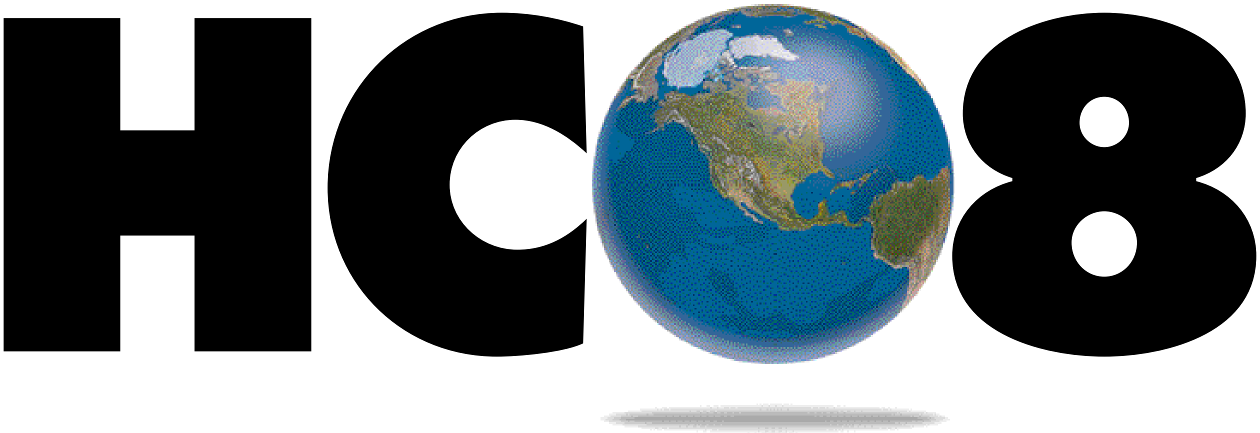
MC68HC08XL36/D



MC68HC08XL36

HCMOS Microcontroller Unit

TECHNICAL DATA

Home Page:
www.freescale.com
email:
support@freescale.com
USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
(800) 521-6274
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
+81 2666 8080
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
(800) 441-2447
303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor
@hibbertgroup.com

RoHS-compliant and/or Pb- free versions of Freescale products have the functionality and electrical characteristics of their non-RoHS-compliant and/or non-Pb- free counterparts. For further information, see http://www.freescale.com or contact your Freescale sales representative.

For information on Freescale.s Environmental Products program, go to http://www.freescale.com/epp.

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.
Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including “Typicals” must be validated for each customer application by customer’s technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

List of Sections

Table of Contents

Introduction

Memory Map

Random Access Memory (RAM)

Read-Only Memory (ROM)

Central Processor Unit (CPU)

Resets and Interrupts

Low-Power Modes

Clock Generator Module (CGM)

Direct Memory Access Module (DMA)

Break Module (BRK)

Monitor ROM (MON)

Timer Interface Module (TIM)

Serial Peripheral Interface Module (SPI)

Serial Communications Interface Module (SCI)

Input/Output Ports

Computer Operating Properly Module (COP)

External Interrupt Module (IRQ)

Keyboard Interrupt Module (KBI)

Low-Voltage Inhibit Module (LVI)

Specifications

Introduction

Contents

Features

Features of the MC68HC08XL36 include the following:

- High-Performance M68HC08 Architecture
- Fully Upward-Compatible Object Code with M6805, M146805, and M68HC05 Families
- 8-MHz Internal Bus Frequency
- 36 Kbytes of On-Chip Read-Only Memory (ROM)
- On-Chip Programming Firmware for Use with Host Personal Computer
- ROM Data Security
- One Kbyte of On-Chip Random-Access Memory (RAM)
- Serial Peripheral Interface Module (SPI)
- Serial Communications Interface Module (SCI)
- 16-Bit, 4-Channel Timer Interface Module (TIM)
- 3-Channel Direct Memory Access Module (DMA)
- Clock Generator Module (CGM)
- System Protection Features
  - Optional Computer Operating Properly (COP) Reset
  - Low-Voltage Detection with Optional Reset
  - Illegal Opcode Detection with Optional Reset
  - Illegal Address Detection with Optional Reset
- 56-Pin Plastic Shrink Dual-In-Line Package (SDIP) or 64-Pin Plastic Quad Flat Pack (QFP)
- Low-Power Design (Fully Static with Stop and Wait Modes)
- Master Reset Pin and Power-On Reset
- 8-Bit Keyboard Wakeup Port

Features of the CPU08 include the following:

- Enhanced HC05 Programming Model
- Extensive Loop Control Functions
- 16 Addressing Modes (Eight More Than the HC05)
- 16-Bit Index Register and Stack Pointer
- Memory-to-Memory Data Transfers
- Fast 8 × 8 Multiply Instruction
- Fast 16/8 Divide Instruction
- Binary-Coded Decimal (BCD) Instructions
- Optimization for Controller Applications
- Third Party C Language Support

Mask Options

Table 1. Mask Options

Feature	Mask Options	
COP[1] Operation	Enabled	Disabled
COP Timeout Period	$2^{18} – 2^4$ CGMXCLK Cycles	$2^{13} – 2^4$ CGMXCLK Cycles
LVI[2] Operation	Enabled	Disabled
LVI Reset	Enabled	Disabled
LVI Operation in Stop Mode	Enabled	Disabled
Stop Mode Recovery Time	4096 CGMXCLK Cycles	32 CGMXCLK Cycles
STOP Instruction	Legal Opcode	Illegal Opcode

[1] See Computer Operating Properly Module (COP) on page 293.

[2] See Low-Voltage Inhibit Module (LVI) on page 317.

MCU Block Diagram

Figure 1 shows the MCU structure.

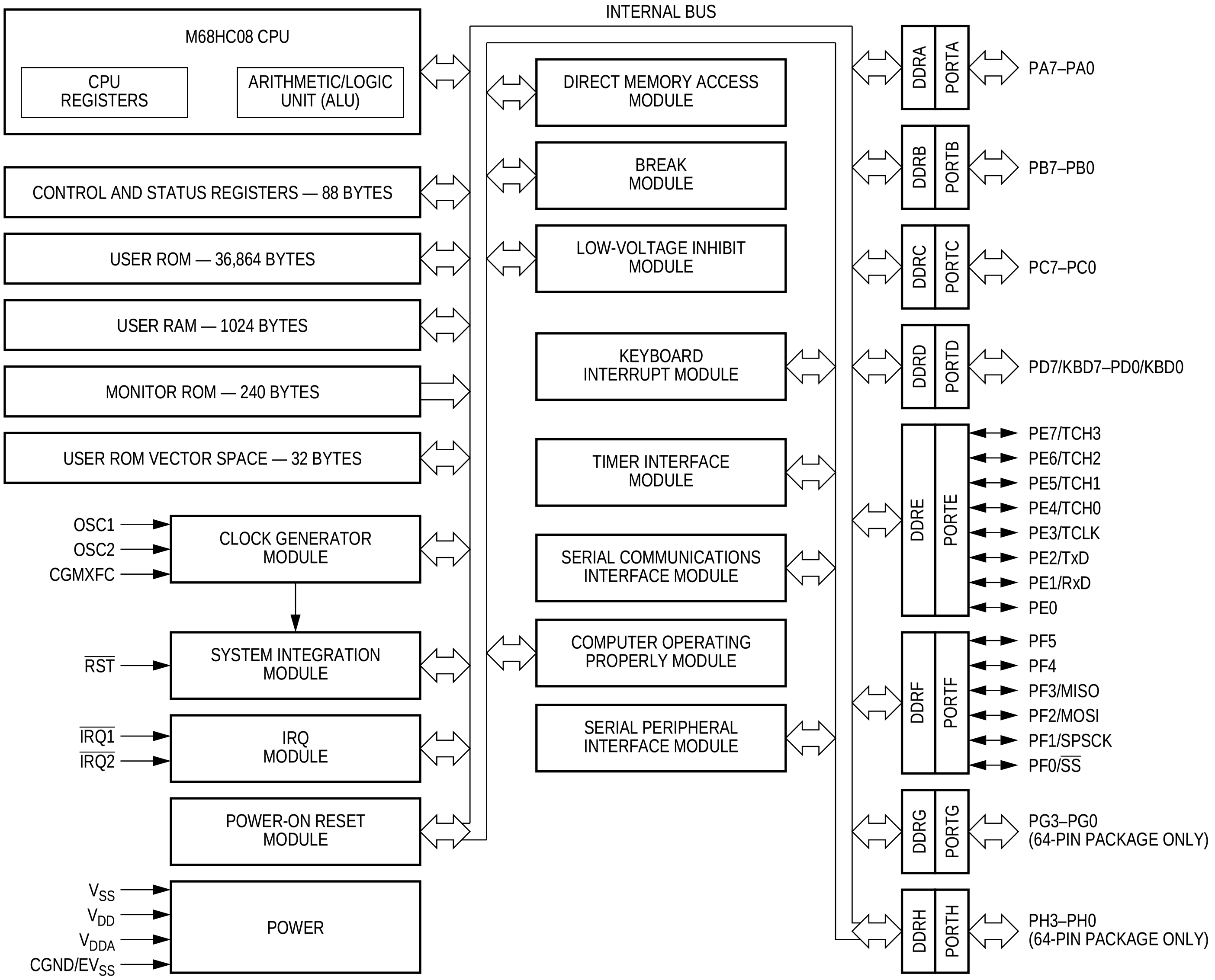


Figure 1. MCU Block Diagram

4-intro_a

Pin Assignments

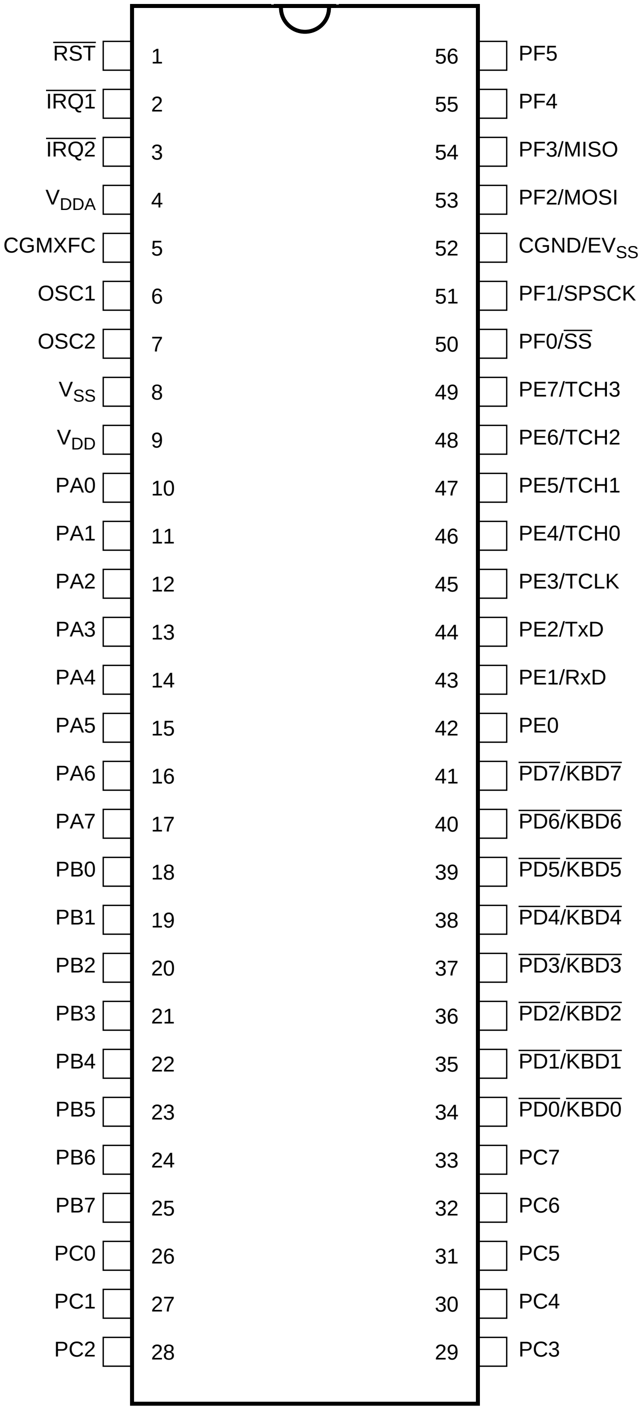
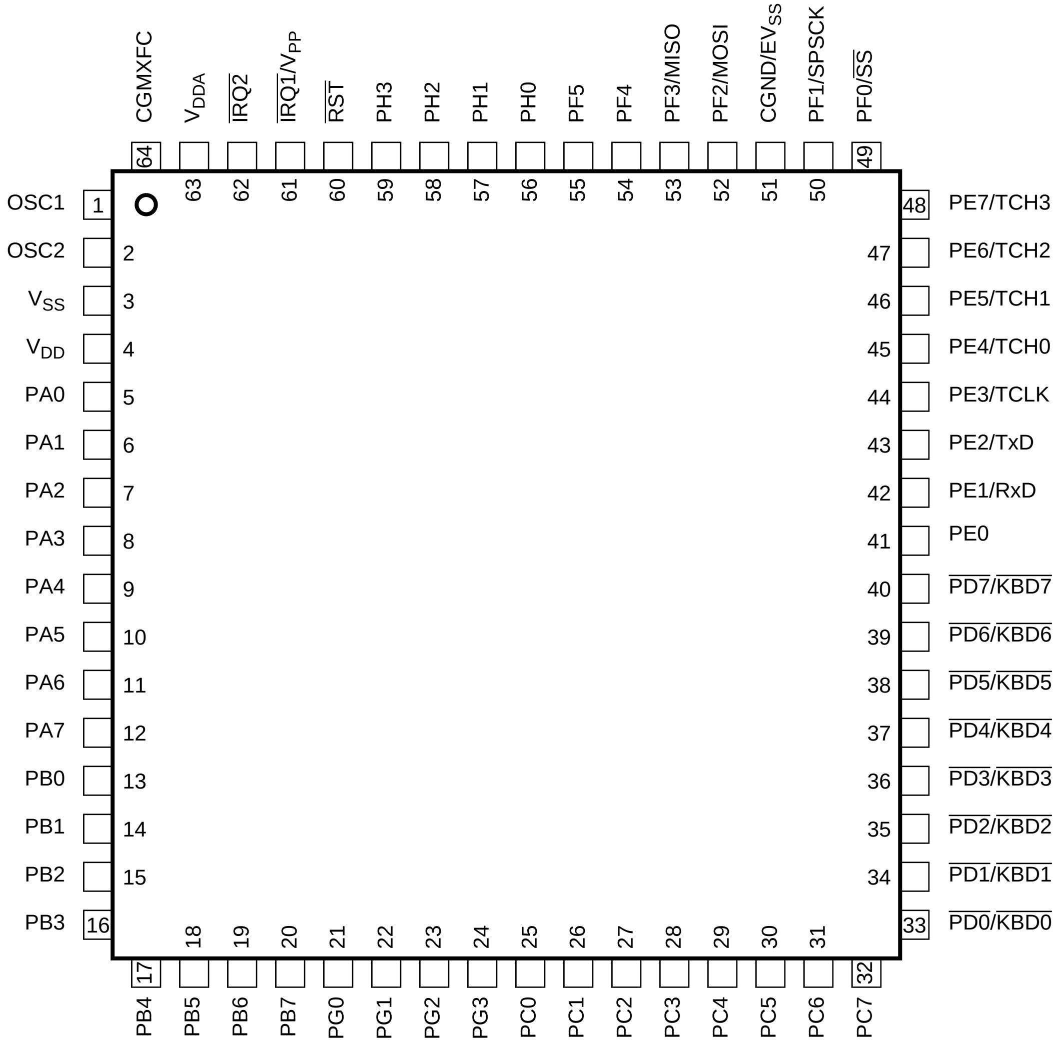


Figure 2. SDIP Pin Assignments



NOTE: Ports G and H are available only with the QFP.

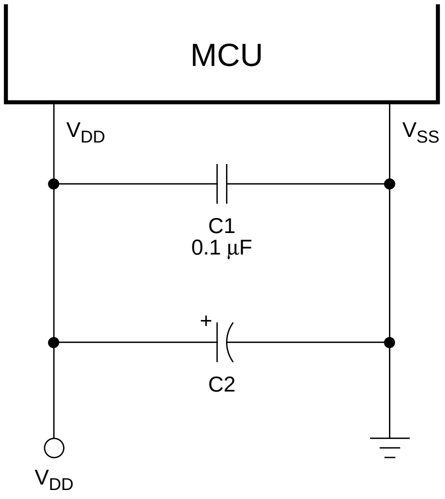
Figure 3. QFP Pin Assignments

Pin Functions

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



NOTE: Component values shown
represent typical applications.

Figure 4. Power Supply Bypassing

Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. (See Clock Generator Module (CGM) on page 81.)

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.

External Interrupt Pins ($\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$)

$\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ are asynchronous external interrupt pins. (See External Interrupt Module (IRQ) on page 299.)

Clock Ground Pin (CGND/EV$_{SS}$)

CGND/EV$_{SS}$ is the ground for the port output buffers and the ground return for the serial clock in the SPI. (See Serial Peripheral Interface Module (SPI) on page 191.)

NOTE: *CGND/EV$_{SS}$ must be grounded for proper MCU operation.*

CGM Power Supply Pin (V$_{DDA}$)

V$_{DDA}$ is the power supply pin for the analog portion of the CGM. (See Clock Generator Module (CGM) on page 81.)

External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. (See Clock Generator Module (CGM) on page 81.)

Port A Input/Output (I/O) Pins (PA7–PA0)

PA7–PA0 are general-purpose bidirectional I/O port pins. (See Input/Output Ports on page 271.)

Port B I/O Pins (PB7–PB0)

PB7–PB0 are general-purpose bidirectional I/O port pins. (See Input/Output Ports on page 271.)

Port C I/O Pins (PC7–PC0)

PC7–PC0 are general-purpose bidirectional I/O port pins. (See Input/Output Ports on page 271.)

Port D I/O Pins ($\overline{\text{PD7}}$/$\overline{\text{KBD7}}$–$\overline{\text{PD0}}$/$\overline{\text{KBD0}}$)

$\overline{\text{PD7}}$/$\overline{\text{KBD7}}$–$\overline{\text{PD0}}$/$\overline{\text{KBD0}}$ are general-purpose bidirectional I/O port pins. Any or all of the port D pins can be programmed to serve as keyboard interrupt pins. (See Input/Output Ports on page 271.)

Port E I/O Pins (PE7/TCH3–PE0)

Port E is an 8-bit special function port that shares five of its pins with the TIM and two of its pins with the SCI. (See **Timer Interface Module (TIM)** on page 163, **Serial Communications Interface Module (SCI)** on page 227, and **Input/Output Ports** on page 271.)

Port F I/O Pins (PF5–PF0/$\overline{\text{SS}}$)

Port F is a 6-bit special function port that shares four of its pins with the SPI. (See **Serial Peripheral Interface Module (SPI)** on page 191 and **Input/Output Ports** on page 271.)

Port G I/O Pins (PG3–PG0)

PG3–PG0 are general-purpose bidirectional I/O pins. (See Input/Output Ports on page 271.) Port G is available only with the 64-pin package.

Port H I/O Pins (PH3–PH0)

PH3–PH0 are general-purpose bidirectional I/O pins. (See Input/Output Ports on page 271.) Port H is available only with the 64-pin package.

Ordering Information

MCU Ordering Forms

To initiate an order for a ROM-based MCU, first obtain the current ordering form for the MCU from a Motorola representative. Submit the following items when ordering MCUs:

- A current MCU ordering form that is **completely filled out** (Contact your Motorola sales office for assistance.)
- A copy of the customer specification if the customer specification deviates from the Motorola specification for the MCU
- Customer's application program on one of the media listed in **Application Program Media** on page 20**.**

The current MCU ordering form is also available through the Motorola Freeware Bulletin Board Service (BBS). The telephone number is (512) 891-FREE. After making the connection, type bbs in lowercase letters. Then press the return key to start the BBS software.

Application Program Media

Please deliver the application program to Motorola in one of the following media:

- Macintosh®[1] 3-1/2-inch diskette (double-sided 800 K or double-sided high-density 1.4 M)
- MS-DOS®[2] or PC-DOS™[3] 3-1/2-inch diskette (double-sided 720 K or double-sided high-density 1.44 M)
- MS-DOS® or PC-DOS™ 5-1/4-inch diskette (double-sided double-density 360 K or double-sided high-density 1.2 M)

Use positive logic for data and addresses.

When submitting the application program on a diskette, clearly label the diskette with the following information:

- Customer name
- Customer part number
- Project or product name
- File name of object code
- Date
- Name of operating system that formatted diskette
- Formatted capacity of diskette

On diskettes, the application program must be in Motorola's S-record format (S1 and S9 records), a character-based object file format generated by M6805 cross assemblers and linkers.

[1] Macintosh is a registered trademark of Apple Computer, Inc.

[2] MS-DOS is a registered trademark of Microsoft Corporation.

[3] PC-DOS is a trademark of International Business Machines Corporation.

Begin the application program at the first user ROM location. Program addresses must correspond exactly to the available on-chip user ROM addresses as shown in the memory map. **Write $00 in all nonuser ROM locations or leave all nonuser ROM locations blank**. Refer to the current MCU ordering form for additional requirements. Motorola may request pattern resubmission if nonuser areas contain any nonzero code.

If the memory map has two user ROM areas with the same addresses, then write the two areas in separate files on the diskette. Label the diskette with both filenames.

In addition to the object code, a file containing the source code can be included. Motorola keeps this code confidential and uses it only to expedite ROM pattern generation in case of any difficulty with the object code. Label the diskette with the filename of the source code.

ROM Program Verification

The primary use for the on-chip ROM is to hold the customer's application program. The customer develops and debugs the application program and then submits the MCU order along with the application program.

Motorola enters the customer's application program code into a computer program that generates a listing verify file. The listing verify file represents the memory map of the MCU. The listing verify file contains the user ROM code and may also contain nonuser ROM code, such as selfcheck code. Motorola sends the customer a computer printout of the listing verify file along with a listing verify form.

To aid the customer in checking the listing verify file, Motorola programs the listing verify file into customer-supplied blank preformatted Macintosh or DOS disks. All original pattern media are filed for contractual purposes and are not returned.

Check the listing verify file thoroughly, then complete and sign the listing verify form and return the listing verify form to Motorola. The signed listing verify form constitutes the contractual agreement for the creation of the custom mask.

ROM Verification Units (RVUs)

After receiving the signed listing verify form, Motorola manufactures a custom photographic mask. The mask contains the customer's application program and is used to process silicon wafers. The application program cannot be changed after the manufacture of the mask begins. Motorola then produces 10 MCUs, called RVUs, and sends the RVUs to the customer. RVUs are usually packaged in unmarked ceramic and tested to 5 Vdc at room temperature. RVUs are not tested to environmental extremes because their sole purpose is to demonstrate that the customer's user ROM pattern was properly implemented. The 10 RVUs are free of charge with the minimum order quantity. These units are not to be used for qualification or production. RVUs are not guaranteed by Motorola Quality Assurance.

MC Order Numbers

Table 2. MC Order Numbers

MC Order Number	Operating Temperature Range
MC68HC08XL36CB[1][2]	– 40 °C to + 85 °C
MC68HC08XL36CFU[3]	

1. C = extended temperature range
2. B = plastic shrink dip package
3. FU = plastic quad flat pack package

Memory Map

Contents

Introduction

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

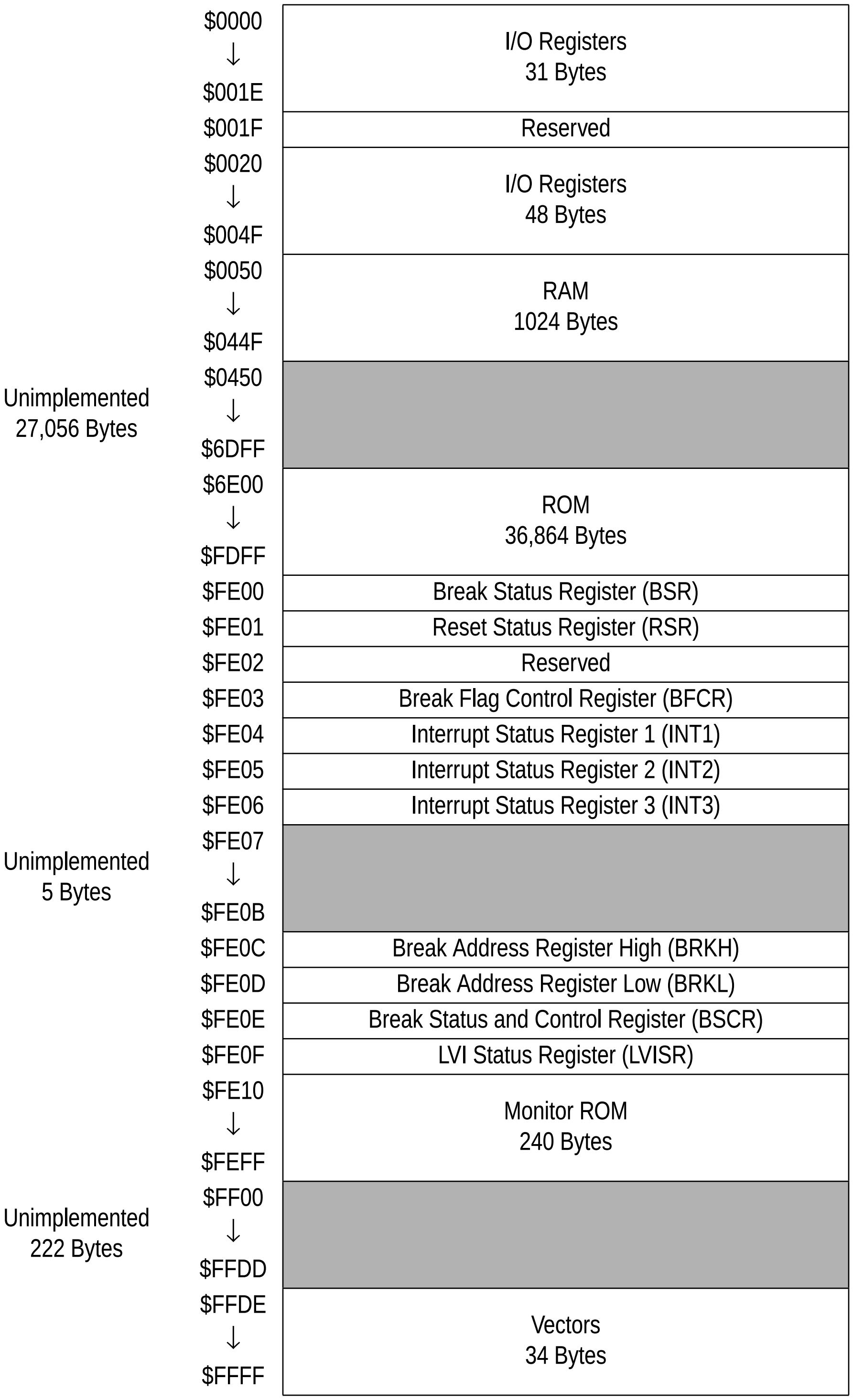
- 36 Kbytes of erasable programmable read-only memory (ROM)
- One Kbyte of random-access memory (RAM)
- 34 bytes of user-defined vectors
- 240 bytes of monitor ROM

Unimplemented Memory Locations

Accessing an unimplemented location can cause an illegal address reset if illegal address resets are enabled. In the memory map figure and in register figures in this document, unimplemented locations are shaded.

Reserved Memory Locations

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



	Address	Contents
	$0000 ↓ $001E	I/O Registers 31 Bytes
	$001F	Reserved
	$0020 ↓ $004F	I/O Registers 48 Bytes
	$0050 ↓ $044F	RAM 1024 Bytes
Unimplemented 27,056 Bytes	$0450 ↓ $6DFF	
	$6E00 ↓ $FDFF	ROM 36,864 Bytes
	$FE00	Break Status Register (BSR)
	$FE01	Reset Status Register (RSR)
	$FE02	Reserved
	$FE03	Break Flag Control Register (BFCR)
	$FE04	Interrupt Status Register 1 (INT1)
	$FE05	Interrupt Status Register 2 (INT2)
	$FE06	Interrupt Status Register 3 (INT3)
Unimplemented 5 Bytes	$FE07 ↓ $FE0B	
	$FE0C	Break Address Register High (BRKH)
	$FE0D	Break Address Register Low (BRKL)
	$FE0E	Break Status and Control Register (BSCR)
	$FE0F	LVI Status Register (LVISR)
	$FE10 ↓ $FEFF	Monitor ROM 240 Bytes
Unimplemented 222 Bytes	$FF00 ↓ $FFDD	
	$FFDE ↓ $FFFF	Vectors 34 Bytes

Figure 1. Memory Map

Input/Output (I/O) Section

Addresses $0000–$004F contain most of the control, status, and data registers. Additional I/O registers have the following addresses:

- $FE00 (break status register, BSR)
- $FE01 (reset status register, RSR)
- $FE03 (break flag control register, BFCR)
- $FE04 (interrupt status register 1, INT1)
- $FE05 (interrupt status register 2, INT2)
- $FE06 (interrupt status register 3, INT3)
- $FE0C and $FE0D (break address registers, BRKH and BRKL)
- $FE0E (break status and control register, BSCR)
- $FE0F (LVI status register, LVISR)
- $FFFF (COP control register, COPCTL)

Register Name	Addr.		Bit 7	6	5	4	3	2	1	Bit 0
Port A Data Register (PORTA)	$0000	Read:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
		Write:								
		Reset:	Unaffected by Reset							
Port B Data Register (PORTB)	$0001	Read:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
		Write:								
		Reset:	Unaffected by Reset							
Port C Data Register (PORTC)	$0002	Read:	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
		Write:								
		Reset:	Unaffected by Reset							
Port D Data Register (PORTD)	$0003	Read:	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
		Write:								
		Reset:	Unaffected by Reset							
Data Direction Register A (DDRA)	$0004	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Data Direction Register B (DDRB)	$0005	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Data Direction Register C (DDRC)	$0006	Read:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Data Direction Register D (DDRD)	$0007	Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Port E Data Register (PORTE)	$0008	Read:	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
		Write:								
		Reset:	Unaffected by Reset							
Port F Data Register (PORTF)	$0009	Read:	0	0	PF5	PF4	PF3	PF2	PF1	PF0
		Write:	▨	▨						
		Reset:	Unaffected by Reset							
Port G Data Register (PORTG)	$000A	Read:	0	0	0	0	PG3	PG2	PG1	PG0
		Write:	▨	▨	▨	▨				
		Reset:	Unaffected by Reset							
Port H Data Register (PORTH)	$000B	Read:	0	0	0	0	PH3	PH2	PH1	PH0
		Write:	▨	▨	▨	▨				
		Reset:	Unaffected by Reset							
Data Direction Register E (DDRE)	$000C	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Data Direction Register F (DDRF)	$000D	Read:	0	0	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
		Write:	▨	▨						
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary

Register Name	Addr.		Bit 7	6	5	4	3	2	1	Bit 0
Data Direction Register G (DDRG)	$000E	Read:	0	0	0	0	DDRG3	DDRG2	DDRG1	DDRG0
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
Data Direction Register H (DDRH)	$000F	Read:	0	0	0	0	DDRH3	DDRH2	DDRH1	DDRH0
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
SPI Control Register (SPCR)	$0010	Read:	SPRIE	DMAS	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:								
		Reset:	0	0	1	0	0	0	0	0
SPI Status and Control Register (SPSCR)	$0011	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:	▨		▨	▨	▨			
		Reset:	0	0	0	0	1	0	0	0
SPI Data Register (SPDR)	$0012	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by Reset							
SCI Control Register 1 (SCC1)	$0013	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
SCI Control Register 2 (SCC2)	$0014	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
SCI Control Register 3 (SCC3)	$0015	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		Write:	▨							
		Reset:	U	U	0	0	0	0	0	0
SCI Status Register 1 (SCS1)	$0016	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	1	1	0	0	0	0	0	0
SCI Status Register 2 (SCS2)	$0017	Read:	▨	▨	▨	▨	▨	▨	BKF	RPF
		Write:							▨	▨
		Reset:	0	0	0	0	0	0	0	0
SCI Data Register (SCDR)	$0018	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by Reset							
SCI Baud Rate Register (SCBR)	$0019	Read:	▨	▨	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Keyboard Status and Control Register (KBSCR)	$001A	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
		Write:	▨	▨	▨	▨	▨	ACKK		
		Reset:	0	0	0	0	0	0	0	0
Keyboard Interrupt Enable Register (KBIER)	$001B	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary (Continued)

Register Name	Addr.		Bit 7	6	5	4	3	2	1	Bit 0
PLL Control Register (PCTL)	$001C	Read:	PLLIE	PLLF	PLLON	BCS	1	1	1	1
		Write:		▨			▨	▨	▨	▨
		Reset:	0	0	1	0	1	1	1	1
PLL Bandwidth Control Register (PBWC)	$001D	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	XLD	0	0	0	0
		Write:		▨			▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
PLL Programming Register (PPG)	$001E	Read:	MUL7	MUL6	MUL5	MUL4	VRS7	VRS6	VRS5	VRS4
		Write:								
		Reset:	0	1	1	0	0	1	1	0
	$001F		Reserved							
TIM Status and Control Register (TSC)	$0020	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	▨			
		Reset:	0	0	1	0	0	0	0	0
TIM DMA Select Register (TDMA)	$0021	Read:	0	0	0	0	DMA3S	DMA2S	DMA1S	DMA0S
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
TIM Counter Register High (TCNTH)	$0022	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
TIM Counter Register Low (TCNTL)	$0023	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
TIM Counter Modulo Register High (TMODH)	$0024	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
TIM Counter Modulo Register Low (TMODL)	$0025	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
TIM Channel 0 Status and Control Register (TSC0)	$0026	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
TIM Channel 0 Register High (TCH0H)	$0027	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after Reset							
TIM Channel 0 Register Low (TCH0L)	$0028	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after Reset							
TIM Channel 1 Status and Control Register (TSC1)	$0029	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		▨					
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary (Continued)

Register Name	Addr.		Bit 7	6	5	4	3	2	1	Bit 0
TIM Channel 1 Register High (TCH1H)	$002A	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after Reset							
TIM Channel 1 Register Low (TCH1L)	$002B	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after Reset							
TIM Channel 2 Status and Control Register (TSC2)	$002C	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
TIM Channel 2 Register High (TCH2H)	$002D	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after Reset							
TIM Channel 2 Register Low (TCH2L)	$002E	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after Reset							
TIM Channel 3 Status and Control Register (TSC3)	$002F	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
		Write:	0		▨					
		Reset:	0	0	0	0	0	0	0	0
TIM Channel 3 Register High (TCH3H)	$0030	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after Reset							
TIM Channel 3 Register Low (TCH3L)	$0031	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after Reset							
IRQ Status and Control Register (ISCR)	$0032	Read:	IRQF2	0	IMASK2	MODE2	IRQF1	0	IMASK1	MODE1
		Write:	▨	ACK2			▨	ACK1		
		Reset:	0	0	0	0	0	0	0	0
	$0033		Reserved							
DMA Channel 0 Source Address Register High) (D0SH)	$0034	Read:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
		Write:								
		Reset:	Indeterminate after Reset							
DMA Channel 0 Source Address Register Low (D0SL)	$0035	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:	Indeterminate after Reset							
DMA Channel 0 Destination Address Register High (D0DH)	$0036	Read:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
		Write:								
		Reset:	Indeterminate after Reset							
DMA Channel 0 Destination Address Register Low (D0DL)	$0037	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:	Indeterminate after Reset							

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary (Continued)

Register Name	Addr.		Bit 7	6	5	4	3	2	1	Bit 0
DMA Channel 0 Control Register (D0C)	$0038	Read: Write:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
		Reset:	Indeterminate after Reset							
DMA Channel 0 Block Length Register (D0BL)	$0039	Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
		Reset:	Indeterminate after Reset							
	$003A		Reserved							
DMA Channel 0 Byte Count Register (D0BC)	$003B	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
		Reset:	0	0	0	0	0	0	0	0
DMA Channel 1 Source Address Register High) (D1SH)	$003C	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
		Reset:	Indeterminate after Reset							
DMA Channel 1 Source Address Register Low (D1SL)	$003D	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Reset:	Indeterminate after Reset							
DMA Channel 1 Destination Address Register High (D1DH)	$003E	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
		Reset:	Indeterminate after Reset							
DMA Channel 1 Destination Address Register Low (D1DL)	$003F	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Reset:	Indeterminate after Reset							
DMA Channel 1 Control Register (D1C)	$0040	Read: Write:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
		Reset:	Indeterminate after Reset							
DMA Channel 1 Block Length Register (D1BL)	$0041	Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
		Reset:	Indeterminate after Reset							
	$0042		Reserved							
DMA Channel 1 Byte Count Register (D1BC)	$0043	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
		Reset:	0	0	0	0	0	0	0	0
DMA Channel 2 Source Address Register High) (D2SH)	$0044	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
		Reset:	Indeterminate after Reset							
DMA Channel 2 Source Address Register Low (D2SL)	$0045	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Reset:	Indeterminate after Reset							

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary (Continued)

Register Name	Addr.		Bit 7	6	5	4	3	2	1	Bit 0
DMA Channel 2 Destination Address Register High (D2DH)	$0046	Read:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
		Write:								
		Reset:	Indeterminate after Reset							
DMA Channel 2 Destination Address Register Low (D2DL)	$0047	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:								
		Reset:	Indeterminate after Reset							
DMA Channel 2 Control Register (D2C)	$0048	Read:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
		Write:								
		Reset:	Indeterminate after Reset							
DMA Channel 2 Block Length Register (D2BL)	$0049	Read:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
		Write:								
		Reset:	Indeterminate after Reset							
	$004A		Reserved							
DMA Channel 2 Byte Count Register (D2BC)	$004B	Read:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
DMA Control Register 1 (DC1)	$004C	Read:	BB1	BB0	TEC2	IEC2	TEC1	IEC1	TEC0	IEC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
DMA Status and Control Register (DSC)	$004D	Read:	DMAP	L2	L1	L0	DMAWE	IFC2	IFC1	IFC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
DMA Control Register 2 (DC2)	$004E	Read:	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
	$004F		Reserved							
Break Status Register (BSR)	$FE00	Read:	R	R	R	R	R	R	BW	R
		Write:							Clear BW	
		Reset:							0	
Reset Status Register (RSR)	$FE01	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	1	0	0	0	0	0	0	0
Break Flag Control Register (BFCR)	$FE03	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
Interrupt Status Register 1 (INT1)	$FE04	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary (Continued)

Register Name	Addr.		Bit 7	6	5	4	3	2	1	Bit 0
Interrupt Status Register 2 (INT2)	$FE05	Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
Interrupt Status Register 3 (INT3)	$FE06	Read:	0	0	0	0	0	0	0	IF15
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
	$FE07		▨	▨	▨	▨	▨	▨	▨	▨
Break Address Register High (BRKH)	$FE0C	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Break Address Register Low (BRKL)	$FE0D	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
Break Status and Control Register (BSCR)	$FE0E	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:			▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
LVI Status Register (LVISR)	$FE0F	Read:	LVIOUT	0	0	0	0	0	0	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
COP Control Register (COPCTL)	$FFFF	Read:	Low byte of reset vector							
		Write:	Writing clears COP counter							
		Reset:	Unaffected by Reset							

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary (Continued)

Table 1 is a list of vector locations.

Table 1. Vector Addresses

	Address	Vector
Lower Priority	$FFDE	Keyboard Vector (High)
	$FFDF	Keyboard Vector (Low)
	$FFE0	IRQ2 Vector (High)
	$FFE1	IRQ2 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 Overflow Vector (High)
	$FFED	TIM Overflow Vector (Low)
	$FFEE	TIM Channel 3 Vector (High)
	$FFEF	TIM Channel 3 Vector (Low)
	$FFF0	TIM Channel 2 Vector (High)
	$FFF1	TIM Channel 2 Vector (Low)
	$FFF2	TIM Channel 1 Vector (High)
	$FFF3	TIM Channel 1 Vector (Low)
	$FFF4	TIM Channel 0 Vector (High)
	$FFF5	TIM Channel 0 Vector (Low)
	$FFF6	DMA Vector (High)
	$FFF7	DMA Vector (Low)
	$FFF8	PLL Vector (High)
	$FFF9	PLL Vector (Low)
	$FFFA	IRQ1 Vector (High)
Higher Priority	$FFFB	IRQ1 Vector (Low)
	$FFFC	SWI Vector (High)
	$FFFD	SWI Vector (Low)
	$FFFE	Reset Vector (High)
	$FFFF	Reset Vector (Low)

Random Access Memory (RAM)

Contents

Introduction

This section describes the 1024 bytes of RAM.

Functional Description

Addresses $0050 through $044F are RAM locations. The location of the stack RAM is programmable with the reset stack pointer instruction (RSP). The 16-bit stack pointer allows the stack to be anywhere in the 64-Kbyte memory space.

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

Within page 0 are 176 bytes of RAM. Because the location of the stack RAM is programmable, all page 0 RAM locations can be used for I/O control and user data or code. When the stack pointer is moved from its reset location at $00FF, direct addressing mode instructions can access efficiently all page 0 RAM locations. Page 0 RAM, therefore, provides ideal locations for frequently accessed global variables.

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

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

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

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

Read-Only Memory (ROM)

Contents

Introduction

This MCU has 36 Kbytes of ROM.

Functional Description

An unprogrammed ROM location reads $00. The following addresses are user ROM locations:

- $6E00–$FDFF
- $FFE0–$FFFF — These locations are reserved for user-defined interrupt and reset vectors.

NOTE: *A security feature discourages viewing of the ROM.*[1]

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

Central Processor Unit (CPU)

Contents

Introduction

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

Features

Features of the CPU include the following:

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

CPU Registers

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

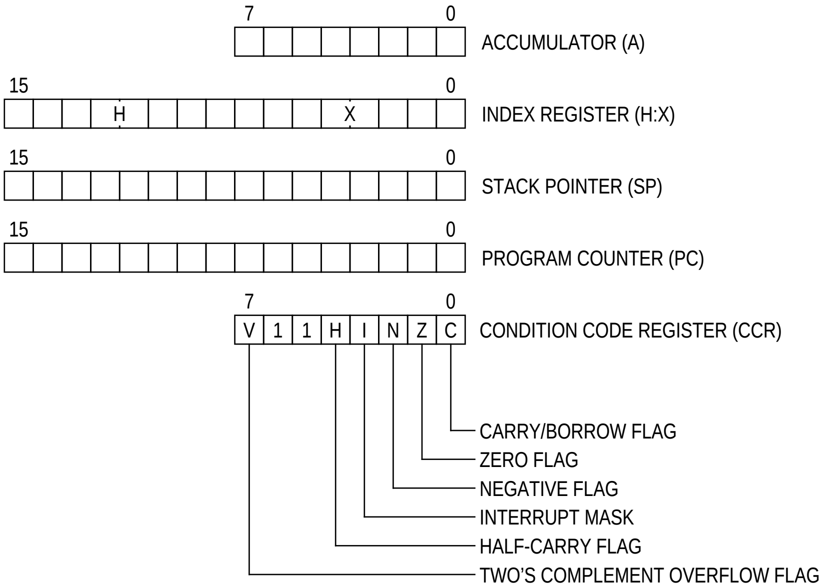


Figure 1. CPU Registers

Accumulator

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

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

Figure 2. Accumulator (A)

Index Register

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

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

	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 3. Index Register (H:X)

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

Stack Pointer

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

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

	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 4. Stack Pointer (SP)

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

Program Counter

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

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

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

	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 5. Program Counter (PC)

Condition Code Register

The 8-bit condition code register contains the interrupt mask and five flags that indicate the results of the instruction just executed. Bits 6 and 5 are set permanently to logic 1. The following paragraphs describe the functions of the condition code register.

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

X = Indeterminate

Figure 6. Condition Code Register (CCR)

V — Overflow Flag

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

1 = Overflow
0 = No overflow

H — Half-Carry Flag

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

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

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

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

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

N — Negative flag

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

1 = Negative result
0 = Non-negative result

Z — Zero flag

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

1 = Zero result
0 = Nonzero result

C — Carry/Borrow Flag

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

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

Arithmetic/Logic Unit (ALU)

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

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

Low-Power Modes

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

Wait Mode

The WAIT instruction:

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

Stop Mode

The STOP instruction:

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

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

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 Break Module (BRK) on page 141.) 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

Instruction Set Summary

Table 1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr*	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM	A9	ii	2
ADC *opr*									DIR	B9	dd	3
ADC *opr*									EXT	C9	hh ll	4
ADC *opr*,X									IX2	D9	ee ff	4
ADC *opr*,X									IX1	E9	ff	3
ADC ,X									IX	F9		2
ADC *opr*,SP									SP1	9EE9	ff	4
ADC *opr*,SP									SP2	9ED9	ee ff	5
ADD #*opr*	Add without Carry	A ← (A) + (M)	↕	↕	–	↕	↕	↕	IMM	AB	ii	2
ADD *opr*									DIR	BB	dd	3
ADD *opr*									EXT	CB	hh ll	4
ADD *opr*,X									IX2	DB	ee ff	4
ADD *opr*,X									IX1	EB	ff	3
ADD ,X									IX	FB		2
ADD *opr*,SP									SP1	9EEB	ff	4
ADD *opr*,SP									SP2	9EDB	ee ff	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*	Logical AND	A ← (A) & (M)	0	–	–	↕	↕	–	IMM	A4	ii	2
AND *opr*									DIR	B4	dd	3
AND *opr*									EXT	C4	hh ll	4
AND *opr*,X									IX2	D4	ee ff	4
AND *opr*,X									IX1	E4	ff	3
AND ,X									IX	F4		2
AND *opr*,SP									SP1	9EE4	ff	4
AND *opr*,SP									SP2	9ED4	ee ff	5

Table 1. Instruction Set Summary (Continued)

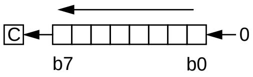
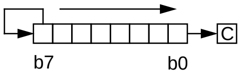
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X ASL *opr*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
ASR *opr* ASRA ASRX ASR *opr*,X 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
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

Table 1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n*,*opr*,*rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2	–	–	–	–	–	–	REL	21	rr	3
BRSET *n*,*opr*,*rel*	Branch if Bit *n* in M Set	PC ← (PC) + 3 + *rel* ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr,rel* CBEQA #*opr,rel* CBEQX #*opr,rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2
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}$) = $FF – (M) A ← ($\underline{A}$) = $FF – (M) X ← ($\underline{X}$) = $FF – (M) M ← ($\underline{M}$) = $FF – (M) M ← ($\underline{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

Table 1. Instruction Set Summary (Continued)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
CPHX #*opr* CPHX *opr*	Compare H:X with M	(H:X) – (M:M + 1)	↕	–	–	↕	↕	↕	IMM DIR	65 75	ii ii+1 dd	3 4
CPX #*opr* CPX *opr* CPX *opr* CPX ,X CPX *opr*,X CPX *opr*,X CPX *opr*,SP CPX *opr*,SP	Compare X with M	(X) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9EE3 9ED3	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	$(A)_{10}$	U	–	–	↕	↕	↕	INH	72		2
DBNZ *opr,rel* DBNZA *rel* DBNZX *rel* DBNZ *opr*,X,*rel* DBNZ X,*rel* DBNZ *opr*,SP,*rel*	Decrement and Branch if Not Zero	A ← (A) – 1 or M ← (M) – 1 or X ← (X) – 1 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 4 + *rel* ? (result) ≠ 0	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6
DEC *opr* DECA DECX DEC *opr*,X DEC ,X DEC *opr*,SP	Decrement	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1 M ← (M) – 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	4 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		7
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
INC *opr* INCA INCX INC *opr*,X INC ,X INC *opr*,SP	Increment	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1 M ← (M) + 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	4 5 6 5 4
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X LDA *opr*,SP LDA *opr*,SP	Load A from M	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9EE6 9ED6	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LDHX #*opr* LDHX *opr*	Load H:X from M	H:X ← (M:M + 1)	0	–	–	↕	↕	–	IMM DIR	45 55	ii jj dd	3 4

12-cpu8_a

Table 1. Instruction Set Summary (Continued)

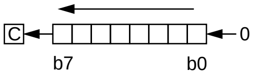
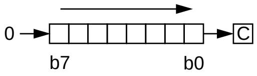
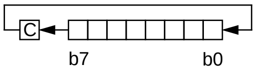
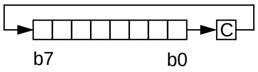
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LDX *#opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X LDX *opr*,SP LDX *opr*,SP	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee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

Table 1. Instruction Set Summary (Continued)

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

14-cpu8_a

Table 1. Instruction Set Summary (Continued)

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*	Test for Negative or Zero	(A) – $00 or (X) – $00 or (M) – $00	0	–	–	↕	↕	–	DIR	3D	dd	3
TSTA									INH	4D		1
TSTX									INH	5D		1
TST *opr*,X									IX1	6D	ff	3
TST ,X									IX	7D		2
TST *opr*,SP									SP1	9E6D	ff	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
C	Carry/borrow bit
CCR	Condition code register
dd	Direct address of operand
dd rr	Direct address of operand and relative offset of branch instruction
DD	Direct to direct addressing mode
DIR	Direct addressing mode
DIX+	Direct to indexed with post increment addressing mode
ee ff	High and low bytes of offset in indexed, 16-bit offset addressing
EXT	Extended addressing mode
ff	Offset byte in indexed, 8-bit offset addressing
H	Half-carry bit
H	Index register high byte
hh ll	High and low bytes of operand address in extended addressing
I	Interrupt mask
ii	Immediate operand byte
IMD	Immediate source to direct destination addressing mode
IMM	Immediate addressing mode
INH	Inherent addressing mode
IX	Indexed, no offset addressing mode
IX+	Indexed, no offset, post increment addressing mode
IX+D	Indexed with post increment to direct addressing mode
IX1	Indexed, 8-bit offset addressing mode
IX1+	Indexed, 8-bit offset, post increment addressing mode
IX2	Indexed, 16-bit offset addressing mode
M	Memory location
N	Negative bit
n	Any bit
opr	Operand (one or two bytes)
PC	Program counter
PCH	Program counter high byte
PCL	Program counter low byte
REL	Relative addressing mode
rel	Relative program counter offset byte
rr	Relative program counter offset byte
SP1	Stack pointer, 8-bit offset addressing mode
SP2	Stack pointer 16-bit offset addressing mode
SP	Stack pointer
U	Undefined
V	Overflow bit
X	Index register low byte
Z	Zero bit
&	Logical AND
|	Logical OR
⊕	Logical EXCLUSIVE OR
( )	Contents of
–( )	Negation (two's complement)
#	Immediate value
«	Sign extend
←	Loaded with
?	If
:	Concatenated with
↕	Set or cleared
—	Not affected

Opcode Map

See Table 2 on page 54.

Table 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	IX1+	Indexed, 1-Byte Offset with Post Increment
DD	Direct-Direct	IMD	Immediate-Direct		
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

16-cpu8_a

Resets and Interrupts

Contents

Introduction

Resets and interrupts are responses to exceptional events during program execution. A reset reinitializes the MCU to its startup condition. An interrupt vectors the program counter to a service routine.

Resets

A reset returns the MCU to a known startup condition and begins program execution from a user-defined memory location.

Effects

A reset:

- Immediately stops the operation of the instruction being executed.
- Initializes certain control and status bits.
- Loads the program counter with a user-defined reset vector address from locations $FFFE and $FFFF.
- Selects CGMXCLK divided by four as the bus clock.

External Reset

A logic 0 applied to the $\overline{\text{RST}}$ pin for a time, t_{IRL}, generates an external reset. An external reset sets the PIN bit in the reset status register.

Internal Reset

Sources:

- Power-on reset
- COP
- Low-voltage inhibit
- Illegal opcode
- Illegal address

All internal reset sources pull the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles to allow resetting of external devices. The MCU is held in reset for an additional 32 CGMXCLK cycles after releasing the $\overline{\text{RST}}$ pin.

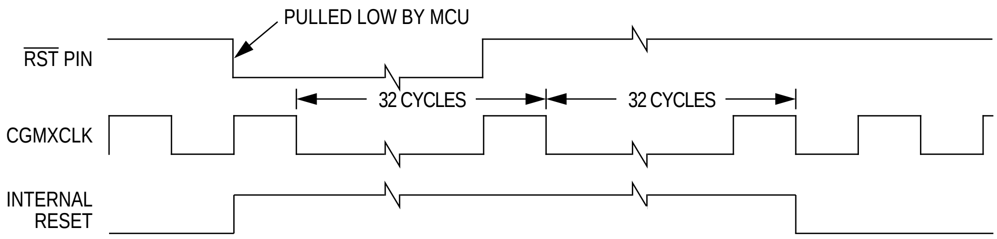
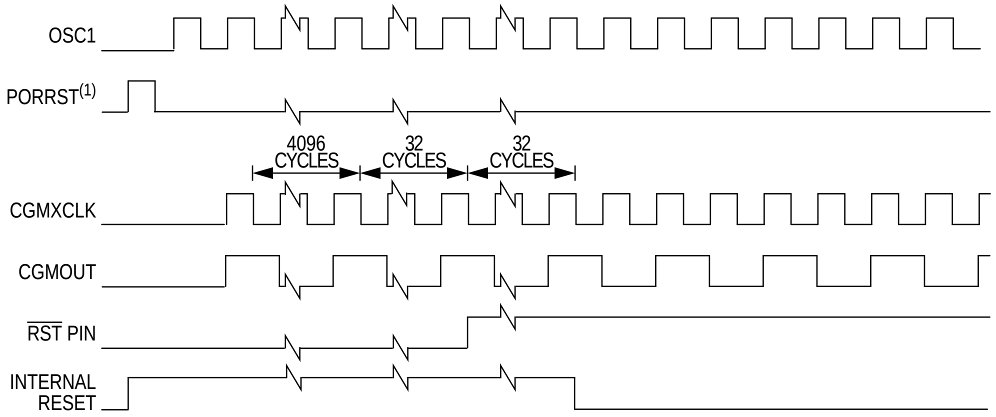


Figure 1. Internal Reset Timing

Power-On Reset

A power-on reset (POR) is an internal reset caused by a positive transition on the V_{DD} pin. A power-on reset:

- Holds the clocks to the CPU and modules inactive for an oscillator stabilization delay of 4096 CGMXCLK cycles.
- Drives the $\overline{\text{RST}}$ pin low during the oscillator stabilization delay.
- Releases the RST pin 32 CGMXCLK cycles after the oscillator stabilization delay.
- Releases the CPU to begin the reset vector sequence 64 CGMXCLK cycles after the oscillator stabilization delay.
- Sets the POR bit in the reset status register and clears all other bits in the register.



1. PORRST is an internally generated power-on reset pulse.

Figure 2. Power-On Reset Recovery

COP Reset

A COP reset is an internal reset caused by an overflow of the COP counter. A COP reset sets the COP bit in the reset status register.

To clear the COP counter and prevent a COP reset, write any value to the COP control register at location $FFFF.

Low-Voltage Inhibit Reset

A low-voltage inhibit (LVI) reset is an internal reset caused by a drop in the power supply voltage to the $\mathrm{LVI_{tripf}}$ voltage. An LVI reset:

- Holds the clocks to the CPU and modules inactive for an oscillator stabilization delay of 4096 CGMXCLK cycles after the power supply voltage rises to the $\mathrm{LVI_{tripr}}$ voltage.
- Drives the $\overline{\mathrm{RST}}$ pin low for as long as V_{DD} is below the $\mathrm{LVI_{tripr}}$ voltage and during the oscillator stabilization delay.
- Releases the $\overline{\mathrm{RST}}$ pin 32 CGMXCLK cycles after the oscillator stabilization delay.
- Releases the CPU to begin the reset vector sequence 64 CGMXCLK cycles after the oscillator stabilization delay.
- Sets the LVI bit in the reset status register.

Illegal Opcode Reset

An illegal opcode reset is an internal reset caused by an opcode that is not in the instruction set. An illegal opcode reset sets the ILOP bit in the reset status register.

A mask option enables the STOP instruction. If not enabled by mask option, the STOP instruction causes an illegal opcode reset.

Illegal Address Reset

An illegal address reset is an internal reset caused by an opcode fetch from an unmapped address. An illegal address reset sets the ILAD bit in the reset status register.

A data fetch from an unmapped address does not generate a reset.

Reset Status Register

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

NOTE: *A reset source that becomes active before recovery from a previous reset can prevent the previous reset from setting its reset status bit.*

Address: $FE01

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

▨ = Unimplemented

Figure 3. Reset Status Register (RSR)

POR — Power-On Reset Bit

1 = Last reset caused by power-on
0 = Read of RSR

PIN — External Reset Bit

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

COP — Computer Operating Properly Reset Bit

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

ILOP — Illegal Opcode Reset Bit

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

ILAD — Illegal Address Reset Bit

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

LVI — Low-Voltage Inhibit Reset Bit

1 = Last reset caused by low power supply voltage
0 = POR or read of RSR

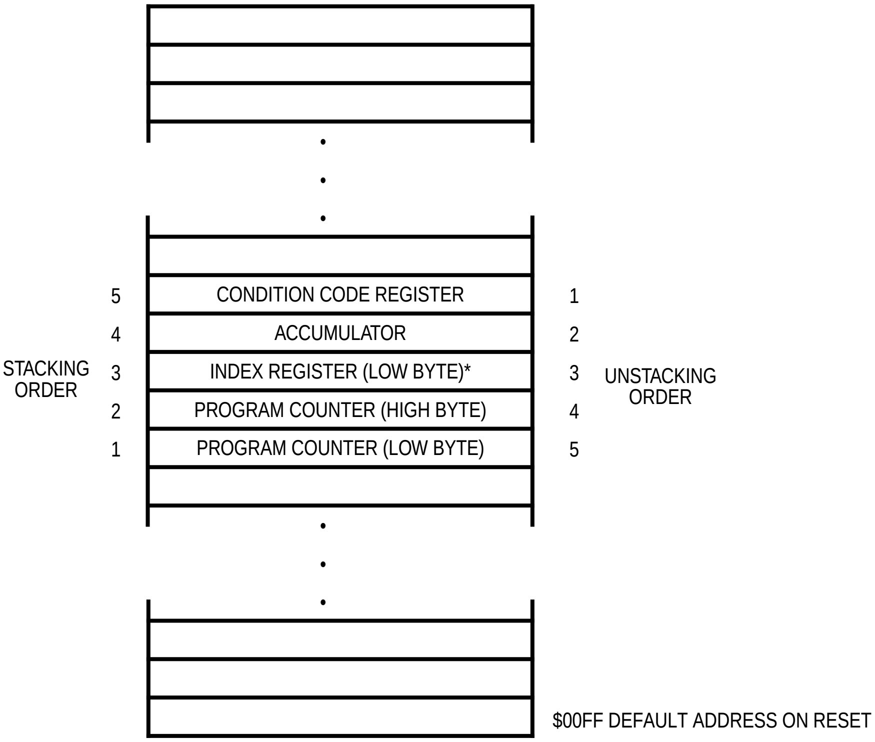
Interrupts

An interrupt temporarily changes the sequence of program execution to respond to a particular event. An interrupt does not stop the operation of the instruction being executed, but begins when the current instruction completes its operation.

Effects

An interrupt:

- Saves the CPU registers on the stack. At the end of the interrupt, the RTI instruction recovers the CPU registers from the stack so that normal processing can resume.


5
CONDITION CODE REGISTER
1
4
ACCUMULATOR
2
STACKING ORDER
3
INDEX REGISTER (LOW BYTE)*
3
UNSTACKING ORDER
2
PROGRAM COUNTER (HIGH BYTE)
4
1
PROGRAM COUNTER (LOW BYTE)
5
$00FF DEFAULT ADDRESS ON RESET

*High byte of index register is not stacked.

Figure 4. Interrupt Stacking Order

- Sets the interrupt mask (I bit) to prevent additional interrupts. Once an interrupt is latched, no other interrupt can take precedence, regardless of its priority.
- Loads the program counter with a user-defined vector address.

After every instruction, the CPU checks all pending interrupts if the I bit is not set. If more than one interrupt is pending when an instruction is done, the highest priority interrupt is serviced first. If an interrupt is pending upon exit from the interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

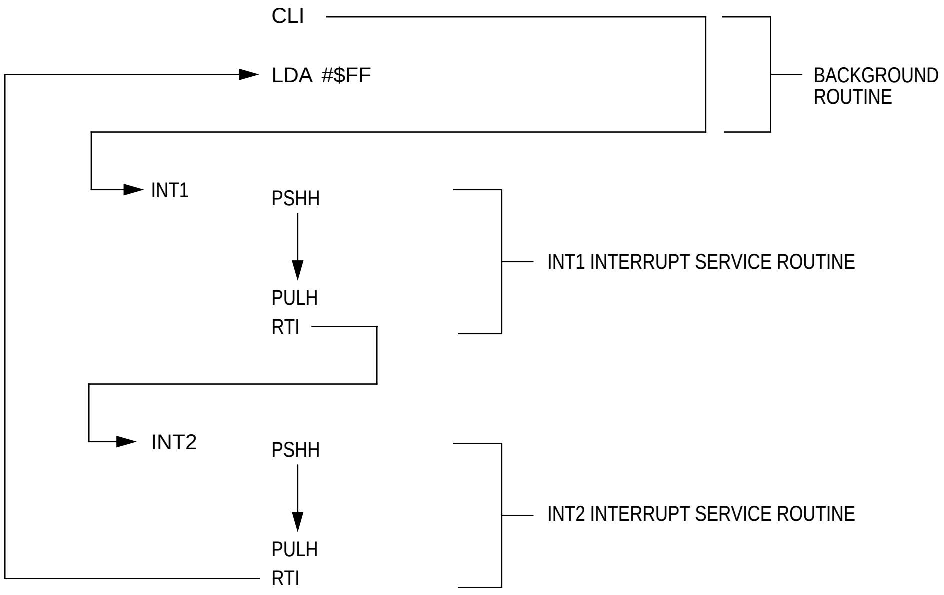

CLI
LDA #$FF
BACKGROUND ROUTINE
INT1
PSHH
PULH
RTI
INT1 INTERRUPT SERVICE ROUTINE
INT2
PSHH
PULH
RTI
INT2 INTERRUPT SERVICE ROUTINE

Figure 5. Interrupt Recognition Example

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

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

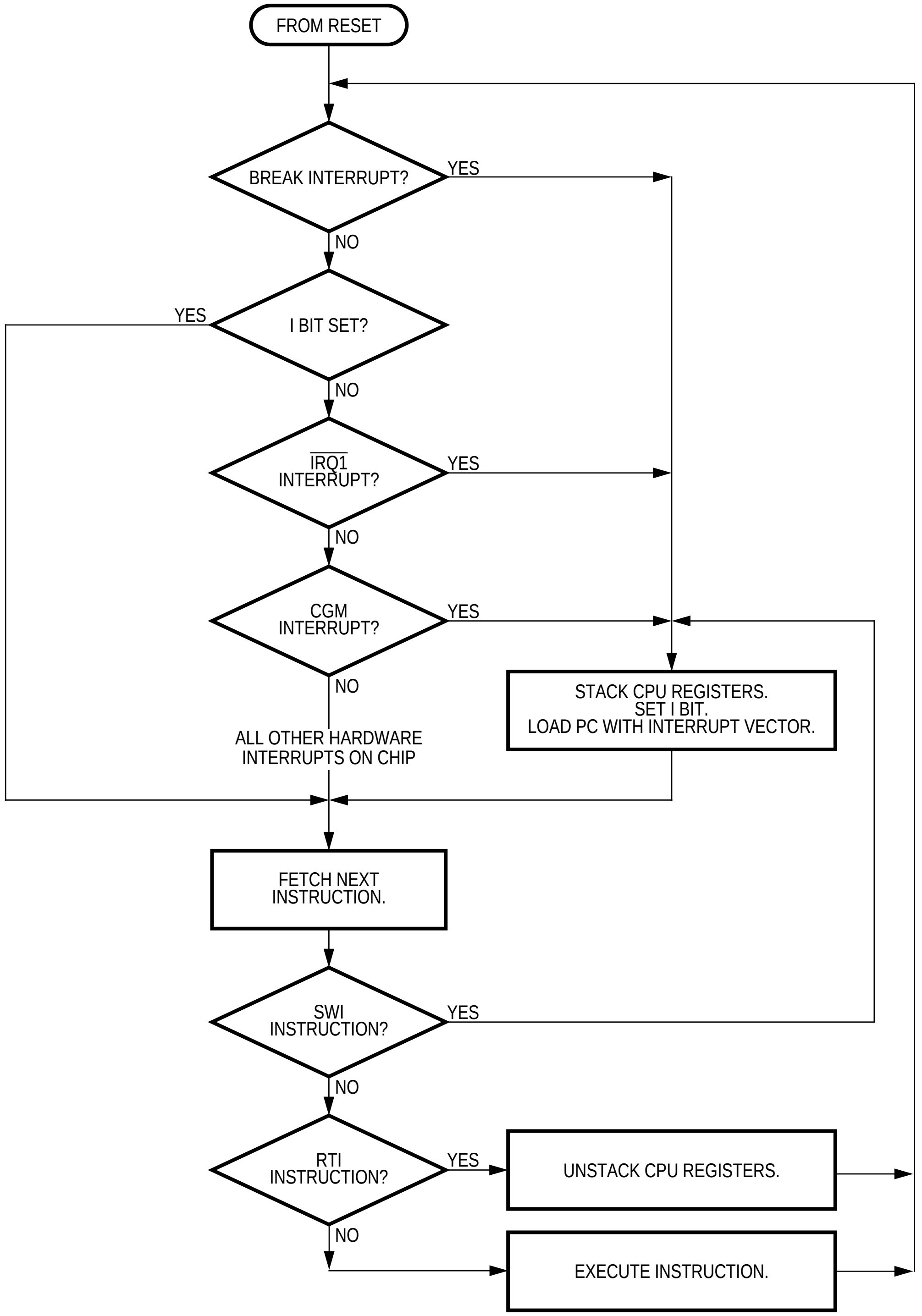


Figure 6. Interrupt Processing

Sources

The following sources can generate CPU interrupt requests:

Table 1. Interrupt Sources

Source	Flag	Mask[1]	INT Register Flag	Priority[2]	Vector Address
SWI Instruction	None	None	None	0	$FFFC–$FFFD
$\overline{\text{IRQ1}}$ Pin	IRQ1F	IMASK1	IF1	1	$FFFA–$FFFB
CGM	PLLF	PLLIE	IF2	2	$FFF8–$FFF9
DMA Channel 0	IFC0	IEC0	IF3	3	$FFF6–$FFF7
DMA Channel 1	IFC1	IEC1			
DMA Channel 2	IFC2	IEC2			
TIM Channel 0	CH0F	CH0IE	IF4	4	$FFF4–$FFF5
TIM Channel 1	CH1F	CH1IE	IF5	5	$FFF2–$FFF3
TIM Channel 2	CH2F	CH2IE	IF6	6	$FFF0–$FFF1
TIM Channel 3	CH3F	CH3IE	IF7	7	$FFEE–$FFEF
TIM Overflow	TOF	TOIE	IF8	8	$FFEC–$FFED
SPI Receiver Full	SPRF	SPRIE	IF9	9	$FFEA–$FFEB
SPI Overflow	OVRF	ERRIE			
SPI Mode Fault	MODF	ERRIE			
SPI Transmitter Empty	SPTE	SPTIE	IF10	10	$FFE8–$FFE9
SCI Receiver Overrun	OR	ORIE	IF11	11	$FFE6–$FFE7
SCI Noise Flag	NF	NEIE			
SCI Framing Error	FE	FEIE			
SCI Parity Error	PE	PEIE			
SCI Receiver Full	SCRF	SCRIE	IF12	12	$FFE4–$FFE5
SCI Input Idle	IDLE	ILIE			
SCI Transmitter Empty	SCTE	SCTIE	IF13	13	$FFE2–$FFE3
SCI Transmission Complete	TC	TCIE			
$\overline{\text{IRQ2}}$ Pin	IRQ2F	IMASK2	IF14	14	$FFE0–$FFE1
Keyboard Pin	KEYF	IMASKK	IF15	15	$FFDE–$FFDF

1. The I bit in the condition code register is a global mask for all interrupt sources except the SWI instruction.
2. 0 = highest priority

SWI Instruction

The software interrupt instruction (SWI) causes a nonmaskable interrupt.

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

Break Interrupt

The break module causes the CPU to execute an SWI instruction at a software-programmable break point.

$\overline{\textit{IRQ1}}$ Pin

A logic 0 on the $\overline{\text{IRQ1}}$ pin latches an external interrupt request.

CGM

The CGM can generate a CPU interrupt request every time the phase-locked loop circuit (PLL) enters or leaves the locked state. When the LOCK bit changes state, the PLL flag (PLLF) is set. The PLL interrupt enable bit (PLLIE) enables PLLF CPU interrupt requests. LOCK is in the PLL bandwidth control register. PLLF is in the PLL control register.

DMA

The DMA module can generate a CPU interrupt request when a channel x CPU interrupt flag (IFCx) becomes set.

- IFCx is set at the end of a DMA block transfer. The channel x CPU interrupt enable bit, IECx, enables DMA channel x CPU interrupt requests.
- IFCx is set at the end of a DMA transfer loop. The channel x CPU interrupt enable bit, IECx, enables DMA channel x CPU interrupt requests.

The IFCx bit is the DMA status and control register. The IECx bit is in DMA control register 1.

TIM

TIM CPU interrupt sources:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter value rolls over to $0000 after matching the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.

- TIM channel flags (CH3F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. The channel x interrupt enable bit, CHxIE, enables channel x TIM CPU interrupt requests. CHxF and CHxIE are in the TIM channel x status and control register.

SPI

SPI CPU interrupt sources:

- SPI receiver full bit (SPRF) — The SPRF bit is set every time a byte transfers from the shift register to the receive data register. The SPI receiver interrupt enable bit, SPRIE, enables SPRF CPU interrupt requests. SPRF is in the SPI status and control register and SPRIE is in the SPI control register.
- SPI transmitter empty (SPTE) — The SPTE bit is set every time a byte transfers from the transmit data register to the shift register. The SPI transmit interrupt enable bit, SPTIE, enables SPTE CPU interrupt requests. SPTE is in the SPI status and control register and SPTIE is in the SPI control register.
- Mode fault bit (MODF) — The MODF bit is set in a slave SPI if the $\overline{\text{SS}}$ pin goes high during a transmission with the mode fault enable bit (MODFEN) set. In a master SPI, the MODF bit is set if the $\overline{\text{SS}}$ pin goes low at any time with the MODFEN bit set. The error interrupt enable bit, ERRIE, enables MODF CPU interrupt requests. MODF, MODFEN, and ERRIE are in the SPI status and control register.
- Overflow bit (OVRF) — The OVRF bit is set if software does not read the byte in the receive data register before the next full byte enters the shift register. The error interrupt enable bit, ERRIE, enables OVRF CPU interrupt requests. OVRF and ERRIE are in the SPI status and control register.

SCI

SCI CPU interrupt sources:

- SCI transmitter empty bit (SCTE) — SCTE is set when the SCI data register transfers a character to the transmit shift register. The SCI transmit interrupt enable bit, SCTIE, enables transmitter CPU interrupt requests. SCTE is in SCI status register 1. SCTIE is in SCI control register 2.
- Transmission complete bit (TC) — TC is set when the transmit shift register and the SCI data register are empty and no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE, enables transmitter CPU interrupt requests. TC is in SCI status register 1. TCIE is in SCI control register 2.
- SCI receiver full bit (SCRF) — SCRF is set when the receive shift register transfers a character to the SCI data register. The SCI receive interrupt enable bit, SCRIE, enables receiver CPU interrupts. SCRF is in SCI status register 1. SCRIE is in SCI control register 2.
- Idle input bit (IDLE) — IDLE is set when 10 or 11 consecutive logic 1s shift in from the RxD pin. The idle line interrupt enable bit, ILIE, enables IDLE CPU interrupt requests. IDLE is in SCI status register 1. ILIE is in SCI control register 2.
- Receiver overrun bit (OR) — OR is set when the receive shift register shifts in a new character before the previous character was read from the SCI data register. The overrun interrupt enable bit, ORIE, enables OR to generate SCI error CPU interrupt requests. OR is in SCI status register 1. ORIE is in SCI control register 3.
- Noise flag (NF) — NF 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, enables NF to generate SCI error CPU interrupt requests. NF is in SCI status register 1. NEIE is in SCI control register 3.

- Framing error bit (FE) — FE is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, enables FE to generate SCI error CPU interrupt requests. FE is in SCI status register 1. FEIE is in SCI control register 3.
- Parity error bit (PE) — PE is set when the SCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, enables PE to generate SCI error CPU interrupt requests. PE is in SCI status register 1. PEIE is in SCI control register 3.

$\overline{\text{IRQ2}}$ Pin A logic 0 on the $\overline{\text{IRQ2}}$ pin latches an external interrupt request.

$\overline{\text{KB0}}$–$\overline{\text{KB7}}$ Pins A logic 0 on a keyboard interrupt pin latches an external interrupt request.

Interrupt Status Registers The flags in the interrupt status registers identify maskable interrupt sources. **Table 2** summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

Table 2. Interrupt Source Flags

Interrupt Source	Interrupt Status Register Flag
Reset	—
SWI Instruction	—
$\overline{\text{IRQ1}}$ Pin	IF1
CGM	IF2
DMA	IF3
TIM Channel 0	IF4
TIM Channel 1	IF5
TIM Channel 2	IF6
TIM Channel 3	IF7
TIM Overflow	IF8
SPI Receiver	IF9
SPI Transmitter	IF10
SCI Error	IF11
SCI Receiver	IF12
SCI Transmitter	IF13
$\overline{\text{IRQ2}}$ Pin	IF14
Keyboard Pin	IF15

Interrupt Status Register 1

Address: $FE04

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7. Interrupt Status Register 1 (INT1)

IF6–IF1 — Interrupt Flags 6–1

These flags indicate the presence of interrupt requests from the sources shown in **Table 2**.

1 = Interrupt request present
0 = No interrupt request present

Bits 0–1 — Always read 0

Interrupt Status Register 2

Address: $FE05

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 8. Interrupt Status Register 2 (INT2)

IF14–IF7 — Interrupt Flags 14–7

These flags indicate the presence of interrupt requests from the sources shown in **Table 2**.

1 = Interrupt request present
0 = No interrupt request present

Interrupt Status Register 3

Address: $FE06

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	0	0	IF15
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 9. Interrupt Status Register 3 (INT3)

Bits 7–1 — Always read 0

IF15 — Interrupt Flag 15

This flag indicates the presence of an interrupt request from the source shown in **Table 2**.

1 = Interrupt request present

0 = No interrupt request present

Low-Power Modes

Contents

Introduction

The WAIT instruction puts the MCU in a low-power standby mode in which the CPU clock is disabled but the bus clock continues to run. The STOP instruction disables both the CPU clock and the bus clock.

Central Processor Unit (CPU)

Wait Mode

The WAIT instruction:

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

Stop Mode

The STOP instruction:

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

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

Clock Generator Module (CGM)

Wait Mode

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

Stop Mode

The STOP instruction disables the CGM and holds 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.

Break Module (BRK)

Wait Mode

If enabled, the break module is active in wait mode. A DMA-generated address that matches the break address registers in wait mode sets the BSW in the break status register.

The DMA can also use the break status and control register as its destination address in order to write to the BRKA and BRKE bits during wait mode. A DMA write to the break status and control register sets the BSW bit.

Stop Mode

The break module is inactive in stop mode. The STOP instruction does not affect break module register states.

Computer Operating Properly Module (COP)

Wait Mode

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

Stop Mode

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

To prevent inadvertently turning off the COP with a STOP instruction, a mask option that disables the STOP instruction is available.

Direct Memory Access Module (DMA)

Wait Mode

If enabled by the DMAWE bit in the DMA status and control register, the DMA remains active in wait mode. The DMA can transfer data to and from peripherals while the MCU remains in wait mode.

If the WAIT instruction occurs during a DMA transfer while DMAWE is set, the DMA transfer continues to completion. If the DMAWE bit is clear, a WAIT instruction suspends the current DMA transfer. If the DMA priority bit (DMAP) is set, the suspended transfer resumes when the MCU exits wait mode.

Stop Mode

The DMA is inactive during stop mode. A STOP instruction suspends any DMA transfer in progress. If an external interrupt brings the MCU out of stop mode and the DMA priority bit (DMAP) is set, the suspended DMA transfer resumes. If a reset brings the MCU out of stop mode, the transfer is aborted.

Entering stop mode when a DMA channel is enabled may fail to clear the the interrupt mask (I bit) in the condition code register. To make sure the I bit is cleared when entering stop mode:

- Before executing the STOP instruction, wait until any current DMA transfer is complete. Then disable DMA transfers by clearing bits TEC[2:0] in DMA control register 1.

  Or,

- Execute the clear-interrupt-mask instruction (CLI) before entering stop mode.

External Interrupt Module (IRQ)

Wait Mode

The IRQ module remains active in wait mode. Clearing the IMASK1 or IMASK2 bit in the IRQ status and control register enables $\overline{\text{IRQ1}}$ or $\overline{\text{IRQ2}}$ CPU interrupt requests to bring the MCU out of wait mode.

Stop Mode

The IRQ module remains active in stop mode. Clearing the IMASK1 or IMASK2 bit in the IRQ status and control register enables $\overline{\text{IRQ1}}$ or $\overline{\text{IRQ2}}$ CPU interrupt requests to bring the MCU out of stop mode.

Keyboard Interrupt Module (KBI)

Wait Mode

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

Stop Mode

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

Low-Voltage Inhibit Module (LVI)

Wait Mode

If enabled, the LVI module remains active in wait mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of wait mode.

Stop Mode

If enabled, the LVI module remains active in stop mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of stop mode.

Serial Communications Interface Module (SCI)

Wait Mode

The SCI module remains active in wait mode. Any enabled CPU interrupt request from the SCI module can bring the MCU out of wait mode.

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

The DMA can service the SCI without exiting wait mode.

Stop Mode

The SCI module is inactive in stop mode. The STOP instruction does not affect SCI register states. SCI module operation resumes after the MCU exits stop mode.

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

Serial Peripheral Interface Module (SPI)

Wait Mode

The SPI module remains active in wait mode. Any enabled CPU interrupt request from the SPI module can bring the MCU out of wait mode.

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

The DMA can service the SPI without exiting wait mode.

Stop Mode

The SPI module is inactive in stop mode. The STOP instruction does not affect SPI register states. SPI operation resumes after an external interrupt. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

Timer Interface Module (TIM)

Wait Mode

The TIM remains active in wait mode. Any enabled CPU interrupt request from the TIM can bring the MCU out of wait mode.

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

The DMA can service the TIM without exiting wait mode.

Stop Mode

The TIM is inactive in stop mode. The STOP instruction does not affect register states or the state of the TIM counter. TIM operation resumes when the MCU exits stop mode after an external interrupt.

Exiting Wait Mode

The following events restart the CPU clock and load the program counter with the reset vector or with an interrupt vector:

- External reset — A logic 0 on the $\overline{\text{RST}}$ pin resets the MCU and loads the program counter with the contents of locations $FFFE and $FFFF.
- External interrupt — A high-to-low transition on an external interrupt pin loads the program counter with the contents of locations:
  - $FFFA and $FFFB ($\overline{\text{IRQ1}}$ pin)
  - $FFE0 and $FFE1 ($\overline{\text{IRQ2}}$ pin)
- Break interrupt — A break interrupt loads the program counter with the contents of $FFFC and $FFFD.
- Computer operating properly module (COP) reset — A timeout of the COP counter resets the MCU and loads the program counter with the contents of $FFFE and $FFFF.
- Low-voltage inhibit module (LVI) reset — A power supply voltage below the $\text{LVI}_{\text{tripf}}$ voltage resets the MCU and loads the program counter with the contents of locations $FFFE and $FFFF.
- Clock generator module (CGM) interrupt — A CPU interrupt request from the phase-locked loop (PLL) loads the program counter with the contents of $FFF8 and $FFF9.
- Direct memory access module (DMA) interrupt — A CPU interrupt request from the DMA loads the program counter with the contents of $FFF6 and $FFF7.

- Timer interface module (TIM) interrupt — A CPU interrupt request from the TIM loads the program counter with the contents of:
  - $FFEC and $FFED (TIM overflow)
  - $FFEE and $FFEF (TIM channel 3)
  - $FFF0 and $FFF1 (TIM channel 2)
  - $FFF2 and $FFF3 (TIM channel 1)
  - $FFF4 and $FFF5 (TIM channel 0)
- Serial peripheral interface module (SPI) interrupt — A CPU interrupt request from the SPI loads the program counter with the contents of:
  - $FFE8 and $FFE9 (SPI transmitter)
  - $FFEA and $FFEB (SPI receiver)
- Serial communications interface module (SCI) interrupt — A CPU interrupt request from the SCI loads the program counter with the contents of:
  - $FFE2 and $FFE3 (SCI transmitter)
  - $FFE4 and $FFE5 (SCI receiver)
  - $FFE6 and $FFE7 (SCI receiver error)

Exiting Stop Mode

The following events restart the system clocks and load the program counter with the reset vector or with an interrupt vector:

- External reset — A logic 0 on the $\overline{\text{RST}}$ pin resets the MCU and loads the program counter with the contents of locations \$FFFE and \$FFFF.
- External interrupt — A high-to-low transition on an external interrupt pin loads the program counter with the contents of locations:
  - \$FFFA and \$FFFB ($\overline{\text{IRQ1}}$ pin)
  - \$FFE0 and \$FFE1 ($\overline{\text{IRQ2}}$ pin)
  - \$FFDE and \$FFDF (keyboard interrupt pins)
- Low-voltage inhibit (LVI) reset — A power supply voltage below the $\text{LVI}_{\text{tripf}}$ voltage resets the MCU and loads the program counter with the contents of locations \$FFFE and \$FFFF.
- Break interrupt — A break interrupt loads the program counter with the contents of locations \$FFFC and \$FFFD.

Upon exit from stop mode, the system clocks begin running after an oscillator stabilization delay. A 12-bit stop recovery counter inhibits the system clocks for 4096 CGMXCLK cycles after the reset or external interrupt.

A mask option determines whether the oscillator stabilization delay during stop recovery is 4096 CGMXCLK cycles or 32 CGMXCLK cycles.

NOTE: *Select the mask option for the full stop recovery time of 4096 CGMXCLK cycles in applications that use an external crystal.*

Clock Generator Module (CGM)

Contents

Introduction

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

Features

Features of the CGM include the following:

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

Functional Description

The CGM consists of three major submodules:

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

Figure 1 shows the structure of the CGM.

Crystal Oscillator Circuit

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

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

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

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

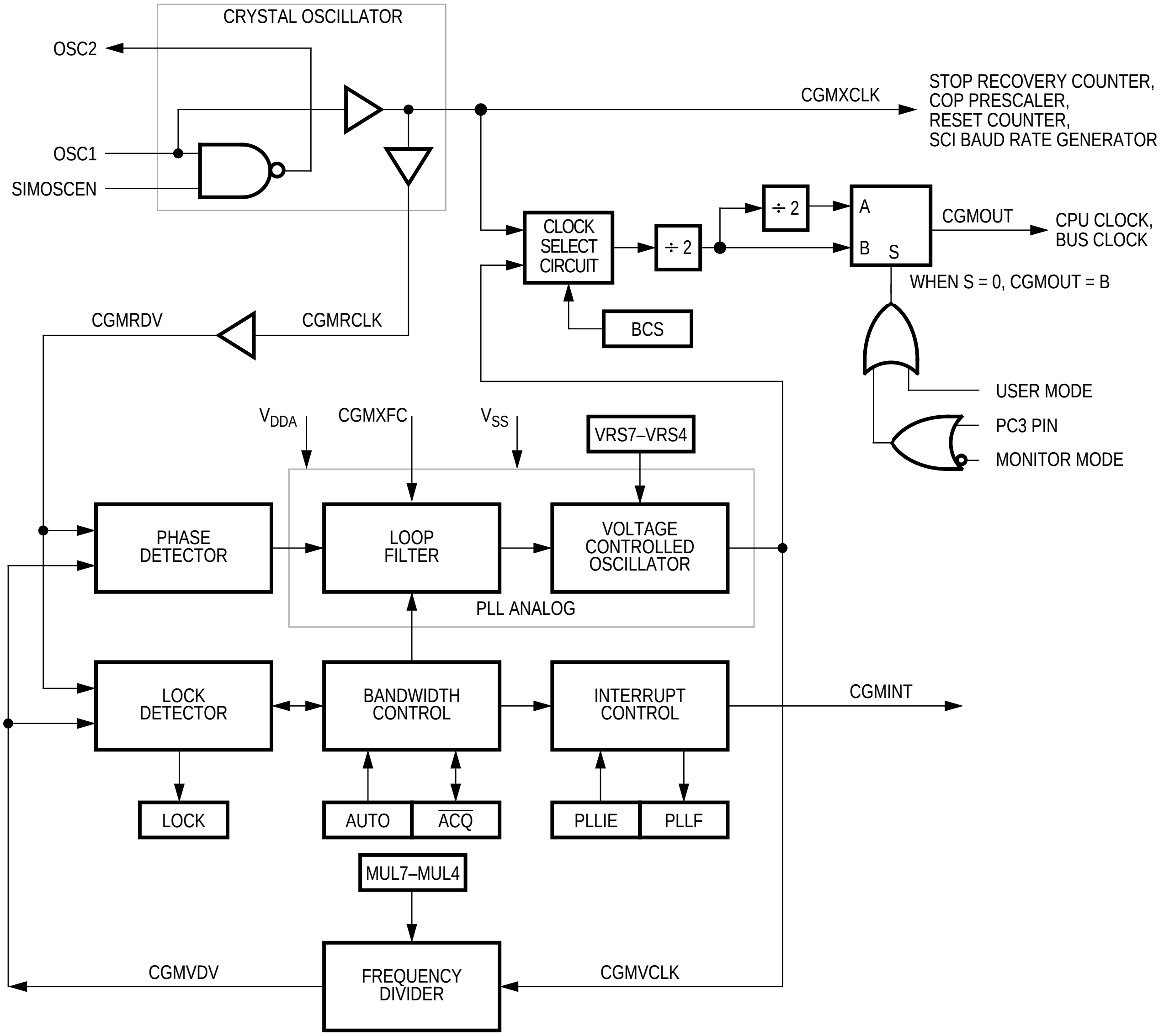


Figure 1. CGM Block Diagram

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

▨ = Unimplemented

Figure 2. I/O Register Summary

Table 1. I/O Register Address Summary

Register	PCTL	PBWC	PPG
Address	$001C	$001D	$001E

Phase-Locked Loop Circuit (PLL)

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

Circuits

The PLL consists of the following circuits:

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

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

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

The VCO's output clock, CGMVCLK, running at a frequency f_{vclk}, is fed back through a programmable modulo divider. The modulo divider reduces the VCO clock by a factor, N. The divider's output is the VCO feedback clock, CGMVDV, running at a frequency $f_{vdv} = f_{vclk}/N$. (See Programming the PLL on page 89 for more information.)

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the dc voltage on the external capacitor connected to CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in Acquisition and Tracking Modes on page 87. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

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

Acquisition and Tracking Modes

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

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL startup or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the $\overline{\text{ACQ}}$ bit is clear in the PLL bandwidth control register. (See PLL Bandwidth Control Register on page 96.)
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. (See Base Clock Selector Circuit on page 91.) The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{\text{ACQ}}$ bit is set.

Manual and Automatic PLL Bandwidth Modes

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

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. (See PLL Bandwidth Control Register on page 96.) If PLL CPU interrupt requests are enabled, the software can wait for a PLL CPU interrupt request and then check the LOCK bit. If CPU interrupts are disabled, software can poll the LOCK bit continuously (during PLL startup, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. (See Base Clock Selector Circuit on page 91.) 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 Interrupts on page 100.)

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

- The $\overline{\text{ACQ}}$ bit (see PLL Bandwidth Control Register on page 96) is a read-only indicator of the mode of the filter. (See Acquisition and Tracking Modes on page 87.)
- 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 Specifications on page 319.)
- 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 Specifications on page 319.)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See PLL Control Register on page 94.)

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

- $\overline{\text{ACQ}}$ 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 Specifications on page 319), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time, t_{al}, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.

Programming the PLL

Use the following procedure to program the PLL.

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

$$\text{Example: } f_{busdes} = 8\text{ MHz}$$

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

$$f_{vclkdes} = 4 \times f_{busdes}$$

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

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

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

$$N = \text{round}\left(\frac{f_{vclkdes}}{f_{rclk}}\right)$$

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

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

$$f_{vclk} = N \times f_{rclk}$$

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

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

$$f_{bus} = \frac{f_{vclk}}{4}$$

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

If the calculated f_{bus} is not within the tolerance limits of your application, select another f_{busdes} or another f_{rclk}.

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: *For proper operation,*

$$|f_{vrs} - f_{vclk}| \leq \frac{f_{nom}}{2}$$

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

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.

Special Programming Exceptions

The programming method described in Programming the PLL on page 89 does not account for two possible exceptions. A value of 0 for N or L is meaningless when used in the equations given. To account for these exceptions:

- A 0 value for N is interpreted the same as a value of 1.
- A 0 value for L disables the PLL and prevents its selection as the source for the base clock. (See Base Clock Selector Circuit on page 91.)

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.

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

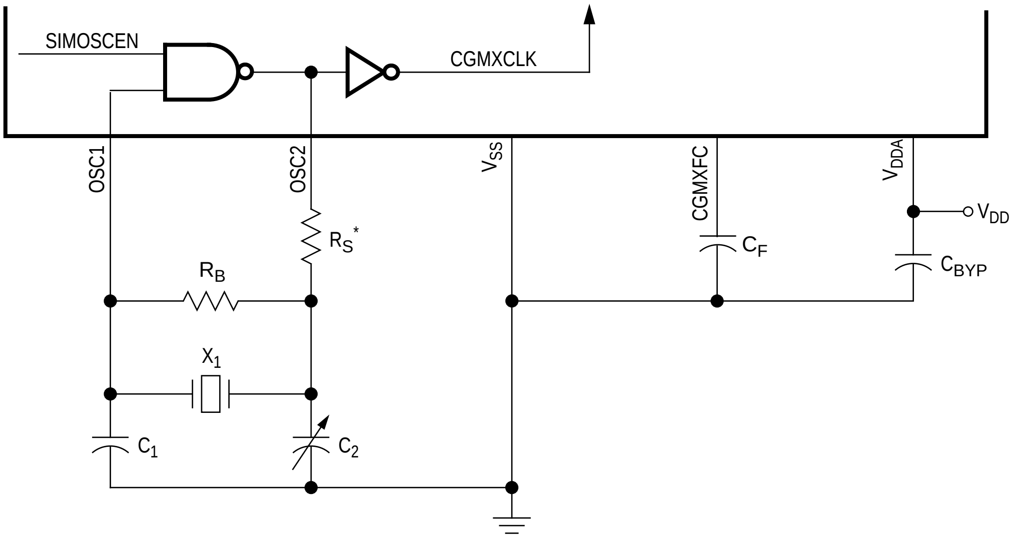
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S (optional)

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

Figure 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 Acquisition/Lock Time Specifications on page 102 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 3. CGM External Connections

I/O Signals

The following paragraphs describe the CGM I/O signals.

Crystal Amplifier Input Pin (OSC1)

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

Crystal Amplifier Output Pin (OSC2)

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

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.*

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.*

Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal enables the oscillator and PLL.

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 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.

CGM Base Clock Output (CGMOUT)

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

CGM CPU Interrupt (CGMINT)

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

CGM Registers

The following registers control and monitor operation of the CGM:

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

PLL Control Register

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

Address: $001C

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

▨ = Unimplemented

Figure 4. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

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

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

PLLF — PLL Flag Bit

This read-only bit is set whenever the LOCK bit toggles. PLLF generates a CPU interrupt request if the PLLIE bit also is set. PLLF always reads as 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 Base Clock Selector Circuit on page 91.) 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 Base Clock Selector Circuit on page 91.) 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 Base Clock Selector Circuit* on page 91.)

Bits 3–0 — Unimplemented

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

PLL Bandwidth Control Register

The PLL bandwidth control register does the following:

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

Address: $001D

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

▨ = Unimplemented

Figure 5. PLL Bandwidth Control Register (PBWC)

AUTO — Automatic Bandwidth Control Bit

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

1 = Automatic bandwidth control
0 = Manual bandwidth control

LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as 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.

1 = Crystal reference not active
0 = Crystal reference active

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

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.

Bits 3–0 — Reserved for Test

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

PLL Programming Register

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

Address: $001E

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

Figure 6. PLL Programming Register (PPG)

MUL7–MUL4 — Multiplier Select Bits

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier, N. (See Circuits on page 85 and Programming the PLL on page 89.) 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 2. VCO Frequency Multiplier (N) Selection

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

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

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 Circuits on page 85, Programming the PLL on page 89, and PLL Control Register on page 94.) VRS7–VRS4 cannot be written when the PLLON bit in the PLL control register (PCTL) is set. (See Special Programming Exceptions on page 90.) A value of $0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See Base Clock Selector Circuit on page 91 and Special Programming Exceptions on page 90 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.

Interrupts

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

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

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

Low-Power Modes

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

Wait Mode

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

Stop Mode

The STOP instruction disables the CGM and holds 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.

CGM During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See Break Module (BRK) on page 141.)

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.

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.

Acquisition/Lock Time Definitions

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

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

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

- Acquisition time, t_{acq}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the tracking mode entry tolerance, Δ_{trk}. Acquisition time is based on an initial frequency error,

$(f_{des} – f_{orig})/f_{des}$, of not more than ±100%. In automatic bandwidth control mode (see Manual and Automatic PLL Bandwidth Modes on page 87), acquisition time expires when the $\overline{\text{ACQ}}$ bit becomes set in the PLL bandwidth control register (PBWC).

- Lock time, t_{Lock}, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the lock mode entry tolerance, Δ_{Lock}. Lock time is based on an initial frequency error, $(f_{des} – f_{orig})/f_{des}$, of not more than ±100%. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). (See Manual and Automatic PLL Bandwidth Modes on page 87.)

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.

Parametric Influences on Reaction Time

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

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

Another critical parameter is the external filter capacitor. The PLL modifies the voltage on the VCO by adding or subtracting charge from this capacitor. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. (See Choosing a Filter Capacitor on page 104.)

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

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

Choosing a Filter Capacitor

As described in Parametric Influences on Reaction Time on page 103, the external filter capacitor, C_F, is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to the following equation:

$$C_F = C_{fact}\left(\frac{V_{DDA}}{f_{rdv}}\right)$$

For acceptable values of C_{fact}, see Specifications on page 319. For the value of V_{DDA}, choose the voltage potential at which the MCU is operating. If the power supply is variable, choose a value near the middle of the range of possible supply values.

This equation does not always yield a commonly available capacitor size, so round to the nearest available size. If the value is between two different sizes, choose the higher value for better stability. Choosing the lower size may seem attractive for acquisition time improvement, but the

PLL may become unstable. Also, always choose a capacitor with a tight tolerance (±20% or better) and low dissipation.

Reaction Time Calculation

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

- Correct selection of filter capacitor, C_F (See Choosing a Filter Capacitor on page 104.)
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters. K_{acq} is the K factor when the PLL is configured in acquisition mode, and K_{trk} is the K factor when the PLL is configured in tracking mode. (See Acquisition and Tracking Modes on page 87.)

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

$$t_{al} = \left(\frac{V_{DDA}}{f_{rdv}}\right)\left(\frac{4}{K_{trk}}\right)$$

$$t_{Lock} = t_{acq} + t_{al}$$

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

In automatic bandwidth control mode, the acquisition and lock times are quantized into units based on the reference frequency. (See Manual and Automatic PLL Bandwidth Modes on page 87.) A certain number of clock cycles, n_{acq}, is required to ascertain that the PLL is within the tracking mode entry tolerance, Δ_{trk}, before exiting acquisition mode. A certain number of clock cycles, n_{trk}, is required to ascertain that the PLL is within the lock mode entry tolerance, Δ_{Lock}. Therefore, the acquisition time, t_{acq}, is an integer multiple of n_{acq}/f_{rdv}, and the acquisition to lock time, t_{al}, is an integer multiple of n_{trk}/f_{rdv}. Also, since the average frequency over the entire measurement period must be within the specified tolerance, the total time usually is longer than t_{Lock} as calculated above.

In manual mode, it is usually necessary to wait considerably longer than t_{Lock} before selecting the PLL clock (see Base Clock Selector Circuit on page 91) because the factors described in Parametric Influences on Reaction Time on page 103 may slow the lock time considerably.

Direct Memory Access Module (DMA)

Contents

Introduction

The DMA can perform data transfers to and from any two CPU-addressable locations without CPU intervention.

Features

Features of the DMA include the following:

- Modular Architecture
- Service Request-Driven Operation without CPU Intervention
- Three Independent Channels
- Byte or Word Transfer Capability
- Block Transfers and Loop Transfers
- CPU Interrupt Capability on Completion of Block Transfer or on Loop Restart
- Programmable DMA Bus Bandwidth (25%, 50%, 67%, or 100% of Total Bus Bandwidth)
- Programmable DMA Service Request/CPU Interrupt Request Priority
- Programmable DMA Enable during Wait Mode
- Block Transfers Up to 256 Bytes
- Expandable Architecture Up to Seven Channels and Eight Transfer Source Inputs

Functional Description

The DMA is a coprocessor for servicing peripheral devices that require data block transfers. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts. The following tasks that contribute to CPU interrupt overhead are not part of a DMA transfer:

- Stacking and unstacking CPU registers
- Loading interrupt vectors
- Loading address pointers
- Incrementing address pointers
- Storing address pointers
- Clearing interrupt flags
- Returning from interrupt

Once the DMA is initialized to transfer a block of data, a DMA service request usually requires only two bus cycles per 8-bit byte or four cycles per 16-bit word to transfer the source data to a destination.

Figure 1 shows the structure of the DMA. Each DMA channel can transfer data independently between any addresses in the memory map.

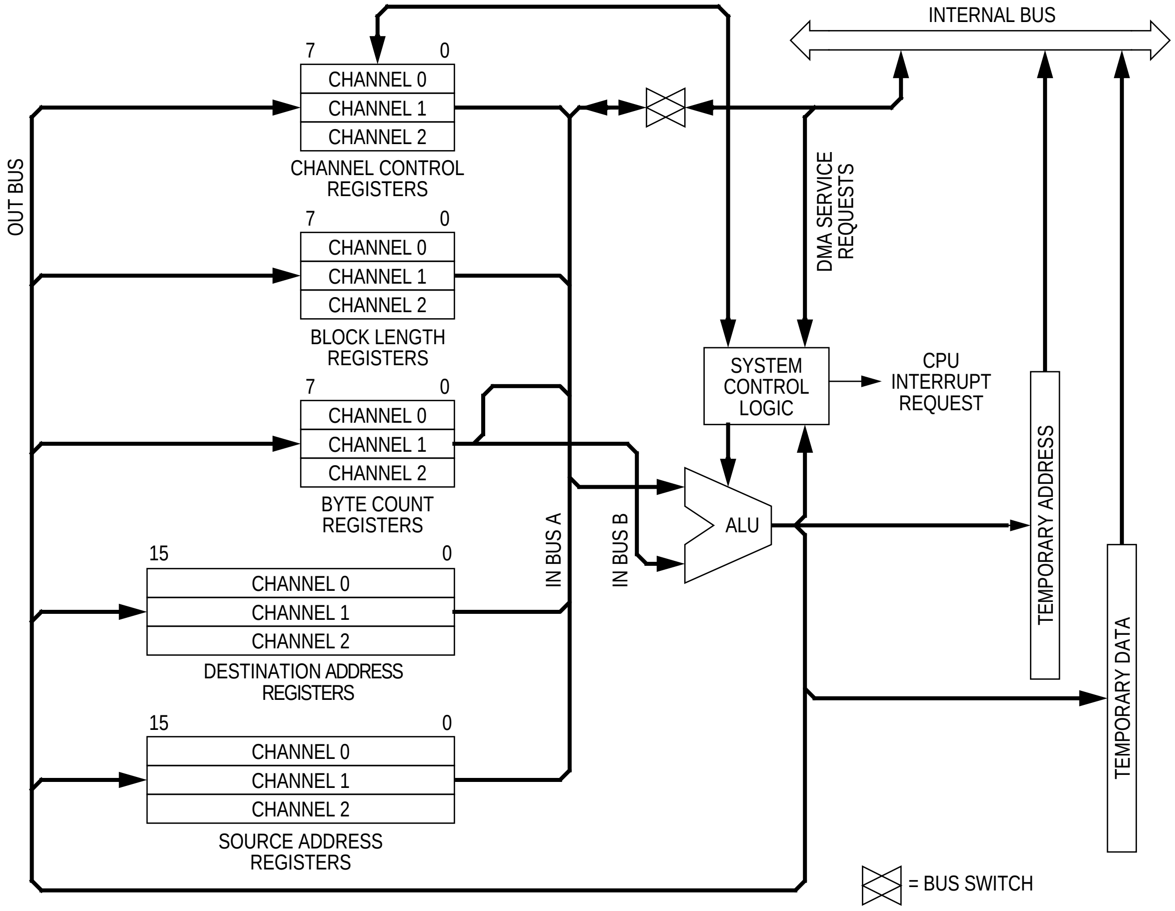


Figure 1. DMA Module Block Diagram

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
DMA Channel 0 Source Address Register High (D0SH)	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Reset:	Indeterminate after Reset							
DMA Channel 0 Source Address Register Low (D0SL)	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after Reset							
DMA Channel 0 Destination Address Register High (D0DH)	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Reset:	Indeterminate after Reset							
DMA Channel 0 Destination Address Register Low (D0DL)	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after Reset							

Figure 2. I/O Register Summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
DMA Channel 0 Control Register (D0C)	Read: Write:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
	Reset:	Indeterminate after Reset							
DMA Channel 0 Block Length Register (D0BL)	Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
	Reset:	Indeterminate after Reset							
DMA Channel 0 Byte Count Register (D0BC)	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
	Reset:	0	0	0	0	0	0	0	0
DMA Channel 1 Source Address Register High (D1SH)	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Reset:	Indeterminate after Reset							
DMA Channel 1 Source Address Register Low (D1SL)	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after Reset							
DMA Channel 1 Destination Address Register High (D1DH)	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Reset:	Indeterminate after Reset							
DMA Channel 1 Destination Address Register Low (D1DL)	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after Reset							
DMA Channel 1 Control Register (D1C)	Read: Write:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
	Reset:	Indeterminate after Reset							
DMA Channel 1 Block Length Register (D1BL)	Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
	Reset:	Indeterminate after Reset							
DMA Channel 1 Byte Count Register (D1BC)	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
	Reset:	0	0	0	0	0	0	0	0
DMA Channel 2 Source Address Register High (D2SH)	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Reset:	Indeterminate after Reset							
DMA Channel 2 Source Address Register Low (D2SL)	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after Reset							
DMA Channel 2 Destination Address Register High (D2DH)	Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
	Reset:	Indeterminate after Reset							
DMA Channel 2 Destination Address Register Low (D2DL)	Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
	Reset:	Indeterminate after Reset							

Figure 2. I/O Register Summary (Continued)

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
DMA Channel 2 Control Register (D2C)	Read: Write:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
	Reset:	Indeterminate after Reset							
DMA Channel 2 Block Length Register (D2BL)	Read: Write:	BL7	BL6	BL5	BL4	BL3	BLL2	BL1	BL0
	Reset:	Indeterminate after Reset							
DMA Channel 2 Byte Count Register (D2BC)	Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
	Reset:	0	0	0	0	0	0	0	0
DMA Control Register 1 (DC1)	Read: Write:	BB1	BB0	TEC2	IEC2	TEC1	IEC1	TEC0	IEC0
	Reset:	0	0	0	0	0	0	0	0
DMA Status and Control Register (DSC)	Read: Write:	DMAP	L2	L1	L0	DMAWE	IFC2	IFC1	IFC0
	Reset:	0	0	0	0	0	0	0	0
DMA Control Register 2 (DC2)	Read: Write:	SWI7	SWI6	SWI5	SWI4	SWI33	SWI2	SWI1	SWI0
	Reset:	0	0	0	0	0	0	0	0

Figure 2. I/O Register Summary (Continued)

Table 1. I/O Register Address Summary

Register	D0SH	D0SL	D0DH	D0DL	D0C	D0BL	D0BC	D1SH	D1SL	D1DH
Address	$0034	$0035	$0036	$0037	$0038	$0039	$003B	$003C	$003D	$003E

Register	D1DL	D1C	D1BL	D1BC	D2SH	D2SL	D2DH	D2DL	D2C	D2BL
Address	$003F	$0040	$0041	$0043	$0044	$0045	$0046	$0047	$0048	$0049

Register	D2BC	DC1	DSC	DC2
Address	$004B	$004C	$004D	$004E

DMA/CPU Timing

When the DMA transfers data, it takes control of the address bus, data bus, and R/$\overline{\text{W}}$ line. During DMA transfers, the CPU clock is suspended. The state of the CPU remains unchanged until the end of the DMA transfer when the DMA relinquishes control of the buses and R/$\overline{\text{W}}$ line. Then the CPU resumes operation as though nothing had happened.

Figure 3 and **Figure 4** show the timing of DMA transfers.

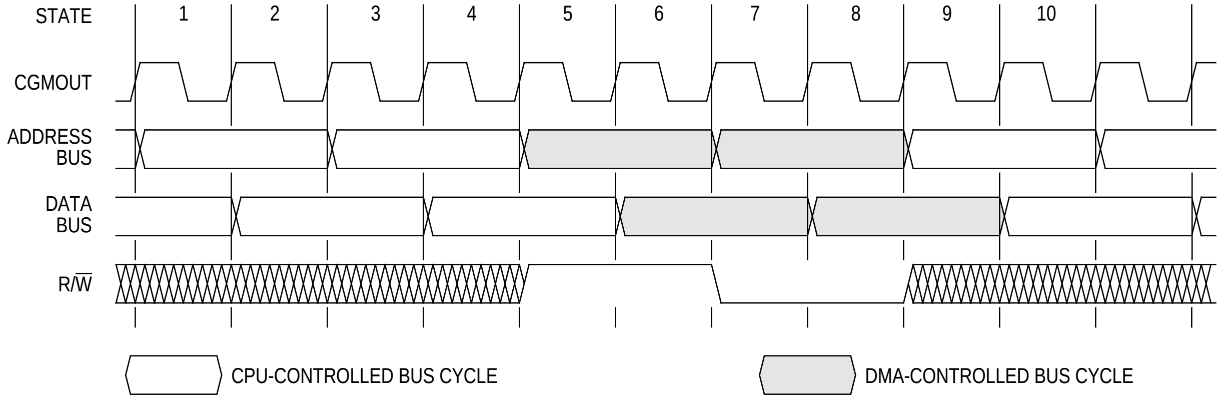


Figure 3. Single Byte Transfer Timing (Any DMA Bus Bandwidth)

Table 2. DMA Byte Transfer Activity

State	Activity
1	DMA service request occurs.
2	DMA arbitrates channel priority.
3	DMA generates internal control signals.
4	DMA calculates source address. DMA latches source address in temporary register.
5	DMA drives source address onto address bus. DMA drives $R/\overline{W}$ line high. DMA calculates destination address. DMA latches destination address into temporary register.
6	DMA latches source data into temporary register. DMA increments byte count register.
7	DMA drives destination address onto address bus. DMA drives $R/\overline{W}$ line low. DMA subtracts byte count register from block length register. If difference = 0, DMA disables channel by clearing TECx bit. If difference = 0 and IECx = 1, DMA generates CPU interrupt request.
8	DMA drives source data onto data bus.
9	DMA releases address bus and $R/\overline{W}$ line to CPU.
10	DMA releases data bus to CPU.

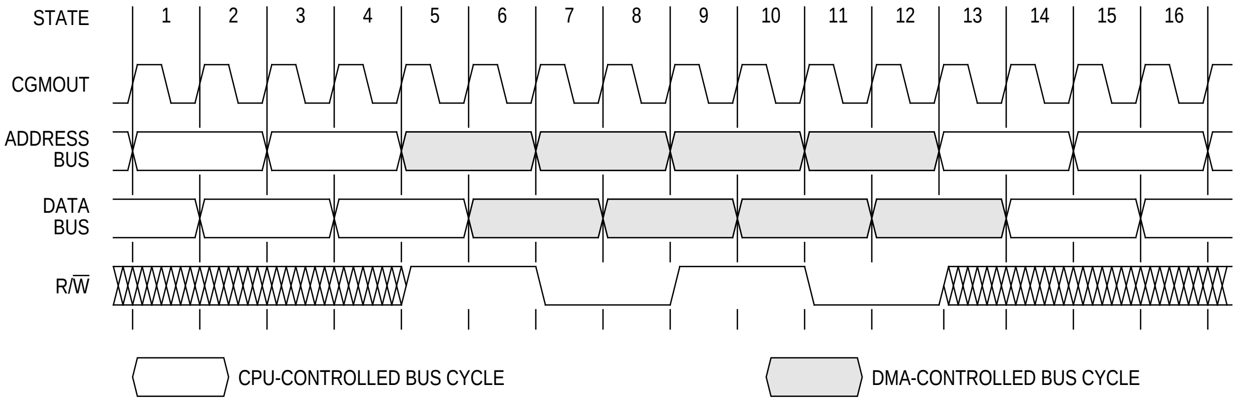


Figure 4. Single Word Transfer Timing (100% DMA Bus Bandwidth)

Table 3. DMA Word Transfer Activity

State	Activity
1	DMA service request occurs.
2	DMA arbitrates channel priority.
3	DMA generates internal control signals.
4	DMA calculates low byte of source address. DMA latches low byte of source address in temporary register.
5	DMA drives low byte of source address onto address bus. DMA drives R/$\overline{W}$ line high. DMA calculates low byte of destination address. DMA latches low byte of destination address into temporary register.
6	DMA latches low byte of source data into temporary register. DMA increments byte count register.
7	DMA drives low byte of destination address onto address bus. DMA drives R/$\overline{W}$ line low. DMA subtracts byte count register from block length register. If difference = 0, DMA disables channel by clearing TECx bit. If difference = 0 and IECx = 1, DMA generates CPU interrupt request.
8	DMA drives low byte of source data onto data bus. DMA calculates high byte of source address. DMA latches high byte of source address into temporary register.
9	DMA drives the high byte of source address onto address bus. DMA drives R/$\overline{W}$ line high. DMA calculates high byte of destination address. DMA latches high byte of destination address in temporary register.
10	DMA latches high byte of source data into temporary register. DMA increments the byte count register.
11	DMA drives high byte of destination address onto address bus. DMA drives R/$\overline{W}$ line low. DMA subtracts byte count register from block length register. If difference = 0, DMA disables channel by clearing TECx bit. If difference = 0 and IECx bit set, CPU receives interrupt request.
12	DMA drives high byte of destination address onto address bus.
13	DMA releases the address bus and R/$\overline{W}$ line to CPU.
14	DMA releases data bus to CPU.

The following procedure shows how to program a DMA transfer on a selected channel:

1. In DMA control register 1 (DC1), disable the channel by clearing the TECx bit. (See DMA Control Register 1 on page 129.)
2. In the source address registers (DxSH and DxSL), write the source base address. (See DMA Source Address Registers on page 137.)
3. In the destination address registers (DxDH and DxDL), write the destination base address. (See DMA Destination Address Registers on page 138.)
4. In the DMA channel x control register (DxC), make the following selections (see DMA Channel Control Registers on page 135):
   a. Select increment, decrement, or remain static for the source and destination addresses by writing to the source/destination address control bits, SDC3–SDC0.

   b. Select 8-bit or 16-bit data by writing to the byte/word control bit, BWC.

   c. Assign a DMA channel to the DMA transfer source input by writing to the DMA transfer source bits, DTS2–DTS0.
5. In the channel x DMA block length register (DxBL), write the number of bytes to transfer. (See DMA Block Length Registers on page 139.) For word transfers, the block length number is two times the number of words.
6. In the DMA status and control register (DSC), make the following selections (see DMA Status and Control Register on page 131):
   a. Enable or disable looping of the source and destination addresses by writing to the loop enable bit, Lx.

   b. Select DMA service request/CPU interrupt request priority by writing to the DMA priority bit, DMAP.

   c. Enable or disable DMA transfers during wait mode by writing to the DMA wait enable bit, DMAWE.

7. In DMA control register 1 (DC1), make the following selections (see DMA Control Register 1 on page 129):
   a. Enable the DMA channel x by writing to the transfer enable bit, TECx.
   b. Enable or disable DMA channel x to generate CPU interrupts on transfer completion by writing to the CPU interrupt enable bit, IECx.
   c. Select the DMA bus bandwidth by writing to the bus bandwidth control bits, BB0 and BB1.
8. To initiate the DMA transfer with software, set the software initiate bit, SWIx, in DMA control register 2 (DC2). (See DMA Control Register 2 on page 134.)

Hardware-Initiated DMA Service Requests

The following sources can generate DMA service requests:

- Timer interface module TIM) — The TIM can generate the following DMA service requests:
  - TIM channel 0 input capture/output compare
  - TIM channel 1 input capture/output compare
  - TIM channel 2 input capture/output compare
  - TIM channel 3 input capture/output compare
- Serial peripheral interface module (SPI) — The SPI can generate the following DMA service requests:
  - SPI receiver full
  - SPI transmitter empty
- Serial communications interface module (SCI) — The SCI can generate the following DMA service requests:
  - SCI receiver full
  - SCI transmitter empty

The DMA has eight inputs for transfer sources. Each DMA transfer source input corresponds to one of the above DMA service requests. To enable a transfer on one of the three DMA channels, software must first assign the channel to one of the transfer source inputs. The channel control register of each channel determines its transfer source assignment. (See Table 10. DMA Transfer Source Selection.)

Software-Initiated DMA Service Requests

Software can initiate a DMA service request by writing to DMA control register 2 (DC2). A software-initiated transfer begins when the following conditions are met:

- The channel is enabled by the channel x transfer enable bit, TECx, in DMA control register 1 (DC1).
- The channel is assigned to a DMA transfer source input by the DMA transfer source bits, DTS2–DTS0, in the channel x control register (DxC).
- The corresponding software initiate bit, SWIx, in DMA control register 2 (DC2) is set, enabling a transfer on the transfer source input to which the channel is assigned.

During a DMA transfer on channel x, the channel x byte count register increments with every byte transferred. When the value in the channel x byte count register matches the value in the channel x block length register, the channel x CPU interrupt flag, IFCx, becomes set. If the channel x CPU interrupt enable bit, IECx, is also set, the DMA issues a CPU interrupt request.

DMA Latency

When one DMA channel is active, the normal DMA latency is two cycles. Writing to the destination/source address registers, the channel control registers, or the block length registers of another DMA channel during a transfer adds three cycles to DMA latency.

If more than one DMA channel is active, the latency of lower-priority channels increases.

If two or more DMA channels have pending service requests, at least one CPU cycle executes between each channel transfer.

DMA Source/Destination Address Calculation

Three 16-bit buses connect the 16-bit DMA arithmetic/logic unit (ALU) to the DMA channel registers. During a DMA transfer, the DMA ALU:

- Calculates the transfer source and transfer destination addresses.
- Increments the DMA byte count register for each byte transferred.
- Determines when a block or loop transfer is complete by comparing the DMA byte count register with the value programmed in the DMA block length register.

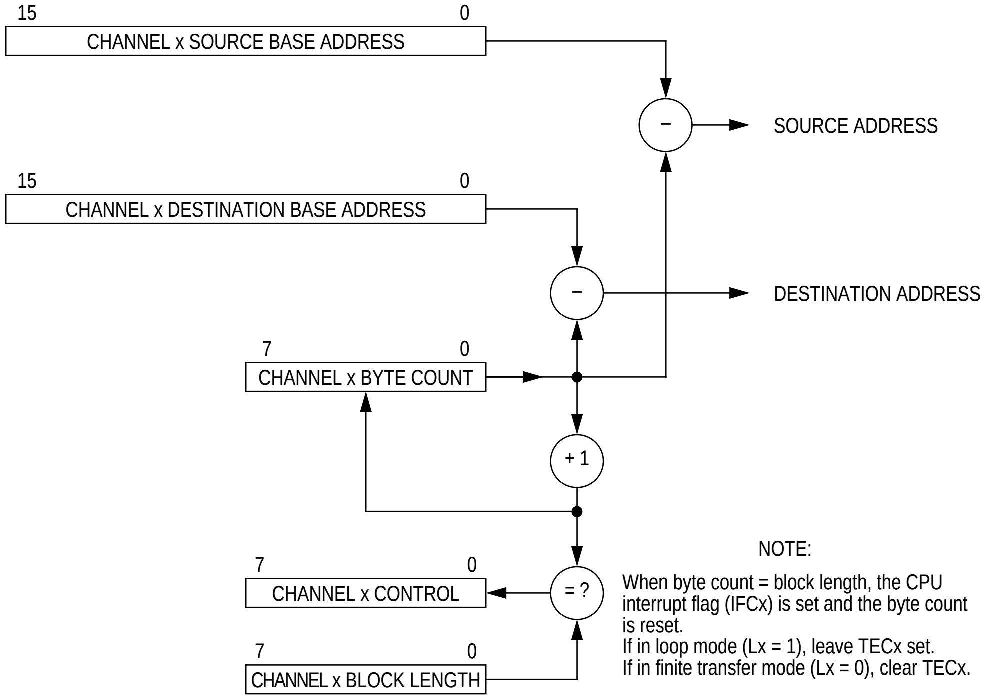
The DMA source address register and destination address register contain the base addresses for a DMA transfer. The DMA ALU uses these address registers as base pointers when it starts the transfer. The DMA byte count register contains the number of bytes transferred in the current DMA operation. The DMA ALU uses the source/destination address registers and the byte count register to calculate the actual source and destination addresses in the following manner:

- When an address is configured to increment, the DMA ALU adds the byte count register to the base address.
- When an address is configured to decrement, the DMA ALU subtracts the byte counter register from the base address.
- When an address is configured to remain static, the DMA ALU uses the base address as is.

The DMA can be programmed to:

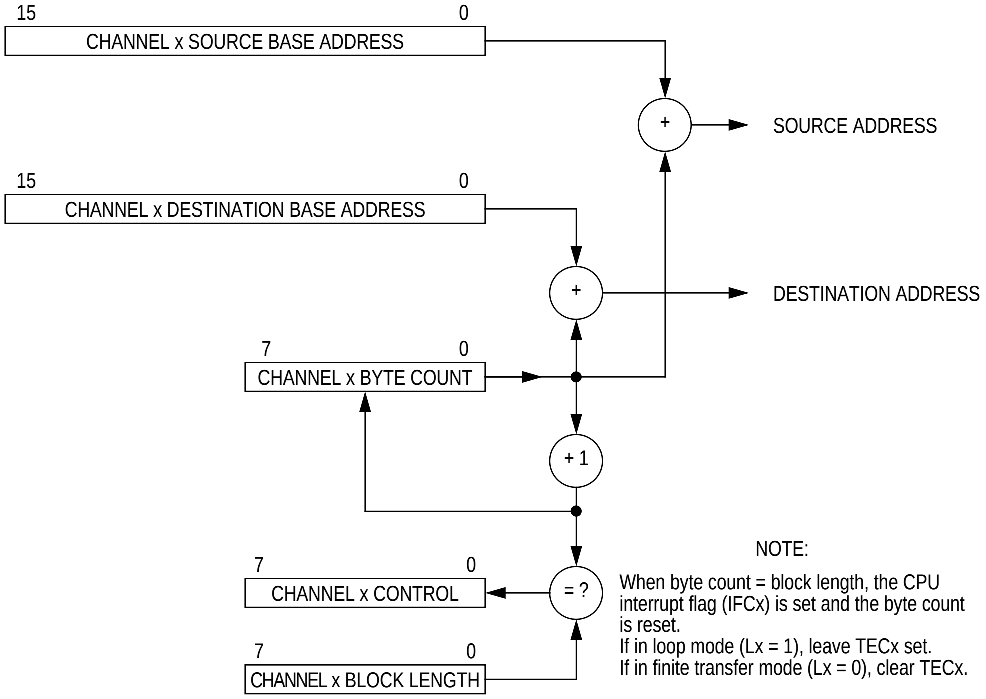
- Stop the transfer after a number of bytes is transferred or
- After a number of bytes is transferred, loop back to the base addresses and continue the transfer.

Figure 5 through **Figure 13** show how the DMA calculates source and destination addresses.



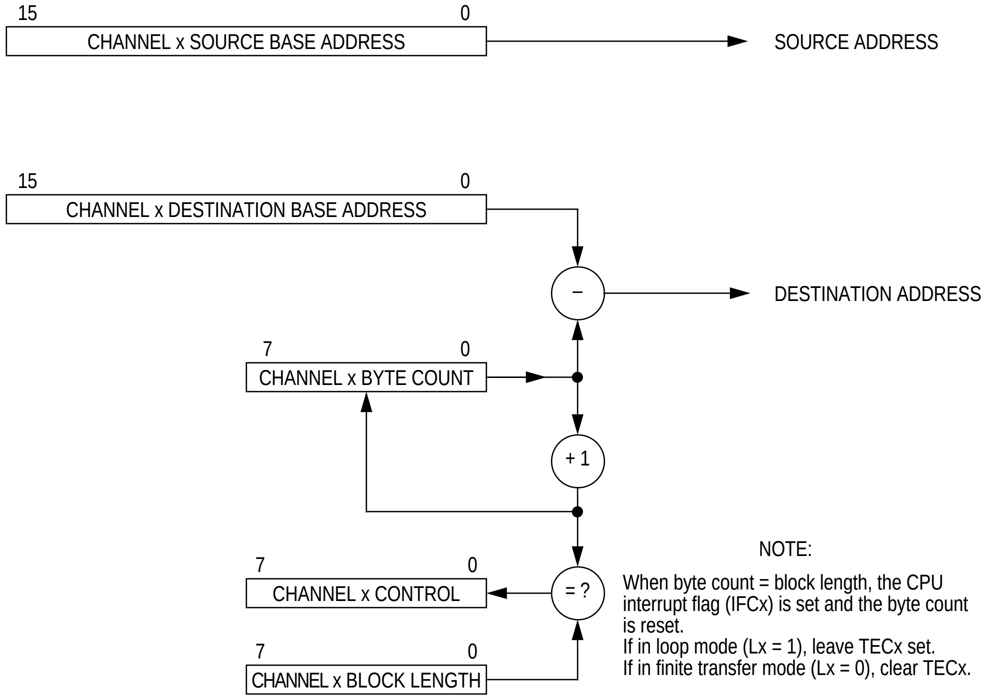
NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 5. Decremented Source and Decremented Destination



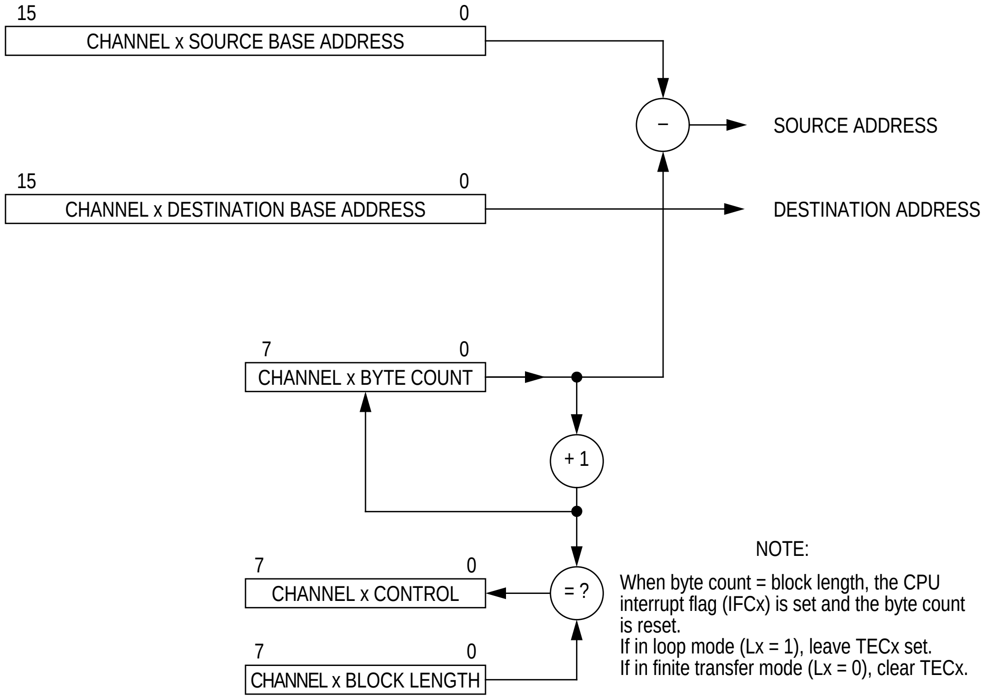
NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 6. Incremented Source and Incremented Destination



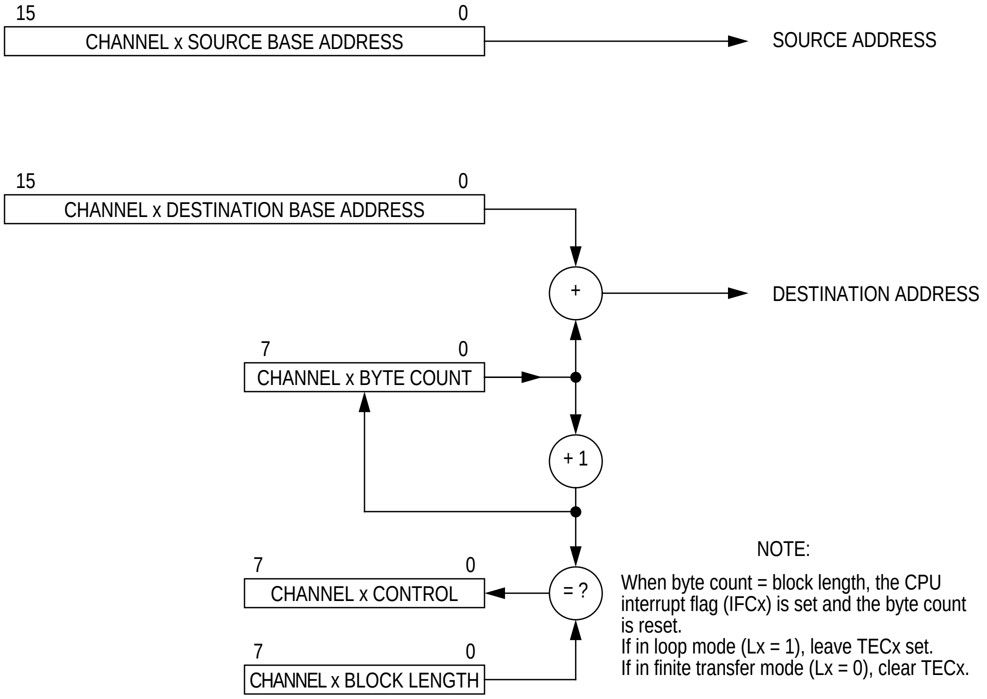
NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 7. Static Source and Decremented Destination



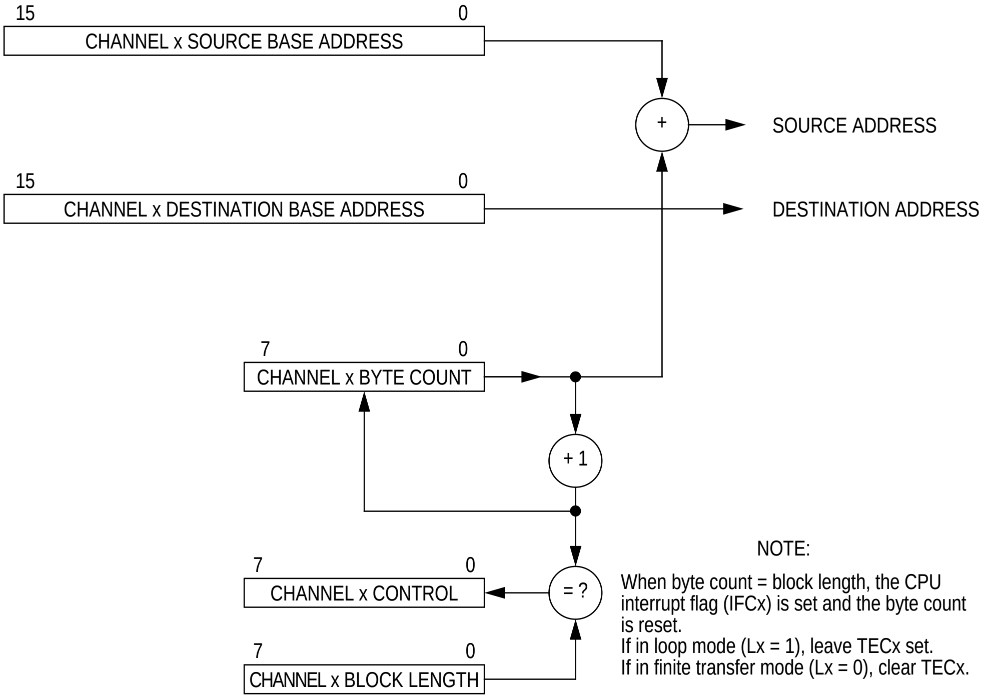
NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 8. Decremented Source and Static Destination



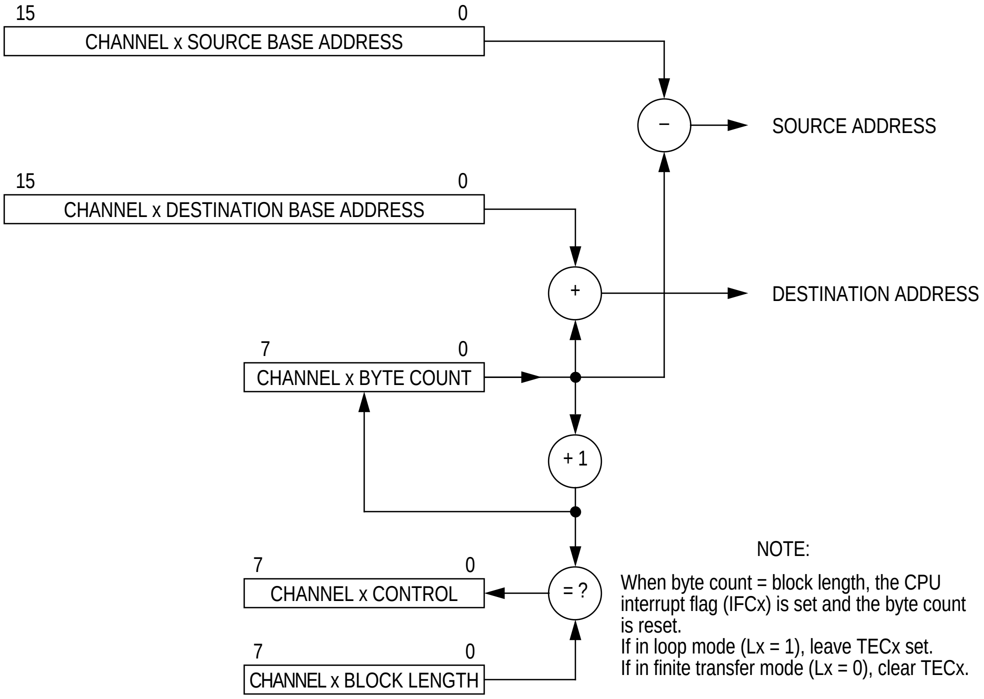
NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 9. Static Source and Incremented Destination



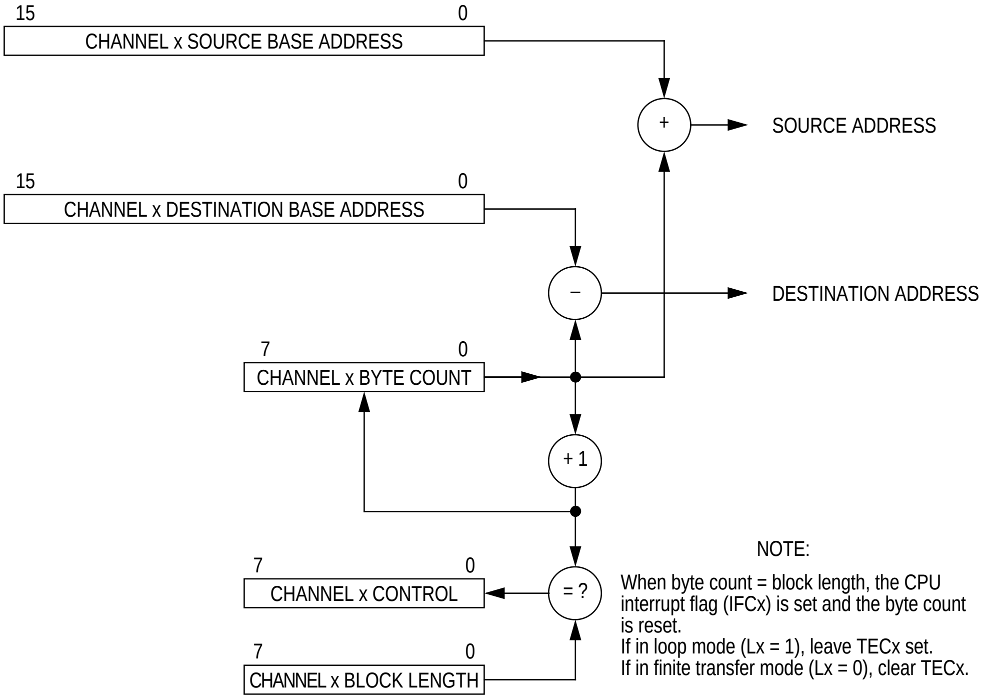
NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 10. Incremented Source and Static Destination



NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 11. Decremented Source and Incremented Destination



NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 12. Incremented Source and Decremented Destination

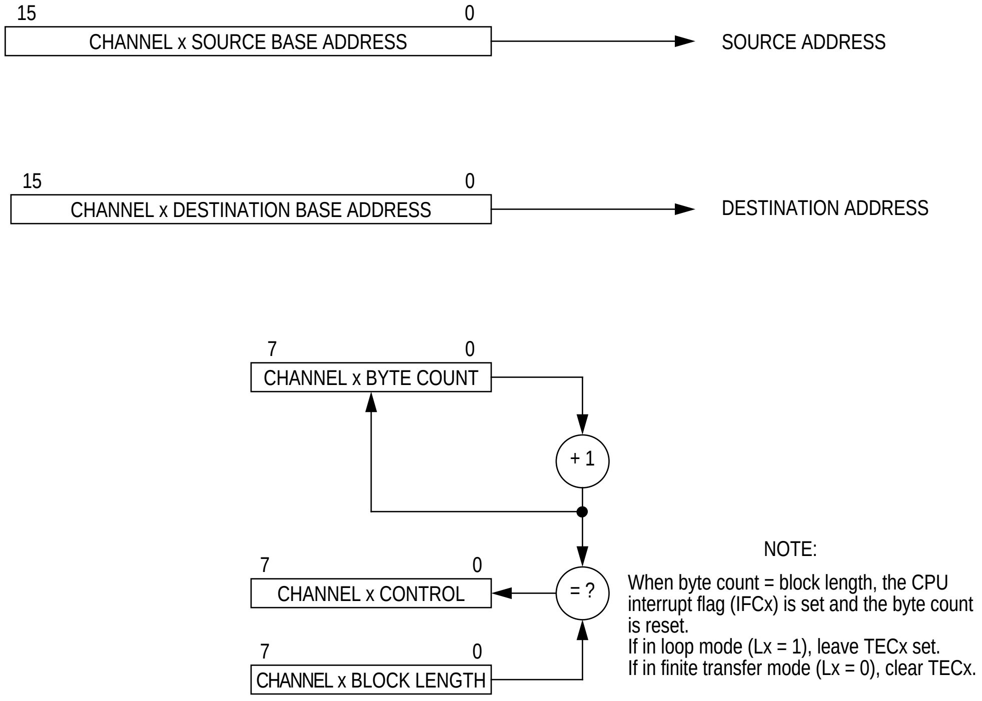

15
0
CHANNEL x SOURCE BASE ADDRESS
SOURCE ADDRESS
15
0
CHANNEL x DESTINATION BASE ADDRESS
DESTINATION ADDRESS
7
0
CHANNEL x BYTE COUNT
+ 1
7
0
CHANNEL x CONTROL
= ?
7
0
CHANNEL x BLOCK LENGTH
NOTE:
When byte count = block length, the CPU interrupt flag (IFCx) is set and the byte count is reset.
If in loop mode (Lx = 1), leave TECx set.
If in finite transfer mode (Lx = 0), clear TECx.

Figure 13. Static Source and Static Destination

Table 4. DMA Address Calculation (Byte Mode)

	Static Source	**Static Destination**	**Incremented Source**	**Static Destination**	**Static Source**	**Incremented Destination**	**Decremented Source**	**Static Destination**	**Static Source**	**Decremented Destination**
Byte	**From**	**To**	**From**	**To**	**From**	**To**	**From**	**To**	**From**	**To**
1	S[1]	D[2]	S	D	S	D	S	D	S	D
2	S	D	S + 1	D	S	D + 1	S – 1	D	S	D – 1
3	S	D	S + 2	D	S	D + 2	S – 2	D	S	D – 2
4	S	D	S + 3	D	S	D + 3	S – 3	D	S	D – 3
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
n	S	D	S + n – 1	D	S	D + n – 1	S – (n – 1)	D	S	D – (n – 1)

[1] S = Source base address
[2] D = Destination base address

Table 5. DMA Address Calculation (Word Mode)

		Static Source	Static Destination	Incremented Source	Static Destination	Static Source	Incremented Destination	Decremented Source	Static Destination	Static Source	Decremented Destination
Word	**Byte**	**From**	**To**	**From**	**To**	**From**	**To**	**From**	**To**	**From**	**To**
1	1	S[1]	D[2]	S	D	S	D	S	D	S	D
	2	S + 1	D + 1	S + 1	D + 1	S + 1	D + 1	S – 1	D + 1	S + 1	D – 1
2	3	S	D	S + 2	D	S	D + 2	S – 2	D	S	D – 2
	4	S + 1	D + 1	S + 3	D + 1	S + 1	D + 3	S – 3	D + 1	S + 1	D – 3
3	5	S	D	S + 4	D	S	D + 4	S – 4	D	S	D – 4
	6	S + 1	D + 1	S + 5	D + 1	S + 1	D + 5	S – 5	D + 1	S + 1	D – 5
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
n	2n – 1	S	D	S + 2n – 2	D	S	D + 2n – 2	S – (2n – 2)	D	S	D – (2n – 2)
	2n	S + 1	D + 1	S + 2n – 1	D + 1	S + 1	D + 2n – 1	S – (2n – 1)	D + 1	S + 1	D – (2n – 1)

[1] S = Source base address

[2] D = Destination base address

Low-Power Modes

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

Wait Mode

If enabled by the DMAWE bit in the DMA status and control register, the DMA remains active in wait mode. The DMA can transfer data to and from peripherals while the MCU remains in wait mode.

If the WAIT instruction occurs during a DMA transfer while DMAWE is set, the DMA transfer continues to completion. If the DMAWE bit is clear, a WAIT instruction suspends the current DMA transfer. If the DMA priority bit (DMAP) is set, the suspended transfer resumes when the MCU exits wait mode.

Stop Mode

The DMA is inactive during stop mode. A STOP instruction suspends any DMA transfer in progress. If an external interrupt brings the MCU out of stop mode and the DMA priority bit (DMAP) is set, the suspended DMA transfer resumes. If a reset brings the MCU out of stop mode, the transfer is aborted.

Entering stop mode when a DMA channel is enabled may fail to clear the the interrupt mask (I bit) in the condition code register. To make sure the I bit is cleared when entering stop mode:

- Before executing the STOP instruction, wait until any current DMA transfer is complete. Then disable DMA transfers by clearing bits TEC2–TEC0 in DMA control register 1.

  Or,

- Execute the clear-interrupt-mask instruction (CLI) before entering stop mode.

DMA During Break Interrupts

If the DMA is enabled, clear the DMAP bit in the DMA status and control register before executing a break interrupt.

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

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

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See Break Module (BRK) on page 141.)

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.

DMA Registers

The following registers control and monitor operation of the DMA:

- DMA control register 1 (DC1)
- DMA status and control register (DSC)
- DMA control register 2 (DC2)

DC1, DSC, and DC2 can be written during a DMA transfer without affecting DMA latency.

The following registers control operation of each of the DMA channels:

- DMA source address registers, high and low (D0SH:D0SL, D1SH:D1SL, and D2SH:D2SL)
- DMA destination address registers, high and low (D0DH:D0DL, D1DH:D1DL, and D2DH:D2DL)
- DMA channel x control registers (D0C–D2C)
- DMA channel x byte count registers (D0BC–D2BC)
- DMA channel x block length registers (D0BL–D2BL)

Writing to DxSH:DxSL, DxDH:DxDL, DxC, and DxBL during a transfer affects DMA latency. A write to a channel x control register during a transfer has a two-bus cycle latency if the transfer is first suspended by disabling the channel. Disable the channel by writing a 0 to the TECx bit in DMA control register 1. Without first suspending the transfer, a write to a channel x control register during a transfer has a three-bus cycle latency.

DMA Control Register 1

DMA control register 1:

- Enables channels to transfer data when DMA service requests occur.
- Enables channels to generate CPU interrupt requests.
- Controls how much of the bus bandwidth the DMA uses.

Address: $004C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BB1	BB0	TEC2	IEC2	TEC1	IEC1	TEC0	IEC0
Reset:	0	0	0	0	0	0	0	0

Figure 14. DMA Control Register 1 (DC1)

BB1 and BB0 — Bus Bandwidth Control Bits

These read/write bits control the ratio of DMA/CPU bus activity during a DMA transfer. As **Table 6** shows, the DMA can use 25%, 50%, 67%, or 100% of the bus bandwidth. Reset clears bits BB1 and BB0.

Table 6. DMA/CPU Bus Control Selection

	DMA Transfer	
BB1:BB0	**DMA Bus Cycles**	**CPU Bus Cycles**
00	2 (25%)	6 (75%)
01	2 (50%)	2 (50%)
10	2 (67%)	1 (33%)
11	All (100%)	0 (0%)

Figure 15, **Figure 16**, and **Figure 17** show the timing of DMA transfers with DMA bus bandwidths of 25%, 50%, and 67%.

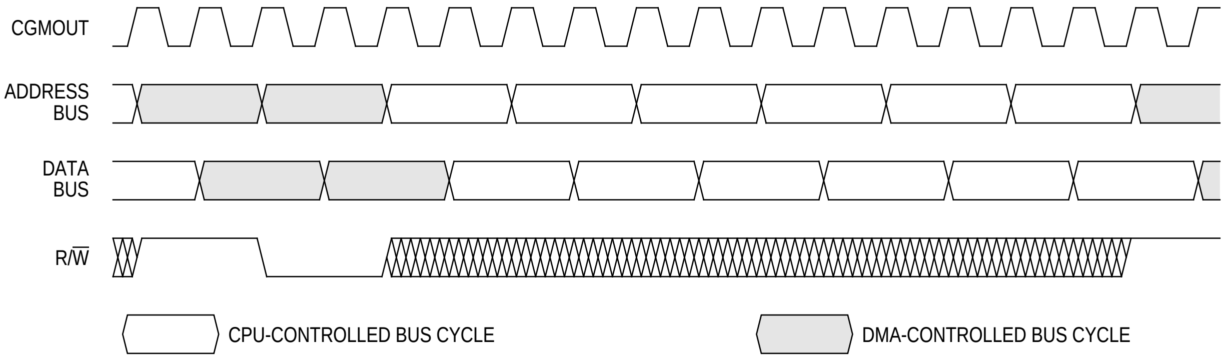


Figure 15. Multiple Byte/Word Transfer Timing: 25% DMA Bus Bandwidth

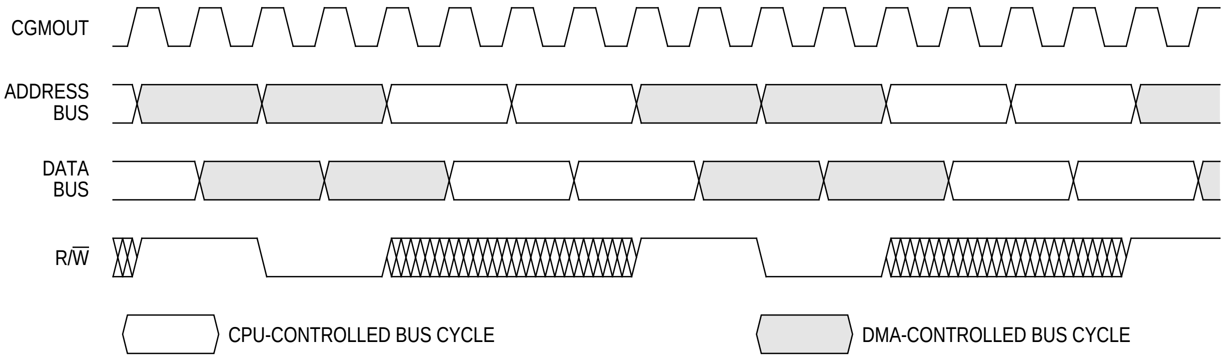


Figure 16. Multiple Byte/Word Transfer Timing: 50% DMA Bus Bandwidth

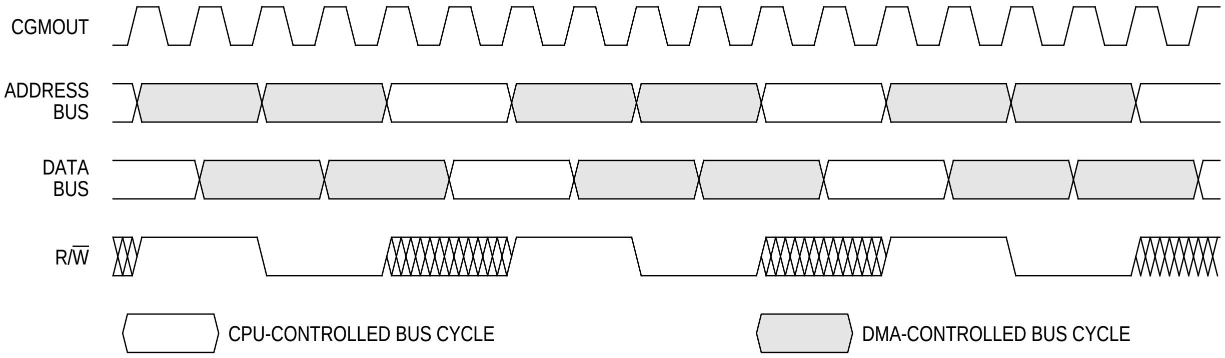


Figure 17. Multiple Byte/Word Transfer Timing: 67% DMA Bus Bandwidth

NOTE: *When two or more DMA channels have transfers pending, the CPU executes at least one cycle between each DMA block length, even if the DMA channels have 100% of the bus bandwidth.*

For DMA transfers of one byte or one word, giving the DMA 100% of the bus bandwidth is appropriate. However, for large, software-initiated transfers, limiting the bus bandwidth of the DMA may be useful to keep from slowing CPU activity.

TEC2–TEC0 — Transfer Enable Bits

These read/write bits enable the corresponding channels to perform transfers when DMA service requests occur. When two or more channels are enabled, a transfer on one channel cannot begin while another channel is transferring a byte or word. Reset clears the TEC2–TEC0 bits.

1 = Corresponding DMA channel enabled
0 = Corresponding DMA channel disabled

IEC2–IEC0 — CPU Interrupt Enable Bits

These read/write bits enable the corresponding channels to generate CPU interrupt requests upon completion of DMA block transfers or at the restart of DMA transfer loops. Reset clears the IEC2–IEC0 bits.

1 = CPU interrupts from corresponding channel enabled
0 = CPU interrupts from corresponding channel disabled

DMA Status and Control Register

The DMA status and control register:

- Flags completion of DMA transfers.
- Controls looping of source and destination address counts.
- Controls priority of DMA service requests and CPU interrupt requests.

Address: $004D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DMAP	L2	L1	L0	DMAWE	IFC2	IFC1	IFC0
Reset:	0	0	0	0	0	0	0	0

Figure 18. DMA Status and Control Register (DSC)

DMAP — DMA Priority Bit

This read/write bit controls the priority of CPU interrupt requests during DMA transfers. Reset clears the DMAP bit.

1 = CPU interrupt requests inhibited during DMA transfers — When DMAP is set, a CPU interrupt request is not recognized until the end of the current DMA transfer. During a block transfer, the increase in CPU interrupt latency depends on the block size and on the bus bandwidth bits, BB1 and BB0. (See DMA Control Register 1 on page 129.)

0 = CPU interrupt requests recognized during DMA transfers — When DMAP is clear, a CPU interrupt request is recognized after the transfer of the current byte or word in the current DMA transfer. The CPU interrupt disables the DMA by clearing the transfer enable bits, TEC2–TEC0. (See DMA Control Register 1 on page 129.) The DMA can increase CPU interrupt latency by up to three cycles in a byte transfer or five cycles in a word transfer.

NOTE: *When DMAP = 0, a CPU interrupt clears the TECx bit if the channel has a pending DMA transfer. Software must re-enable channel x after each CPU interrupt by setting the TECx bit.*

Table 7 shows the effect of the DMAP bit when the DMA has 100% of the bus bandwidth (BB1:BB0 = 1:1).

Table 7. DMA Transfer/CPU Interrupt Request Priority Selection

	DMAP = 0	**DMAP = 1**
Highest Priority	CPU Interrupt Requests	DMA Channel 0 Transfer
	DMA Channel 0 Transfer	DMA Channel 1 Transfer
	DMA Channel 1 Transfer	DMA Channel 2 Transfer
Lowest Priority	DMA Channel 2 Transfer	CPU Interrupt Requests

L2–L0 — Loop Enable Bits

These read/write bits enable looping of the DMA back to the base addresses in the source address and destination address registers during block transfers. Reset clears the L2–L0 bits.

1 = Looping enabled — After transferring the number of bytes equal to the number programmed in the DMA block length register, the DMA:

- Sets the CPU interrupt flag (IFCx) for that channel.
- Generates a CPU interrupt request if enabled (IECx = 1).
- Clears the byte count register.
- Continues the transfer from the base address.

0 = Looping disabled — After transferring the number of bytes equal to the number programmed in the DMA block length register, the DMA:

- Sets the CPU interrupt flag (IFCx) for that channel.
- Generates a CPU interrupt request if enabled (IECx = 1).
- Clears the byte count register.
- Disables the channel by clearing the TECx bit.

NOTE: *The CPU executes a minimum of one cycle before the next DMA loop begins, even if the DMA has 100% of the bus bandwidth.*

DMAWE — DMA Wait Enable Bit

This read/write bit enables the DMA to operate while in wait mode. Reset clears the DMAWE bit.

1 = DMA transfer enabled after WAIT instruction
0 = DMA transfer suspended after WAIT instruction

IFC2–IFC0 — CPU Interrupt Flag Bits

These read/write bits become set when a DMA transfer is complete or at the end of each transfer loop. IFC2, IFC1, or IFC0 can generate a CPU interrupt request if the corresponding IECx bit is set in DMA control register 1. Clear IFC2–IFC0 by reading them and then writing 0s to them. Reset clears the IFC2–IFC0 bits.

1 = DMA transfer complete
0 = DMA transfer not complete

DMA Control Register 2

DMA control register 2 can perform two functions:

- Initiate DMA transfers through software
- Simulate DMA service requests for test purposes

Address: $004E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0
Reset:	0	0	0	0	0	0	0	0

Figure 19. DMA Control Register 2 (DC2)

SWI7–SWI0 — Software Initiate Bits

Each of these read/write bits corresponds to one of the eight DMA transfer sources. (See Table 10 on page 137.) Setting an SWIx bit can initiate a DMA service request from the selected transfer source.

1 = DMA software transfer initiated
0 = DMA software transfer halted or not initiated

Use the following steps to generate a software-initiated DMA service request:

1. Enable a channel to perform a transfer by setting its TECx bit. (See DMA Control Register 1 on page 129.)

2. Assign the channel to a DMA transfer source by writing a binary value from 000 to 111 to its DTS2–DTS0 bits. (See DMA Channel Control Registers on page 135.)
3. Set the SWIx bit that corresponds to the selected transfer source. The bit positions (0–7) of the SWIx bits correspond to the binary values (000–111) that select the DMA transfer source. For example, after selecting transfer source 100 (binary), set bit SWI4 to initiate the DMA service request.

DMA Channel Control Registers

Each DMA channel control register:

- Controls calculation of source and destination addresses.
- Selects transfer of 8-bit bytes or 16-bit words on the channel.
- Assigns the channel to one of eight DMA transfer sources.

The state of the DMA channel control registers after reset is indeterminate.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	SDC3	SDC2	SDC1	SDC0	BWC	DTS2	DTS1	DTS0
Reset:	Indeterminate after Reset							

Figure 20. DMA Channel Control Registers (D0C–D2C)

Table 8. DMA Channel Control Register Address Summary

Register	D0C	D1C	D2C
Address	$0038	$0040	$0048

SDC3–SDC0 — Source/Destination Address Control Bits

These read/write bits control calculation of the source and destination addresses as shown in **Table 9**.

Table 9. Source/Destination Address Register Control

SDC[3:2:1:0]	Source Address	Destination Address
1010	Increment	Increment
1001	Increment	Decrement
1000	Increment	Static
0110	Decrement	Increment
0101	Decrement	Decrement
0100	Decrement	Static
0010	Static	Increment
0001	Static	Decrement
0000	Static	Static

The DMA calculates an incremented address by adding the byte count register to the base address. To calculate a decremented address, the DMA subtracts the byte count register from the base address. To determine a static address, the DMA reads the base address.

BWC — Byte/Word Control Bit

This read/write bit determines whether the DMA channel transfers 8-bit bytes or 16-bit words. The BWC bit has no effect unless either the source or destination address is static or both are static.

1 = 16-bit words

0 = 8-bit bytes

NOTE: *To transfer a block of 16-bit words (BWC = 1), set the block length to the number of words times two. (See DMA Block Length Registers* on page 139.)

When both the source and destination addresses are static, the first byte of the word transfers from the source base address to the destination base address. The second byte transfers from the source base address plus one to the destination address plus one. When either the source or destination address increments or decrements, the DMA transfers bytes from or to incrementing or decrementing addresses.

The CPU interrupt flag (IFCx) becomes set when the byte count register equals the block length register.

DTS2–DTS0 — DMA Transfer Source Bits

These read/write bits assign the DMA channels to the eight transfer source inputs as shown in **Table 10**.

Table 10. DMA Transfer Source Selection

Transfer Source	DTS2:DTS1:DTS0
TIM Channel 0 Interrupt Request	000
TIM Channel 1 Interrupt Request	001
TIM Channel 2 Interrupt Request	010
TIM Channel 3 Interrupt Request	011
SPI Receive Interrupt Request	100
SPI Transmit Interrupt Request	101
SCI Receive Interrupt Request	110
SCI Transmit Interrupt Request	111

DMA Source Address Registers

Each DMA channel takes its data from a source base address contained in a 16-bit source address register. During a block transfer, the DMA determines successive source addresses by adding to (to increment) or subtracting from (to decrement) the base address. In static address transfers, the DMA finds the source address by merely reading the source address registers. **Figure 21** shows the DMA source address registers. The state of the source address registers after reset is indeterminate.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
Reset:	Indeterminate after Reset							

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
Reset:	Indeterminate after Reset							

Figure 21. DMA Source Address Registers (D0SH/L–D2SH/L)

Table 11. DMA Source Address Register Address Summary

Register	D0SH	D0SL	D1SH	D1SL	D2SH	D2SL
Address	$0034	$0035	$003C	$003D	$0044	$0045

DMA Destination Address Registers

Each DMA channel transfers data to the destination base address contained in a 16-bit destination address register. During a block transfer, the DMA determines successive destination addresses by adding to (to increment) or subtracting from (to decrement) the base address. In static address transfers, the DMA finds the destination address by merely reading the destination address registers. **Figure 22** shows the DMA destination address registers. The state of the destination address registers after reset is indeterminate.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
Reset:	Indeterminate after Reset							

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
Reset:	Indeterminate after Reset							

Figure 22. DMA Destination Address Registers (D0DH/L–D2DH/L)

Table 12. Destination Address Register Address Summary

Register	D0DH	D0DL	D1DH	D1DL	D2DH	D2DL
Address	$0036	$0037	$003E	$003F	$0046	$0047

DMA Block Length Registers

The read/write block length registers control the number of bytes transferred. During a block transfer, the DMA compares the number programmed into the channel’s DMA block length register to the number in its DMA byte count register. When the byte count reaches the value in the block length register, the DMA:

- Sets the CPU interrupt flag (IFCx) for that channel in the DMA status and control register.
- Generates a CPU interrupt request if enabled.
- Resets the byte count register.

If looping is disabled (Lx bit in DMA status and control register = 0), the DMA then stops the transfer by clearing the TECx bit in DMA control register 1, disabling the channel. If looping is enabled (Lx bit = 1), the DMA continues the transfer from the base address.

The block length of a word transfer is twice the number of words. The state of the DMA block length registers after reset is indeterminate.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0
Reset:	Indeterminate after Reset							

Figure 23. DMA Block Length Registers (D0BL–D2BL)

Table 13. DMA Block Length Register Address Summary

Register	D0BL	D1BL	D2BL
Address	$0039	$0041	$0049

DMA Byte Count Registers

Each read/write DMA byte count register contains the number of bytes transferred on that channel in the current DMA transfer.

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BC7	BC6	BC5	BC4	BC3	BC2	BC1	BC0
Reset:	0	0	0	0	0	0	0	0

Figure 24. DMA Byte Count Registers (D0BC–D2BC)

Table 14. DMA Byte Count Register Address Summary

Register	D0BC	D1BC	D2BC
Address	$003B	$0043	$004B

Writing to the channel x source address or destination address register clears the channel x byte count register. The channel x byte count register also is cleared when its count reaches the value in the channel x block length register. Reset clears the byte count registers.

Break Module (BRK)

Contents

Introduction

The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

Features

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

Functional Description

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

The following events can cause a break interrupt to occur:

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

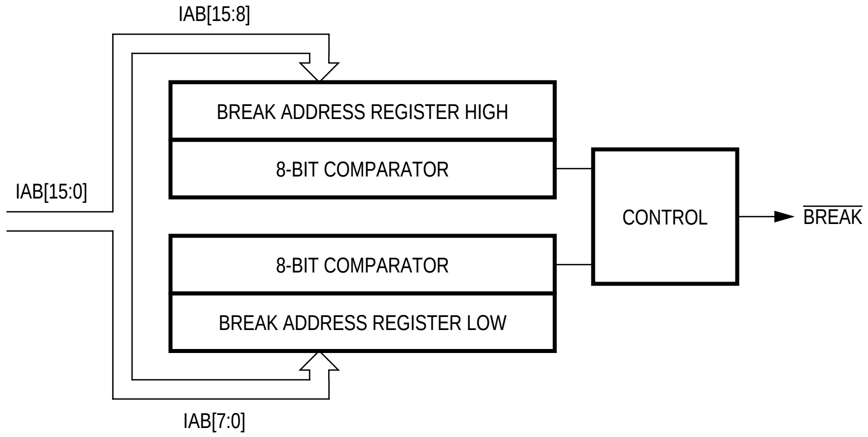


Figure 1. Break Module Block Diagram

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
Break Status Register (BSR)	Read:	0	0	0	1	0	0	BW	0
	Write:	R	R	R	R	R	R	0	R
	Reset:	0	0	0	1	0	0	0	0
Break Flag Control Register (BFCR)	Read: Write:	BFCE	R	R	R	R	R	R	R
	Reset:	0							
Break Address Register High (BRKH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	0	0	0	0	0	0	0	0
Break Address Register Low (BRKL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	0	0	0	0	0	0	0	0
Break Status and Control Register (BSCR)	Read:	BRKE	BRKA	0	0	0	0	0	0
	Write:			▨	▨	▨	▨	▨	▨
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 2. I/O Register Summary

Table 1. I/O Register Address Summary

Register	BSR	BFCR	BRKH	BRKL	BSCR
Address	$FE00	$FE03	$FE0C	$FE0D	$FE0E

Flag Protection During Break Interrupts

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

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.

DMA During Break Interrupts

If the DMA is enabled, clear the DMAP bit in the DMA status and control register before executing a break interrupt.

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

TIM During Break Interrupts

A break interrupt stops the timer counter.

COP During Break Interrupts

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

Low-Power Modes

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

Wait Mode

If enabled, the break module is active in wait mode. A DMA-generated address that matches the break address registers in wait mode sets the BW in the break status register.

The DMA can also use the break status and control register as its destination address in order to write to the BRKA and BRKE bits during wait mode. A DMA write to the break status and control register sets the BW bit.

Stop Mode

The break module is inactive in stop mode. The STOP instruction does not affect break module register states.

Break Module Registers

These registers control and monitor operation of the break module:

- Break status and control register (BSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)
- Break status register (BSR)
- Break flag control register (BFCR)

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:			▨	▨	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 3. Break Status and Control Register (BSCR)

BRKE — Break Enable Bit

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

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

BRKA — Break Active Bit

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

1 = (When read) Break address match
0 = (When read) No break address match

Break Address Registers

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

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

Figure 4. Break Address Registers (BRKH and BRKL)

Break Status Register

The break status register contains a flag to indicate that a break caused an exit from wait mode. The flag is useful in applications requiring a return to wait mode after exiting from a break interrupt.

Address:	$FE00							
	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	1	0	0	BW	0
Write:	R	R	R	R	R	R	NOTE	R
Reset:	0	0	0	1	0	0	0	0

R = Reserved NOTE: Writing a logic 0 clears BW.

Figure 5. Break Status Register (BSR)

BW — Break Wait Bit

This read/write bit is set when a break interrupt causes an exit from wait mode. Clear BW by writing a logic 0 to it. Reset clears BW.

1 = Break interrupt during wait mode
0 = No break interrupt during wait mode

BW is for applications that require a return to wait mode after exiting wait mode for a DMA-generated break interrupt. BW can be read within the break interrupt routine. The user can modify the return address on the stack by subtracting 1 from it. The following code is an example.

```
; This code works if the H register was stacked in the break
; interrupt routine. Execute this code at the end of the break
; interrupt routine.
  HIBYTE EQU   5
  LOBYTE EQU   6
;            If not BW, do RTI
             BRCLR BW,BSR, RETURN    ; See if wait mode or stop mode
                                     ; was exited by break.
             TST   LOBYTE,SP         ; If RETURNLO is not 0,
             BNE   DOLO              ; then just decrement low byte.
             DEC   HIBYTE,SP         ; Else deal with high byte also.
  DOLO       DEC   LOBYTE,SP         ; Point to WAIT/STOP opcode.
  RETURN     PULH                    ; Restore H register.
             RTI
```

Break Flag Control Register

The break flag 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 6. Break Flag Control Register (BFCR)

BCFE — Break Clear Flag Enable Bit

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

1 = Status bits clearable during break

0 = Status bits not clearable during break

Monitor ROM (MON)

Contents

Introduction

Execution of code in the monitor ROM in monitor mode allows complete testing of the MCU through a single-wire interface with a host computer.

Features

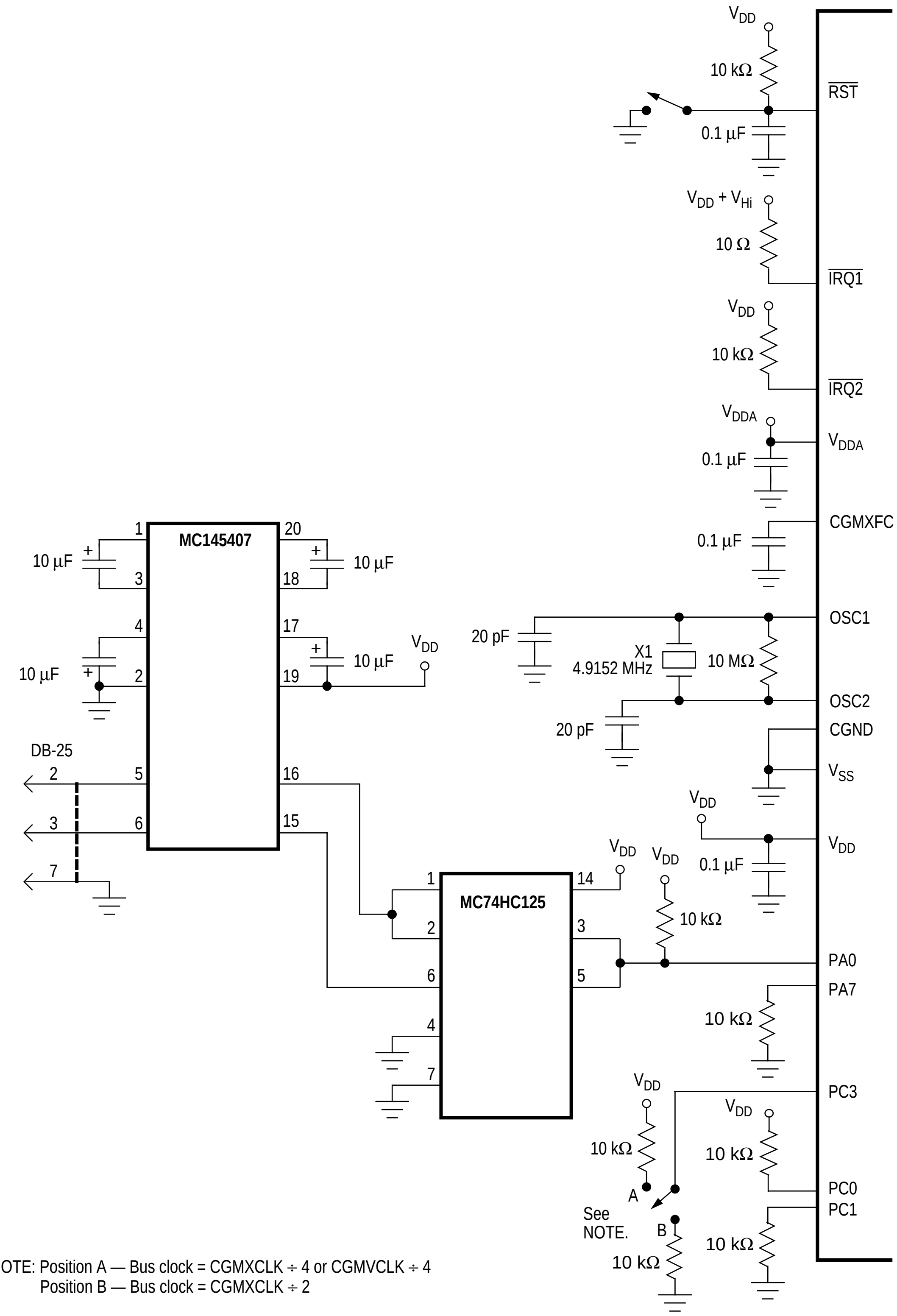
Features of monitor mode include the following:

- Normal User-Mode Pin Functionality
- One Pin Dedicated to Serial Communication between Monitor ROM and Host Computer
- Standard Mark/Space Non-Return-to-Zero (NRZ) Communication with Host Computer
- Execution of Code in Either RAM or ROM
- ROM Security

Functional Description

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

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute the code in RAM loaded by the host while all MCU pins retain normal operating mode functions. All communication between the host and the MCU is through the PA0 pin at a chosen baud rate. A level-shifting and multiplexing interface is required between PA0 and the host. PA0 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 1. Monitor Mode Circuit

Entering Monitor Mode

Table 1 shows the pin conditions for entering monitor mode.

Table 1. Monitor Mode Entry

$\overline{\text{IRQ}}$ Pin	PA7 Pin	PC0 Pin	PC1 Pin	PA0 Pin	PC3 Pin	Bus Clock Frequency
$V_{DD} + V_{Hi}$	0	1	0	1	1	$\frac{\text{CGMXCLK}}{4}$ or $\frac{\text{CGMVCLK}}{4}$
					0	$\frac{\text{CGMXCLK}}{2}$

If the PC3 pin is low upon monitor mode entry, the bus frequency is equal to the frequency of CGMXCLK divided by two. CGMXCLK is a buffered version of the clock on the OSC1 pin. If PC3 is high upon monitor mode entry, the bus frequency is equal to the frequency of CGMXCLK divided by four. The PLL can be engaged after monitor mode entry to multiply the bus frequency by programming the CGM. For information on how to program the PLL, see Clock Generator Module (CGM) on page 81. To use the PLL, PC3 must be high during monitor mode entry. With the PLL engaged, the bus frequency is equal to the PLL output, CGMVCLK, divided by four.

NOTE: *If CGMXCLK divided by two is selected as the bus frequency (PC3 = 0), the OSC1 signal must have a 50% duty cycle at maximum bus frequency.*

Enter monitor mode with one of the pin configurations shown in **Table 1** by pulling $\overline{\text{RST}}$ low and then high. The rising edge of $\overline{\text{RST}}$ latches monitor mode. Once monitor mode is latched, the values on the PC0, PC1, PA0, and PC3 pins can be changed.

NOTE: *The PA7 pin must remain at logic 0 for 24 bus cycles after the $\overline{\text{RST}}$ pin goes high.*

Once out of reset, the MCU waits for the host to send eight security bytes. (See Security on page 161.) After the receiving security bytes, the MCU sends a break character (10 consecutive logic 0s) to the host, indicating that it is ready to receive a command.

In monitor mode, the MCU uses different vectors for reset, SWI, and break interrupt than those for user mode. The alternate vectors are in the $FE page instead of the $FF page and allow code execution from the internal monitor firmware instead of user code.

The COP module is disabled in monitor mode as long as V_{DD} + V_{Hi} is applied to either the $\overline{\text{IRQ}}$ pin or the $\overline{\text{RST}}$ pin.

Table 2 summarizes the differences between user mode and monitor mode.

Table 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 $\overline{\text{IRQ}}$ pin or the $\overline{\text{RST}}$ pin, the COP is enabled. The COP is a mask option.

Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. Transmit and receive baud rates must be identical.

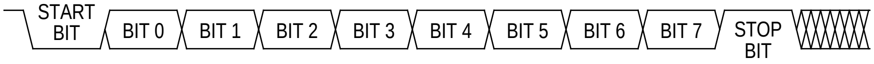


Figure 2. Monitor Data Format

Break Character

A start bit (logic 0) followed by nine logic 0 bits is a break character. When the monitor receives a break character, it drives the PA0 pin high for the duration of two bits and then echoes back the break character.

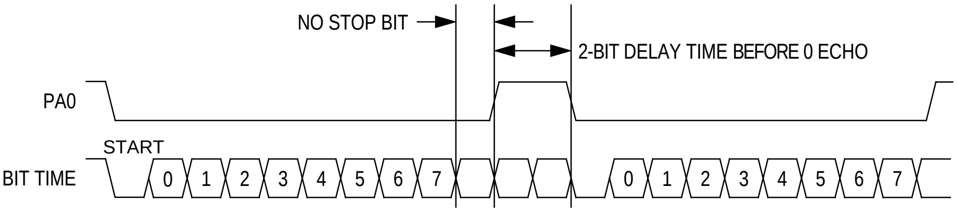


Figure 3. Break Transaction

Baud Rate

The bus clock frequency of the MCU in monitor mode is determined by:

- The external clock frequency
- The value on the PC3 pin
- Whether the phase-locked loop (PLL) is engaged

The internal monitor firmware performs a division by 256 (for sampling data); therefore, the bus frequency divided by 256 is the baud rate of the monitor mode data transfer.

For example, with a 4.9152-MHz external clock and the PC3 pin at logic 1 during reset, data is transferred between the monitor and host at 4800 baud. If the PC3 pin is at logic 0 during reset, the monitor baud rate is 9600.

The PLL can be engaged to increase the baud rate of instruction transfer between the host and the MCU and to increase the speed of program execution. Monitor mode must be entered with PTC high to use the PLL. See Entering Monitor Mode on page 152. Initially, the bus frequency is a divide-by-four of the input clock.

After the PLL is programmed and selected as the base clock, communication between the host and MCU must be re-established at the new baud rate. One way to accomplish this is with a program downloaded from the host into the MCU RAM. The downloaded routine can program the PLL and send a new baud rate flag to the host just before engaging the PLL onto the bus. Then an SWI instruction can be used to return program control to the monitor firmware.

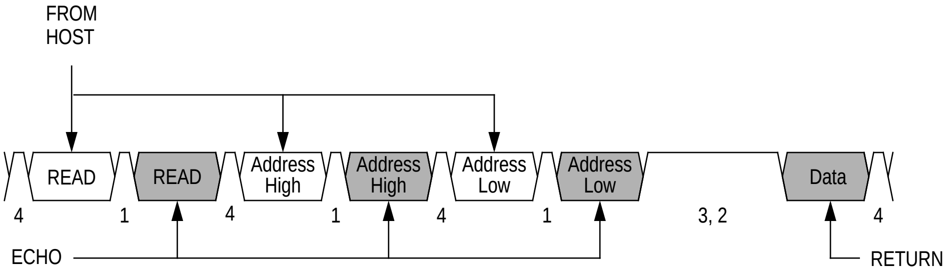
Commands

The monitor ROM firmware uses the following commands:

- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer + 1)
- RUN (run user program)

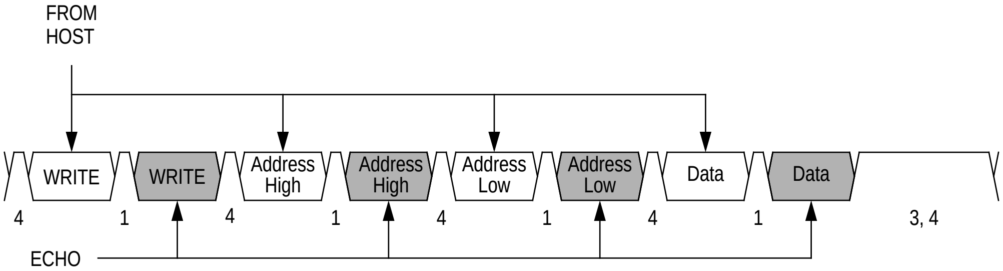
The monitor ROM firmware echoes each received byte back to the PA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

NOTE: *Wait one bit time after each echo before sending the next byte.*



NOTE: 1 = Echo delay (2 bit times)
2 = Data return delay (2 bit times)
3 = Cancel command delay (11 bit times)
4 = Wait 1 bit time before sending next byte.

Figure 4. Read Transaction



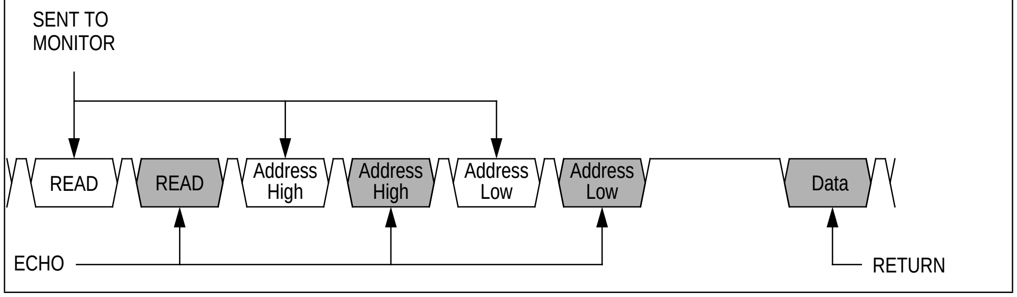
NOTE: 1 = Echo delay (2 bit times)
3 = Cancel command delay (11 bit times)
4 = Wait 1 bit time before sending next byte.

Figure 5. Write Transaction

A brief description of each monitor mode command follows:

Table 3. READ (Read Memory) Command

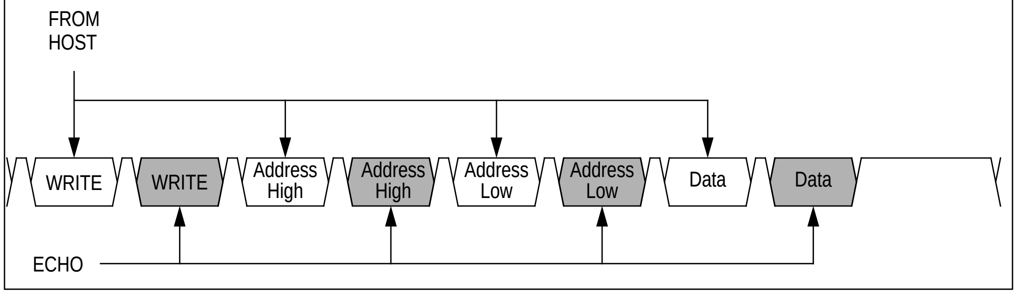
<table>
<tr><td><b>Description</b></td><td>Read byte from memory</td></tr>
<tr><td><b>Operand</b></td><td>2-byte address in high byte:low byte order</td></tr>
<tr><td><b>Data Returned</b></td><td>Returns contents of specified address</td></tr>
<tr><td><b>Opcode</b></td><td>$4A</td></tr>
<tr><td colspan="2"><b>Command Sequence</b>



</td></tr>
</table>

Table 4. WRITE (Write Memory) Command

<table>
<tr><td><b>Description</b></td><td>Write byte to memory</td></tr>
<tr><td><b>Operand</b></td><td>2-byte address in high byte:low byte order; low byte followed by data byte</td></tr>
<tr><td><b>Data Returned</b></td><td>None</td></tr>
<tr><td><b>Opcode</b></td><td>$49</td></tr>
<tr><td colspan="2"><b>Command Sequence</b>



</td></tr>
</table>

Table 5. IREAD (Indexed Read) Command

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

Command Sequence

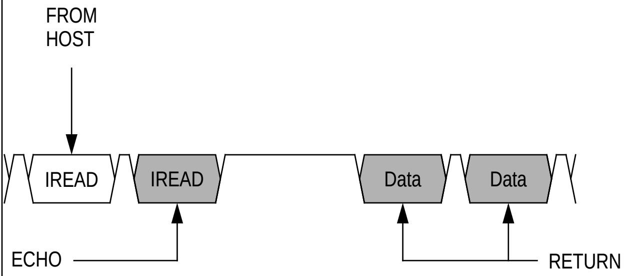
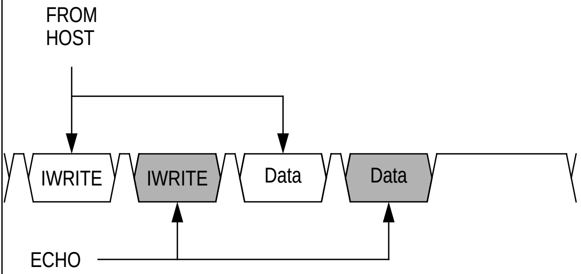


Table 6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Single data byte
Data Returned	None
Opcode	$19

Command Sequence



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

Table 7. READSP (Read Stack Pointer) Command

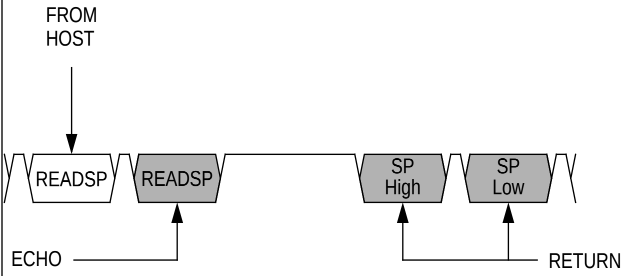
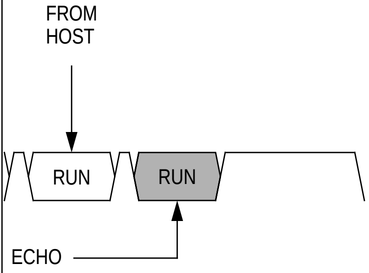
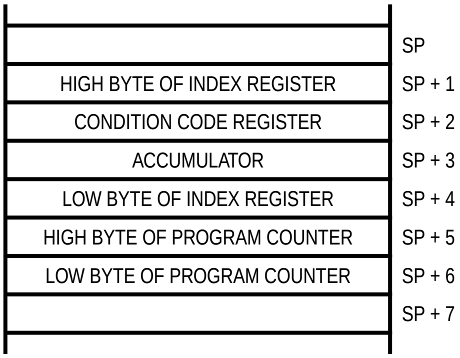
Description	Reads stack pointer
Operand	None
Data Returned	Returns incremented stack pointer value (SP + 1) in high byte:low byte order.
Opcode	$0C
Command Sequence	

Table 8. RUN (Run User Program) Command

Description	Executes PULH and RTI instructions
Operand	None
Data Returned	None
Opcode	$28
Command Sequence	

The MCU executes the SWI and PSHH instructions when it enters monitor mode. The RUN command tells the MCU to execute the PULH and RTI instructions. Before sending the RUN command, the host can modify the stacked CPU registers to prepare to run the host program. The READSP command returns the incremented stack pointer value, SP + 1. The high and low bytes of the program counter are at addresses SP + 5 and SP + 6.



	SP
HIGH BYTE OF INDEX REGISTER	SP + 1
CONDITION CODE REGISTER	SP + 2
ACCUMULATOR	SP + 3
LOW BYTE OF INDEX REGISTER	SP + 4
HIGH BYTE OF PROGRAM COUNTER	SP + 5
LOW BYTE OF PROGRAM COUNTER	SP + 6
	SP + 7

Figure 6. Stack Pointer at Monitor Mode Entry

Security

A security feature[1] discourages unauthorized reading of ROM locations while in monitor mode. The host can bypass the security feature at monitor mode entry by sending eight security bytes that match the bytes at locations $FFF6–$FFFD. Locations $FFF6–$FFFD contain user-defined data.

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

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PA0.

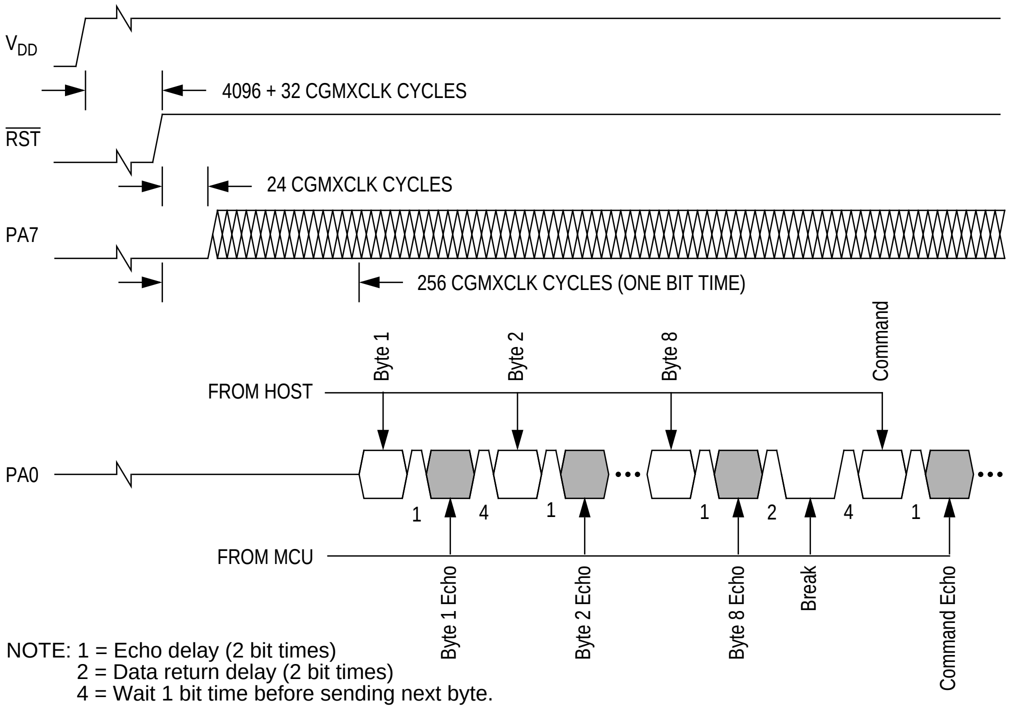


Figure 7. Monitor Mode Entry Timing

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

If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all ROM locations and execute code from ROM. Security remains bypassed until a power-on reset occurs. After the host bypasses security, any reset other than a power-on reset requires the host to send another eight bytes. If the reset was not a power-on reset, security remains bypassed regardless of the data that the host sends.

If the received bytes do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading ROM locations returns undefined data, and trying to execute code from ROM causes an illegal address reset. After the host fails to bypass security, any reset other than a power-on reset causes an endless loop of illegal address resets.

After receiving the eight security bytes from the host, the MCU transmits a break character signalling that it is ready to receive a command.

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

Timer Interface Module (TIM)

Contents

Introduction

The TIM is a 4-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. **Figure 1** is a block diagram of the TIM.

Features

Features of the TIM include the following:

- Four Input Capture/Output Compare Channels
  - Rising-Edge, Falling-Edge, or Any-Edge Input Capture Trigger
  - Set, Clear, or Toggle Output Compare Action
- Buffered and Unbuffered Pulse Width Modulation (PWM) Signal Generation
- Programmable TIM Clock Input
  - 7-Frequency Internal Bus Clock Prescaler Selection
  - External TIM Clock Input (4-MHz Maximum Frequency)
- Free-Running or Modulo Up-Count Operation
- Toggle Any Channel Pin on Overflow
- TIM Counter Stop and Reset Bits
- DMA Service Request Generation
- Modular Architecture Expandable to Eight Channels

Pin Name Conventions

The generic names of the TIM I/O pins are:

- TCLK (TIM external clock input pin)
- TCH0 (TIM channel 0 I/O pin)
- TCH1 (TIM channel 1 I/O pin)
- TCH2 (TIM channel 2 I/O pin)
- TCH3 (TIM channel 3 I/O pin)

TIM pins are shared by parallel I/O ports. The full name of a TIM pin reflects the name of the shared port pin. The generic pin names appear in the text that follows. **Table 1** shows the full names of the TIM I/O pins.

Table 1. Pin Name Conventions

Generic Pin Names	TCLK	TCH0	TCH1	TCH2	TCH3
Full Pin Names	PE3/TCLK	PE4/TCH0	PE5/TCH1	PE6/TCH2	PE7/TCH3

Functional Description

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

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

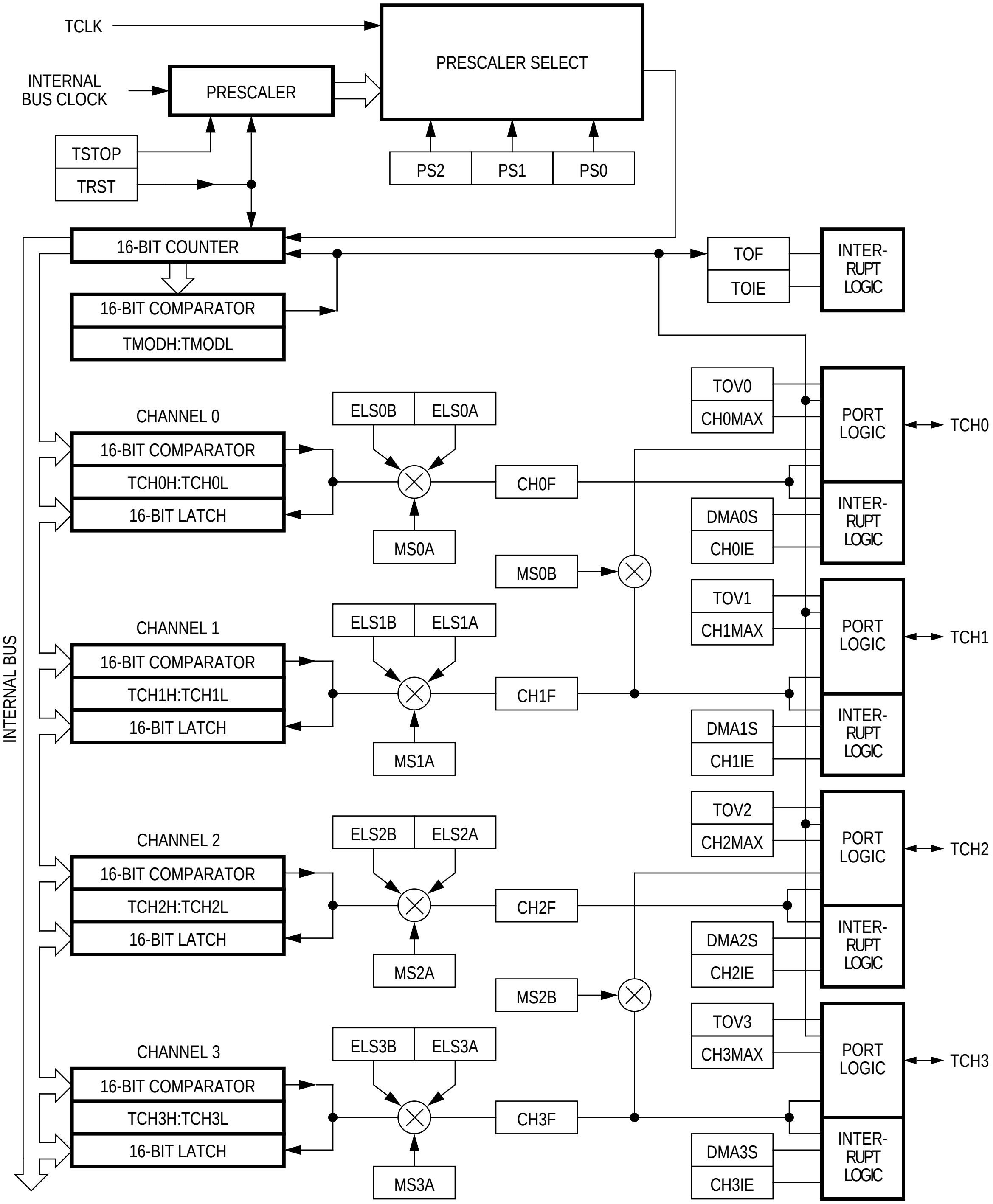


Figure 1. TIM Block Diagram

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
TIM Status and Control Register (TSC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
	Write:	0			TRST	▨			
	Reset:	0	0	1	0	0	0	0	0
TIM DMA Select Register (TDMA)	Read:	0	0	0	0	DMA3S	DMA2S	DMA1S	DMA0S
	Write:	▨	▨	▨	▨				
	Reset:	0	0	0	0	0	0	0	0
TIM Counter Register High (TCNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	0	0	0	0	0	0	0	0
TIM Counter Register Low (TCNTL	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	0	0	0	0	0	0	0	0
TIM Counter Modulo Register High (TMODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	1	1	1	1	1	1	1	1
TIM Counter Modulo Register Low (TMODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:								
	Reset:	1	1	1	1	1	1	1	1
TIM Channel 0 Status and Control Register (TSC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TIM Channel 0 Register High (TCH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	Indeterminate after Reset							
TIM Channel 0 Register Low (TCH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:								
	Reset:	Indeterminate after Reset							
TIM Channel 1 Status and Control Register (TSC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
	Write:	0		▨					
	Reset:	0	0	0	0	0	0	0	0
TIM Channel 1 Register High (TCH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	Indeterminate after Reset							
TIM Channel 1 Register Low (TCH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
	Write:								
	Reset:	Indeterminate after Reset							
TIM Channel 2 Status and Control Register (TSC2)	Read:	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	TOV2	CH2MAX
	Write:	0							
	Reset:	0	0	0	0	0	0	0	0
TIM Channel 2 Register High (TCH2H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
	Write:								
	Reset:	Indeterminate after Reset							

▨ = Unimplemented

Figure 2. I/O Register Summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
TIM Channel 2 Register Low (TCH2L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after Reset							
TIM Channel 3 Status and Control Register (TSC3)	Read:	CH3F	CH3IE	0	MS3A	ELS3B	ELS3A	TOV3	CH3MAX
	Write:	0		▨					
	Reset:	0	0	0	0	0	0	0	0
TIM Channel 3 Register High (TCH3H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
	Reset:	Indeterminate after Reset							
TIM Channel 3 Register Low (TCH3L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
	Reset:	Indeterminate after Reset							

▨ = Unimplemented

Figure 2. I/O Register Summary (Continued)

Table 2. I/O Register Address Summary

Register	TSC	TDMA	TCNTH	TCNTL	TMODH	TMODL	TSC0	TCH0H	TCH0L	TSC1
Address	$0020	$0021	$0022	$0023	$0024	$0025	$0026	$0027	$0028	$0029

Register	TCH1H	TCH1L	TSC2	TCH2H	TCH2L	TSC3	TCH3H	TCH3L
Address	$002A	$002B	$002C	$002D	$002E	$002F	$0030	$0031

TIM Counter Prescaler

The TIM clock source can be one of the seven prescaler outputs or the TIM clock pin, TCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS2–PS0, in the TIM status and control register select the TIM clock source.

Input Capture

With the input capture function, the TIM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TIM latches the contents of the TIM counter into the TIM channel registers, TCHxH:TCHxL. The polarity of the active edge is programmable. Input capture latency can be up to three bus clock cycles. Input captures can generate TIM CPU interrupt requests or TIM DMA service requests.

Output Compare

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

Unbuffered Output Compare

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

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

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

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

Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the TCH0 pin. The TIM channel registers of the linked pair alternately control the output.

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

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

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

Pulse Width Modulation

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

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

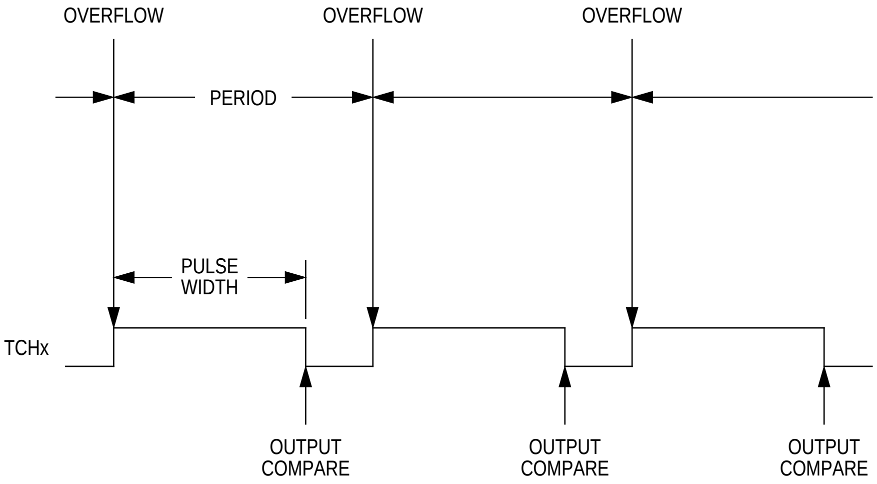


Figure 3. PWM Period and Pulse Width

The value in the TIM counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000. See **TIM Status and Control Register** on page 178.

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

Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in **Pulse Width Modulation** on page 171. The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

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

Use the following methods to synchronize unbuffered changes in the PWM pulse width on channel x:

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

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

Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the TCH0 pin. The TIM channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIM channel 0 status and control register (TSC0) links channel 0 and channel 1. The TIM channel 0 registers initially control the pulse width on the TCH0 pin. Writing to the TIM channel 1 registers enables the TIM channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (0 or 1) that control the pulse width are the ones written to last. TSC0 controls and monitors the buffered PWM function, and TIM channel 1 status and control register (TSC1) is unused. While the MS0B bit is set, the channel 1 pin, TCH1, is available as a general-purpose I/O pin.

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

Setting the MS2B bit in TIM channel 2 status and control register (TSC2) links channel 2 and channel 3. The TIM channel 2 registers initially control the pulse width on the TCH2 pin. Writing to the TIM channel 3 registers enables the TIM channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIM channel registers (2 or 3) that control the pulse width are the ones written to last. TSC2 controls and monitors the buffered PWM function, and TIM channel 3 status and control register (TSC3) is unused. While the MS2B bit is set, the channel 3 pin, TCH3, is available as a general-purpose I/O pin.

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

PWM Initialization

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

1. In the TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. See Table 4 on page 188.
   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 4 on page 188.

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

5. In the TIM status control register (TSC), clear the TIM stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIM channel 0 registers (TCH0H:TCH0L) initially control the buffered PWM output. TIM status control register 0 (TSCR0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIM channel 2 registers (TCH2H:TCH2L) initially control the PWM output. TIM status control register 2 (TSCR2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Clearing the toggle-on-overflow bit, TOVx, inhibits output toggles on TIM overflows. Subsequent output compares try to force the output to a state it is already in and have no effect. The result is a 0% duty cycle output.

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100% duty cycle output. See TIM Channel Status and Control Registers on page 183.

Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter value rolls over to $0000 after matching the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH3F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests and TIM DMA service requests are controlled by the channel x interrupt enable bit, CHxIE, and the channel x DMA select bit, DMAxS. Channel x TIM CPU interrupt requests are enabled when CHxIE:DMAxS = 1:0. Channel x TIM DMA service requests are enabled when CHxIE:DMAxS = 1:1. CHxF and CHxIE are in the TIM channel x status and control register. DMAxS is in the TIM DMA select register.

Low-Power Modes

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

Wait Mode

The TIM remains active in wait mode. Any enabled CPU interrupt request from the TIM can bring the MCU out of wait mode.

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

The DMA can service the TIM without exiting wait mode.

Stop Mode

The TIM is inactive in stop mode. The STOP instruction does not affect register states or the state of the TIM counter. TIM operation resumes when the MCU exits stop mode after an external interrupt.

TIM During Break Interrupts

A break interrupt stops the TIM counter.

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See **Break Module (BRK)** on page 141.

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.

I/O Signals

Port E shares five of its pins with the TIM. TCLK is an external clock input to the TIM prescaler. The four TIM channel I/O pins are TCH0, TCH1, TCH2, and TCH3.

TIM Clock Pin (TCLK)

TCLK is an external clock input that can be the clock source for the TIM counter instead of the prescaled internal bus clock. Select the TCLK input by writing logic 1s to the three prescaler select bits, PS[2:0]. See TIM Status and Control Register on page 178. The minimum TCLK pulse width, $TCLK_{lmin}$ or $TCLK_{hmin}$, is:

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

The maximum TCLK frequency is:

bus frequency ÷ 2

TCLK is available as a general-purpose I/O pin when not used as the TIM clock input. When the TCLK pin is the TIM clock input, it is an input regardless of the state of the DDRE3 bit in data direction register E.

TIM Channel I/O Pins (TCH0–TCH3)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. TCH0 and TCH2 can be configured as buffered output compare or buffered PWM pins.

I/O Registers

These I/O registers control and monitor operation of the TIM:

- TIM status and control register (TSC)
- TIM DMA select register (TDMA)
- TIM control registers (TCNTH:TCNTL)
- TIM counter modulo registers (TMODH:TMODL)
- TIM channel status and control registers (TSC0, TSC1, TSC2, and TSC3)
- TIM channel registers (TCH0H:TCH0L, TCH1H:TCH1L, TCH2H:TCH2L, and TCH3H:TCH3L)

TIM Status and Control Register

The TIM status and control register:

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

Address: $0020

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

▨ = Unimplemented

Figure 4. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

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

TOIE — TIM Overflow Interrupt Enable Bit

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

1 = TIM overflow interrupts enabled
0 = TIM overflow interrupts disabled

TSTOP — TIM Stop Bit

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

1 = TIM counter stopped
0 = TIM counter active

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

TRST — TIM Reset Bit

Setting this write-only bit resets the TIM counter and the TIM prescaler. Setting TRST has no effect on any other registers. Counting resumes from $0000. TRST is cleared automatically after the TIM counter is reset and always reads as logic 0. Reset clears the TRST bit.

1 = Prescaler and TIM counter cleared
0 = No effect

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

PS2–PS0 — Prescaler Select Bits

These read/write bits select either the TCLK pin or one of the seven prescaler outputs as the input to the TIM counter as **Table 3** shows. Reset clears the PS2–PS0 bits.

Table 3. Prescaler Selection

PS[2:0]	TIM Clock Source
000	Internal Bus Clock
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	TCLK

TIM DMA Select Register

The TIM DMA select register enables either TIM CPU interrupt requests or TIM DMA service requests.

Address: $0021

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	DMA3S	DMA2S	DMA1S	DMA0S
Write:	▨	▨	▨	▨				
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 5. TIM DMA Select Register (TDMA)

DMA3S — DMA Channel 3 Select Bit

This read/write bit enables TIM DMA service requests on channel 3. Reset clears the DMA3S bit.

1 = TIM DMA service requests enabled on channel 3
TIM CPU interrupt requests disabled on channel 3
0 = TIM DMA service requests disabled on channel 3
TIM CPU interrupt requests enabled on channel 3

DMA2S — DMA Channel 2 Select Bit

This read/write bit enables TIM DMA service requests on channel 2. Reset clears the DMA2S bit.

1 = TIM DMA service requests enabled on channel 2
TIM CPU interrupt requests disabled on channel 2
0 = TIM DMA service requests disabled on channel 2
TIM CPU interrupt requests enabled on channel 2

DMA1S — DMA Channel 1 Select Bit

This read/write bit enables TIM DMA service requests on channel 1. Reset clears the DMA1S bit.

1 = TIM DMA service requests enabled on channel 1
TIM CPU interrupt requests disabled on channel 1
0 = TIM DMA service requests disabled on channel 1
TIM CPU interrupt requests enabled on channel 1

DMA0S — DMA Channel 0 Select Bit

This read/write bit enables TIM DMA service requests on channel 0. Reset clears the DMA0S bit.

1 = TIM DMA service requests enabled on channel 0
TIM CPU interrupt requests disabled on channel 0
0 = TIM DMA service requests disabled on channel 0
TIM CPU interrupt requests enabled on channel 0

TIM Counter Registers

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

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

Address: $0022:$0023

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

▨ = Unimplemented

Figure 6. TIM Counter Registers (TCNTH and TCNTL)

TIM Counter Modulo Registers

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

Address: $0024:$0025

	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
	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 7. TIM Counter Modulo Registers (TMODH and TMODL)

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

TIM Channel Status and Control Registers

Each of the TIM channel status and control registers:

- Flags input captures and output compares.
- Enables input capture and output compare interrupts.
- Selects input capture, output compare, or PWM operation.
- Selects high, low, or toggling output on output compare.
- Selects rising, falling, or any edge as the input capture trigger.
- Selects output toggling on TIM overflow.
- Selects 100% PWM duty cycle.
- Selects buffered or unbuffered output compare/PWM operation.

TIM Channel 0 Status and Control Register

Address: $0026

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

Figure 8. TIM Channel 0 Status and Control Register (TSC0)

TIM Channel 1 Status and Control Register

Address: $0029

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

▨ = Unimplemented

Figure 9. TIM Channel 1 Status and Control Register (TSC1)

TIM Channel 2 Status and Control Register

Address: $002C

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

Figure 10. TIM Channel 2 Status and Control Register (TSC2)

TIM Channel 3 Status and Control Register

Address: $002F

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

▨ = Unimplemented

Figure 11. TIM Channel 3 Status and Control Register (TSC3)

CHxF— Channel x Flag Bit

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

When TIM CPU interrupt requests are enabled (CHxIE:DMAxS = 1:0), clear CHxF by reading TIM 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.

When TIM DMA service requests are enabled (CHxIE:DMAxS = 1:1), clear CHxF by reading or writing to the low byte of the TIM channel x registers (TCHxL).

Reset clears the CHxF bit. Writing a logic 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

NOTE: *Reading the high byte of the timer channel x registers (TCHxH) inhibits the CHxF bit until the low byte (TCHxL) is read.*

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM CPU interrupts and TIM DMA service requests on channel x. The DMAxS bit in the TIM DMA select register selects channel x TIM DMA service requests or TIM CPU interrupt requests.

NOTE: *TIM DMA service requests cannot be used in buffered PWM mode. In buffered PWM mode, disable TIM DMA service requests by clearing the DMAxS bit in the TIM DMA select register.*

Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests and DMA service requests enabled

0 = Channel x CPU interrupt requests and DMA service requests disabled

MSxB — Mode Select Bit B

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

Setting MS0B disables the channel 1 status and control register and reverts TCH1 to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TCH3 to general-purpose I/O.

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 4.

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 4. Reset clears the MSxA bit.

1 = Initial output level low
0 = Initial output level high

NOTE: *Before changing a channel function by writing to the MSxB or MSxA bit, set the TSTOP and TRST bits in the TIM status and control register (TSC).*

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port E, and pin TCHx is available as a general-purpose I/O pin. Table 4 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 4. Mode, Edge, and Level Selection

MSx[B:A]	ELSx[B:A]	Mode	Configuration
X0	00	Output Preset	Pin under Port Control; Initial Output Level High
X1	00		Pin under Port Control; Initial Output Level Low
00	01	Input Capture	Capture on Rising Edge Only
00	10		Capture on Falling Edge Only
00	11		Capture on Rising or Falling Edge
01	01	Output Compare or PWM	Toggle Output on Compare
01	10		Clear Output on Compare
01	11		Set Output on Compare
1X	01	Buffered Output Compare or Buffered PWM	Toggle Output on Compare
1X	10		Clear Output on Compare
1X	11		Set Output on Compare

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

TOVx — Toggle-On-Overflow Bit

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

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

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

NOTE: *Reading the high byte of the timer channel x registers (TCHxH) prevents the channel x pin from toggling until the low byte (TCHxL) is read.*

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%. As **Figure 12** shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

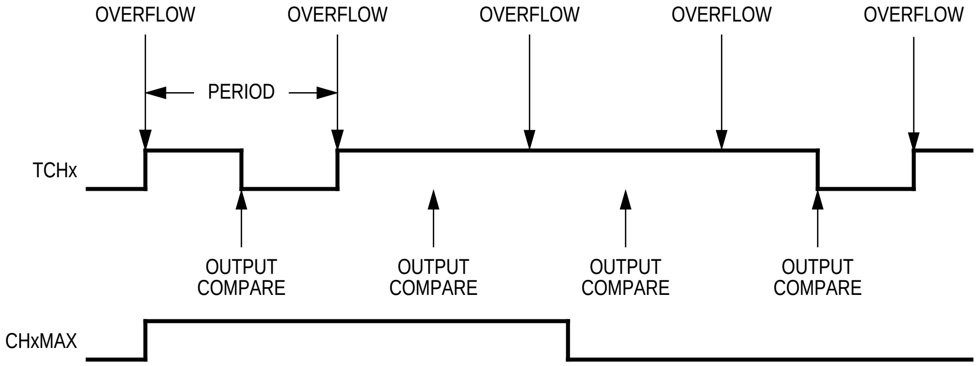


Figure 12. CHxMAX Latency

TIM Channel Registers

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

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

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

Figure 13. TIM Channel Registers (TCH0H/L and TCH3H/L)

Table 5. TIM Channel Register Address Summary

Register	TCH0H	TCH0L	TCH1H	TCH1L	TCH2H	TCH2L	TCH3H	TCH3L
Address	$0027	$0028	$002A	$002B	$002D	$002E	$0030	$0031

In input capture mode, reading TCHxH prevents the input capture value from latching into the channel registers and inhibits the CHxF bit until TCHxL is read.

In output compare mode, writing to TCHxH prevents the channel x pin from toggling and inhibits the CHxF bit until TCHxL is written.

Serial Peripheral Interface Module (SPI)

Contents

The SPI allows full-duplex, synchronous, serial communications with peripheral devices.

Features

Features of the SPI module include the following:

- Full-Duplex Operation
- Master and Slave Modes
- Double-Buffered Operation with Separate Transmit and Receive Registers
- Four Master Mode Frequencies (Maximum = Bus Frequency ÷ 2)
- Maximum Slave Mode Frequency = Bus Frequency
- Clock Ground for Reduced Radio Frequency (RF) Interference
- Serial Clock with Programmable Polarity and Phase
- Two Separately Enabled Interrupts with DMA or CPU Service:
  - SPRF (SPI Receiver Full)
  - SPTE (SPI Transmitter Empty)
- Mode Fault Error Flag with CPU Interrupt Capability
- Overflow Error Flag with CPU Interrupt Capability
- Programmable Wired-OR Mode
- Limited I^{2}C (Inter-Integrated Circuit) Compatibility

Pin Name Conventions

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

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

SPI pins are shared by parallel I/O ports or have alternate functions. The full name of an SPI pin reflects the name of the shared port pin or the name of an alternate pin function. The generic pin names appear in the text that follows. **Table 1** shows the full names of the SPI I/O pins.

Table 1. Pin Name Conventions

Generic Pin Names	MISO	MOSI	$\overline{\text{SS}}$	SPSCK	CGND
Full Pin Names	PF3/MISO	PF2/MOSI	PF0/$\overline{\text{SS}}$	PF1/SPSCK	CGND/EV_{SS}

Functional Description

Figure 1 shows the structure of the SPI module and Figure 2 shows the locations and contents of the SPI I/O registers.

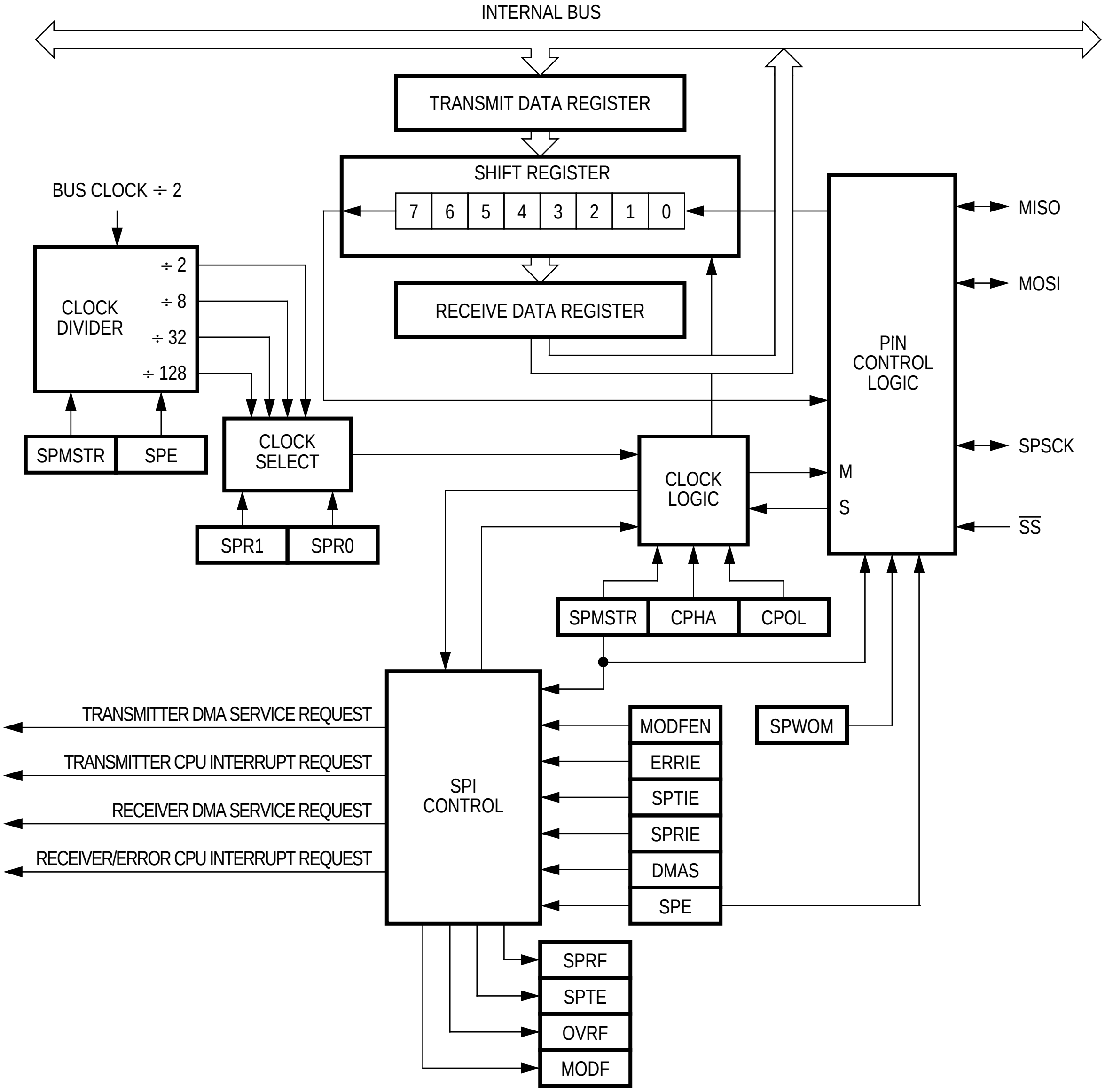


Figure 1. SPI Module Block Diagram

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

▨ = Unimplemented

Figure 2. SPI I/O Register Summary

Table 2. I/O Register Address Summary

Register	SPCR	SPSCR	SPDR
Address	$0010	$0011	$0012

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

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

The following paragraphs describe the operation of the SPI module.

Master Mode

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

NOTE: *Configure the SPI modules as master or slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI.*

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

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See SPI Status and Control Register on page 221.) Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

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

When the DMAS bit is set, the SPI status and control register does not have to be read to clear the SPRF bit. A read of the SPI data register by either the CPU or the DMA clears the SPRF bit. A write to the SPI data register by the CPU or by the DMA clears the SPTE bit.

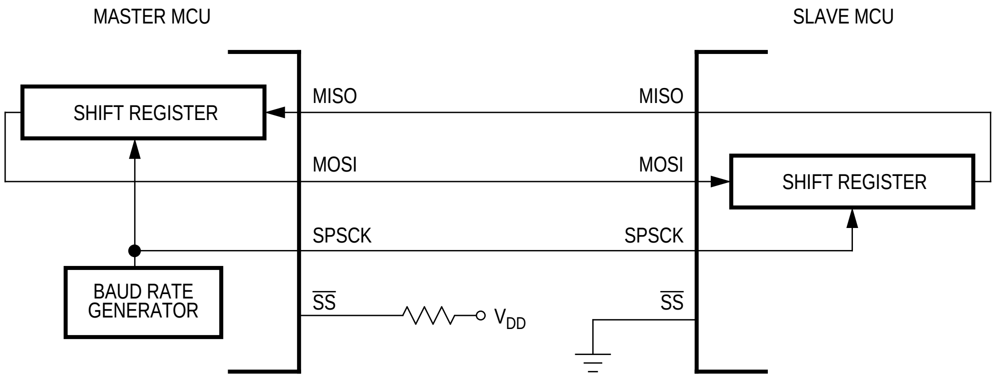


Figure 3. Full-Duplex Master-Slave Connections

Slave Mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode, the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be at logic 0. $\overline{SS}$ must remain low until the transmission is complete. (See Mode Fault Error on page 208.)

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

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

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

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. (See Transmission Formats on page 198.)

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

Transmission Formats

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

Clock Phase and Polarity Controls

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

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

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

Transmission Format When CPHA = 0

Figure 4 shows an SPI transmission in which CPHA is logic 0. The figure should not be used as a replacement for data sheet parametric information.Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See Mode Fault Error on page 208.) When CPHA = 0, the first SPSCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each byte transmitted as shown in **Figure 5**.

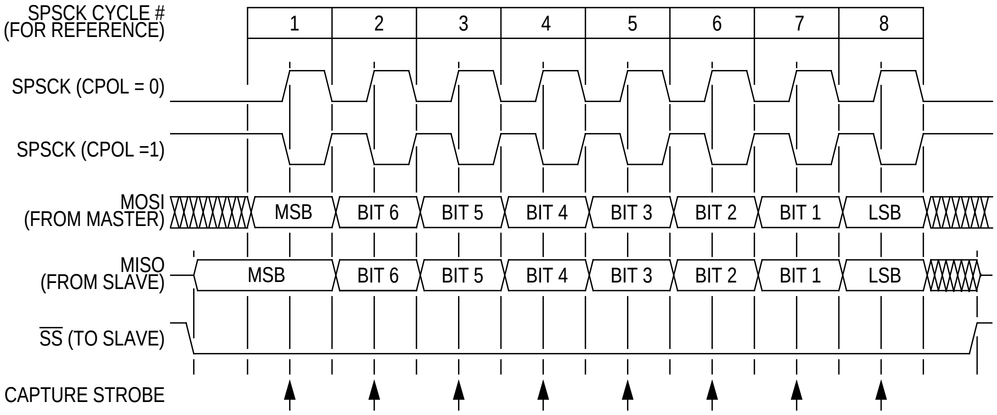


Figure 4. Transmission Format (CPHA = 0)

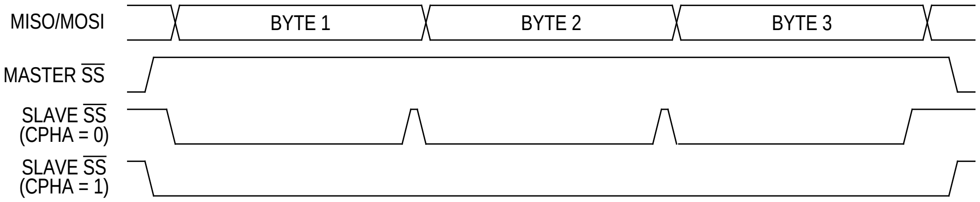


Figure 5. CPHA/$\overline{SS}$ Timing

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.

Transmission Format When CPHA = 1

Figure 6 shows an SPI transmission in which CPHA is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See Mode Fault Error on page 208.) When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

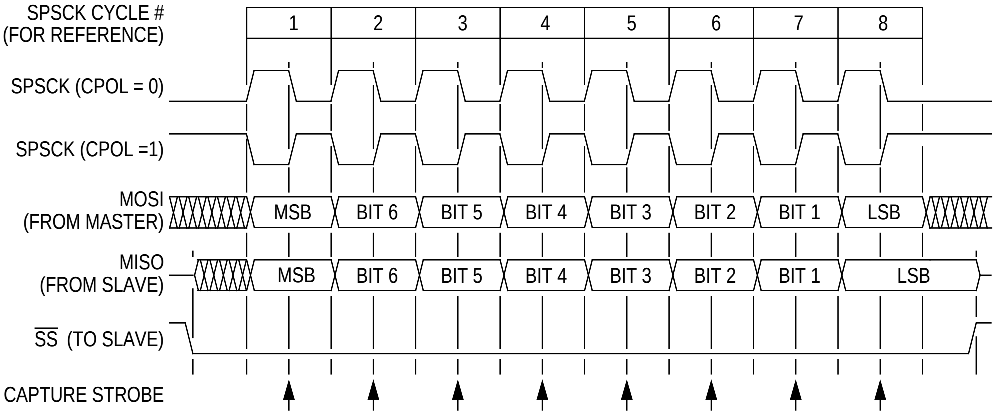


Figure 6. Transmission Format (CPHA = 1)

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

Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCK signal. When CPHA = 0, the SPSCK signal remains inactive for the first half of the first SPSCK cycle. When CPHA = 1, the first SPSCK cycle begins with an edge on the SPSCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See Figure 7 on page 202.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. SPSCK edges occur halfway through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR occurs relative to the slower SPSCK. This uncertainty causes the variation in the initiation delay shown in **Figure 7**. This delay is no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

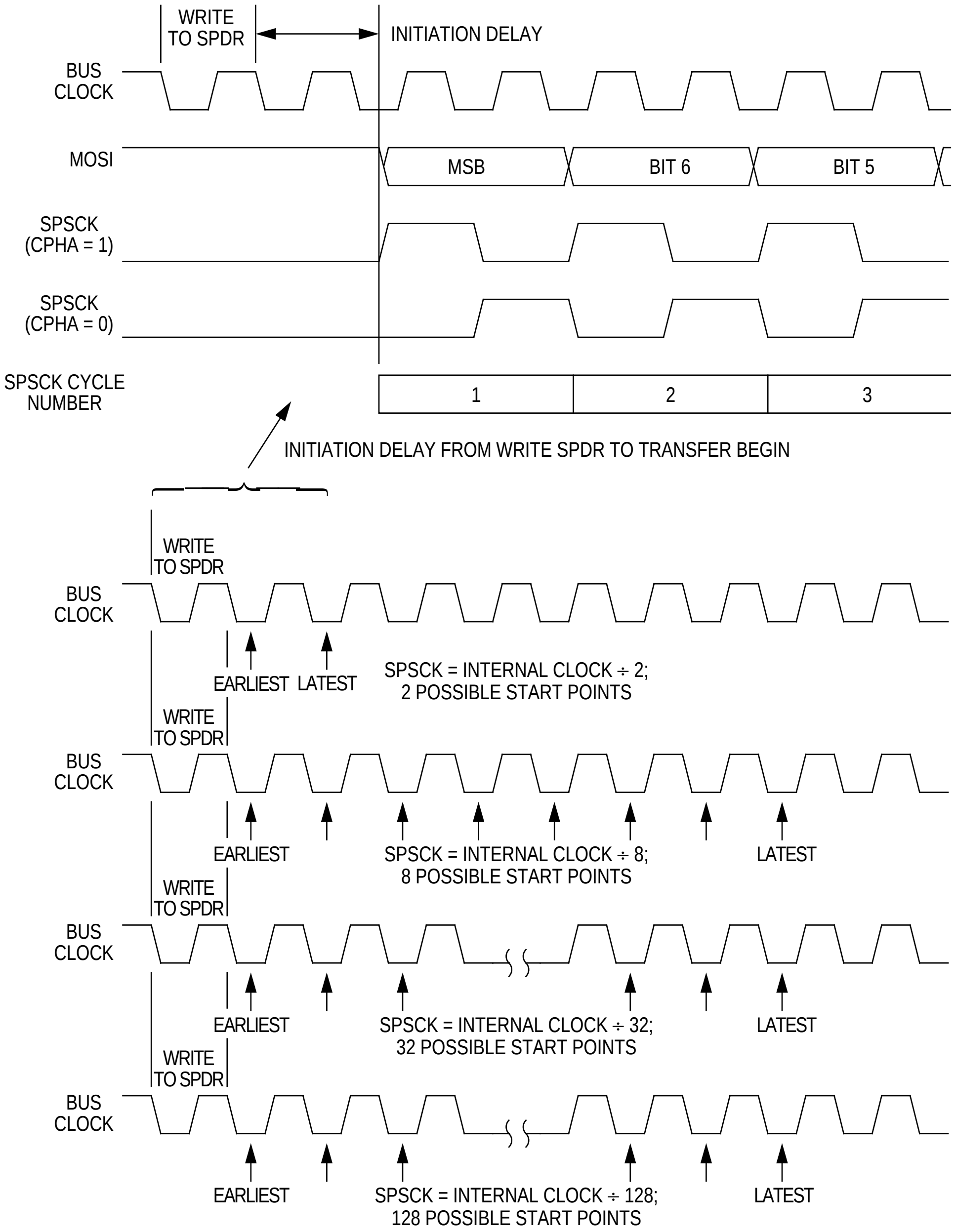
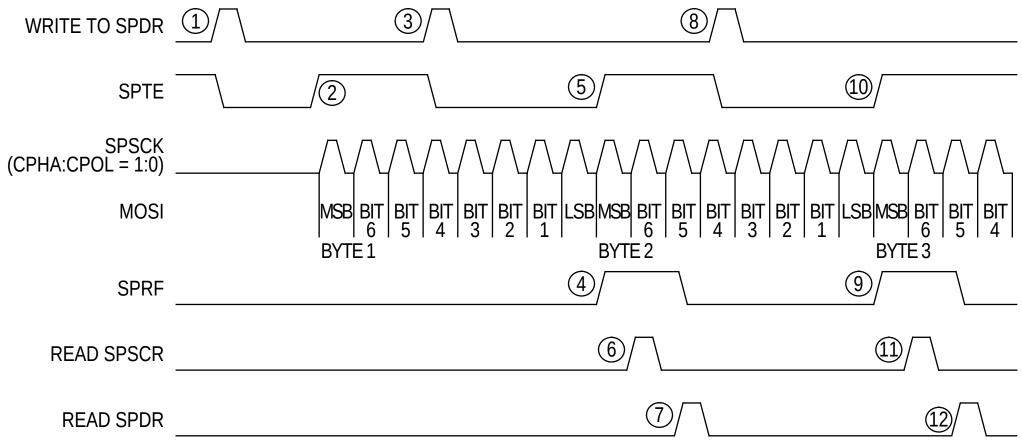


Figure 7. Transmission Start Delay (Master)

Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when the SPTE bit is high. Figure 8 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 8. SPRF/SPTE CPU Interrupt Timing

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

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

Error Conditions

The following flags signal SPI error conditions:

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

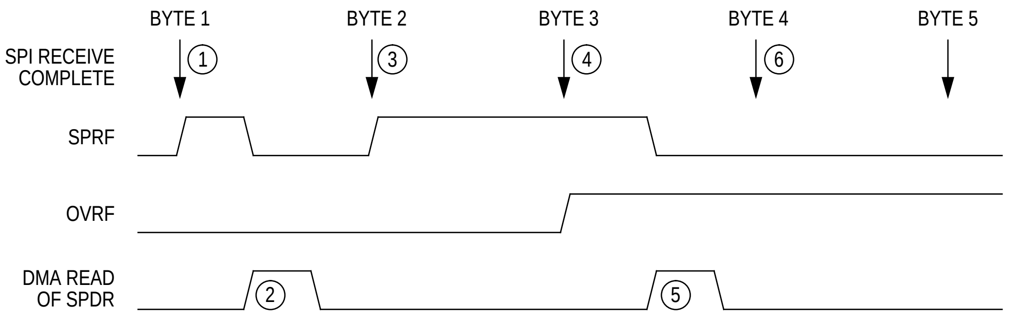
Overflow Error

The overflow flag (OVRF) becomes set if the receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. The bit 1 capture strobe occurs in the middle of SPSCK cycle 7. (See **Figure 4** on page 199 and **Figure 6** on page 200.) If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the receive data register and does not set the SPI receiver full bit (SPRF). The unread data that transferred to the receive data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI status and control register and then reading the SPI data register.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request, and MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 12 on page 212.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

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

OVRF interrupt requests to the CPU should be enabled when using the DMA to service the SPRF if there is any chance that the overflow condition might occur. (See Figure 9 on page 206.) Even if the DMA clears the SPRF bit, no new data transfers from the shift register to the receive data register with the OVRF bit high. This means that no new SPRF interrupt requests are generated until the CPU clears the OVRF bit. If the CPU reads the data register to clear the OVRF bit, it could clear a pending SPRF service request to the DMA.



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

2 DMA READS BYTE 1, CLEARING SPRF BIT.

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

4 BYTE 3 CAUSES OVERFLOW. BYTE 3 IS LOST.

5 DMA READS BYTE 2, CLEARING SPRF BIT.

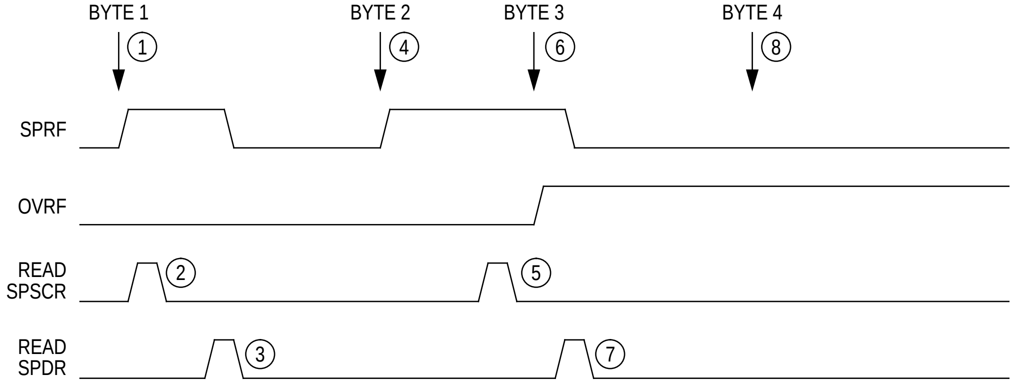
6 BYTE 4 IS LOST. NO NEW SPRF DMA SERVICE REQUESTS AND NO TRANSFERS TO DATA REGISTER UNTIL OVRF IS CLEARED.

Figure 9. Overflow Condition with DMA Service of SPRF

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

If an application requires the DMA to bring the MCU out of wait mode, enable the OVRF bit to generate CPU interrupt requests. An overflow condition in wait mode can cause the MCU to hang in wait mode because the DMA cannot reach its byte count. Setting the error interrupt enable bit (ERRIE) in the SPI status and control register enables the OVRF bit to bring the MCU out of wait mode.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. **Figure 10** shows how it is possible to miss an overflow. The first part of **Figure 10** shows how it is possible to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set in between the time that SPSCR and SPDR are read.



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 SPSCR WITH SPRF BIT SET AND OVRF BIT CLEAR.

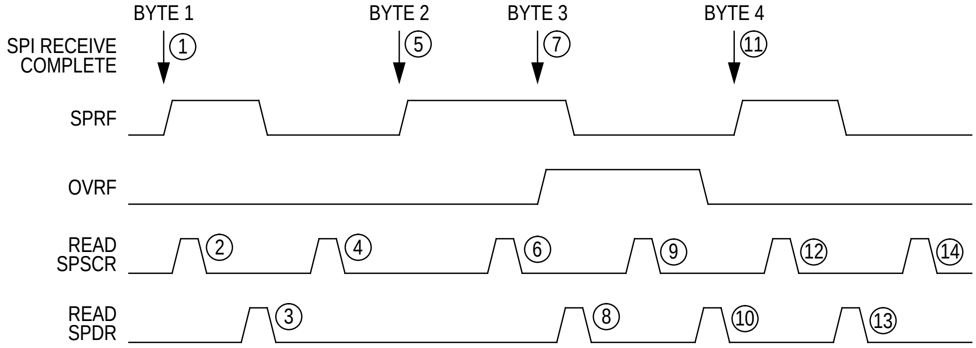
6 BYTE 3 SETS OVRF BIT. BYTE 3 IS LOST.

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 NOT CLEARED. BYTE 4 IS LOST.

Figure 10. Missed Read of Overflow Condition

In this case, an overflow can easily be missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it is not obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR following the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions can set the SPRF bit. **Figure 11** illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.



1 BYTE 1 SETS SPRF BIT.

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

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

4 CPU READS SPSCR AGAIN TO CHECK OVRF BIT.

5 BYTE 2 SETS SPRF BIT.

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

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

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

9 CPU READS SPSCR AGAIN TO CHECK OVRF BIT.

10 CPU READS BYTE 2 SPDR, CLEARING OVRF BIT.

11 BYTE 4 SETS SPRF BIT.

12 CPU READS SPSCR.

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

14 CPU READS SPSCR AGAIN TO CHECK OVRF BIT.

Figure 11. Clearing SPRF When OVRF Interrupt Is Not Enabled

Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SPSCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR. To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

- The $\overline{SS}$ pin of a slave SPI goes high during a transmission.
- The $\overline{SS}$ pin of a master SPI goes low at any time.

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. When the DMAS bit is low, the SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. When the DMAS bit is high, SPRF generates a receiver DMA service request instead of a CPU interrupt request, but MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 12 on page 212.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

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

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

NOTE: *When the MODF flag is set, it does not clear the SPMSTR bit. The SPMSTR bit has no function when SPE = 0. Reading SPMSTR when MODF = 1 indicates whether the SPI was a master or a slave when MODF became set.*

NOTE: *To prevent bus contention with another master SPI after a mode fault error, clear all SPI bits of the data direction register of the shared I/O port before enabling the SPI.*

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{\text{SS}}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{\text{SS}}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{\text{SS}}$ is already low. The transmission continues until the SPSCK returns to its idle level following the shift of the last data bit. (See Transmission Formats on page 198.)

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

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

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

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

Interrupts

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

Table 3. SPI Interrupts

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

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

The SPI transmitter interrupt enable bit (SPTIE) enables the SPTE flag to generate transmitter CPU interrupt requests or transmitter DMA service requests, provided that the SPI is enabled (SPE = 1).

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt requests or receiver DMA service requests, regardless of the state of the SPE bit. (See Figure 12.)

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

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

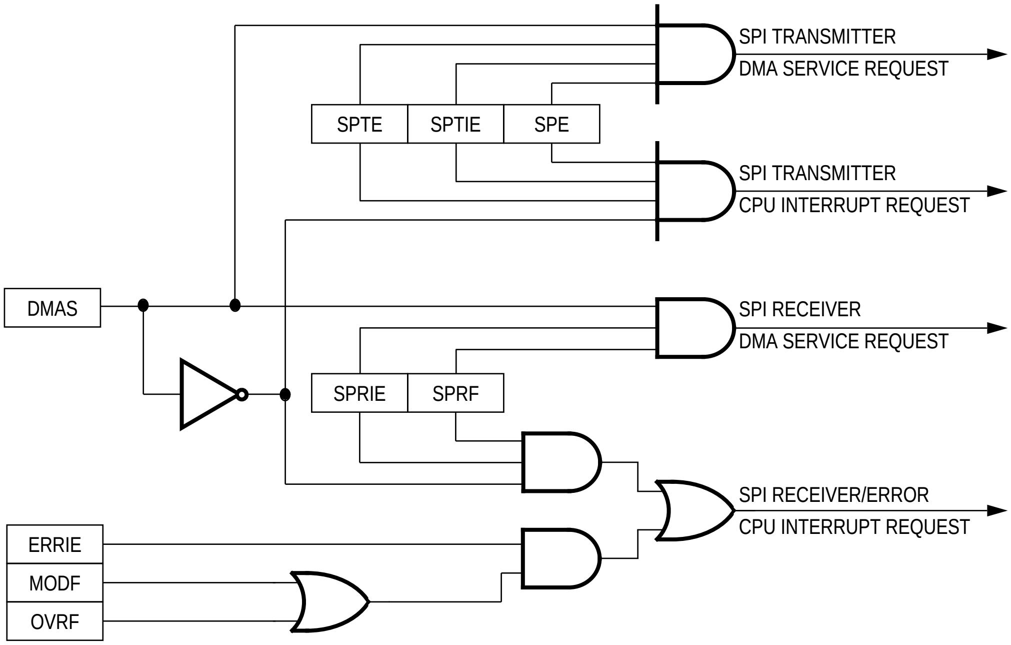


Figure 12. SPI Interrupt Request Generation

Resetting the SPI

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

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

The following items are reset only by a system reset:

- All control bits in the SPCR register.
- All control bits in the SPSCR register (MODFEN, ERRIE, SPR1, and SPR0).
- The status flags SPRF, OVRF, and MODF.

By not resetting the control bits when SPE is low, the user can clear SPE between transmissions without having to set all control bits again when SPE is set back high for the next transmission.

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

Low-Power Modes

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

Wait Mode

The SPI module remains active in wait mode. Any enabled CPU interrupt request from the SPI module can bring the MCU out of wait mode.

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

The DMA can service the SPI without exiting wait mode.

Stop Mode

The SPI module is inactive in stop mode. The STOP instruction does not affect SPI register states. SPI operation resumes after an external interrupt. If stop mode is exited by reset, any transfer in progress is aborted, and the SPI is reset.

SPI During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See Break Module (BRK) on page 141.)

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.

Since the SPTE bit cannot be cleared during a break with the BCFE bit cleared, a write to the transmit data register in break mode does not initiate a transmission nor is this data transferred into the shift register. Therefore, a write to the SPDR in break mode with the BCFE bit cleared has no effect.

I/O Signals

The SPI module has five I/O pins and shares four of them with a parallel I/O port.

- MISO — Master data in, slave data out
- MOSI — Master data out, slave data in
- SPSCK — Serial clock
- $\overline{\text{SS}}$ — Slave select
- CGND — 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}.

MISO (Master In/Slave Out)

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

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

MOSI (Master Out/Slave In)

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

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

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.

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

The $\overline{\text{SS}}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{\text{SS}}$ is used to select a slave. For CPHA = 0, the $\overline{\text{SS}}$ is used to define the start of a transmission. (See Transmission Formats on page 198.) Since it is used to indicate the start of a transmission, the $\overline{\text{SS}}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format. See **Figure 13**.

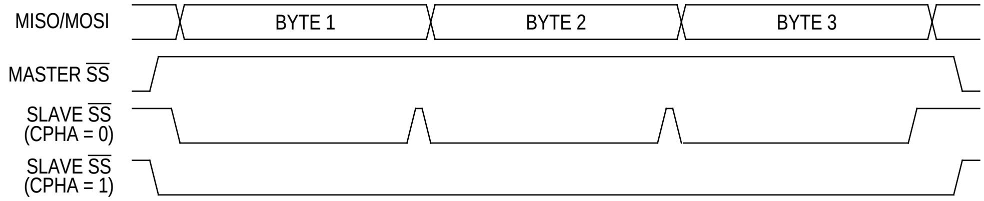

MISO/MOSI
BYTE 1
BYTE 2
BYTE 3
MASTER $\overline{\text{SS}}$
SLAVE $\overline{\text{SS}}$
(CPHA = 0)
SLAVE $\overline{\text{SS}}$
(CPHA = 1)

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

When an SPI is configured as a slave, the $\overline{\text{SS}}$ pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{\text{SS}}$ from creating a MODF error. (See SPI Status and Control Register on page 221.)

NOTE: *A logic 1 voltage on the $\overline{\text{SS}}$ pin of a slave SPI puts the MISO pin in a high-impedance state. The slave SPI ignores all incoming SPSCK clocks, even if it was already in the middle of a transmission.*

When an SPI is configured as a master, the $\overline{\text{SS}}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See Mode Fault Error on page 208.) For the state of the $\overline{\text{SS}}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{\text{SS}}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{\text{SS}}$ pin by configuring the appropriate pin as an input and reading the port data register. (See Table 4.)

Table 4. SPI Configuration

SPE	SPMSTR	MODFEN	SPI Configuration	State of $\overline{\text{SS}}$ Logic
0	X[1]	X	Not Enabled	General-purpose I/O; $\overline{\text{SS}}$ ignored by SPI
1	0	X	Slave	Input-only to SPI
1	1	0	Master without MODF	General-purpose I/O; $\overline{\text{SS}}$ ignored by SPI
1	1	1	Master with MODF	Input-only to SPI

[1] X = don't care

CGND (Clock Ground)

CGND 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 CGND pin of the master.

I/O Registers

Three registers control and monitor SPI operation:

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

SPI Control Register

The SPI control register does the following:

- Enables SPI module interrupt requests
- Selects CPU interrupt requests or DMA service requests
- Configures the SPI module as master or slave
- Selects serial clock polarity and phase

- Configures the SPSCK, MOSI, and MISO pins as open-drain outputs
- Enables the SPI module

Address: $0010

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

Figure 14. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

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

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

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

DMAS —DMA Select Bit

This read/write bit selects DMA service requests when:

- The SPI receiver full bit, SPRF, becomes set and the SPI receiver interrupt enable bit, SPIE, is also set
- The SPI transmitter empty bit, SPTE, becomes set and the SPI transmitter interrupt enable bit, SPTIE, is also set

Setting the DMAS bit disables SPRF CPU interrupt requests and SPTE CPU interrupt requests. Reset clears the DMAS bit.

1 = SPRF DMA and SPTE DMA service requests selected
SPRF CPU and SPTE CPU interrupt requests disabled
0 = SPRF DMA and SPTE DMA service requests disabled
SPRF CPU and SPTE CPU interrupt requests selected

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 4 on page 199 and **Figure 6** on page 200.) To transmit data between SPI modules, the SPI modules must have identical CPOL values. Reset clears the CPOL bit.

CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See **Figure 4** on page 199 and **Figure 6** on page 200.) To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the $\overline{\text{SS}}$ pin of the slave SPI module must be set to logic 1 between bytes. (See Figure 13 on page 217.) Reset sets the CPHA bit.

SPWOM — SPI Wired-OR Mode Bit

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

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

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See Resetting the SPI on page 213.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

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

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

SPI Status and Control Register

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

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

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

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

Address: $0011

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

▨ = Unimplemented

Figure 15. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

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

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

Reset clears the SPRF bit.

1 = Receive data register full

0 = Receive data register not full

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

ERRIE — Error Interrupt Enable Bit

This read/write bit enables the MODF and OVRF bits to generate CPU interrupt requests. Reset clears the ERRIE bit.

1 = MODF and OVRF can generate CPU interrupt requests
0 = MODF and OVRF cannot generate CPU interrupt requests

OVRF — Overflow Bit

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

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

This clearable, read-only flag is set in a slave SPI if the $\overline{\text{SS}}$ pin goes high during a transmission with the MODFEN bit set. In a master SPI, the MODF flag is set if the $\overline{\text{SS}}$ pin goes low at any time with the MODFEN bit set. Clear the MODF bit by reading the SPI status and control register (SPSCR) with MODF set and then writing to the SPI control register (SPCR). Reset clears the MODF bit.

1 = $\overline{\text{SS}}$ pin at inappropriate logic level
0 = $\overline{\text{SS}}$ pin at appropriate logic level

SPTE — SPI Transmitter Empty Bit

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

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

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

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

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 for general-purpose I/O.

If the MODFEN bit is set, then this pin is not available for 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 SS (Slave Select) on page 216.)

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 Mode Fault Error on page 208.)

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 5**. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 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:

$$\text{Baud rate} = \frac{\text{BUS CLOCK}}{\text{BD}}$$

where:

CGMOUT = base clock output of the clock generator module (CGM)
BD = baud rate divisor

SPI Data Register

The SPI data register consists of the read-only receive data register and the write-only transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate registers that can contain different values. (See Figure 1 on page 194.)

Address: $0012

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

Figure 16. SPI Data Register (SPDR)

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

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

Serial Communications Interface Module (SCI)

Contents

Introduction

The SCI allows asynchronous communications with peripheral devices and other MCUs.

Features

- Full Duplex Operation
- Standard Mark/Space Non-Return-to-Zero (NRZ) Format
- 32 Programmable Baud Rates
- Programmable 8-Bit or 9-Bit Character Length
- Separately Enabled Transmitter and Receiver
- Separate Receiver and Transmitter CPU Interrupt Requests
- Separate Receiver and Transmitter DMA Service Requests
- Programmable Transmitter Output Polarity
- Two Receiver 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

Pin Name Conventions

The generic names of the SCI I/O pins are:

- RxD (receive data)
- TxD (transmit data)

SCI I/O lines are implemented by sharing parallel I/O port pins. The full name of an SCI input or output reflects the name of the shared port pin. **Table 1** shows the full names and the generic names of the SCI I/O pins.The generic pin names appear in the text of this section.

Table 1. Pin Name Conventions

Generic Pin Names	RxD	TxD
Full Pin Names	PE1/RxD	PE2/TxD

Functional Description

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

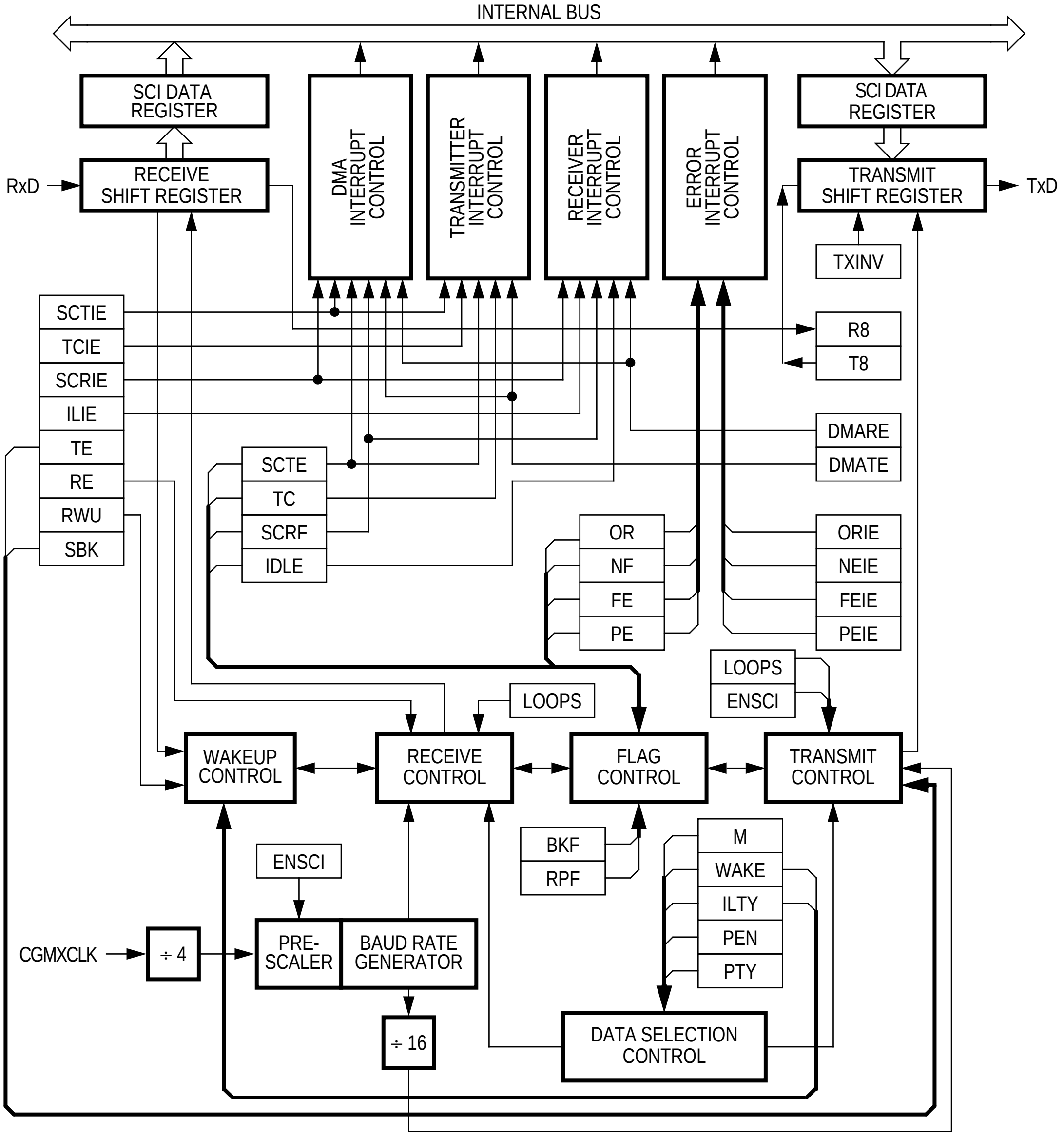


Figure 1. SCI Module Block Diagram

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
SCI Control Register 1 (SCC1)	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 2 (SCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 3 (SCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
	Write:	▨							
	Reset:	U	U	0	0	0	0	0	0
SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	1	1	0	0	0	0	0	0
SCI Status Register 2 (SCS2)	Read:	▨	▨	▨	▨	▨	▨	BKF	RPF
	Write:							▨	▨
	Reset:	0	0	0	0	0	0	0	0
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							
SCI Baud Rate Register (SCBR)	Read:	▨	▨	SCP1	SCP0	R	SCR2	SCR1	SCR0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented   U = Unaffected   R = Reserved

Figure 2. SCI I/O Register Summary

Table 2. SCI I/O Register Address Summary

Register	SCC1	SCC2	SCC3	SCS1	SCS2	SCDR	SCBR
Address	$0013	$0014	$0015	$0016	$0017	$0018	$0019

Data Format

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

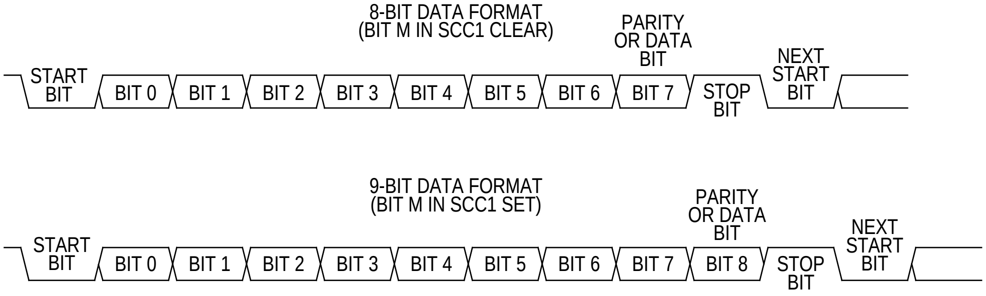


Figure 3. SCI Data Formats

Transmitter

Figure 4 shows the structure of the SCI transmitter.

Character Length

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When transmitting 9-bit data, bit T8 in SCI control register 3 (SCC3) is the ninth bit (bit 8).

Character Transmission

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

1. Enable the SCI by writing a logic 1 to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2).
3. Clear the SCI transmitter empty bit by first reading SCI status register 1 (SCS1) and then writing to the SCDR. In a DMA transfer, the DMA automatically clears the SCTE bit by writing to the SCDR.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of logic 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

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

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

When the transmit shift register is not transmitting a character, the 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.

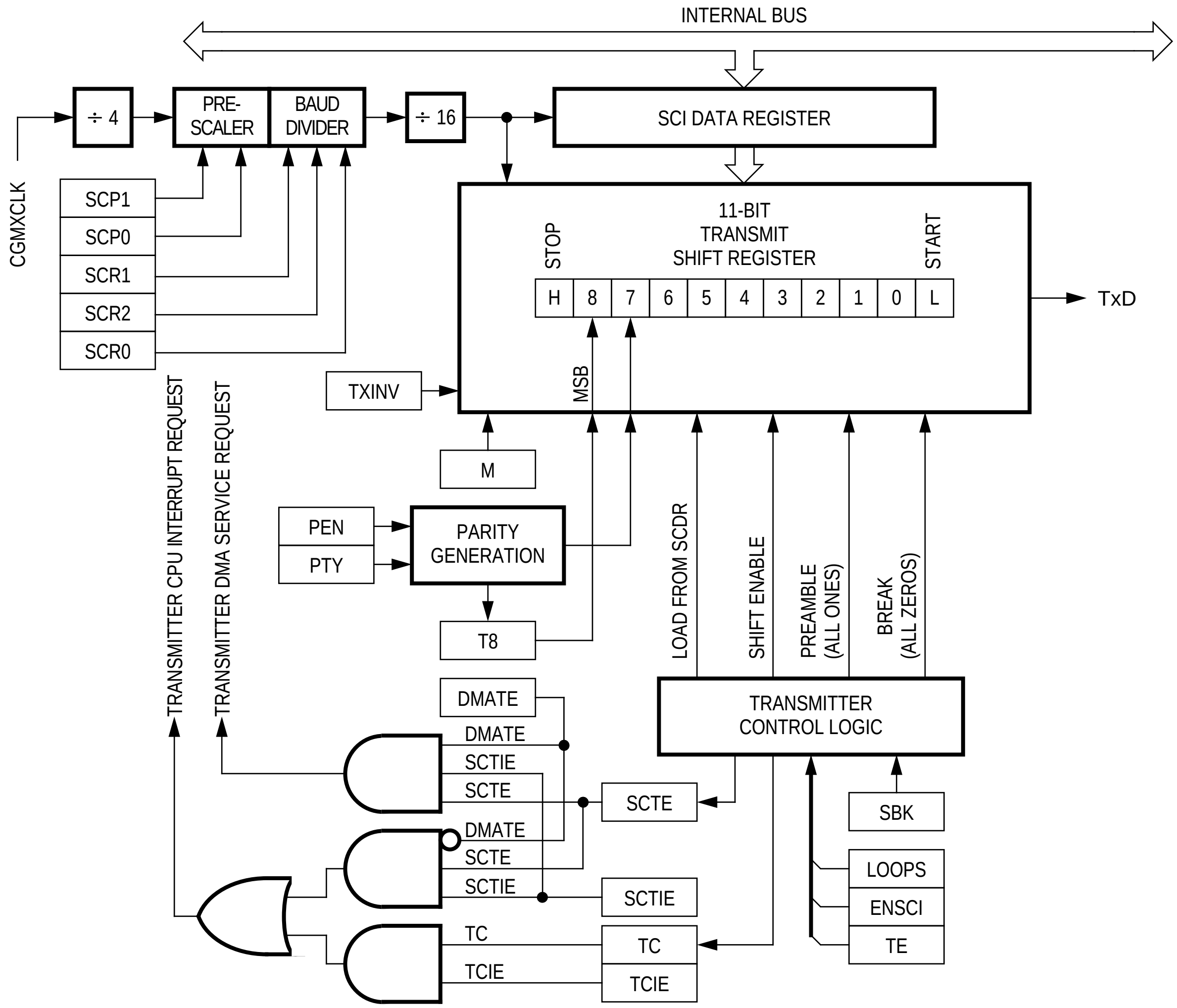


Figure 4. SCI Transmitter

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
SCI Control Register 1 (SCC1)	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 2 (SCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 3 (SCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
	Write:	▨							
	Reset:	U	U	0	0	0	0	0	0
SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	1	1	0	0	0	0	0	0
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							
SCI Baud Rate Register (SCBR)	Read:	▨	▨	SCP1	SCP0	R	SCR2	SCR1	SCR0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented U = Unaffected R = Reserved

Figure 5. SCI Transmitter I/O Register Summary

Table 3. SCI Transmitter I/O Address Summary

Register	SCC1	SCC2	SCC3	SCS1	SCDR	SCBR
Address	$0013	$0014	$0015	$0016	$0018	$0019

Break Characters

Writing a logic 1 to the send break bit, SBK, in SCC2 loads the transmit shift register with a break character. A break character contains all logic zeros and has no start, stop, or parity bit. Break character length depends on the M bit in SCC1. As long as SBK is at logic 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

Idle Characters

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

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

NOTE: *When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current character shifts out to the TxD pin. Setting TE after the stop bit appears on TxD causes data previously written to the SCDR to be lost.*

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

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 SCI Control Register 1 on page 253.)

Transmitter Interrupts

The following conditions can generate CPU interrupt requests from the SCI transmitter:

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

Receiver

Figure 6 shows the structure of the SCI receiver.

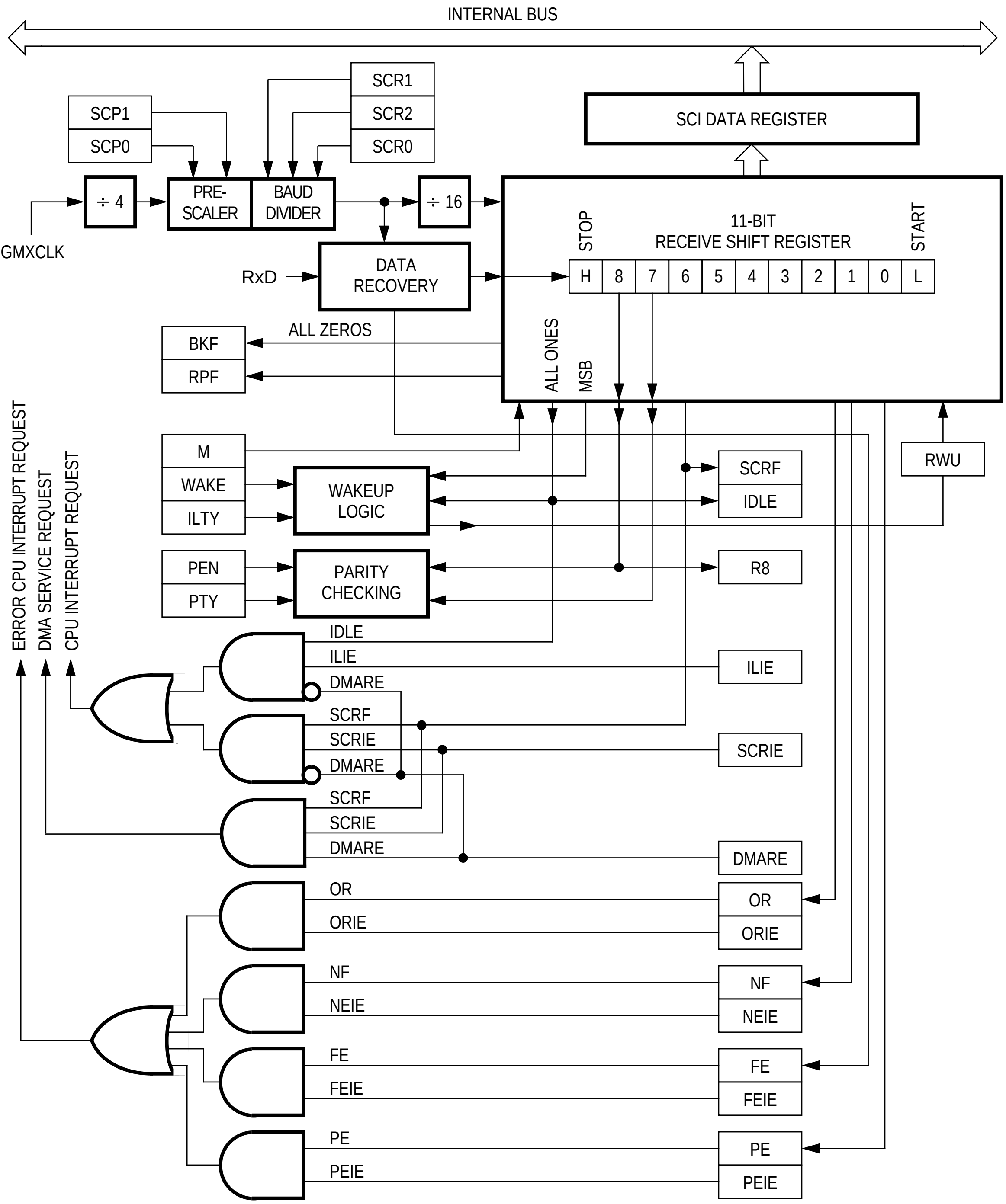


Figure 6. SCI Receiver Block Diagram

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
SCI Control Register 1 (SCC1)	Read:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 2 (SCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
	Write:								
	Reset:	0	0	0	0	0	0	0	0
SCI Control Register 3 (SCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
	Write:	▨							
	Reset:	U	U	0	0	0	0	0	0
SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	Reset:	1	1	0	0	0	0	0	0
SCI Status Register 2 (SCS2)	Read:	▨	▨	▨	▨	▨	▨	BKF	RPF
	Write:							▨	▨
	Reset:	0	0	0	0	0	0	0	0
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							
SCI Baud Rate Register (SCBR)	Read:	▨	▨	SCP1	SCP0	R	SCR2	SCR1	SCR0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented U = Unaffected R = Reserved

Figure 7. SCI I/O Register Summary

Table 4. SCI Receiver I/O Address Summary

Register	SCC1	SCC2	SCC3	SCS1	SCS2	SCDR	SCBR
Address	$0013	$0014	$0015	$0016	$0017	$0018	$0019

Character Length

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in SCI control register 2 (SCC2) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

Character Reception

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

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

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

Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see Figure 8):

- 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.

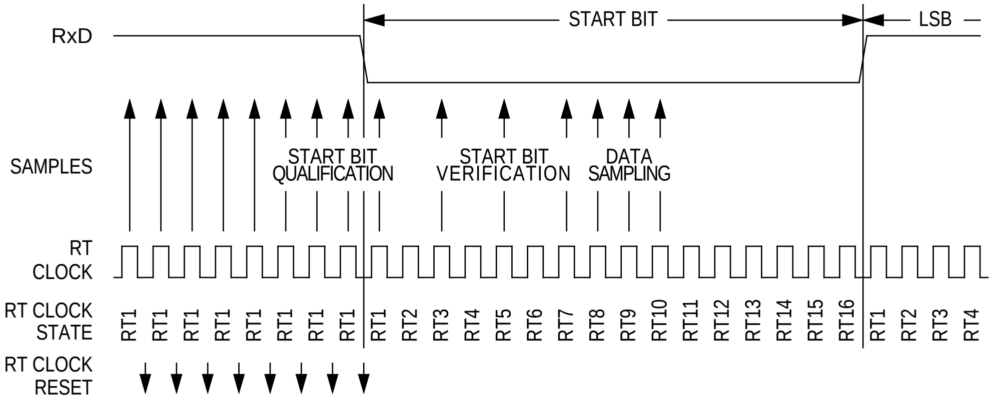

RxD
START BIT
LSB
SAMPLES
START BIT QUALIFICATION
START BIT VERIFICATION
DATA SAMPLING
RT CLOCK
RT CLOCK STATE
RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT1 RT2 RT3 RT4 RT5 RT6 RT7 RT8 RT9 RT10 RT11 RT12 RT13 RT14 RT15 RT16 RT1 RT2 RT3 RT4
RT CLOCK RESET

Figure 8. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. **Table 5** summarizes the results of the start bit verification samples.

Table 5. 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 6** summarizes the results of the data bit samples.

Table 6. 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 7 summarizes the results of the stop bit samples.

Table 7. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

Framing Errors

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

Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming character, it resynchronizes the RT clock on any valid falling edge within the character. Resynchronization within characters corrects misalignments between transmitter bit times and receiver bit times.

Slow Data Tolerance

Figure 9 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

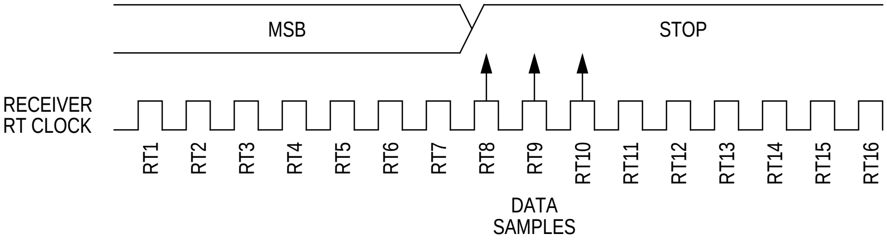


Figure 9. Slow Data

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

With the misaligned character shown in Figure 9, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

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

$$\left|\frac{154 - 147}{154}\right| \times 100 = 4.54\%$$

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

With the misaligned character shown in Figure 9, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

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

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

Fast Data Tolerance

Figure 10 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

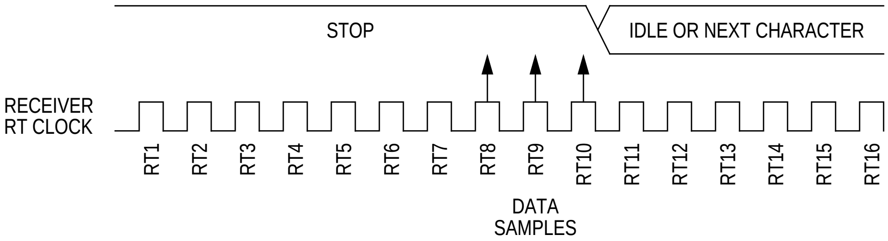


Figure 10. Fast Data

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

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

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

$$\left|\frac{154-160}{154}\right| \times 100 = 3.9\dot{0}\%$$

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

With the misaligned character shown in Figure 10, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

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

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

Receiver Wakeup

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

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

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

NOTE: *With the WAKE bit clear, setting the RWU bit after the RxD pin has been idle may cause the receiver to wake up immediately.*

Receiver Interrupts

The following sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate a receiver CPU interrupt request or a receiver DMA service request. Setting the SCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver CPU interrupts. Setting both the SCRIE bit and the DMA receive enable bit, DMARE, in SCC3 enables receiver DMA service requests and disables receiver CPU interrupt requests.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

NOTE: *When receiver DMA service requests are enabled (DMARE = 1), then receiver CPU interrupt requests are disabled.*

Error Interrupts

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

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate SCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the SCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate SCI error CPU interrupt requests.
- Framing error (FE) — The FE bit in SCS1 is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in SCC3 enables FE to generate SCI error CPU interrupt requests.

- Parity error (PE) — The PE bit in SCS1 is set when the SCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, in SCC3 enables PE to generate SCI error CPU interrupt requests.

Error Flags During DMA Service Requests

When the DMA is servicing the SCI receiver, it clears the SCRF bit when it reads the SCI data register. The DMA does not clear the other status bits (BKF or RPF), nor does it clear error flags (OR, NF, FE, and PE). To clear error flags while the DMA is servicing the receiver, enable SCI error CPU interrupts and clear the bits in an interrupt routine. The application may require retransmission in case of error. If the application requires the receptions to continue, note the following latency considerations:

1. If interrupt latency is short enough for an error bit to be serviced before the next SCRF, then it can be determined which byte caused the error. If interrupt latency is long enough for a new SCRF to occur before servicing an error bit, then:
   a. It cannot be determined whether the error bit being serviced is due to the byte in the SCI data register or to a previous byte. Multiple errors can accumulate that correspond to different bytes. In a message-based system, you may have to repeat the entire message.
   b. When the DMA is enabled to service the SCI receiver, merely reading the SCI data register clears the SCRF bit. The second step in clearing an error bit, reading the SCI data register, could inadvertently clear a new, unserviced SCRF that occurred during the error-servicing routine. Then the DMA would ignore the byte that set the new SCRF, and the new byte would be lost.
      To prevent clearing of an unserviced SCRF bit, clear the SCRIE bit at the beginning of the error-servicing interrupt routine and set it at the end. Clearing SCRIE disables DMA service so that both a read of SCS1 and a read of SCDR are required to clear the SCRF bit. Setting SCRIE enables DMA service so that the DMA can recognize a service request that occurred during the error-servicing interrupt routine.

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

2. DMA latency should be short enough so that an SCRF is serviced before the next SCRF occurs. If DMA latency is long enough for a new SCRF to occur before servicing an error bit, then:

   a. Overruns occur. Set the ORIE bit to enable SCI error CPU interrupt requests and service the overrun in an interrupt routine. In a message-based system, disable the DMA in the interrupt routine and manually recover. Otherwise, the byte that was lost in the overrun could prevent the DMA from reaching its byte count. If the DMA reaches it byte count in the following message, two messages may be corrupted.

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

Low-Power Modes

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

Wait Mode

The SCI module remains active in wait mode. Any enabled CPU interrupt request from the SCI module can bring the MCU out of wait mode.

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

The DMA can service the SCI without exiting wait mode.

Stop Mode

The SCI module is inactive in stop mode. The STOP instruction does not affect SCI register states. SCI module operation resumes after the MCU exits stop mode.

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

SCI During Break Module Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See Break Module (BRK) on page 141.)

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.

I/O Signals

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

- TxD — Transmit data
- RxD — Receive data

TxD (Transmit Data)

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

RxD (Receive Data)

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

I/O Registers

The following I/O registers control and monitor SCI operation:

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

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:	LOOPS	ENSCI	TXINV	M	WAKE	ILLTY	PEN	PTY
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 11. SCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

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

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI Bit

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

1 = SCI enabled
0 = SCI disabled

TXINV — Transmit Inversion Bit

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

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

M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are eight or nine bits long. (See Table 8.) 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 RxD pin. Reset clears the WAKE bit.

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

This read/write bit determines when the SCI starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. Reset clears the ILTY bit.

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

PEN — Parity Enable Bit

This read/write bit enables the SCI parity function. (See Table 8.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See Figure 3 on page 232.) 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 8.) 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 8. 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

SCI Control Register 2

SCI control register 2:

- Enables the following CPU interrupt requests and DMA service requests:
  - Enables the SCTE bit to generate transmitter CPU interrupt requests or transmitter DMA service requests.
  - Enables the TC bit to generate transmitter CPU interrupt requests.
  - Enables the SCRF bit to generate receiver CPU interrupt requests or receiver DMA service requests.
  - Enables the IDLE bit to generate receiver CPU interrupt requests.
- Enables the transmitter.
- Enables the receiver.
- Enables SCI 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 12. SCI Control Register 2 (SCC2)

SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests or DMA service requests. Setting the SCTIE bit and clearing the DMA transfer enable bit, DMATE, in SCC3 enables the SCTE bit to generate CPU interrupt requests. Setting both the SCTIE and DMATE bits enables the SCTE bit to generate DMA service requests. Reset clears the SCTIE bit.

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

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate SCI transmitter CPU interrupt requests. Reset clears the TCIE bit.

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

SCRIE — SCI Receive Interrupt Enable Bit

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

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

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate SCI receiver CPU interrupt requests. Reset clears the ILIE bit.

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

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

TE — Transmitter Enable Bit

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

1 = Transmitter enabled
0 = Transmitter disabled

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

RE — Receiver Enable Bit

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

1 = Receiver enabled
0 = Receiver disabled

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

RWU — Receiver Wakeup Bit

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

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

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

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

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

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 SCI receiver full (SCRF) DMA service requests.
- Enables SCI transmitter empty (SCTE) DMA service requests.
- Enables the following interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts

Address: $0015

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

▨ = Unimplemented U = Unaffected

Figure 13. SCI Control Register 3 (SCC3)

R8 — Received Bit 8

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

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

T8 — Transmitted Bit 8

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

DMARE — DMA Receive Enable Bit

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

1 = DMA enabled to service SCI receiver DMA service requests generated by the SCRF bit (SCI receiver CPU interrupt requests disabled)

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

NOTE: *To enable the SCRF bit to generate DMA service requests, the SCI receive interrupt enable bit (SCRIE) must be set.*

DMATE — DMA Transfer Enable Bit

This read/write bit enables SCI transmitter empty (SCTE) DMA service requests. (See SCI Status Register 1 on page 262.) Setting the DMATE bit disables SCTE CPU interrupt requests. Reset clears DMATE.

1 = SCTE DMA service requests enabled
SCTE CPU interrupt requests disabled

0 = SCTE DMA service requests disabled
SCTE CPU interrupt requests enabled

NOTE: *To enable the SCTE bit to generate DMA service requests, the SCI transmit interrupt enable bit (SCTIE) must be set.*

ORIE — Receiver Overrun Interrupt Enable Bit

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

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

NEIE — Receiver Noise Error Interrupt Enable Bit

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

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

FEIE — Receiver Framing Error Interrupt Enable Bit

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

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

PEIE — Receiver Parity Error Interrupt Enable Bit

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

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

SCI Status Register 1

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

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

Address: $0016

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

▨ = Unimplemented

Figure 14. SCI Status Register 1 (SCS1)

SCTE — SCI Transmitter Empty Bit

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an SCI transmitter CPU interrupt request or an SCI transmitter DMA service request. When the SCTIE bit in SCC2 is set and the DMATE bit in SCC3 is clear, SCTE generates an SCI transmitter CPU interrupt request. With both the SCTIE and DMATE bits set, SCTE generates an SCI transmitter DMA service request. In normal operation, clear the SCTE bit by reading SCS1 with SCTE set and then writing to SCDR. In DMA transfers, the DMA automatically clears the SCTE bit when it writes to the SCDR. Reset sets the SCTE bit.

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

NOTE: *When DMATE = 1, a write by the CPU to the SCI data register can clear the SCTE bit inadvertently and cause the DMA to miss a service request.*

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

TC — Transmission Complete Bit

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an SCI transmitter CPU interrupt request if the TCIE bit in SCC2 is also set. When the DMA services an SCI transmitter DMA service request, the DMA clears the TC bit by writing to the SCDR. TC is cleared automatically when data, preamble, or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — SCI Receiver Full Bit

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

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

NOTE: *When DMARE = 1, a read by the CPU of the SCI data register can clear the SCRF bit inadvertently and cause the DMA to miss a service request.*

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 15** shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

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

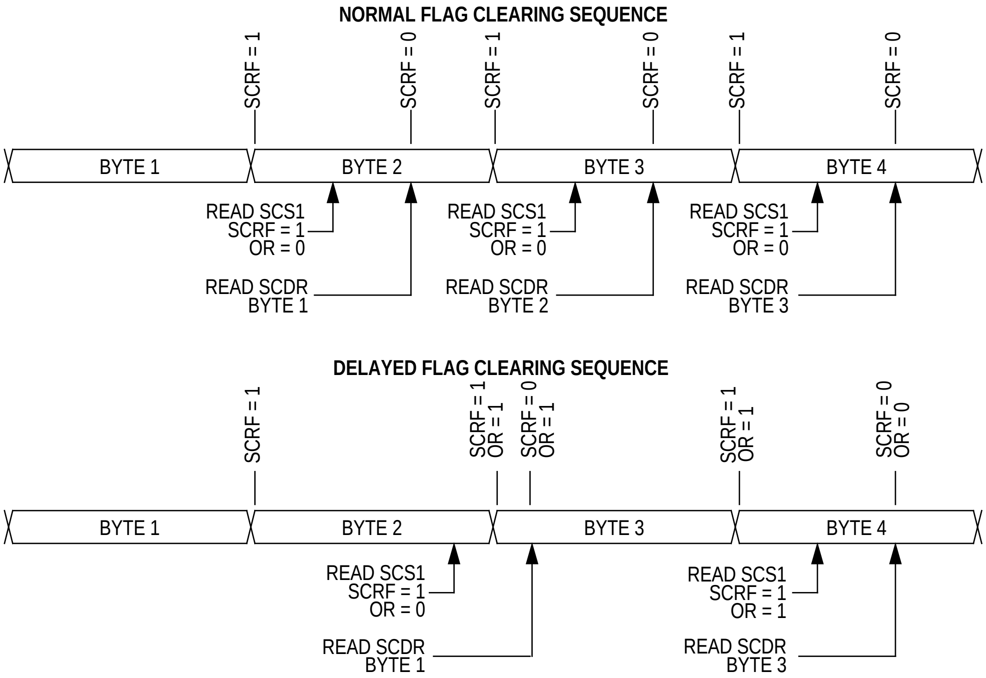


Figure 15. Flag Clearing Sequence

NF — Receiver Noise Flag Bit

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

1 = Noise detected

0 = No noise detected

FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a 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

SCI Status Register 2

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

- Break character detected
- Incoming data

Address: $0017

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

▨ = Unimplemented

Figure 16. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

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

1 = Break character detected

0 = No break character detected

RPF —Reception in Progress Flag Bit

This read-only bit is set when the receiver detects a 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

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. SCI Data Register (SCDR)

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

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

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

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:	▨	▨	SCP1	SCP0	R	SCR2	SCR1	SCR0
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 18. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in **Table 9**. Reset clears SCP1 and SCP0.

Table 9. SCI Baud Rate Prescaling

SCP[1:0]	Prescaler Divisor (PD)
00	1
01	3
10	4
11	13

SCR2 –SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in **Table 10**. Reset clears SCR2–SCR0.

Table 10. SCI Baud Rate Selection

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

Use the following formula to calculate the SCI baud rate:

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

where:

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

Table 11 shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

Table 11. SCI Baud Rate Selection Examples

SCP[1:0]	Prescaler Divisor (PD)	SCR[2:1:0]	Baud Rate Divisor (BD)	Baud Rate ($f_{Crystal}$ = 4.9152 MHz)
00	1	000	1	76,800
00	1	001	2	38,400
00	1	010	4	19,200
00	1	011	8	9600
00	1	100	16	4800
00	1	101	32	2400
00	1	110	64	1200
00	1	111	128	600
01	3	000	1	25,600
01	3	001	2	12,800
01	3	010	4	6400
01	3	011	8	3200
01	3	100	16	1600
01	3	101	32	800
01	3	110	64	400
01	3	111	128	200
10	4	000	1	19,200
10	4	001	2	9600
10	4	010	4	4800
10	4	011	8	2400
10	4	100	16	1200
10	4	101	32	600
10	4	110	64	300
10	4	111	128	150
11	13	000	1	5908
11	13	001	2	2954
11	13	010	4	1477
11	13	011	8	739
11	13	100	16	369
11	13	101	32	185
11	13	110	64	92
11	13	111	128	46

Input/Output Ports

Contents

Introduction

Fifty-four bidirectional input-output (I/O) pins form eight parallel ports. All I/O pins are programmable as inputs or outputs.

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

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
Port A Data Register (PORTA)	Read:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
	Write:								
	Reset:	Unaffected by Reset							
Port B Data Register (PORTB)	Read:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
	Write:								
	Reset:	Unaffected by Reset							
Port C Data Register (PORTC)	Read:	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
	Write:								
	Reset:	Unaffected by Reset							
Port D Data Register (PORTD)	Read:	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
	Write:								
	Reset:	Unaffected by Reset							
Data Direction Register A (DDRA)	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
	Write:								
	Reset:	0	0	0	0	0	0	0	0
Data Direction Register B (DDRB)	Read:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
	Write:								
	Reset:	0	0	0	0	0	0	0	0
Data Direction Register C (DDRC)	Read:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
	Write:								
	Reset:	0	0	0	0	0	0	0	0
Data Direction Register D (DDRD)	Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
	Write:								
	Reset:	0	0	0	0	0	0	0	0
Port E Data Register (PORTE)	Read:	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
	Write:								
	Reset:	Unaffected by Reset							
Port F Data Register (PORTF)	Read:	0	0	PF5	PF4	PF3	PF2	PF1	PF0
	Write:	▨	▨						
	Reset:	Unaffected by Reset							

▨ = Unimplemented

Figure 1. I/O Register Summary

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
Port G Data Register (PORTG)	Read:	0	0	0	0	PG3	PG2	PG1	PG0
	Write:	▨	▨	▨	▨				
	Reset:	Unaffected by Reset							
Port H Data Register (PORTH)	Read:	0	0	0	0	PH3	PH2	PH1	PH0
	Write:	▨	▨	▨	▨				
	Reset:	Unaffected by Reset							
Data Direction Register E (DDRE)	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
	Write:								
	Reset:	0	0	0	0	0	0	0	0
Data Direction Register F (DDRF)	Read:	0	0	DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
	Write:	▨	▨						
	Reset:	0	0	0	0	0	0	0	0
Data Direction Register G (DDRG)	Read:	0	0	0	0	DDRG3	DDRG2	DDRG1	DDRG0
	Write:	▨	▨	▨	▨				
	Reset:	0	0	0	0	0	0	0	0
Data Direction Register H (DDRH)	Read:	0	0	0	0	DDRH3	DDRH2	DDRH1	DDRH0
	Write:	▨	▨	▨	▨				
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 1. I/O Register Summary (Continued)

Table 1. I/O Register Address Summary

Register	PORTA	PORTB	PORTC	PORTD	DDRA	DDRB	DDRC	DDRD	PORTE	PORTF
Address	$0000	$0001	$0002	$0003	$0004	$0005	$0006	$0007	$0008	$0009

Register	PORTG	PORTH	DDRE	DDRF	DDRG	DDRH
Address	$000A	$000B	$000C	$000D	$000E	$000F

Port A

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

Port A Data Register

PORTA contains the data latches for the eight port A pins.

Address: $0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Reset:	Unaffected by Reset							

Figure 2. Port A Data Register (PORTA)

PA7–PA0 — 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.

Data Direction Register A

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

Address: $0004

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

Figure 3. Data Direction Register A (DDRA)

DDRA7–DDRA0 — Data Direction Register A Bits

These read/write bits control port A data direction. Reset clears DDRA7–DDRA0, 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 4 shows the port A I/O logic.

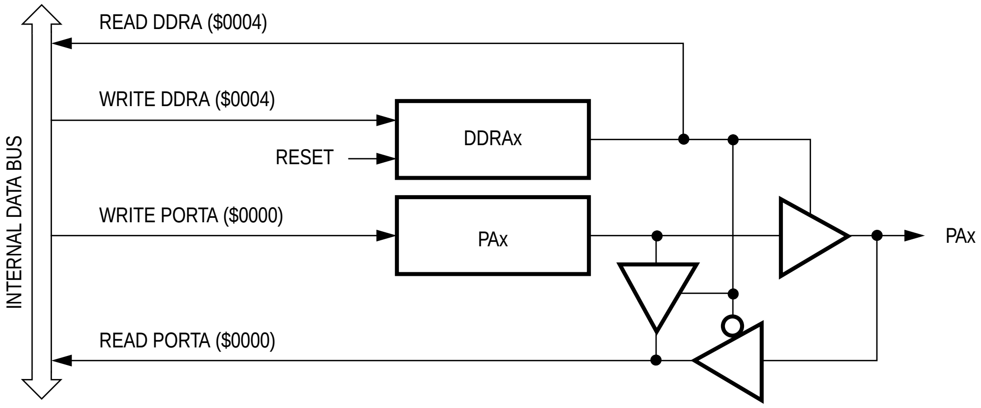


Figure 4. Port A I/O Circuit

When bit DDRAx is a logic 1, reading address $0000 reads the PAx 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 2** summarizes the operation of the port A pins.

Table 2. Port A Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	Write
0	Input, high-impedance	Pin	Latch[1]
1	Output	Latch	Latch

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

Port B

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

Port B Data Register

PORTB contains the data latches for the eight port B pins.

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Reset:	Unaffected by Reset							

Figure 5. Port B Data Register (PORTB)

PB7–PB0 — 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.

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 6. Data Direction Register B (DDRB)

DDRB7–DDRB0 — Data Direction Register B Bits

These read/write bits control port B data direction. Reset clears DDRB7–DDRB0, 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 7 shows the port B I/O logic.

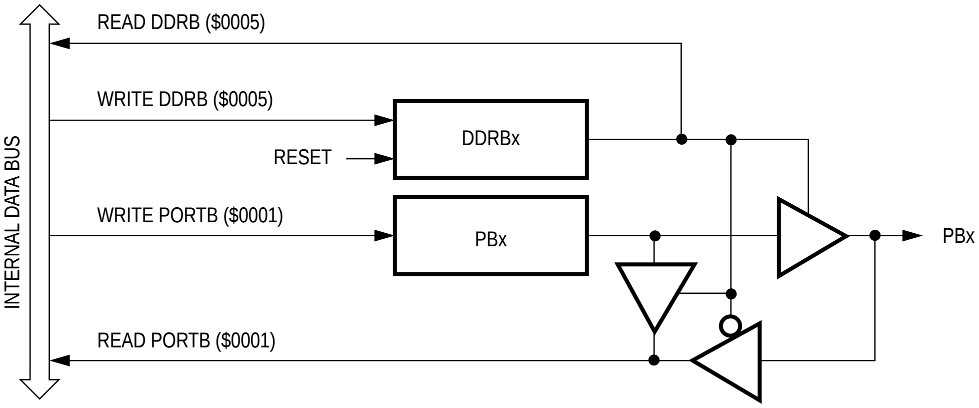


Figure 7. Port B I/O Circuit

When bit DDRBx is a logic 1, reading address $0001 reads the PBx data latch. When bit DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 3** summarizes the operation of the port B pins.

Table 3. Port B Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	Write
0	Input, high-impedance	Pin	Latch[(1)]
1	Output	Latch	Latch

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

Port C

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

Port C Data Register

PORTC contains the data latches for the eight port C pins.

Address: $0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
Reset:	Unaffected by Reset							

Figure 8. Port C Data Register (PORTC)

PC7–PC0 — 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.

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: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC	DDRC0
Reset:	0	0	0	0	0	0	0	0

Figure 9. Data Direction Register C (DDRC)

DDRC7–DDRC0 — Data Direction Register C Bits

These read/write bits control port C data direction. Reset clears DDRC7–DDRC0, 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 10 shows the port C I/O logic.

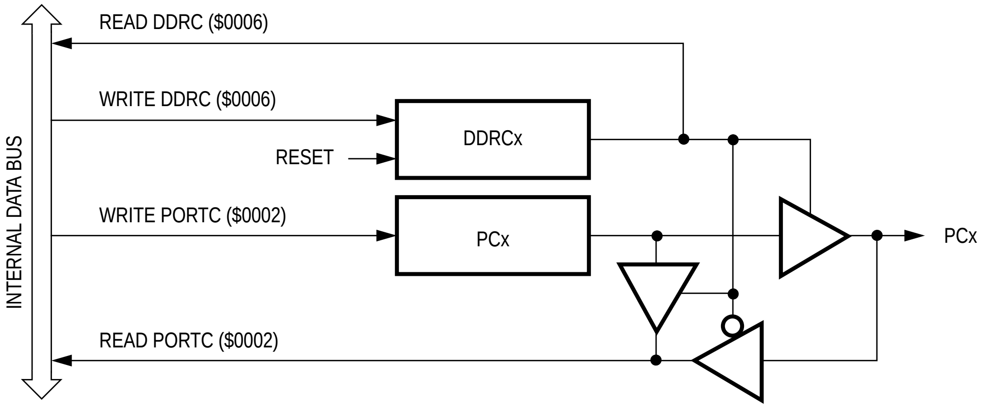


Figure 10. Port C I/O Circuit

When bit DDRCx is a logic 1, reading address $0002 reads the PCx 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 4** summarizes the operation of the port C pins.

Table 4. Port C Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	Write
0	Input, high-impedance	Pin	Latch[1]
1	Output	Latch	Latch

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

Port D

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

Port D Data Register

PORTD contains the data latches for the eight port D pins.

Address: $0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
Reset:	Unaffected by Reset							

Figure 11. Port D Data Register (PORTD)

PD7–PD0 — Port D Data Bits

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

The keyboard interrupt enable bits, KBIE7—KBIE0, in the keyboard interrupt control register (KBICR), enable the port D pins as external interrupt pins. (See External Interrupt Module (IRQ) on page 299.)

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: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Reset:	0	0	0	0	0	0	0	0

Figure 12. Data Direction Register D (DDRD)

DDRD7–DDRD0 — Data Direction Register D Bits

These read/write bits control port D data direction. Reset clears DDRD7–DDRD0, configuring all port D pins as inputs.

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

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

Figure 13 shows the port D I/O logic.

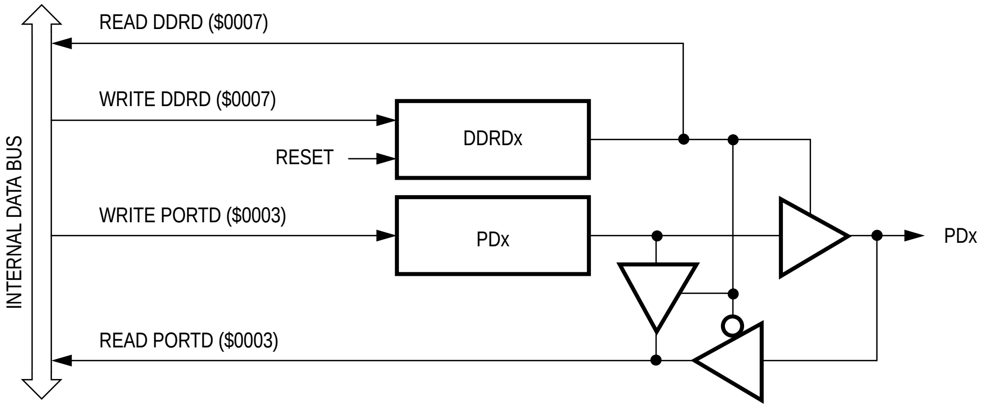


Figure 13. Port D I/O Circuit

When bit DDRDx is a logic 1, reading address $0003 reads the PDx 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 5** summarizes the operation of the port D pins.

Table 5. Port D Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	Write
0	Input, high-impedance	Pin	Latch(1)
1	Output	Latch	Latch

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

Port E

Port E is an 8-bit special function port that shares five of its pins with the timer interface module (TIM) and two of its pins with the serial communications interface module (SCI).

Port E Data Register

PORTE contains the data latches for the eight port E pins.

Address: $0008

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
Reset:	Unaffected by Reset							
Alternate Function:	TCH3	TCH2	TCH1	TCH0	TCLK	TxD	RxD	

Figure 14. Port E Data Register (PORTE)

PE7—PE0 — Port E Data Bits

PE7—PE0 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.

TCH3–TCH0 — Timer Channel I/O Bits

The PE7/TCH3–PE4/TCH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB and ELSxA, determine whether the PE7/TCH3–PE4/TCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See Timer Interface Module (TIM) on page 163.)

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 6. Port E Pin Operation.)*

TCLK — Timer Clock Input

The PE3/TCLK pin is the external clock input for the TIM. The prescaler select bits, PS2–PS0, select PE3/TCLK as the TIM clock input. (See Timer Interface Module (TIM) on page 163.) When not selected as the TIM clock, PE3/TCLK is available for general-purpose I/O.

TxD — SCI Transmit Data Output

The PE2/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 PE2/TxD pin is available for general-purpose I/O. (See Serial Communications Interface Module (SCI) on page 227.)

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 6. Port E Pin Operation.)*

RxD — SCI Receive Data Input

The PE1/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 PE1/RxD pin is available for general-purpose I/O. (See Serial Communications Interface Module (SCI) on page 227.)

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:	DDR37	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDTE1	DDRE0
Reset:	0	0	0	0	0	0	0	0

Figure 15. Data Direction Register E (DDRE)

DDRE7–DDRE0 — Data Direction Register E Bits

These read/write bits control port E data direction. Reset clears DDRE7–DDRE0, 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 shows the port E I/O logic.

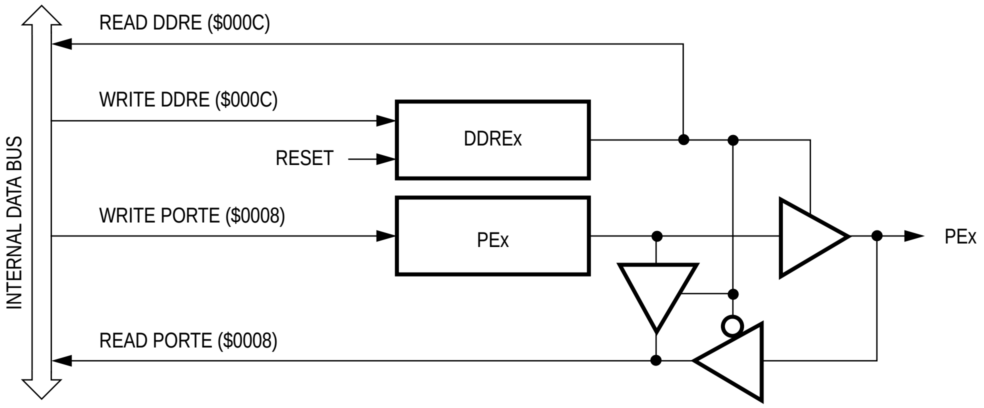


Figure 16. Port E I/O Circuit

When bit DDREx is a logic 1, reading address $0008 reads the PEx 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 6** summarizes the operation of the port E pins.

Table 6. Port E Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	**Write**
0	Input, high-impedance	Pin	Latch[1]
1	Output	Latch	Latch

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

Port F

PORTF is a 6-bit, special function port that shares four of its pins with the serial peripheral interface module (SPI).

Port F Data Register

PORTF contains the data latches for the six port F pins.

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	PF5	PF4	PF3	PF2	PF1	PF0
Write:	▨	▨						
Reset:	Unaffected by Reset							
Alternate Function:					MISO	MOSI	SPSCK	$\overline{\text{SS}}$

▨ = Unimplemented

Figure 17. Port F Data Register (PORTF)

PF5–PF0 — 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 PF5–PF0.

MISO — Master In/Slave Out

The PF3/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 PF3/MISO pin is available for general-purpose I/O. (See Serial Peripheral Interface Module (SPI) on page 191.)

NOTE: *Data direction register F (DDRF) does not affect the data direction of port F pins that are being used by the SPI module. However, the DDRF bits always determine whether reading port F returns the states of the latches or the states of the pins. (See Table 7. Port F Pin Operation.)*

MOSI — Master Out/Slave In

The PF2/MOSI pin is the master out/slave in terminal of the SPI module. When the SPE bit is clear, the PPF2/MOSI pin is available for general-purpose I/O. (See Serial Peripheral Interface Module (SPI) on page 191.)

SPSCK — SPI Serial Clock

The PF1/SPSCK pin is the serial clock input of the SPI module. When the SPE bit is clear, the PF1/SPSCK pin is available for general-purpose I/O.

$\overline{\text{SS}}$ — Slave Select

The PF0/$\overline{\text{SS}}$ 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, the PF0/$\overline{\text{SS}}$ pin is available for general-purpose I/O. (See Serial Peripheral Interface Module (SPI) on page 191.) When the SPI is enabled, the DDRF0 bit in data direction register F (DDRF) has no effect on the PF0/$\overline{\text{SS}}$ pin.

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:			DDRF5	DDRF4	DDRF3	DDRF2	DDRF1	DDRF0
Write:	▨	▨						
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 18. Data Direction Register F (DDRF)

DDRF5—DDRF0 — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF5—DDRF0, 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 19 shows the port F I/O logic.

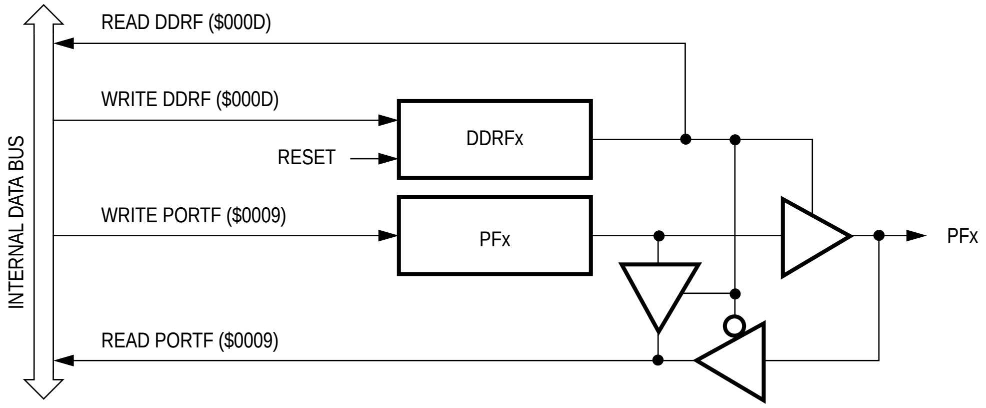


Figure 19. Port F I/O Circuit

When bit DDRFx is a logic 1, reading address $0009 reads the PFx 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 7 summarizes the operation of the port F pins.

Table 7. Port F Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	**Write**
0	Input, high-impedance	Pin	Latch][1]
1	Output	Latch	Latch

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

Port G

Port G is a 4-bit, general-purpose bidirectional I/O port.

NOTE: *Port G is available only on the 64-pin QFP.*

Port G Data Register

PORTG contains the data latches for the four port G pins.

Address: $000A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	PG3	PG2	PG1	PG0
Write:	▨	▨	▨	▨				
Reset:	Unaffected by Reset							

▨ = Unimplemented

Figure 20. Port G Data Register (PORTG)

PG3–PG0 — Port G Data Bits

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

Data Direction Register G

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

Address: $000E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	DDRG3	DDRG2	DDRG1	DDRG0
Write:	▨	▨	▨	▨				
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 21. Data Direction Register G (DDRG)

DDRG3–DDRG0 — Data Direction Register G Bits

These read/write bits control port G data direction. Reset clears DDRG3–DDRG0, configuring all port G pins as inputs.

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

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

Figure 22 shows the port G I/O logic.

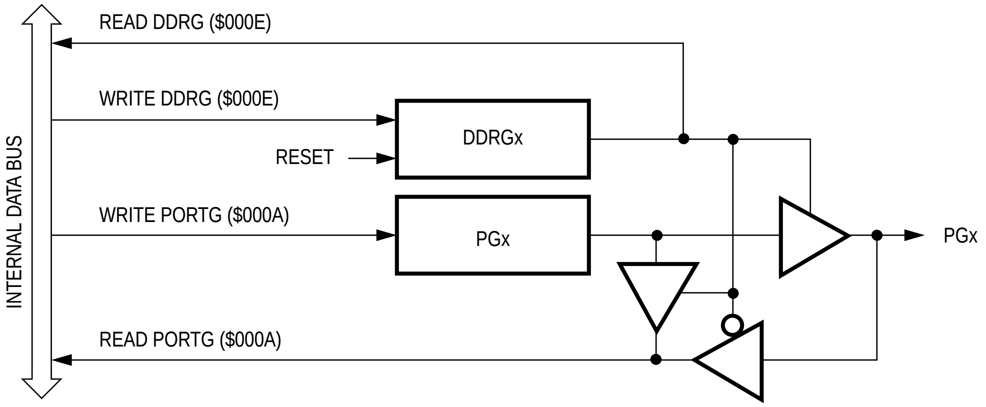


Figure 22. Port G I/O Circuit

When bit DDRGx is a logic 1, reading address $000A reads the PGx data latch. When bit DDRGx is a logic 0, reading address $000A reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 8 summarizes the operation of the port G pins.

Table 8. Port G Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	Write
0	Input, high-impedance	Pin	Latch][1]
1	Output	Latch	Latch

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

Port H

Port H is a 4-bit, general-purpose bidirectional I/O port.

NOTE: *Port H is available only on the 64-pin QFP.*

Port H Data Register

PORTH contains the latches for the four port H pins.

Address: $000B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	PH3	PH2	PH1	PH0
Write:	▨	▨	▨	▨				
Reset:	Unaffected by Reset							

▨ = Unimplemented

Figure 23. Port H Data Register (PORTH)

PH3–PH0 — Port H Data Bits

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

Data Direction Register H

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

Address: $000F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	DDRH3	DDRH2	DDRH1	DDRH0
Write:	▨	▨	▨	▨				
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 24. Data Direction Register H (DDRH)

DDRH3–DDRH0 — Data Direction Register H Bits

These read/write bits control port H data direction. Reset clears DDRH3–DDRH0, configuring all port H pins as inputs.

1 = Corresponding port H pin configured as output

0 = Corresponding port H pin configured as input

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

Figure 25 shows the port H I/O logic.

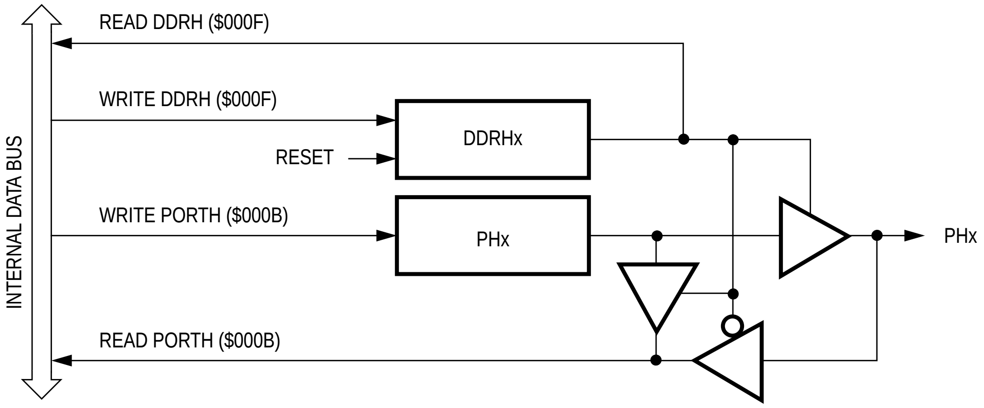


Figure 25. Port H I/O Circuit

When bit DDRHx is a logic 1, reading address $000B reads the PHx data latch. When bit DDRHx is a logic 0, reading address $000B reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. **Table 9** summarizes the operation of the port H pins.

Table 9. Port H Pin Operation

Data Direction Bit	I/O Pin Mode	Access to Data Bit	
		Read	Write
0	Input, high-impedance	Pin	Latch[1]
1	Output	Latch	Latch

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

Computer Operating Properly Module (COP)

Contents

Introduction

The COP module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by periodically clearing the COP counter.

Functional Description

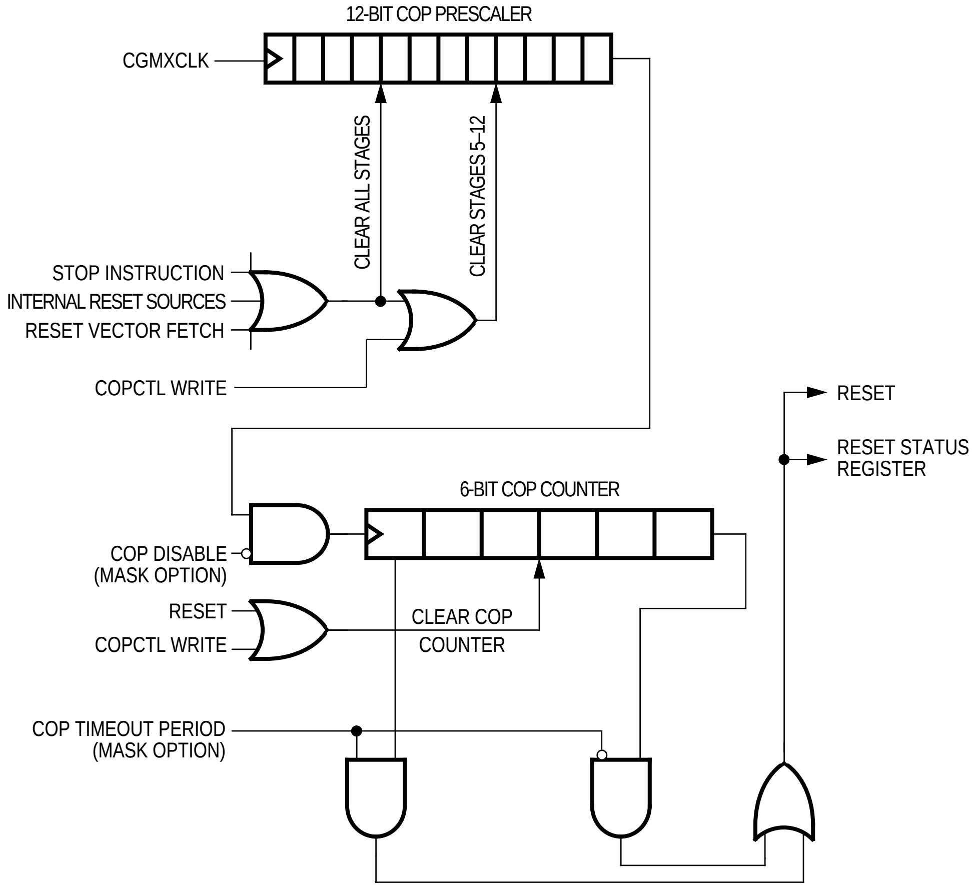


Figure 1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler. If not cleared by software, the COP counter overflows and generates an asynchronous reset after $2^{13} – 2^4$ or $2^{18} – 2^4$ CGMXCLK cycles, depending on the mask option selected for COP timeout period. With the $2^{18} – 2^4$ CGMXCLK cycle overflow option, a 4.9152-MHz crystal gives a COP timeout period of 53.3 ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 5 through 12 of the prescaler.

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

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

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

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

I/O Signals

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

CGMXCLK

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

STOP Instruction

The STOP instruction clears the COP prescaler.

COPCTL Write

Writing any value to the COP control register (COPCTL) (see COP Control Register on page 296) clears the COP counter and clears stages 12 through 5 of the COP prescaler. Reading the COP control register returns the reset vector.

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

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

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

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 2. COP Control Register (COPCTL)

Interrupts

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

Low-Power Modes

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

Wait Mode

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

Stop Mode

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

To prevent inadvertently turning off the COP with a STOP instruction, a mask option is available that disables the STOP instruction.

COP Module During Monitor Mode

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

COP Module During Break Interrupts

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

External Interrupt Module (IRQ)

Contents

Introduction

The IRQ module provides two independently maskable external interrupt pins.

Features

Features of the IRQ module include the following:

- Two Dedicated External Interrupt Pins with Separate External Interrupt Masks
- Hysteresis Buffers
- Programmable Edge-Only or Edge- and Level- Interrupt Sensitivity
- Automatic Interrupt Acknowledge
- Exit from Low-Power Modes

Functional Description

A logic 0 applied to any of the external interrupt pins can latch a CPU interrupt request. **Figure 1** shows the structure of the IRQ module.

Interrupt signals on the $\overline{\text{IRQ1}}$ pin are latched separately from interrupt signals on the $\overline{\text{IRQ2}}$ pin. CPU interrupt requests remain latched until one of the following actions occurs:

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the CPU interrupt request that caused the vector fetch.
- Software clear — Software can clear a latched CPU interrupt request by writing to the appropriate acknowledge bit in the IRQ status and control register (ISCR). Writing a logic 1 to the ACK1 bit clears the $\overline{\text{IRQ1}}$ CPU interrupt request. Writing a logic 1 to the ACK2 bit clears the $\overline{\text{IRQ2}}$ CPU interrupt request.
- Reset — A reset automatically clears both $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ CPU interrupt requests.

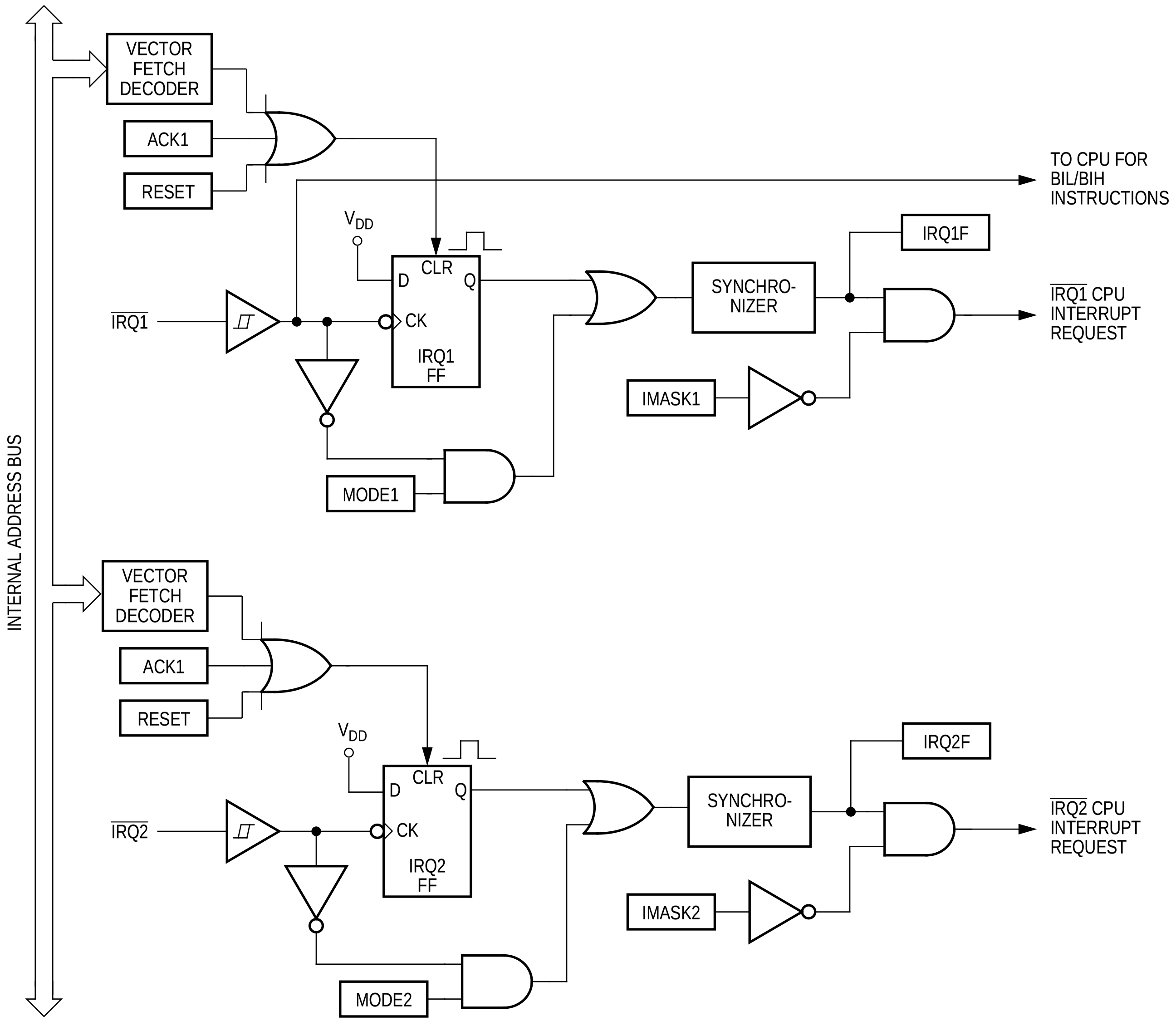


Figure 1. IRQ Module Block Diagram

3-intirq2_a

All of the external interrupt pins are falling-edge-triggered and are software-configurable to be both falling-edge and low-level-triggered. The MODE1 bit in the ISCR controls the triggering sensitivity of the $\overline{\text{IRQ1}}$ pin. The MODE2 bit controls the triggering sensitivity of the $\overline{\text{IRQ2}}$ pin.

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

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

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

The vector fetch or software clear can occur before or after the interrupt pin returns to logic 1. As long as the pin is low, the CPU interrupt request remains pending. A reset clears the CPU interrupt request and the MODEx control bit even if the pin stays low.

When set, the IMASK1 and IMASK2 bits in the ISCR mask all external interrupt requests. A latched CPU 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 CPU interrupt requests, including external interrupt requests.*

$\overline{\text{IRQ1}}$ Pin

A logic 0 on the $\overline{\text{IRQ1}}$ pin can latch a CPU interrupt request. A vector fetch, software clear, or reset clears the $\overline{\text{IRQ1}}$ CPU interrupt request.

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

- Vector fetch, software clear, or reset — A vector fetch generates an interrupt acknowledge signal to clear the CPU interrupt request. Software can generate the interrupt acknowledge signal by writing a logic 1 to the ACK1 bit in the IRQ status and control register (ISCR). The ACK1 bit is useful in applications that poll the $\overline{\text{IRQ1}}$ pin and require software to clear the $\overline{\text{IRQ1}}$ CPU interrupt request. Writing to the ACK1 bit before leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK1 does not affect subsequent transitions on the $\overline{\text{IRQ1}}$ pin. A falling edge that occurs after writing to the ACK1 bit latches another CPU interrupt request. If the $\overline{\text{IRQ1}}$ mask bit, IMASK1, is clear, the CPU loads the program counter with the vector address at locations $FFFA and $FFFB.
- Return of the $\overline{\text{IRQ1}}$ pin to logic 1 — As long as the $\overline{\text{IRQ1}}$ pin is at logic 0, the $\overline{\text{IRQ1}}$ CPU interrupt request remains latched.

The vector fetch, software clear, or reset and the return of the $\overline{\text{IRQ1}}$ pin to logic 1 can occur in any order. A reset clears the CPU interrupt request and the MODE1 bit, clearing the CPU interrupt request even if the pin stays low.

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

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

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

NOTE: *To avoid spurious CPU interrupts caused by noise, mask CPU interrupt requests in the interrupt routine when using the level-sensitive interrupt trigger.*

IRQ2 Pin

A logic 0 on the IRQ2 pin can latch a CPU interrupt request. A vector fetch, software clear, or reset clears the IRQ2 CPU interrupt request.

If the MODE2 bit is set, the IRQ2 pin is both falling-edge-sensitive and low-level-sensitive. With MODE2 set, both of the following actions must occur to clear an IRQ2 CPU interrupt request:

- Vector fetch, software clear, or reset — A vector fetch generates an interrupt acknowledge signal to clear the CPU interrupt request. Software can generate the interrupt acknowledge signal by writing a logic 1 to the ACK2 bit in the IRQ status and control register (ISCR). The ACK2 bit is useful in applications that poll the IRQ2 pin and require software to clear the IRQ2 CPU interrupt request. Writing to the ACK2 bit before leaving an interrupt service routine can also prevent spurious CPU interrupts due to noise. Setting ACK2 does not affect subsequent transitions on the IRQ2 pin. A falling edge that occurs after writing to the ACK2 bit latches another CPU interrupt request. If the IRQ2 mask bit, IMASK2, is clear, the CPU loads the program counter with the vector address at locations $FFE0 and $FFE1.
- Return of the IRQ2 pin to logic 1 — As long as the IRQ2 pin is at logic 0, the IRQ2 CPU interrupt request remains latched.

The vector fetch, software clear, or reset and the return of the IRQ2 pin to logic 1 can occur in any order. A reset clears the CPU interrupt request and the MODE2 bit, clearing the CPU interrupt request even if the pin stays low.

If the MODE2 bit is clear, the IRQ2 pin is falling-edge-sensitive only. With MODE2 clear, a vector fetch or software clear immediately clears the IRQ2 CPU interrupt request.

The IRQF2 bit in the ISCR register can be used to check for pending CPU interrupts. The IRQF2 bit is not affected by the IMASK2 bit, which makes it useful in applications where polling is preferred.

There is no direct way to determine the logic level on the $\overline{\text{IRQ2}}$ pin. However, it is possible to use the IRQF2 bit in the ISCR to infer the state of the $\overline{\text{IRQ2}}$ pin. If the MODE2 bit is a logic 1, the IRQF2 bit in the ISCR is the opposite value of the $\overline{\text{IRQ2}}$ pin as long as the $\overline{\text{IRQ2}}$ CPU interrupt request is cleared. (See **Figure 1**.) Clear the $\overline{\text{IRQ2}}$ CPU interrupt request by writing a logic 1 to the acknowledge bit. Recall, however, that every falling edge on the $\overline{\text{IRQ2}}$ pin latches an $\overline{\text{IRQ2}}$ CPU interrupt request. So an additional acknowledge is necessary after each falling edge on $\overline{\text{IRQ2}}$ to maintain the opposite relationship between IRQF2 and the $\overline{\text{IRQ2}}$ pin. Set the IMASK2 bit in the ISCR to prevent the IRQF2 from generating CPU interrupts when used in this manner.

NOTE: *To avoid spurious CPU interrupts caused by noise, mask CPU interrupt requests in the interrupt routine when using the level-sensitive interrupt trigger.*

Low-Power Modes

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

Wait Mode

The IRQ module remains active in wait mode. Clearing the IMASK1 or IMASK2 bit in the IRQ status and control register enables $\overline{\text{IRQ1}}$ or $\overline{\text{IRQ2}}$ CPU interrupt requests to bring the MCU out of wait mode.

Stop Mode

The IRQ module remains active in stop mode. Clearing the IMASK1 or IMASK2 bit in the IRQ status and control register enables $\overline{\text{IRQ1}}$ or $\overline{\text{IRQ2}}$ CPU interrupt requests to bring the MCU out of stop mode.

IRQ Module During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear CPU interrupt requests during the break state. (See Break Module (BRK) on page 141.)

To allow software to clear $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ CPU interrupt requests during a break interrupt, write a logic 1 to the BCFE bit. If a CPU interrupt request is cleared during the break state, it remains cleared when the MCU exits the break state.

To protect CPU interrupt flags during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), writing to the ACK1 and ACK2 bits in the IRQ status and control register during the break state has no effect on the IRQ interrupt flags.

IRQ Status and Control Register

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

- Shows the state of the $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ interrupt flags
- Clears $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ CPU interrupt flags
- Masks $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ CPU interrupt requests
- Controls triggering sensitivity of the $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ CPU interrupt pins

Address: $0032

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IRQF2	0	IMASK2	MODE2	IRQF1	0	IMASK1	MODE1
Write:	▨	ACK2			▨	ACK1		
Reset:								

▨ = Unimplemented

Figure 2. IRQ Status and Control Register (ISCR)

IRQ2F — $\overline{\text{IRQ2}}$ Flag

This read-only bit is high when an $\overline{\text{IRQ2}}$ CPU interrupt is pending. Reset clears IRQ2F.

1 = $\overline{\text{IRQ2}}$ CPU interrupt pending
0 = $\overline{\text{IRQ2}}$ CPU interrupt not pending

ACK2 — $\overline{\text{IRQ2}}$ Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the $\overline{\text{IRQ2}}$ CPU interrupt request. ACK2 always reads as logic 0. Reset clears ACK2.

IMASK2 — $\overline{\text{IRQ2}}$ Interrupt Mask Bit

Writing a logic 1 to this read/write bit disables $\overline{\text{IRQ2}}$ CPU interrupt requests. Reset clears IMASK2.

1 = $\overline{\text{IRQ2}}$ CPU interrupt requests masked
0 = $\overline{\text{IRQ2}}$ CPU interrupt requests not masked

MODE2 — $\overline{\text{IRQ2}}$ Pin Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ2}}$ pin. Reset clears MODE2.

1 = $\overline{\text{IRQ2}}$ CPU interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ2}}$ CPU interrupt requests on falling edges only

IRQ1F — $\overline{\text{IRQ1}}$ Flag

This read-only bit is high when an $\overline{\text{IRQ1}}$ CPU interrupt is pending.

1 = $\overline{\text{IRQ1}}$ CPU interrupt pending
0 = $\overline{\text{IRQ1}}$ CPU interrupt not pending

ACK1 — $\overline{\text{IRQ1}}$ Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the $\overline{\text{IRQ1}}$ CPU interrupt request. ACK1 always reads as logic 0. Reset clears ACK1.

IMASK1 — $\overline{\text{IRQ1}}$ Interrupt Mask Bit

Writing a logic 1 to this read/write bit disables $\overline{\text{IRQ1}}$ CPU interrupt requests. Reset clears IMASK1.

1 = $\overline{\text{IRQ1}}$ CPU interrupt requests masked
0 = $\overline{\text{IRQ1}}$ CPU interrupt requests not masked

MODE1 — $\overline{\text{IRQ1}}$ Pin Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ1}}$ pin. Reset clears MODE1.

1 = $\overline{\text{IRQ1}}$ CPU interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ1}}$ CPU interrupt requests on falling edges only

Keyboard Interrupt Module (KBI)

Contents

Introduction

The keyboard interrupt module provides eight independently maskable external interrupt pins.

Features

- Eight Keyboard Interrupt Pins with Separate Keyboard Interrupt Enable Bits and One Keyboard Interrupt Mask
- Hysteresis Buffers
- Programmable Edge-Only or Edge- and Level- Interrupt Sensitivity
- Automatic Interrupt Acknowledge
- Exit from Low-Power Modes

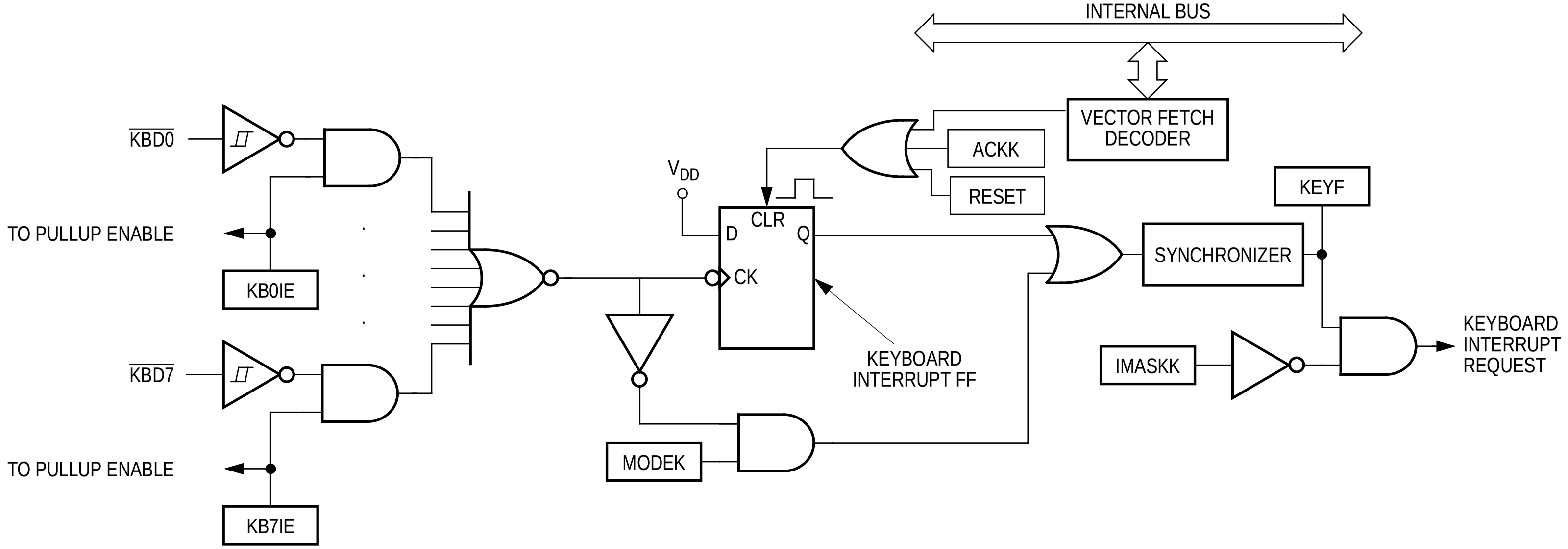


Figure 1. KBI Block Diagram

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
	Write:	▨	▨	▨	▨	▨	ACKK		
	Reset:	0	0	0	0	0	0	0	0
Keyboard Interrupt Enable Register (KBIER)	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
	Write:								
	Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 2. KBI I/O Register Summary

Table 1. KBI I/O Register Address Summary

Register	KBSCR	KBIER
Address	$001A	$001B

Functional Description

Writing to the KBIE7–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port D pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

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

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

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

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software can generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register. The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit in an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations $FFDE and $FFDF.

- Return of all enabled keyboard interrupt pins to logic 1 — As long as any enabled keyboard interrupt pin is at logic 0, the keyboard interrupt request remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic 1 may occur in any order.

If the MODEK bit is clear, the keyboard interrupt pin is falling-edge-sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

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

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

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

Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup to reach a logic 1. Therefore a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.
3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDRD bits in data direction register D.
2. Write logic 1s to the appropriate port D data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

Low-Power Modes

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

Wait Mode

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

Stop Mode

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

KBI During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See **Break Module (BRK)** on page 141.)

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

To protect the KEYF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0, writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. (See **Keyboard Status and Control Register** on page 315.)

I/O Registers

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

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

Keyboard Status and Control Register

The keyboard status and control register:

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

Address: $001A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
Write:	▨	▨	▨	▨	▨	ACKK		
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 3. Keyboard Status and Control Register (KBSCR)

Bits 7–4 — Not used

These read-only bits always read as logic 0s.

KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYF bit.

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

Writing a logic 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as logic 0. Reset clears ACKK.

IMASKK — Keyboard Interrupt Mask Bit

Writing a logic 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKK bit.

1 = Keyboard interrupt requests masked
0 = Keyboard interrupt requests not masked

MODEK — Keyboard Triggering Sensitivity Bit

This read/write bit controls the triggering sensitivity of the keyboard interrupt pins. Reset clears MODEK.

1 = Keyboard interrupt requests on falling edges and low levels
0 = Keyboard interrupt requests on falling edges only

Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables each port D pin to operate as a keyboard interrupt pin.

Address: $001B

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Reset:	0	0	0	0	0	0	0	0

Figure 4. Keyboard Interrupt Enable Register (KBIER)

KBIE7–KBIE0 — Keyboard Interrupt Enable Bits

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

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

Low-Voltage Inhibit Module (LVI)

Contents

Introduction

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

Features

Features of the LVI module include the following:

- Programmable LVI Reset
- Programmable Power Consumption
- Programmable stop mode operation

Functional Description

Figure 1 shows the structure of the LVI module. The LVI module contains a bandgap reference circuit and comparator. LVI operation, LVI resets, and LVI operation in stop mode are mask options.

Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, $V_{LVR} + H_{LVR}$. A power-on reset occurs when V_{DD} reaches $V_{LVR} + H_{LVR}$.The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

An LVI reset also drives the RST pin low to provide low-voltage protection to external peripheral devices.

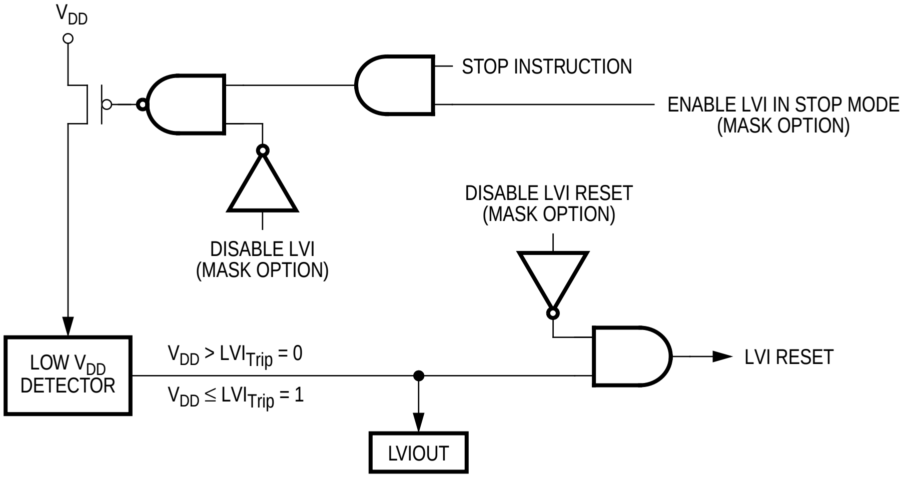


Figure 1. LVI Module Block Diagram

Polled LVI Operation

In applications that can operate at V_{DD} levels below the V_{LVR} level, disabling LVI resets allows software to monitor V_{DD} by polling the LVIOUT bit.

Forced Reset Operation

In applications that require V_{DD} to remain above the V_{LVR} trip level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls to the V_{LVR} level.

LVI Status Register

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

Address: $FE0F

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

▨ = Unimplemented

Figure 2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the V_{LVR} trip voltage. (See **Table 1**.) Reset clears the LVIOUT bit.

Table 1. LVIOUT Bit Indication

V_{DD}	LVIOUT
$V_{DD} > V_{LVR} + H_{LVR}$	0
$V_{DD} < V_{LVR}$	1
$V_{LVR} < V_{DD} < V_{LVR} + H_{LVR}$	Previous Value

LVI Interrupts

The LVI module does not generate CPU interrupt requests.

Low-Power Modes

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

Wait Mode

If enabled, the LVI module remains active in wait mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of wait mode.

Stop Mode

If enabled, the LVI module remains active in stop mode. If enabled to generate resets, the LVI module can generate a reset and bring the MCU out of stop mode.

Specifications

Contents

Preliminary Electrical Specifications

These electrical and timing specifications are design targets and have not been fully characterized.

Absolute Maximum Ratings

Absolute maximum ratings are the extreme limits the device can be exposed to without causing permanent damage, and are not intended to indicate functional operating range.

Table 1. Absolute Maximum Ratings[1]

Characteristic	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

[1] Voltages referenced to V_{SS}.

Functional Operating Range

Table 2. Operating Range

Characteristic	Symbol	Value	Unit
Operating Temperature Range	T_A	– 40 to + 85	°C
Operating Voltage Range	V_{DD}	2.0 ± 10% 3.0 ± 10% 5.0 ± 10%	V

Thermal Characteristics

Table 3. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal Resistance QFP(64 pin) SDIP (56 pin)	θ_{JA}	85 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}	125	°C

[1] Power dissipation is a function of temperature.

[2] K is a constant unique to the device. K can be determined for a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

DC Electrical Characteristics

Table 4. DC Electrical Characteristics (V_{DD} = 5.0 Vdc ± 10%) [1]

Characteristic	Symbol	Min	Typ[2]	Max	Unit
Output High Voltage (I_{Load} = – 2.0 mA) All I/O Pins	V_{OH}	V_{DD} – 0.8	—	—	V
Output Low Voltage (I_{Load} = 1.6 mA) All I/O Pins	V_{OL}	—	—	0.4	V
Input High Voltage All Ports, IRQs, RESET, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input Low Voltage All Ports, IRQs, RESET, OSC1	V_{IL}	V_{SS}	—	0.2 x V_{DD}	V
V_{DD} Supply Current	I_{DD}				
Run[3]		—	—	30	mA
Wait[4]		—	—	12	mA
Stop[5]					
25 °C		—	—	5	µA
0 °C to 85 °C		—	—	15	µA
25 °C with LVI Enabled		—	—	320	µA
0 °C to 85 °C with LVI Enabled		—	—	380	µA
I/O Ports Hi-Z Leakage Current	I_{IL}	—	—	± 10	µA
Input Current	I_{IN}	—	—	1	µA
Capacitance Ports (as Input or Output)	C_{Out} C_{In}	— —	— —	12 8	pF pF
Low Voltage Reset Inhibit	V_{LVR}	2.6	2.7	2.8	V
Low Voltage Reset Inhibit/Recover Hysteresis	H_{LVR}	60	80	100	mV
POR ReArm Voltage[6]	V_{POR}	0	—	100	mV
POR Reset Voltage[7]	V_{PORRST}	0	700	800	mV
POR Rise Time Ramp Rate[8]	R_{POR}	0.035	—	—	V/ms

[1] V_{DD} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted.

[2] Typical values reflect average measurements at midpoint of voltage range, 25°C only.

[3] Run (operating) I_{DD} measured using external square wave clock source (f_{osc} = 32.8 MHz). All inputs 0.2V 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_{osc} = 32.8 MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects Wait I_{DD}. Measured with PLL and LVI enabled.

[5] Stop I_{DD} 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 power-on reset is released, $\overline{\text{RST}}$ must be driven low externally until minimum V_{DD} is reached.

Table 5. DC Electrical Characteristics (V_{DD} =3.0 Vdc ± 10%) [1]

Characteristic	Symbol	Min	Typ[2]	Max	Unit
Output High Voltage (I_{Load} = – 1.0 mA) All Ports	V_{OH}	V_{DD} – 0.8	—	—	V
Output Low Voltage (I_{Load} = 0.8 mA) All Ports	V_{OL}	—	—	0.4	V
Input High Voltage All ports, IRQs, RESET, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input Low Voltage All ports, IRQs, RESET, OSC1	V_{IL}	V_{SS}	—	0.2 x V_{DD}	V
V_{DD} Supply Current			—		
Run[3]		—	—	10	mA
Wait[4]		—	—	6	mA
Stop[5]	I_{DD}				
25°C		—	—	3	µA
0° to 85°		—	—	10	µA
25°C with LVI Enabled		—	—	200	µA
0° to 85° with LVI Enabled		—	—	250	µA
I/O Ports Hi-Z Leakage Current	I_{IL}	—	—	± 10	µA
Input Current	I_{IN}	—	—	1	µA
Capacitance Ports (as Input or Output)	C_{OUT} C_{IN}	— —	— —	12 8	pF pF
Low Voltage Reset Inhibit	V_{LVII}	2.6	2.7	2.8	V
Low Voltage Reset Inhibit/Recover Hysteresis	H_{LVI}	60	80	100	mV
POR ReArm Voltage[6]	V_{POR}	0	—	200	mV
POR Reset Voltage[7]	V_{PORRST}	0	700	800	mV
POR Rise Time Ramp Rate[8]	R_{POR}	0.02	—	—	V/ms

[1] V_{DD} = 3.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted.

[2] Typical values reflect average measurements at midpoint of voltage range, 25°C only.

[3] Run (operating) I_{DD} measured using external square wave clock source (f_{osc} = 16.4 MHz). All inputs 0.2V 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_{osc} = 16.4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects Wait I_{DD}. Measured with PLL and LVI enabled.

[5] Stop I_{DD} 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 poer-on reset is released, $\overline{RST}$ must be driven low externally until minimum V_{DD} is reached.

Table 6. DC Electrical Characteristics (V_{DD} = 2.0 Vdc ± 10%)[1]

Characteristic	Symbol	Min	Typ[2]	Max	Unit
Output High Voltage (I_{LOAD} = – 0.5 mA) all ports	V_{OH}	V_{DD} – 0.4	—	—	V
Output Low Voltage (I_{LOAD} = 0.4 mA) all ports	V_{OL}	—	—	0.2	V
Input High Voltage All ports, IRQs, RESET, OSC1	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V
Input Low Voltage All ports, IRQs, RESET, OSC1	V_{IL}	V_{SS}	—	0.2 x V_{DD}	V
V_{DD} Supply Current	I_{DD}		—		
Run [3]		—	—	6	mA
Wait [4]		—	—	3	mA
Stop [5]					
25°C		—	—	1	µA
0° to 85°		—	—	2	µA
I/O Ports Hi-Z Leakage Current	I_{IL}	—	—	± 10	µA
Input Current	I_{IN}	—	—	1	µA
Capacitance Ports (as Input or Output)	C_{OUT} C_{IN}	— —	— —	12 8	pF pF
POR ReArm Voltage[6]*	V_{POR}	0	—	200	mV
POR Reset Voltage[7]*	V_{PORRST}	0	700	800	mV
POR Rise Time Ramp Rate[8]	R_{POR}	0.02	—	—	V/ms

1. V_{DD} = 2.0 Vdc ± 10%, V_{SS} = 0 Vdc, T_A = T_L to T_H, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25°C only.
3. Run (operating) I_{DD} measured using external square wave clock source (f_{osc} = 8.4 MHz). All inputs 0.2V 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_{osc} = 8.4 MHz). All inputs 0.2V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects Wait I_{DD}. Measured with PLL 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, $\overline{RST}$ must be driven low externally until minimum V_{DD} is reached.

Control Timing

Table 7. Control Timing (V_{DD} = 5.0 Vdc ± 10%)[1]

Characteristic	Symbol	Min	Max	Unit
Frequency of Operation[2] Crystal Option External Clock Option[3]	f_{OSC}	1 dc[4]	8 32.8	MHz MHz
Internal Operating Frequency	f_{op}	—	8.2	MHz
$\overline{\text{RESET}}$ Input Pulse Width Low[5]	t_{IRL}	50	—	ns
$\overline{\text{IRQ}}$ Interrupt Pulse Width Low[6] (Edge-Triggered)	t_{ILIH}	50	—	ns

1. V_{SS} = 0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{DD} unless noted
2. See **Table 14** and **Table 15** for more information.
3. No more than 10% duty cycle deviation from 50%
4. Some modules may require a minimum frequency greater than dc for proper operation; see appropriate table for this information.
5. Minimum pulse width reset is guaranteed to be recognized; it is possible for a smaller pulse width to cause a reset.
6. Minimum pulse width is for guaranteed interrupt; it is possible for a smaller pulse width to be recognized.

Table 8. Control Timing (V_{DD} = 3.0 Vdc ± 10%)[1]

Characteristic	Symbol	Min	Max	Unit
Frequency of Operation[2] Crystal Option External Clock Option[3]	f_{OSC}	1 dc[4]	8 16.4	MHz MHz
Internal Operating Frequency	f_{op}	—	4.1	MHz
$\overline{\text{RESET}}$ Input Pulse Width Low[5]	t_{IRL}	125	—	ns
$\overline{\text{IRQ}}$ Interrupt Pulse Width Low[6] (Edge-Triggered)	t_{ILIH}	125	—	ns

1. V_{SS} = 0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{DD} unless noted
2. See **Table 14** and **Table 15** for more information.
3. No more than 10% duty cycle deviation from 50%
4. Some modules may require a minimum frequency greater than dc for proper operation; see appropriate table for this information.
5. Minimum pulse width reset is guaranteed to be recognized; it is possible for a smaller pulse width to cause a reset.
6. Minimum pulse width is for guaranteed interrupt; it is possible for a smaller pulse width to be recognized.

Table 9. Control Timing (V_{DD} = 2.0 Vdc ± 10%)[1]

Characteristic	Symbol	Min	Max	Unit
Frequency of Operation[2] Crystal Option External Clock Option[3]	f_{osc}	1 dc[4]	8 8.4	MHz MHz
Internal Operating Frequency	f_{op}	—	2.1	MHz
$\overline{\text{RESET}}$ Input Pulse Width Low[5]	t_{IRL}	125	—	ns
$\overline{\text{IRQ}}$ Interrupt Pulse Width Low[6] (Edge-Triggered)	t_{ILIH}	125	—	ns

[1] V_{SS} = 0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{DD} unless noted

[2] See **Table 14** and **Table 15** for more information.

[3] No more than 10% duty cycle deviation from 50%

[4] Some modules may require a minimum frequency greater than dc for proper operation; see appropriate table for this information.

[5] Minimum pulse width reset is guaranteed to be recognized; it is possible for a smaller pulse width to cause a reset.

[6] Minimum pulse width is for guaranteed interrupt; it is possible for a smaller pulse width to be recognized.

SPI Characteristics

Table 10. SPI Timing (V_{DD} = 5.0 Vdc ± 10%)[1]

Diagram Number[2]	Characteristic	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	$f_{op(m)}$ $f_{op(s)}$	fop/128 dc	$f_{op}/2$ f_{op}	MHz MHz
1	Cycle Time Master Slave	$t_{cyc(m)}$ $t_{cyc(s)}$	2 1	128 —	t_{cyc} t_{cyc}
2	Enable Lead Time	$t_{Lead(s)}$	15		ns
3	Enable Lag Time	$t_{Lag(s)}$	15		ns
4	Clock (SCK) High Time Master Slave	$t_{sckh(m)}$ $t_{sckh(s)}$	100 50	— —	ns ns
5	Clock (SCK) Low Time Master Slave	$t_{sckl(m)}$ $t_{sckl(s)}$	100 50	— —	ns ns
6	Data Setup Time (Inputs) Master Slave	$t_{su(m)}$ $t_{su(s)}$	45 5	— —	ns ns
7	Data Hold Time (Inputs) Master Slave	$t_{h(m)}$ $t_{h(s)}$	0 15	— —	ns ns
8	Access Time, Slave[3] CPHA = 0 CHPA = 1	$t_{a(cp0)}$ $t_{a(cp1)}$	0 0	40 20	ns ns
9	Disable Time, Slave[4]	$t_{dis(s)}$	—	25	ns
10	Data Valid Time (After Enable Edge) Master Slave[5]	$t_{v(m)}$ $t_{v(s)}$	— —	10 40	ns ns
11	Data Hold Time (Outputs, After Enable Edge) Master Slave	$t_{ho(m)}$ $t_{ho(s)}$	0 5	— —	ns ns

1. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; assumes 100 pf load on all SPI pins
2. Numbers refer to dimensions in **Figure 1 . SPI Master Timing** and **Figure 2 . SPI Slave Timing.**
3. Time to data active from high-impedance state.
4. Hold time to high-impedance state.
5. With 100 pF on all SPI pins.

Table 11. SPI Timing (V_{DD} = 3.0 Vdc ± 10%)[1]

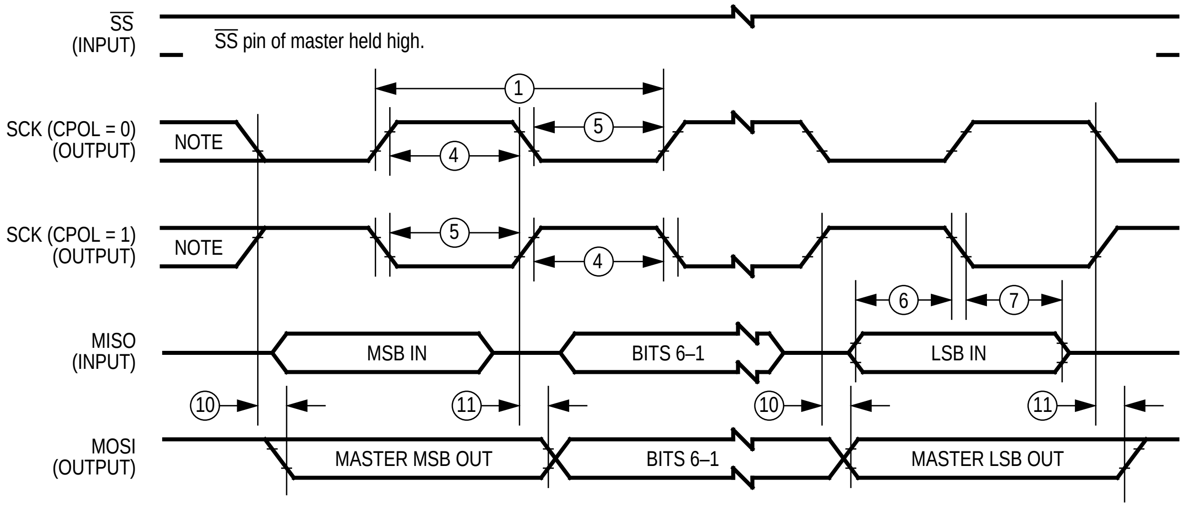
Diagram Number[2]	Characteristic	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	$f_{op(m)}$ $f_{op(s)}$	f_{op}/128 dc	f_{op}/2 f_{op}	MHz MHz
1	Cycle Time Master Slave	$t_{cyc(m)}$ $t_{cyc(s)}$	2 1	128 —	t_{cyc} t_{cyc}
2	Enable Lead Time	$t_{Lead(s)}$	30		ns
3	Enable Lag Time	$t_{Lag(s)}$	30		ns
4	Clock (SCK) High Time Master Slave	$t_{sckh(m)}$ $t_{sckh(s)}$	200 100	— —	ns ns
5	Clock (SCK) Low Time Master Slave	$t_{sckl(m)}$ $t_{sckl(s)}$	200 100	— —	ns ns
6	Data Setup Time (Inputs) Master Slave	$t_{su(m)}$ $t_{su(s)}$	90 10	— —	ns ns
7	Data Hold Time (Inputs) Master Slave	$t_{h(m)}$ $t_{h(s)}$	0 30	— —	ns ns
8	Access Time, Slave[3] CPHA = 0 CPHA = 1	$t_{a(cp0)}$ $t_{a(cp1)}$	0 0	80 40	ns ns
9	Disable Time, Slave[4]	$t_{dis(s)}$	—	50	ns
10	Data Valid Time (After Enable Edge) Master Slave[5]	$t_{v(m)}$ $t_{v(s)}$	— —	20 80	ns ns
11	Data Hold Time (Outputs, After Enable Edge) Master Slave	$t_{ho(m)}$ $t_{ho(s)}$	0 10	— —	ns ns

1. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; assumes 100 pf load on all SPI pins
2. Numbers refer to dimensions in **Figure 1** and **Figure 2**.
3. Time to data active from high-impedance state.
4. Hold time to high-impedance state.
5. With 100 pF on all SPI pins.

Table 12. SPI Timing (V_{DD} = 2.0 Vdc ± 10%)[1]

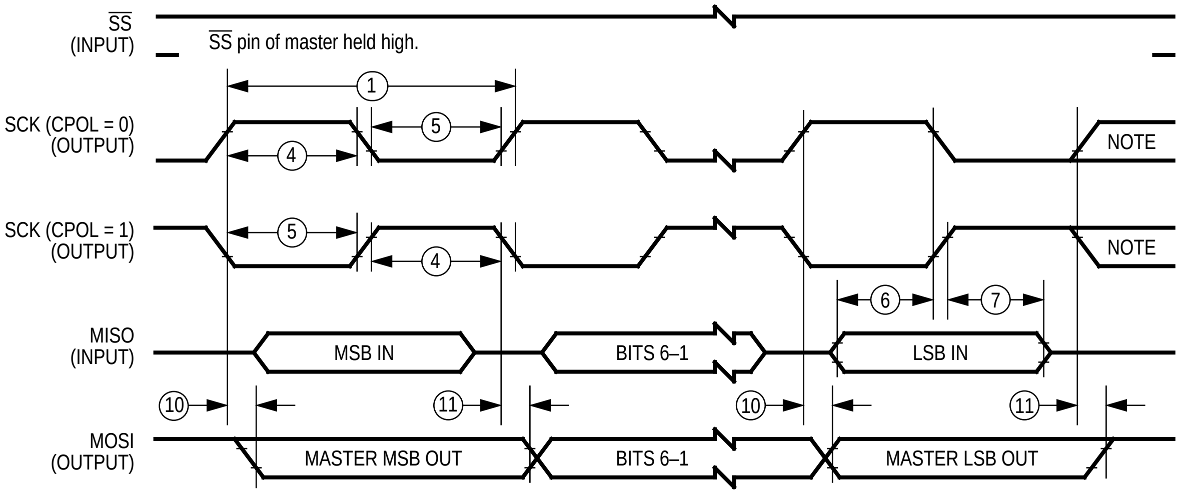
Diagram Number[2]	Characteristic	Symbol	Min	Max	Unit
	Operating Frequency Master Slave	$f_{op(m)}$ $f_{op(s)}$	f_{op}/128 dc	f_{op}/2 f_{op}	MHz MHz
1	Cycle Time Master Slave	$t_{cyc(m)}$ $t_{cyc(s)}$	2 1	128 —	t_{cyc} t_{cyc}
2	Enable Lead Time	$t_{Lead(s)}$	60		ns
3	Enable Lag Time	$t_{Lag(s)}$	60		ns
4	Clock (SCK) High Time Master Slave	$t_{sckh(m)}$ $t_{sckh(s)}$	400 200	— —	ns ns
5	Clock (SCK) Low Time Master Slave	$t_{sckl(m)}$ $t_{sckl(s)}$	400 200	— —	ns ns
6	Data Setup Time (Inputs) Master Slave	$t_{su(m)}$ $t_{su(s)}$	180 20	— —	ns ns
7	Data Hold Time (Inputs) Master Slave	$t_{h(m)}$ $t_{h(s)}$	0 60	— —	ns ns
8	Access Time, Slave[3] CPHA = 0 CPHA = 1	$t_{a(cp0)}$ $t_{a(cp1)}$	0 0	160 80	ns ns
9	Disable Time, Slave[4]	$t_{dis(s)}$	—	100	ns
10	Data Valid Time (After Enable Edge) Master Slave[5]	$t_{v(m)}$ $t_{v(s)}$	— —	40 160	ns ns
11	Data Hold Time (Outputs, After Enable Edge) Master Slave	$t_{ho(m)}$ $t_{ho(s)}$	0 20	— —	ns ns

1. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; assumes 100 pf load on all SPI pins
2. Numbers refer to dimensions in **Figure 1** and **Figure 2**.
3. Time to data active from high-impedance state.
4. Hold time to high-impedance state.
5. With 100 pF on all SPI pins.



NOTE: This first clock edge is generated internally, but is not seen at the SCK pin.

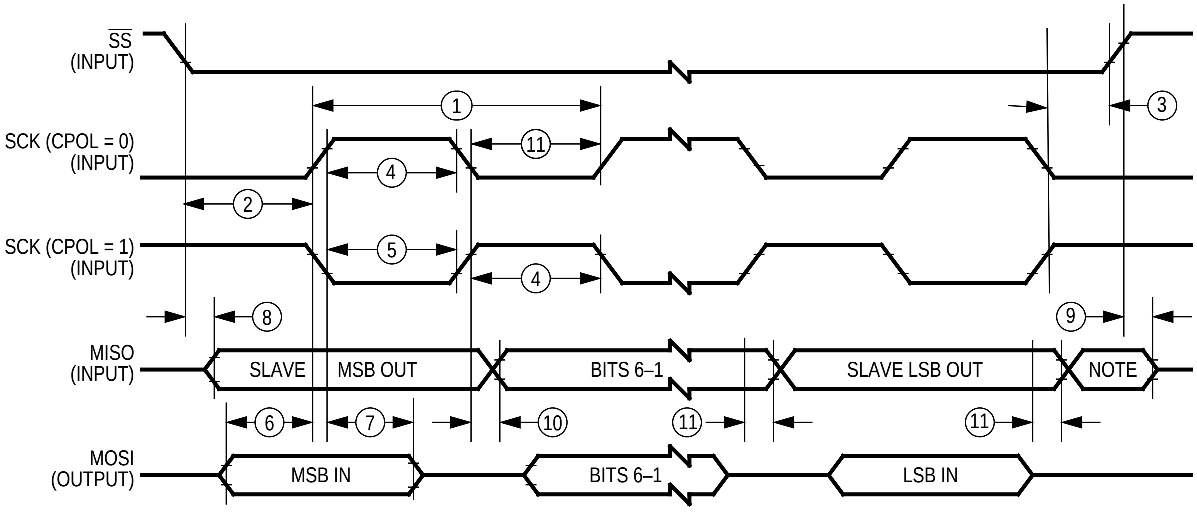
a) SPI Master Timing (CPHA = 0)



NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

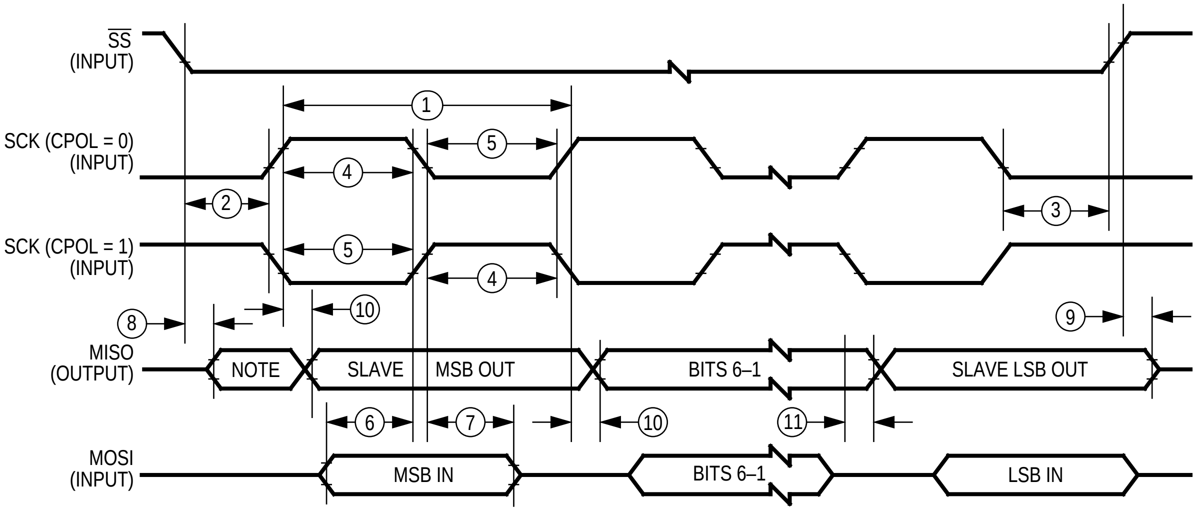
b) SPI Master Timing (CPHA = 1)

Figure 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 2. SPI Slave Timing

TImer Interface Module Characteristics

Table 13. TIM Timing

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

CGM Characteristics

Table 14. CGM Component Specifications

Characteristic	Symbol	Min	Typ	Max	Unit
Crystal (X1) Frequency[1]	f_{xclk}	1	4.9152	16	MHz
Crystal Load Capacitance[2]	C_L	—	—	—	pF
Crystal Fixed Capacitance[2]	C_1	—	$2 \times C_L$	—	pF
Crystal Tuning Capacitance[2]	C_2	—	$2 \times C_L$	—	pF
Feedback Bias Resistor	R_B	—	1	—	MΩ
Series Resistor[3]	R_S	0	—	3.3	kΩ
Filter Capacitor	C_F	—	$C_{fact} \times (V_{DDA}/f_{XCLK})$	—	pF
Bypass Capacitor[4]	C_{byp}	—	0.1	—	µF

[1] Fundamental mode crystals only

[2] Consult crystal manufacturer's data.

[3] Not required

[4] C_{byp} must provide low ac impedance from $f = f_{xclk}/100$ to $100 \times f_{vclk}$, so series resistance must be considered.

Table 15. CGM Operating Conditions

Characteristic	Symbol	Min	Typ	Max	Notes
Crystal Reference Frequency	f_{xclk}	1	—	8	MHz
Range Nominal Multiplier	f_{nom}	—	4.9152	—	MHz
VCO Center-of-Range Frequency[1]	f_{vrs}	4.9152	—	32.8	MHz
Medium Voltage VCO Center-of-Range Frequency[2]		4.9152	—	16.4	MHz
Low Voltage VCO Center-of-Range Frequency[3]		4.9152	—	8.4	MHz
VCO Frequency Multiplier	N	1	—	15	—
VCO Center-of-Range Multiplier	L	1	—	15	—
VCO Operating Frequency	f_{vclk}	f_{vrsmin}	—	f_{vrsmax}	MHz

1. 5.0 V ± 10% V_{DD} only
2. 3.0 V ± 10% V_{DD} only
3. 2.0 V ± 10% V_{DD} only

Table 16. CGM Acquisition and Lock Time Specifications

Description	Symbol	Min	Typ	Max	Unit
Filter Capacitor Multiply Factor	C_{fact}	—	0.0154	—	F/sV
Acquisition Mode Time Factor	K_{acq}	—	0.1135	—	V
Tracking Mode Time Factor	K_{trk}	—	0.0174	—	V
Manual Mode Time to Stable[1]	t_{acq}	—	$\frac{8 \times V_{DDA}}{f_{xclk} \times K_{acq}}$	—	s
Manual Stable to Lock Time[1]	t_{al}	—	$\frac{4 \times V_{DDA}}{f_{xclk} \times K_{trk}}$	—	s
Manual Acquisition Time	t_{Lock}	—	$t_{acq} + t_{al}$	—	s
Tracking Mode Entry Frequency Tolerance	Δ_{trk}	0	—	3.6%	—
Acquisition Mode Entry Frequency Tolerance	Δ_{acq}	6.3%	—	7.2%	—
LOCK Entry Freq. Tolerance	Δ_{Lock}	0	—	0.9%	—
LOCK Exit Freq. Tolerance	Δ_{unl}	0.9%	—	1.8%	—
Reference cycles per Acquisition Mode Measurement	n_{acq}	—	32	—	Cyc.

Table 16. CGM Acquisition and Lock Time Specifications (Continued)

Description	Symbol	Min	Typ	Max	Unit
Reference cycles per Tracking Mode Measurement	n_{trk}	—	128	—	Cyc.
Automatic Mode Time to Stable[1]	t_{acq}	$\frac{n_{acq}}{f_{xclk}}$	$\frac{8 \times V_{DDA}}{f_{xclk} \times K_{acq}}$	—	s
Automatic Stable to Lock Time[1]	t_{al}	$\frac{n_{trk}}{f_{xclk}}$	$\frac{4 \times V_{DDA}}{f_{xclk} \times K_{trk}}$	—	s
Automatic Lock Time	t_{Lock}	—	$t_{acq} + t_{al}$	—	s
PLL Jitter[2]	f_J	0	—	$(f_{crys}) \times 0.025\%$ $\times 2^P N/4$	Hz

[1] If C_F chosen correctly

[2] Deviation of average bus frequency over 2 ms. N = VCO frequency multiplier.

Memory Characteristics

Table 17. Memory Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
RAM Data Retention Voltage	V_{rm}	0.9	—	—	V

Mechanical Specifications

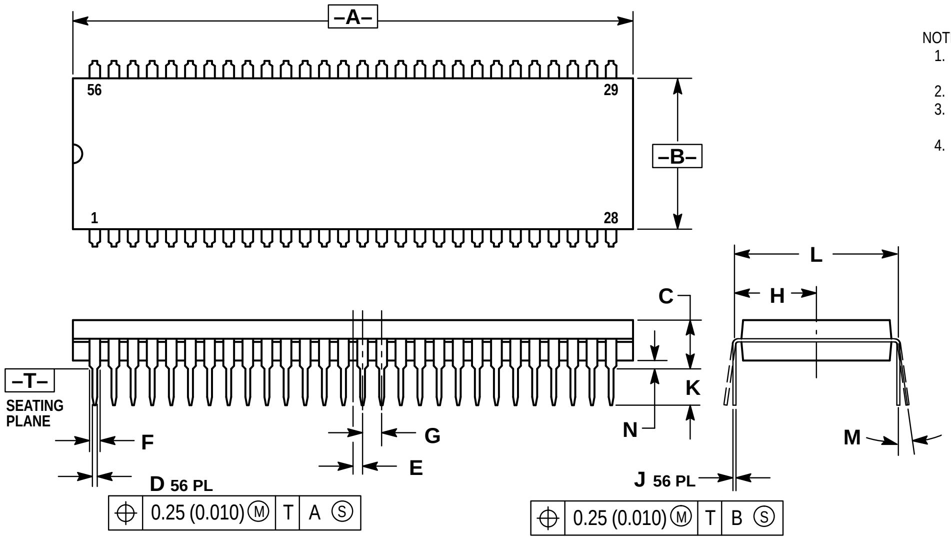
The MC68HC08XL36 is available in these packages:

- 56-pin plastic dual in-line shrink (SDIP)
- 64-pin plastic quad flat pack (QFP)

The following figures show the latest packages at the time of this publication. To make sure that you have the latest package specifications, contact one of the following:

- Local Motorola Sales Office
- Motorola Mfax
  - Phone 602-244-6609
  - EMAIL rmfax0 @email.sps.mot.com
- Worldwide Web (wwweb) at http://design-net.com

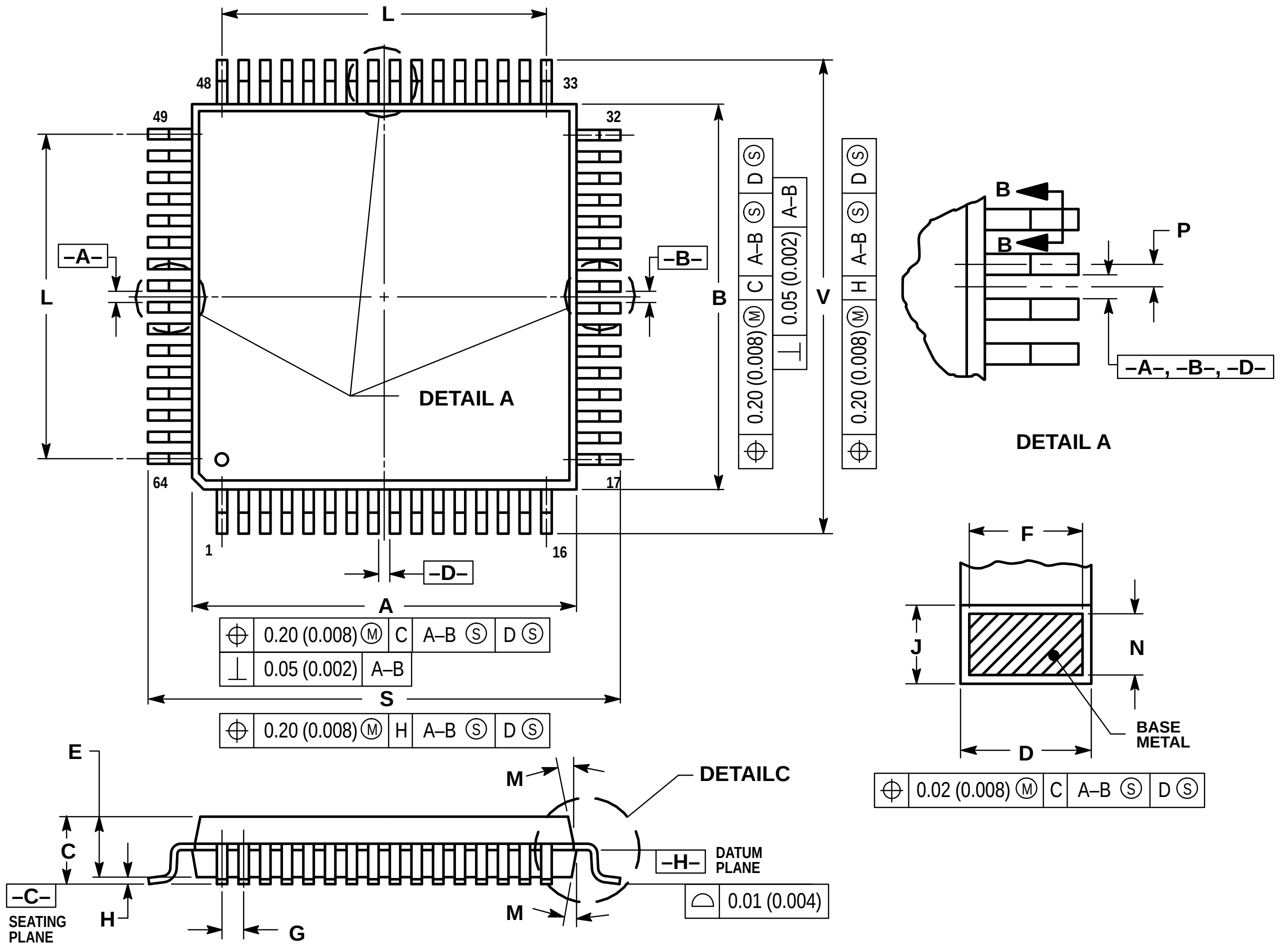
Follow Mfax or wwweb on-line instructions to retrive the current mechanical specifications.

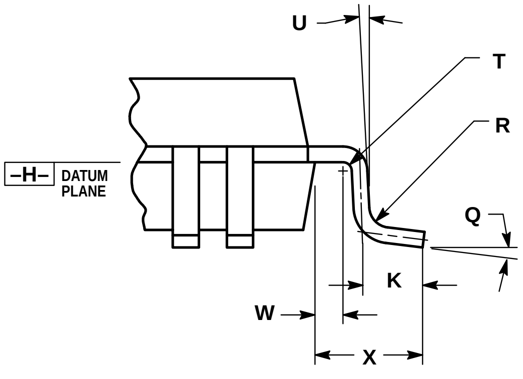


NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH. MAXIMUM MOLD FLASH 0.25 (0.010)

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	2.035	2.065	51.69	52.45
B	0.540	0.560	13.72	14.22
C	0.155	0.200	3.94	5.08
D	0.014	0.022	0.36	0.56
E	0.035 BSC		0.89 BSC	
F	0.032	0.046	0.81	1.17
G	0.070 BSC		1.778 BSC	
H	0.300 BSC		7.62 BSC	
J	0.008	0.015	0.20	0.38
K	0.115	0.135	2.92	3.43
L	0.600 BSC		15.24 BSC	
M	0 °	15 °	0 °	15 °
N	0.020	0.040	0.51	1.02

Figure 3. Case Outline Drawing 859-01





DETAIL C

NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE –H– IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS –A–, –B– AND –D– TO BE DETERMINED AT DATUM PLANE –H–.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE –C–.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE –H–.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) PER SIDE. TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	13.90	14.10	0.547	0.555
B	13.90	14.10	0.547	0.555
C	2.15	2.45	0.085	0.096
D	0.30	0.45	0.012	0.018
E	2.00	2.40	0.079	0.094
F	0.30	0.40	0.012	0.016
G	0.80 BSC		0.031 BSC	
H	——	0.25	——	0.010
J	0.13	0.23	0.005	0.009
K	0.65	0.95	0.026	0.037
L	12.00 REF		0.472 REF	
M	5°	10°	5°	10°
N	0.13	0.17	0.005	0.007
P	0.40 BSC		0.016 BSC	
Q	0°	7°	0°	7°
R	0.13	0.30	0.005	0.012
S	16.95	17.45	0.667	0.687
T	0.13	——	0.005	——
U	0°	——	0°	——
V	16.95	17.45	0.667	0.687
W	0.35	0.45	0.014	0.018
X	1.6 REF		0.063 REF	

Figure 4. Case Outline Drawing 840b-01

Glossary

A — See “accumulator (A).”

accumulator (A) — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

acquisition mode — A mode of PLL operation during startup before the PLL locks on a frequency. Also see "tracking mode."

address bus — The set of wires that the CPU or DMA uses to read and write memory locations.

addressing mode — The way that the CPU determines the operand address for an instruction. The M68HC08 CPU has 16 addressing modes.

ALU — See “arithmetic logic unit (ALU).”

arithmetic logic unit (ALU) — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

asynchronous — Refers to logic circuits and operations that are not synchronized by a common reference signal.

baud rate — The total number of bits transmitted per unit of time.

BCD — See “binary-coded decimal (BCD).”

binary — Relating to the base 2 number system.

binary number system — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

binary-coded decimal (BCD) — A notation that uses 4-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

234 (decimal) = 0010 0011 0100 (BCD)

bit — A binary digit. A bit has a value of either logic 0 or logic 1.

branch instruction — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

break module — A module in the M68HC08 Family. The break module allows software to halt program execution at a programmable point in order to enter a background routine.

breakpoint — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

break interrupt — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

bus — A set of wires that transfers logic signals.

bus clock — The bus clock is derived from the CGMOUT output from the CGM. The bus clock frequency, f_{op}, is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

byte — A set of eight bits.

C — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

CCR — See “condition code register.”

central processor unit (CPU) — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

CGM — See “clock generator module (CGM).”

clear — To change a bit from logic 1 to logic 0; the opposite of set.

clock — A square wave signal used to synchronize events in a computer.

clock generator module (CGM) — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.

comparator — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

computer operating properly module (COP) — A counter module in the M68HC08 Family that resets the MCU if allowed to overflow.

condition code register (CCR) — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.

control bit — One bit of a register manipulated by software to control the operation of the module.

control unit — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

COP — See "computer operating properly module (COP)."

counter clock — The input clock to the TIM counter. This clock is the output of the TIM prescaler.

CPU — See “central processor unit (CPU).”

CPU08 — The central processor unit of the M68HC08 Family.

CPU clock — The CPU clock is derived from the CGMOUT output from the CGM. The CPU clock frequency is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

CPU cycles — A CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

CPU registers — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

- A (8-bit accumulator)
- H:X (16-bit index register)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

CSIC — customer-specified integrated circuit

cycle time — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

decimal number system — Base 10 numbering system that uses the digits zero through nine.

direct memory access module (DMA) — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

DMA — See "direct memory access module (DMA)."

DMA service request — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

duty cycle — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

EEPROM — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically reprogrammed.

EPROM — Erasable, programmable, read-only memory. A nonvolatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

exception — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

external interrupt module (IRQ) — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

fetch — To copy data from a memory location into the accumulator.

firmware — Instructions and data programmed into nonvolatile memory.

free-running counter — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

full-duplex transmission — Communication on a channel in which data can be sent and received simultaneously.

H — The upper byte of the 16-bit index register (H:X) in the CPU08.

H — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

hexadecimal — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

high byte — The most significant eight bits of a word.

illegal address — An address not within the memory map

illegal opcode — A nonexistent opcode.

I — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

index register (H:X) — A 16-bit register in the CPU08.   The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

input/output (I/O) — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

instructions — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

interrupt — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

interrupt request — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

I/O — See “input/output (I/0).”

IRQ — See "external interrupt module (IRQ)."

jitter — Short-term signal instability.

latch — A circuit that retains the voltage level (logic 1 or logic 0) written to it for as long as power is applied to the circuit.

latency — The time lag between instruction completion and data movement.

least significant bit (LSB) — The rightmost digit of a binary number.

logic 1 — A voltage level approximately equal to the input power voltage (V_{DD}).

logic 0 — A voltage level approximately equal to the ground voltage (V_{SS}).

low byte — The least significant eight bits of a word.

low voltage inhibit module (LVI) — A module in the M68HC08 Family that monitors power supply voltage.

LVI — See "low voltage inhibit module (LVI)."

M68HC08 — A Motorola family of 8-bit MCUs.

mark/space — The logic 1/logic 0 convention used in formatting data in serial communication.

mask — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

mask option — A optional microcontroller feature that the customer chooses to enable or disable.

mask option register (MOR) — An EPROM location containing bits that enable or disable certain MCU features.

MCU — Microcontroller unit. See “microcontroller.”

memory location — Each M68HC08 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

memory map — A pictorial representation of all memory locations in a computer system.

microcontroller — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

modulo counter — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

monitor ROM — A section of ROM that can execute commands from a host computer for testing purposes.

MOR — See "mask option register (MOR)."

most significant bit (MSB) — The leftmost digit of a binary number.

multiplexer — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

N — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

nibble — A set of four bits (half of a byte).

object code — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

opcode — A binary code that instructs the CPU to perform an operation.

open-drain — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic 1 output voltage.

operand — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

oscillator — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

OTPROM — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

overflow — A quantity that is too large to be contained in one byte or one word.

page zero — The first 256 bytes of memory (addresses $0000–$00FF).

parity — An error-checking scheme that counts the number of logic 1s in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic 1s. In an even parity system, every byte should have an even number of logic 1s. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic 1s odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic 1s in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic 1s.

PC — See “program counter (PC).”

peripheral — A circuit not under direct CPU control.

phase-locked loop (PLL) — A oscillator circuit in which the frequency of the oscillator is synchronized to a reference signal.

PLL — See "phase-locked loop (PLL)."

pointer — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

polarity — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels, V_{DD} and V_{SS}.

polling — Periodically reading a status bit to monitor the condition of a peripheral device.

port — A set of wires for communicating with off-chip devices.

prescaler — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

program — A set of computer instructions that cause a computer to perform a desired operation or operations.

program counter (PC) — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

pull — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

pullup — A transistor in the output of a logic gate that connects the output to the logic 1 voltage of the power supply.

pulse-width — The amount of time a signal is on as opposed to being in its off state.

pulse-width modulation (PWM) — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

push — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

PWM period — The time required for one complete cycle of a PWM waveform.

RAM — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

RC circuit — A circuit consisting of capacitors and resistors having a defined time constant.

read — To copy the contents of a memory location to the accumulator.

register — A circuit that stores a group of bits.

reserved memory location — A memory location that is used only in special factory test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

reset — To force a device to a known condition.

ROM — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

SCI — See "serial communication interface module (SCI)."

serial — Pertaining to sequential transmission over a single line.

serial communications interface module (SCI) — A module in the M68HC08 Family that supports asynchronous communication.

serial peripheral interface module (SPI) — A module in the M68HC08 Family that supports synchronous communication.

set — To change a bit from logic 0 to logic 1; opposite of clear.

shift register — A chain of circuits that can retain the logic levels (logic 1 or logic 0) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

signed — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic 0 for positive and logic 1 for negative. The other seven bits indicate the magnitude of the number.

software — Instructions and data that control the operation of a microcontroller.

software interrupt (SWI) — An instruction that causes an interrupt and its associated vector fetch.

SPI — See "serial peripheral interface module (SPI)."

stack — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

stack pointer (SP) — A 16-bit register in the CPU08 containing the address of the next available storage location on the stack.

start bit — A bit that signals the beginning of an asynchronous serial transmission.

status bit — A register bit that indicates the condition of a device.

stop bit — A bit that signals the end of an asynchronous serial transmission.

subroutine — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

synchronous — Refers to logic circuits and operations that are synchronized by a common reference signal.

TIM — See "timer interface module (TIM)."

timer interface module (TIM) — A module used to relate events in a system to a point in time.

timer — A module used to relate events in a system to a point in time.

toggle — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

tracking mode — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. Also see "acquisition mode."

two’s complement — A means of performing binary subtraction using addition techniques. The most significant bit of a two’s complement number indicates the sign of the number (1 indicates negative). The two’s complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

unbuffered — Utilizes only one register for data; new data overwrites current data.

unimplemented memory location — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value. Executing an opcode at an unimplemented location causes an illegal address reset.

V —The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

variable — A value that changes during the course of program execution.

VCO — See "voltage-controlled oscillator."

vector — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

voltage-controlled oscillator (VCO) — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

waveform — A graphical representation in which the amplitude of a wave is plotted against time.

wired-OR — Connection of circuit outputs so that if any output is high, the connection point is high.

word — A set of two bytes (16 bits).

write — The transfer of a byte of data from the CPU to a memory location.

X — The lower byte of the index register (H:X) in the CPU08.

Z — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

Index

Literature Updates

This document contains the latest data available at publication time. For updates, contact one of the centers listed below:

Literature Distribution Centers

Order literature by mail or phone.

USA/Europe

Motorola Literature Distribution
P.O. Box 20912
Phoenix, Arizona 85036
Phone 1 800 441-2447 or 602 303-5454

Japan

Nippon Motorola Ltd.
Tatsumi-SPD-JLDC
Toshikatsu Otsuki
6F Seibu-Butsuryu Center
3-14-2 Tatsumi Koto-Ku
Tokyo 135, Japan
Phone 03-3521-8315

Hong Kong

Motorola Semiconductors H.K. Ltd.
8B Tai Ping Industrial Park
51 Ting Kok Road
Tai Po, N.T., Hong Kong
Phone 852-26629298

Mfax

To access this worldwide faxing service call or contact by electronic mail:

RMFAX0@email.sps.mot.com
TOUCH-TONE 602-244-6609

Or, on the http://Design-NET.com home page, select the Mfax icon. Obtain a fax of complete, easy-to-use Mfax instructions by entering your FAX number and then pressing the 1 key.

Motorola SPS World Marketing World Wide Web Server

Use the Internet to access Motorola’s World Wide Web server. Use the following URL:

http://design-net.com

CSIC Microcontroller Division’s Web Site

Directly access the CSIC Microcontroller Division’s web site with the following URL:

http://design-net.com/csic/CSIC_home.html

MC68HC08XL36 TECHNICAL DATA
CUSTOMER RESPONSE SURVEY

To make M68HC08 documentation as clear, complete, and easy to use as possible, we need your comments. Please complete this form and return it by mail, or FAX it to 512-891-3236.

1. How do you rate the quality of this document?

	High				Low		High				Low
Organization	□	□	□	□	□	Tables	□	□	□	□	□
Readability	□	□	□	□	□	Table of contents	□	□	□	□	□
Accuracy	□	□	□	□	□	Page size/binding	□	□	□	□	□
Figures	□	□	□	□	□	Overall impression	□	□	□	□	□

Comments: ____________________

____

____

2. What is your intended use for this document?

Device selection for new application □ Other □ Please specify: ____________________

System design □ ____________________

Training □ ____________________

3. Does this document help you to perform your job?

Yes				No
□	□	□	□	□

Comments: ____________________

____

____

4. Are you able to easily find the information you need?

Yes				No
□	□	□	□	□

Comments: ____________________

____

____

5. Does each section of the document provide you with enough information?

	Yes				No		Yes				No
Introduction	□	□	□	□	□	Timer Interface Module	□	□	□	□	□
Memory	□	□	□	□	□	SPI Module	□	□	□	□	□
RAM	□	□	□	□	□	SCI Module	□	□	□	□	□
ROM	□	□	□	□	□	I/O Ports	□	□	□	□	□
CPU	□	□	□	□	□	COP Module	□	□	□	□	□
Resets and Interrupts	□	□	□	□	□	IRQ Module	□	□	□	□	□
Low-Power Modes	□	□	□	□	□	Keyboard Interrupt Module	□	□	□	□	□
Clock Generator Module	□	□	□	□	□	LVI Module	□	□	□	□	□
Direct Memory Access Module	□	□	□	□	□	Specifications	□	□	□	□	□
Break Module	□	□	□	□	□	Glossary	□	□	□	□	□
Monitor ROM	□	□	□	□	□	Index	□	□	□	□	□
							□	□	□	□	□

6. What would you like us to do to improve this document? ____________________

____

____

____
____
____

Motorola
6501 William Cannon Drive West
Mail Stop OE17
Austin, Texas 78735-8598

Attn: CSIC Publications Department

MOTOROLA
CSIC Microcontroller Division

Second: fold back along this line

First: cut along this line to remove

Please supply the following information (optional).

Name: ____

Company Name: ______________________________________

Title: ____

Address: ____

City: __________________________State: _____Zip:__________

Phone Number: _____________________________________