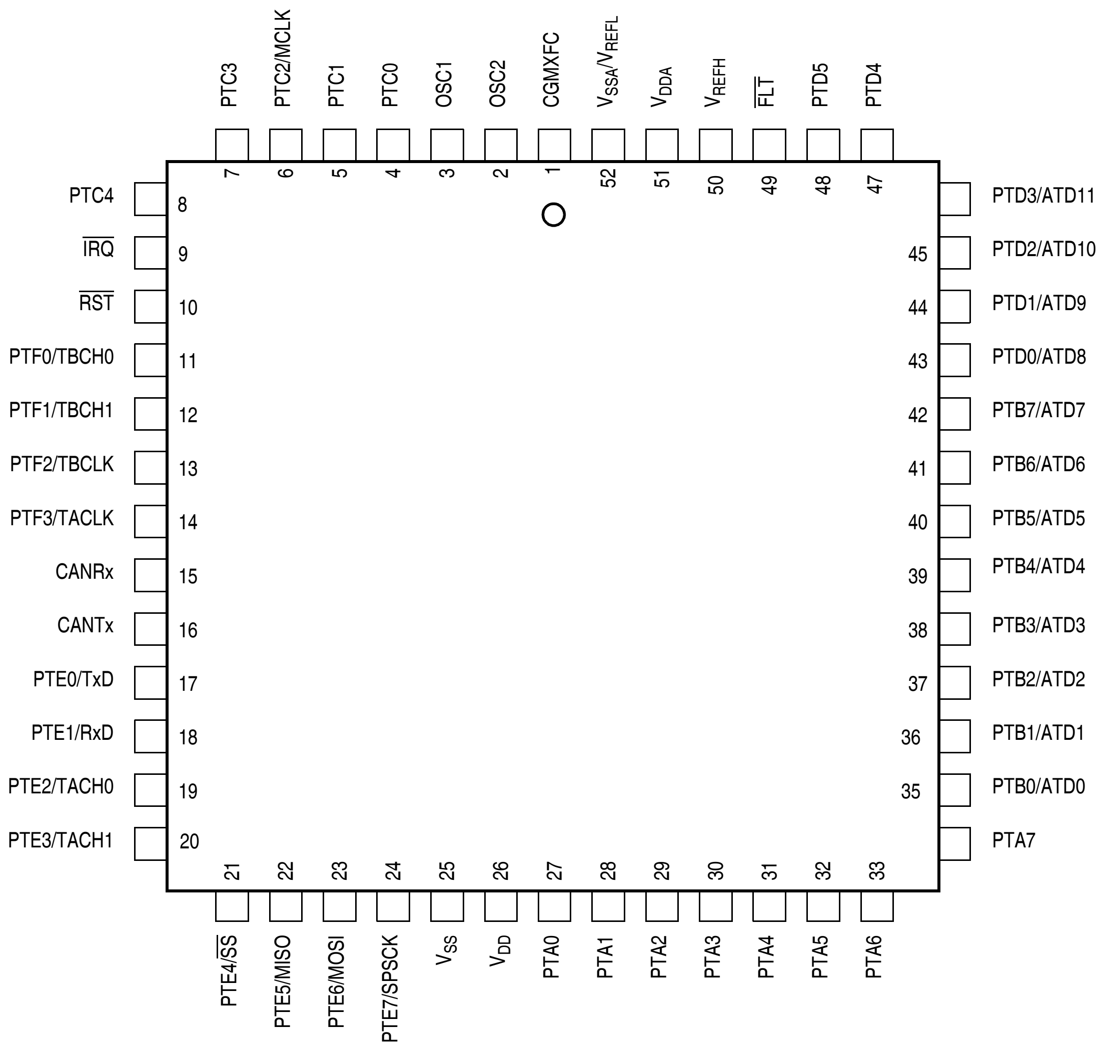
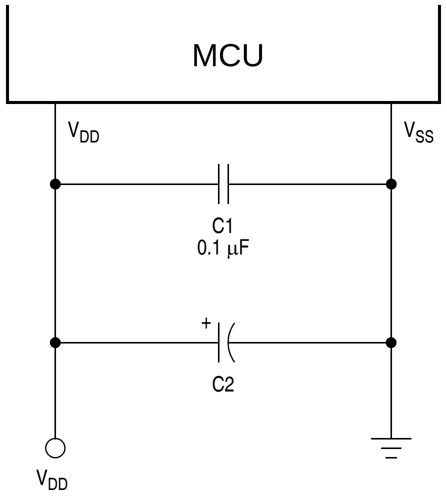


Figure 1-2. 52-Pin PLCC Assignments (Top View)

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 1-3** shows. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency-response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.



Note: Component values shown represent typical applications.

Figure 1-3. Power Supply Bypassing

V_{SS} is also the ground for the port output buffers and the ground return for the serial clock in the serial peripheral interface module (SPI). (See **Section 18. 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 9. Clock Generator Module (CGM)**.)

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

A logic 0 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 10. System Integration Module (SIM)** for more information.)

1.5.4 External Interrupt Pin ($\overline{\text{IRQ}}$)

$\overline{\text{IRQ}}$ is an asynchronous external interrupt pin. (See **Section 15. External Interrupt (IRQ)**.)

1.5.5 Analog Power Supply Pin (V_{DDA})

V_{DDA} is the power supply pin for the analog portion of the chip. These modules are the analog-to-digital converter (ADC) and the clock generator module (CGM). (See **Section 9. Clock Generator Module (CGM)** and **Section 19. 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). V_{SSA}/V_{REFL} is also the lower reference supply for the ADC. (See **Section 9. Clock Generator Module (CGM)** and **Section 19. Analog-to-Digital Converter (ADC)**.)

1.5.7 External Filter Capacitor Pin (CGMXFC)

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

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

PTA7–PTA0 are general-purpose bidirectional input/output (I/O) port pins. (See **Section 16. 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 19. Analog-to-Digital Converter (ADC)** and **Section 16. Input/Output (I/O) Ports**.)

1.5.10 Port C I/O Pins (PTC4–PTC0)

PTC4–PTC3 and PTC1–PTC0 are general-purpose, bidirectional I/O port pins. PTC2/MCLK is a special function port that shares its pin with the system clock. (See **Section 16. Input/Output (I/O) Ports**.)

1.5.11 Port D I/O Pins (PTD5–PTD0/ATD8)

Port D is a 6-bit special function port that shares four of its pins with the analog-to-digital converter module (ADC). (See **Section 19. Analog-to-Digital Converter (ADC)** and **Section 16. Input/Output (I/O) Ports**.)

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 A interface module (TIMA), 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 17. Serial Communications Interface (SCI)**, **Section 18. Serial Peripheral Interface (SPI)**, **Section 20. Timer Interface A (TIMA)**, and **Section 16. Input/Output (I/O) Ports**.)

1.5.13 Port F I/O Pins (PTF3/TACLK–PTF0/TBCH0)

Port F is a 4-bit special function port that shares three of its pins with the timer B interface module (TIMB) and one of its pins with the timer A interface module(TIMA). (See **Section 20. Timer Interface A (TIMA)**, **Section 21. Timer Interface B (TIMB)** and **Section 16. Input/Output (I/O) Ports**.)

1.5.14 CAN Transmit Pin (CANTx)

This pin is the digital output from the controller area network (CAN) module (CANTx). (See **Section 22. CAN Controller**.)

1.5.15 CAN Receive Pin (CANRx)

This pin is the digital input to the CAN module (CANRx). (See **Section 22. CAN Controller**.)

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
PTC4–PTC0	General-purpose I/O	Dual state	No	Input hi-z
$\overline{\text{FLT}}$	Output only for dual CPU fault	Dual state	No	Output
PTD5–PTD4	General-purpose I/O	Dual state	No	Input hi-z
PTD3/ATD11–PTD0/ATD8	General-purpose I/O ADC channel	Dual state	No	Input hi-z
PTE7/SPSCK	General-purpose I/O SPI clock	Dual state open drain	Yes	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/TACH1	General-purpose I/O timer A channel 1	Dual state	Yes	Input hi-z
PTE2/TACH0	General-purpose I/O timer A channel 0	Dual state	Yes	Input hi-z
PTE1/RxD	General-purpose I/O SCI receive data	Dual state	Yes	Input hi-z
PTE0/TxD	General-purpose I/O SCI transmit data	Dual state	Yes	Input hi-z
PTF3/TACLK	General-purpose I/O timer A input clock	Dual state	Yes	Input hi-z
PTF2/TBCLK	General-purpose I/O timer B input clock	Dual state	Yes	Input hi-z
PTF1/TBCH1	General-purpose I/O timer B channel 1	Dual state	Yes	Input hi-z
PTF0/TBCH0	General-purpose I/O timer B channel 0	Dual state	Yes	Input hi-z
V_{DD}	Chip power supply	N/A	N/A	N/A

Table 1-1. External Pins Summary (Continued)

Pin Name	Function	Driver Type	Hysteresis	Reset State
V_{SS}	Chip ground	N/A	N/A	N/A
V_{DDA}	Analog power supply	N/A	N/A	N/A
V_{SSA}/V_{REFL}	Analog ground/ ADC reference voltage	N/A	N/A	N/A
V_{REFH}	A/D reference voltage	N/A	N/A	N/A
OSC1	External clock in	N/A	N/A	Input hi-z
OSC2	External clock out	N/A	N/A	Output
CGMXFC	PLL loop filter capacitor	N/A	N/A	N/A
$\overline{\text{IRQ}}$	External interrupt request	N/A	N/A	Input hi-z
$\overline{\text{RST}}$	Reset	N/A	N/A	Output low
CANRx	CAN serial input	N/A	Yes	Input hi-z
CANTx	CAN serial output	Output	No	Output

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:

- 24,064 bytes of user ROM
- 768 bytes of RAM
- 512 bytes of EEPROM
- 40 bytes of user-defined vectors
- 224 bytes of monitor ROM

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

- Reserved — Accessing a reserved location can have unpredictable effects on MCU operation.
- Unimplemented — Accessing an unimplemented location causes an illegal address reset if illegal address resets are enabled.

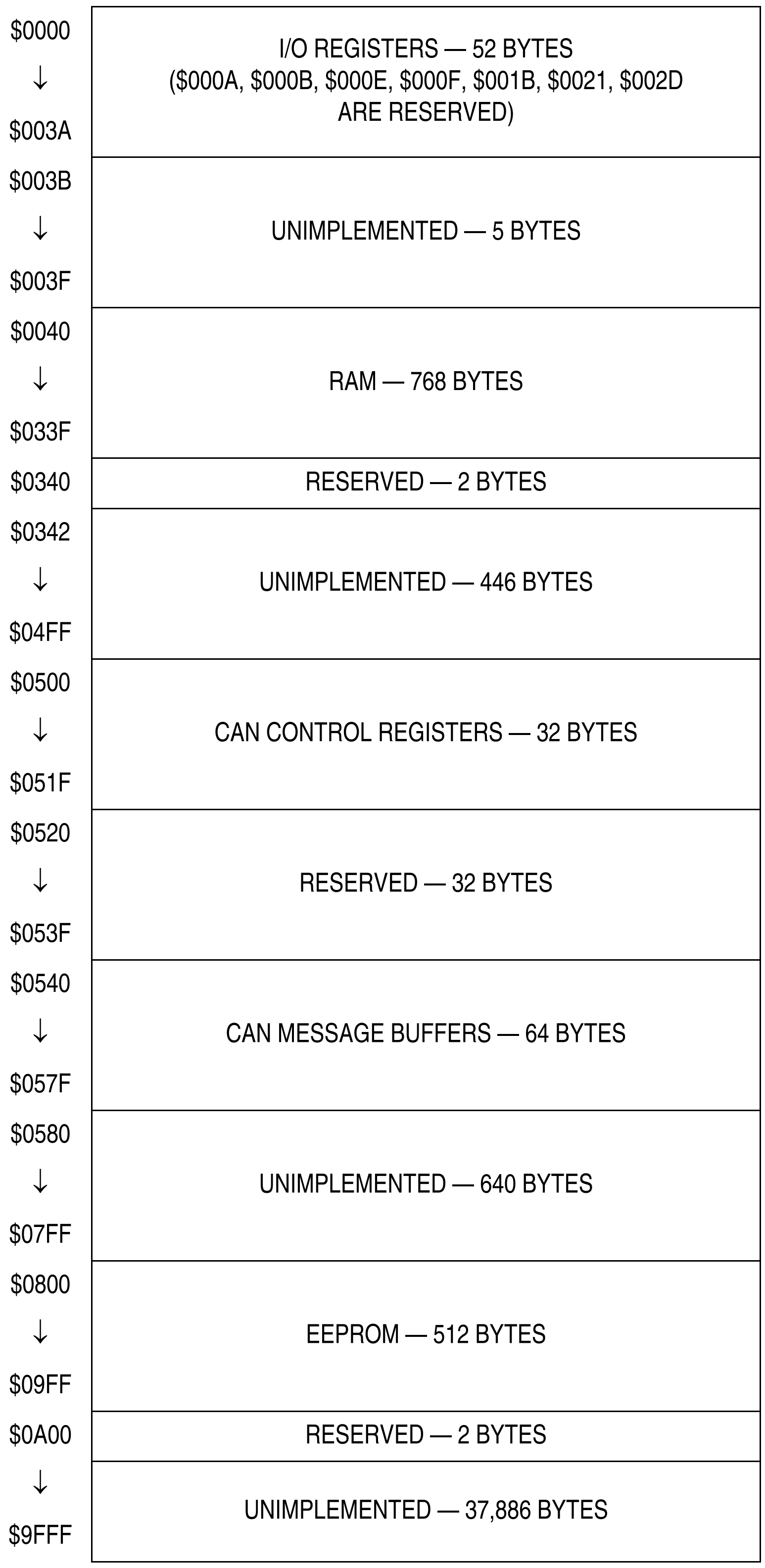


Figure 2-1. Memory Map

$A000 ↓ $FDFF	ROM — 24,064 BYTES
$FE00 ↓ $FE1F	SYSTEM AND MEMORY CONTROL REGISTERS — 32 BYTES
$FE20 ↓ $FEFF	MONITOR ROM — 224 BYTES
$FF00 ↓ $FF7F	UNIMPLEMENTED — 128 BYTES
$FF80	RESERVED — 1 BYTE
$FF81 ↓ $FFD7	UNIMPLEMENTED — 87 BYTES
$FFD8 ↓ $FFFF	INTERRUPT AND RESET VECTORS — 40 BYTES

Figure 2-1. Memory Map (Continued)

2.3 Summary of Internal Registers

Control and status registers for most of the internal functional modules are located within addresses $0000–$003A, shown in **Figure 2-2**. The control and message buffers for the CAN are located in addresses $0500–$057F as shown in the CAN module section of this specification. Additional control registers are located between $FE00 and $FE1F and are also shown in **Figure 2-2**.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA) See page 188.	Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB) See page 190.	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC) See page 193.	Read:	0	0	0	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:	R	R	R					
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD) See page 196.	Read:	0	0	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Write:	R	R						
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA) See page 188.	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB) See page 191.	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC) See page 194.	Read:	MCLKEN	0	0	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:		R	R					
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD) See page 197.	Read:	0	0	DDRD5	DDRD4	DDRD3	DDR2	DDRD1	DDRD0
		Write:	R	R						
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE) See page 199.	Read:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Write:								
		Reset:	Unaffected by reset							

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
	Port F Data Register	Read:	0	0	0	0	PTF3	PTF2	PTF1	PTF0
$0009	(PTF)	Write:	R	R	R	R				
	See page 203.	Reset:	Unaffected by reset							
$000A	Reserved		R	R	R	R	R	R	R	R
$000B	Reserved		R	R	R	R	R	R	R	R
	Data Direction Register E	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
$000C	(DDRE)	Write:								
	See page 201.	Reset:	0	0	0	0	0	0	0	0
	Data Direction Register F	Read:	0	0	0	0	DDRF3	DDRF2	DDRF1	DDRF0
$000D	(DDRF)	Write:	R	R	R	R				
	See page 204.	Reset:	0	0	0	0	0	0	0	0
$000E	Reserved		R	R	R	R	R	R	R	R
$000F	Reserved		R	R	R	R	R	R	R	R
	SPI Control Register	Read:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
$0010	(SPCR)	Write:								
	See page 266.	Reset:	0	0	1	0	1	0	0	0
	SPI Status and Control	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
$0011	Register (SPSCR)	Write:	R		R	R	R			
	See page 269.	Reset:	0	0	0	0	1	0	0	0
	SPI Data Register	Read:	R7	R6	R5	R4	R3	R2	R1	R0
$0012	(SPDR)	Write:	T7	T6	T5	T4	T3	T2	T1	T0
	See page 272.	Reset:	Unaffected by reset							

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 2 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0013	SCI Control Register 1 (SCC1) See page 225.	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0014	SCI Control Register 2 (SCC2) See page 228.	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 3 (SCC3) See page 231.	Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
		Write:	R							
		Reset:	U	U	0	0	0	0	0	0
$0016	SCI Status Register 1 (SCS1) See page 233.	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:	R	R	R	R	R	R	R	R
		Reset:	1	1	0	0	0	0	0	0
$0017	SCI Status Register 2 (SCS2) See page 237.	Read:	0	0	0	0	0	0	BKF	RPF
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR) See page 238.	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0019	SCI Baud Rate Register (SCBR) See page 238.	Read:	R	R	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001A	IRQ Status and Control Register (ISCR) See page 184.	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:	R	R	R	R	R	ACK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Reserved		R	R	R	R	R	R	R	R

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001C	PLL Control Register (PCTL) See page 118.	Read:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
		Write:		R			R	R	R	R
		Reset:	0	0	1	0	1	1	1	1
$001D	PLL Bandwidth Control Register (PBWC) See page 120.	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	XLD	0	0	0	0
		Write:		R			R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$001E	PLL Programming Register (PPG) See page 122.	Read:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		Write:								
		Reset:	0	1	1	0	0	1	1	0
$001F	Mask Option Register (MOR) See page 68.	Read:	LVISTOP	ROMSEC	LVIRST	LVIPWR	SSREC	COPS	STOP	COPD
		Write:	R	R	R	R	R	R	R	R
		Reset:	Unaffected by reset							
$0020	TIMA Status and Control Register (TASC) See page 300.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$0021	Reserved		R	R	R	R	R	R	R	R
$0022	TIMA Counter Register High (TACNTH) See page 302.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0023	TIMA Counter Register Low (TACNTL) See page 302.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0024	TIMA Modulo Register High (TAMODH) See page 303.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1

R = Reserved   U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0025	TIMA Modulo Register Low (TAMODL) See page 303.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0026	TIMA Channel 0 Status and Control Register (TASC0) See page 304.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0027	TIMA Channel 0 Register High (TACH0H) See page 308.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	TIMA Channel 0 Register Low (TACH0L) See page 308.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0029	TIMA Channel 1 Status and Control Register (TASC1) See page 304.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$002A	TIMA Channel 1 Register High (TACH1H) See page 308.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002B	TIMA Channel 1 Register Low (TACH1L) See page 308.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$002C	TIMB Status and Control Register (TBSC) See page 326.	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$002D	Reserved		R	R	R	R	R	R	R	R

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002E	TIMB Counter Register High (TBCNTH) See page 328.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$002F	TIMB Counter Register Low (TBCNTL) See page 328.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0030	TIMB Modulo Register High (TBMODH) See page 329.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0031	TIMB Modulo Register Low (TBMODL) See page 329.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0032	TIMB Channel 0 Status and Control Register (TBSC0) See page 330.	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0033	TIMB Channel 0 Register High (TBCH0H) See page 334.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0034	TIMB Channel 0 Register Low (TBCH0L) See page 334.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0035	TIMB Channel 1 Status and Control Register (TBSC1) See page 330.	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0036	TIMB Channel 1 Register High (TBCH1H) See page 334.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0037	TIMB Channel 1 Register Low (TBCH1L) See page 334.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0038	Analog-to-Digital Status and Control Register (ADSCR) See page 279.	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:	R							
		Reset:	0	0	0	1	1	1	1	1
$0039	Analog-to-Digital Data Register (ADR) See page 282.	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	R	R	R	R	R	R	R	R
		Reset:	Indeterminate after reset							
$003A	Analog-to-Digital Input Clock Register (ADICLK) See page 282.	Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
		Write:					R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0500	CAN Module Control Register 0 (CMCR0) See page 365.	Read:	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
		Write:	R	R	R	R		R		
		Reset:	0	0	0	0	0	0	0	0
$0501	CAN Module Control Register 1 (CMCR1) See page 367.	Read:	0	0	0	0	0	LOOPB	WUPM	CLKSRC
		Write:	R	R	R	R	R			
		Reset:	0	0	0	0	0	0	0	0
$0502	CAN Bus Timing Register 0 (CBTR0) See page 368.	Read:	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0503	CAN Bus Timing Register 1 (CBTR1) See page 369.	Read:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0504	CAN Receiver Flag Register (CRFLG) See page 371.	Read:	WUPF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
		Write:								
		Reset:	0	0	0	0	0	0	0	0

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 7 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0505	CAN Receiver Interrupt Enable Register (CRIER) See page 373.	Read:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0506	CAN Transmitter Flag Register (CTFLG) See page 374.	Read:	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
		Write:	R	R	R	R	R			
		Reset:	0	0	0	0	0	1	1	1
$0507	CAN Transmitter Control Register (CTCR) See page 375.	Read:	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
		Write:	R				R			
		Reset:	0	0	0	0	0	1	1	1
$0508	CAN Identifier Acceptance Control Register (CIDAC) See page 376.	Read:	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
		Write:	R	R			R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0509	Reserved		R	R	R	R	R	R	R	R
↓	↓									
$050D	Reserved		R	R	R	R	R	R	R	R
$050E	CAN Receiver Error Counter Register (CRXERR) See page 378.	Read:	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$050F	CAN Transmitter Error Counter Register (CTXERR) See page 378.	Read:	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0510	CAN Identifier Acceptance Register 0 (CIDAR0) See page 379.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							

R = Reserved   U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 8 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0511	CAN Identifier Acceptance Register 1 (CIDAR1) See page 379.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0512	CAN Identifier Acceptance Register 2 (CIDAR2) See page 379.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0513	CAN Identifier Acceptance Register 3 (CIDAR3) See page 379.	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0514	CAN Identifier Mask Register 0 (CIDMR0) See page 381.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0515	CAN Identifier Mask Register 1 (CIDMR1) See page 381.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0516	CAN Identifier Mask Register 0 (CIDMR0) See page 381.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0517	CAN Identifier Mask Register 1 (CIDMR1) See page 381.	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0540	CAN Receive Identifier Register 0 (CRIDR0) See page 360.	Read:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
		Write:								
		Reset:	Unaffected by reset							
$0541	CAN Receive Identifier Register 1 (CRIDR1) See page 360.	Read:	ID20	ID19	ID16	SRR(1)	IDE(1)	ID17	ID16	ID15
		Write:								
		Reset:	Unaffected by reset							

R	= Reserved	U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 9 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0542	CAN Receive Identifier Register 2 (CRIDR2) See page 360.	Read:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
		Write:								
		Reset:	Unaffected by reset							
$0543	CAN Receive Identifier Register 3 (CRIDR3) See page 360.	Read:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
		Write:								
		Reset:	Unaffected by reset							
$0544	CAN Receiver Data Segment Register 0 (CRDSR0) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0545	CAN Receiver Data Segment Register 1 (CRDSR1) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0546	CAN Receiver Data Segment Register 2 (CRDSR2) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0547	CAN Receiver Data Segment Register 3 (CRDSR3) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0548	CAN Receiver Data Segment Register 4 (CRDSR4) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0549	CAN Receiver Data Segment Register 5 (CRDSR5) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$054A	CAN Receiver Data Segment Register 6 (CRDSR6) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							

R = Reserved   U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 10 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$054B	CAN Receiver Data Segment Register 7 (CRDSR7) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$054C	CAN Receiver Data Length Register (CRDSR7) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$054D	Reserved		R	R	R	R	R	R	R	R
↓	↓									
$054F	Reserved		R	R	R	R	R	R	R	R
$0550	CAN Transmit 0 Identifier Register 0 (CT0IDR0) See page 360.	Read: Write:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
		Reset:	Unaffected by reset							
$0551	CAN Transmit 0 Identifier Register 1 (CT0IDR1) See page 360.	Read: Write:	ID20	ID19	ID16	SRR(1)	IDE(1)	ID17	ID16	ID15
		Reset:	Unaffected by reset							
$0552	CAN Transmit 0 Identifier Register 2 (CT0IDR2) See page 360.	Read: Write:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
		Reset:	Unaffected by reset							
$0553	CAN Transmit 0 Identifier Register 3 (CT0IDR3) See page 360.	Read: Write:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
		Reset:	Unaffected by reset							
$0554	CAN Transmit 0 Data Segment Register 0 (CT0DSR0) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							

R = Reserved   U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 11 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0555	CAN Transmit 0 Data Segment Register 1 (CT0DSR1) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0556	CAN Transmit 0 Data Segment Register 2 (CT0DSR2) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0557	CAN Transmit 0 Data Segment Register 3 (CT0DSR3) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0558	CAN Transmit 0 Data Segment Register 4 (CT0DSR4) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0559	CAN Transmit 0 Data Segment Register 5 (CT0DSR5) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$055A	CAN Transmit 0 Data Segment Register 6 (CT0DSR6) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$055B	CAN Transmit 0 Data Segment Register 7 (CT0DSR7) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$055C	CAN Transmit 0 Data Length Register (CT0DSR7) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$055D	CAN Transmit 0 Buffer Priority Register (CT0TBPR) See page 362.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 12 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$055E	Reserved		R	R	R	R	R	R	R	R
$055F	Reserved		R	R	R	R	R	R	R	R
$0560	CAN Transmit 1 Identifier Register 0 (CT1IDR0) See page 360.	Read: Write:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
		Reset:	Unaffected by reset							
$0561	CAN Transmit 1 Identifier Register 1 (CT1IDR1) See page 360.	Read: Write:	ID20	ID19	ID16	SRR(1)	IDE(1)	ID17	ID16	ID15
		Reset:	Unaffected by reset							
$0562	CAN Transmit 1 Identifier Register 2 (CT1IDR2) See page 360.	Read: Write:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
		Reset:	Unaffected by reset							
$0563	CAN Transmit 1 Identifier Register 3 (CT1IDR3) See page 360.	Read: Write:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
		Reset:	Unaffected by reset							
$0564	CAN Transmit 1 Data Segment Register 0 (CT1DSR0) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$0565	CAN Transmit 1 Data Segment Register 1 (CT1DSR1) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$0566	CAN Transmit 1 Data Segment Register 2 (CT1DSR2) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							

R = Reserved   U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 13 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0567	CAN Transmit 1 Data Segment Register 3 (CT1DSR3) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$0568	CAN Transmit 1 Data Segment Register 4 (CT1DSR4) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$0569	CAN Transmit 1 Data Segment Register 5 (CT1DSR5) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$056A	CAN Transmit 1 Data Segment Register 6 (CT1DSR6) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$056B	CAN Transmit 1 Data Segment Register 7 (CT1DSR7) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$056C	CAN Transmit 1 Data Length Register (CT1DSR7) See page 361.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$056D	CAN Transmit 1 Buffer Priority Register (CT1TBPR) See page 362.	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Reset:	Unaffected by reset							
$056E	Reserved		R	R	R	R	R	R	R	R
$056F	Reserved		R	R	R	R	R	R	R	R

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 14 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0570	CAN Transmit 2 Identifier Register 0 (CT2IDR0) See page 360.	Read:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
		Write:								
		Reset:	Unaffected by reset							
$0571	CAN Transmit 2 Identifier Register 1 (CT2IDR1) See page 360.	Read:	ID20	ID19	ID16	SRR(1)	IDE(1)	ID17	ID16	ID15
		Write:								
		Reset:	Unaffected by reset							
$0572	CAN Transmit 2 Identifier Register 2 (CT2IDR2) See page 360.	Read:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
		Write:								
		Reset:	Unaffected by reset							
$0573	CAN Transmit 2 Identifier Register 3 (CT2IDR3) See page 360.	Read:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
		Write:								
		Reset:	Unaffected by reset							
$0574	CAN Transmit 2 Data Segment Register 0 (CT2DSR0) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0575	CAN Transmit 2 Data Segment Register 1 (CT2DSR1) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0576	CAN Transmit 2 Data Segment Register 2 (CT2DSR2) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0577	CAN Transmit 2 Data Segment Register 3 (CT2DSR3) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$0578	CAN Transmit 2 Data Segment Register 4 (CT2DSR4) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							

R = Reserved   U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 15 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0579	CAN Transmit 2 Data Segment Register 5 (CT2DSR5) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$057A	CAN Transmit 2 Data Segment Register 6 (CT2DSR6) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$057B	CAN Transmit 2 Data Segment Register 7 (CT2DSR7) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$057C	CAN Transmit 2 Data Length Register (CT2DSR7) See page 361.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$057D	CAN Transmit 2 Buffer Priority Register (CT2TBPR) See page 362.	Read:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
		Write:								
		Reset:	Unaffected by reset							
$057E	Reserved		R	R	R	R	R	R	R	R
$057F	Reserved		R	R	R	R	R	R	R	R
$FE00	SIM Break Status Register (SBSR) See page 147.	Read:	R	R	R	R	R	R	SBSW	R
		Write:								
		Reset:							0	
$FE01	SIM Reset Status Register (SRSR) See page 149.	Read:	POR	PIN	COP	ILOP	ILAD	EDC	LVI	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	1	U	0	0	0	U	U	0

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 16 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE03	SIM Break Flag Control Register (SBFCR) See page 150.	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04	Reserved		R	R	R	R	R	R	R	R
↓	↓									
$FE07	Reserved		R	R	R	R	R	R	R	R
$FE09	EDC Control Register (EDCR) See page 100.	Read:	0	R	0	0	R	FDB	FABH	FABL
		Write:	SFLTB		R	R				
		Reset:	1		0	0	1	0	0	0
$FE0C	Break Address Register High (BRKH) See page 162.	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address Register Low (BRKL) See page 162.	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Status and Control Register (BRKSCR) See page 161.	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:			R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE0F	LVI Status Register (LVISR) See page 153.	Read:	LVIOUT	0	0	0	0	0	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE1C	EEPROM Non-Volatile Register (EENVR) See page 79.	Read:	EERA	R	R	R	EEBP3	EEBP2	EEBP1	EEBP0
		Write:								
		Reset:	PV	R	R	R	PV	PV	PV	PV

PV = Programmed value or 1 in the erased state.

R	= Reserved	U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 17 of 18)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE1D	EEPROM Control Register (EECR) See page 77.	Read:	EEBCLK	0	EEOFF	EERAS1	EERAS0	EELAT	0	EEPGM
		Write:		R					R	
		Reset:	0	0	0	0	0	0	0	0
$FE1E	Reserved		R	R	R	R	R	R	R	R
$FE1F	EEPROM Array Control Register (EEACR) See page 79.	Read:	EERA	R	R	R	EEBP3	EEBP2	EEBP1	EEBP0
		Write:	R	R	R	R	R	R	R	R
		Reset:	EENVR	R	R	R	EENVR	EENVR	EENVR	EENVR
$FF80	Reserved		R	R	R	R	R	R	R	R
$FFFF	COP Control Register (COPCTL) See page 177.	Read:	Low byte of reset vector							
		Write:	Clear COP counter							
		Reset:	Unaffected by reset							

R = Reserved U = Unaffected

Figure 2-2. Control, Status, and Data Registers (Sheet 18 of 18)

Table 2-1. Vector Addresses

Lowest Priority

$\downarrow$

Highest Priority

Address	Vector
$FFD8	CAN Transmit Vector (High)
$FFD9	CAN Transmit Vector (Low)
$FFDA	CAN Receive Vector (High)
$FFDB	CAN Receive Vector (Low)
$FFDC	CAN Error Vector (High)
$FFDD	CAN Error Vector (Low)
$FFDE	CAN Wakeup Vector (High)
$FFDF	CAN Wakeup Vector (Low)
$FFE0	ADC Vector (High)
$FFE1	ADC Vector (Low)
$FFE2	SCI Transmit Vector (High)
$FFE3	SCI Transmit Vector (Low)
$FFE4	SCI Receive Vector (High)
$FFE5	SCI Receive Vector (Low)
$FFE6	SCI Error Vector (High)
$FFE7	SCI Error Vector (Low)
$FFE8	SPI Transmit Vector (High)
$FFE9	SPI Transmit Vector (Low)
$FFEA	SPI Receive Vector (High)
$FFEB	SPI Receive Vector (Low)
$FFEC	TIM B Overflow Vector (High)
$FFED	TIM B Overflow Vector (Low)
$FFEE	TIM B Channel 1 Vector (High)
$FFEF	TIM B Channel 1 Vector (Low)
$FFF0	TIM B Channel 0 Vector (High)
$FFF1	TIM B Channel 0 Vector (Low)
$FFF2	TIM A Overflow Vector (High)
$FFF3	TIM A Overflow Vector (Low)
$FFF4	TIM A Channel 1 Vector (High)
$FFF5	TIM A Channel 1 Vector (Low)
$FFF6	TIM A Channel 0 Vector (High)
$FFF7	TIM A Channel 0 Vector (Low)
$FFF8	PLL Vector (High)
$FFF9	PLL Vector (Low)
$FFFA	IRQ Vector (High)
$FFFB	IRQ Vector (Low)
$FFFC	SWI Vector (High)
$FFFD	SWI Vector (Low)
$FFFE	Reset Vector (High)
$FFFF	Reset Vector (Low)

Section 3. Random-Access Memory (RAM)

3.1 Contents

3.2 Introduction

This section describes the 768 bytes of random-access memory (RAM).

3.3 Functional Description

Addresses \$0040–\$033F are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 768-byte memory space.

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

Within page zero are 192 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for input/output (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 M68HC05, M6805, and M146805 compatibility, the H register is not stacked.*

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

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

Section 4. Read-Only Memory (ROM)

4.1 Contents

4.2 Introduction

This section describes the 24,064 bytes of user read-only memory (ROM), 224 bytes of monitor ROM, and 36 bytes of user vectors.

4.3 Functional Description

The user ROM consists of 24,064 bytes of ROM from addresses $A000–$FDFF. The monitor ROM is located from $FE20–$FEFF. See **Figure 2-1. Memory Map**.

Twenty of the user vectors, $FFD8–$FFFF, are dedicated to user-defined reset and interrupt vectors.

Security has been incorporated into the MC68HC08QA24 to prevent external viewing of the ROM contents. This feature ensures that customer-developed software remains proprietary.(1)

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

Section 5. Mask Options

5.1 Contents

5.2 Introduction

This section describes use of mask options by custom-masked read-only memory (ROM) and the mask option register (MOR) in the MC68HC08QA24.

5.3 Functional Description

The mask options are hard-wired connections, specified at the same time as the ROM code, which allow the user to customize the MCU. The options control the enable or disable ability of these functions:

- ROM security(1)
- Resets caused by the low-voltage inhibit (LVI) module
- Power to the LVI module
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- COP timeout period (2^{18}–2^4 CGMXCLK cycles or 2^{13}–2^4 CGMXCLK cycles)

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

- STOP instruction
- Computer operating properly module (COP)

The mask option register ($001F) is used in the initialization of various options. For error-free compatibility with the emulator one time programmable (OTP) M68HC908QA24, a write to $001F in the MC68HC08QA24has no effect in MCU operation.

5.4 Mask Option Register

Address: $001F

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

R = Reserved

Figure 5-1. Mask Option Register (MOR)

LVISTOP — LVI Stop Mode Enable Bit

LVISTOP enables the LVI module in stop mode. (See **Section 11. Low-Voltage Inhibit (LVI)**.)
1 = LVI enabled during stop mode
0 = LVI disabled during stop mode

NOTE: *To have the LVI enabled in stop mode, the LVIPWR must be at a logic 0 and the LVISTOP bit must be at a logic 1. Take note that by enabling the LVI in stop mode, the stop I_{DD} current will be higher.*

ROMSEC — ROM Security Bit

ROMSEC enables the ROM security feature. Setting the ROMSEC bit prevents reading of the ROM contents. Access to the ROM is denied to unauthorized users of customer specified software.
1 = ROM security enabled
0 = ROM security disabled

LVIRST — Low-Voltage Inhibit Reset Bit

LVIRST enables the reset signal from the LVI module.
(See **Section 11. Low-Voltage Inhibit (LVI)**.)

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

LVIPWR— LVI Power Enable Bit

LVIPWR enables the LVI module. (See **Section 11. Low-Voltage Inhibit (LVI)**.)

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

SSREC — Short Stop Recovery Bit

SSREC enables the CPU to exit stop mode with a delay of 32 CGMXCLK cycles instead of a 4096-CGMXCLK cycle delay. (See **10.7.2 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.*

COPS— COP Short Timeout Bit

COPS selects the short COP timeout period. (See **Section 14. Computer Operating Properly (COP)**.)

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

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. (See **Section 14. Computer Operating Properly (COP)**.)

1 = COP module disabled
0 = COP module enabled

Section 6. Electrically Erasable Programmable Read-Only Memory (EEPROM)

6.1 Contents

6.2 Introduction

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

6.3 Features

EEPROM features include:

- Byte, block, or bulk erasable
- Non-volatile redundant array option
- Non-volatile block protection option
- Non-volatile MCU configuration bits
- On-chip charge pump for programming/erasing

6.4 Functional Description

Addresses \$0800–\$09FF are EEPROM locations. The 512 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, 128-byte, block protection option. These options are stored in the EEPROM non-volatile register (EENVR) and are loaded into the EEPROM array configuration register (EEACR) after reset or a read of EENVR. The EEPROM array can also be disabled to reduce current.

6.4.1 EEPROM Programming

The unprogrammed state is a logic 1. Programming changes the state to a logic 0. 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 256 bytes addresses will also program the last 256 bytes addresses 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.

Use this step-by-step procedure to program a byte of EEPROM. Refer to **23.6 Control Timing** for timing values.

1. Clear EERAS1 and EERAS0 and set EELAT in the EEPROM control register (EECR) ($FE1D). Set value of t_{EEPGM}. (See notes 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 to fall, t_{EEFPV}.
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, the mode will be erase mode.

b. Setting the EELAT bit configures the address and data buses to latch data for programming the array. Only data with a valid EEPROM address will be latched. If 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 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 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 only EEPGM to allow time for removal of high voltage from the EEPROM array.

6.4.2 EEPROM Erasing

The unprogrammed state is a logic 1. 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 256 bytes ($0800–$08FF) will also erase the last 256 bytes ($0900–$09FF).

Use this step-by-step procedure to erase EEPROM. Refer to **23.6 Control Timing** for timing values.

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_{EEPGM}, to program the byte.
5. Clear EEPGM bit.
6. Wait for the erasing voltage time, t_{EEFPV}, to fall.
7. Clear EELAT bits. (See note c.)
8. Repeat steps 1 through 7 for more EEPROM byte/block erasing.

EEBPx bit must be cleared to erase EEPROM data in the corresponding block. If any EEBPx is set, the corresponding block cannot be erased and bulk erase mode does not apply.

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 can be used in step 2. All locations within this block will be erased. In bulk erase mode, any EEPROM address can 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 after a valid EEPROM write.

b. To ensure proper erasing sequence, the EEPGM bit cannot be set if the EELAT bit is cleared and a non-EEPROM write has occurred. Once EEPGM 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 EEPGM and EELAT bits with a single instruction will clear only EEPGM to allow time for removal of high voltage from the EEPROM array.

In general, all bits should be erased before being programmed. However, if program/erase cycling is of concern, this procedure can be used to minimize bit cycling in each EEPROM byte. If any bit in a byte must be changed from a 0 to a 1, the byte needs to be erased before programming. Table 6-1 summarizes the conditions for erasing before programming.

Table 6-1. 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

6.4.3 EEPROM Block Protection

The 512 bytes of EEPROM are divided into four 128-byte blocks. Each of these blocks can be protected separately 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 6-2** shows the address ranges within the blocks.

Table 6-2. EEPROM Array Address Blocks

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

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. 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, EEBP3 and EEBP2 will have no meaning.

6.4.4 EEPROM Redundant Mode

To extend the EEPROM data retention, the array can be placed in redundant mode. In this mode, the first 256 bytes of user EEPROM array are mapped to the last 256 bytes. Reading, programming, and erasing of the first 256 EEPROM bytes (\$0800–\$08FF) will physically affect two bytes of EEPROM. Addressing the last 256 bytes will not be recognized. Block protection still applies but EEBP3 and EEBP2 are meaningless.

NOTE: *Before entering redundant mode, program the EEPROM in non-redundant mode.*

6.4.5 EEPROM Configuration

The 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 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 effect with a system reset or a read of the EENVR.

6.4.6 EEPROM Control Register

EECR is a read/write register that controls programming/erasing of the array.

Address: $FE1D

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

R = Reserved

Figure 6-1. EEPROM Control Register (EECR)

EEBCLK — EEPROM Bus Clock Enable Bit

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: *Use the internal RC oscillator for applications in the 3- to 5-V range.*

EEOFF — EEPROM Power Down Bit

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 to stabilize after clearing the EEOFF bit.*

EERAS1 and EERAS0 — EEPROM Erase Bits

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

Table 6-3. 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 Bit

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 Bit

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

6.4.7 EEPROM Non-Volatile Register and EEPROM Array Configuration Register

These registers configure the EEPROM array blocks for programming purposes. EEACR loads its contents from the EENVR register at reset and upon any read of the EENVR register.

Address: $FE1F

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

R = Reserved

Figure 6-2. EEPROM Array Control Register (EEACR)

Address: $FE1C

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

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

Figure 6-3. EEPROM Non-Volatile Register (EENVR)

EERA — EEPROM Redundant Array Bit

This programmable/eraseable/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 in redundant mode configuration
0 = EEPROM array in normal mode configuration

EEBP3–EEBP0 — EEPROM Block Protection Bits

These programmable/eraseable/readable bits in EENVR and read-only bits in EEACR select blocks of EEPROM array from being programmed or erased. Reset loads EEBP3–EEBP0 from EENVR to EEACR. See **6.4.3 EEPROM Block Protection**.

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

6.4.8 Low-Power Modes

This subsection describes the low-power modes.

6.4.8.1 Wait Mode

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

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

NOTE: *The module requires a recovery time to stabilize after leaving stop mode. Attempts to access the array during the recovery time will result in unpredictable behavior.*

Section 7. Central Processor Unit (CPU)

7.1 Contents

7.2 Introduction

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

7.3 Features

Features of the CPU include:

- Fully 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
- 8.4-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 mode and wait mode

7.4 CPU Registers

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

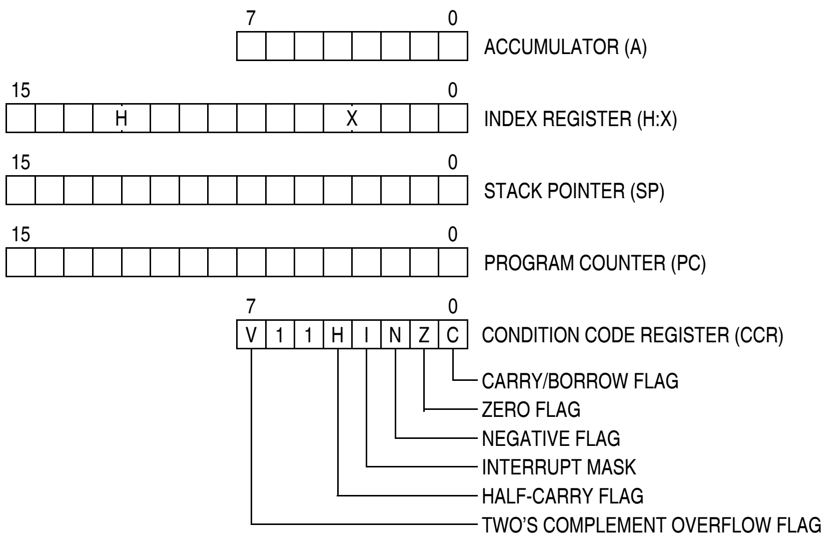


Figure 7-1. CPU Registers

7.4.1 Accumulator

The accumulator (A) 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
Read:								
Write:								
Reset:	Unaffected by reset							

Figure 7-2. Accumulator (A)

7.4.2 Index Register

The 16-bit index register (H:X) 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
Read:																
Write:																
Reset:	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X

X = Indeterminate

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

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

7.4.3 Stack Pointer

The stack pointer (SP) 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
Read:																
Write:																
Reset:	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

Figure 7-4. Stack Pointer (SP)

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

7.4.4 Program Counter

The program counter (PC) 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
Read:																
Write:																
Reset:	Loaded with vector from $FFFE and $FFFF															

Figure 7-5. Program Counter (PC)

7.4.5 Condition Code Register

The 8-bit condition code register (CCR) contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bit 6 and bit 5 are set permanently to logic 1. This subsection describes the functions of the condition code register.

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

X = Indeterminate

Figure 7-6. Condition Code Register (CCR)

V — Overflow Flag Bit

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 Bit

The CPU sets the half-carry flag when a carry occurs between accumulator bits 3 and 4 during an add without carry (ADD) or add with carry (ADC) operation. The half-carry flag is required for binary coded decimal (BCD) arithmetic operations. The decimal adjust A (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 Bit

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 M68HC05 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 push H onto stack (PSHH) and pull H from stack (PULH) instructions.*

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

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

N — Negative Flag Bit

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 Bit

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 Bit

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

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

7.5 Arithmetic/Logic Unit (ALU)

The arithmetic/logic unit (ALU) performs the arithmetic and logic operations defined by the instruction set.

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

7.6 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 12. 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.

7.7 Instruction Set Summary

Table 7-1 provides a summary of the M68HC08 instruction set.

Table 7-1. Instruction Set Summary (Sheet 1 of 7)

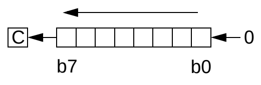
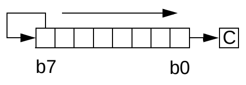
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ADD #*opr* ADD *opr* ADD *opr* ADD *opr*,X ADD *opr*,X ADD ,X ADD *opr*,SP ADD *opr*,SP	Add without Carry	A ← (A) + (M)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	AB BB CB DB EB FB 9EEB 9EDB	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
AIS #*opr*	Add Immediate Value (Signed) to SP	SP ← (SP) + (16 « M)	–	–	–	–	–	–	IMM	A7	ii	2
AIX #*opr*	Add Immediate Value (Signed) to H:X	H:X ← (H:X) + (16 « M)	–	–	–	–	–	–	IMM	AF	ii	2
AND #*opr* AND *opr* AND *opr* AND *opr*,X AND *opr*,X AND ,X AND *opr*,SP AND *opr*,SP	Logical AND	A ← (A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A4 B4 C4 D4 E4 F4 9EE4 9ED4	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X ASL *opr*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
ASR *opr* ASRA ASRX ASR *opr*,X ASR *opr*,X ASR *opr*,SP	Arithmetic Shift Right		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	4 1 1 4 3 5
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + 2 + rel ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BCLR *n*, *opr*	Clear Bit n in M	Mn ← 0	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + 2 + *rel* ? (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *opr*	Branch if Greater Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) = 0	–	–	–	–	–	–	REL	90	rr	3
BGT *opr*	Branch if Greater Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) | (N ⊕ V) = 0	–	–	–	–	–	–	REL	92	rr	3

Table 7-1. Instruction Set Summary (Sheet 2 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BHCC *rel*	Branch if Half Carry Bit Clear	PC ← (PC) + 2 + *rel* ? (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS *rel*	Branch if Half Carry Bit Set	PC ← (PC) + 2 + *rel* ? (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI *rel*	Branch if Higher	PC ← (PC) + 2 + *rel* ? (C) | (Z) = 0	–	–	–	–	–	–	REL	22	rr	3
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X BIT *opr*,SP BIT *opr*,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE *opr*	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) | (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? (C) | (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *opr*	Branch if Less Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n*,*opr*,*rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2	–	–	–	–	–	–	REL	21	rr	3
BRSET *n*,*opr*,*rel*	Branch if Bit *n* in M Set	PC ← (PC) + 3 + *rel* ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5

Table 7-1. Instruction Set Summary (Sheet 3 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr,rel* CBEQA *#opr,rel* CBEQX *#opr,rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = \$00 PC ← (PC) + 3 + rel ? (A) – (M) = \$00 PC ← (PC) + 3 + rel ? (X) – (M) = \$00 PC ← (PC) + 3 + rel ? (A) – (M) = \$00 PC ← (PC) + 2 + rel ? (A) – (M) = \$00 PC ← (PC) + 4 + rel ? (A) – (M) = \$00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2
CLR *opr* CLRA CLRX CLRH CLR *opr*,X CLR ,X CLR *opr*,SP	Clear	M ← \$00 A ← \$00 X ← \$00 H ← \$00 M ← \$00 M ← \$00 M ← \$00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4
CMP *#opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X CMP *opr*,SP CMP *opr*,SP	Compare A with M	(A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
COM *opr* COMA COMX COM *opr*,X COM ,X COM *opr*,SP	Complement (One's Complement)	M ← $(\overline{M})$ = \$FF – (M) A ← $(\overline{A})$ = \$FF – (M) X ← $(\overline{X})$ = \$FF – (M) M ← $(\overline{M})$ = \$FF – (M) M ← $(\overline{M})$ = \$FF – (M) M ← (M) = \$FF – (M)	0	–	–	↕	↕	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	4 1 1 4 3 5
CPHX *#opr* CPHX *opr*	Compare H:X with M	(H:X) – (M:M + 1)	↕	–	–	↕	↕	↕	IMM DIR	65 75	ii ii+1 dd	3 4
CPX *#opr* CPX *opr* CPX *opr* CPX ,X CPX *opr*,X CPX *opr*,X CPX *opr*,SP CPX *opr*,SP	Compare X with M	(X) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9EE3 9ED3	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	$(A)_{10}$	U	–	–	↕	↕	↕	INH	72		2

Table 7-1. Instruction Set Summary (Sheet 4 of 7)

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Cycles
DBNZ *opr,rel*	Decrement and Branch if Not Zero	A ← (A) – 1 or M ← (M) – 1 or X ← (X) – 1 PC ← (PC) + 3 + *rel* ? (result) ≠ 0	–	–	–	–	–	–	DIR	3B	dd rr	5
DBNZA *rel*		PC ← (PC) + 2 + *rel* ? (result) ≠ 0							INH	4B	rr	3
DBNZX *rel*		PC ← (PC) + 2 + *rel* ? (result) ≠ 0							INH	5B	rr	3
DBNZ *opr*,X,*rel*		PC ← (PC) + 3 + *rel* ? (result) ≠ 0							IX1	6B	ff rr	5
DBNZ X,*rel*		PC ← (PC) + 2 + *rel* ? (result) ≠ 0							IX	7B	rr	4
DBNZ *opr*,SP,*rel*		PC ← (PC) + 4 + *rel* ? (result) ≠ 0							SP1	9E6B	ff rr	6
DEC *opr*	Decrement	M ← (M) – 1	↕	–	–	↕	↕	–	DIR	3A	dd	4
DECA		A ← (A) – 1							INH	4A		1
DECX		X ← (X) – 1							INH	5A		1
DEC *opr*,X		M ← (M) – 1							IX1	6A	ff	4
DEC ,X		M ← (M) – 1							IX	7A		3
DEC *opr*,SP		M ← (M) – 1							SP1	9E6A	ff	5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		7
EOR #*opr*	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM	A8	ii	2
EOR *opr*									DIR	B8	dd	3
EOR *opr*									EXT	C8	hh ll	4
EOR *opr*,X									IX2	D8	ee ff	4
EOR *opr*,X									IX1	E8	ff	3
EOR ,X									IX	F8		2
EOR *opr*,SP									SP1	9EE8	ff	4
EOR *opr*,SP									SP2	9ED8	ee ff	5
INC *opr*	Increment	M ← (M) + 1	↕	–	–	↕	↕	–	DIR	3C	dd	4
INCA		A ← (A) + 1							INH	4C		1
INCX		X ← (X) + 1							INH	5C		1
INC *opr*,X		M ← (M) + 1							IX1	6C	ff	4
INC ,X		M ← (M) + 1							IX	7C		3
INC *opr*,SP		M ← (M) + 1							SP1	9E6C	ff	5
JMP *opr*	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR	BC	dd	2
JMP *opr*									EXT	CC	hh ll	3
JMP *opr*,X									IX2	DC	ee ff	4
JMP *opr*,X									IX1	EC	ff	3
JMP ,X									IX	FC		2
JSR *opr*	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Unconditional Address	–	–	–	–	–	–	DIR	BD	dd	4
JSR *opr*									EXT	CD	hh ll	5
JSR *opr*,X									IX2	DD	ee ff	6
JSR *opr*,X									IX1	ED	ff	5
JSR ,X									IX	FD		4
LDA #*opr*	Load A from M	A ← (M)	0	–	–	↕	↕	–	IMM	A6	ii	2
LDA *opr*									DIR	B6	dd	3
LDA *opr*									EXT	C6	hh ll	4
LDA *opr*,X									IX2	D6	ee ff	4
LDA *opr*,X									IX1	E6	ff	3
LDA ,X									IX	F6		2
LDA *opr*,SP									SP1	9EE6	ff	4
LDA *opr*,SP									SP2	9ED6	ee ff	5
LDHX #*opr*	Load H:X from M	H:X ← (M:M + 1)	0	–	–	↕	↕	–	IMM	45	ii jj	3
LDHX *opr*									DIR	55	dd	4
LDX #*opr*	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM	AE	ii	2
LDX *opr*									DIR	BE	dd	3
LDX *opr*									EXT	CE	hh ll	4
LDX *opr*,X									IX2	DE	ee ff	4
LDX *opr*,X									IX1	EE	ff	3
LDX ,X									IX	FE		2
LDX *opr*,SP									SP1	9EEE	ff	4
LDX *opr*,SP									SP2	9EDE	ee ff	5

Table 7-1. Instruction Set Summary (Sheet 5 of 7)

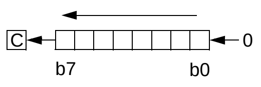
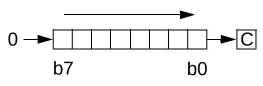
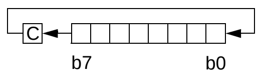
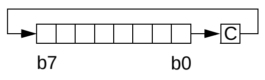
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X LSL *opr*,SP	Logical Shift Left (Same as ASL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
LSR *opr* LSRA LSRX LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr*,X+ MOV #*opr,opr* MOV X+,*opr*	Move	$(M)_{Destination} \leftarrow (M)_{Source}$ H:X ← (H:X) + 1 (IX+D, DIX+)	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X NEG *opr*,SP	Negate (Two's Complement)	M ← –(M) = \$00 – (M) A ← –(A) = \$00 – (A) X ← –(X) = \$00 – (X) M ← –(M) = \$00 – (M) M ← –(M) = \$00 – (M)	↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap A	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		3
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X ORA *opr*,SP ORA *opr*,SP	Inclusive OR A and M	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); SP ← (SP) – 1	–	–	–	–	–	–	INH	87		2
PSHH	Push H onto Stack	Push (H); SP ← (SP) – 1	–	–	–	–	–	–	INH	8B		2
PSHX	Push X onto Stack	Push (X); SP ← (SP) – 1	–	–	–	–	–	–	INH	89		2
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X ROL *opr*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5
ROR *opr* RORA RORX ROR *opr*,X ROR ,X ROR *opr*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5

Table 7-1. Instruction Set Summary (Sheet 6 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
RSP	Reset Stack Pointer	SP ← $FF	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	↕	INH	80		7
RTS	Return from Subroutine	SP ← SP + 1; Pull (PCH) SP ← SP + 1; Pull (PCL)	–	–	–	–	–	–	INH	81		4
SBC #*opr* SBC *opr* SBC *opr* SBC *opr*,X SBC *opr*,X SBC ,X SBC *opr*,SP SBC *opr*,SP	Subtract with Carry	A ← (A) – (M) – (C)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A2 B2 C2 D2 E2 F2 9EE2 9ED2	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask	I ← 1	–	–	1	–	–	–	INH	9B		2
STA *opr* STA *opr* STA *opr*,X STA *opr*,X STA ,X STA *opr*,SP STA *opr*,SP	Store A in M	M ← (A)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	B7 C7 D7 E7 F7 9EE7 9ED7	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
STHX *opr*	Store H:X in M	(M:M + 1) ← (H:X)	0	–	–	↕	↕	–	DIR	35	dd	4
STOP	Enable $\overline{\text{IRQ}}$ Pin; Stop Oscillator	I ← 0; Stop Oscillator	–	–	0	–	–	–	INH	8E		1
STX *opr* STX *opr* STX *opr*,X STX *opr*,X STX ,X STX *opr*,SP STX *opr*,SP	Store X in M	M ← (X)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X SUB *opr*,SP SUB *opr*,SP	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		9
TAP	Transfer A to CCR	CCR ← (A)	↕	↕	↕	↕	↕	↕	INH	84		2
TAX	Transfer A to X	X ← (A)	–	–	–	–	–	–	INH	97		1

Table 7-1. Instruction Set Summary (Sheet 7 of 7)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
TPA	Transfer CCR to A	A ← (CCR)	–	–	–	–	–	–	INH	85		1
TST *opr* TSTA TSTX TST *opr*,X TST ,X TST *opr*,SP	Test for Negative or Zero	(A) – \$00 or (X) – \$00 or (M) – \$00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	3 1 1 3 2 4
TSX	Transfer SP to H:X	H:X ← (SP) + 1	–	–	–	–	–	–	INH	95		2
TXA	Transfer X to A	A ← (X)	–	–	–	–	–	–	INH	9F		1
TXS	Transfer H:X to SP	(SP) ← (H:X) – 1	–	–	–	–	–	–	INH	94		2

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

7.8 Opcode Map

The opcode map is provided in **Table 7-2**.

Table 7-2. Opcode Map

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

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

*Pre-byte for stack pointer indexed instructions

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

Section 8. Error Detection Central Processor Unit (EDC)

8.1 Contents

8.2 Introduction

This section describes the error detection central processor unit (EDC, version 0) which is based on the CPU08, version A. The M68HC08 CPU is an enhanced and fully object-code-compatible version of the M68HC05 CPU. The error detection CPU is a duplicate CPU which runs synchronously with the primary CPU. The EDC operates on all data in parallel with the primary CPU, executing all of the same instruction flows. In addition to the duplicate CPU, the EDC contains circuits for comparing address and data with the primary CPU. If either the address or data values do not match, an internal reset is asserted.

8.3 Features

Features of the EDC include:

- Separate compare of data, low address byte, high address byte
- Control register bits for forcing mismatch conditions for testing
- Output pin active on EDC reset only; cleared by software or power-on reset (POR)

8.4 Functional Description

The EDC performs a cycle-by-cycle compare of address and data information. In the event that the address or the data generated by the duplicate CPU does not match the value on the internal bus, then an internal reset occurs. This reset functions in the same manner as all other internal resets in the HC08 (see **10.4.2 Active Resets from Internal Sources**) except that the $\overline{\text{FLT}}$ pin is asserted. This pin remains active low until it is cleared by software.

The block diagram for the EDC, in **Figure 8-1**, shows that the compare is done in three segments:

- Address low (lower byte of address)
- Address high (upper byte of address)
- Data

These circuits perform a compare of the address and data values generated by the duplicate CPU with the values on the internal bus on every machine cycle.

On a read cycle, the duplicate CPU latches data provided by the data buffers. No data compare happens on read cycles since the EDC data bus is driven with the same value as the system data bus. On write cycles, the data buffers are three-stated and the data circuit compares the value being written by the duplicate CPU to that being written by the primary CPU.

NOTE: *It is necessary to initialize all CPU internal registers before any of those registers are output from the CPU by an instruction or by stacking. This* *is because the internal registers such as the accumulator are not initialized by reset so the EDC CPU and the primary CPU may have different values upon power-up.*

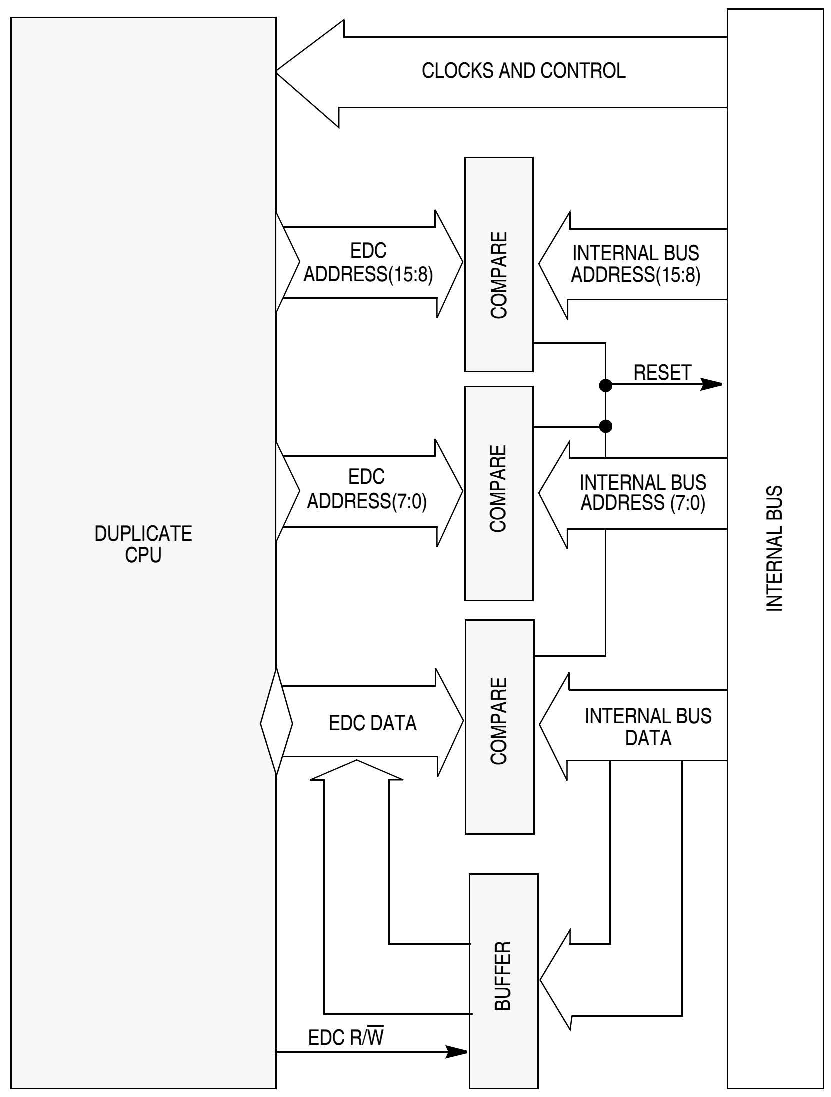


Figure 8-1. EDC Block Diagram

8.5 $\overline{\text{FLT}}$ Pin

This pin is unaffected by all resets except for the error detect reset. On the occurrence of a mismatch between the CPU and the EDC on any address or data, the pin is driven to a logic 0 and remains low until it is cleared by storing a 0 to bit 7of the EDC control register.

8.6 EDC Control Register

The control register for the EDC is shown in **Figure 8-2**. The bits in this register are used to simulate an error condition in each of the compare circuits. When a write to the register is done, it is followed by an internal reset if the data to any of the three control bits is a logic 1.

Address: $FE09

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	R	0	0	R	FDB	FABH	FABL
Write:	SFLTP		R	R				
Reset:	1		0	0	1	0	0	0

R = Reserved

Figure 8-2. EDC Control Register (EDCR)

SFLTP — Set FLT Pin

This bit always reads as a 0. A logic 1 is set on the occurrence of an error detect reset. It is cleared by writing a 0.

FDB — Force Data Bus Bit

Writing a 1 to this bit forces the output of the data bus compare to generate a reset. The bit will be set on failure to compare data.

1 = Generate reset

0 = No reset

FABH — Force Address Bus High Bit

Writing a 1 to this bit forces the output of the address bus high compare to generate a reset. The bit will be set on failure to compare high address.

1 = Generate reset
0 = No reset

FABL — Force Address Bus Low Bit

Writing a 1 to this bit forces the output of the address bus low compare to generate a reset. The bit will be set on failure to compare low address.

1 = Generate reset
0 = No reset

Section 9. Clock Generator Module (CGM)

9.1 Contents

9.2 Introduction

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

9.3 Features

Features of the CGM include:

- PLL with output frequency in integer multiples of the crystal reference
- Programmable hardware voltage-controlled oscillator (VCO) for low-jitter operation
- Automatic bandwidth control mode for low-jitter operation
- Automatic frequency lock detector
- CPU interrupt on entry or exit from locked condition

9.4 Functional Description

The CGM consists of three major 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 9-1 shows the structure of the CGM. **Figure 9-2** shows a summary of the input/output (I/O) registers.

9.4.1 Crystal Oscillator Circuit

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

CGMXCLK can be used by other modules which require precise timing for operation. The duty cycle of CGMXCLK is not guaranteed to be 50 percent 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.

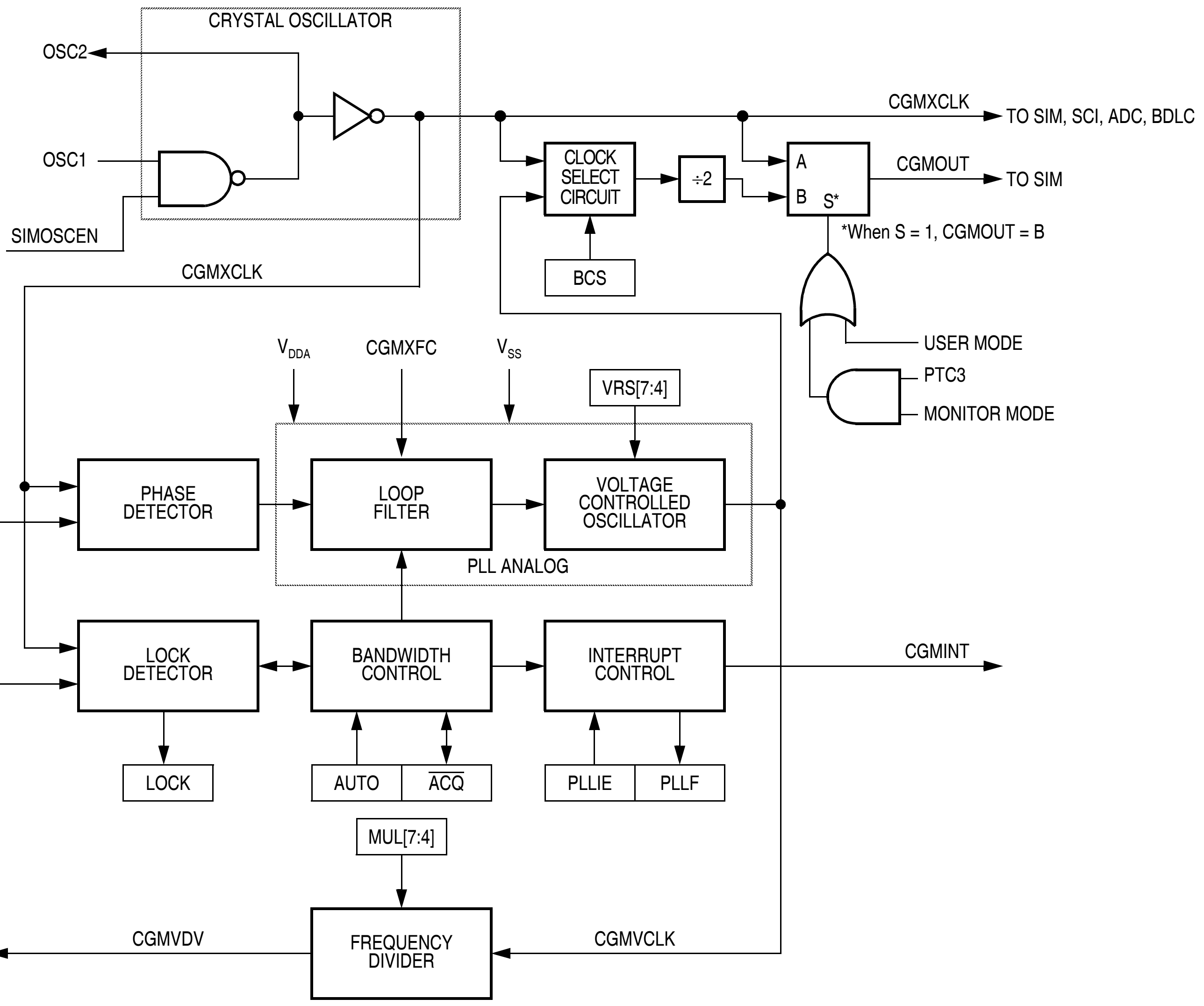


Figure 9-1. CGM Block Diagram

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

R	= Reserved

Figure 9-2. CGM I/O Register Summary

9.4.2 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. Refer to **23.9 CGM Operating Conditions** for operating frequencies while reading this section.

9.4.2.1 Circuits

The PLL consists of these circuits:

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

The operating range of the VCO is programmable for a wide range of frequencies and for maximum immunity to external noise, including supply and CGMXFC noise. For maximum immunity guidelines, refer to:

- *Designing for Electromagnetic Compatibility (EMC) with HCMOS Microcontrollers*, Freescale document order number AN1050/D
- *Designing for Electromagnetic Compatibility with Single-Chip Microcontrollers*, Freescale document order number AN1263

These application notes on electromagnetic compatibility are available from local Freescale sales offices.) 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 f_{NOM}.

CGMXCLK is the PLL reference clock which runs at the crystal frequency, f_{XCLK},

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 (see **9.4.2.4 Programming the PLL**). The divider's output is the VCO feedback clock, CGMVDV, running at a frequency equal to f_{VCLK}/N. See **23.9 CGM Operating Conditions** for more information.

The phase detector then compares the VCO feedback clock (CGMVDV) with the reference clock (CGMXCLK). A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the dc voltage on the external capacitor connected to CGMXFC, based on the width and direction of the correction pulse. The filter can make fast or slow corrections, depending on its mode, described in **9.4.2.2 Acquisition and Tracking Modes**. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

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

9.4.2.2 Acquisition and Tracking Modes

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

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO (see **23.11 CGM Acquisition/Lock Time Information**). 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 **9.6.2 PLL Bandwidth Control Register**).
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO (see **23.11 CGM Acquisition/Lock Time Information**). PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source (see **9.4.3 Base Clock Selector Circuit**). The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{\text{ACQ}}$ bit is set.

9.4.2.3 Manual and Automatic PLL Bandwidth Modes

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

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. (See **9.6.2 PLL Bandwidth Control Register**.) 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 **9.4.3 Base Clock Selector Circuit**.) If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. See **9.7 Interrupts** for information and precautions on using interrupts.

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

- The $\overline{\text{ACQ}}$ bit is a read-only indicator of the mode of the filter. See **9.6.2 PLL Bandwidth Control Register** and **9.4.2.2 Acquisition and Tracking Modes**.
- The $\overline{\text{ACQ}}$ bit is set when the VCO frequency is within a certain tolerance, Δ_{TRK}, and is cleared when the VCO frequency is out of a certain tolerance, Δ_{UNT}. See **9.10 Acquisition/Lock Time Specifications** 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 **9.10 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 **9.6.1 PLL Control Register**.)

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

These conditions apply when in manual mode:

- $\overline{\text{ACQ}}$ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the $\overline{\text{ACQ}}$ bit must be clear.
- Before entering tracking mode ($\overline{\text{ACQ}}$ = 1), software must wait a given time, t_{ACQ} (see **9.10 Acquisition/Lock Time Specifications**), after turning on the PLL by setting PLLON in the PLL control register (PCTL).

- Software must wait a given time, t_{AL}, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.

9.4.2.4 Programming the PLL

Use this procedure to program the PLL:

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

   Example: f_{BUSDES} = 8 MHz

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

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

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

3. Using a reference frequency, f_{XCLK}, equal to the crystal frequency, calculate the VCO frequency multiplier, N.

NOTE: *The round function means that the result is rounded to the nearest integer.*

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

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

4. Calculate the VCO frequency, f_{VCLK}.

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

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

5. Calculate the bus frequency, f_{BUS}, and compare f_{BUS} with f_{BUSDES}. If the calculated f_{BUS} is not within the tolerance limits of your application, select another f_{BUSDES} or another f_{XCLK}.

$$f_{BUS} = \frac{f_{VCLK}}{4}$$

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

6. Using the value 4.9152 MHz for f_{NOM}, calculate the VCO linear range multiplier, L. The linear range multiplier controls the frequency range of the PLL.

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

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

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

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

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

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

For proper operation,

$$|f_{VRS} - f_{VCLK}| \le \frac{f_{NOM}}{2}$$

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

9.4.2.5 Special Programming Exceptions

The programming method described in **9.4.2.4 Programming the PLL** does not account for two possible exceptions. A value of 0 for N or L is meaningless when used in the equations given. To account for these exceptions:

- A 0 value for N is interpreted exactly the same as a value of 1.
- A 0 value for L disables the PLL and prevents its selection as the source for the base clock. (See **9.4.3 Base Clock Selector Circuit**.)

9.4.3 Base Clock Selector Circuit

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

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

9.4.4 CGM External Connections

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

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in **Figure 9-3**. **Figure 9-3** shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

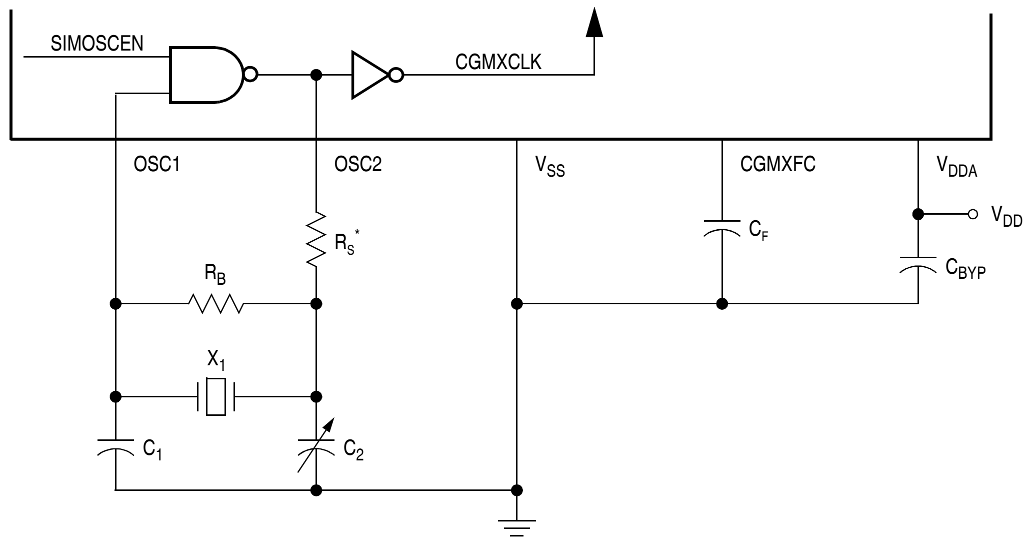
1. Crystal, X_1
2. Fixed capacitor, C_1
3. Tuning capacitor, C_2, can also be a fixed capacitor
4. Feedback resistor, R_B
5. Series resistor, R_S, optional

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

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

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

Routing should be done with great care to minimize signal cross talk and noise. See **9.10 Acquisition/Lock Time Specifications** for routing information and more information on the filter capacitor's value and its effects on PLL performance.



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

Figure 9-3. CGM External Connections

9.5 I/O Signals

This section describes the CGM input/output (I/O) signals.

9.5.1 Crystal Amplifier Input Pin (OSC1)

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

9.5.2 Crystal Amplifier Output Pin (OSC2)

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

9.5.3 External Filter Capacitor Pin (CGMXFC)

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

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

9.5.4 Analog Power Pin (V_{DDA})

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

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

9.5.5 Oscillator Enable Signal (SIMOSCEN)

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

9.5.6 Crystal Output Frequency Signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal, f_{XCLK}, and comes directly from the crystal oscillator circuit. **Figure 9-3** shows only the logical relation of CGMXCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of CGMXCLK is unknown and may depend on the crystal and other external factors. Also, the frequency and amplitude of CGMXCLK can be unstable at startup.

9.5.7 CGM Base Clock Output (CGMOUT)

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

9.5.8 CGM CPU Interrupt (CGMINT)

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

9.6 CGM Registers

These registers control and monitor operation of the CGM:

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

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

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

R	= Reserved

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

Figure 9-4. CGM I/O Register Summary

9.6.1 PLL Control Register

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

Address: $001C

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

R = Reserved

Figure 9-5. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

This read/write bit enables the PLL to generate an interrupt request when the LOCK bit toggles, setting the PLL flag, PLLF. When the AUTO bit in the PLL bandwidth control register (PBWC) is clear, PLLIE cannot be written and reads as logic 0. Reset clears the PLLIE bit.

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL 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 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. Clear the PLLF bit by reading the PLL control register. Reset clears the PLLF bit.

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

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

PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). (See **9.4.3 Base Clock Selector Circuit**.) Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See **9.4.3 Base Clock Selector Circuit**.) Reset and the STOP instruction clear the BCS bit.

1 = 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* ***9.4.3 Base Clock Selector Circuit****.)*

PCTL[3:0] — Reserved bits

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

9.6.2 PLL Bandwidth Control Register

The PLL bandwidth control register (PBWC):

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

In manual operation, forces the PLL into acquisition or tracking mode.

Address: $001D

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

R	= Reserved

Figure 9-6. 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 0 and has no meaning. Reset clears the LOCK bit.

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

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

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

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

1 = Tracking mode
0 = Acquisition mode

XLD — Crystal Loss Detect Bit

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

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

The crystal loss detect function works only when the BCS bit is set, selecting CGMVCLK to drive CGMOUT. When BCS is clear, XLD always reads as logic 0.

1 = Crystal reference not active
0 = Crystal reference active

PBWC3–PBWC0 — Reserved for Test

These bits enable test functions not available in user mode. To ensure software portability from development systems to user applications, software should write 0s to PBWC3–PBWC0 whenever writing to PBWC.

9.6.3 PLL Programming Register

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

Address: $001E

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

Figure 9-7. PLL Programming Register (PPG)

MUL7–MUL4 — Multiplier Select Bits

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier, N. (See **9.4.2 Phase-Locked Loop Circuit (PLL)**.) A value of $0 in the multiplier select bits configures the modulo feedback divider the same as a value of $1. Reset initializes these bits to $6 to give a default multiply value of 6.

Table 9-1. VCO Frequency Multiplier (N) Selection

MUL7–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).*

VRS7–VRS4 — VCO Range Select Bits

These read/write bits control the hardware center-of-range linear multiplier L, which controls the hardware center-of-range frequency, f_{VRS}. (see **9.4.2 Phase-Locked Loop Circuit (PLL)**.) VRS7–VRS4 cannot be written when the PLLON bit in the PLL control register (PCTL) is set. (See **9.4.2.5 Special Programming Exceptions**.) A value of \$0 in the VCO range selects bits disables the PLL and clears the BCS bit in the PCTL. (See **9.4.3 Base Clock Selector Circuit** and **9.4.2.5 Special Programming Exceptions** for more information.) Reset initializes the bits to \$6 to give a default range multiply value of 6.

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

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

9.7 Interrupts

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

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

NOTE: *Software can select 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.*

9.8 Special Modes

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

9.8.1 Wait Mode

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

9.8.2 Stop Mode

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

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

9.9 CGM During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **10.8.3 SIM Break Flag Control Register**.)

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

To protect the PLLF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write the PLL control register during the break state without affecting the PLLF bit.

9.10 Acquisition/Lock Time Specifications

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

9.10.1 Acquisition/Lock Time Definitions

Typical control systems refer to the acquisition time or lock time as the reaction time, 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 percent 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 percent 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 percent of the 100-kHz step input.

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

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

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

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

9.10.2 Parametric Influences on Reaction Time

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

The most critical parameter which affects the PLL reaction times is the reference frequency, f_{XCLK}. 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 an external crystal frequency.

Another critical parameter is the external filter capacitor. The PLL modifies the voltage on the VCO by adding or subtracting charge from this capacitor. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. (See **9.10.3 Choosing a Filter Capacitor**.)

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

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

9.10.3 Choosing a Filter Capacitor

As described in **9.10.2 Parametric Influences on Reaction Time**, the external filter capacitor, C_F, is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency, f_{RDV}, and supply voltage, V_{DD}. The value of the capacitor, therefore, must 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. Refer to **9.4.2 Phase-Locked Loop Circuit (PLL)** for the value of f_{XCLK} and **23.10 CGM Component Information** for the value of C_{Fact}.

$$C_F = C_{Fact}\left(\frac{V_{DDA}}{f_{XCLK}}\right)$$

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 can become unstable. Also, always choose a capacitor with a tight tolerance ($\pm$20 percent or better) and low dissipation.

Section 10. System Integration Module (SIM)

10.1 Contents

10.2 Introduction

This section describes the system integration module (SIM), which supports up to 24 external and/or internal interrupts. The SIM is a system state controller that coordinates CPU and exception timing. Together with the central processor unit (CPU), the SIM controls all MCU activities. A block diagram of the SIM is shown in **Figure 10-1**. **Figure 10-2** is a summary of the SIM input/output (I/O) registers.

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

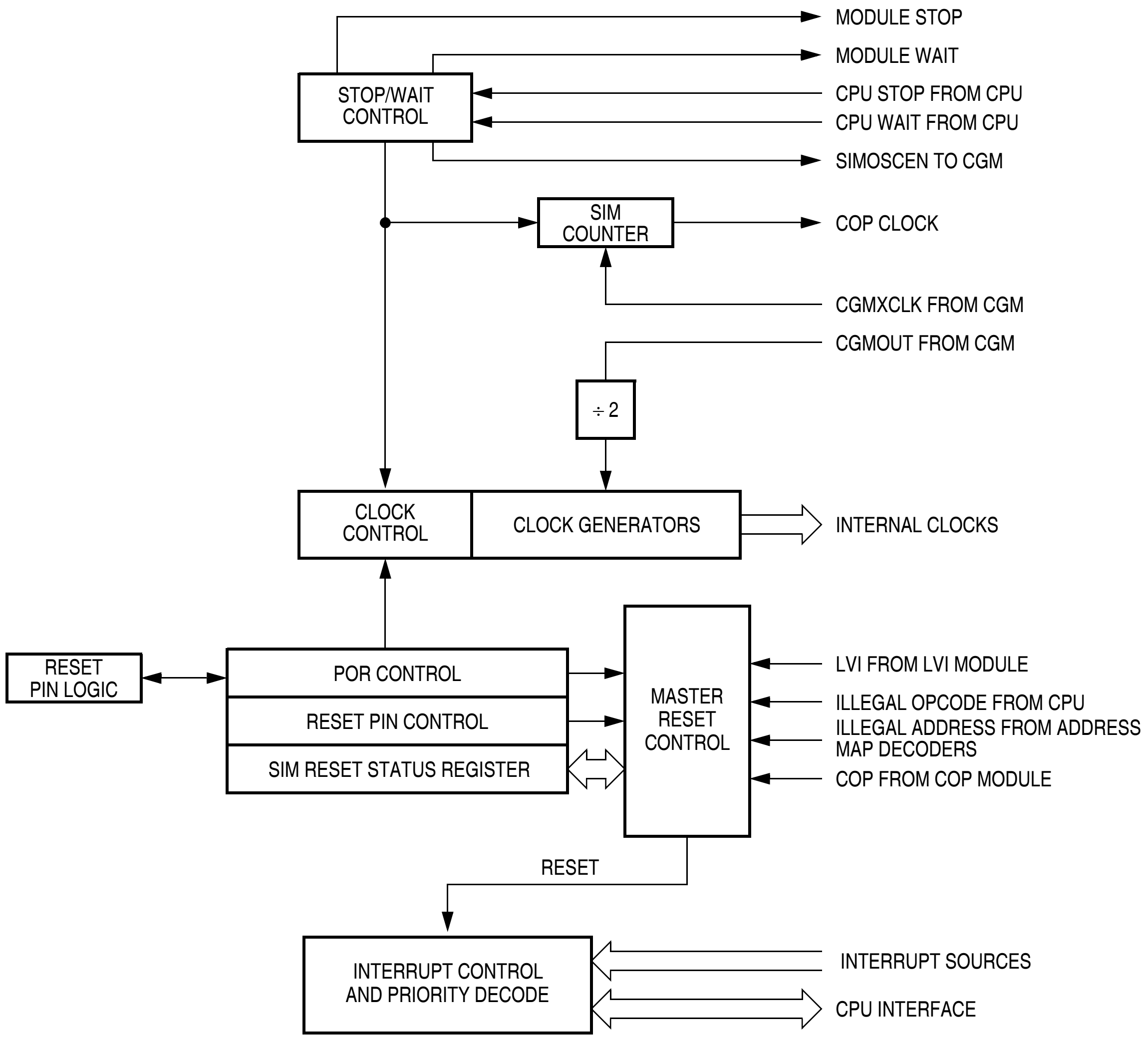


Figure 10-1. SIM Block Diagram

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

R = Reserved U = Unaffected

Figure 10-2. SIM I/O Register Summary

Table 10-1 shows the internal signal names used in this section.

Table 10-1. Signal Name Conventions

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

10.3 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 10-3**. This clock can come from either an external oscillator or from the on-chip PLL. (See **Section 9. Clock Generator Module (CGM)**.)

10.3.1 Bus Timing

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

10.3.2 Clock Startup from POR or LVI Reset

When the power-on reset (POR) module or the low-voltage inhibit (LVI) 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.

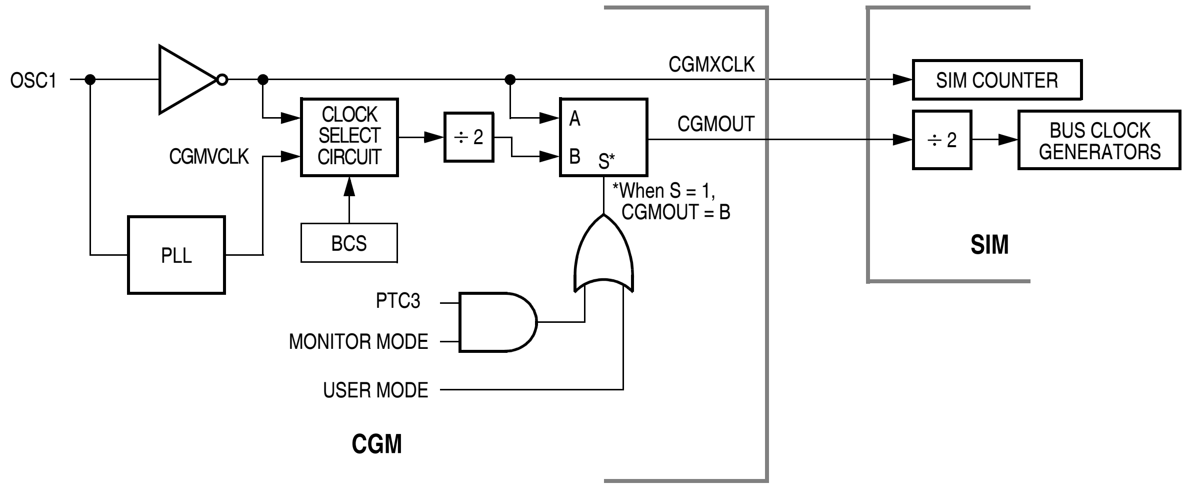


Figure 10-3. CGM Clock Signals

10.3.3 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 **10.7.2 Stop Mode**.)

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

10.4 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
- Error detect CPU (EDC)

Each of these resets produces the vector \$FFFE–FFFF (\$FEFE–FEFF in monitor mode) and asserts 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 **10.5 SIM Counter**), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See **10.8 SIM Registers**.)

10.4.1 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 10-2** for details. **Figure 10-4** shows the relative timing.

Table 10-2. PIN Bit Set Timing

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

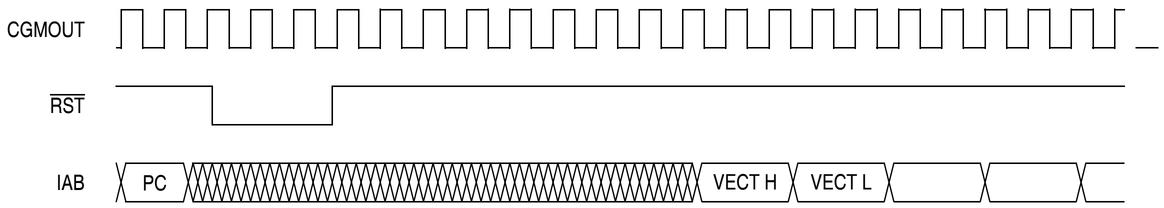


Figure 10-4. External Reset Timing

10.4.2 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 10-5**. An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR. (See **Figure 10-6**.) Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the $\overline{\text{RST}}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ shown in **Figure 10-5**.

The COP reset is asynchronous to the bus clock.

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

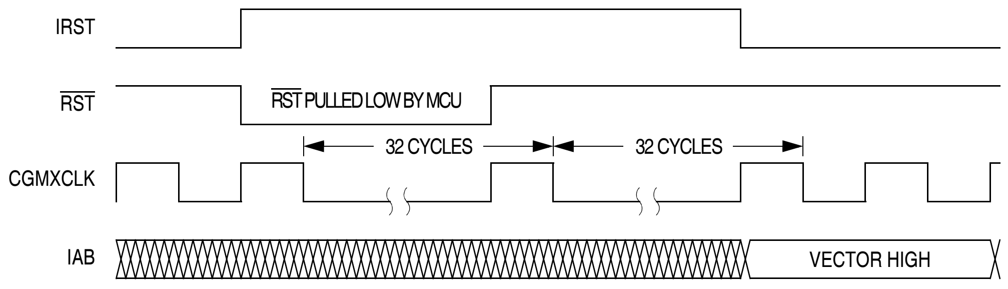


Figure 10-5. Internal Reset Timing

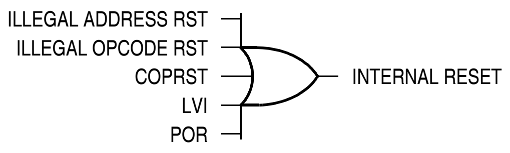


Figure 10-6. Sources of Internal Reset

10.4.2.1 Power-On Reset

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

At power-on, these events occur:

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

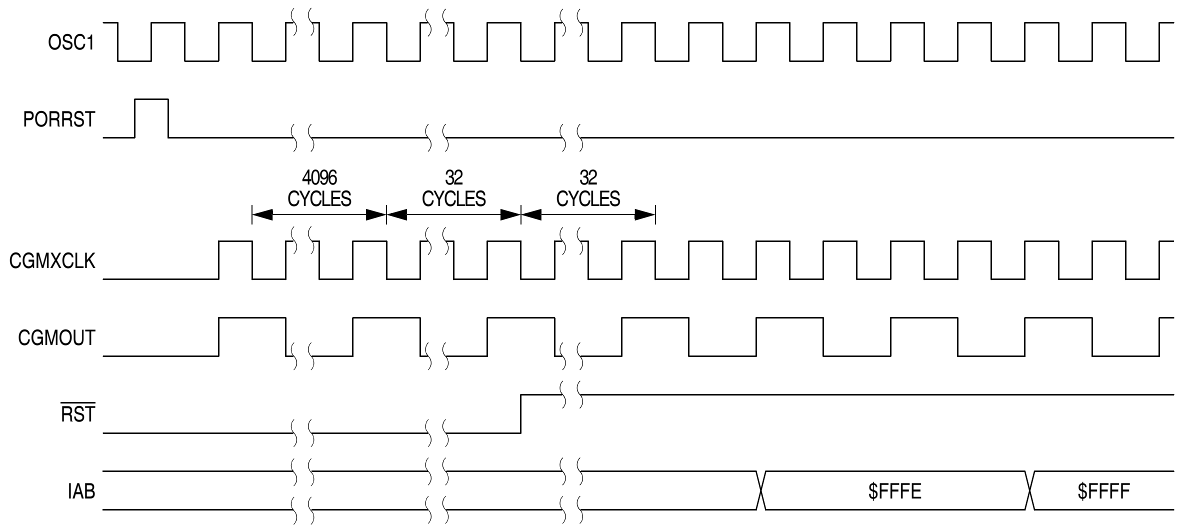


Figure 10-7. POR Recovery

10.4.2.2 Computer Operating Properly (COP) Reset

The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR) if the COPD bit in the CONFIG register ($001F) is at logic 0. (See **Section 14. Computer Operating Properly (COP)**.)

10.4.2.3 Illegal Opcode Reset

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

NOTE: *A $9E opcode (pre-byte for SP instructions) followed by an $8E opcode (stop instruction) generates a stop mode recovery reset.*

If the stop enable bit, STOP, in the CONFIG register ($001F) is logic 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset.

10.4.2.4 Illegal Address Reset

An opcode fetch from an unmapped address generates an illegal address reset. The SIM verifies that the CPU is fetching an opcode prior to asserting the ILAD bit in the SIM reset status register (SRSR) and resetting the MCU. A data fetch from an unmapped address does not generate a reset.

10.4.2.5 Low-Voltage Inhibit (LVI) Reset

The low-voltage inhibit (LVI) module 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 0. The $\overline{\text{RST}}$ pin will be held low until the SIM counts 4096 CGMXCLK cycles after V_{DD} rises above V_{LVIR}. Another 64 CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. (See **Section 11. Low-Voltage Inhibit (LVI)**.)

10.5 SIM Counter

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

10.5.1 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.

10.5.2 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 CONFIG register ($001F). If the SSREC bit is a logic 1, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles. This is ideal for applications using canned oscillators that do not require long startup times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

10.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See **10.7.2 Stop Mode** for details.) The SIM counter is free-running after all reset states. (See **10.4.2 Active Resets from Internal Sources** for counter control and internal reset recovery sequences.)

10.6 Program Exception Control

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

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

10.6.1 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 return-from-interrupt (RTI) instruction recovers the CPU register contents from the stack so that normal processing can resume. **Figure 10-8** shows interrupt entry timing.
Figure 10-10 shows interrupt recovery timing.

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced or the I bit is cleared.
(See **Figure 10-9**.)

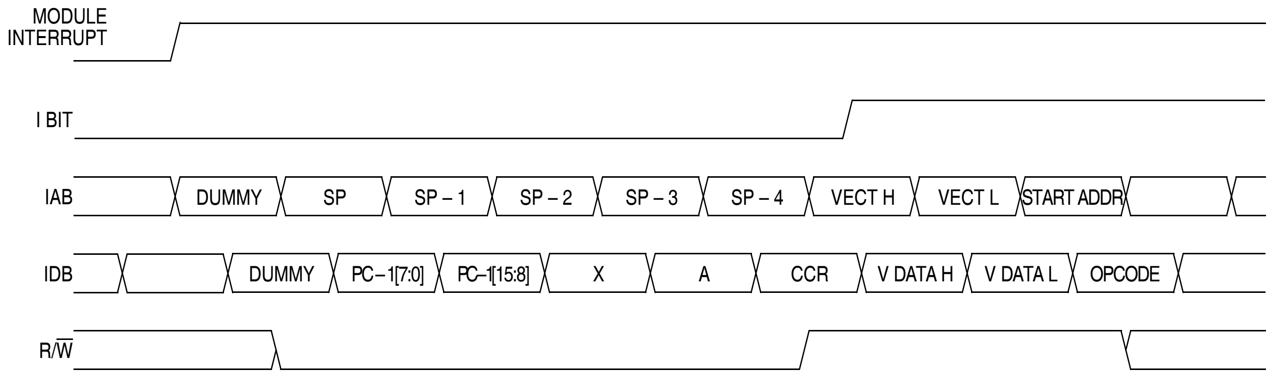


Figure 10-8. Hardware Interrupt Entry

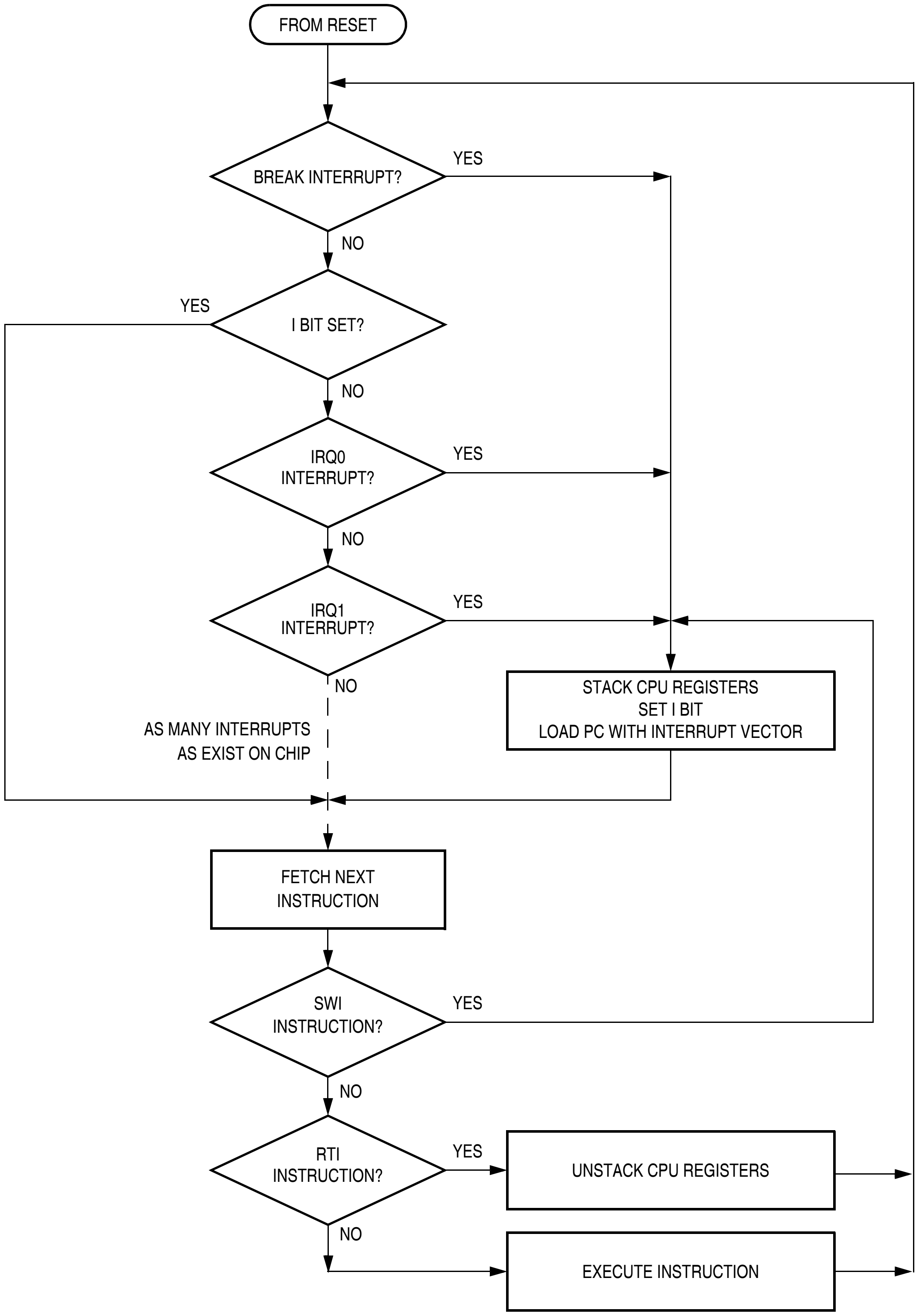


Figure 10-9. Interrupt Processing

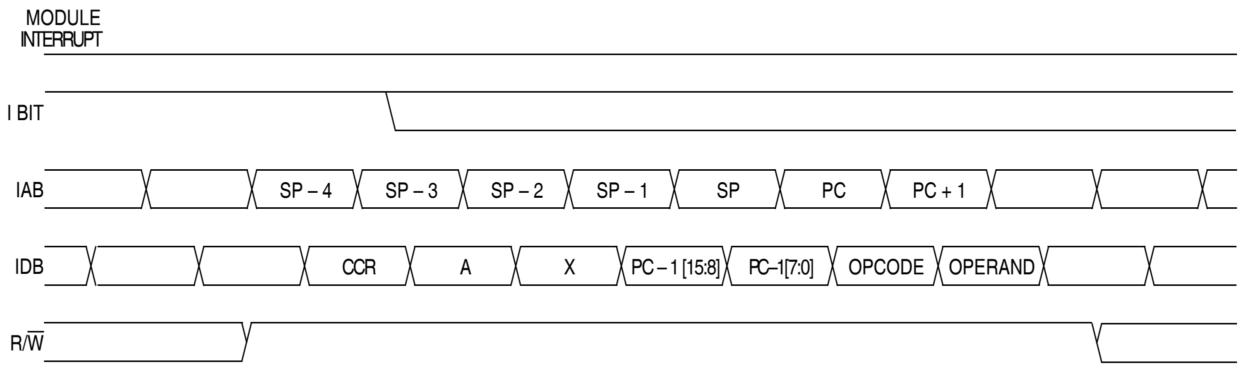


Figure 10-10. Hardware Interrupt Recovery

10.6.1.1 Hardware Interrupts

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

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. **Figure 10-11** demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

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

NOTE: *To maintain compatibility with the M68HC05, M6805, and M146805 Families, the H register is not pushed on the stack during interrupt entry. If the interrupt service routine modifies the H register or uses the indexed addressing mode, software should save the H register and then restore it prior to exiting the routine.*

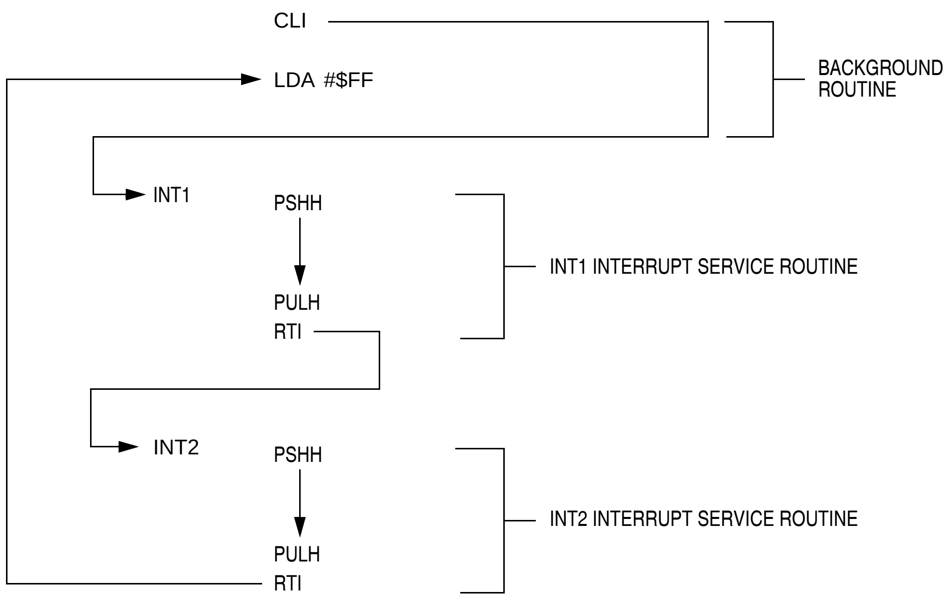

CLI
LDA #$FF
BACKGROUND ROUTINE
INT1
PSHH
PULH
RTI
INT1 INTERRUPT SERVICE ROUTINE
INT2
PSHH
PULH
RTI
INT2 INTERRUPT SERVICE ROUTINE

Figure 10-11. Interrupt Recognition Example

10.6.1.2 SWI Instruction

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

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

10.6.2 Reset

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

10.6.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See **Section 12. 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 the break state affects each module.

10.6.4 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 to protect flags from being cleared by properly initializing the break clear flag enable bit (BCFE) in the SIM break flag control register (SBFCR). (See **10.8.3 SIM Break Flag Control Register**.)

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 2-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 usual.

10.7 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 in this section. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing interrupts to occur.

10.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while one set of peripheral clocks continues to run. **Figure 10-12** shows the timing for wait mode entry.

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

Wait mode also can be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the configuration register ($001F) is logic 0, then the computer operating properly module (COP) is enabled and remains active in wait mode.

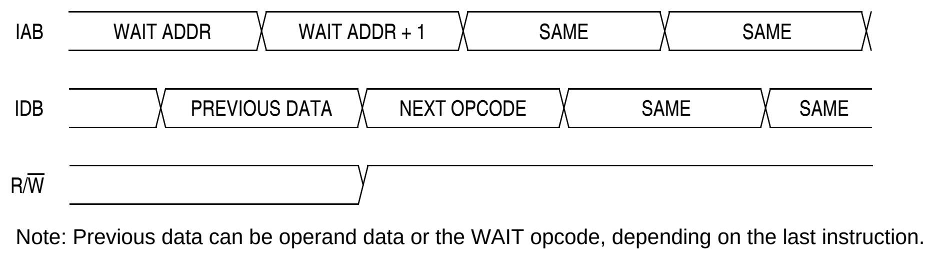

IAB
WAIT ADDR
WAIT ADDR + 1
SAME
SAME
IDB
PREVIOUS DATA
NEXT OPCODE
SAME
SAME
R/$\overline{W}$
Note: Previous data can be operand data or the WAIT opcode, depending on the last instruction.

Figure 10-12. Wait Mode Entry Timing

Figure 10-13 and **Figure 10-14** show the timing for WAIT recovery.

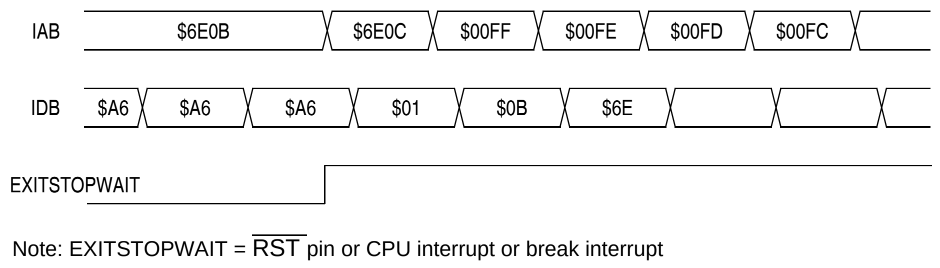

IAB
$6E0B
$6E0C
$00FF
$00FE
$00FD
$00FC
IDB
$A6
$A6
$A6
$01
$0B
$6E
EXITSTOPWAIT
Note: EXITSTOPWAIT = $\overline{RST}$ pin or CPU interrupt or break interrupt

Figure 10-13. Wait Recovery from Interrupt or Break

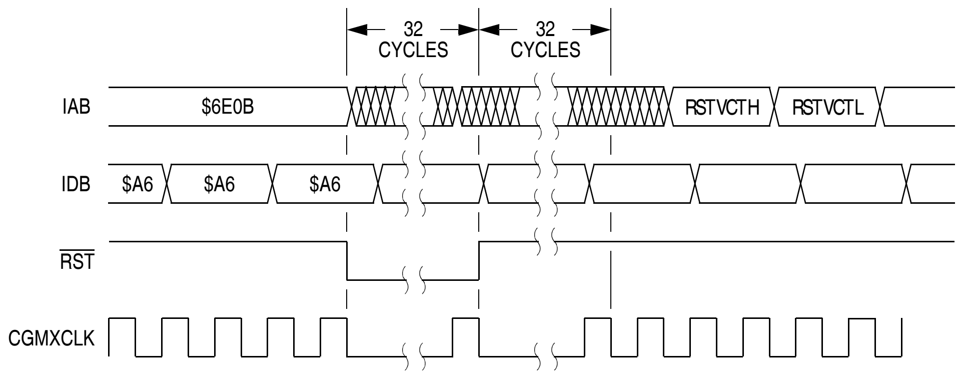


Figure 10-14. Wait Recovery from Internal Reset

10.7.2 Stop Mode

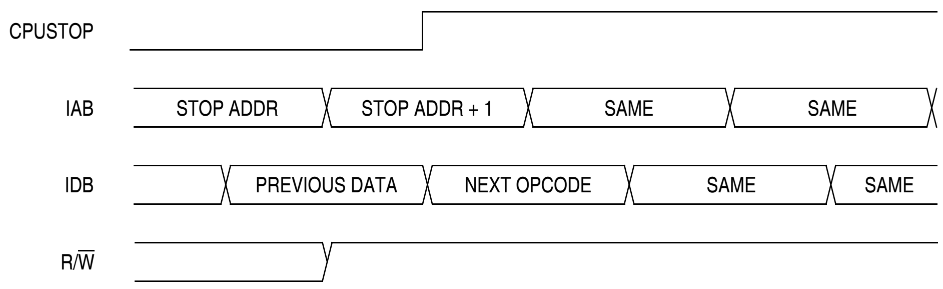
In stop mode, the SIM counter is reset and the system clocks are disabled. An interrupt request from a module can cause an exit from stop mode. Stacking for interrupts begins after the selected stop recovery time has elapsed. Reset or break also causes an exit from stop mode.

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 short stop recovery (SSREC) bit in the CONFIG register ($001F). 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 10-15** shows stop mode entry timing.



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

Figure 10-15. Stop Mode Entry Timing

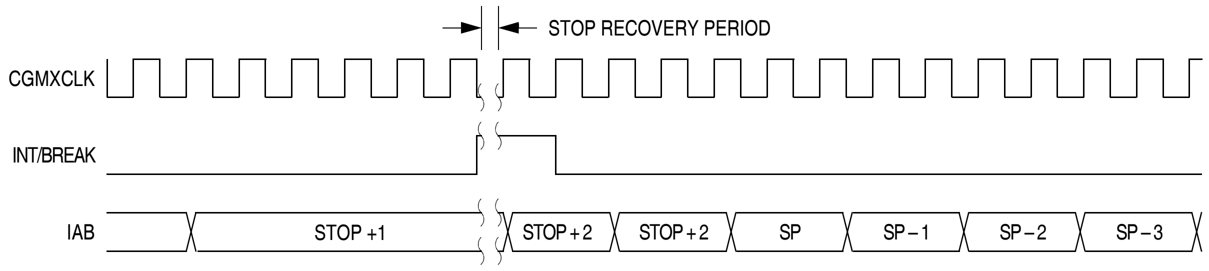


Figure 10-16. Stop Mode Recovery from Interrupt or Break

10.8 SIM Registers

The SIM has three memory mapped registers.

10.8.1 SIM Break Status Register

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

Address: $FE00

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

R	= Reserved

Figure 10-17. SIM Break Status Register (SBSR)

SBSW — SIM Break Stop/Wait Bit

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

1 = Stop mode or wait mode exited by break interrupt
0 = Stop mode or wait mode 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 0 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
```

10.8.2 SIM Reset Status Register

This register contains seven flags that show the source of the last reset. All flag bits are automatically cleared following a read of the register. The register is initialized on power-up as shown with the POR bit set and all other bits cleared. However, since the POR circuit releases the part at a lower voltage than the LVI trip point, the LVI bit may also be set when first reading the register after a power-up. Also, during a POR or any other internal reset, the $\overline{\text{RST}}$ pin is pulled low. After the pin is released, it will be sampled 32 XCLK cycles later. If the pin is not above a V_{IH} at that time then the PIN bit in the SRSR may be set in addition to whatever other bits are set.

Address: $FE01

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

R = Reserved   U = Unaffected

Figure 10-18. SIM Reset Status Register (SRSR)

POR — Power-On Reset Bit
- 1 = A POR has occurred.
- 0 = Read of SRSR

PIN — External Reset Bit
- 1 = An external reset has occurred since the last read of the SRSR.
- 0 = Read of SRSR

COP — Computer Operating Properly Reset Bit
- 1 = A COP reset has occurred since the last read of the SRSR.
- 0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit
- 1 = An illegal opcode reset has occurred since the last read of the SRSR.
- 0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (opcode fetches only)
- 1 = An illegal address reset has occurred since the last read of the SRSR.
- 0 = POR or read of SRSR

EDC — Error Detect CPU Reset Bit
- 1 = An EDC reset has occurred since the last read of the SRSR.
- 0 = POR or read of SRSR

LVI — Low-Voltage Inhibit Reset Bit
- 1 = An LVI reset has occurred since the last read of the SRSR.
- 0 = Read of SRSR

10.8.3 SIM Break Flag Control Register

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

Address: $FE03

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

R = Reserved

Figure 10-19. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

In some module registers, this read/write bit will enable software to clear status bits by accessing status registers only while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.This operation is important for modules with status bits which can be cleared only by being read. See the register descriptions in each module for additional details.

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

Section 11. Low-Voltage Inhibit (LVI)

11.1 Contents

11.2 Introduction

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

11.3 Features

Features of the LVI module include:

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

11.4 Functional Description

Figure 11-1 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWR, enables the LVI to monitor V_{DD} voltage. The LVI reset bit, LVIRST, enables the LVI module to generate a reset when V_{DD} falls below a voltage, LVI_{TRIPF}, and remains at or below that level for nine or more consecutive CPU cycles. LVISTOP enables the LVI module during stop mode. This will ensure that when the STOP instruction is implemented, the LVI will continue to monitor the voltage level on V_{DD}. LVIPWR, LVISTOP, and LVIRST are in the mask option register (MOR) ($001F). See **5.4 Mask Option Register**. Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, LVI_{TRIPR}. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset. (See **11.4.2 Forced Reset Operation**.) The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

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

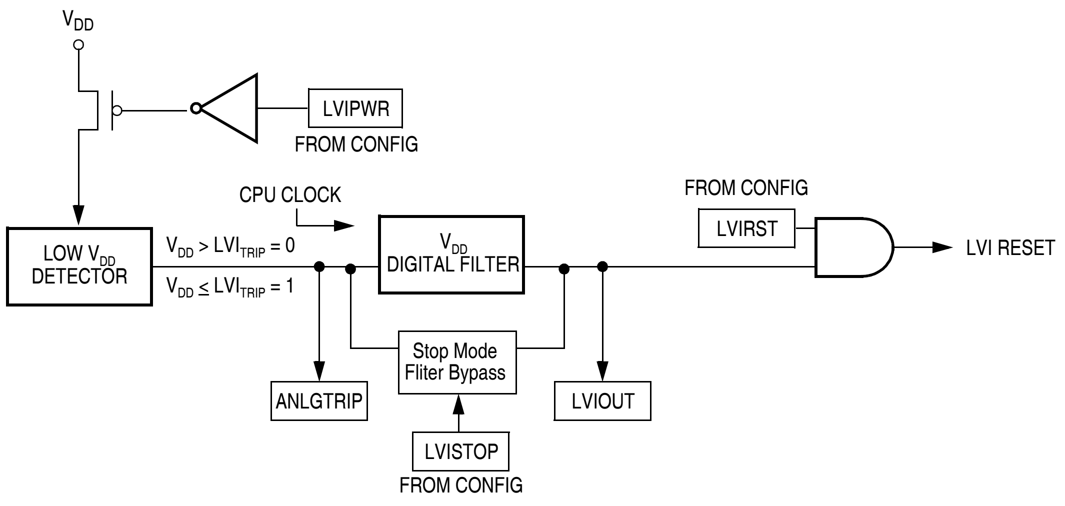


Figure 11-1. LVI Module Block Diagram

11.4.1 Polled LVI Operation

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

11.4.2 Forced Reset Operation

In applications that require V_{DD} to remain above the LVI_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls to the LVI_{TRIPF} level and remains at or below that level for nine or more consecutive CPU cycles. In the configuration register, the LVIPWR and LVIRST bits must be at logic 0 to enable the LVI module and to enable LVI resets.

11.4.3 False Reset Protection

The V_{DD} pin level is digitally filtered to reduce false resets due to power supply noise. For the LVI module to reset the MCU, V_{DD} must remain at or below the LVI_{TRIPF} level for nine or more consecutive CPU cycles. V_{DD} must be above LVI_{TRIPR} for only one CPU cycle to bring the MCU out of reset.

11.5 LVI Status Register

The LVI status register (LVISR) flags V_{DD} voltages below the LVI_{TRIPF} level.

Address $FE0F

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

R = Reserved

Figure 11-2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the LVI_{TRIPF} voltage for 32 to 40 CGMXCLK cycles. (See **Table 11-1**.) Reset clears the LVIOUT bit.

Table 11-1. LVIOUT Bit Indication

V_{DD} at Level:	For Number of CGMXCLK Cycles:	LVIOUT
$V_{DD} > LVI_{TRIPR}$	Any	0
$V_{DD} < LVI_{TRIPF}$	< 32 CGMXCLK cycles	0
$V_{DD} < LVI_{TRIPF}$	Between 32 & 40 CGMXCLK cycles	0 or 1
$V_{DD} < LVI_{TRIPF}$	> 40 CGMXCLK cycles	1
$LVI_{TRIPF} < V_{DD} < LVI_{TRIPR}$	Any	Previous value

11.6 LVI Interrupts

The LVI module does not generate interrupt requests.

11.7 Wait Mode

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

With the LVIPWR bit in the configuration register programmed to logic 0, the LVI module is active after a WAIT instruction.

With the LVIRST bit in the configuration register programmed to logic 0, the LVI module can generate a reset and bring the MCU out of wait mode.

11.8 Stop Mode

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

With the LVISTOP and LVIPWR bits in the configuration register programmed to a logic 0, the LVI module will be active after a STOP instruction. Due to the fact that the CPU clocks are disabled during stop mode, the LVI trip must bypass the digital filter to generate a reset and bring the MCU out of stop.

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

Section 12. Break Module (Break)

12.1 Contents

12.2 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.

12.3 Features

Features of the break module include:

- Accessible input/output (I/O) registers during the break interrupt
- CPU-generated break interrupts
- Software-generated break interrupts
- COP disabling during break interrupts

12.4 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).

These events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic 1 to the BRKA bit in the break status and control register.

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. **Figure 12-1** shows the structure of the break module.

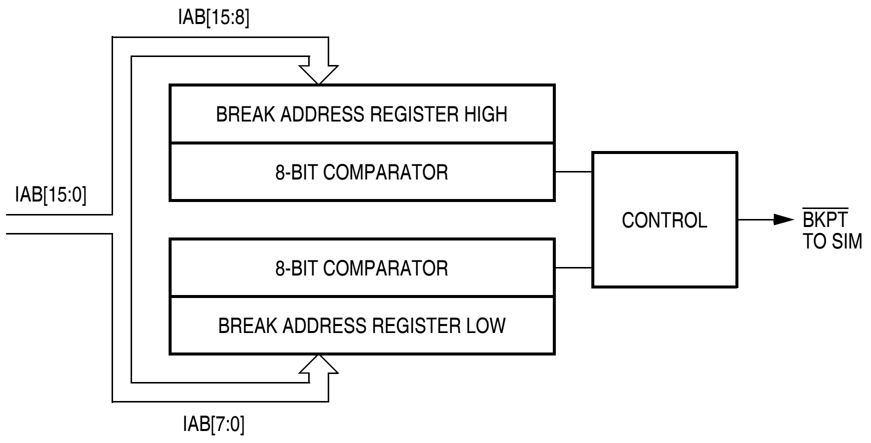

IAB[15:8]
BREAK ADDRESS REGISTER HIGH
8-BIT COMPARATOR
IAB[15:0]
CONTROL
$\overline{\text{BKPT}}$ TO SIM
8-BIT COMPARATOR
BREAK ADDRESS REGISTER LOW
IAB[7:0]

Figure 12-1. Break Module Block Diagram

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

R	= Reserved

Figure 12-2. Break I/O Register Summary

12.4.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether module status bits can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **10.8.3 SIM Break Flag Control Register** and the **Break Interrupts** subsection for each module.)

12.4.2 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.

12.4.3 TIM During Break Interrupts

A break interrupt stops the timer counter.

12.4.4 COP During Break Interrupts

The COP is disabled during a break interrupt when $V_{DD} + V_{HI}$ is present on the $\overline{RST}$ pin. For V_{HI}, see **23.5 DC Electrical Characteristics**.

12.5 Break Module Registers

Three registers control and monitor operation of the break module:

1. Break status and control register (BRKSCR)
2. Break address register high (BRKH)
3. Break address register low (BRKL)

12.5.1 Break Status and Control Register

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

Address: $FE0E

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

R = Reserved

Figure 12-3. Break Status and Control Register (BRKSCR)

BRKE — Break Enable Bit

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a logic 0 to bit 7. Reset clears the BRKE bit.

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

BRKA — Break Active Bit

This read/write status and control bit is set when a break address match occurs. Writing a logic 1 to BRKA generates a break interrupt. Clear BRKA by writing a logic 0 to it before exiting the break routine. Reset clears the BRKA bit.

1 = Break address match
0 = No break address match

12.5.2 Break Address Registers

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

Address: $FE0C

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

Figure 12-4. Break Address Register (BRKH)

Address: $FE0D

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

Figure 12-5. Break Address Register (BRKL)

12.6 Low-Power Modes

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

12.6.1 Wait Mode or Stop Mode

A break interrupt causes exit from wait mode or stop mode and sets the SBSW bit in the SIM break status register. (See **10.8 SIM Registers**.)

Section 13. Monitor ROM (MON)

13.1 Contents

13.2 Introduction

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

13.3 Features

Features of the monitor ROM include:

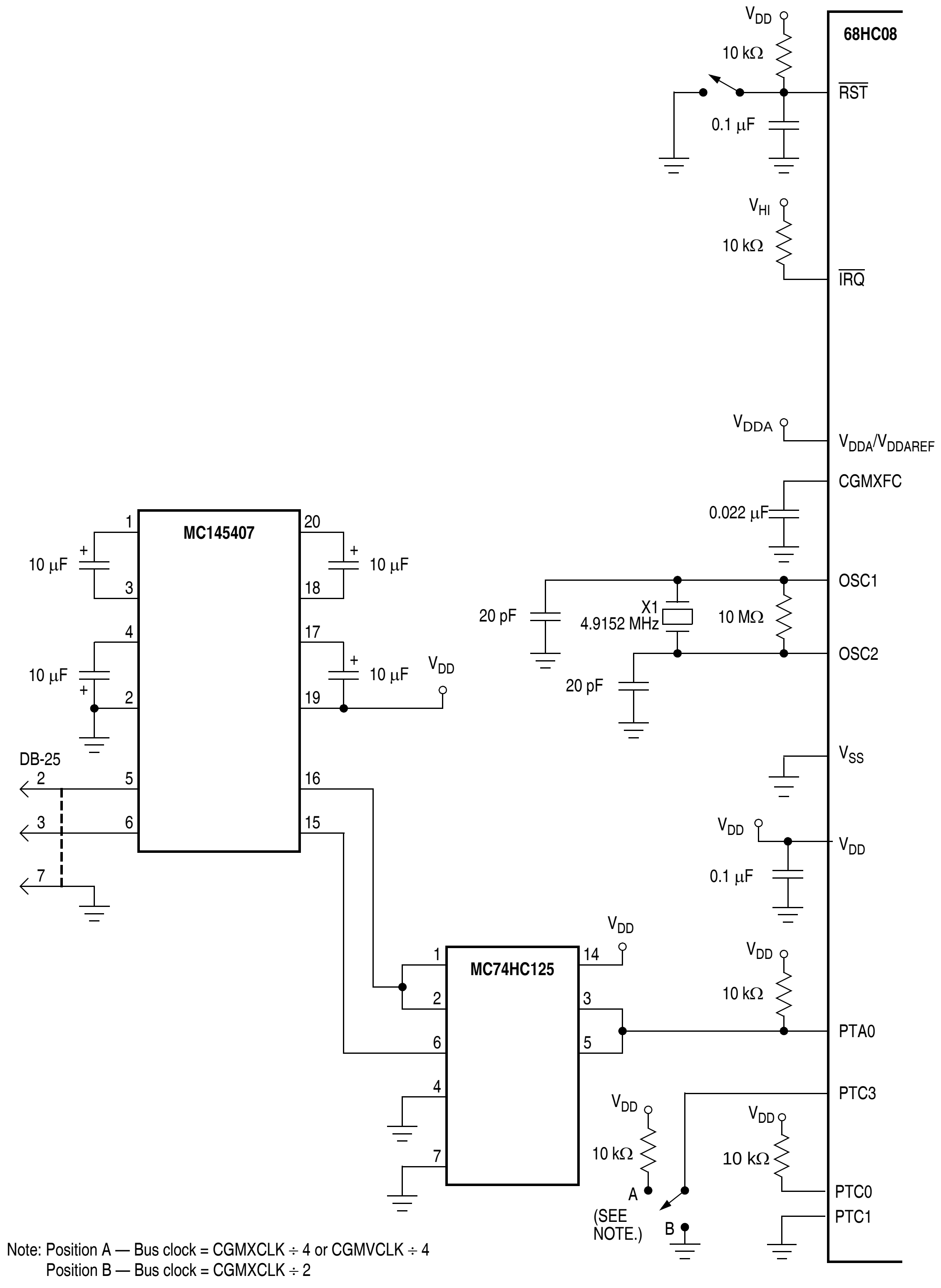
- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- 4800 baud–28.8 Kbaud communication with host computer
- Execution of code in RAM or ROM

13.4 Functional Description

Monitor ROM receives and executes commands from a host computer. **Figure 13-1** 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 MC68HC08QA24has a ROM security feature that requires proper procedures to be followed before the ROM can be accessed. Access to the ROM is denied to unauthorized users of customer-specified software.

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



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

Figure 13-1. Monitor Mode Circuit

13.4.1 Entering Monitor Mode

Table 13-1 shows the pin conditions for entering monitor mode.

Table 13-1. Mode Selection

$\overline{\text{IRQ}}$ Pin	PTC0 Pin	PTC1 Pin	PTA0 Pin	PTC3 Pin	Mode	CGMOUT	Bus Frequency
V_{DD} + V_{HI}(1)	1	0	1	1	Monitor	$\frac{\text{CGMXCLK}}{2}$ or $\frac{\text{CGMVCLK}}{2}$	$\frac{\text{CGMOUT}}{2}$
V_{DD} + V_{HI}(1)	1	0	1	0	Monitor	CGMXCLK	$\frac{\text{CGMOUT}}{2}$

1. For V_{HI}, see **23.5 DC Electrical Characteristics** and **23.2 Maximum Ratings**.

Enter monitor mode by either

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

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

Monitor mode uses alternate vectors for reset, SWI, and break interrupt. The alternate vectors are in the \$FE page instead of the \$FF page and allow code execution from the internal monitor firmware instead of user code. The COP module is disabled in monitor mode as long as V_{DD} + V_{HI} (see **23.5 DC Electrical Characteristics**) is applied to either the $\overline{\text{IRQ}}$ pin or the $\overline{\text{RST}}$ pin. (See **Section 10. 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 percent duty cycle at maximum bus frequency.*

Table 13-2 is a summary of the differences between user mode and monitor mode.

Table 13-2. 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 $\overline{IRQ}/V_{PP}$ pin while in monitor mode, the SIM asserts its COP enable output. The COP is a mask option enabled or disabled by the COPD bit in the configuration register. (See **23.5 DC Electrical Characteristics**.)

13.4.2 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See **Figure 13-2** and **Figure 13-3**.)

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

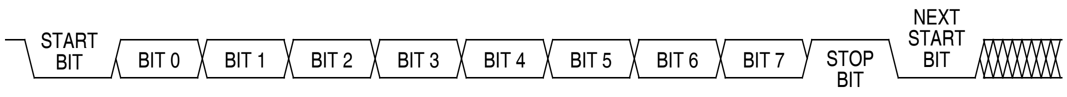


Figure 13-2. Monitor Data Format

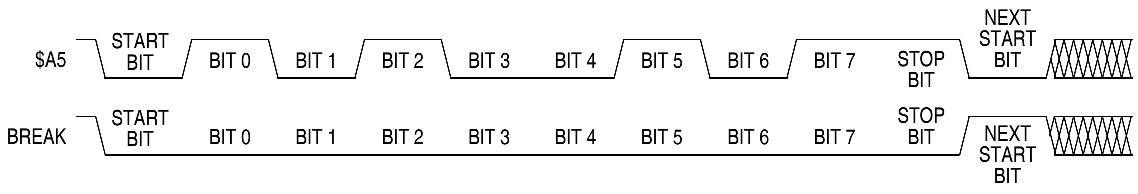


Figure 13-3. Sample Monitor Waveforms

13.4.3 Echoing

As shown in **Figure 13-4**, 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.

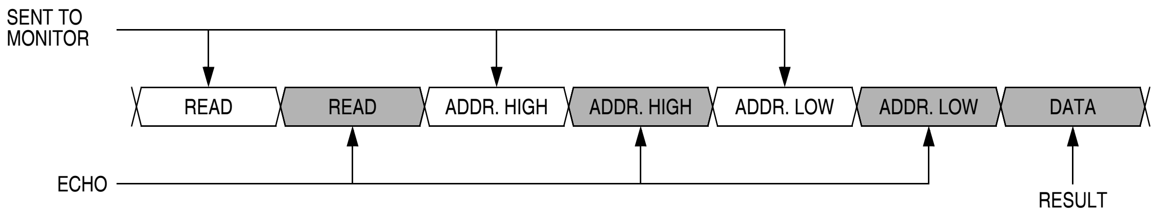


Figure 13-4. Read Transaction

13.4.4 Break Signal

A start bit followed by nine low bits is a break signal. (See **Figure 13-5**.) When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.

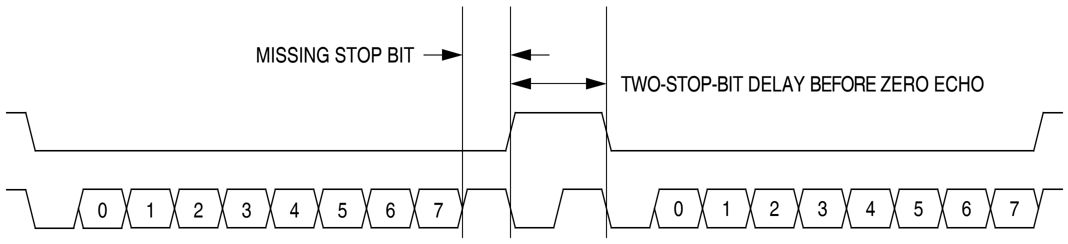


Figure 13-5. Break Transaction

13.4.5 Commands

The monitor ROM uses these commands:

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

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

Table 13-3. READ (Read Memory) Command

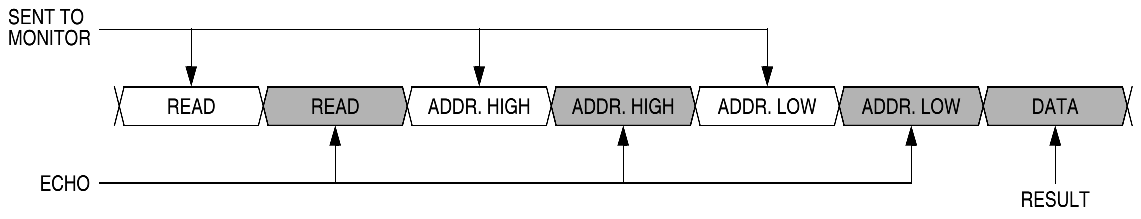
Description	Read byte from memory
Operand	Specifies 2-byte address in high byte:low byte order
Data Returned	Returns contents of specified address
Opcode	$4A
Command Sequence  SENT TO MONITOR READ READ ADDR. HIGH ADDR. HIGH ADDR. LOW ADDR. LOW DATA ECHO RESULT	

Table 13-4. WRITE (Write Memory) Command

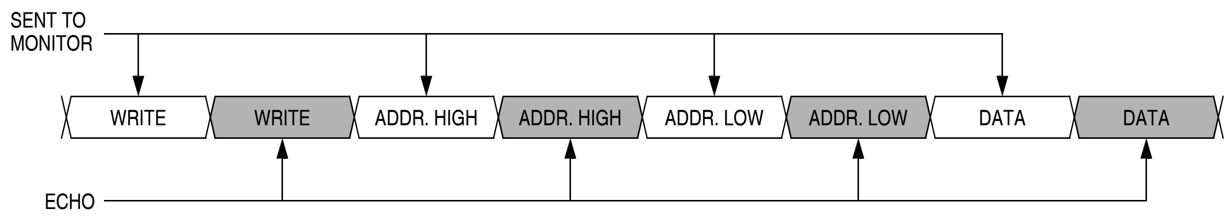
Description	Write byte to memory
Operand	Specifies 2-byte address in high byte:low byte order; low byte followed by data byte
Data Returned	None
Opcode	$49
Command Sequence 	

Table 13-5. IREAD (Indexed Read) Command

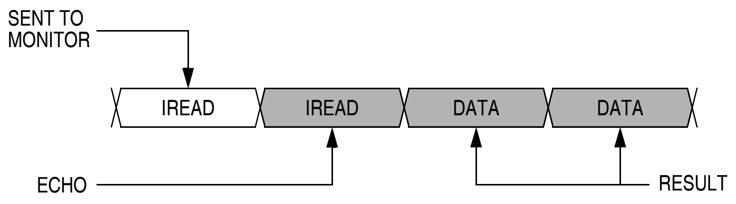
Description	Read next 2 bytes in memory from last address accessed
Operand	Specifies 2-byte address in high byte:low byte order
Data Returned	Returns contents of next two addresses
Opcode	$1A
Command Sequence 	

Table 13-6. IWRITE (Indexed Write) Command

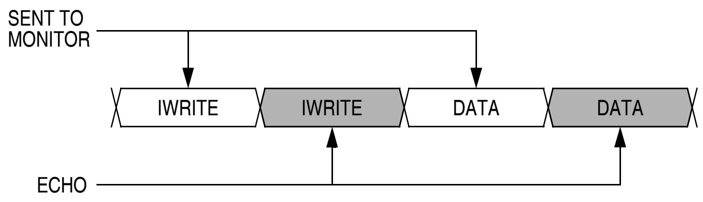
Description	Write to last address accessed + 1
Operand	Specifies single data byte
Data Returned	None
Opcode	$19
Command Sequence 	

Table 13-7. READSP (Read Stack Pointer) Command

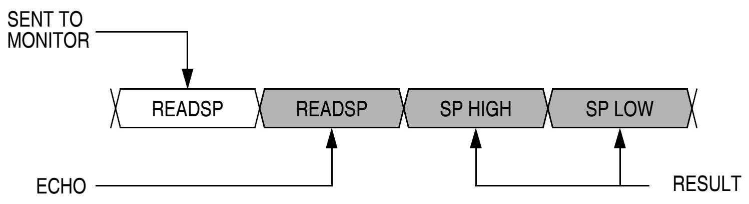
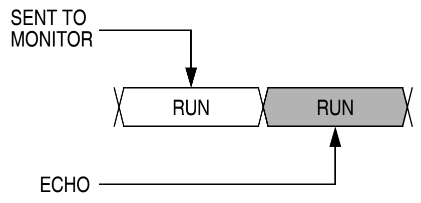
Description	Reads stack pointer
Operand	None
Data Returned	Returns stack pointer in high byte:low byte order
Opcode	$0C
Command Sequence 	

Table 13-8. RUN (Run User Program) Command

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

13.4.6 Baud Rate

With a 4.9152-MHz crystal and the PTC3 pin at logic 1 during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is at logic 0 during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL7–MUL4 bits in the PLL programming register (PPG). (See **Section 9. Clock Generator Module (CGM)**.)

Table 13-9. Monitor Baud Rate Selection

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

Section 14. Computer Operating Properly (COP)

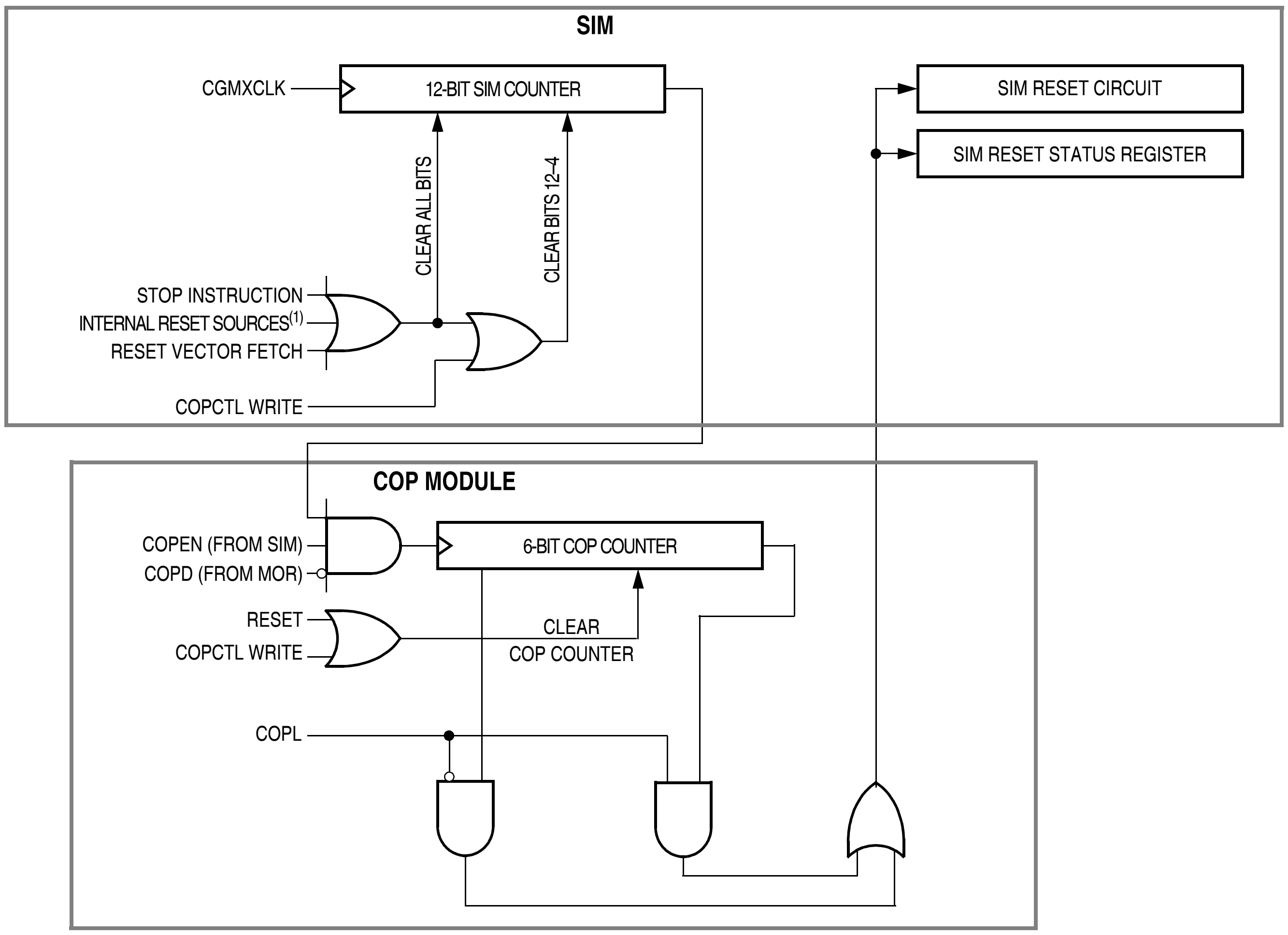
14.1 Contents

14.2 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.

14.3 Functional Description

Figure 14-1 shows the structure of the COP module.



Note:

1. See **10.4.2 Active Resets from Internal Sources**.

Figure 14-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by the 12-bit system integration module (SIM) counter. COP timeouts are determined strictly by the clock generator module (CGM) crystal oscillator clock signal (CGMXCLK), not the CGMOUT signal (see **Figure 9-1. CGM Block Diagram**).

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 COPS bit in the mask option register (MOR) ($001F) (see **5.4 Mask Option Register**). With a 4.194-MHz crystal and the COPS bit in the MOR ($001F) set to a logic 1, the COP timeout period is approximately 62.5 ms. Writing any value to location $FFFF before overflow occurs clears the COP counter, clears bits 12–4 of the SIM counter, and prevents reset. A CPU interrupt routine can be used to clear the COP.

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

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles and sets the COP bit in the SIM reset status register (SRSR) (see **10.8.2 SIM Reset Status Register**).

While the microcontroller is in monitor mode, the COP module is disabled if the $\overline{\text{RST}}$ pin or the $\overline{\text{IRQ}}$ pin is held at $V_{DD} + V_{HI}$ (see **23.5 DC Electrical Characteristics**). 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.*

14.4 I/O Signals

The following subsections describe the signals shown in **Figure 14-1**.

14.4.1 CGMXCLK

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

14.4.2 STOP Instruction

The STOP instruction clears the SIM counter.

14.4.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see **14.5 COP Control Register**) clears the COP counter and clears bits 12–4 of the SIM counter. Reading the COP control register returns the reset vector.

14.4.4 Internal Reset Resources

An internal reset clears the SIM counter and the COP counter. (See **10.4.2 Active Resets from Internal Sources**.)

14.4.5 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.

14.4.6 COPD (COP Disable)

The COPD bit reflects the state of the COP disable bit (COPD) in the MOR ($001F). This signal disables COP generated resets when asserted. (See **5.4 Mask Option Register**.)

14.4.7 COPS (COP Short Timeout)

The COPS bit selects the state of the COP short timeout bit (COPS) in the MOR (\$001F). Timeout periods can be $(2^{18} - 2^4)$ or $(2^{13} - 2^4)$ CGMXCLK cycles. (See **5.4 Mask Option Register**.)

14.5 COP Control Register

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

Address: \$FFFF

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

Figure 14-2. COP Control Register (COPCTL)

14.6 Interrupts

The COP does not generate CPU interrupt requests.

14.7 Monitor Mode

The COP is disabled in monitor mode when $V_{DD} + V_{HI}$ (see **23.5 DC Electrical Characteristics**) is present on the $\overline{\text{IRQ}}$ pin or on the $\overline{\text{RST}}$ pin.

14.8 Low-Power Modes

The following subsections describe the low-power modes.

14.8.1 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.

NOTE: *If the COP is enabled in wait mode, it must be periodically refreshed.*

14.8.2 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 MOR register ($001F) (see **5.4 Mask Option Register**) 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 0.

14.9 COP Module During Break Interrupts

The COP is disabled during a break interrupt when $V_{DD} + V_{HI}$ (see **23.5 DC Electrical Characteristics**) is present on the $\overline{RST}$ pin.

Section 15. External Interrupt ($\overline{\text{IRQ}}$)

15.1 Contents

15.2 Introduction

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

15.3 Features

Features include:

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

15.4 Functional Description

A logic 0 applied to the external interrupt pin can latch a CPU interrupt request. Figure 15-1 shows the structure of the IRQ module.

Interrupt signals on the $\overline{\text{IRQ}}$ pin are latched into the IRQ latch. An interrupt latch remains set until one of these 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 1 to the ACK bit clears the IRQ latch.
- Reset — A reset automatically clears both interrupt latches.

The external interrupt pin is falling-edge triggered and is software-configurable to be both falling-edge and low-level triggered. The MODE bit in the ISCR controls the triggering sensitivity of the $\overline{\text{IRQ}}$ pin.

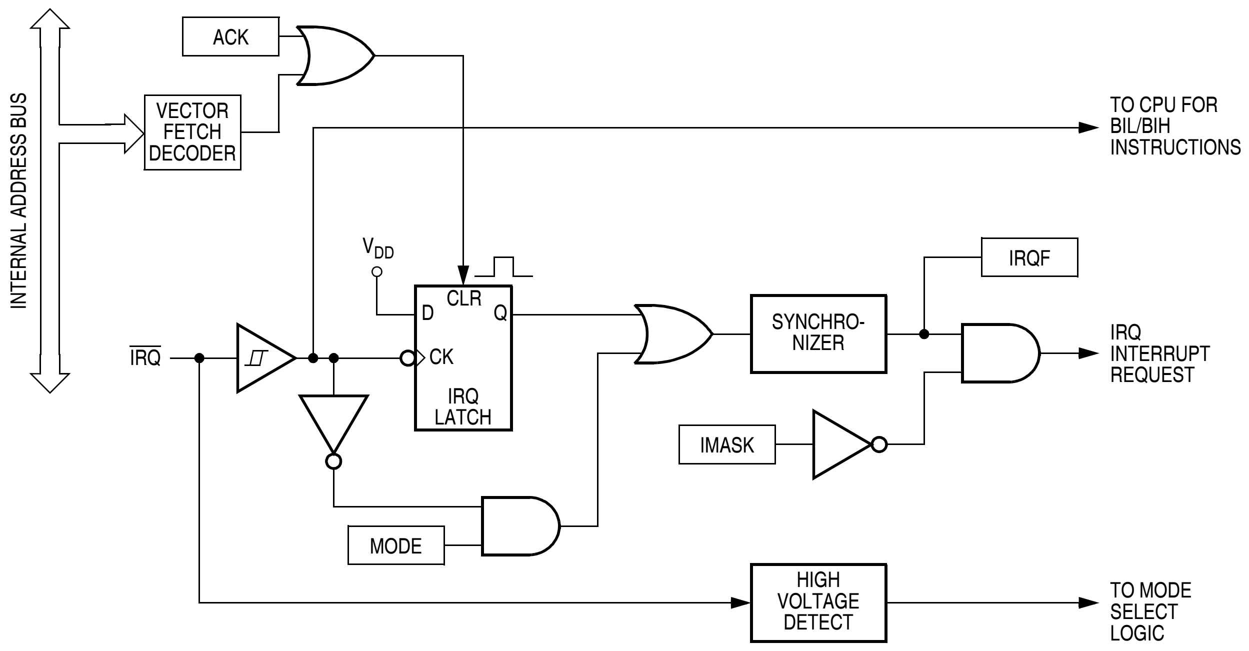


Figure 15-1. IRQ Block Diagram

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 these occur:

- Vector fetch or software clear
- Return of the interrupt pin to logic 1

The vector fetch or software clear may occur before or after the interrupt pin returns to logic 1. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

When set, the IMASK bit in the ISCR 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 15-2****.)*

15.5 $\overline{\text{IRQ}}$ Pin

A logic 0 on the $\overline{\text{IRQ}}$ pin can latch an interrupt request into the IRQ latch. A vector fetch, software clear, or reset clears the IRQ latch.

If the MODE bit is set, the $\overline{\text{IRQ}}$ pin is both falling-edge sensitive and low-level sensitive. With MODE set, both of these actions must occur to clear the IRQ latch:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACK bit in the interrupt status and control register (ISCR). The ACK bit is useful in applications that poll the $\overline{\text{IRQ}}$ pin and require software to clear the IRQ latch. Writing to the ACK bit can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the $\overline{\text{IRQ}}$ pin. A falling edge on $\overline{\text{IRQ}}$ that occurs after writing to the ACK bit latches another interrupt

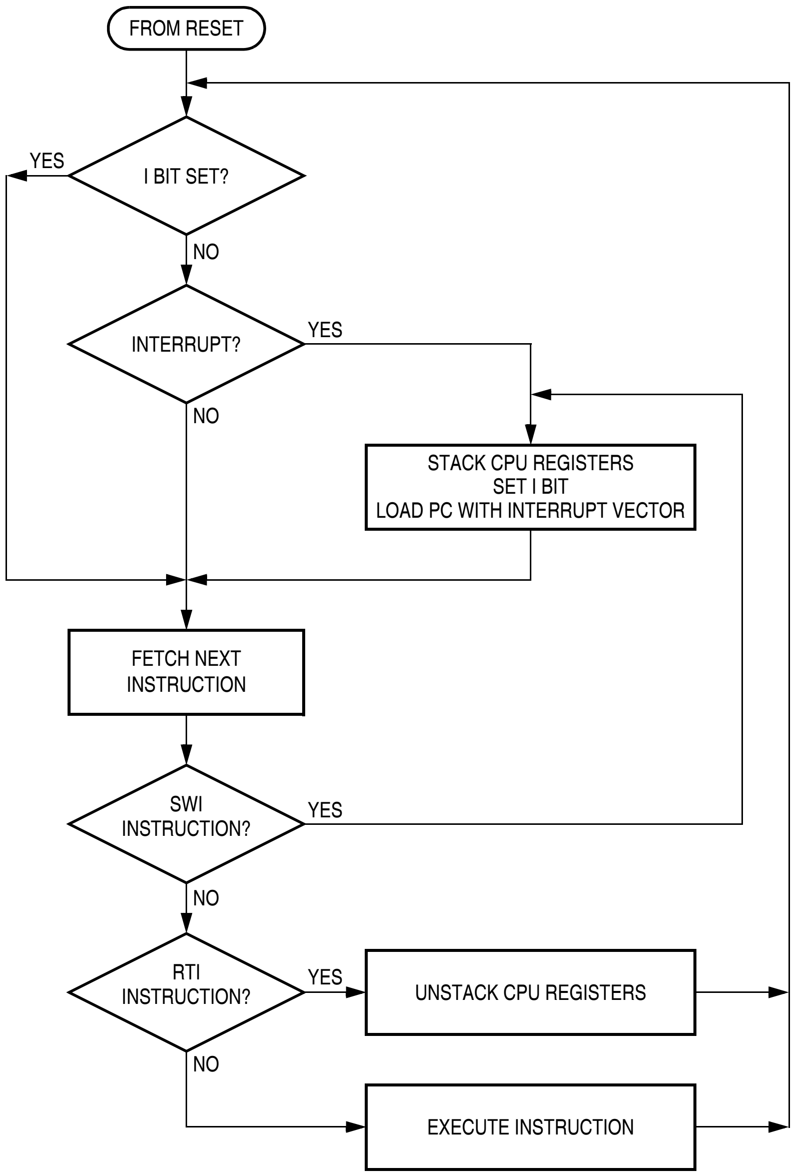


Figure 15-2. IRQ Interrupt Flowchart

request. If the IRQ mask bit, IMASK, is clear, the CPU loads the program counter with the vector address at locations \$FFFA and \$FFFB.

- Return of the $\overline{\text{IRQ}}$ pin to logic 1 — As long as the $\overline{\text{IRQ}}$ pin is at logic 0, the IRQ latch remains set.

The vector fetch or software clear and the return of the $\overline{\text{IRQ}}$ pin to logic 1 can occur in any order. The interrupt request remains pending as long as the $\overline{\text{IRQ}}$ pin is at logic 0. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

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

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

Use the branch if interrupt pin is high (BIH) or branch if interrupt pin is low (BIL) instruction to read the logic level on the $\overline{\text{IRQ}}$ pin.

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

15.6 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ interrupt latch can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latches during the break state. (See **10.8.3 SIM Break Flag Control Register**.)

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

To protect the latch during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), writing to the ACK bit in the IRQ status and control register during the break state has no effect on the IRQ latch.

15.7 IRQ Status and Control Register

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

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

Address: $001A

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

R = Reserved

Figure 15-3. IRQ Status and Control Register (ISCR)

IRQF — $\overline{\text{IRQ}}$ Flag

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

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

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

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

Writing a logic 1 to this read/write bit disables IRQ interrupt requests. Reset clears IMASK.
1 = $\overline{\text{IRQ}}$ interrupt requests disabled
0 = $\overline{\text{IRQ}}$ interrupt requests enabled

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

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

Section 16. Input/Output (I/O) Ports

16.1 Contents

16.2 Introduction

Forty bidirectional input/output (I/O) pins form six parallel ports. All I/O pins are programmable as inputs or outputs.

NOTE: *Connect any unused I/O pins to an appropriate logic level, either V_{DD} or V_{SS}. Although the I/O ports do not require termination for proper operation, termination reduces excess current consumption and the possibility of electrostatic damage.*

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read:	0	0	0	PTC4	PTC3	PTC2	PTC1	PTC0
		Write:	R	R	R					
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read:	0	0	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Write:	R	R						
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 16-1. I/O Port Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0006	Data Direction Register C (DDRC)	Read:	MCLKEN	0	0	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:		R	R					
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read:	0	0	DDRD5	DDRD4	DDRD3	DDR2	DDRD1	DDRD0
		Write:	R	R						
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE)	Read:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Write:								
		Reset:	Unaffected by reset							
$0009	Port F Data Register (PTF)	Read:	0	0	0	0	PTF3	PTF2	PTF1	PTF0
		Write:	R	R	R	R				
		Reset:	Unaffected by reset							
$000C	Data Direction Register E (DDRE)	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000D	Data Direction Register F (DDRF)	Read:	0	0	0	0	DDRF3	DDRF2	DDRF1	DDRF0
		Write:	R	R	R	R				
		Reset:	0	0	0	0	0	0	0	0

Figure 16-1. I/O Port Register Summary (Continued)

16.3 Port A

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

16.3.1 Port A Data Register

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

Address: $0000

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

Figure 16-2. Port A Data Register (PTA)

PTA[7:0] — Port A Data Bits

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

16.3.2 Data Direction Register A

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

Address: $0004

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

Figure 16-3. Data Direction Register A (DDRA)

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

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

1 = Corresponding port A pin configured as output

0 = Corresponding port A pin configured as input

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

Figure 16-4 shows the port A I/O logic.

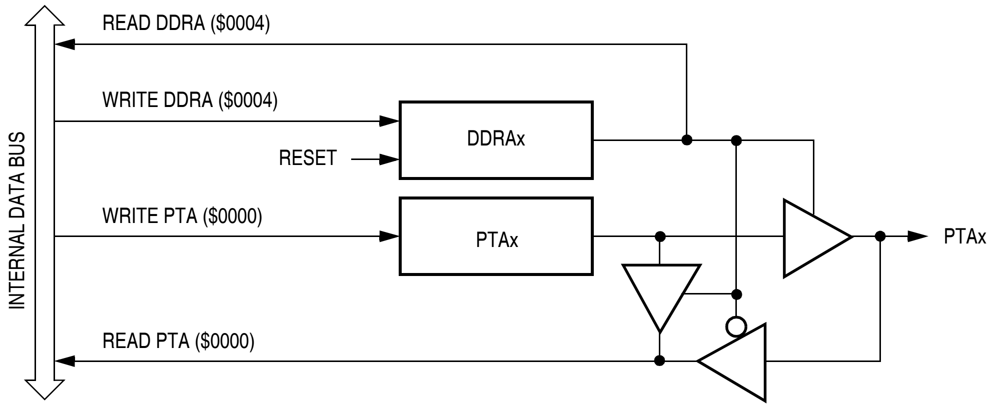


Figure 16-4. Port A I/O Circuit

When bit DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 16-1 summarizes the operation of the port A pins.

Table 16-1. Port A Pin Functions

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

X = don't care

Hi-Z = high impedance

1. Writing affects data register, but does not affect input.

16.4 Port B

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

16.4.1 Port B Data Register

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

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Reset:	Unaffected by reset							
Alternate Functions:	ATD7	ATD6	ATD5	ATD4	ATD3	ATD2	ATD1	ATD0

Figure 16-5. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

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

ATD[7:0] — ADC Channels

PTB7/ATD7–PTB0/ATD0 are eight of the 12 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 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See **Section 19. Analog-to-Digital Converter (ADC)**.) Data direction register B (DDRB) does not affect the data direction of port B pins that are being used by the ADC. However, the DDRB bits always determine whether reading port B returns to the states of the latches or logic 0.

16.4.2 Data Direction Register B

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

Address: $0005

	Bit 7	6	5	4	3	2	1	Bit 0
Read:/Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
Reset:	0	0	0	0	0	0	0	0

Figure 16-6. Data Direction Register B (DDRB)

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

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

1 = Corresponding port B pin configured as output

0 = Corresponding port B pin configured as input

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

Figure 16-7 shows the port B I/O logic.

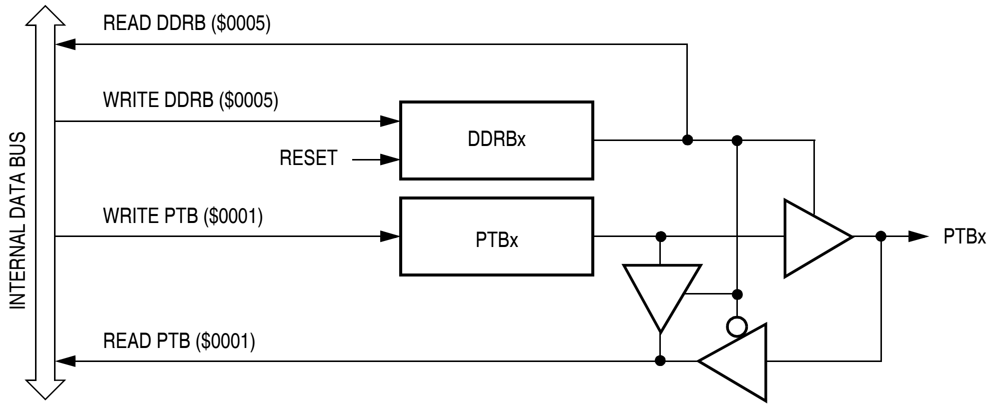


Figure 16-7. Port B I/O Circuit

When bit DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin, or logic 0 if that particular bit is in use by the ADC. The data latch can always be written, regardless of the state of its data direction bit. **Table 16-2** summarizes the operation of the port B pins.

Table 16-2. Port B Pin Functions

DDRB Bit	PTB Bit	Bit in Use by ADC	I/O Pin Mode	Accesses to DDRB	Accesses to PTB	
				Read/Write	Read	Write
0	X	No	Input, Hi-Z	DDRB[7:0]	Pin	PTB[7:0][1]
1	X	No	Output	DDRB[7:0]	PTB[7:0]	PTB[7:0]
0	X	Yes	Input, Hi-Z	DDRB[7:0]	0	PTB[7:0][1]
1	X	Yes	Input, Hi-Z	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.

16.5 Port C

Port C is a 5-bit, general-purpose, bidirectional I/O port that shares one of its pins with the bus clock (MCLK).

16.5.1 Port C Data Register

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

Address: $0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	PTC4	PTC3	PTC2	PTC1	PTC0
Write:	R	R	R					
Reset:	Unaffected by reset							
Alternate Functions:						MCLK		

R = Reserved

Figure 16-8. Port C Data Register (PTC)

PTC[4:0] — Port C Data Bits

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

MCLK — T12 System Bus Clock Bit

The bus clock (MCLK) is driven out of PTC2 when enabled by the MCLKEN bit in PTCDDR7.

16.5.2 Data Direction Register C

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

Address: $0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MCLKEN	0	0	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Write:		R	R					
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 16-9. Data Direction Register C (DDRC)

MCLKEN — MCLK Enable Bit

This read/write bit enables MCLK to be an output signal on PTC2. If MCLK is enabled, PTC2 is under the control of MCLKEN. Reset clears this bit.

1 = MCLK output enabled
0 = MCLK output disabled

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

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

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

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

Figure 16-10 shows the port C I/O logic.

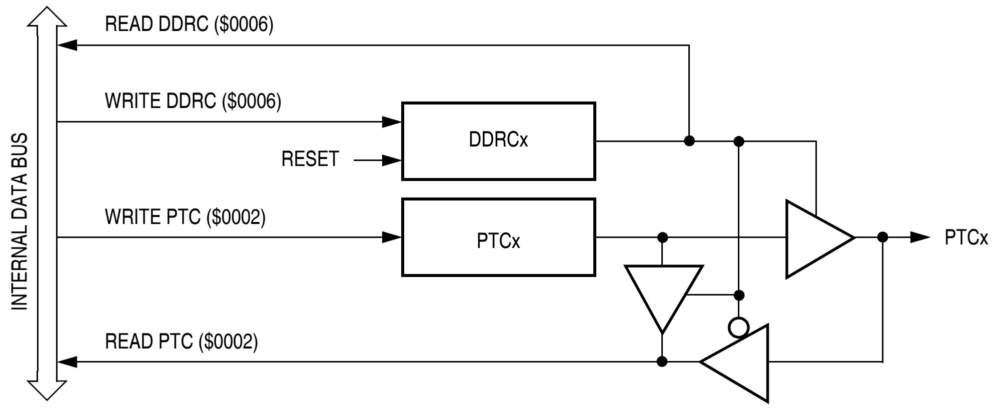


Figure 16-10. Port C I/O Circuit

When bit DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 16-3** summarizes the operation of the port C pins.

Table 16-3. Port C Pin Functions

DDRC Bit	PTC Bit	I/O Pin Mode	Accesses to DDRC	Accesses to PTC	
			Read/Write	**Read**	**Write**
[7] = 0	PTC2	Input, Hi-Z	DDRC[7]	0	PTC2
[7] = 1	PTC2	Output, MCLK	DDRC[7]	Data Latch	—
0	X	Input, Hi-Z	DDRC[4:0]	Pin	PTC[4:3,1:0][(1)]
1	X	Output	DDRC[4:0]	PTC[4:0]	PTC[4:3,1:0]

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

16.6 Port D

Port D is a 6-bit, special function I/O port that shares some of its pins with the analog-to-digital converter (ADC).

16.6.1 Port D Data Register

The port D data register contains a data latch for the seven port D pins.

Address: $0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Write:	R	R						
Reset:	Unaffected by reset							
Alternate Functions:					ATD11	ATD10	ATD9	ATD8

R = Reserved

Figure 16-11. Port D Data Register (PTD)

PTD[5:0] — Port D Data Bits

PTD[5:0] are read/write, software programmable bits. Data direction of PTD[5:0] pins are under the control of the corresponding bit in data direction register D.

ATD[11:8] — ADC Channel Status Bits

PTD3/ATD11–PTD0/ATD8 are four of the 12 analog-to-digital converter channels. The ADC channel select bits, CH[4:0], determine whether the PTD3/ATD11–PTD0/ATD8 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See **Section 19. Analog-to-Digital Converter (ADC)**.)

NOTE: *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 0.*

NOTE: *Do not use ADC channel ATD14 when using the $\overline{\text{FLT}}$ pin as the clock input for the TIM.*

16.6.2 Data Direction Register D

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

Address: $0007

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Write:	R	R						
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 16-12. Data Direction Register D (DDRD)

DDRD[5:0] — Data Direction Register D Bits

These read/write bits control port D data direction. Reset clears DDRD[5:0], configuring all port D pins as inputs.

1 = Corresponding port D pin configured as output

0 = Corresponding port D pin configured as input

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

Figure 16-13 shows the port D I/O logic.

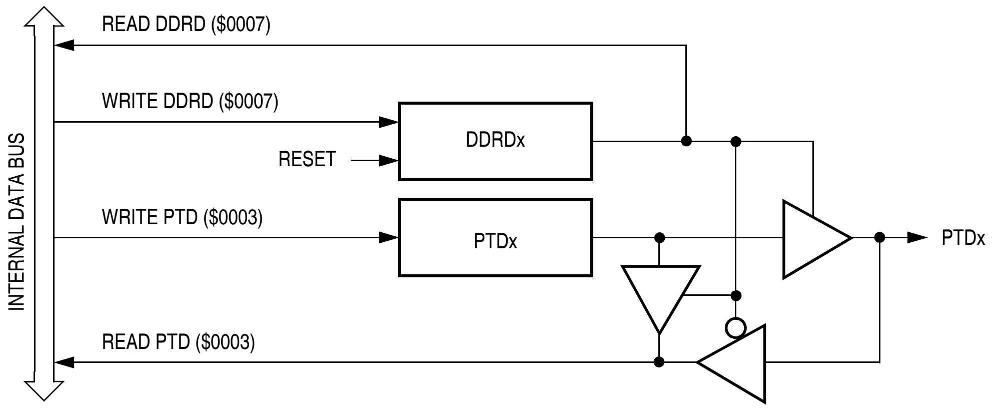


Figure 16-13. Port D I/O Circuit

When bit DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 16-4** summarizes the operation of the port D pins.

Table 16-4. Port D Pin Functions

DDRD Bit	PTD Bit	Bit in Use by ADC	I/O Pin Mode	Accesses to DDRD	Accesses to PTD	
				Read/Write	Read	Write
0	X	No	Input, Hi-Z	DDRD[5:0]	Pin	PTD[5:0][1]
1	X	No	Output	DDRD[5:0]	PTD[5:0]	PTD[5:0]
0	X	Yes	Input, Hi-Z	DDRD[5:0]	0	PTD[5:0][1]
1	X	Yes	Input, Hi-Z	DDRD[5:0]	PTD[5:0]	PTD[5:0]

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

16.7 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).

16.7.1 Port E Data Register

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

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
Reset:	Unaffected by reset							
Alternate Function:	SPSCK	MOSI	MISO	$\overline{\text{SS}}$	TACH1	TACH0	RxD	TxD

Figure 16-14. Port E Data Register (PTE)

PTE[7:0] — Port E Data Bits

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

SPSCK — SPI Serial Clock Bit

The PTE7/SPSCK pin is the serial clock input of an SPI slave module and serial clock output of an SPI master module. When the SPE bit is clear, the PTE7/SPSCK pin is available for general-purpose I/O.

MOSI — Master Out/Slave In Bit

The PTE6/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PTE6/MOSI pin is available for general-purpose I/O. (See **18.14.1 SPI Control Register**.)

MISO — Master In/Slave Out Bit

The PTE5/MISO pin is the master in/slave out terminal of the SPI module. When the SPI enable bit, SPE, is clear, the SPI module is disabled, and the PTE5/MISO pin is available for general-purpose I/O. (See **18.14.1 SPI Control Register**.)

$\overline{\text{SS}}$ — Slave Select Bit

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 **18.13.4 SS (Slave Select)**.) When the SPI is enabled as a slave, the DDRE4 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 16-5**.)*

TACH[1:0] — Timer A Channel I/O Bits

The PTE3/TACH1–PTE2/TACH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE3/TACH1–PTE2/TACH0 pins are timer A channel I/O pins or general-purpose I/O pins. (See **20.9.4 TIMA Channel Status and Control Registers**.)

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 16-5**.)*

RxD — SCI Receive Data Input Bit

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. (See **17.9.1 SCI Control Register 1**.)

TxD — SCI Transmit Data Output

The PTE0/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. (See **17.9.1 SCI Control Register 1**.)

NOTE: *Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the 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 16-5**.)*

16.7.2 Data Direction Register E

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

Address: $000C

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

Figure 16-15. Data Direction Register E (DDRE)

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

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

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

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

Figure 16-16 shows the port E I/O logic.

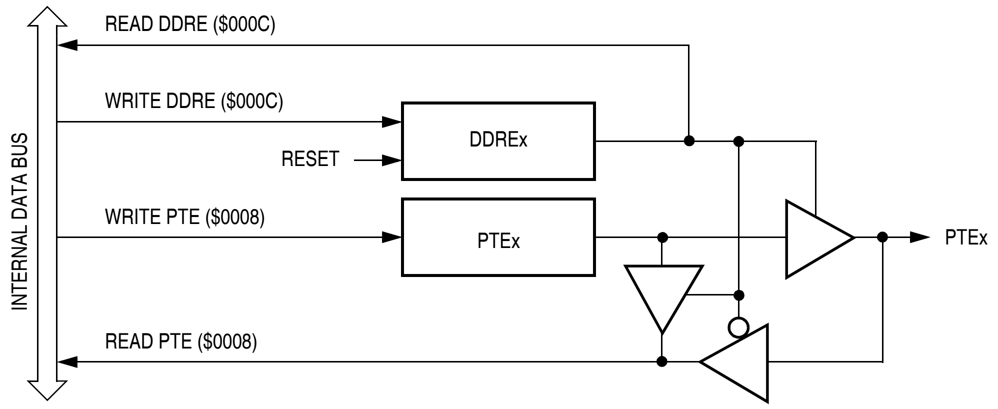


Figure 16-16. Port E I/O Circuit

When bit DDREx is a logic 1, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic 0, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 16-5** summarizes the operation of the port E pins.

Table 16-5. Port E Pin Functions

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

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

16.8 Port F

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

16.8.1 Port F Data Register

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

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	PTF3	PTF2	PTF1	PTF0
Write:	R	R	R	R				
Reset:	Unaffected by reset							
Alternate Function:					TACLK	TBCLK	TBCH1	TBCH2

R = Reserved

Figure 16-17. Port F Data Register (PTF)

PTF[3:0] — Port F Data Bits

These read/write bits are software programmable. Data direction of each port F pin is under the control of the corresponding bit in data direction register F. Reset has no effect on PTF[3:0].

TACLK — Timer A Clock Input Bit

The PTF3/TACLK pin is the external clock input for TIMA. The prescaler select bits, PS[2:0], select PTF3/TACLK as the TIMA clock input. (See **20.9.1 TIMA Status and Control Register**.) When not selected as the TIMA clock, PTF3/TACLK is available for general-purpose I/O.

TBCLK — Timer B Clock Input Bit

The PTF2/TBCLK pin is the external clock input for TIMB. The prescaler select bits, PS[2:0], select $\overline{\text{FLT}}$ as the TIMB clock input. (See **21.9.1 TIMB Status and Control Register**.) When not selected as the TIMB clock, PTF2/TBCLK is available for general-purpose I/O.

TBCH[1:0] — Timer B Channel I/O Bits

The PTF1/TBCH1–PTF0/TBCH0 pins are the TIMB input capture/output compare pins. The edge/level select bits, ELSxB–ELSxA, determine whether the PTF1/TBCH1–PTF0/TBCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See **21.9.1 TIMB Status and Control Register**.)

NOTE: *Data direction register F (DDRF) does not affect the data direction of port F pins that are being used by 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 16-6**.)*

16.8.2 Data Direction Register F

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

Address: $000D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	DDRF3	DDRF2	DDRF1	DDRF0
Write:	R	R	R	R				
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 16-18. Data Direction Register F (DDRF)

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

These read/write bits control port F data direction. Reset clears DDRF[3:0], configuring all port F pins as inputs.

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

NOTE: *Avoid glitches on port F pins by writing to the port F data register before changing data direction register F bits from 0 to 1.*

Figure 16-19 shows the port F I/O logic.

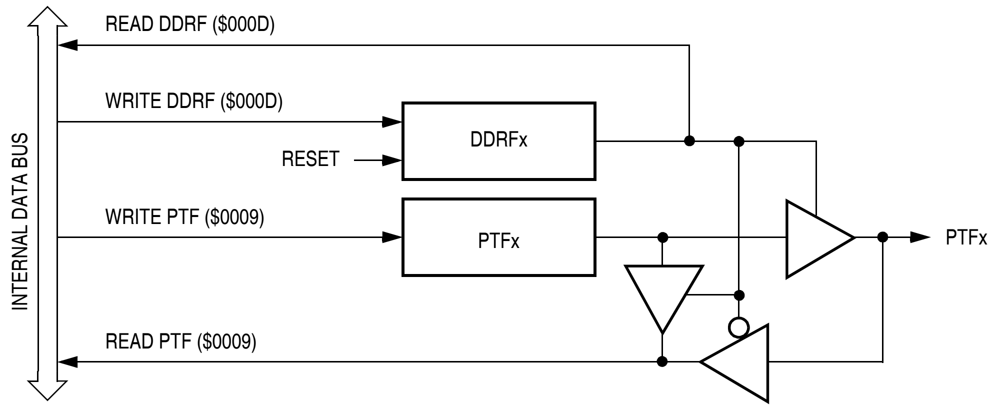


Figure 16-19. Port F I/O Circuit

When bit DDRFx is a logic 1, reading address $0009 reads the PTFx data latch. When bit DDRFx is a logic 0, reading address $0009 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 16-6 summarizes the operation of the port F pins.

Table 16-6. Port F Pin Functions

DDRF Bit	PTF Bit	I/O Pin Mode	Accesses to DDRF	Accesses to PTF	
			Read/Write	Read	Write
0	X	Input, Hi-Z	DDRF[3:0]	Pin	PTF[3:0][1]
1	X	Output	DDRF[3:0]	PTF[3:0]	PTF[3:0]

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

Section 17. Serial Communications Interface (SCI)

17.1 Contents

17.2 Introduction

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

17.3 Features

Features of the SCI module include:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 32 programmable baud rates
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter central processor unit (CPU) interrupt requests
- Programmable transmitter output polarity
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup

- Interrupt-driven operation with eight interrupt flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

17.4 Functional Description

Figure 17-1 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.

Refer to **Figure 17-2** for a summary of the input/output (I/O) registers.

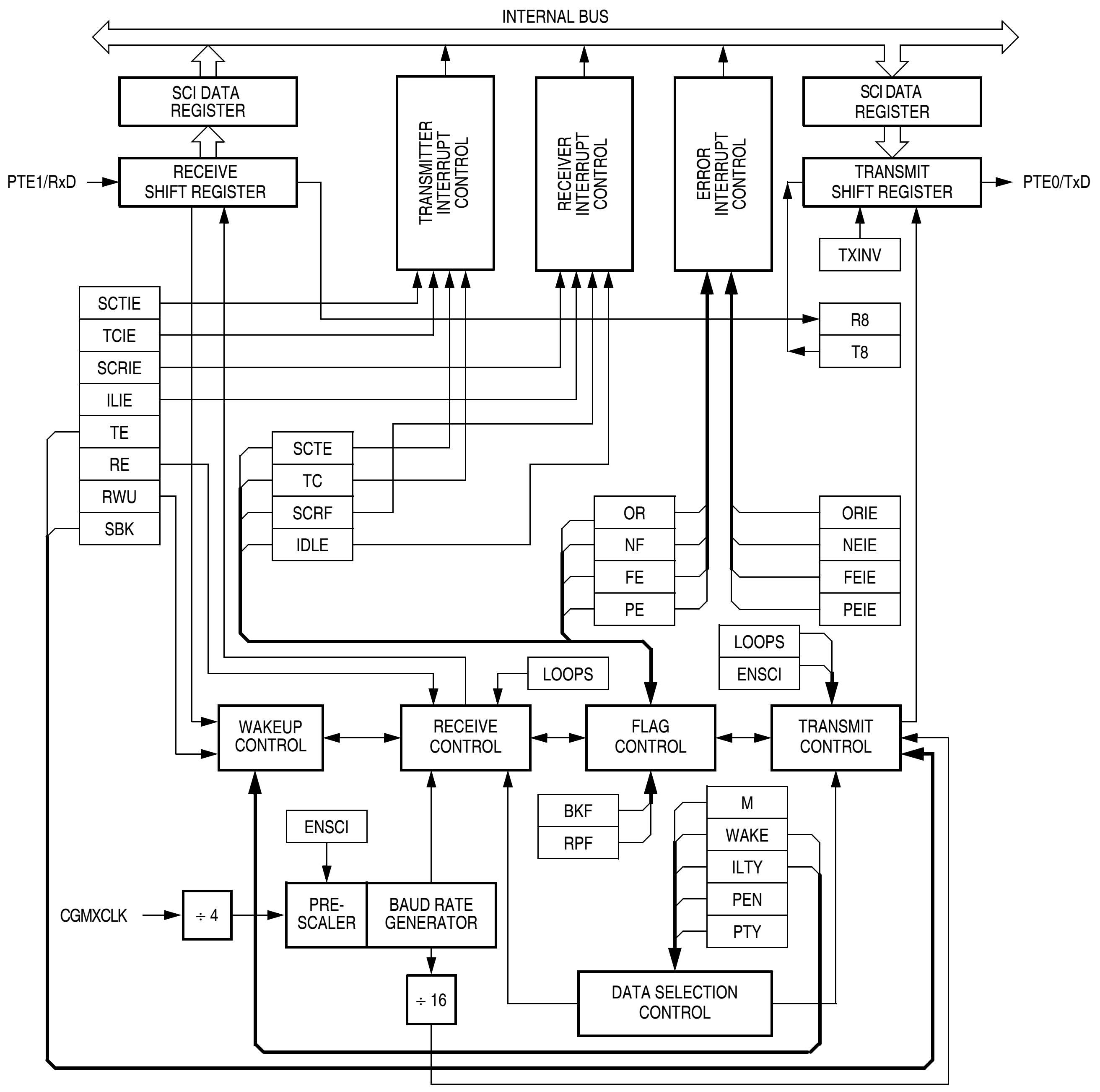


Figure 17-1. SCI Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0013	SCI Control Register 1 (SCC1)	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0014	SCI Control Register 2 (SCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 3 (SCC3)	Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
		Write:	R							
		Reset:	U	U	0	0	0	0	0	0
$0016	SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:	R	R	R	R	R	R	R	R
		Reset:	1	1	0	0	0	0	0	0
$0017	SCI Status Register 2 (SCS2)	Read:	0	0	0	0	0	0	BKF	RPF
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0019	SCI Baud Rate Register (SCBR)	Read:	R	R	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

R	= Reserved
U	= Unaffected

Figure 17-2. SCI I/O Register Summary

17.4.1 Data Format

The SCI uses the standard non-return-to-zero (NRZ) mark/space data format illustrated in **Figure 17-3**.

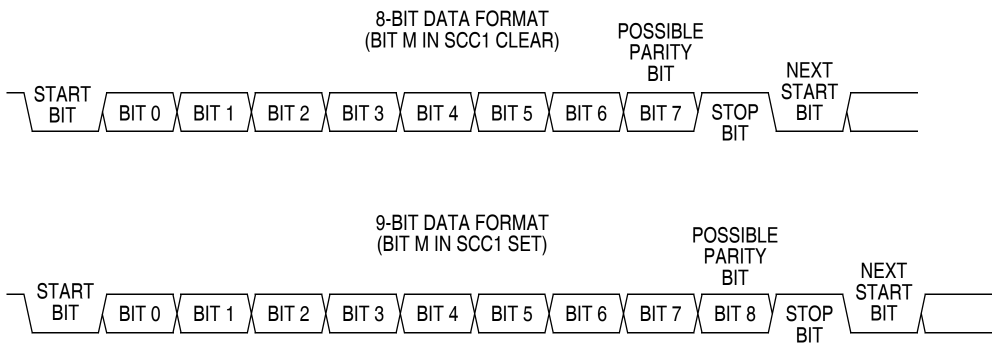


Figure 17-3. SCI Data Formats

17.4.2 Transmitter

Figure 17-4 shows the structure of the SCI transmitter.

17.4.3 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).

17.4.4 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. Initialize the Tx and Rx rates in the SCI baud register (SCBR) ($0019). See **17.9.7 SCI Baud Rate Register**.
2. Enable the SCI by writing a logic 1 to ENSCI in SCI control register 1 (SCC1) ($0013).

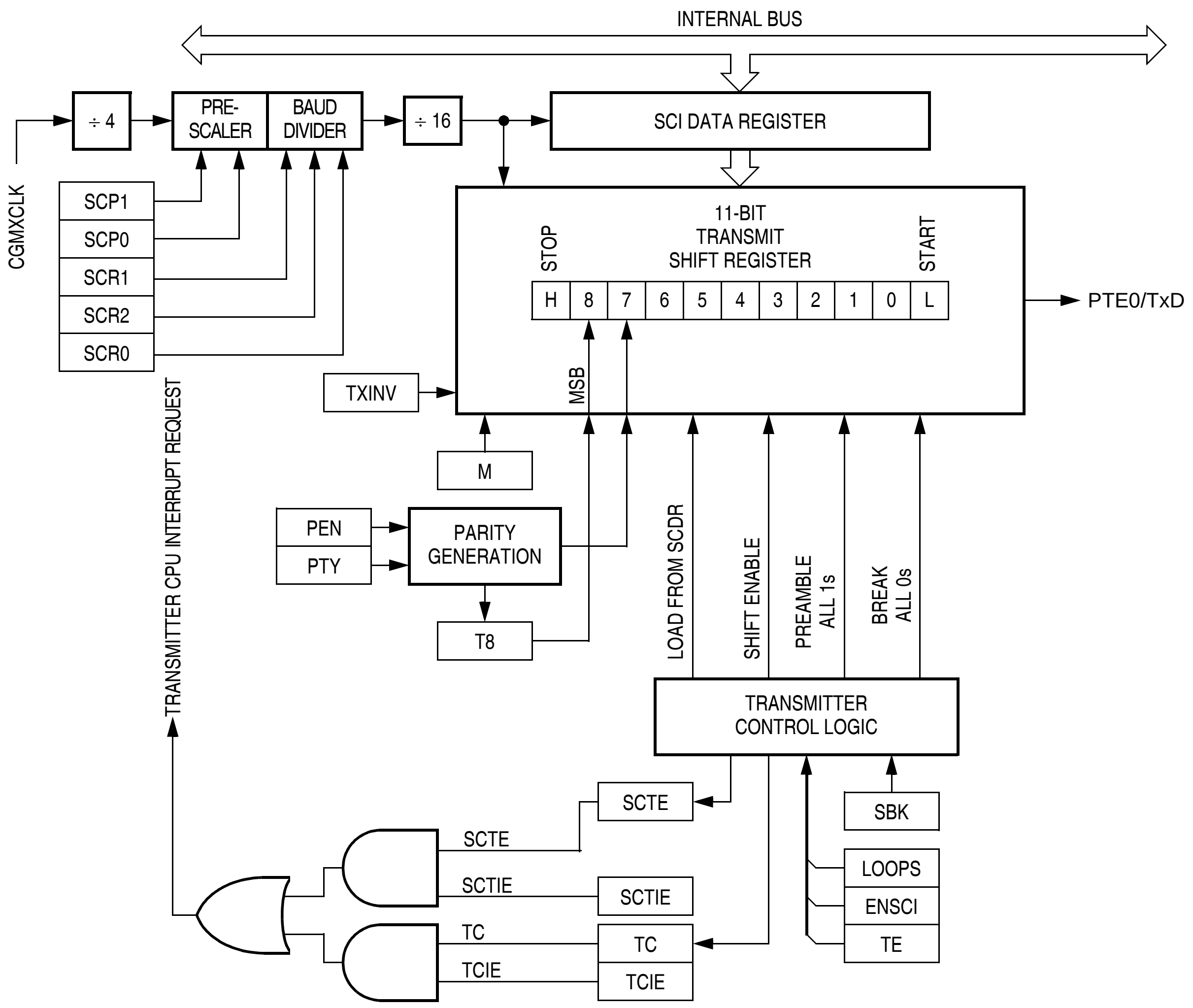


Figure 17-4. SCI Transmitter

3. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2) ($0014).
4. Clear the SCI transmitter empty bit (SCTE) by first reading SCI status register (SCS1) ($0016) and then writing to the SCDR ($0018).
5. 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 10 or 11 logic 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit (LSB) position of the transmit shift register. A logic 1 stop bit goes into the most significant bit (MSB) position.

The SCI transmitter empty bit, SCTE in the SCI status control register (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, SCTI E (SCC2), is also set, the SCTE bit generates a transmitter CPU interrupt request.

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

17.4.5 Break Characters

Writing a logic 1 to the send break bit, SBK (SCC2), loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit (SCC1). As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

The SCI recognizes a break character when a start bit is followed by eight or nine logic 0 data bits and a logic 0 where the stop bit should be. Receiving a break character has 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

17.4.6 Idle Characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit (mode character length) in SCC1. The preamble is a synchronizing idle character that begins every transmission.

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

NOTE: *When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current character shifts out to the 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.

17.4.7 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at logic 1. (See **17.9.1 SCI Control Register 1**.)

17.4.8 Transmitter Interrupts

These conditions can generate CPU interrupt requests from the SCI transmitter:

- SCI transmitter empty (SCTE) — The SCTE bit in SCS1 indicates that the SCDR has transferred a character to the transmit shift register. SCTE can generate a transmitter CPU interrupt request. Setting the SCI transmit interrupt enable bit, SCTIE (SCC2), enables the SCTE bit to generate transmitter CPU interrupt requests.

- Transmission complete (TC) — The TC bit in SCS1 indicates that the transmit shift register and the SCDR are empty and that no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE (SCC2), enables the TC bit to generate transmitter CPU interrupt requests.

17.4.9 Receiver

Figure 17-5 shows the structure of the SCI receiver.

17.4.10 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).

17.4.11 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 (SCC2), is also set, the SCRF bit generates a receiver CPU interrupt request.

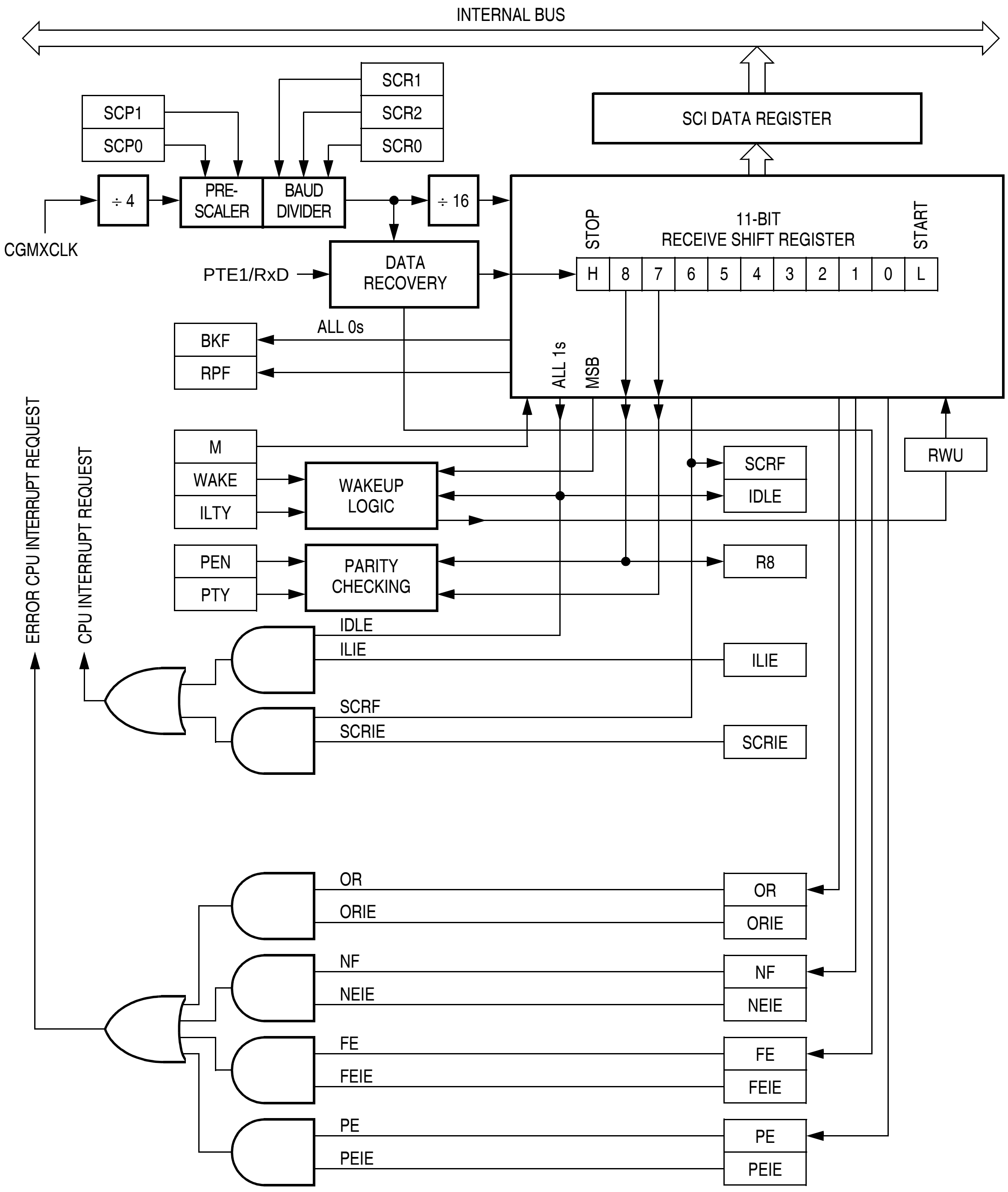


Figure 17-5. SCI Receiver Block Diagram

17.4.12 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 these times (see **Figure 17-6**):

- After every start bit
- After the receiver detects a data bit change from logic 1 to logic 0 (after the majority of data bit samples at RT8, RT9, and RT10 returns a valid logic 1 and the majority of the next RT8, RT9, and RT10 samples returns a valid logic 0)

To locate the start bit, data recovery logic does an asynchronous search for a logic 0 preceded by three logic 1s. When the falling edge of a possible start bit occurs, the RT clock begins to count to 16.

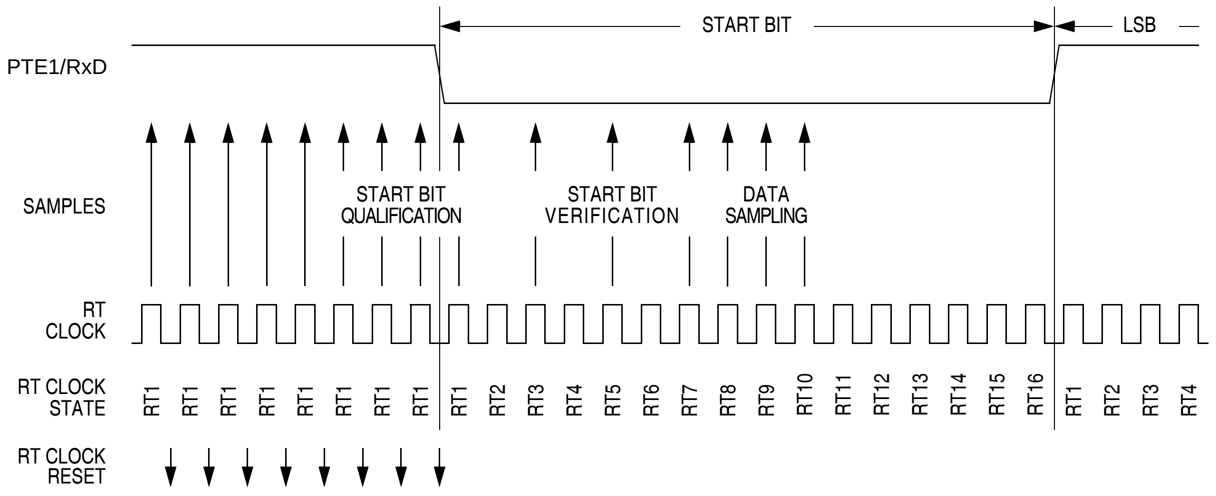


Figure 17-6. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. **Table 17-1** summarizes the results of the start bit verification samples.

Table 17-1. 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 17-2** summarizes the results of the data bit samples.

Table 17-2. Data Bit Recovery

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

NOTE: *The RT8, RT9, and RT10 samples do not affect start bit verification. If any or all of the RT8, RT9, and RT10 start bit samples are logic 1s following a successful start bit verification, the noise flag (NF) is set and the receiver assumes that the bit is a start bit.*

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. **Table 17-3** summarizes the results of the stop bit samples.

Table 17-3. 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

17.4.13 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming character, it sets the framing error bit, FE, in SCS1. The FE flag is set at the same time that the SCRF bit (SCS1) is set. A break character that has no stop bit also sets the FE bit.

17.4.14 Receiver Wakeup

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU (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:

1. Address mark — An address mark is a logic 1 in the most significant bit position of a received character. When the WAKE bit is set, an address mark wakes the receiver from the standby state by clearing the RWU bit. The address mark also sets the SCI receiver full bit, SCRF. Software can then compare the character containing the address mark to the user-defined address of the receiver. If they are the same, the receiver remains awake and processes the characters that follow. If they are not the same, software can set the RWU bit and put the receiver back into the standby state.
2. Idle input line condition — When the WAKE bit is clear, an idle character on the PTE1/RxD pin wakes the receiver from the standby state by clearing the RWU bit. The idle character that wakes the receiver does not set the receiver idle bit, IDLE, or the SCI receiver full bit, SCRF. The idle line type bit, ILTY, determines whether the receiver begins counting logic 1s as idle character bits after the start bit or after the stop bit.

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

17.5 Receiver Interrupts

These sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate a receiver CPU interrupt request. Setting the SCI receive interrupt enable bit, SCRIE (SCC2), enables the SCRF bit to generate receiver CPU interrupts.

- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the PTE1/RxD pin. The idle line interrupt enable bit, ILIE (SCC2), enables the IDLE bit to generate CPU interrupt requests.

17.5.1 Error Interrupts

These 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 (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 (SCC3), enables NF to generate SCI error CPU interrupt requests.
- Framing error (FE) — The FE bit in SCS1 is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE (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 (SCC3), enables PE to generate SCI error CPU interrupt requests.

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 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.

17.6.2 Stop Mode

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

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

17.7 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.

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 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 0. After the break, doing the second step clears the status bit.

17.8 I/O Signals

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

- PTE0/TxD — Transmit data
- PTE1/RxD — Receive data

17.8.1 PTE0/TxD (Transmit Data)

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

17.8.2 PTE1/RxD (Receive Data)

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

17.9 I/O Registers

These I/O registers control and monitor SCI operation:

- SCI control register 1 (SCC1)
- SCI control register 2 (SCC2)
- SCI control register 3 (SCC3)
- SCI status register 1 (SCS1)
- SCI status register 2 (SCS2)
- SCI data register (SCDR)
- SCI baud rate register (SCBR)

17.9.1 SCI Control Register 1

SCI control register 1:

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

Address: $0013

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

Figure 17-7. SCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

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

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI Bit

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

1 = SCI enabled
0 = SCI disabled

TXINV — Transmit Inversion Bit

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

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

M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are eight or nine bits long. (See **Table 17-4**.) The ninth bit can serve as an extra stop bit, as a receiver wakeup signal, or as a parity bit. Reset clears the M bit.

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

WAKE — Wakeup Condition Bit

This read/write bit determines which condition wakes up the SCI: a logic 1 (address mark) in the most significant bit position of a received character or an idle condition on the PTE1/RxD pin. Reset clears the WAKE bit.

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

This read/write bit determines when the SCI starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can 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 17-4**.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See **Figure 17-3**.) Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See **Table 17-4**.) 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 17-4. Character Format Selection

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

17.9.2 SCI Control Register 2

SCI control register 2:

- Enables the following CPU interrupt requests:
  - Enables the SCTE bit to generate transmitter CPU interrupt requests
  - Enables the TC bit to generate transmitter CPU interrupt requests
  - Enables the SCRF bit to generate receiver CPU interrupt requests
  - Enables the IDLE bit to generate receiver CPU interrupt requests
- Enables the transmitter
- Enables the receiver
- Enables SCI wakeup
- Transmits SCI break characters

Address: $0014

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

Figure 17-8. SCI Control Register 2 (SCC2)

SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests. Reset clears the SCTIE bit.
1 = SCTE enabled to generate CPU interrupt requests
0 = SCTE not enabled to generate CPU interrupt 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. Reset clears the SCRIE bit.

1 = SCRF enabled to generate CPU interrupt requests
0 = SCRF not enabled to generate CPU interrupt 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

TE — Transmitter Enable Bit

Setting this read/write bit begins the transmission by sending a preamble of 10 or 11 logic 1s from the transmit shift register to the PTE0/TxD pin. If software clears the TE bit, the transmitter completes any transmission in progress before the PTE0/TxD returns to the idle condition (logic 1). Clearing and then setting TE during a transmission queues an idle character to be sent after the character currently being transmitted. Reset clears the TE bit.

1 = Transmitter enabled
0 = Transmitter disabled

NOTE: *Writing to the TE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.*

RE — Receiver Enable Bit

Setting this read/write bit enables the receiver. Clearing the RE bit disables the receiver but does not affect receiver interrupt flag bits. Reset clears the RE bit.

1 = Receiver enabled
0 = Receiver disabled

NOTE: *Writing to the RE bit is not allowed when the enable SCI bit (ENSCI) is clear. ENSCI is in SCI control register 1.*

RWU — Receiver Wakeup Bit

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

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

Setting and then clearing this read/write bit transmits a break character followed by a logic 1. The logic 1 after the break character guarantees recognition of a valid start bit. If SBK remains set, the transmitter continuously transmits break characters with no logic 1s between them. Reset clears the SBK bit.

1 = Transmit break characters
0 = No break characters transmitted

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

17.9.3 SCI Control Register 3

SCI control register 3:

- Stores the ninth SCI data bit received and the ninth SCI data bit to be transmitted
- Enables these interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts

Address: $0015

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R8	T8	R	R	ORIE	NEIE	FEIE	PEIE
Write:	R							
Reset:	U	U	0	0	0	0	0	0

R = Reserved   U = Unaffected

Figure 17-9. 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 eight bits.

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

T8 — Transmitted Bit 8

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

ORIE — Receiver Overrun Interrupt Enable Bit

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

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

NEIE — Receiver Noise Error Interrupt Enable Bit

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

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

FEIE — Receiver Framing Error Interrupt Enable Bit

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

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

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables SCI receiver CPU interrupt requests generated by the parity error bit, PE. (See **Figure 17-10**.) Reset clears PEIE.

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

17.9.4 SCI Status Register 1

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

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

Address: $0016

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

R = Reserved

Figure 17-10. SCI Status Register 1 (SCS1)

SCTE — SCI Transmitter Empty Bit

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

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

NOTE: *Setting the TE bit for the first time also sets the SCTE bit. Setting the TE and SCTIE bits generates an SCI transmitter CPU 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. TC is cleared automatically when data, preamble, or break character is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break character 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. In normal operation, clear the SCRF bit by reading SCS1 with SCRF set and then reading the SCDR. Reset clears SCRF.

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

IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive logic 1s appear on the receiver input. IDLE generates an SCI error CPU interrupt request if the ILIE bit in SCC2 is also set 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 17-11** shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

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

NF — Receiver Noise Flag Bit

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

1 = Noise detected
0 = No noise detected

FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a logic 0 is accepted as the stop bit. FE generates an SCI error CPU interrupt request if the FEIE bit in SCC3 also is set. Clear the FE bit by reading SCS1 with FE set and then reading the SCDR. Reset clears the FE bit.

1 = Framing error detected
0 = No framing error detected

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

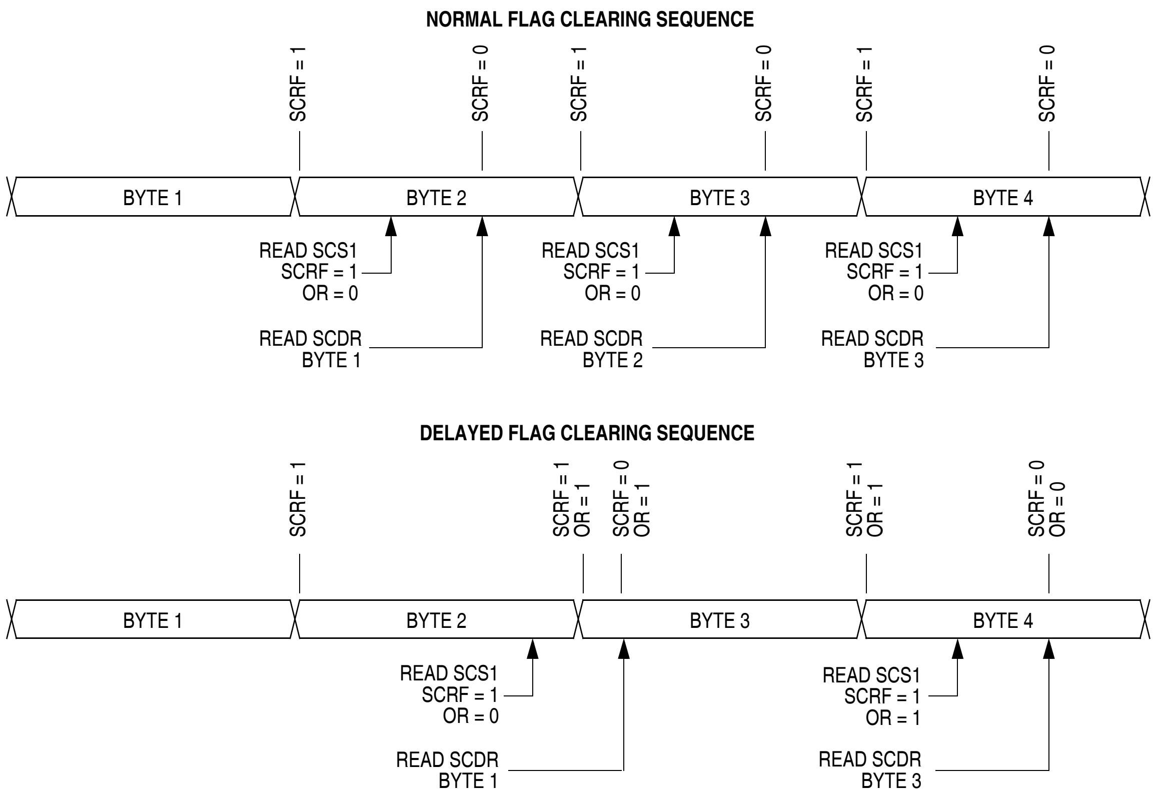


Figure 17-11. Flag Clearing Sequence

17.9.5 SCI Status Register 2

SCI status register 2 contains flags to signal two conditions:

1. Break character detected
2. Incoming data

Address: $0017

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

R = Reserved

Figure 17-12. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the PTE1/RxD pin. In SCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in SCC3 is cleared. BKF does not generate a CPU interrupt request. Clear BKF by reading SCS2 with BKF set and then reading the SCDR. Once cleared, BKF can become set again only after logic 1s again appear on the PTE1/RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF — Reception-in-Progress Flag Bit

This read-only bit is set when the receiver detects a logic 0 during the RT1 time period of the start bit search. RPF does not generate an interrupt request. RPF is reset after the receiver detects false start bits, usually from noise or a baud rate mismatch or when the receiver detects an idle character. Polling RPF before disabling the SCI module or entering stop mode can show whether a reception is in progress.

1 = Reception in progress
0 = No reception in progress

17.9.6 SCI Data Register

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

Address: $0018

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

Figure 17-13. 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.

17.9.7 SCI Baud Rate Register

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

Address: $0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	R	R	SCP1	SCP0	R	SCR2	SCR1	SCR0
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 17-14. SCI BAUD Rate Register 1 (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in **Table 17-5**. Reset clears SCP1 and SCP0.

Table 17-5. SCI Baud Rate Prescaling

SCP1:SCP0	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in **Table 17-6**. Reset clears SCR2:SCR0.

Table 17-6. SCI Baud Rate Selection

SCR2:SCR1:SCR0	Baud Rate Divisor (BD)
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

Use this formula to calculate the SCI baud rate:

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

PD = Prescale divisor (see **Table 17-5**)

BD = Baud rate divisor (see **Table 17-6**)

Table 17-7 shows the SCI baud rates that can be generated with a 4.194-MHz crystal.

Table 17-7. SCI Baud Rate Selection Examples

SCP1:SCP0	Prescaler Divisor (PD)	SCR2:SCR1:SCR0	Baud Rate Divisor (BD)	Baud Rate (f_{XCLK} = 4.194 MHz)
00	1	000	1	65,531
00	1	001	2	32,766
00	1	010	4	16,383
00	1	011	8	8191
00	1	100	16	4095
00	1	101	32	2048
00	1	110	64	1024
00	1	111	128	512
01	3	000	1	21,844
01	3	001	2	10,922
01	3	010	4	5461
01	3	011	8	2730
01	3	100	16	1365
01	3	101	32	683
01	3	110	64	341
01	3	111	128	171
10	4	000	1	16,383
10	4	001	2	8191
10	4	010	4	4096
10	4	011	8	2048
10	4	100	16	1024
10	4	101	32	512
10	4	110	64	256
10	4	111	128	128
11	13	000	1	5041
11	13	001	2	1664
11	13	010	4	1260
11	13	011	8	630
11	13	100	16	315
11	13	101	32	158
11	13	110	64	78.8
11	13	111	128	39.4

Section 18. Serial Peripheral Interface (SPI)

18.1 Contents

18.2 Introduction

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

18.3 Features

Features of the SPI module include:

- Full-duplex operation
- Master and slave modes
- Double-buffered operation with separate transmit and receive registers
- Four master mode frequencies (maximum = bus frequency ÷ 2)
- Maximum slave mode frequency = bus frequency
- Serial clock with programmable polarity and phase
- Two separately enabled interrupts with CPU service:
  - SPRF (SPI receiver full)
  - SPTE (SPI transmitter empty)
- Mode fault error flag with CPU interrupt capability
- Overflow error flag with CPU interrupt capability
- Programmable wired-OR mode
- I^2C (inter-integrated circuit) compatibility

18.4 Pin Name and Register Name Conventions

The generic names of the SPI input/output (I/O) pins are:

- $\overline{\text{SS}}$ (slave select)
- SPSCK (SPI serial clock)
- MOSI (master out/slave in)
- MISO (master in/slave out)

The SPI shares four I/O pins with a parallel I/O port. The full name of an SPI pin reflects the name of the shared port pin. **Table 18-1** shows the full names of the SPI I/O pins. The generic pin names appear in the text that follows.

Table 18-1. Pin Name Conventions

SPI Generic Pin Name:	MISO	MOSI	$\overline{\text{SS}}$	SPSCK
Full SPI pin name:	PTE5/MISO	PTE6/MOSI	PTE4/$\overline{\text{SS}}$	PTE7/SPSCK

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

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

Table 18-2 shows the names and the addresses of the SPI I/O registers.

Table 18-2. I/O Register Addresses

Register Name	Address
SPI control register	\$0010
SPI status and control register	\$0011
SPI data register	\$0012

18.5 Functional Description

Figure 18-1 summarizes the SPI I/O registers and **Figure 18-2** shows the structure of the SPI module.

Addr	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0010	SPI Control Register (SPCR)	Read:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Reset:	0	0	1	0	0	0	0	0
$0011	SPI Status and Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:	R	ERRIE	R	R	R	MODFEN	SPR1	SPR0
		Reset:	0	0	0	0	1	0	0	0
$0012	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							

R = Reserved

Figure 18-1. SPI I/O Register Summary

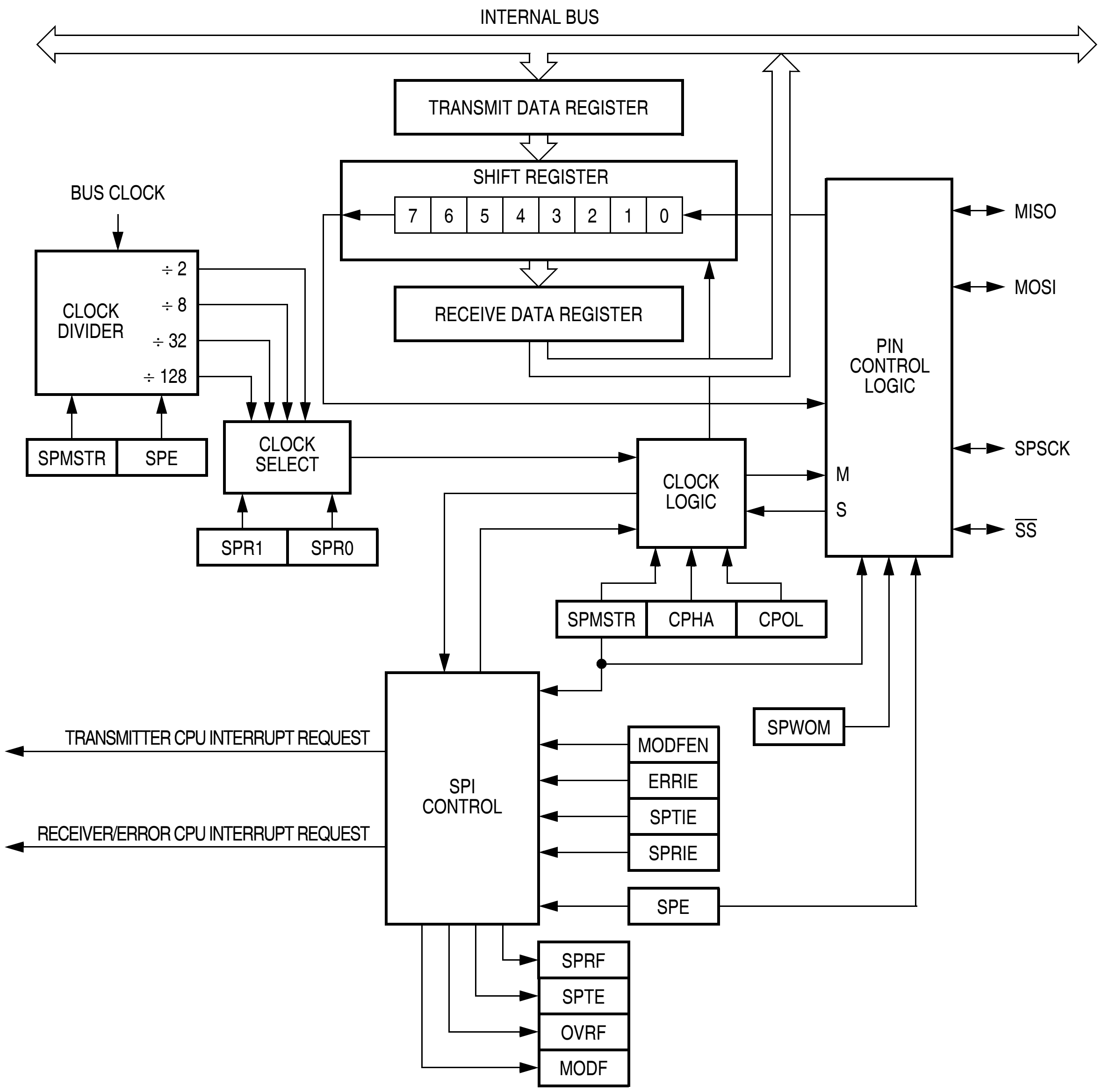


Figure 18-2. SPI Module Block Diagram

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.

The operation of the SPI module is described here.

18.5.1 Master Mode

The SPI operates in master mode when the SPI master bit, SPMSTR (SPCR $0010), is set.

NOTE: *Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. (See* ***18.14.1 SPI Control Register****.)*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE (SPSCR $0011). The byte begins shifting out on the MOSI pin under the control of the serial clock. (See **Figure 18-3**.)

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See **18.14.2 SPI Status and Control Register**.) Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

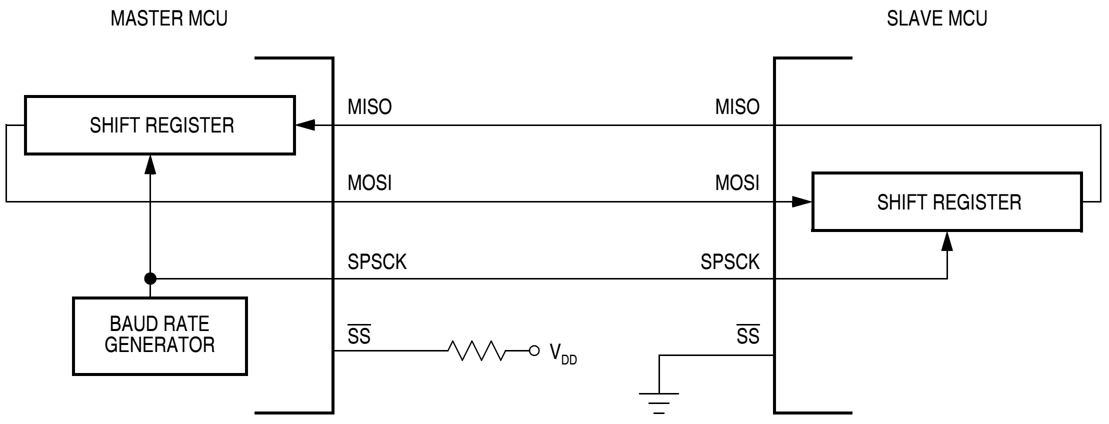


Figure 18-3. Full-Duplex Master-Slave Connections

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

18.5.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit (SPCR $0010) is clear. In slave mode the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave MCU must be at logic 0. $\overline{SS}$ must remain low until the transmission is complete. (See **18.7.2 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 (SPSCR) is set. To prevent an overflow condition, slave software then must read the SPI data register before another byte enters the shift register.

The maximum frequency of the SPSCK for an SPI configured as a slave is the bus clock speed, which is twice as fast as the fastest master SPSCK clock that can be generated. The frequency of the SPSCK for an SPI configured as a slave does not have to correspond to any SPI baud rate. The baud rate only controls the speed of the SPSCK generated by an SPI configured as a master. Therefore, the frequency of the SPSCK for an SPI configured as a slave can be any frequency less than or equal to the bus speed.

When the master SPI starts a transmission, the data in the slave shift register begins shifting out on the MISO pin. The slave can load its shift register with a new byte for the next transmission by writing to its transmit data register. The slave must write to its transmit data register at least one bus cycle before the master starts the next transmission. Otherwise, the byte already in the slave shift register shifts out on the MISO pin.

Data written to the slave shift register during a a transmission remains in a buffer until the end of the transmission.

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{\text{SS}}$ starts a transmission. (See **18.6 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: *To prevent SPSCK from appearing as a clock edge, SPSCK must be in the proper idle state before the slave is enabled.*

18.6 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 be used optionally to indicate a multiple-master bus contention.

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

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

18.6.2 Transmission Format When CPHA = 0

Figure 18-4 shows an SPI transmission in which CPHA (SPCR) is logic 0. The figure should not be used as a replacement for data sheet parametric information.

Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{\text{SS}}$ 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 0, 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 18.7.2 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 $\overline{\text{SS}}$ pin is used to start the transmission. The $\overline{\text{SS}}$ pin must be toggled high and then low again between each byte transmitted as shown in Figure 18-4.

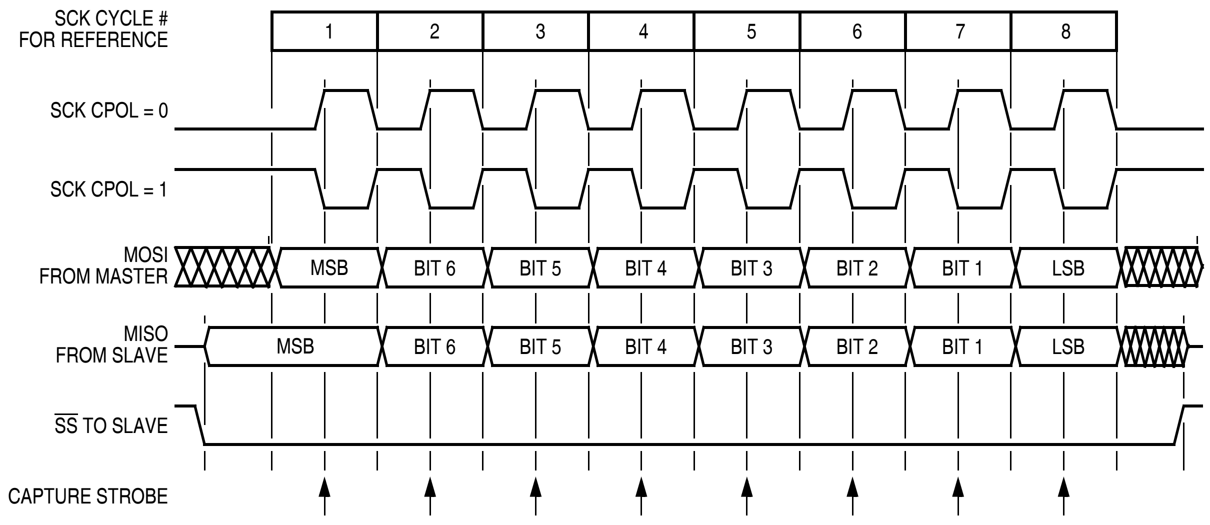


Figure 18-4. Transmission Format (CPHA = 0)

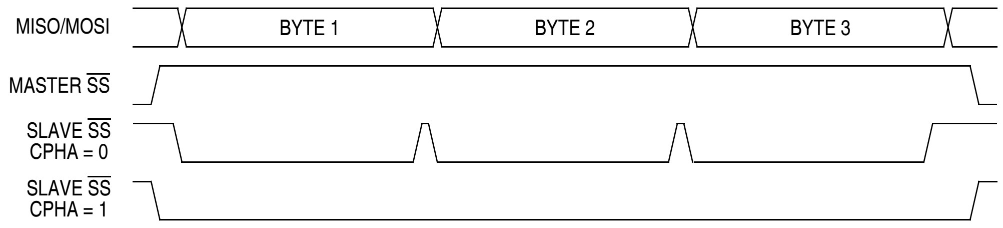

MISO/MOSI
BYTE 1
BYTE 2
BYTE 3
MASTER $\overline{SS}$
SLAVE $\overline{SS}$ CPHA = 0
SLAVE $\overline{SS}$ CPHA = 1

Figure 18-5. CPHA/$\overline{SS}$ Timing

When CPHA = 0 for a slave, the falling edge of $\overline{SS}$ indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the transmit data register and transferred to the shift register after the current transmission.

18.6.3 Transmission Format When CPHA = 1

Figure 18-6 shows an SPI transmission in which CPHA (SPCR) is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See **18.7.2 Mode Fault Error**.) When CPHA = 1, the master

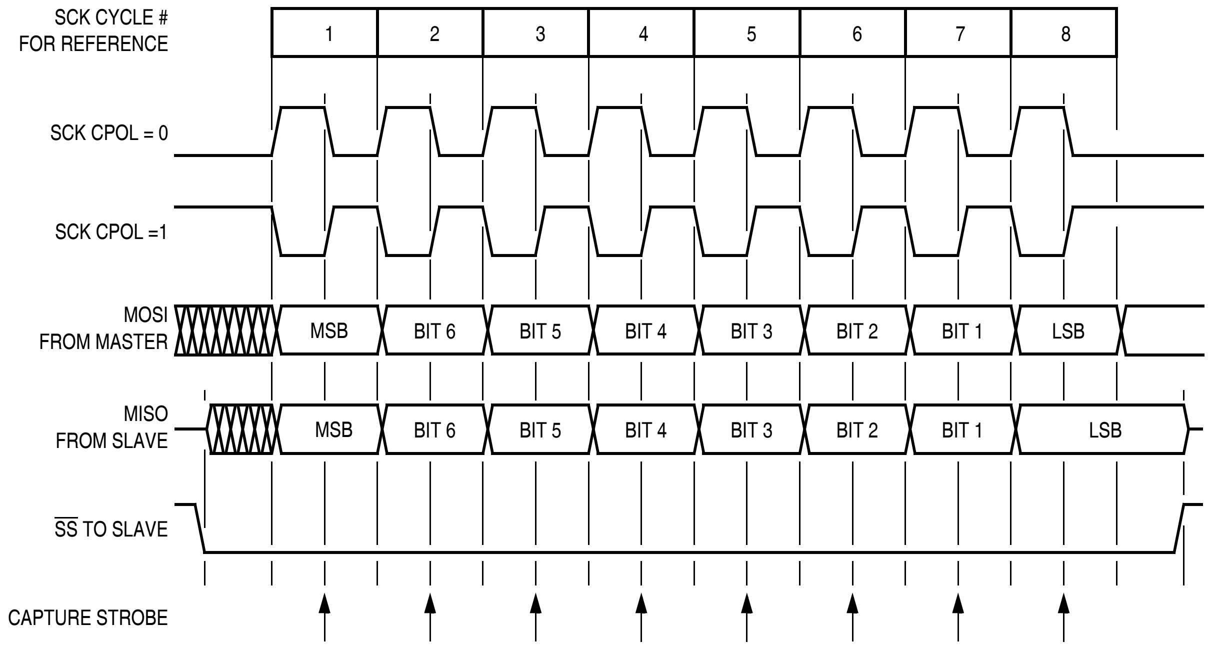


Figure 18-6. Transmission Format (CPHA = 1)

begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. This causes the SPI to leave its idle state and begin driving the MISO pin with the MSB of its data. Once the transmission begins, no new data is allowed into the shift register from the transmit data register. Therefore, the SPI data register of the slave must be loaded with transmit data before the first edge of SPSCK. Any data written after the first edge is stored in the transmit data register and transferred to the shift register after the current transmission.

18.6.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), transmissions are started by a software write to the SPDR ($0012). CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When CPHA = 0, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA = 1, the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by SPR1–SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See **Figure 18-7**.) 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 (SPCR) 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 18-7**. 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.

18.7 Error Conditions

Two flags signal SPI error conditions:

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

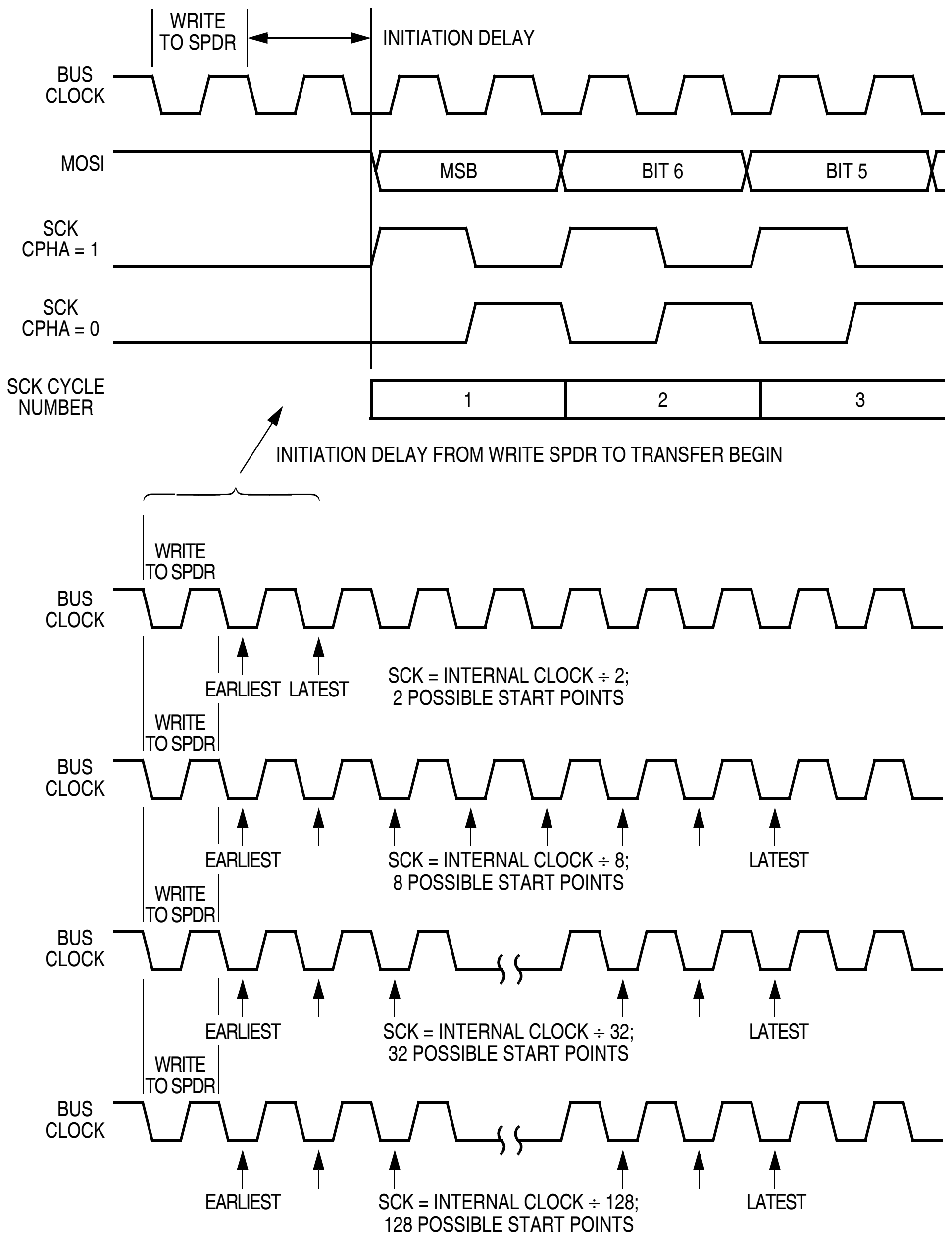

WRITE TO SPDR
INITIATION DELAY
BUS CLOCK
MOSI
MSB
BIT 6
BIT 5
SCK CPHA = 1
SCK CPHA = 0
SCK CYCLE NUMBER
1
2
3
INITIATION DELAY FROM WRITE SPDR TO TRANSFER BEGIN
WRITE TO SPDR
BUS CLOCK
EARLIEST
LATEST
SCK = INTERNAL CLOCK ÷ 2; 2 POSSIBLE START POINTS
WRITE TO SPDR
BUS CLOCK
EARLIEST
SCK = INTERNAL CLOCK ÷ 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 18-7. Transmission Start Delay (Master)

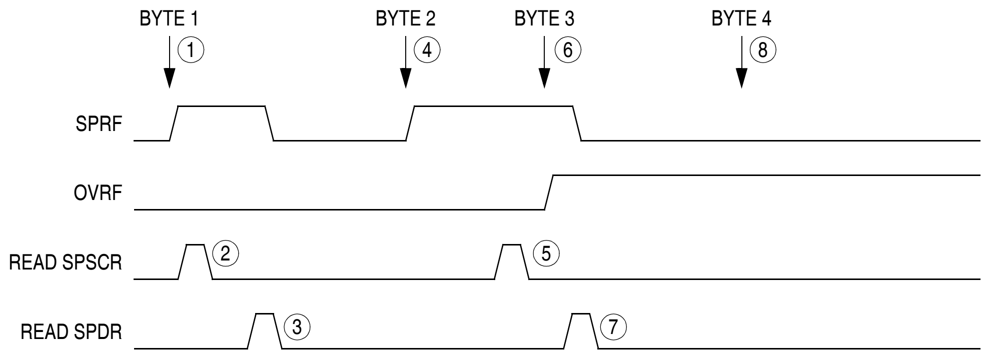
18.7.1 Overflow Error

The overflow flag (OVRF in SPSCR) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. (See **Figure 18-4** and **Figure 18-6**.) 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 in SPSCR) is also set. MODF and OVRF can generate a receiver/error CPU interrupt request. (See **Figure 18-10**.) It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If an end-of-block transmission interrupt was meant to pull the MCU out of wait, having an overflow condition without overflow interrupts enabled causes the MCU to hang in wait mode. If the OVRF is enabled to generate an interrupt, it can pull the MCU out of wait mode instead.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. **Figure 18-8** shows how it is possible to miss an overflow.

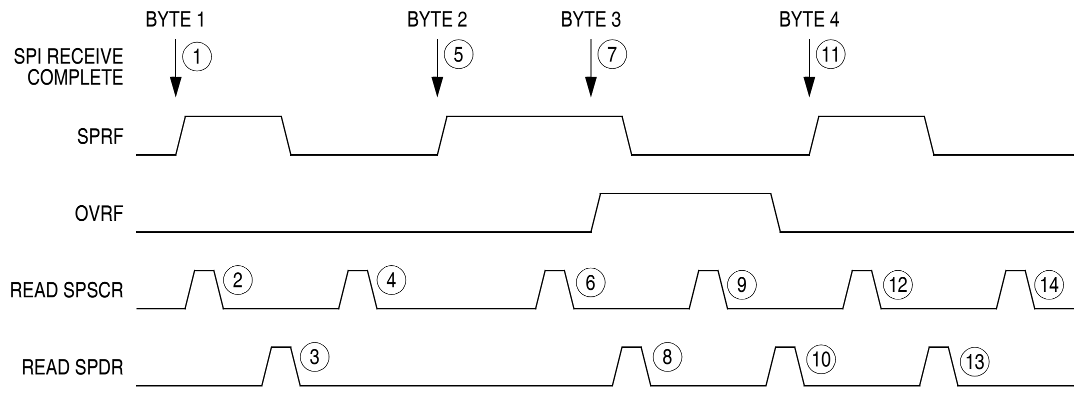


1. BYTE 1 SETS SPRF BIT.
2. CPU READS SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.
3. CPU READS BYTE 1 IN SPDR, CLEARING SPRF BIT.
4. BYTE 2 SETS SPRF BIT.
5. CPU READS SPSCRW WITH SPRF BIT SET AND OVRF BIT CLEAR.
6. BYTE 3 SETS OVRF BIT. BYTE 3 IS LOST.
7. CPU READS BYTE 2 IN SPDR, CLEARING SPRF BIT, BUT NOT OVRF BIT.
8. BYTE 4 FAILS TO SET SPRF BIT BECAUSE OVRF BIT IS SET. BYTE 4 IS LOST.

Figure 18-8. Missed Read of Overflow Condition

The first part of **Figure 18-8** shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in between the time that SPSCR and SPDR are read.

In this case, an overflow can be easily missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR after the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will complete with an SPRF interrupt. **Figure 18-9** illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit (SPSCR).



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 18-9. Clearing SPRF When OVRF Interrupt Is Not Enabled

18.7.2 Mode Fault Error

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. MODF and OVRF can generate a receiver/error CPU interrupt request. (See **Figure 18-10**.) It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{\text{SS}}$ goes to logic 0. A mode fault in a master SPI causes these events to occur:

- If ERRIE = 1, the SPI generates an SPI receiver/error CPU interrupt request.
- The SPE bit is cleared.
- The SPTE bit is set.
- The SPI state counter is cleared.
- The data direction register of the shared I/O port regains control of port drivers.

NOTE: *To prevent bus contention with another master SPI after a mode fault error, clear all data direction register (DDR) bits associated with the SPI shared port pins.*

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

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{\text{SS}}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{\text{SS}}$ goes low and ends once the incoming SPSCK returns to its idle level after the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its IDLE level after the shift of the last data bit. (See **18.6 Transmission Formats**.)

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

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

NOTE: *A logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high impedance state. Also, the slave SPI ignores all incoming SPSCK clocks, even if a transmission has begun.*

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

18.8 Interrupts

The four SPI status flags shown in **Table 18-3** can be enabled to generate CPU interrupt requests:.

Table 18-3. SPI Interrupts

Flag	Request
SPTE (transmitter empty)	SPI transmitter CPU interrupt request (SPTIE = 1)
SPRF (receiver full)	SPI receiver CPU interrupt request (SPRIE = 1)
OVRF (overflow)	SPI receiver/error interrupt request (SPRIE = 1, ERRIE = 1)
MODF (mode fault)	SPI receiver/error interrupt request (SPRIE = 1, ERRIE = 1, MODFEN = 1)

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

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

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

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

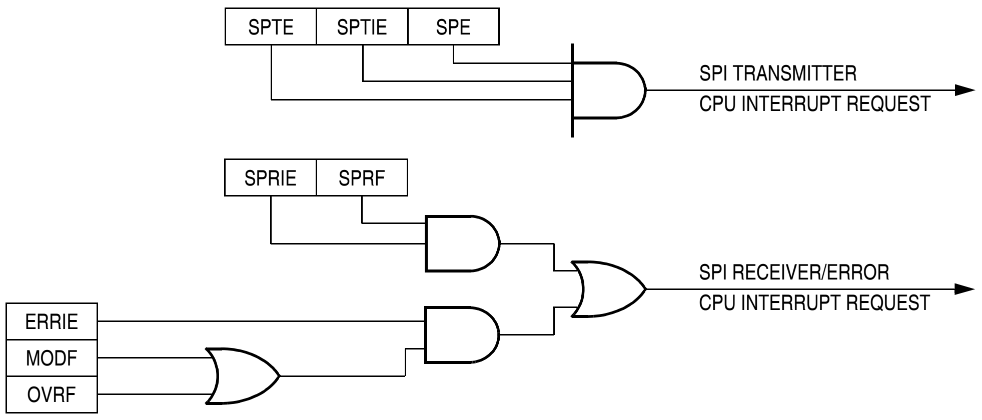


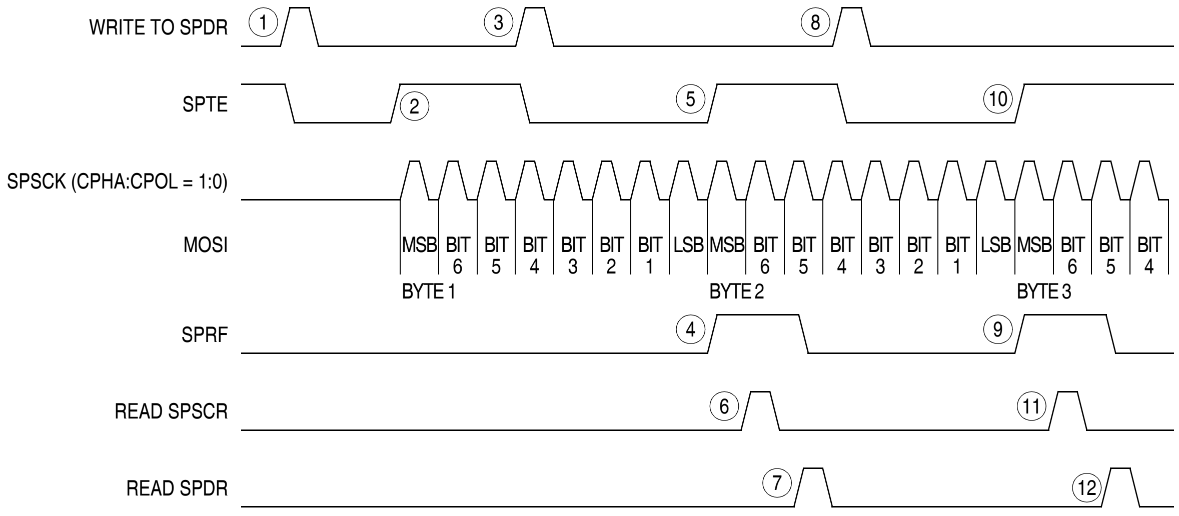
Figure 18-10. SPI Interrupt Request Generation

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

1. SPI receiver full bit (SPRF) — The SPRF bit becomes set every time a byte transfers from the shift register to the receive data register. If the SPI receiver interrupt enable bit, SPRIE, is also set, SPRF can generate an SPI receiver/error CPU interrupt request.
2. SPI transmitter empty (SPTE) — The SPTE bit becomes set every time a byte transfers from the transmit data register to the shift register. If the SPI transmit interrupt enable bit, SPTIE, is also set, SPTE can generate an SPTE CPU interrupt request.

18.9 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 in SPSCR) indicates when the transmit data buffer is ready to accept new data. Write to the SPI data register only when the SPTE bit is high. **Figure 18-11** shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA–CPOL = 1–0).



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 18-11. SPRF/SPTE CPU Interrupt Timing

The transmit data buffer allows back-to-back transmissions without the slave precisely timing its writes between transmissions as in a system with a single data buffer. Also, if no new data is written to the data buffer, the last value contained in the shift register is the next data word to be transmitted.

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

18.10 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, these occur:

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

These additional items are reset only by a system reset:

- All control bits in the SPCR register
- All control bits in the SPSCR register (MODFEN, ERRIE, SPR1, and SPR0)
- The status flags SPRF, OVRF, and MODF

By not resetting the control bits when SPE is low, the user can clear SPE between transmissions without having to reset all control bits when SPE is set back to high for the next transmission.

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

18.11 Low-Power Modes

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

18.11.1 Wait Mode

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

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

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). (See **18.8 Interrupts**.)

18.11.2 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.

18.12 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, $FE03) enables software to clear status bits during the break state. (See **10.8.3 SIM Break Flag Control Register**.)

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the 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.

18.13 I/O Signals

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

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{\text{SS}}$ — 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}.

18.13.1 MISO (Master In/Slave Out)

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

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic 0 and its $\overline{\text{SS}}$ pin is at logic 0. To support a multiple-slave system, a logic 1 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.

18.13.2 MOSI (Master Out/Slave In)

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

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

18.13.3 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.

18.13.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transmission. (See **18.6 Transmission Formats**.) Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low throughout the transmission for the CPHA = 1 format. See **Figure 18-12**.

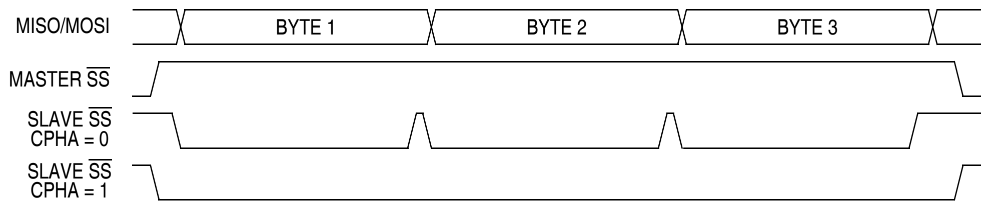

MISO/MOSI
BYTE 1
BYTE 2
BYTE 3
MASTER $\overline{SS}$
SLAVE $\overline{SS}$ CPHA = 0
SLAVE $\overline{SS}$ CPHA = 1

Figure 18-12. CPHA/$\overline{SS}$ Timing

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{SS}$ from creating a MODF error. (See **18.14.2 SPI Status and Control Register**.)

NOTE: *A logic 1 voltage on the $\overline{SS}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if a transmission already has begun.*

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See **18.7.2 Mode Fault Error**.) For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the data register. (See **Table 18-4**.)

Table 18-4. SPI Configuration

SPE	SPMSTR	MODFEN	SPI Configuration	State of $\overline{SS}$ Logic
0	X	X	Not enabled	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; $\overline{SS}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

X = don’t care

18.13.5 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.

18.14 I/O Registers

Three registers control and monitor SPI operation:

- SPI control register (SPCR, $0010)
- SPI status and control register (SPSCR, $0011)
- SPI data register (SPDR, $0012)

18.14.1 SPI Control Register

The SPI control register:

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

Address: $0010

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

R = Reserved

Figure 18-13. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

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

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

SPMSTR — SPI Master Bit

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

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See **Figure 18-4** and **Figure 18-6**.) 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 18-4** and **Figure 18-6**.) 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 1 between bytes. (See **Figure 18-12**.) 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 **18.7.2 Mode Fault Error**.) A logic 1 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 pullup devices on pins SPSCK, MOSI, and MISO so that those pins become open-drain outputs.

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

SPE — SPI Enable Bit

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See **18.10 Resetting the SPI**.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable Bit

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

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

18.14.2 SPI Status and Control Register

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

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

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

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

Address: $0011

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

R = Reserved

Figure 18-14. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

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

During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Any read of the SPI data register clears the SPRF bit.

Reset clears the SPRF bit.

1 = Receive data register full

0 = Receive data register not full

ERRIE — Error Interrupt Enable Bit

This 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 status and control 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 if the SPTIE bit in the SPI control register is set also.

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

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

Reset sets the SPTE bit.

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

MODFEN — Mode Fault Enable Bit

This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the $\overline{\text{SS}}$ 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 $\overline{\text{SS}}$ pin is not available as a general-purpose I/O regardless of the value of MODFEN. (See **18.13.4 SS (Slave Select)**.)

If the MODFEN bit is low, the level of the $\overline{\text{SS}}$ 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 **18.7.2 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 18-5**. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 18-5. 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), see **Section 9. Clock Generator Module (CGM)**.

BD = baud rate divisor

18.14.3 SPI Data Register

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate buffers that can contain different values. See **Figure 18-2**.

Address: $0012

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

Figure 18-15. SPI Data Register (SPDR)

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

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

Section 19. Analog-to-Digital Converter (ADC)

19.1 Contents

19.2 Introduction

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

19.3 Features

Features of the ADC module include:

- 12 channels (52-PLCC) with multiplexed input
- Linear successive approximation
- 8-bit resolution
- Single or continuous conversion
- Conversion complete flag or conversion complete interrupt
- Selectable ADC clock

19.4 Functional Description

Twelve ADC channels are available for sampling external sources at pins PTD3/ATD11–PTD0/ATD8 and PTB7/ATD7–PTB0/ATD0. An analog multiplexer allows the single ADC converter to select one of the 12 ADC channels as ADC voltage input (ADCVIN). ADCVIN is converted by the successive approximation register-based counters. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. (See **Figure 19-1**.)

19.4.1 ADC Port I/O Pins

PTD3/ATD11–PTD0/ATD8 and PTB7/ATD7–PTB0/ATD0 are general-purpose input/output (I/O) pins that are shared with the ADC channels.

The channel select bits (ADC status control register, $0038), define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC.

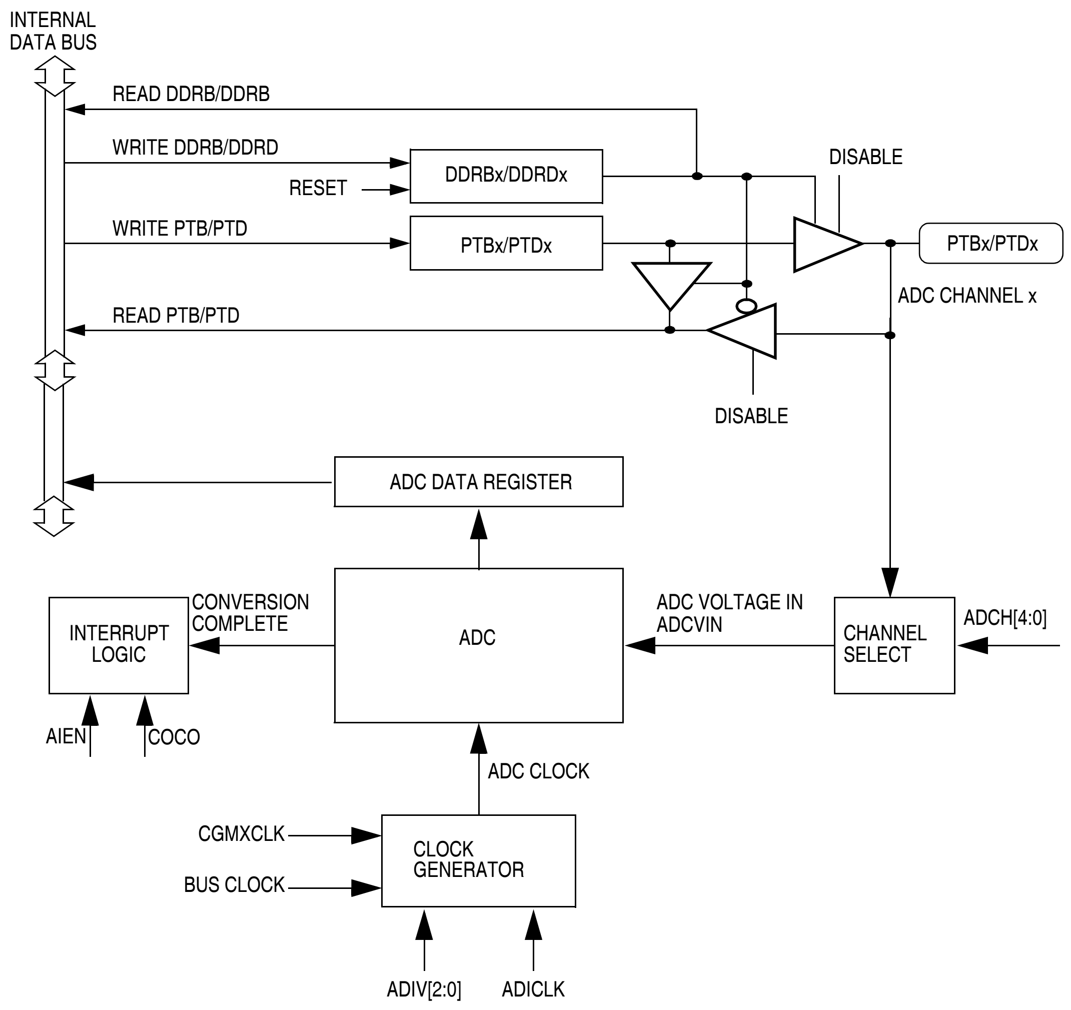


Figure 19-1. ADC Block Diagram

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 0 if the corresponding DDR bit is at logic 0. If the DDR bit is at logic 1, the value in the port data latch is read.

19.4.2 Voltage Conversion

When the input voltage to the ADC equals V_{REFH} (see **23.7 ADC Characteristics**), 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.*

19.4.3 Conversion Time

Sixteen ADC internal clocks are required to perform one conversion. The ADC starts a conversion on the first rising edge of the ADC internal clock immediately following a write to the ADSCR. If the ADC internal clock is selected to run at 1 MHz, then one conversion will take 16 μs to complete. But since the ADC can run almost completely asynchronously to the bus clock, (for example, the ADC is configured to derive its internal clock from CGMXCLK and the bus clock is being derived from the PLL within the CGM [CGMOUT]), this 16 μs conversion can take up to 17 μs to complete. This worst-case could occur if the write to the ADSCR happened directly after the rising edge of the ADC internal clock causing the conversion to wait until the next rising edge of the ADC internal clock. With a 1-MHz ADC internal clock the maximum sample rate is 59 kHz to 62 kHz. Refer to **23.7 ADC Characteristics**

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

Number of bus cycles = Conversion time x bus frequency

19.4.4 Continuous Conversion

In the continuous conversion mode, the ADC continuously converts the selected channel filling the ADC data register 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 (ADC status control register, $0038) is set after each conversion and can be cleared by writing the ADC status and control register or reading of the ADC data register.

19.4.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes. See **23.7 ADC Characteristics** for accuracy information.

19.5 Interrupts

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

19.6 Low-Power Modes

This section describes the low-power modes.

19.6.1 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 to logic 1s before executing the WAIT instruction.

19.6.2 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.

19.7 I/O Signals

The ADC module has 12 channels that are shared with I/O port B and port D. Refer to **23.7 ADC Characteristics** for voltages referenced in the next three subsections.

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

The ADC analog portion uses V_{DDA}/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_{DDA}/V_{DDAREF} for good results.

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.

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

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

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

V_{REFL} is the lower reference supply for the ADC. Connect the V_{REFL} pin to a voltage potential between V_{SSA} and 0.5 volts depending on the desired lower conversion boundary.

NOTE: *In a 52-pin package V_{SSA} and V_{REFL} are bonded together.*

19.7.3 ADC Voltage In (ADCVIN)

ADCVIN is the input voltage signal from one of the 12 ADC channels to the ADC module.

19.8 I/O Registers

These I/O registers control and monitor ADC operation:

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

19.8.1 ADC Status and Control Register

These paragraphs describe the function of the ADC status and control register.

Address: $0038

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

R = Reserved

Figure 19-2. ADC Status and Control Register (ADSCR)

COCO — Conversions Complete Bit

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

1 = Conversion completed (AIEN = 0)
0 = Conversion not completed (AIEN = 0)

or

0 = CPU interrupt enabled (AIEN = 1)

AIEN — ADC Interrupt Enable Bit

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

1 = ADC interrupt enabled
0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion Bit

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

1 = Continuous ADC conversion
0 = One ADC conversion

ADCH[4:0] — ADC Channel Select Bits

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of the ADC channels. The five channel select bits are detailed in **Table 19-1**.

NOTE: *Care should be taken when using a port pin as both an analog and a digital input simultaneously to prevent switching noise from corrupting the analog signal.*

The ADC subsystem is turned off when the channel select bits are all set to 1. This feature allows for reduced power consumption for the MCU when the ADC is not used. Reset sets all of these bits to a logic 1.

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

Table 19-1. 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
Range 01100 (\$0C) to 11010 (\$1A)					Unused (see Note 1)
1	1	0	1	1	Reserved
1	1	1	0	0	Reserved
1	1	1	0	1	V_{REFH} (see Note 2)
1	1	1	1	0	V_{SSA}/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.

19.8.2 ADC Data Register

One 8-bit result register is provided. This register is updated each time an ADC conversion completes.

Address: $0039

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
Write:	R	R	R	R	R	R	R	R
Reset:	Indeterminate after reset							

R = Reserved

Figure 19-3. ADC Data Register (ADR)

19.8.3 ADC Input Clock Register

This register selects the clock frequency for the ADC.

Address: $003A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	ADIV2	ADIV1	ADIV0	ADICLK	0	0	0	0
Write:					R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 19-4. ADC Input Clock Register (ADICLK)

ADIV2:ADIV0 — ADC Clock Prescaler Bits

ADIV2, ADIV1, and ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. **Table 19-2** shows the available clock configurations. The ADC clock should be set to approximately 1 MHz.

Table 19-2. ADC Clock Divide Ratio

ADIV2	ADIV1	ADIV0	ADC Clock Rate
0	0	0	ADC input clock / 1
0	0	1	ADC input clock / 2
0	1	0	ADC input clock / 4
0	1	1	ADC input clock / 8
1	X	X	ADC input clock / 16

X = don't care

ADICLK — ADC Input Clock Register Bit

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

If the external clock (CGMXCLK) is equal to or greater than 1 MHz, CGMXCLK can be used as the clock source for the ADC. If CGMXCLK is less than 1 MHz, use the PLL-generated bus clock as the clock source. As long as the internal ADC clock is at approximately 1 MHz, correct operation can be guaranteed. (See **23.7 ADC Characteristics**.)

1 = Internal bus clock
0 = External clock (CGMXCLK)

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

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

Section 20. Timer Interface A (TIMA)

20.1 Contents

20.2 Introduction

This section describes the timer interface module (TIMA). The TIMA is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation functions. **Figure 20-1** is a block diagram of the TIMA.

20.3 Features

Features of the TIMA include:

- Two input capture/output compare channels:
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered pulse width modulation (PWM) signal generation
- Programmable TIMA clock input:
  - 7-frequency internal bus clock prescaler selection
  - External TIMA clock input (4-MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMA counter stop and reset bits
- Modular architecture expandable to eight channels

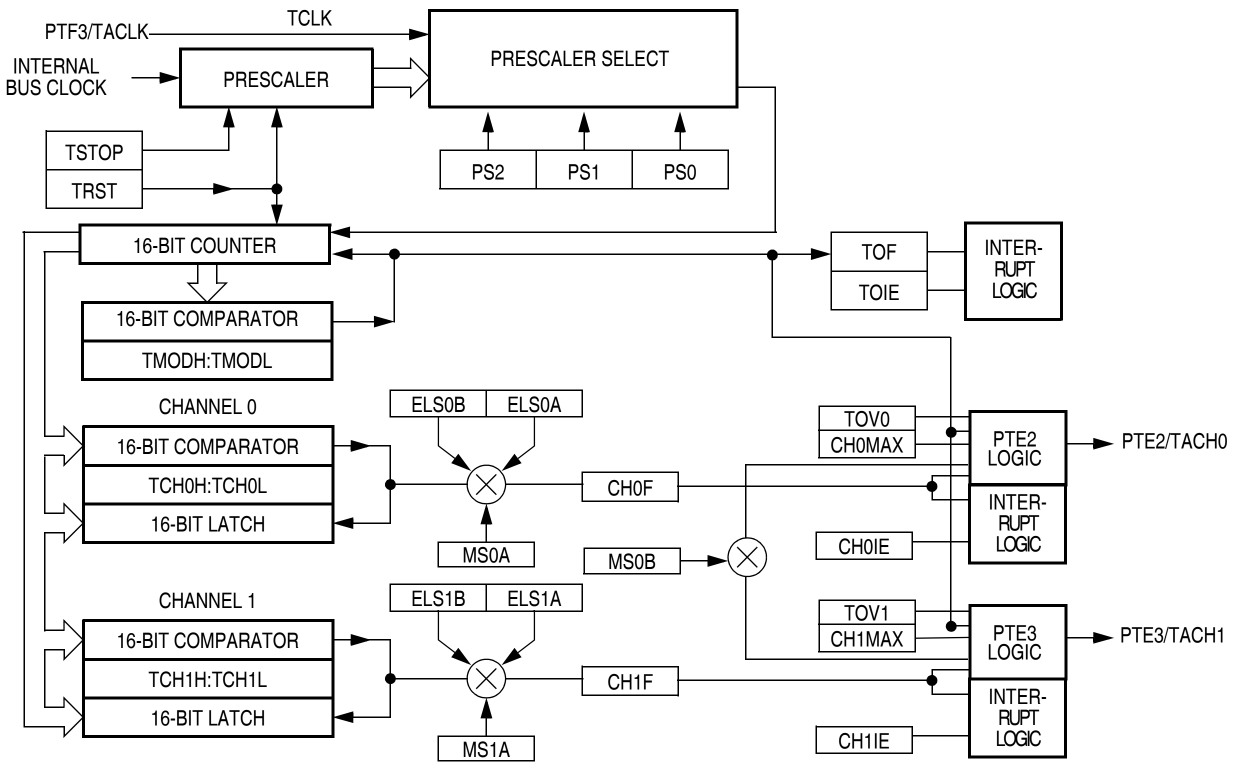


Figure 20-1. TIMA Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	TIMA Status and Control Register (TASC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$0022	TIMA Counter Register High (TACNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0023	TIMA Counter Register Low (TACNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 20-2. TIMA I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0024	TIMA Modulo Register High (TAMODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	TIMA Modulo Register Low (TAMODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0026	TIMA Channel 0 Status and Control Register (TASC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0027	TIMA Channel 0 Register High (TACH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0028	TIMA Channel 0 Register Low (TACH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0029	TIMA Channel 1 Status and Control Register (TASC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$002A	TIMA Channel 1 Register High (TACH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002B	TIMA Channel 1 Register Low (TACH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

R = Reserved

Figure 20-2. TIMA I/O Register Summary (Continued)

20.4 Functional Description

Figure 20-1 shows the TIMA structure. The central component of the TIMA is the 16-bit TIMA counter that can operate as a free-running counter or a modulo up-counter. The TIMA counter provides the timing reference for the input capture and output compare functions. The TIMA counter modulo registers, TAMODH–TAMODL, control the modulo value of the TIMA counter. Software can read the TIMA counter value at any time without affecting the counting sequence.

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

20.4.1 TIMA Counter Prescaler

The TIMA clock source can be one of the seven prescaler outputs or the TIMA clock pin, PTF3/TACLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMA status and control register select the TIMA clock source.

20.4.2 Input Capture

An input capture function has three basic parts:

- Edge select logic
- Input capture latch
- 16-bit counter

Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TASC0–TASC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMA latches the contents of the TIMA counter into the TIMA channel registers, TCHxH–TCHxL. Input captures can generate TIMA CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The result obtained by an input capture will be one more than the value of the free-running counter on the rising edge of the internal bus clock preceding the external transition. This delay is required for internal synchronization.

The free-running counter contents are transferred to the TIMA channel status and control register (TASC0–TASC1, see **20.9.4 TIMA Channel Status and Control Registers**) on each proper signal transition regardless of whether the TIMA channel flag (CH0F–CH1F in TASC0–TASC1 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or "captured" is the time of the event. Because this value is stored in the input capture register when the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see **20.9.5 TIMA Channel Registers**). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture register.

20.4.3 Output Compare

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

20.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **20.4.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMA channel registers.

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

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

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

20.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE3/TACH1 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The output compare value in the TIMA channel 0 registers initially controls the output on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

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.*

20.4.4 Pulse Width Modulation (PWM)

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

As **Figure 20-3** shows, the output compare value in the TIMA channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMA to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIMA to set the pin if the state of the PWM pulse is logic 0.

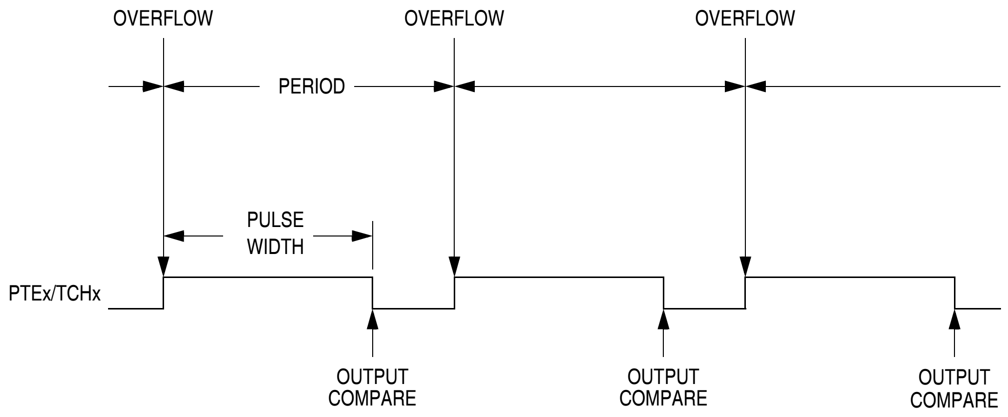


Figure 20-3. PWM Period and Pulse Width

The value in the TIMA counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing \$00FF (255) to the TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is \$000 (see **20.9.1 TIMA Status and Control Register**).

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

20.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **20.4.4 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMA channel registers.

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

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

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

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0 percent 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.*

20.4.4.2 Buffered PWM Signal Generation

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

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The TIMA channel 0 registers initially control the pulse width on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

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.*

20.4.4.3 PWM Initialization

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

1. In the TIMA status and control register (TASC):
   a. Stop the TIMA counter by setting the TIMA stop bit, TSTOP.
   b. Reset the TIMA counter by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH–TAMODL), write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH–TACHxL), write the value for the required pulse width.
4. In TIMA channel x status and control register (TASCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA. (See **Table 20-2**.)
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See **Table 20-2**.)

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

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

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H–TACH0L) initially control the buffered PWM output. TIMA channel 0 status control register 0 (TASC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

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

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100 percent duty cycle output. (See **20.9.4 TIMA Channel Status and Control Registers**.)

20.5 Interrupts

These TIMA sources can generate interrupt requests:

- TIMA overflow flag (TOF) — The TOF bit is set when the TIMA counter value rolls over to $0000 after matching the value in the TIMA counter modulo registers. The TIMA overflow interrupt enable bit, TOIE, enables TIMA overflow CPU interrupt requests. TOF and TOIE are in the TIMA status and control register.
- TIMA channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMA CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

20.6 Low-Power Modes

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

20.6.1 Wait Mode

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

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

20.6.2 Stop Mode

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

20.7 TIMA During Break Interrupts

A break interrupt stops the TIMA counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **10.8.3 SIM Break Flag Control Register**.)

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

20.8 I/O Signals

Port F shares one of its pins with the TIMA. Port E shares two of its pins with the TIMA. PTF3/TACLK is an external clock input to the TIMA prescaler. The two TIMA channel I/O pins are PTE2/TACH0 and PTE3/TACH1.

20.8.1 TIMA Clock Pin (PTF3/TACLK)

PTF3/TACLK is an external clock input that can be the clock source for the TIMA counter instead of the prescaled internal bus clock. Select the PTF3/TACLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. (See **20.9.1 TIMA Status and Control Register**.) The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

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

See **23.8 5.0 Vdc ± 10% Serial Peripheral Interface (SPI) Timing** number 6.

The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

PTF3/TACLK is available as a general-purpose I/O pin or ADC channel when not used as the TIMA clock input. When the PTF3/TACLK pin is the TIMA clock input, it is an input regardless of the state of the DDRF3 bit in data direction register F.

20.8.2 TIMA Channel I/O Pins (PTE3/TACH1–PTE2/TACH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE2/TACH0 can be configured as a buffered output compare or a buffered PWM pin.

20.9 I/O Registers

These I/O registers control and monitor TIMA operation:

- TIMA status and control register (TASC)
- TIMA control registers (TACNTH–TACNTL)
- TIMA counter modulo registers (TAMODH–TAMODL)
- TIMA channel status and control registers (TASC0 and TASC1)
- TIMA channel registers (TACH0H–TACH0L and TACH1H–TACH1L)

20.9.1 TIMA Status and Control Register

The TIMA status and control register:

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

Address: $0020

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

R = Reserved

Figure 20-4. TIMA Status and Control Register (TASC)

TOF — TIMA Overflow Flag Bit

This read/write flag is set when the TIMA counter resets to $0000 after reaching the modulo value programmed in the TIMA counter modulo registers. Clear TOF by reading the TIMA status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMA overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIMA counter has reached modulo value.
0 = TIMA counter has not reached modulo value.

TOIE — TIMA Overflow Interrupt Enable Bit

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

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

TSTOP — TIMA Stop Bit

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

1 = TIMA counter stopped
0 = TIMA counter active

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

TRST — TIMA Reset Bit

Setting this write-only bit resets the TIMA counter and the TIMA prescaler. Setting TRST has no effect on any other registers. Counting resumes from \$0000. TRST is cleared automatically after the TIMA counter is reset and always reads as logic 0. Reset clears the TRST bit.

1 = Prescaler and TIMA counter cleared
0 = No effect

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD4 pin or one of the seven prescaler outputs as the input to the TIMA counter as **Table 20-1** shows. Reset clears the PS[2:0] bits.

Table 20-1. Prescaler Selection

PS[2:0]	TIMA Clock Source
000	Internal bus clock ÷1
001	Internal bus clock ÷ 2
010	Internal bus clock ÷ 4
011	Internal bus clock ÷ 8
100	Internal bus clock ÷ 16
101	Internal bus clock ÷ 32
110	Internal bus clock ÷ 64
111	PTF3/TACLK

20.9.2 TIMA Counter Registers

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

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

Register Name and Address: TACNTH — $0022

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

Register Name and Address: TACNTL — $0023

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

R = Reserved

Figure 20-5. TIMA Counter Registers (TACNTH and TACNTL)

20.9.3 TIMA Counter Modulo Registers

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

Register Name and Address: TAMODH — $0024

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

Register Name and Address: TAMODL — $0025

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

Figure 20-6. TIMA Counter Modulo Registers (TMODH and TMODL)

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

20.9.4 TIMA Channel Status and Control Registers

Each of the TIMA channel status and control registers:

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

Register Name and Address: TASC0 — $0026

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

Register Name and Address: TASC1 — $0029

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

R = Reserved

Figure 20-7. TIMA Channel Status and Control Registers (TASC0–TASC1)

CHxF — Channel x Flag Bit

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

When CHxIE = 0, clear CHxF by reading TIMA channel x status and control register with CHxF set, and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

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

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMA CPU interrupts on channel x.

Reset clears the CHxIE bit.

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

MSxB — Mode Select Bit B

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

Setting MS0B disables the channel 1 status and control register and reverts TCH1 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 20-2**.)

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. (See **Table 20-2**.). Reset clears the MSxA bit.

1 = Initial output level low

0 = Initial output level high

NOTE: *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMA status and control register (TASC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port E, and pin PTEx/TCHx is available as a general-purpose I/O pin. **Table 20-2** shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 20-2. 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 TIMA channel register for input capture operation, make sure that the PTEx/TACHx pin is stable for at least two bus clocks.*

TOVx — Toggle-On-Overflow Bit

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

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

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 0, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100 percent. As **Figure 20-8** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100 percent duty cycle level until the cycle after CHxMAX is cleared.

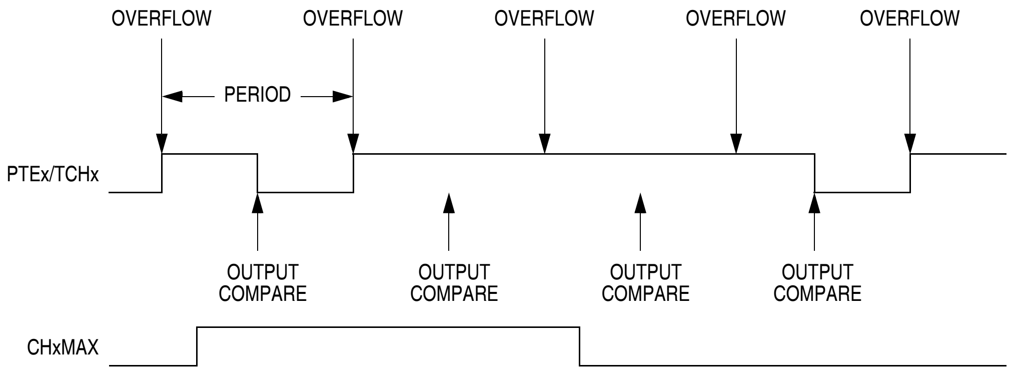


Figure 20-8. CHxMAX Latency

20.9.5 TIMA Channel Registers

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

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

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

Register Name and Address: TACH0H — $0027

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

Register Name and Address: TACH0L — $0028

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

Register Name and Address: TACH1H — $002A

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

Figure 20-9. TIMA Channel Registers (TACH0H/L–TACH1H/L)

Register Name and Address: TACH1L — $002B

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

Figure 20-9. TIMA Channel Registers (TACH0H/L–TACH1H/L) (Continued)

Section 21. Timer Interface B (TIMB)

21.1 Contents

21.2 Introduction

This section describes the timer interface module (TIMB). The TIMB is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse width modulation functions. **Figure 21-1** is a block diagram of the TIMB.

21.3 Features

Features of the TIMB include:

- Two input capture/output compare channels:
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered pulse width modulation (PWM) signal generation
- Programmable TIMB clock input:
  - 7-frequency internal bus clock prescaler selection
  - External TIMB clock input (4-MHz maximum frequency)
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIMB counter stop and reset bits
- Modular architecture expandable to eight channels

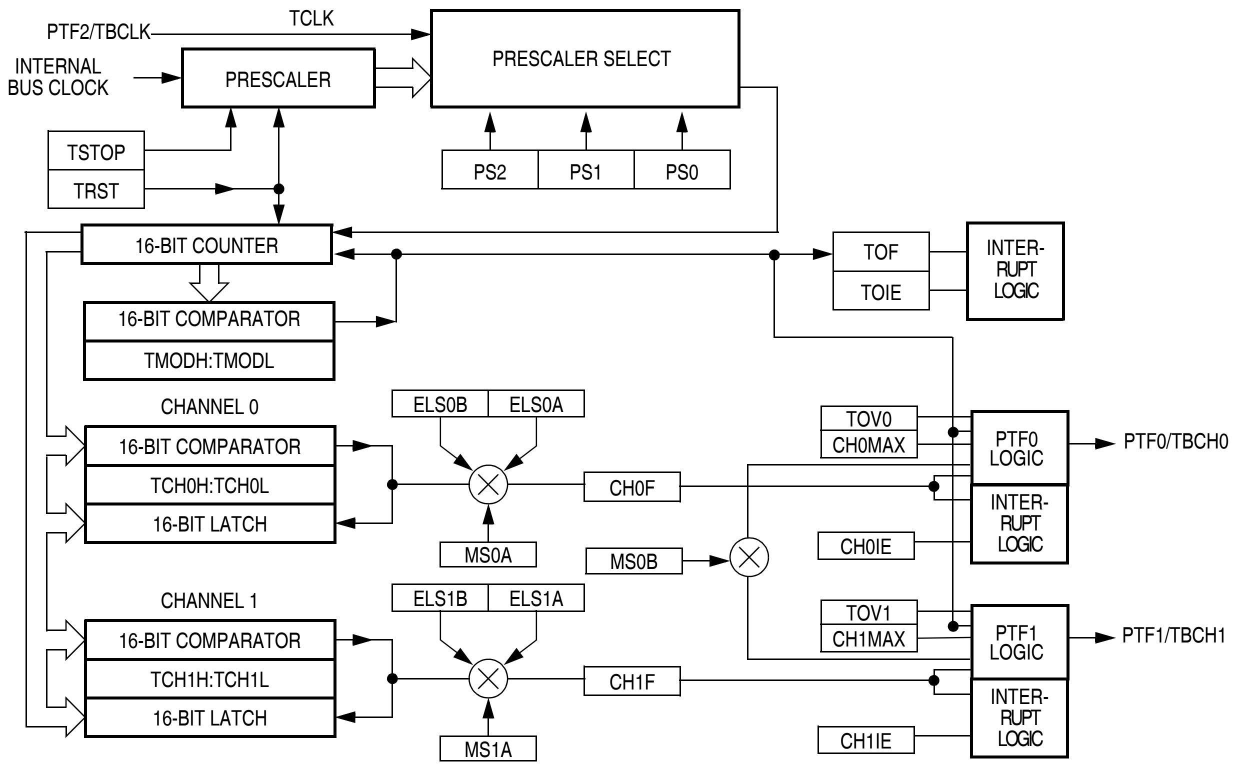


Figure 21-1. TIMB Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002C	TIMB Status and Control Register (TBSC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	R			
		Reset:	0	0	1	0	0	0	0	0
$002E	TIMB Counter Register High (TBCNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$002F	TIMB Counter Register Low (TBCNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 21-2. TIMB I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0030	TIMB Modulo Register High (TBMODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0031	TIMB Modulo Register Low (TBMODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0032	TIMB Channel 0 Status and Control Register (TBSC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0033	TIMB Channel 0 Register High (TBCH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0034	TIMB Channel 0 Register Low (TBCH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0035	TIMB Channel 1 Status and Control Register (TBSC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		R					
		Reset:	0	0	0	0	0	0	0	0
$0036	TIMB Channel 1 Register High (TBCH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0037	TIMB Channel 1 Register Low (TBCH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

R = Reserved

Figure 21-2. TIMB I/O Register Summary (Continued)

21.4 Functional Description

Figure 21-1 shows the TIMB structure. The central component of the TIMB is the 16-bit TIMB counter that can operate as a free-running counter or a modulo up-counter. The TIMB counter provides the timing reference for the input capture and output compare functions. The TIMB counter modulo registers, TBMODH–TBMODL, control the modulo value of the TIMB counter. Software can read the TIMB counter value at any time without affecting the counting sequence.

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

21.4.1 TIMB Counter Prescaler

The TIMB clock source can be one of the seven prescaler outputs or the TIMB clock pin, PTF2/TBCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMB status and control register select the TIMB clock source.

21.4.2 Input Capture

An input capture function has three basic parts:

- Edge select logic
- Input capture latch
- 16-bit counter

Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TBSC0 through TBSC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMB latches the contents of the TIMB counter into the TIMB channel registers, TCHxH–TCHxL. Input captures can generate TIMB CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The result obtained by an input capture will be one more than the value of the free-running counter on the rising edge of the internal bus clock preceding the external transition. This delay is required for internal synchronization.

The free-running counter contents are transferred to the TIMB channel status and control register (TBSC0–TBSC1, see **21.9.4 TIMB Channel Status and Control Registers**) on each proper signal transition regardless of whether the TIMB channel flag (CH0F–CH1F in TBSC0–TBSC1 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register when the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see **21.9.5 TIMB Channel Registers**). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture register.

21.4.3 Output Compare

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

21.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in **21.4.3 Output Compare**. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMB channel registers.

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

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

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

21.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTF0/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The output compare value in the TIMB channel 0 registers initially controls the output on the PTF0/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the output are the ones written to last. TSC0 controls and monitors the buffered output compare function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF1/TBCH1, is available as a general-purpose I/O pin.

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

21.4.4 Pulse Width Modulation (PWM)

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

As **Figure 21-3** shows, the output compare value in the TIMB channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMB to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIMB to set the pin if the state of the PWM pulse is logic 0.

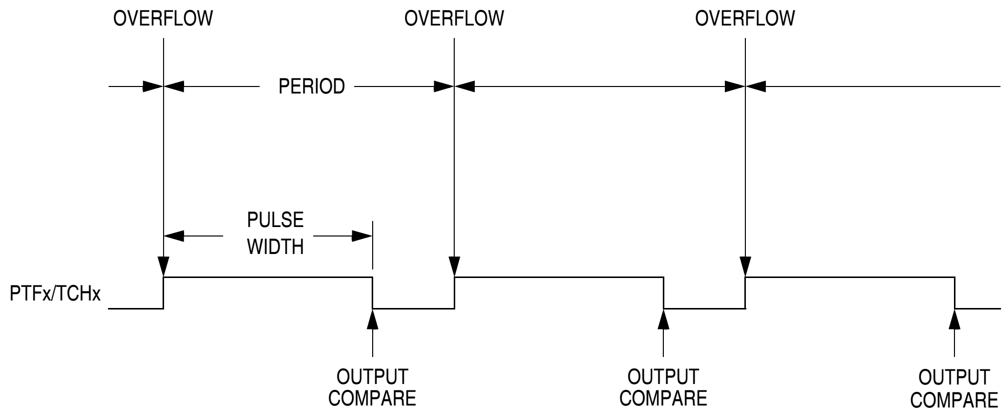


Figure 21-3. PWM Period and Pulse Width

The value in the TIMB counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMB counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see **21.9.1 TIMB Status and Control Register**).

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

21.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **21.4.4 Pulse Width Modulation (PWM)**. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMB channel registers.

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

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

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

NOTE: *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0 percent 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.*

21.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTF0/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The TIMB channel 0 registers initially control the pulse width on the PTF0/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF1/TBCH1, is available as a general-purpose I/O pin.

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

21.4.4.3 PWM Initialization

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

1. In the TIMB status and control register (TBSC):
   a. Stop the TIMB counter by setting the TIMB stop bit, TSTOP.
   b. Reset the TIMB counter by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH–TBMODL), write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH–TBCHxL), write the value for the required pulse width.
4. In TIMB channel x status and control register (TBSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA. (See **Table 21-2**.)
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See **Table 21-2**.)

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

5. In the TIMB status control register (TBSC), clear the TIMB stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H–TBCH0L) initially control the buffered PWM output. TIMB status control register 0 (TBSCR0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

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

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100 percent duty cycle output. (See **21.9.4 TIMB Channel Status and Control Registers**.)

21.5 Interrupts

These TIMB sources can generate interrupt requests:

- TIMB overflow flag (TOF) — The TOF bit is set when the TIMB counter value rolls over to $0000 after matching the value in the TIMB counter modulo registers. The TIMB overflow interrupt enable bit, TOIE, enables TIMB overflow CPU interrupt requests. TOF and TOIE are in the TIMB status and control register.
- TIMB channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMB CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

21.6 Low-Power Modes

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

21.6.1 Wait Mode

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

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

21.6.2 Stop Mode

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

21.7 TIMB During Break Interrupts

A break interrupt stops the TIMB counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See **10.8.3 SIM Break Flag Control Register**.)

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

21.8 I/O Signals

Port F shares three of its pins with the TIMB. PTF2/TBCLK is an external clock input to the TIMB prescaler. The two TIMB channel I/O pins are PTF0/TBCH0 and PTF1/TBCH1.

21.8.1 TIMB Clock Pin (PTF2/TBCLK)

PTF2/TBCLK is an external clock input that can be the clock source for the TIMB counter instead of the prescaled internal bus clock. Select the PTF2/TBCLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. (See **21.9.1 TIMB Status and Control Register**.) The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

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

See **23.8 5.0 Vdc ± 10% Serial Peripheral Interface (SPI) Timing** number 6.

The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

PTF2/TBCLK is available as a general-purpose I/O pin or ADC channel when not used as the TIMB clock input. When the PTF2/TBCLK pin is the TIMB clock input, it is an input regardless of the state of the DDRF2 bit in data direction register F.

21.8.2 TIMB Channel I/O Pins (PTF1/TBCH1–PTF0/TBCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF0/TBCH0 can be configured as a buffered output compare or a buffered PWM pin.

21.9 I/O Registers

These I/O registers control and monitor TIMB operation:

- TIMB status and control register (TBSC)
- TIMB control registers (TBCNTH–TBCNTL)
- TIMB counter modulo registers (TBMODH–TBMODL)
- TIMB channel status and control registers (TBSC0 and TBSC1)
- TIMB channel registers (TBCH0H–TBCH0L and TBCH1H–TBCH1L)

21.9.1 TIMB Status and Control Register

The TIMB status and control register:

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

Address: $002C

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

R = Reserved

Figure 21-4. TIMB Status and Control Register (TBSC)

TOF — TIMB Overflow Flag Bit

This read/write flag is set when the TIMB counter resets to $0000 after reaching the modulo value programmed in the TIMB counter modulo registers. Clear TOF by reading the TIMB status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMB overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

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

TOIE — TIMB Overflow Interrupt Enable Bit

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

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

TSTOP — TIMB Stop Bit

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

1 = TIMB counter stopped
0 = TIMB counter active

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

TRST — TIMB Reset Bit

Setting this write-only bit resets the TIMB counter and the TIMB prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the TIMB counter is reset and always reads as logic 0. Reset clears the TRST bit.

1 = Prescaler and TIMB counter cleared
0 = No effect

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

PS[2:0] — Prescaler Select Bits

These read/write bits select either the PTD4 pin or one of the seven prescaler outputs as the input to the TIMB counter as **Table 21-1** shows. Reset clears the PS[2:0] bits.

Table 21-1. Prescaler Selection

PS[2:0]	TIMB Clock Source
000	Internal bus clock ÷1
001	Internal bus clock ÷ 2
010	Internal bus clock ÷ 4
011	Internal bus clock ÷ 8
100	Internal bus clock ÷ 16
101	Internal bus clock ÷ 32
110	Internal bus clock ÷ 64
111	PTF2/TBCLK

21.9.2 TIMB Counter Registers

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

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

Register Name and Address: TBCNTH — $002E

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

Register Name and Address: TBCNTL — $002F

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

R	= Reserved

Figure 21-5. TIMB Counter Registers (TBCNTH and TBCNTL)

21.9.3 TIMB Counter Modulo Registers

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

Register Name and Address: TBMODH — $0030

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

Register Name and Address: TBMODL — $0031

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

Figure 21-6. TIMB Counter Modulo Registers (TBMODH and TBMODL)

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

21.9.4 TIMB Channel Status and Control Registers

Each of the TIMB channel status and control registers:

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

Register Name and Address: TBSC0 — $0032

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

Register Name and Address: TBSC1 — $0035

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

R = Reserved

Figure 21-7. TIMB Channel Status and Control Registers (TBSC0–TBSC1)

CHxF — Channel x Flag Bit

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

When CHxIE = 0, clear CHxF by reading TIMB channel x status and control register with CHxF set, and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

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

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMB CPU interrupts on channel x.

Reset clears the CHxIE bit.

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

MSxB — Mode Select Bit B

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

Setting MS0B disables the channel 1 status and control register and reverts TCH1 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 $\neq$ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. (See **Table 21-2**.)

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TCHx pin. (See **Table 21-2**.). Reset clears the MSxA bit.

1 = Initial output level low
0 = Initial output level high

NOTE: *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIMB status and control register (TBSC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port F, and pin PTFx/TCHx is available as a general-purpose I/O pin. **Table 21-2** shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 21-2. 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 TIMB channel register for input capture operation, make sure that the PTFx/TBCHx pin is stable for at least two bus clocks.*

TOVx — Toggle-On-Overflow Bit

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

1 = Channel x pin toggles on TIMB counter overflow.

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

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 0, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100 percent. As **Figure 21-8** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100 percent duty cycle level until the cycle after CHxMAX is cleared.

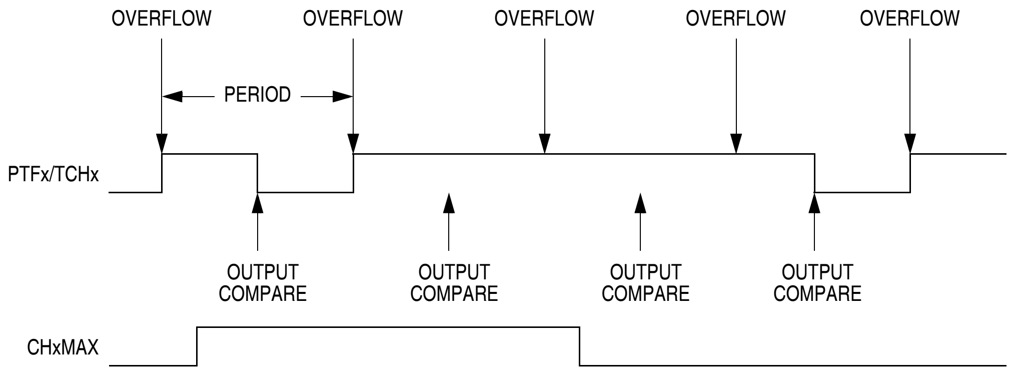


Figure 21-8. CHxMAX Latency

21.9.5 TIMB Channel Registers

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

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

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

Register Name and Address: TBCH0H — $0033

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

Register Name and Address: TBCH0L — $0034

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

Register Name and Address: TBCH1H — $0036

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

Figure 21-9. TIMB Channel Registers (TBCH0H/L–TBCH1H/L)

Register Name and Address: TBCH1L — $0037

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

Figure 21-9. TIMB Channel Registers (TBCH0H/L–TBCH1H/L) (Continued)

Section 22. CAN Controller

22.1 Contents

22.2 Introduction

The MSCAN08 is the specific implementation of the scalable controller area network (MSCAN) concept targeted for the Freescale M68HC08 Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field:

- Real-time processing
- Reliable operation in the electromagnetic interference (EMI) environment of a vehicle
- Cost-effectiveness and required bandwidth

MSCAN08 utilizes an advanced buffer arrangement, resulting in a predictable real-time behavior and simplifies the application software.

22.3 Features

Basic features of the MSCAN08 are:

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

22.4 External Pins

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

A typical CAN system with MSCAN08 is shown in **Figure 22-1**.

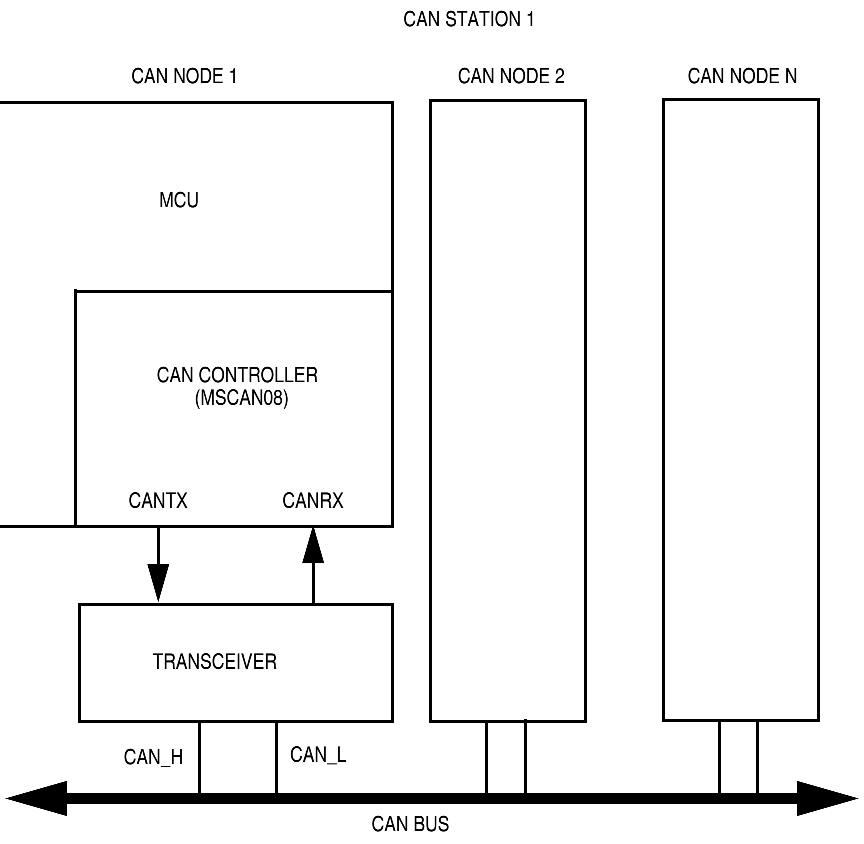


Figure 22-1. CAN System

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

22.5 Message Storage

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

22.5.1 Background

Modern application layer software is built under two fundamental assumptions:

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

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

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

Under all circumstances, at least three transmit buffers are required to meet the first of the above requirements. The MSCAN08 has three transmit buffers.

The second requirement calls for some sort of internal prioritization which the MSCAN08 implements with the "local priority" concept described in **22.5.2 Receive Structures**.

22.5.2 Receive Structures

The received messages are stored in a 2-stage input first in first out (FIFO). The two message buffers are mapped using a ping pong arrangement into a single memory area (see **Figure 22-2**). While the background receive buffer (RxBG) is exclusively associated to the MSCAN08, the foreground receive buffer (RxFG) is addressable by the CPU08. This scheme simplifies the handler software, because only one address area is applicable for the receive process.

Each buffer has 13 bytes to store the CAN control bits, the identifier (standard or extended), and the data content (for details, see See **22.13 Programmer's Model of Message Storage**.).

The receiver full flag (RXF) in the MSCAN08 receiver flag register (CRFLG) (see **22.14.5 MSCAN08 Receiver Flag Register**) signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier, this flag is set.

After the MSCAN08 successfully receives a message into the background buffer, it copies the content of RxBG into RxFG(1), sets the RXF flag, and emits a receive interrupt to the CPU(2). A new message, which may follow immediately after the IFS field of the CAN frame, will be received into RxBG.

The user's receive handler has to read the received message from RxFG and to reset the RXF flag to acknowledge the interrupt and to release the foreground buffer.

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

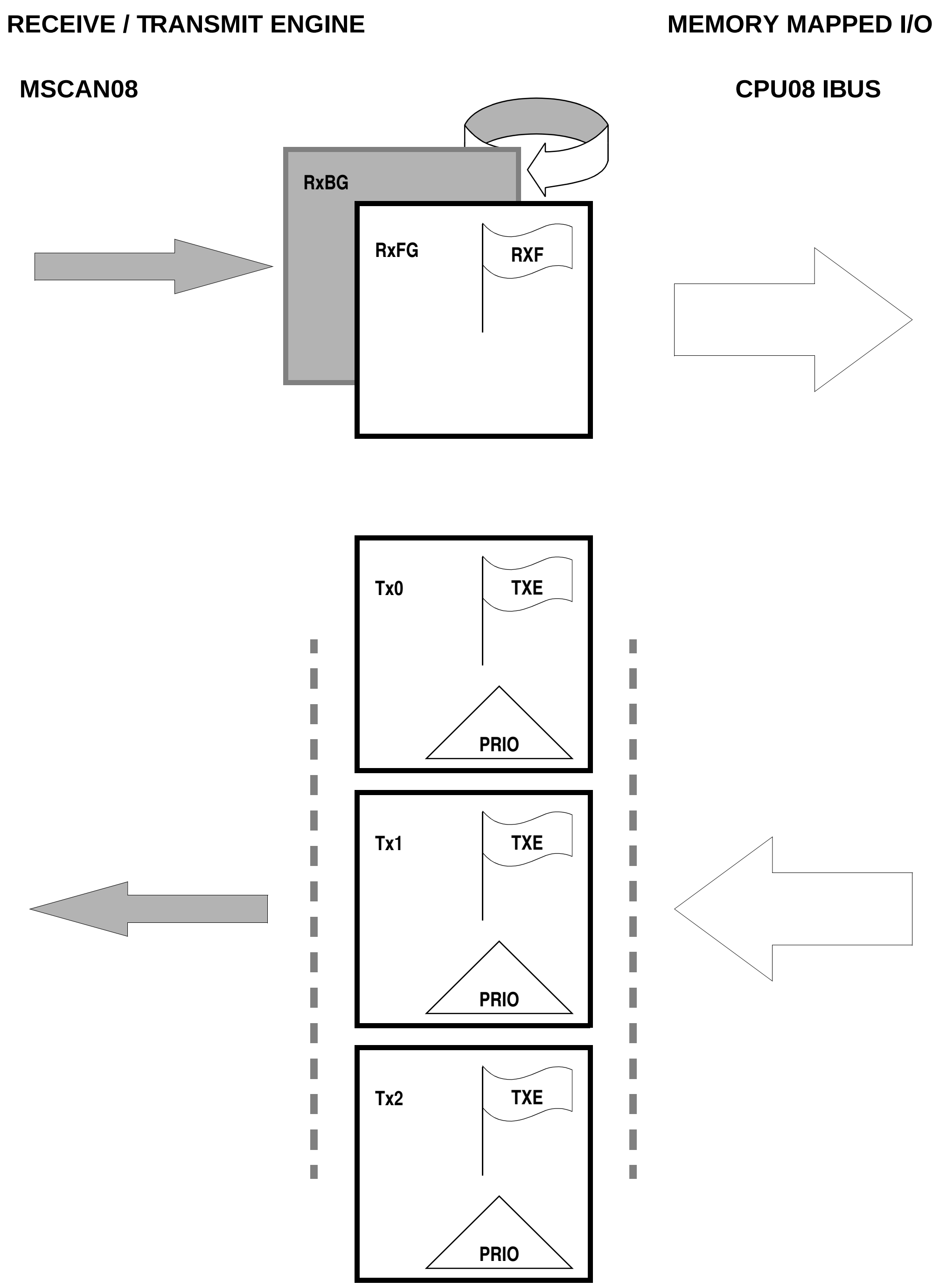


Figure 22-2. User Model for Message Buffer Organization

An overrun condition occurs when both the foreground and the background receive message buffers that are filled with correctly received messages and another message is being received from the bus. The latter message will be discarded and an error interrupt with overrun indication will occur if enabled. The over-writing of the background buffer is independent of the identifier filter function. In the overrun situation, the MSCAN08 will stay synchronized to the CAN bus. While it is able to transmit messages, all incoming messages will be discarded.

NOTE: *MSCAN08 will receive its own messages into the background receive buffer RxBG but will not overwrite RxFG and will NOT emit a receive interrupt. It also will not acknowledge (ACK) its own messages on the CAN bus. The only exception to this rule is in loop-back mode when MSCAN08 will treat its own messages exactly like all other incoming messages.*

22.5.3 Transmit Structures

The MSCAN08 has a triple transmit buffer scheme to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in **Figure 22-2**.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see **22.13 Programmer's Model of Message Storage**). An additional transmit buffer priority register (TBPR) contains an 8-bit "local priority" field (PRIO) (see **22.13.5 Transmit Buffer Priority Registers**).

To transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set transmit buffer empty (TXE) flag in the MSCAN08 transmitter flag register (CTFLG) (see **22.14.7 MSCAN08 Transmitter Flag Register**).

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

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

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

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

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

1. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

22.6 Identifier Acceptance Filter

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

- Single identifier acceptance filter to be applied to the full 29 bits of the identifier and to these bits of the CAN frame: RTR, IDE, and SRR. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier.
- Double identifier acceptance filter to be applied to:
  - The 11 bits of the identifier and the RTR bit of CAN 2.0A messages or
  - The 14 most significant bits of the identifier of CAN 2.0B messages
- Quadruple identifier acceptance filter to be applied to the first eight bits of the identifier. This mode implements four independent filters for the first eight bits of a CAN 2.0A compliant standard identifier.

The identifier acceptance registers (CIAR) define the acceptable pattern of the standard or extended identifier (ID10–ID0 or ID28–ID0). Any of these bits can be marked don't care in the identifier mask register (CIMR).

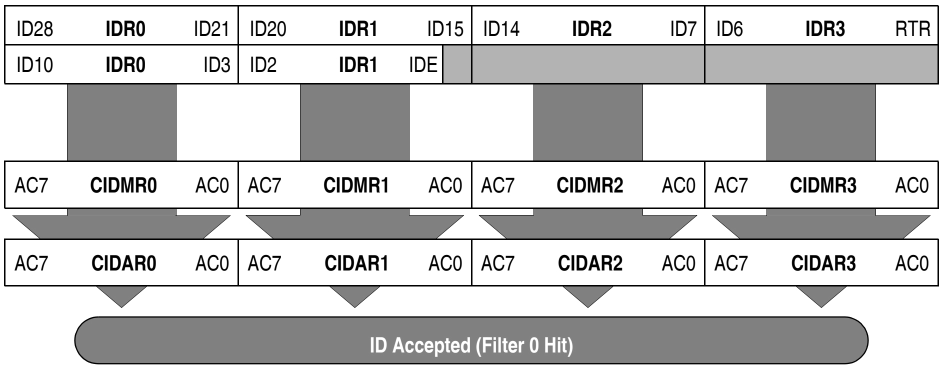


Figure 22-3. Single 32-Bit Maskable Identifier Acceptance Filter

The background buffer, RxBG, will be copied into the foreground buffer, RxFG, and the RxF flag will be set only in case of an accepted identifier (an identifier acceptance filter hit). A hit also will cause a receiver interrupt if enabled.

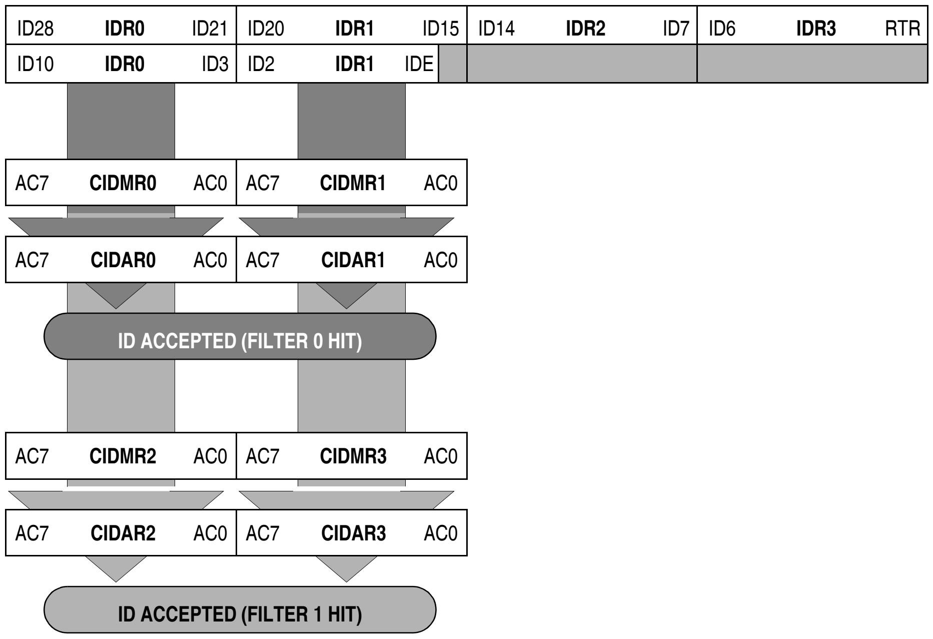


Figure 22-4. Dual 16-Bit Maskable Acceptance Filters

A filter hit is indicated to the application software by a set RXF (receiver buffer full flag, see **22.14.5 MSCAN08 Receiver Flag Register**) and two bits in the identifier acceptance control register (see **22.14.9 MSCAN08 Identifier Acceptance Control Register**). These identifier hit flags (IDHIT1–IDHIT0) clearly identify the filter section that caused the acceptance. They simplify the application software's task to identify the cause of the receiver interrupt. When more than one hit occurs (two or more filters match), the lower hit has priority.

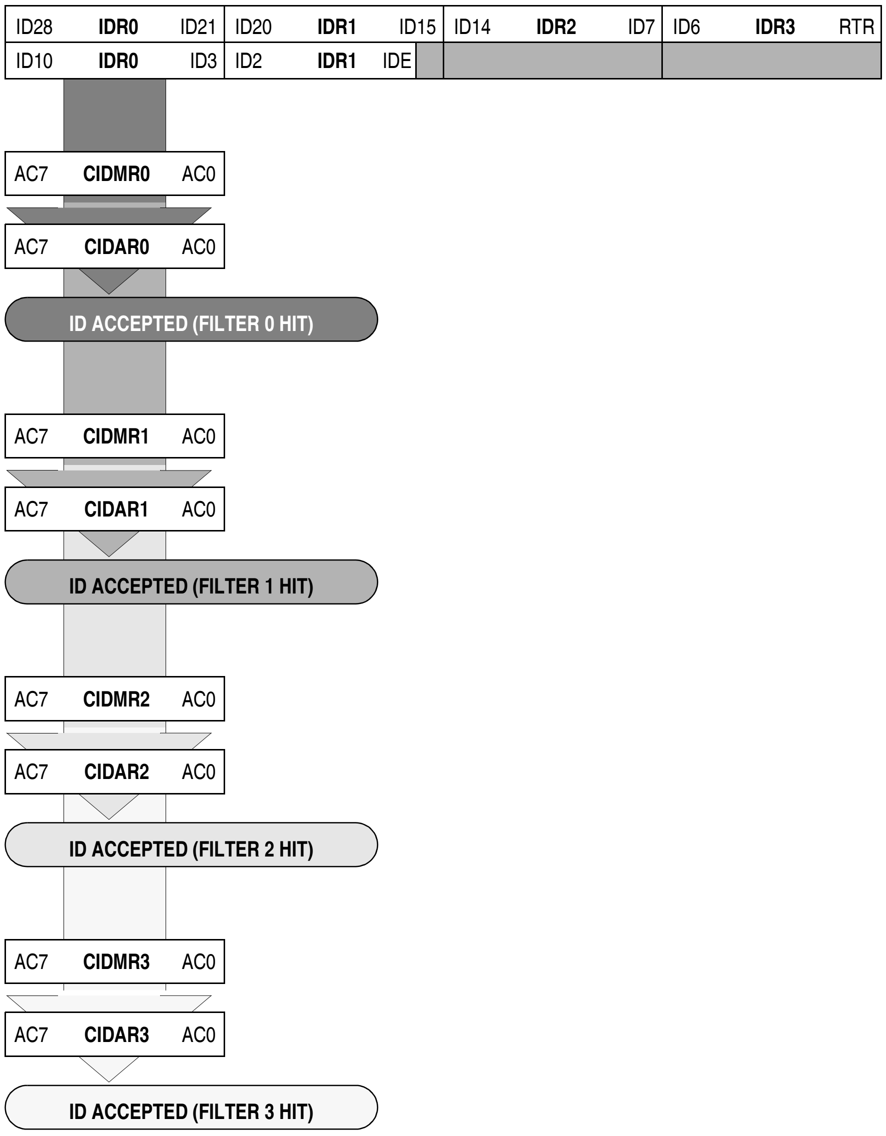


Figure 22-5. Quadruple 8-Bit Maskable Acceptance Filters

22.7 Interrupts

The MSCAN08 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked for details see **22.14.5 MSCAN08 Receiver Flag Register** to **22.14.8 MSCAN08 Transmitter Control Register**.

- Transmit interrupt — At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- Receive interrupt — A message has been received successfully and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
- Wakeup interrupt — An activity on the CAN bus occurred during MSCAN08 internal sleep mode.
- Error interrupt — An overrun, error, or warning condition occurred. The receiver flag register (CRFLG) will indicate one of the following conditions:
  - Overrun — An overrun condition as described in **22.5.2 Receive Structures** has occurred.
  - Receiver warning — The receive error counter has reached the CPU Warning limit of 96.
  - Transmitter warning — The transmit error counter has reached the CPU Warning limit of 96.
  - Receiver error passive — The receive error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
  - Transmitter error passive — The transmit error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
  - Bus off — The transmit error counter has exceeded 255 and MSCAN08 has gone to bus off state.

22.7.1 Interrupt Acknowledge

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

NOTE: *Bit manipulation instructions (BSET) shall not be used to clear interrupt flags. The OR instruction is the appropriate way to clear selected flags.*

22.7.2 Interrupt Vectors

The MSCAN08 supports four interrupt vectors as shown in **Table 22-1**. The vector addresses are dependent on the chip integration and to be defined. The relative interrupt priority is also integration dependent and to be defined.

Table 22-1. MSCAN08 Interrupt Vector Addresses

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

22.8 Protocol Violation Protection

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

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN08 can not be modified while the MSCAN08 is on-line. The SFTRES bit in the MSCAN08 module control register (see **22.14.1 MSCAN08 Module Control Register**) serves as a lock to protect these registers:
  – MSCAN08 module control register 1 (CMCR1)
  – MSCAN08 bus timing register 0 and 1 (CBTR0 and CBTR1)
  – MSCAN08 identifier acceptance control register (CIDAC)
  – MSCAN08 identifier acceptance registers (CIDAR0–CIDAR3)
  – MSCAN08 identifier mask registers (CIDMR0–CIDMR3)
- The CANTx pin is forced to recessive if the CPU goes into stop mode.

22.9 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption stand-by mode.

22.9.1 MSCAN08 Internal Sleep Mode

The CPU can request the MSCAN08 to enter the low-power mode by asserting the SLPRQ bit in the module configuration register (see **Figure 22-6**). This causes the MSCAN08 module internal clock to stop unless the module is active (such as receiving a message). The SLPAK bit indicates whether the MSCAN08 successfully went into sleep mode. The application software should use this flag as a handshake indication for the request to go into sleep mode. If not set after the request, the MSCAN08 is active and has not yet entered sleep mode. No wakeup interrupt will occur in that case.

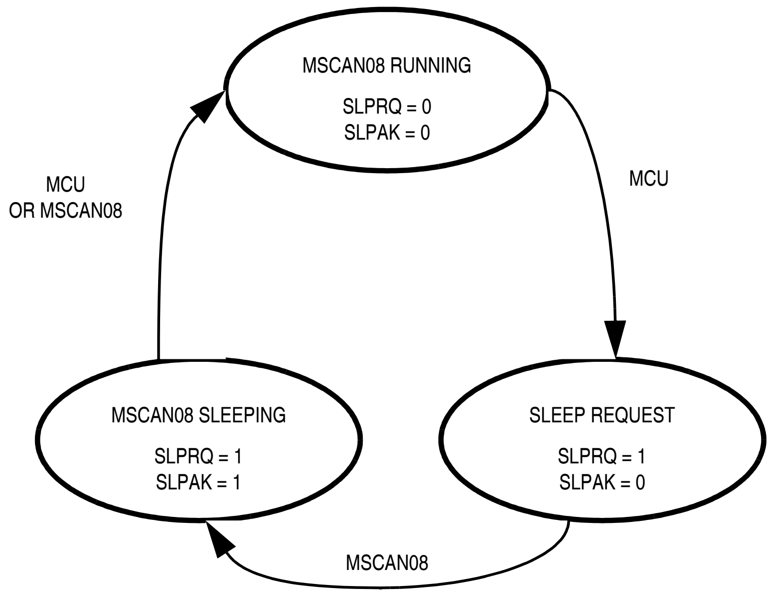


Figure 22-6. Sleep Request/Acknowledge Cycle

When in sleep mode, the MSCAN08 stops its own clocks, leaving the MCU in normal run mode.

The MSCAN08 will leave sleep mode (wakeup) when bus activity occurs or when the MCU clears the SLPRQ bit.

The CANTx pin will stay in a recessive state while the MSCAN08 is in internal sleep mode.

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

22.9.2 CPU Wait Mode

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

22.9.3 CPU Stop Mode

A CPU STOP instruction will stop the crystal oscillator, thus shutting down all system clocks. The user is responsible for ensuring that the MSCAN08 is not active when the CPU goes into stop mode. To protect the CAN bus system from fatal consequences of violations to this rule, the MSCAN08 will drive the CANTx pin into a recessive state.

The recommended procedure is to bring the MSCAN08 into sleep mode before the CPU STOP instruction is executed.

22.9.4 Programmable Wakeup Function

The MSCAN08 can be programmed to apply a low-pass filter function to the CANRx input line while in internal sleep mode (see information on control bit WUPM in **22.14.1 MSCAN08 Module Control Register**). This feature can be used to protect the MSCAN08 from wakeup due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

22.10 Timer Link

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

The previously described timer signal can be routed into the on-chip timer interface module (TIM). Under the control of the timer link enable (TLNKEN) bit in the CMCR0 will this signal be connected to the timer n channel m input.

NOTE: *The timer channel being used for the timer link is integration dependent.*

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

22.11 Clock System

Figure 22-7 shows the structure of the MSCAN08 clock generation circuitry and its interaction with the clock generation module (CGM). With this flexible clocking scheme the MSCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

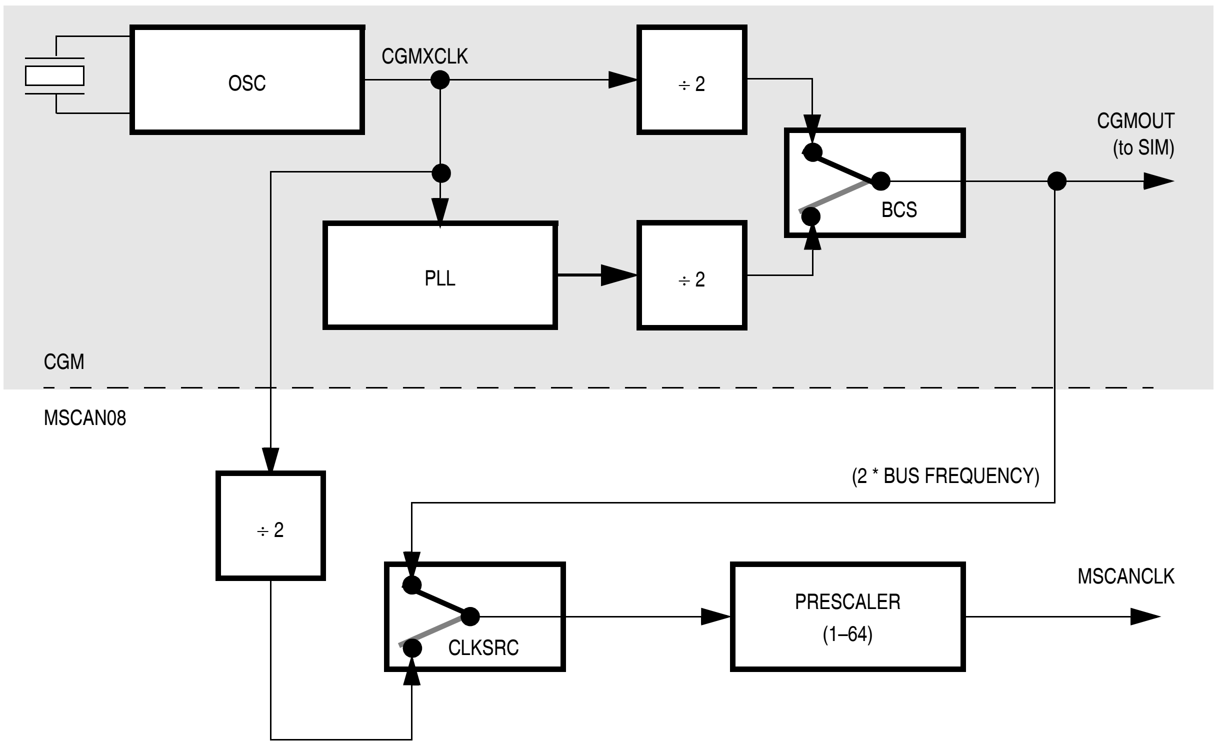


Figure 22-7. Clocking Scheme

The clock source flag (CLKSRC) in the MSCAN08 module control register (CMCR1) (see **22.14.1 MSCAN08 Module Control Register**) defines whether the MSCAN08 is connected to the output of the crystal oscillator or to the PLL output.

The MSCAN08 clock is used to generate the atomic unit of time handled by the MSCAN08: the time quantum. A bit time is subdivided into three segments defined here. For further explanation of the underlying concepts, refer to ISO/DIS 11519-1, Section 10.3.

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

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

These parameters can be set by programming the bus timing registers, CBTR0–CBTR1, see **22.14.3 MSCAN08 Bus Timing Register 0** and **22.14.4 MSCAN08 Bus Timing Register 1**).

The user is responsible for making sure that the bit time settings comply with the CAN standard. **Table 22-2** gives an overview on the CAN conforming segment settings and the related parameter values.

22.12 Memory Map

The MSCAN08 occupies 128 bytes in the CPU08 memory space. The absolute mapping is implementation dependent with the base address being a multiple of 128. The background receive buffer can be read in test mode only.

NOTE: *Due to design requirements, the absolute addresses and bit locations may change with later revisions of this specification.*

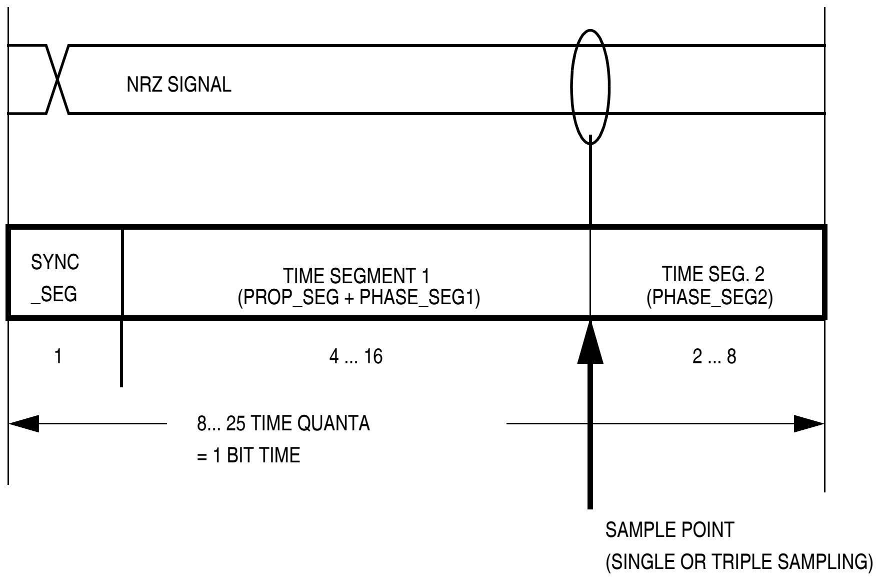


Figure 22-8. Segments within the Bit Time

Table 22-2. CAN Standard Compliant Bit Time Segment Settings

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

22.13 Programmer's Model of Message Storage

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

Addr.	Register Name
xxx0	IDENTIFIER REGISTER 0
xxx1	IDENTIFIER REGISTER 1
xxx2	IDENTIFIER REGISTER 2
xxx3	IDENTIFIER REGISTER 3
xxx4	DATA SEGMENT REGISTER 0
xxx5	DATA SEGMENT REGISTER 1
xxx6	DATA SEGMENT REGISTER 2
xxx7	DATA SEGMENT REGISTER 3
xxx8	DATA SEGMENT REGISTER 4
xxx9	DATA SEGMENT REGISTER 5
xxxA	DATA SEGMENT REGISTER 6
xxxB	DATA SEGMENT REGISTER 7
xxxC	DATA LENGTH REGISTER
xxxD	TRANSMIT BUFFER PRIORITY REGISTER(1)
xxxE	UNUSED
xxxF	UNUSED

1. Not applicable for receive buffers

Figure 22-9. Message Buffer Organization

22.13.1 Message Buffer Outline

Figure 22-10 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 22-1. All bits of the 13-byte data structure are undefined out of reset.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$xxx0	IDR0	Read: Write:	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
$xxx1	IDR1	Read: Write:	ID20	ID19	ID16	SRR(1)	IDE(1)	ID17	ID16	ID15
$xxx2	IDR2	Read: Write:	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
$xxx3	IDR3	Read: Write:	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
$xxx4	DSR0	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxx5	DSR1	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxx6	DSR2	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxx7	DSR3	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxx8	DSR4	Read: Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0

▨ = Unimplemented

Figure 22-10. Receive/Transmit Message Buffer Extended Identifier Registers

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$xxx9	DSR5	Read: / Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxxA	DSR6	Read: / Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxxB	DSR7	Read: / Write:	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
$xxxC	DLR	Read: / Write:	▨	▨	▨	▨	DLC3	DLC2	DLC1	DLC0

▨ = Unimplemented

Figure 22-10. Receive/Transmit Message Buffer Extended Identifier Registers (Continued)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$xxx0	IDR0	Read: / Write:	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3
$xxx1	IDR1	Read: / Write:	ID2	ID1	ID0	RTR	IDE(0)	▨	▨	▨
$xxx2	IDR2	Read: / Write:	▨	▨	▨	▨	▨	▨	▨	▨
$xxx3	IDR3	Read: / Write:	▨	▨	▨	▨	▨	▨	▨	▨

▨ = Unimplemented

Figure 22-11. Standard Identifier Mapping

22.13.2 Identifier Registers

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

SRR — Substitute Remote Request Bit

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

IDE — ID Extended Flag

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

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

RTR — Remote Transmission Request Flag

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

1 = Remote frame
0 = Data frame

22.13.3 Data Length Register

The data length register (DLR) keeps the data length field of the CAN frame.

DLC3–DLC0 — Data Length Code Bits

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. **Table 22-3** shows the effect of setting the DLC bits.

Table 22-3. Data Length Codes

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

22.13.4 Data Segment Registers

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

22.13.5 Transmit Buffer Priority Registers

Address: $xxxD

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

Figure 22-12. Transmit Buffer Priority Register (TBPR)

PRIO7–PRIO0 — Local Priority Bits

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

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

NOTE: *To ensure data integrity, no registers of the transmit buffers shall be written while the associated TXE flag is cleared.*

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

22.14 Programmer's Model of Control Registers

The programmer's model has been laid out for maximum simplicity and efficiency. Figure 22-13 gives an overview on the control register block of the MSCAN08.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0500	CAN Module Control Register 0 (CMCR0)	Read:	0	0	0	SYNCH	TLNKEN	SLPAK	SLPRQ	SFTRES
		Write:	R	R	R	R		R		
		Reset:	0	0	0	0	0	0	0	0
$0501	CAN Module Control Register 1 (CMCR1)	Read:	0	0	0	0	0	LOOPB	WUPM	CLKSRC
		Write:	R	R	R	R	R			
		Reset:	0	0	0	0	0	0	0	0
$0502	CAN Bus Timing Register 0 (CBTR0)	Read:	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0503	CAN Bus Timing Register 1 (CBTR1)	Read:	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0504	CAN Receiver Flag Register (CRFLG)	Read:	WUPIF	RWRNIF	TWRNIF	RERRIF	TERRIF	BOFFIF	OVRIF	RXF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0505	CAN Receiver Interrupt Enable Register (CRIER)	Read:	WUPIE	RWRNIE	TWRNIE	RERRIE	TERRIE	BOFFIE	OVRIE	RXFIE
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0506	CAN Transmitter Flag Register (CTFLG)	Read:	0	ABTAK2	ABTAK1	ABTAK0	0	TXE2	TXE1	TXE0
		Write:	R	R	R	R	R			
		Reset:	0	0	0	0	0	1	1	1

R = Reserved

Figure 22-13. MSCAN08 Control Register Struture (Sheet 1 of 3)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0507	CAN Transmitter Control Register (CTCR)	Read:	0	ABTRQ2	ABTRQ1	ABTRQ0	0	TXEIE2	TXEIE1	TXEIE0
		Write:	R				R			
		Reset:	0	0	0	0	0	1	1	1
$0508	CAN Identifier Acceptance Control Register (CIDAC)	Read:	0	0	IDAM1	IDAM0	0	0	IDHIT1	IDHIT0
		Write:	R	R			R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$050E	CAN Receiver Error Counter Register (CRXERR)	Read:	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$050F	CAN Transmitter Error Counter Register (CTXERR)	Read:	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$0510	CAN Identifier Acceptance Register 0 (CIDAR0)	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0511	CAN Identifier Acceptance Register 1 (CIDAR1)	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0512	CAN Identifier Acceptance Register 2 (CIDAR2)	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0513	CAN Identifier Acceptance Register 3 (CIDAR3)	Read:	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
		Write:								
		Reset:	Unaffected by reset							
$0514	CAN Identifier Mask Register 0 (CIDMR0)	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							

R = Reserved

Figure 22-13. MSCAN08 Control Register Struture (Sheet 2 of 3)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0515	CAN Identifier Mask Register 1 (CIDMR1)	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0516	CAN Identifier Mask Register 0 (CIDMR0)	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							
$0517	CAN Identifier Mask Register 1 (CIDMR1)	Read:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
		Write:								
		Reset:	Unaffected by reset							

R = Reserved

Figure 22-13. MSCAN08 Control Register Struture (Sheet 3 of 3)

22.14.1 MSCAN08 Module Control Register

Address: $0500

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

R = Reserved

Figure 22-14. CAN Module Control Register 0 (CMCR0)

SYNCH — Synchronized Status Bit

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

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

TLNKEN — Timer Enable Flag

This flag is used to establish a link between the MSCAN08 and the on-chip timer (see **22.10 Timer Link**).

1 = The MSCAN08 timer signal output is connected to the timer.
0 = No connection

SLPAK — Sleep Mode Acknowledge Flag

This flag indicates whether the MSCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see **22.9.1 MSCAN08 Internal Sleep Mode**).

1 = Sleep – MSCAN08 in internal sleep mode
0 = Wakeup – MSCAN08 will function normally

SLPRQ — Sleep Request, Go to Internal Sleep Mode Flag

This flag allows a request for the MSCAN08 to go into an internal power-saving mode (see **22.9.1 MSCAN08 Internal Sleep Mode**).

1 = Sleep — The MSCAN08 will go into internal sleep mode if and as long as there is no activity on the bus.
0 = Wakeup — The MSCAN08 will function normally. If SLPAK is cleared by the CPU, then the MSCAN08 will wake up, but will not issue a wakeup interrupt.

SFTRES — Soft Reset Bit

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

These registers will go into the same state as out of hard reset: CMCR0, CRFLG, CRIER, CTFLG, and CTCR.

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

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

1 = MSCAN08 in soft reset state
0 = Normal operation

22.14.2 MSCAN08 Module Control Register

Address: $0501

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

R = Reserved

Figure 22-15. CAN Module Control Register 1 (CMCR1)

LOOPB — Loop Back Self-Test Mode Bit

When this bit is set, the MSCAN08 performs an internal loop back which can be used for self-test operation and the bit stream output of the transmitter is fed back to the receiver. The CANRx input pin is ignored and the CANTx output goes to the recessive state (1). Note that in this state, the MSCAN08 ignores the ACK bit to ensure proper reception of its own message and will treat messages being received while in transmission as received messages from remote nodes.

1 = Activate loop back self-test mode
0 = Normal operation

WUPM — Wakeup Mode Flag

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN08 from spurious wakeups (see **22.9.4 Programmable Wakeup Function**).

1 = MSCAN08 will wake up the CPU only in cases of a dominant pulse on the bus which has a length of at least t_{wup}.
0 = MSCAN08 will wake up the CPU after any recessive to dominant edge on the CAN bus.

CLKSRC — Clock Source Flag

This flag defines which clock source the MSCAN08 module is driven from (see **22.11 Clock System**).

1 = The MSCAN08 clock source is CGMOUT (see **Figure 22-7**).
0 = The MSCAN08 clock source is CGMXCLK/2 (see **Figure 22-7**).

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

22.14.3 MSCAN08 Bus Timing Register 0

Address:	$0502							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
Reset:	0	0	0	0	0	0	0	0

Figure 22-16. CAN Bus Timing Register 0 (CBTR0)

SJW1 and SJW0 — Synchronization Jump Width Bits

The synchronization jump width (SJW) defines the maximum number of system clock (t_{SCL}) cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see **Table 22-4**).

Table 22-4. Synchronization Jump Width

SJW1	SJW0	Synchronization Jump Width
0	0	1 t_{SCL} cycle
0	1	2 t_{SCL} cycles
1	0	3 t_{SCL} cycles
1	1	4 t_{SCL} cycles

BRP5–BRP0 — Baud Rate Prescaler Bits

These bits determine the MSCAN08 system clock cycle time (t_{SCL}), which is used to build up the individual bit timing, according to **Table 22-5**.

Table 22-5. Baud Rate Prescaler

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

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

22.14.4 MSCAN08 Bus Timing Register 1

Address: $0503

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

Figure 22-17. CAN Bus Timing Register 1 (CBTR1)

SAMP — Sampling Bit

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

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

TSEG22–TSEG20 and TSEG13–TSEG10 — Time Segment Bits

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point.

Table 22-6. Time Segment Syntax

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

Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in **Table 22-7** and **Table 22-8**, respectively.

Table 22-7. Time Segment 1 Values

TSEG13	TSEG12	TSEG11	TSEG10	Time Segment 1
0	0	0	0	1 t_{SCL} cycle
0	0	0	1	2 t_{SCL} cycles
0	0	1	0	3 t_{SCL} cycles
0	0	1	1	4 t_{SCL} cycles
. . .	. . .	. . .	. . .	. . .
1	1	1	1	16 t_{SCL} cycles

Table 22-8. Time Segment 2 Values

TSEG22	TSEG21	TSEG20	Time Segment 2
0	0	0	1 t_{SCL} cycle
0	0	1	2 t_{SCL} cycles
. . .	. . .	. . .	. . .
1	1	1	8 t_{SCL} cycles

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of bus clock cycles (t_{SCL}) per bit as shown in **Table 22-7** and **Table 22-8**.

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

22.14.5 MSCAN08 Receiver Flag Register

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

Address: $0504

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

Figure 22-18. CAN Receiver Flag Register (CRFLG)

WUPIF — Wakeup Interrupt Flag

If the MSCAN08 detects bus activity while it is asleep, it clears the SLPAKSLPAK bit in the CMCR0 register; the WUPIF bit will then be set. If not masked, a wakeup interrupt is pending while this flag is set.

1 = MSCAN08 has detected activity on the bus and requested wake up.

0 = No wake up interrupt has occurred.

RWRNIF — Receiver Warning Interrupt Flag

This bit will be set when the MSCAN08 went into warning status because the receive error counter was in the range of 96 to 127. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into warning status.

0 = No warning interrupt has occurred.

TWRNIF — Transmitter Warning Interrupt Flag

This bit will be set when the MSCAN08 went into warning status because the transmit error counter was in the range of 96 to 127. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into warning status.

0 = No warning interrupt has occurred.

RERRIF — Receiver Error Passive Interrupt Flag

This bit will be set when the MSCAN08 went into error passive status because the receive error counter exceeded 127. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into error passive status.
0 = No warning interrupt has occurred.

TERRIF — Transmitter Error Passive Interrupt Flag

This bit will be set when the MSCAN08 went into error passive status due to the transmit error counter exceeding 127. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into error passive status.
0 = No warning interrupt has occurred.

BOFFIF — Bus-Off Interrupt Flag

This bit will be set when the MSCAN08 went into bus-off status, because the transmit error counter exceeded 255. If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 went into warning status.
0 = No warning interrupt has occurred.

OVRIF — Overrun Interrupt Flag

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

1 = A data overrun has been detected.
0 = No data overrun has occurred.

RXF — Receive Buffer Full Flag

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

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

22.14.6 MSCAN08 Receiver Interrupt Enable Register

Address: $0505

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

Figure 22-19. CAN Receiver Interrupt Enable Register (CRIER)

WUPIE — Wakeup Interrupt Enable Bit
1 = A wakeup event will result in a wakeup interrupt.
0 = No interrupt will be generated from this event.

RWRNIE — Receiver Warning Interrupt Enable Bit
1 = A receiver warning status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

TWRNIE — Transmitter Warning Interrupt Enable Bit
1 = A transmitter warning status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

RERRIE — Receiver Error Passive Interrupt Enable Bit
1 = A receiver error passive status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

TERRIE — Transmitter Error Passive Interrupt Enable Bit
1 = A transmitter error passive status event will result in an error interrupt.
0 = No interrupt will be generated from this event.

BOFFIE — Bus-Off Interrupt Enable Bit
1 = A bus-off event will result in an error interrupt.
0 = No interrupt will be generated from this event.

OVRIE — Overrun Interrupt Enable Bit

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

RXFIE — Receiver Full Interrupt Enable Bit

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

22.14.7 MSCAN08 Transmitter Flag Register

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

Address: $0506

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

R = Reserved

Figure 22-20. CAN Transmitter Flag Register (CTFLG)

ABTAK2–ABTAK0 — Abort Acknowledge Flag

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent. The flag is reset implicitly whenever the associated TXE flag is set to 0.

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

TXE2–TXE0 — Transmitter Empty Flag

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN08 will set the flag after the message has been sent successfully. The flag also will be set by the MSCAN08 when the transmission request was successfully aborted due to a pending abort request (see **22.13.5 Transmit Buffer Priority Registers**). If not masked, a receive interrupt is pending while this flag is set.

A reset of this flag also will reset the abort acknowledge (ABTAK) and the abort request (ABTRQ) flags of the particular buffer. See **22.14.8 MSCAN08 Transmitter Control Register**.

1 = The associated message buffer is empty (not scheduled).
0 = The associated message buffer is full (loaded with a message due for transmission).

22.14.8 MSCAN08 Transmitter Control Register

Address: $0507

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

R = Reserved

Figure 22-21. CAN Transmitter Control Register (CTCR)

ABTRQ2–ABTRQ0 — Abort Request Flags

The CPU sets this flag to request that an already scheduled message buffer (TXE = 0) be aborted. The MSCAN08 will grant the request when the message is not already under transmission. When a message is aborted the associated TXE and the abort acknowledge flag (ABTAK) (see **22.14.7 MSCAN08 Transmitter Flag Register**) will be set and an TXE interrupt will occur if enabled. The CPU cannot reset this flag. The flag is reset implicitly whenever the associated TXE flag is set.

1 = Abort request pending
0 = No abort request

TXEIE2–TXEIE0 — Transmitter Empty Interrupt Enable Bits

1 = A transmitter empty (transmit buffer available for transmission) event will result in a transmitter empty interrupt.
0 = No interrupt will be generated from this event.

22.14.9 MSCAN08 Identifier Acceptance Control Register

Address: $0508

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

R = Reserved

Figure 22-22. CAN Identifier Acceptance Control Register (CIDAC)

IDAM1–IDAM0— Identifier Acceptance Mode Flags

The CPU sets these flags to define the identifier acceptance filter organization (see **22.6 Identifier Acceptance Filter**). **Table 22-7** summarizes the different settings. In “filter closed” mode no messages will be accepted so that the foreground buffer will never be reloaded.

Table 22-9. Identifier Acceptance Mode Settings

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

IDHIT1–IDHIT0— Identifier Acceptance Hit Indicator Flags

The MSCAN08 sets these flags to indicate an identifier acceptance hit (see **22.6 Identifier Acceptance Filter**). **Table 22-7** summarizes the different settings.

Table 22-10. Identifier Acceptance Hit Indication

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

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

NOTE: *The CIDAC register can be written only if the SFTRES bit in the MSCAN08 module control register is set.*

22.14.10 MSCAN08 Receive Error Counter Register

Address: $050E

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

R = Reserved

Figure 22-23. CAN Receiver Error Counter Register (CRXERR)

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

22.14.11 MSCAN08 Transmit Error Counter Register

Address: $050F

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

R = Reserved

Figure 22-24. CAN Transmit Error Counter Register (CTXERR)

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

NOTE: *For both error counters, there is no hardware synchronization between the write accesses to those registers from the MSCAN08 side and the read accesses by the CPU. It is the user's responsibility to verify that a stable value has been read by executing a second validation read and comparing the two values.*

22.14.12 MSCAN08 Identifier Acceptance Registers

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

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

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers only the first two (IDAR0 and IDAR1) are applied. In the latter case, the mask register, CIDMR1, the three last bits (AC2–AC0) must be programmed to don't care.

Register Name and Address: CIDAR0 — $0510

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

Register Name and Address: CIDAR1 — $0511

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

Figure 22-25. CAN Identifier Acceptance Registers (CIDAR0–CIDAR3)

Register Name and Address: CIDAR2 — $0512

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

Register Name and Address: CIDAR3 — $0513

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

Figure 22-25. CAN Identifier Acceptance Registers (CIDAR0–CIDAR3) (Continued)

AC7–AC0 — Acceptance Code Bits

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

NOTE: *The CIDAR0–CIDAR3 registers can be written only if the SFTRES bit in the MSCAN08 module control register is set.*

22.14.13 MSCAN08 Identifier Mask Registers

The identifier mask registers specify which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering.

Register Name and Address: CIDMRO — $0514

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Register Name and Address: CIDMR1 — $0515

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Register Name and Address: CIDMR2 — $0516

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Register Name and Address: CIDMR3 — $0517

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AM7	AM6	AM5	AM4	AM3	AM2	AM1	AM0
Reset:	Unaffected by reset							

Figure 22-26. CAN Identifier Mask Registers (CIDMR0–CIDMR3)

AM7–AM0 — Acceptance Mask Bits

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

1 = Ignore corresponding acceptance code register bit.
0 = Match corresponding acceptance code register and identifier bits.

NOTE: *The CIDMR0-CIDMR3 registers can be written only if the SFTRES bit in the MSCAN08 module control register is set.*

Section 23. Electrical Specifications

23.1 Contents

23.2 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* ***23.5 DC Electrical Characteristics*** *for guaranteed operating conditions.*

Rating	Symbol	Value	Unit
Supply voltage	V_{DD}	–0.3 to +6.0	V
Input voltage	V_{in}	V_{SS} –0.3 to V_{DD} +0.3	V
Maximum current per pin excluding V_{DD} and V_{SS}	I	± 25	mA
Storage temperature	T_{stg}	–55 to +150	°C
Maximum current out of V_{SS}	I_{MVSS}	100	mA
Maximum current into V_{DD}	I_{MVDD}	100	mA
Reset $\overline{IRQ}$ input voltage	V_{HI}	V_{DD} to V_{DD} + 2	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}).*

23.3 Functional Operating Range

Rating	Symbol	Value	Unit
Operating temperature range	T_A	–40 to 125	°C
Operating voltage range	V_{DD}	$5.0 \pm 10\%$	V

23.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance PLCC (52 pins)	θ_{JA}	50	°C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273°C)$	W
Constant(2)	K	$P_D \times (T_A + 273°C) + (P_D^2 \times \theta_{JA})$	W/°C
Average junction temperature	T_J	$T_A = P_D \times \theta_{JA}$	°C
Maximum junction temperature	T_{JM}	150	°C

1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined from a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

23.5 DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage (I_{Load} = –2.0 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, $\overline{IRQS}$, RESET, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input low voltage Port A, $\overline{IRQS}$, RESET, OSC1 Port B Ports C, D, E, F	V_{IL}	V_{SS}	— — —	0.3 x V_{DD} 0.16 x V_{DD} 0.2 x V_{DD}	V
V_{DD} + V_{DDA}/V_{DDAREF} supply current Run[3] Wait[4] Stop[5] –40°C to +125°C –40°C to +125°C with LVI enabled	I_{DD}	 — — — —	 — — — —	 35 15 100 500	 mA mA µA µA
I/O ports hi-z leakage current	I_L	—	—	± 1	µA
Input current	I_{In}	—	—	± 1	µA
Capacitance Ports (as input or output)	C_{Out} C_{In}	— —	— —	12 8	pF
Low-voltage reset inhibit	V_{LVII}	4.0	4.5	4.75	V
Low-voltage reset inhibit/recover hysteresis	H_{LVI}	—	200	—	mV
POR re-arm voltage[6]	V_{POR}	0	—	200	mV
POR reset voltage[7]	V_{PORRST}	0	—	800	mV
POR rise time ramp rate[8]	R_{POR}	0.02	—	—	V/ms
High COP disable voltage[9]	V_{HI}	V_{DD}	—	V_{DD} + 2	V

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.
3. Run (operating) I_{DD} measured using external square wave clock source (f_{OP} = 8.4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled.
4. Wait I_{DD} measured using external square wave clock source (f_{OP} = 8.4 MHz). All inputs 0.2 Vdc from rail. No dc loads. Less than 100 pF on all outputs, C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait I_{DD}. Measured with all modules enabled.
5. Stop I_{DD} measured with OSC1 = V_{SS}.
6. Maximum is highest voltage that POR is guaranteed.
7. Maximum is highest voltage that POR is possible.
8. If minimum V_{DD} is not reached before the internal POR reset is released, RST must be driven low externally until minimum V_{DD} is reached.
9. See **14.9 COP Module During Break Interrupts**.

23.6 Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Bus operating frequency (4.5–5.5 V — V_{DD} only)	f_{BUS}	—	8.4	MHz
$\overline{\text{RESET}}$ 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	t_{ILIL}	Note 3	—	t_{cyc}
EEPROM programming time per byte	t_{EEPGM}	10	—	ms
EEPROM erasing time per byte	t_{EBYTE}	10	—	ms
EEPROM erasing time per block	t_{EBLOCK}	10	—	ms
EEPROM erasing time per bulk	t_{EBULK}	10	—	ms
EEPROM programming voltage discharge period	t_{EEFPV}	100	—	µs
Cumulative program time per program cycle[2]	—	—	100	ms
16-bit timers[3] Input capture pulse width[4] Input capture period[5]	 t_{TH}, t_{TL} t_{TLTL}	 2 Note 5	 — —	 t_{cyc}

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted.
2. Programming is done in an iterative program / verify loop which is exited upon a successful verify.
3. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
4. Refer to **Table 20-2 . Mode, Edge, and Level Selection** and supporting note.
5. 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}.

23.7 ADC Characteristics

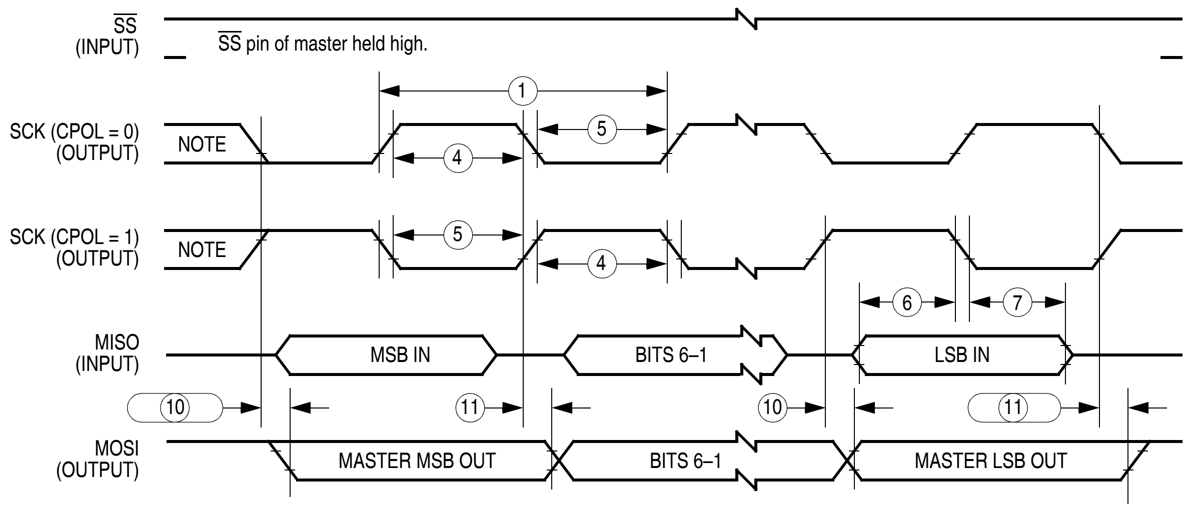
Characteristic	Min	Max	Unit	Comments
Resolution	8	8	Bits	
Absolute accuracy (V_{REFL} = 0 V, V_{DDA} = V_{REFH} = 5 V ± 10%)	–1	+1	LSB	Includes quantization
Conversion range[1]	V_{REFL}	V_{REFH}	V	V_{REFL} = V_{SSA}
Powerup time	16	17	µs	Conversion Time period
Input leakage[2] Ports B and D	—	± 1	µA	
Conversion time	16	17	ADC clock cycles	Includes sampling time
Monotonicity	Inherent within total error			
Zero input reading	00	01	Hex	V_{In} = V_{REFL}
Full-scale reading	FE	FF	Hex	V_{In} = V_{REFH}
Sample time[3]	5	—	ADC clock cycles	
Input capacitance	—	8	pF	Not tested
ADC internal clock	500 k	1.048 M	Hz	Tested only at 1 MHz
Analog input voltage	–0.3	V_{DD} + 0.3	V	

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. The external system error caused by input leakage current is approximately equal to the product of R source and input current.
3. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.

23.8 5.0 Vdc ± 10% Serial Peripheral Interface (SPI) Timing

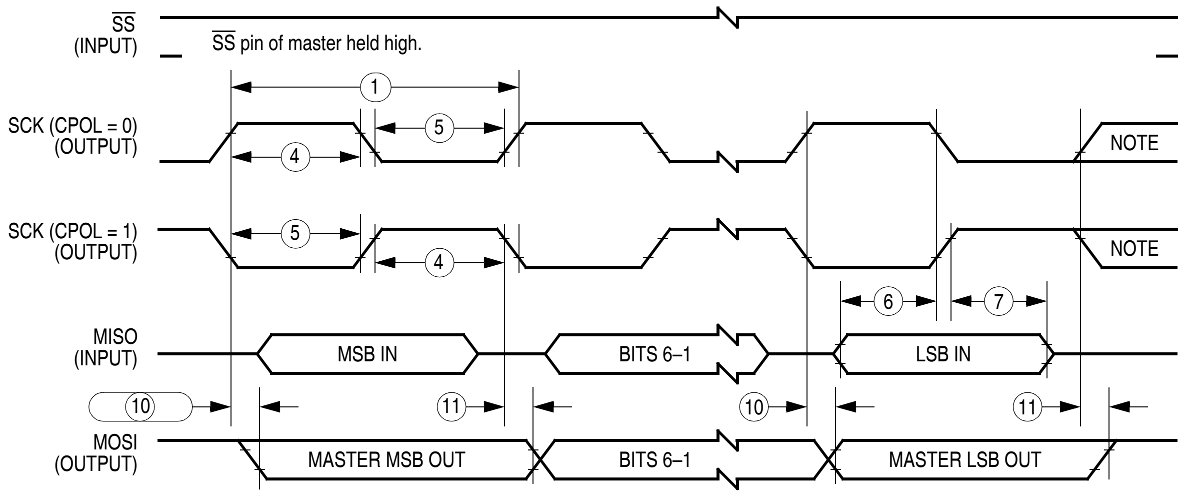
Num [1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency[3] Master Slave	$f_{BUS(M)}$ $f_{BUS(S)}$	f_{Bus}/128 dc	f_{Bus}/2 f_{Bus}	MHz
1	Cycle tme Master Slave	$t_{CYC(M)}$ $t_{CYC(S)}$	2 1	128 —	t_{CYC}
2	Enable lead time	t_{Lead}	15	—	ns
3	Enable lag time	t_{Lag}	15	—	ns
4	Clock (SCK) high time Master Slave	$t_{W(SCKH)M}$ $t_{W(SCKH)S}$	100 50	— —	ns
5	Clock (SCK) low time Master Slave	$t_{W(SCKL)M}$ $t_{W(SCKL)S}$	100 50	— —	ns
6	Data setup time, inputs Master Slave	$t_{SU(M)}$ $t_{SU(S)}$	45 5	— —	ns
7	Data hold time, inputs Master Slave	$t_{H(M)}$ $t_{H(S)}$	0 15	— —	ns
8	Access time, slave[4] CPHA = 0 CPHA = 1	$t_{A(CP0)}$ $t_{A(CP1)}$	0 0	40 20	ns
9	Slave disable time, hold time to high-impedance State[5]	t_{DIS}	—	25	ns
10	Data valid time after enable edge Master Slave	$t_{V(M)}$ $t_{V(S)}$	— —	10 40	ns
11	Data hold time, outputs, after enable edge Master Slave	$t_{HO(M)}$ $t_{HO(S)}$	0 5	— —	ns

1. Item numbers refer to dimensions in **Figure 23-1** and **Figure 23-2**.
2. All timing is shown with respect to 30% V_{DD} and 70% V_{DD}, unless otherwise noted; assumes 100 pF load on all SPI pins.
3. f_{Bus} = the currently active bus frequency for the microcontroller.
4. Time to data active from high-impedance state
5. Hold time to high-impedance state



Note: This first clock edge is generated internally, but is not seen at the SCK pin.

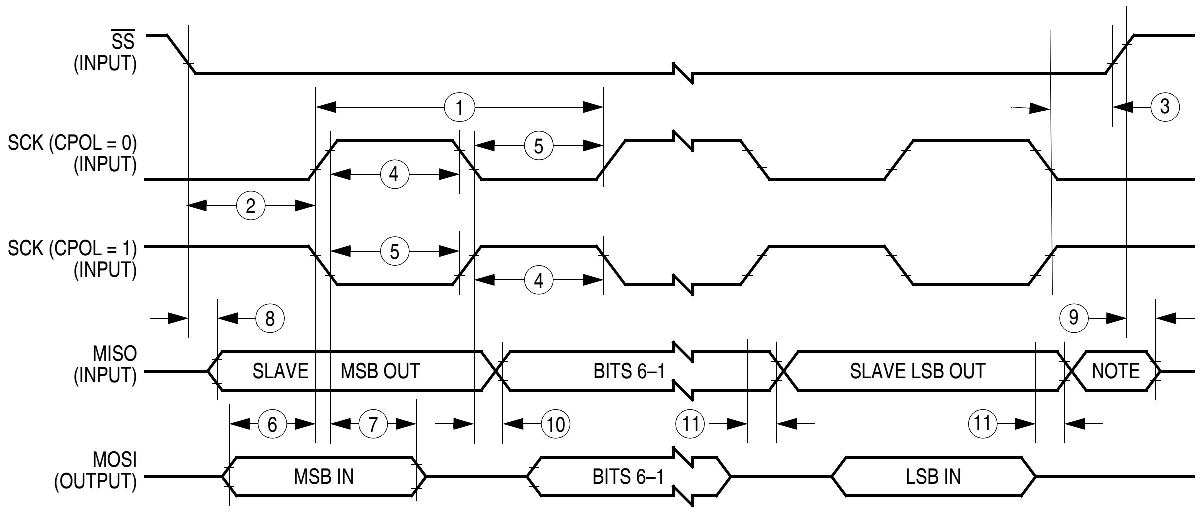
a) SPI Master Timing (CPHA = 0)



Note: This last clock edge is generated internally, but is not seen at the SCK pin.

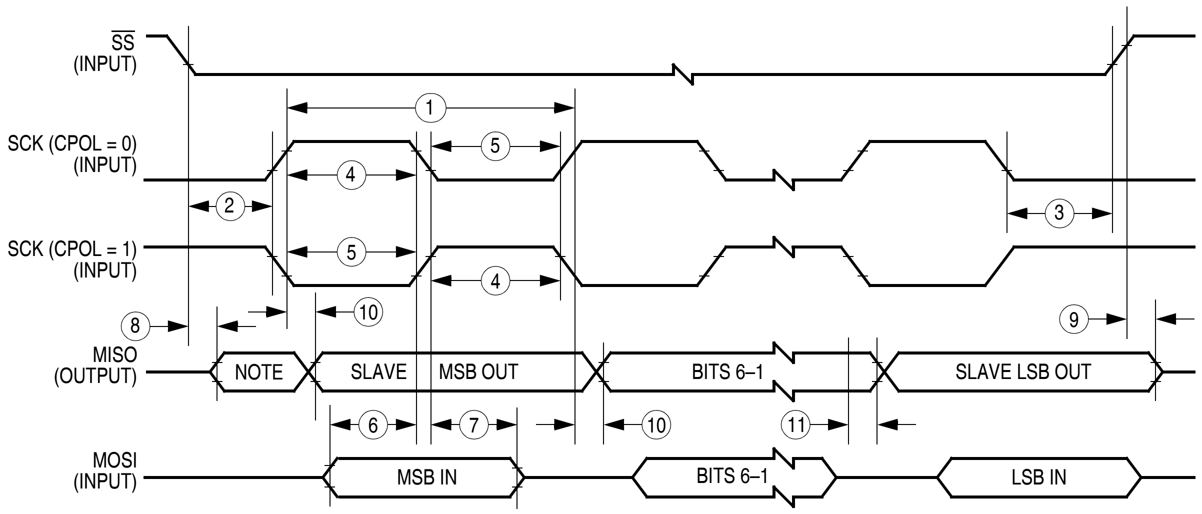
b) SPI Master Timing (CPHA = 1)

Figure 23-1. SPI Master Timing



Note: Not defined but normally MSB of character just received

a) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

b) SPI Slave Timing (CPHA = 1)

Figure 23-2. SPI Slave Timing

23.9 CGM Operating Conditions

Characteristic	Symbol	Min	Typ	Max	Comments
Operating voltage	V_{DD}	4.5 V	—	5.5 V	
Crystal reference frequency	f_{XCLK}	1	4.194 MHz	8 MHz	
Range nominal multiplier (MHz)	f_{NOM}	—	4.194	—	4.5–5.5 V, V_{DD} only
VCO operating frequency (MHz)	f_{VCLK}	4.9152 MHz	—	32.0 MHz	

23.10 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 multiply factor	C_{Fact}		0.0154		F/s V
Filter capacitor	C_F	—	$C_{Fact} \times (V_{DDA}/f_{XCLK})$	—	See **9.5.3 External Filter Capacitor Pin (CGMXFC)**
Bypass capacitor	C_{BYP}	—	0.1 μF	—	C_{BYP} must provide low AC impedance from $f = f_{XCLK}/100$ to $100 \times f_{VCLK}$, so series resistance must be considered.

23.11 CGM Acquisition/Lock Time Information

Description[1]	Symbol	Min	Typ	Max	Notes
Tracking mode entry frequency tolerance	Δ_{TRK}	0	—	± 3.6%	
Acquisition mode entry frequency tolerance	Δ_{UNT}	± 6.3%	—	± 7.2%	
Lock entry frequency tolerance	Δ_{Lock}	0	—	± 0.9%	
Lock exit frequency tolerance	Δ_{UNL}	± 0.9%	—	± 1.8%	
Reference cycles per acquisition mode measurement	n_{ACQ}	—	32	—	
Reference cycles per tracking mode measurement	n_{TRK}	—	128	—	
Automatic Lock Timer[2], [3]	t_{Lock}	—	4.35 ms	7.00 ms	
PLL jitter, deviation of average bus frequency over 2 ms[4]	—	0	—	± (f_{CRYS}) x (.025%) x (N/4)	N = VCO Freq. Mult. (GBNT)

1. V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = –40°C to +125°C, unless otherwise noted.
2. For Typ = 25°C, N/L = 66. For Max: T_A = –40°C, N/L = 77
3. This parameter is periodically sampled rather than 100% tested.
4. GBNT guaranteed but not tested

23.12 Timer Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Input capture pulse width	t_{TIH}, t_{TIL}	125	—	ns
Input clock pulse width	t_{TCH}, t_{TCL}	$(1/f_{OP}) + 5$	—	ns

23.13 Memory Characteristics

Characteristic	Symbol	Min	Max	Unit
RAM data retention voltage	V_{RDR}	0.7	—	V
EEPROM erase / program cycles 10 ms write time @ 125°C	—	—	10,000	Cycles
EEPROM data retention After 10,000 erase / program cycles at 125°C	—	10	—	Years

Section 24. Mechanical Data

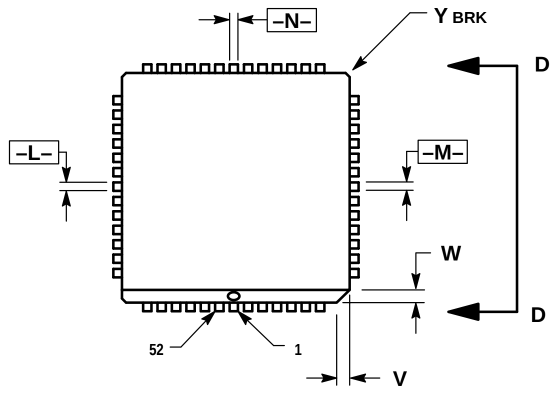
24.1 Contents

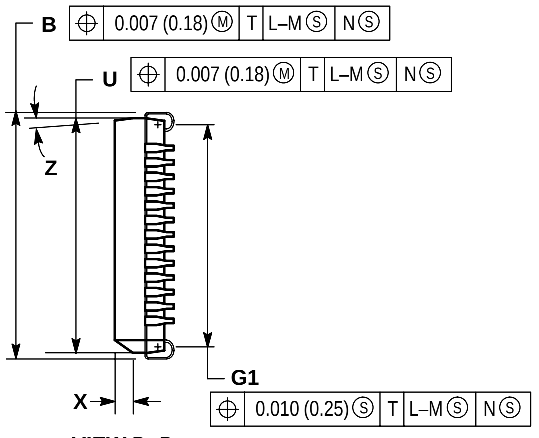
24.2 Introduction

This section describes the dimensions of the 52-pin plastic leaded chip carrier package.

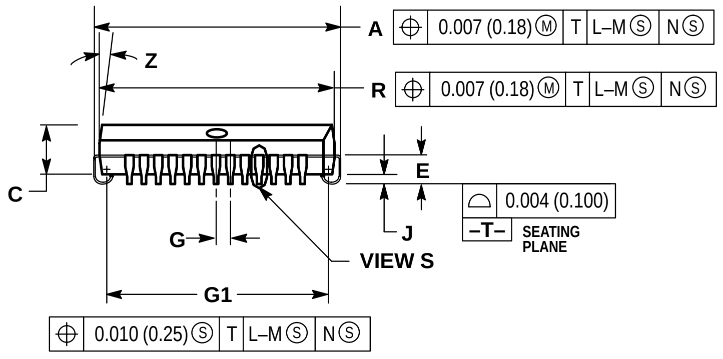
The following figure shows the latest package at the time of this publication. To make sure that you have the latest package specifications, please visit the Freescale website at http://freescale.com. Follow World-Wide Web on-line instructions to retrieve the current mechanical specifications.

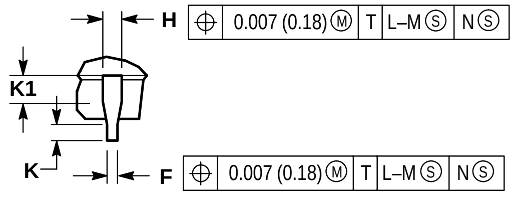
24.3 52-Pin Plastic Leaded Chip Carrier Package (Case 778)




B
0.007 (0.18) Ⓜ T L–M Ⓢ N Ⓢ
U
0.007 (0.18) Ⓜ T L–M Ⓢ N Ⓢ
Z
G1
X
0.010 (0.25) Ⓢ T L–M Ⓢ N Ⓢ

VIEW D–D


A
0.007 (0.18) Ⓜ T L–M Ⓢ N Ⓢ
Z
R
0.007 (0.18) Ⓜ T L–M Ⓢ N Ⓢ
C
E
0.004 (0.100)
–T–
SEATING PLANE
J
G
VIEW S
G1
0.010 (0.25) Ⓢ T L–M Ⓢ N Ⓢ



VIEW S

NOTES:
1. DATUMS –L–, –M–, AND –N– DETERMINED WHERE TOP OF LEAD SHOULDER EXITS PLASTIC BODY AT MOLD PARTING LINE.
2. DIMENSION G1, TRUE POSITION TO BE MEASURED AT DATUM –T–, SEATING PLANE.
3. DIMENSIONS R AND U DO NOT INCLUDE MOLD FLASH. ALLOWABLE MOLD FLASH IS 0.010 (0.250) PER SIDE.
4. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
5. CONTROLLING DIMENSION: INCH.
6. THE PACKAGE TOP MAY BE SMALLER THAN THE PACKAGE BOTTOM BY UP TO 0.012 (0.300). DIMENSIONS R AND U ARE DETERMINED AT THE OUTERMOST EXTREMES OF THE PLASTIC BODY EXCLUSIVE OF MOLD FLASH, TIE BAR BURRS, GATE BURRS AND INTERLEAD FLASH, BUT INCLUDING ANY MISMATCH BETWEEN THE TOP AND BOTTOM OF THE PLASTIC BODY.
7. DIMENSION H DOES NOT INCLUDE DAMBAR PROTRUSION OR INTRUSION. THE DAMBAR PROTRUSION(S) SHALL NOT CAUSE THE H DIMENSION TO BE GREATER THAN 0.037 (0.940). THE DAMBAR INTRUSION(S) SHALL NOT CAUSE THE H DIMENSION TO BE SMALLER THAN 0.025 (0.635).

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.785	0.795	19.94	20.19
B	0.785	0.795	19.94	20.19
C	0.165	0.180	4.20	4.57
E	0.090	0.110	2.29	2.79
F	0.013	0.019	0.33	0.48
G	0.050 BSC		1.27 BSC	
H	0.026	0.032	0.66	0.81
J	0.020	—	0.51	—
K	0.025	—	0.64	—
R	0.750	0.756	19.05	19.20
U	0.750	0.756	19.05	19.20
V	0.042	0.048	1.07	1.21
W	0.042	0.048	1.07	1.21
X	0.042	0.056	1.07	1.42
Y	—	0.020	—	0.50
Z	2°	10°	2°	10°
G1	0.710	0.730	18.04	18.54
K1	0.040	—	1.02	—

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Rev. 0.1
MC68HC08QA24/D
August 5, 2005