MC68HC908LV8

Data Sheet

M68HC08
Microcontrollers

MC68HC908LV8
Rev. 2
12/2005

freescale.com

freescale™
semiconductor

MC68HC908LV8

Data Sheet

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

http://freescale.com/

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

Revision History

Date	Revision Level	Description	Page Number(s)
December, 2005	2	First general release.	N/A

List of Chapters

Table of Contents

Chapter 1
General Description

Chapter 2
Memory

Chapter 3
Configuration Register (CONFIG)

Chapter 4
System Integration Module (SIM)

Chapter 5
Clock Generator Module (CGM)

Chapter 6
Timer Interface Module (TIM)

Chapter 7
Programmable Periodic Interrupt (PPI)

Chapter 8
Analog-to-Digital Converter (ADC)

Chapter 9
Liquid Crystal Display (LCD) Driver

Chapter 10
Input/Output (I/O) Ports

Chapter 11
External Interrupt (IRQ)

Chapter 12
Keyboard Interrupt Module (KBI)

Chapter 13
Computer Operating Properly (COP)

Chapter 14
Low-Voltage Inhibit (LVI)

Chapter 15
Central Processor Unit (CPU)

Chapter 16
Development Support

Chapter 17
Electrical Specifications

Chapter 18
Ordering Information and Mechanical Specifications

Chapter 1
General Description

1.1 Introduction

The MC68HC908LV8 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

1.2 Features

Features include:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- Low-power design; fully static with stop and wait modes
- Maximum internal bus frequency:
    - 8-MHz at 5-V operating voltage
    - 4-MHz at 3-V operating voltage
- 32.768 kHz crystal oscillator clock input with 32 MHz internal PLL
- 8,192 bytes user program FLASH memory with security(1)
- 512 bytes of on-chip random-access memory (RAM)
- Two 16-bit, 2-channel timer interface modules (TIM1 and TIM2) with selectable input capture, output compare, and pulse-width modulation (PWM) capability on each channel
- Programmable periodic interrupt (PPI)
- 6-channel, 10-bit analog-to-digital converter with internal bandgap reference channel (ADC10)
- 4/3 backplanes and static with maximum 24/25 frontplanes liquid crystal display (LCD) driver
- Up to 40 general-purpose input/output (I/O) ports:
    - 4 keyboard interrupt with internal pull up
    - 4 × 15 mA high current sink pins
- Resident routines for in-circuit programming and EEPROM emulation
- System protection features:
    - Optional computer operating properly (COP) reset, driven by internal RC oscillator
    - Optional low-voltage detection with reset and selectable trip points for 3-V and 5-V operation
    - Illegal opcode detection with reset
    - Illegal address detection with reset
- Master reset pin with internal pull-up and power-on reset
- $\overline{\text{IRQ}}$ with schmitt-trigger input and programmable pull up
- 52-pin low-profile quad flat pack (LQFP)

---

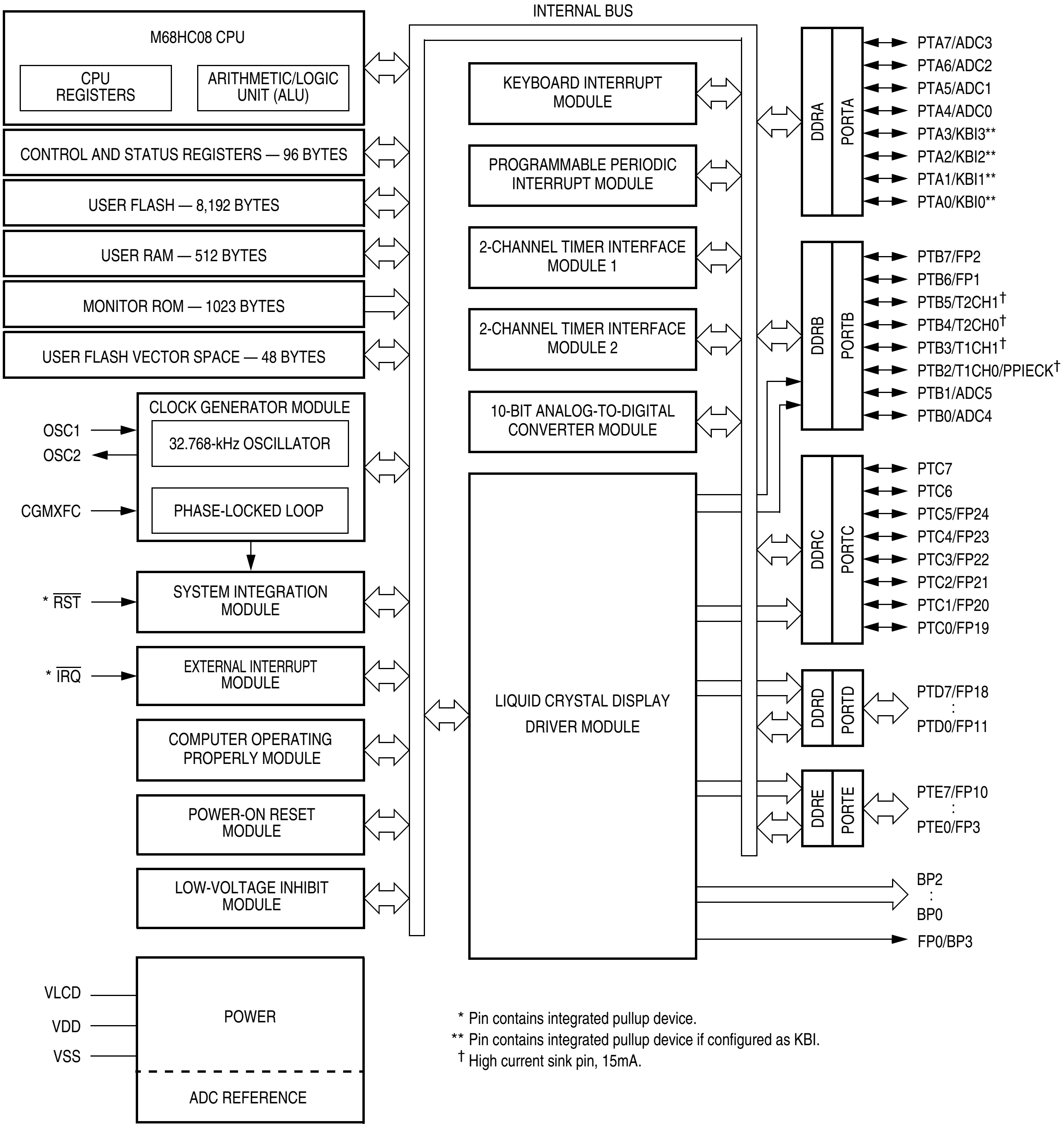
1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

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
- Efficient C language support

1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908LV8.



* Pin contains integrated pullup device.
** Pin contains integrated pullup device if configured as KBI.
† High current sink pin, 15mA.

Figure 1-1. MC68HC908LV8 Block Diagram

]

1.4 Pin Assignments

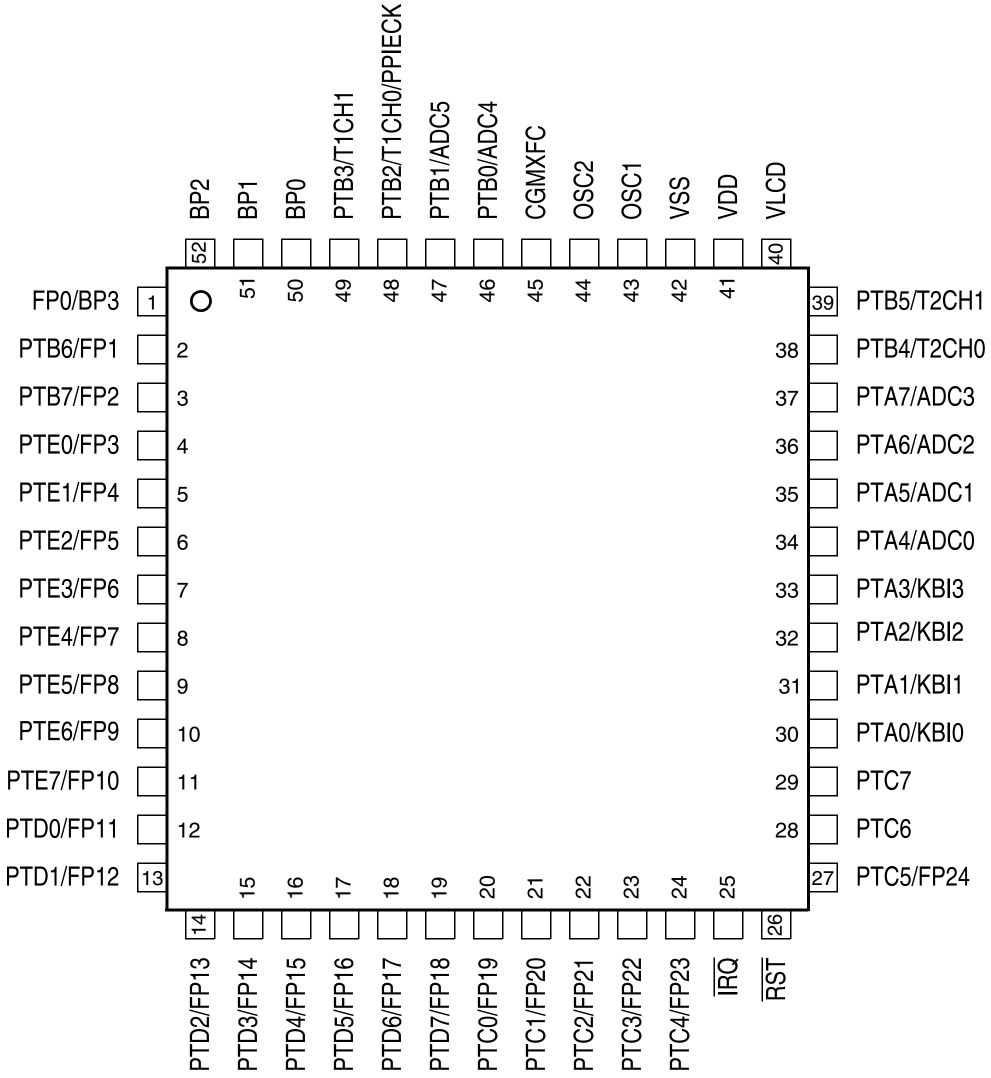


Figure 1-2. 52-Pin LQFP Pin Assignment

1.5 Pin Functions

Description of the pin functions are provided in Table 1-1.

Table 1-1. Pin Functions

Pin Name	Pin Description	Input/Output	Voltage Level
V_{DD}	Power supply	Input	5 V or 3 V
V_{SS}	Power supply ground	Output	0V
V_{LCD}	LCD bias voltage	Input	V_{DD}
$\overline{\text{RST}}$	Reset input, active low; with internal pull up and Schmitt trigger input	Input/output	V_{DD}
$\overline{\text{IRQ}}$	External $\overline{\text{IRQ}}$ pin; with programmable internal pull up and Schmitt trigger input	Input	V_{DD}
	Used for monitor mode entry	Input	V_{DD} to V_{TST}
OSC1	Crystal input	Input	V_{DD}
OSC2	Crystal oscillator output; inverted OSC1 signal	Output	V_{DD}
CGMXFC	CGM external filter capacitor connection.	In/Out	Analog
BP0–BP2	LCD backplane drivers	Output	V_{DD}
BP3/FP0	LCD backplane driver BP3 or frontplane driver FP0	Output	V_{DD}
PTA0/KBI0 PTA1/KBI1 PTA2/KBI2 PTA3/KBI3 PTA4/ADC0 PTA5/ADC1 PTA6/ADC2 PTA7/ADC3	8-bit general-purpose I/O port	Input/output	V_{DD}
	PTA0–PTA3 as keyboard interrupts with pull-up device, KBI0–KBI4	Input	V_{DD}
	PTA4–PTA7 as ADC input channels, ADC0–ADC3	Input	V_{SS} to V_{DD}
PTB0/ADC4 PTB1/ADC5 PTB2/T1CH0/PPIECK PTB3/T1CH1 PTB4/T2CH0 PTB5/T2CH1 PTB6/FP1 PTB7/FP2	8-bit general-purpose I/O port, with high current sinks on PTB2–PTB5	Input/output	V_{DD}
	PTB0–PTB1 as ADC input channels, ADC4–ADC5	Input	V_{SS} to V_{DD}
	PTB2 as PPIECK; external clock source input for PPI	Input	V_{DD}
	PTB2 as T1CH0 of TIM1	Input/output	V_{DD}
	PTB3 as T1CH1 of TIM1	Input/output	V_{DD}
	PTB4 as T2CH0 of TIM2	Input/output	V_{DD}
	PTB5 as T2CH1 of TIM2	Input/output	V_{DD}
	PTB6–PTB7 as LCD frontplane drivers, FP1–FP2	Output	V_{DD}

Continued on next page

Table 1-1. Pin Functions (Continued)

Pin Name	Pin Description	Input/Output	Voltage Level
PTC0/FP19 PTC1/FP20 PTC2/FP21 PTC3/FP22 PTC4/FP23 PTC5/FP24 PTC6 PTC7	8-bit general-purpose I/O port	Input/output	V_{DD}
	PTC0–PTC5 as LCD frontplane drivers, FP19–FP24	Output	V_{DD}
PTD0/FP11 PTD1/FP12 PTD2/FP13 PTD3/FP14 PTD4/FP15 PTD5/FP16 PTD6/FP17 PTD7/FP18	8-bit general-purpose I/O port	Input/output	V_{DD}
	PTD0–PTD7 as LCD frontplane drivers, FP11–FP18	Output	V_{DD}
PTE0/FP3 PTE1/FP4 PTE2/FP5 PTE3/FP6 PTE4/FP7 PTE5/FP8 PTE6/FP9 PTE7/FP10	8-bit general-purpose I/O port	Input/output	V_{DD}
	PTE0–PTE7 as LCD frontplane drivers, FP3–FP10	Output	V_{DD}

Chapter 2
Memory

2.1 Introduction

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

- 8,192 bytes of user FLASH memory
- 512 bytes of random-access memory (RAM)
- 48 bytes of user-defined vectors
- 1023 bytes of monitor ROM

2.2 I/O Section

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

- $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
- $FE07; Reserved
- $FE08; FLASH control register, FLCR
- $FE09; Reserved
- $FE0A; Reserved
- $FE0B; Reserved
- $FE0C; Break address register high, BRKH
- $FE0D; Break address register low, BRKL
- $FE0E; Break status and control register, BRKSCR
- $FE0F; Low-voltage inhibit status register, LVISR
- $FF7E; FLASH block protect register, FLBPR (FLASH register)
- $FFFF; COP control register, COPCTL

2.3 Monitor ROM

The 350 bytes at addresses $FE20–$FF7D are reserved ROM addresses that contain the instructions for the monitor functions.

$0000 ↓ $007F	I/O REGISTERS 128 BYTES
$0080 ↓ $027F	RAM 512 BYTES
$0280 ↓ $0B96	UNIMPLEMENTED 2,327 BYTES
$0B97 ↓ $0E1F	FLASH OPERATION ROM BLOCK 649 BYTES
$0E20 ↓ $DDFF	UNIMPLEMENTED 53,216 BYTES
$DE00 ↓ $FDFF	FLASH MEMORY 8,192 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)
$FE07	RESERVED
$FE08	FLASH CONTROL REGISTER (FLCR)
$FE09 ↓ $FF0B	RESERVED
$FE0C	BREAK ADDRESS HIGH REGISTER (BRKH)
$FE0D	BREAK ADDRESS LOW REGISTER (BRKL)
$FE0E	BREAK STATUS AND CONTROL REGISTER (BRKSCR)
$FE0F	LVI STATUS REGISTER (LVISR)
$FE10 ↓ $FE1F	UNIMPLEMENTED 16 BYTES
$FE20 ↓ $FF7D	MONITOR ROM 350 BYTES
$FF7E	FLASH BLOCK PROTECT REGISTER (FLBPR)
$FF7F ↓ $FF96	MONITOR JUMP TABLE 24 BYTES
$FF97 ↓ $FFCF	UNIMPLEMENTED 57 BYTES
$FFD0 ↓ $FFFF	USER FLASH VECTORS 48 BYTES

Figure 2-1. Memory Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC)	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$0008	Data Direction Register E (DDRE)	Read: Write:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Reset:	0	0	0	0	0	0	0	0
$0009	Port E Data Register (PTE)	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Reset:	Unaffected by reset							
$000A	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨

U = Unaffected X = Indeterminate ▨ = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 8)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000B	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$000C	Port-B High Current Drive Control Register (HDB)	Read:	R	PPI1L	HDB5	HDB4	HDB3	HDB2	PPI1CLKS 1	PPI1CLKS 0
		Write:		▨						
		Reset:			0	0	0	0	0	0
$000D ↓ $001A	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$001B	Keyboard Status and Control Register (KBSCR)	Read:	R	R	R	R	KEYF	0	IMASKK	MODEK
		Write:					▨	ACKK		
		Reset:	0	0	0	0	0	0	0	0
$001C	Keyboard Interrupt Enable Register (KBIER)	Read:	0	PPI1IE2	PPI1IE1	PPI1IE0	KBIE3	KBIE2	KBIE1	KBIE0
		Write:	▨							
		Reset:	0	0	0	0	0	0	0	0
$001D	Configuration Register 2 (CONFIG2)(1)	Read:	STOP_ XCLKEN	R	PEE	PDE	PCEH	PCEL	LVISEL1	LVISEL0
		Write:								
		Reset:	0	0	0	0	0	0	0(2)	1(2)
$001E	IRQ Status and Control Register (INTSCR)	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:	▨	▨	▨	▨	▨	ACK		
		Reset:	0	0	0	0	0	0	0	0
$001F	Configuration Register 1 (CONFIG1)(1)	Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	R	SSREC	STOP	COPD
		Write:								
		Reset:	0	0	0	0(2)	0	0	0	0

1. One-time writable register after each reset.
2. One time writable after each POR and reset by POR only.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0020	Timer 1 Status and Control Register (T1SC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	▨			
		Reset:	0	0	1	0	0	0	0	0
$0021	Timer 1 Counter Register High (T1CNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate ▨ = Unimplemented R = Reserved

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0022	Timer 1 Counter Register Low (T1CNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$0023	Timer 1 Counter Modulo Register High (T1MODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0024	Timer 1 Counter Modulo Register Low (T1MODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer 1 Channel 0 Status and Control Register (T1SC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0026	Timer 1 Channel 0 Register High (T1CH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0027	Timer 1 Channel 0 Register Low (T1CH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0028	Timer 1 Channel 1 Status and Control Register (T1SC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		▨					
		Reset:	0	0	0	0	0	0	0	0
$0029	Timer 1 Channel 1 Register High (T1CH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$002A	Timer 1 Channel 1 Register Low (T1CH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$002B	Timer 2 Status and Control Register (T2SC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST	▨			
		Reset:	0	0	1	0	0	0	0	0
$002C	Timer 2 Counter Register High (T2CNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected
X = Indeterminate
▨ = Unimplemented
R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 8)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002D	Timer 2 Counter Register Low (T2CNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$002E	Timer 2 Counter Modulo Register High (T2MODH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$002F	Timer 2 Counter Modulo Register Low (T2MODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0030	Timer 2 Channel 0 Status and Control Register (T2SC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0031	Timer 2 Channel 0 Register High (T2CH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0032	Timer 2 Channel 0 Register Low (T2CH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0033	Timer 2 Channel 1 Status and Control Register (T2SC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		▨					
		Reset:	0	0	0	0	0	0	0	0
$0034	Timer 2 Channel 1 Register High (T2CH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0035	Timer 2 Channel 1 Register Low (T2CH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0036	PLL Control Register (PTCL)	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		Write:		▨						
		Reset:	0	0	1	0	0	0	0	0
$0037	PLL Bandwidth Control Register (PBWC)	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	0	0	0	0	R
		Write:		▨		▨	▨	▨	▨	
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected   X = Indeterminate   ▨ = Unimplemented   R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 8)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0038	PLL Multiplier Select Register High (PMSH)	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
$0039	PLL Multiplier Select Register Low (PMSL)	Read:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
		Write:								
		Reset:	0	1	0	0	0	0	0	0
$003A	PLL VCO Range Select Register (PMRS)	Read:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
		Write:								
		Reset:	0	1	0	0	0	0	0	0
$003B	PLL Reference Divider Select Register (PMDS)	Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	1
$003C	ADC Status and Control Register (ADCSC)	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$003D	ADC Data Register high (ADRH)	Read:	0	0	0	0	0	0	0/AD9	0/AD8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$003E	ADC Data Register low (ADRL)	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$003F	ADC Clock Register (ADCLK)	Read:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ADACKEN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0040 ↓ $004E	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$004F	LCD Clock Register (LCDCLK)	Read:	0	FCCTL1	FCCTL0	DUTY1	DUTY0	LCLK2	LCLK1	LCLK0
		Write:	▨							
		Reset:	0	0	0	0	0	0	0	0
$0050	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								

U = Unaffected   X = Indeterminate   ▨ = Unimplemented   R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 8)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0051	LCD Control Register (LCDCR)	Read:	LCDE	0	FC	LC	LCCON3	LCCON2	LCCON1	LCCON0
		Write:		▨						
		Reset:	0	0	0	0	0	0	0	0
$0052	LCD Data Register (LDAT1)	Read:	F1B3	F1B2	F1B1	F1B0	F0B3	F0B2	F0B1	F0B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0053	LCD Data Register (LDAT2)	Read:	F3B3	F3B2	F3B1	F3B0	F2B3	F2B2	F2B1	F2B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0054	LCD Data Register (LDAT3)	Read:	F5B3	F5B2	F5B1	F5B0	F4B3	F4B2	F4B1	F4B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0055	LCD Data Register (LDAT4)	Read:	F7B3	F7B2	F7B1	F7B0	F6B3	F6B2	F6B1	F6B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0056	LCD Data Register (LDAT5)	Read:	F9B3	F9B2	F9B1	F9B0	F8B3	F8B2	F8B1	F8B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0057	LCD Data Register (LDAT6)	Read:	F11B3	F11B2	F11B1	F11B0	F10B3	F10B2	F10B1	F10B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0058	LCD Data Register (LDAT7)	Read:	F13B3	F13B2	F13B1	F13B0	F12B3	F12B2	F12B1	F12B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0059	LCD Data Register (LDAT8)	Read:	F15B3	F15B2	F15B1	F15B0	F14B3	F14B2	F14B1	F14B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005A	LCD Data Register (LDAT9)	Read:	F17B3	F17B2	F17B1	F17B0	F16B3	F16B2	F16B1	F16B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005B	LCD Data Register (LDAT10)	Read:	F19B3	F19B2	F19B1	F19B0	F18B3	F18B2	F18B1	F18B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U

U = Unaffected   X = Indeterminate   ▨ = Unimplemented   R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 8)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$005C	LCD Data Register (LDAT11)	Read:	F21B3	F21B2	F21B1	F21B0	F20B3	F20B2	F20B1	F20B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005D	LCD Data Register (LDAT12)	Read:	F23B3	F23B2	F23B1	F23B0	F22B3	F22B2	F22B1	F22B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005E	LCD Data Register (LDAT13)	Read:	0	0	0	0	F24B3	F24B2	F24B1	F24B0
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	U	U	U	U
$005F ↓ $007F	Unimplemented		▨	▨	▨	▨	▨	▨	▨	▨
$FE00	Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							See note	
		Reset:							0	

Note: Writing a logic 0 clears SBSW.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE01	Reset Status Register (SRSR)	Read:	POR	PIN	COP	ILOP	ILAD	0	LVI	0
		Write:	▨	▨	▨	▨	▨	▨	▨	▨
		POR:	1	0	0	0	0	0	0	0
$FE02	Reserved		R	R	R	R	R	R	R	R
$FE03	Break Flag Control Register (SBFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04	Interrupt Status Register 1 (INT1)	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
$FE05	Interrupt Status Register 2 (INT2)	Read:	0	0	0	0	0	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate ▨ = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 7 of 8)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE06	Interrupt Status Register 3 (INT3)	Read:	0	0	0	0	0	IF17	IF16	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE07	Reserved		R	R	R	R	R	R	R	R
$FE08	FLASH Control Register (FLCR)	Read:	0	0	0	0	HVEN	MASS	ERASE	PGM
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
$FE09 ↓ $FE0B	Reserved		R	R	R	R	R	R	R	R
$FE0C	Break Address Register High (BRKH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address Register Low (BRKL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Status and Control Register (BRKSCR)	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:			▨	▨	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$FE0F	Low-Voltage Inhibit Status Register (LVISR)	Read:	LVIOUT	LVIIE	LVIIF	0	0	0	0	0
		Write:	▨		▨	LVIIAK	▨	▨	▨	▨
		Reset:	0	0	0	0	0	0	0	0
$FF7E	FLASH Block Protect Register (FLBPR)(1)	Read:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		Write:								
		Reset:	Unaffected by reset; $FF when blank							

1. Non-volatile FLASH register; write by programming.

Addr.	Register Name		
$FFFF	COP Control Register (COPCTL)	Read:	Low byte of reset vector
		Write:	Writing clears COP counter (any value)
		Reset:	Unaffected by reset

U = Unaffected X = Indeterminate ▨ = Unimplemented R = Reserved

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

Table 2-1. Vector Addresses

Vector Priority	INT Flag	Address	Vector
Lowest	IF17	$FFDA	ADC conversion complete
		$FFDB	
	IF16	$FFDC	KBI
		$FFDD	
	—	$FFDE ↓ $FFE9	Not used
	IF9	$FFEA	TIM2 overflow
		$FFEB	
	IF8	$FFEC	TIM2 channel 1
		$FFED	
	IF7	$FFEE	TIM2 channel 0
		$FFEF	
	IF6	$FFF0	TIM1 overflow
		$FFF1	
	IF5	$FFF2	TIM1 channel 1
		$FFF3	
	IF4	$FFF4	TIM1 channel 0
		$FFF5	
	IF3	$FFF6	PLL
		$FFF7	
	IF2	$FFF8	LVI
		$FFF9	
	IF1	$FFFA	$\overline{\text{IRQ}}$
		$FFFB	
	—	$FFFC	SWI
		$FFFD	
	—	$FFFE	Reset
Highest		$FFFF	

2.4 Random-Access Memory (RAM)

The 512 bytes RAM are located from $0080 through $027F. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64-Kbyte memory space.

NOTE

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

Within page zero are 128 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for I/O control and user data or code. When the stack pointer is moved from its reset location at $00FF, direct addressing mode instructions can access efficiently all page zero RAM locations. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

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

NOTE

For M6805 compatibility, the H register is not stacked.

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

NOTE

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

2.5 FLASH Memory

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

2.5.1 Functional Description

The FLASH memory consists of an array of 8,192 bytes for user memory plus a block of 48 bytes for user interrupt vectors. *An erased bit reads as 1 and a programmed bit reads as a 0*. The FLASH memory page size is defined as 64 bytes, and is the minimum size that can be erased in a page erase operation. Program and erase operations are facilitated through control bits in FLASH control register (FLCR). The address ranges for the FLASH memory are:

- $DE00–$FDFF; user memory; 8,192 bytes
- $FFD0–$FFFF; user interrupt vectors; 48 bytes

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

NOTE

A security feature prevents viewing of the FLASH contents.(1)

---

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

2.6 FLASH Control Register

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

Address: $FE08

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

▨ = Unimplemented

Figure 2-3. FLASH Control Register (FLCR)

HVEN — High Voltage Enable Bit

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

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

MASS — Mass Erase Control Bit

This read/write bit configures the memory for mass erase operation or page erase operation when the ERASE bit is set.

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

ERASE — Erase Control Bit

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

1 = Erase operation selected
0 = Erase operation not selected

PGM — Program Control Bit

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

1 = Program operation selected
0 = Program operation not selected

2.6.1 FLASH Page Erase Operation

Use the following procedure to erase a page of FLASH memory. A page consists of 64 consecutive bytes starting from addresses $xx00, $xx40, $xx80 or $xxC0. The 48-byte user interrupt vectors area also forms a page. Any FLASH memory page can be erased alone.

1. Set the ERASE bit and clear the MASS bit in the FLASH control register.
2. Read the FLASH block protection register, FLBPR.
3. Write any data to any FLASH address within the page address range desired.
4. Wait for a time, t_{nvs} (min. 10μs).
5. Set the HVEN bit.
6. Wait for a time, t_{erase} (1 ms).
7. Clear the ERASE bit.
8. Wait for a time, t_{nvh} (5 μs).
9. Clear the HVEN bit.
10. After time, t_{rcv} (1 μs), the memory can be accessed again in read mode.

NOTE

The COP register at location $FFFF should only be serviced after step 5.

NOTE

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

2.6.2 FLASH Mass Erase Operation

Use the following procedure to erase the entire FLASH memory:

1. Set both the ERASE bit and the MASS bit in the FLASH control register.
2. Read the FLASH block protection register, FLBPR.
3. Write any data to any FLASH address within the FLASH memory address range.
4. Wait for a time, t_{nvs} (10 μs).
5. Set the HVEN bit.
6. Wait for a time t_{merase} (4 ms).
7. Clear the ERASE bit.
8. Wait for a time, t_{nvhl} (100 μs).
9. Clear the HVEN bit.
10. After time, t_{rcv} (1 μs), the memory can be accessed again in read mode.

NOTE

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

NOTE

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

2.6.3 FLASH Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 32 consecutive bytes starting from addresses $xx00, $xx20, $xx40, $xx60, $xx80, $xxA0, $xxC0 or $xxE0.

NOTE

Do not program any byte in the FLASH more than once after a successful erase operation. Reprogramming bits to a byte which is already programmed is not allowed without first erasing the page in which the byte resides or mass erasing the entire FLASH memory. Programming without first erasing may disturb data stored in the FLASH.

The procedure for programming a row of the FLASH memory is outlined below:

1. Set the PGM bit. This configures the memory for program and enables latching of address and data for programming.
2. Read the FLASH block protection register, FLBPR.
3. Write any data to any FLASH address within the row address range desired.
4. Wait for a time, t_{nvs} (10 μs).
5. Set the HVEN bit.
6. Wait for a time, t_{pgs} (5 μs).
7. Write data to the FLASH address to be programmed.
8. Wait for time, t_{prog} (30 μs).
9. Repeat step 7 and 8 until all the bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for time, t_{nvh} (5 μs).
12. Clear the HVEN bit.
13. After time, t_{rcv} (1 μs), the memory can be accessed again in read mode.

Figure 2-4 shows a flowchart representation for programming the FLASH memory.

This program sequence is repeated throughout the memory until all data is programmed.

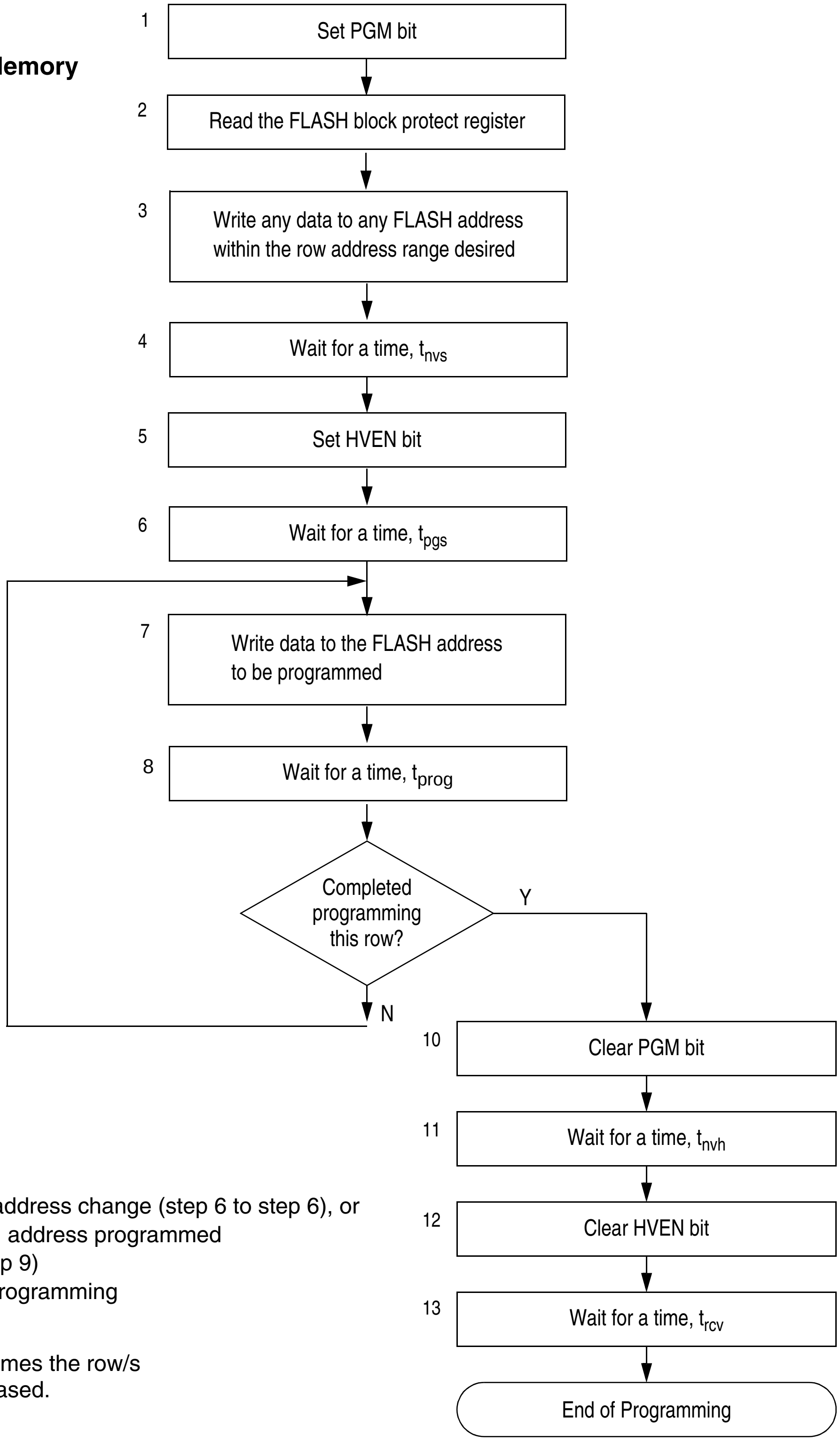
NOTE

The COP register at location $FFFF should not be written between steps 5 and 12, when the HVEN bit is set. Since this register is located at a valid FLASH address, unpredictable behavior may occur if this location is written while HVEN is set.

NOTE

Programming and erasing of FLASH locations cannot be performed by executing code from the FLASH memory; the code must be executed from RAM. While these operations must be performed in the order as shown, but other unrelated operations may occur between the steps. Do not exceed t_{prog} *maximum.*

Algorithm for programming a row (32 bytes) of FLASH Memory



NOTE:

The time between each FLASH address change (step 6 to step 6), or the time between the last FLASH address programmed to clearing PGM bit (step 6 to step 9) must not exceed the maximum programming time, t_{PROG} max.

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

Figure 2-4. FLASH Programming Flowchart

2.7 FLASH Protection

Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made to protect pages of memory from unintentional erase or program operations due to system malfunction. This protection is done by use of a FLASH Block Protect Register (FLBPR). The FLBPR determines the range of the FLASH memory which is to be protected. The range of the protected area starts from a location defined by FLBPR and ends to the bottom of the FLASH memory ($FFFF). When the memory is protected, the HVEN bit cannot be set in either ERASE or PROGRAM operations.

NOTE

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

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

When bits within the FLBPR are programmed, they lock a block of memory. The address ranges are shown in 2.7.1 FLASH Block Protect Register. Once the FLBPR is programmed with a value other than $FF, any erase or program of the FLBPR or the protected block of FLASH memory is prohibited. Mass erase is disabled whenever any block is protected (FLBPR does not equal $FF).

The FLBPR itself can be erased or programmed only with an external voltage, VTST, present on the IRQ pin. This voltage also allows entry from reset into the monitor mode.

2.7.1 FLASH Block Protect Register

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

Address: $FF7E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
Reset:	Unaffected by reset; $FF when blank							

Non-volatile FLASH register; write by programming.

Figure 2-5. FLASH Block Protect Register (FLBPR)

BPR[7:0] — FLASH Block Protect Bits

BPR[7:0] represent bits [13:6] of a 16-bit memory address. Bits [15:14] are 1s, and bits [5:0] are 0s.

16-bit memory address

Start address of FLASH block protect																
	1	1									0	0	0	0	0	0
	▨	▨	BPR[7:0]								▨	▨	▨	▨	▨	▨

The resultant 16-bit address is used for specifying the start address of the FLASH memory for block protection. The FLASH is protected from this start address to the end of FLASH memory, at $FFFF. With this mechanism, the protect start address can be $XX00, $XX40, $XX80 or $XXC0 (at page boundaries — 64 bytes) within the FLASH memory.

Table 2-2. FLASH Block Protection Register to Physical Address

BPR[7:0]	Start Address of Protection Range
$00–$78	The entire FLASH memory is protected.
$79	$DE40 (11**01 1110 01**00 0000)
$7A	$DE80 (11**01 1110 10**00 0000)
$7B	$DEC0 (11**01 1110 11**00 0000)
$7C	$DF00 (11**01 1111 00**00 0000)
and so on...	
$B8	$EE00 (11**10 1110 00**00 0000)
and so on...	
$F7	$FDC0 (11**11 1101 11**00 0000)
$F8	$FE00 (11**11 1110 00**00 0000)
and so on...	
$FC	$FF00 (11**11 1111 00**00 0000)
$FD	$FF40 (11**11 1111 01**00 0000)
$FE	$FF80 (11**11 1111 10**00 0000)
$FF	The entire FLASH memory is not protected.

Chapter 3
Configuration Register (CONFIG)

3.1 Introduction

This section describes the configuration registers, CONFIG1 and CONFIG2.

The configuration registers enable or disable these options:
- Computer operating properly module (COP)
- COP timeout period ($2^{13}–2^4$ or $2^{18}–2^4$ CGMXCLK cycles)
- Crystal oscillator during stop mode
- Low voltage inhibit (LVI) module power
- LVI module reset
- LVI module in stop mode
- LVI module voltage trip point selection
- STOP instruction
- Stop mode recovery time (32 or 4096 CGMXCLK cycles)
- LCD frontplanes FP3–FP10 on port E
- LCD frontplanes FP11–FP18 on port D
- LCD frontplanes FP19–FP24 on port C

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001D	Configuration Register 2 (CONFIG2)[1]	Read: Write:	STOP_ XCLKEN	R	PEE	PDE	PCEH	PCEL	LVISEL1	LVISEL0
		Reset:	0	0	0	0	0	0	0[2]	1[2]
$001F	Configuration Register 1 (CONFIG1)[1]	Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	R	SSREC	STOP	COPD
		Reset:	0	0	0	0[2]	0	0	0	0

1. One-time writable register after each reset.
2. LVIT1, LVIT0, and LVIPWRD reset to 0 by a power-on reset (POR) only.

R = Reserved

Figure 3-1. CONFIG Registers Summary

3.2 Functional Description

The configuration registers are used in the initialization of various options. The configuration registers can be written once after each reset. All of the configuration register bits are cleared during reset. Since the various options affect the operation of the MCU, it is recommended that these registers be written immediately after reset. The configuration registers are located at \$001D and \$001F. The configuration registers may be read at anytime.

> ***NOTE***
>
> *The options except LVIT[1:0] and LVIPWRD are one-time writable by the user after each reset. The LVIT[1:0] and LVIPWRD bits are one-time writable by the user only after each POR (power-on reset). The CONFIG registers are not in the FLASH memory but are special registers containing one-time writable latches after each reset. Upon a reset, the CONFIG registers default to predetermined settings as shown in Figure 3-2 and Figure 3-3.*

The mask option register (MOR) is used to select the oscillator option for the MCU: crystal oscillator or RC oscillator. The MOR is implemented as a byte in FLASH memory. Hence, writing to the MOR requires programming the byte.

3.3 Configuration Register 1 (CONFIG1)

Address: \$001F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	R	SSREC	STOP	COPD
Reset:	0	0	0	U	0	0	0	0
POR:	0	0	0	0	0	0	0	0

R = Reserved   U = Unaffected

Figure 3-2. Configuration Register 1 (CONFIG1)

COPRS — COP Rate Select

COPRS selects the COP time-out period. Reset clears COPRS.

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

LVISTOP — Low Voltage Inhibit Enable in Stop Mode

When the LVIPWRD bit is clear, setting the LVISTOP bit enables the LVI to operate during stop mode. Reset clears LVISTOP.

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

LVIRSTD — Low Voltage Inhibit Reset Disable

LVIRSTD disables the reset signal from the LVI module. Reset clears LVIRSTOP.

1 = LVI module reset disabled
0 = LVI module reset enabled

LVIPWRD — Low Voltage Inhibit Power Disable

LVIPWRD disables the LVI module. This bit is reset to 0 by a POR only.

1 = LVI module disabled
0 = LVI module enabled

NOTE

Exiting stop mode by pulling reset will result in the long stop recovery. If using an external crystal, do not set the SSREC bit.

SSREC — Short Stop Recovery Bit

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

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

STOP — STOP Instruction Enable Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. Reset clears COPD.

1 = COP module disabled
0 = COP module enabled

3.4 Configuration Register 2 (CONFIG2)

Address: $001D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	STOP_ XCLKEN	R	PEE	PDE	PCEH	PCEL	LVISEL1	LVISEL0
Reset:	0	0	0	0	0	0	U	U
POR:	0	0	0	0	0	0	0	1

R = Reserved   U = Unaffected

Figure 3-3. Configuration Register 2 (CONFIG2)

STOP_XCLKEN — Crystal Oscillator Stop Mode Enable

Setting STOP_XCLKEN enables the crystal oscillator to continue operating during stop mode. Reset clears this bit.

1 = Crystal oscillator enabled during stop mode
0 = Crystal oscillator disabled during stop mode

PEE — Port E Enable for LCD Drive

Setting PEE configures the PTE0/FP3–PTE7/FP10 pins for LCD frontplane driver use. Reset clears this bit.

1 = PTE0/FP3–PTE7/FP10 pins configured as LCD frontplane driver pins: FP3–FP10
0 = PTE0/FP3–PTE7/FP10 pins configured as standard I/O pins: PTE0–PTE7

PDE — Port D Enable for LCD Drive

Setting PDE configures the PTD0/FP11–PTD7/FP18 pins for LCD frontplane driver use. Reset clears this bit.

1 = PTD0/FP11–PTD7/FP18 pins configured as LCD frontplane driver pins: FP11–FP18
0 = PTD0/FP11–PTD7/FP18 pins configured as standard I/O pins: PTD0–PTD7

PCEH — Port C High Nibble Enable for LCD Drive

Setting PCEH configures the PTC4/FP23–PTC5/FP24 pins for LCD frontplane driver use. Reset clears this bit.

1 = PTC4/FP23–PTC5/FP24 pins configured as LCD frontplane driver pins: FP23–FP24
0 = PTC4/FP23–PTC5/FP24 pins configured as standard I/O pins: PTC4–PTC5

PCEL — Port C Low Nibble Enable for LCD Drive

Setting PCEL configures the PTC0/FP19–PTC3/FP22 pins for LCD frontplane driver use. Reset clears this bit.

1 = PTC0/FP19–PTC3/FP22 pins configured as LCD frontplane driver pins: FP19–FP22
0 = PTC0/FP19–PTC3/FP22 pins configured as standard I/O pins: PTC0–PTC3

LVISEL1, LVISEL0 — LVI Trip Voltage Selection

These two bits determine at which level of V_{DD} the LVI module will come into action. LVISEL1 and LVISEL0 are cleared by a power-on reset only.

Table 3-1. Trip Voltage Selection

LVISEL1	LVISEL0	Comments[1]
0	0	Reserved
0	1	For V_{DD} = 3 V operation
1	0	For V_{DD} = 5 V operation
1	1	Reserved

1. See Chapter 17 Electrical Specifications for full parameters.

Chapter 4
System Integration Module (SIM)

4.1 Introduction

This section describes the system integration module (SIM). Together with the CPU, the SIM controls all MCU activities. The SIM is a system state controller that coordinates CPU and exception timing. The SIM is responsible for:

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

Table 4-1. Signal Name Conventions

Signal Name	Description
CGMXCLK	Selected oscillator clock from oscillator module
CGMVCLK	PLL output
CGMOUT	PLL-based or oscillator-based clock output from CGM module (Bus clock = CGMOUT ÷ 2)
IAB	Internal address bus
IDB	Internal data bus
PORRST	Signal from the power-on reset module to the SIM
IRST	Internal reset signal
R/$\overline{\text{W}}$	Read/write signal

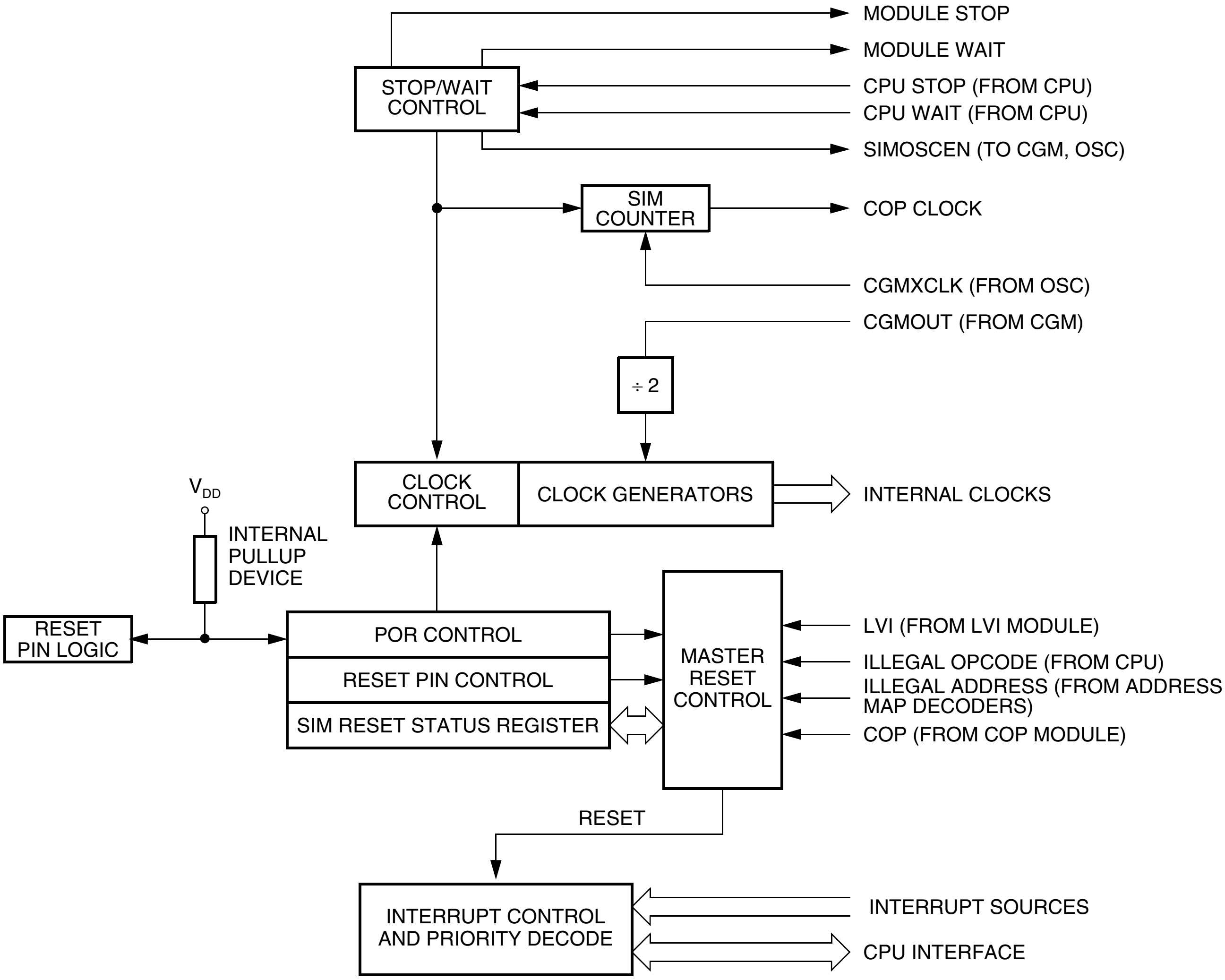


Figure 4-1. SIM Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	Break Status Register (BSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							NOTE	
		Reset:	0	0	0	0	0	0	0	0

Note: Writing a 0 clears SBSW.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE01	Reset Status Register (RSR)	Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
$FE02	Reserved		R	R	R	R	R	R	R	R
$FE03	Break Flag Control Register (BFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							

Figure 4-2. SIM I/O Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE04	Interrupt Status Register 1 (INT1)	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
$FE05	Interrupt Status Register 2 (INT2)	Read:	0	0	0	0	0	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE06	Interrupt Status Register 3 (INT3)	Read:	0	0	0	0	0	IF17	IF16	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented R = Reserved

Figure 4-2. SIM I/O Register Summary (Continued)

4.2 SIM Bus Clock Control and Generation

The bus clock generator provides system clock signals for the CPU and peripherals on the MCU. The system clocks are generated from an incoming clock, CGMOUT, as shown in Figure 4-3. This clock can come from either the oscillator module or from the on-chip PLL. (See Chapter 5 Clock Generator Module (CGM).)

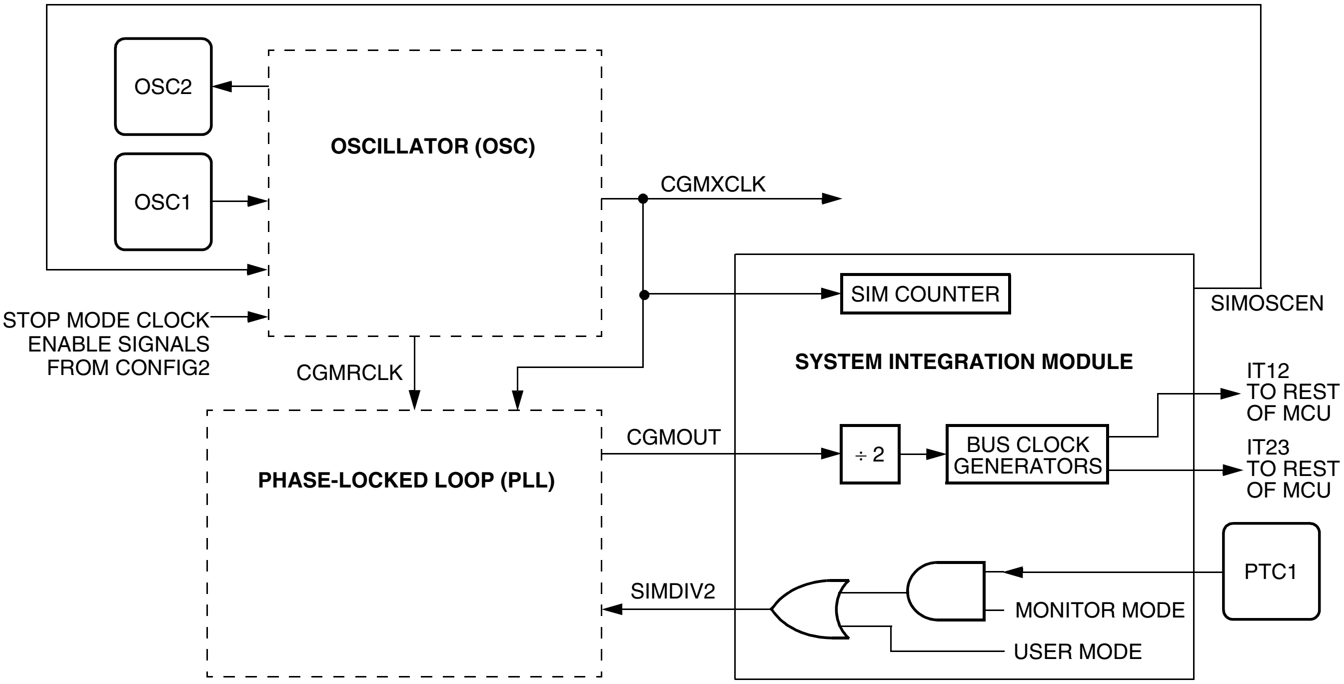


Figure 4-3. CGM Clock Signals

4.2.1 Bus Timing

In user mode, the internal bus frequency is either the oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four.

4.2.2 Clock Start-up from POR or LVI Reset

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

4.2.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. (See 4.6.2 Stop Mode.)

In wait mode, the CPU clocks are inactive. The SIM also produces two sets of clocks for other modules. Refer to the wait mode subsection of each module to see if the module is active or inactive in wait mode. Some modules can be programmed to be active in wait mode.

4.3 Reset and System Initialization

The MCU has these reset sources:

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

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

An internal reset clears the SIM counter (see 4.4 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See 4.7 SIM Registers.)

4.3.1 External Pin Reset

The $\overline{\text{RST}}$ pin circuit includes an internal pull-up device. Pulling the asynchronous $\overline{\text{RST}}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{\text{RST}}$ is held low for a minimum of 67 CGMXCLK cycles, assuming that neither the POR nor the LVI was the source of the reset. See Table 4-2 for details. Figure 4-4 shows the relative timing.

Table 4-2. PIN Bit Set Timing

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

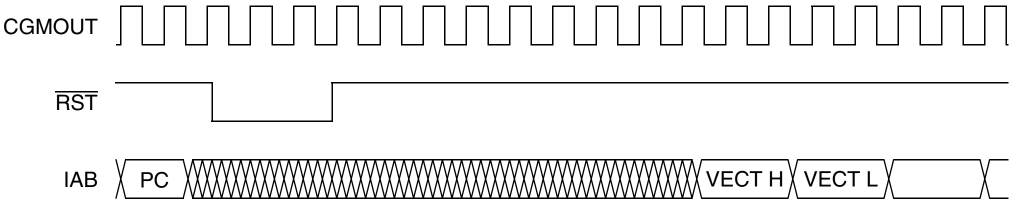


Figure 4-4. External Reset Timing

4.3.2 Active Resets from Internal Sources

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see Figure 4-5). An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR (see Figure 4-6).

NOTE

For LVI or POR resets, the SIM cycles through 4096 + 32 CGMXCLK cycles during which the SIM forces the $\overline{\text{RST}}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ shown in Figure 4-5.

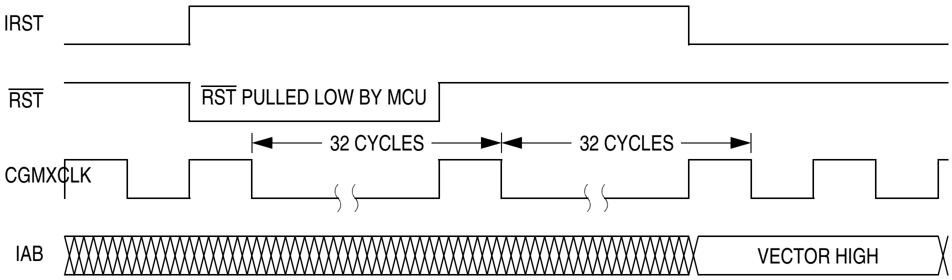


Figure 4-5. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

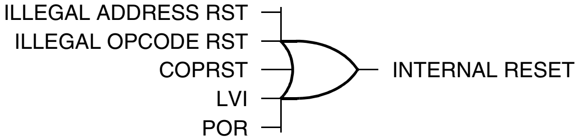


Figure 4-6. Sources of Internal Reset

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

4.3.2.1 Power-On Reset

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power-on has occurred. The external reset pin ($\overline{\text{RST}}$) is held low while the SIM counter counts out 4096 + 32 CGMXCLK cycles. Thirty-two CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, these events occur:

- A POR pulse is generated.
- The internal reset signal is asserted.

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

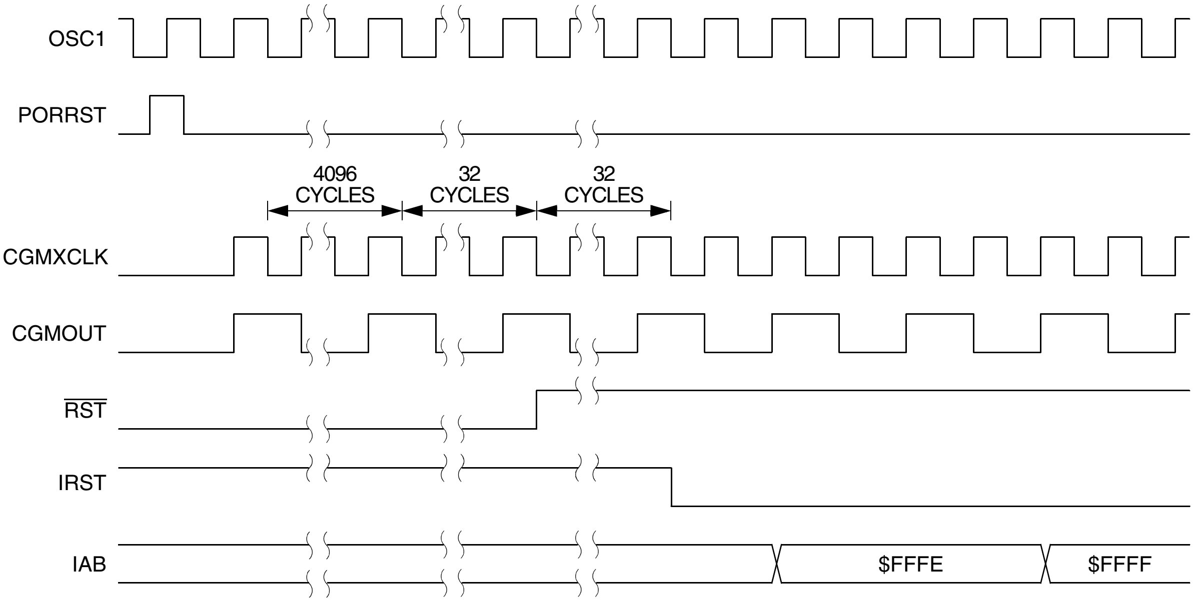


Figure 4-7. POR Recovery

4.3.2.2 Computer Operating Properly (COP) Reset

An input to the SIM is reserved for the COP reset signal. The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR). The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

To prevent a COP module timeout, write any value to location $FFFF. Writing to location $FFFF clears the COP counter and bits 12 through 5 of the SIM counter. The SIM counter output, which occurs at least every $2^{13} - 2^4$ CGMXCLK cycles, drives the COP counter. The COP should be serviced as soon as possible out of reset to guarantee the maximum amount of time before the first timeout.

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

4.3.2.3 Illegal Opcode Reset

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

If the stop enable bit, STOP, in the mask option register is logic 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

4.3.2.4 Illegal Address Reset

An opcode fetch from an unmapped address generates an illegal address reset. The SIM verifies that the CPU is fetching an opcode prior to asserting the ILAD bit in the SIM reset status register (SRSR) and resetting the MCU. A data fetch from an unmapped address does not generate a reset. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

4.3.2.5 Low-Voltage Inhibit (LVI) Reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the LVI_{TRIPF} voltage. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{\text{RST}}$) is held low while the SIM counter counts out 4096 + 32 CGMXCLK cycles. Thirty-two CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

4.3.2.6 Monitor Mode Entry Module Reset (MODRST)

The monitor mode entry module reset (MODRST) asserts its output to the SIM when monitor mode is entered in the condition where the reset vectors are blank ($FF). (See Chapter 16 Development Support.) When MODRST gets asserted, an internal reset occurs. The SIM actively pulls down the $\overline{\text{RST}}$ pin for all internal reset sources.

4.4 SIM Counter

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

4.4.1 SIM Counter During Power-On Reset

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

4.4.2 SIM Counter During Stop Mode Recovery

The SIM counter also is used for stop mode recovery. The STOP instruction clears the SIM counter. After an interrupt, break, or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the configuration register 1 (CONFIG1). If the SSREC bit is a logic 1, then the stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32 CGMXCLK cycles. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode. External crystal applications should use the full stop recovery time, that is, with SSREC cleared.

4.4.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See 4.6.2 Stop Mode for details.) The SIM counter is free-running after all reset states. (See 4.3.2 Active Resets from Internal Sources for counter control and internal reset recovery sequences.)

4.5 Exception Control

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

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

4.5.1 Interrupts

At the beginning of an interrupt, the CPU saves the CPU register contents on the stack and sets the interrupt mask (I bit) to prevent additional interrupts. At the end of an interrupt, the RTI instruction recovers the CPU register contents from the stack so that normal processing can resume. Figure 4-8 shows interrupt entry timing, and Figure 4-9 shows interrupt recovery timing.

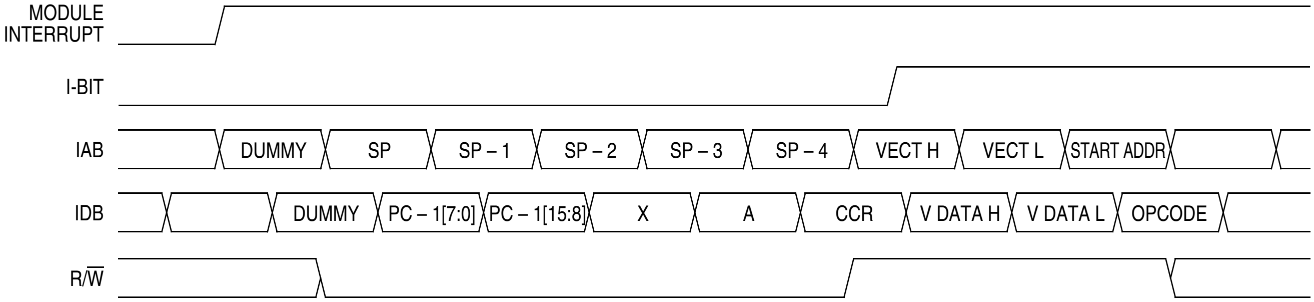


Figure 4-8. Interrupt Entry Timing

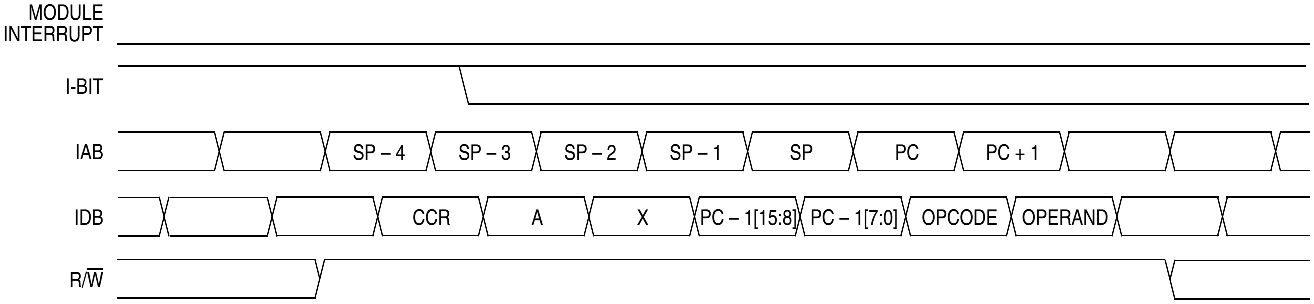


Figure 4-9. Interrupt Recovery Timing

Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared).
(See Figure 4-10.)

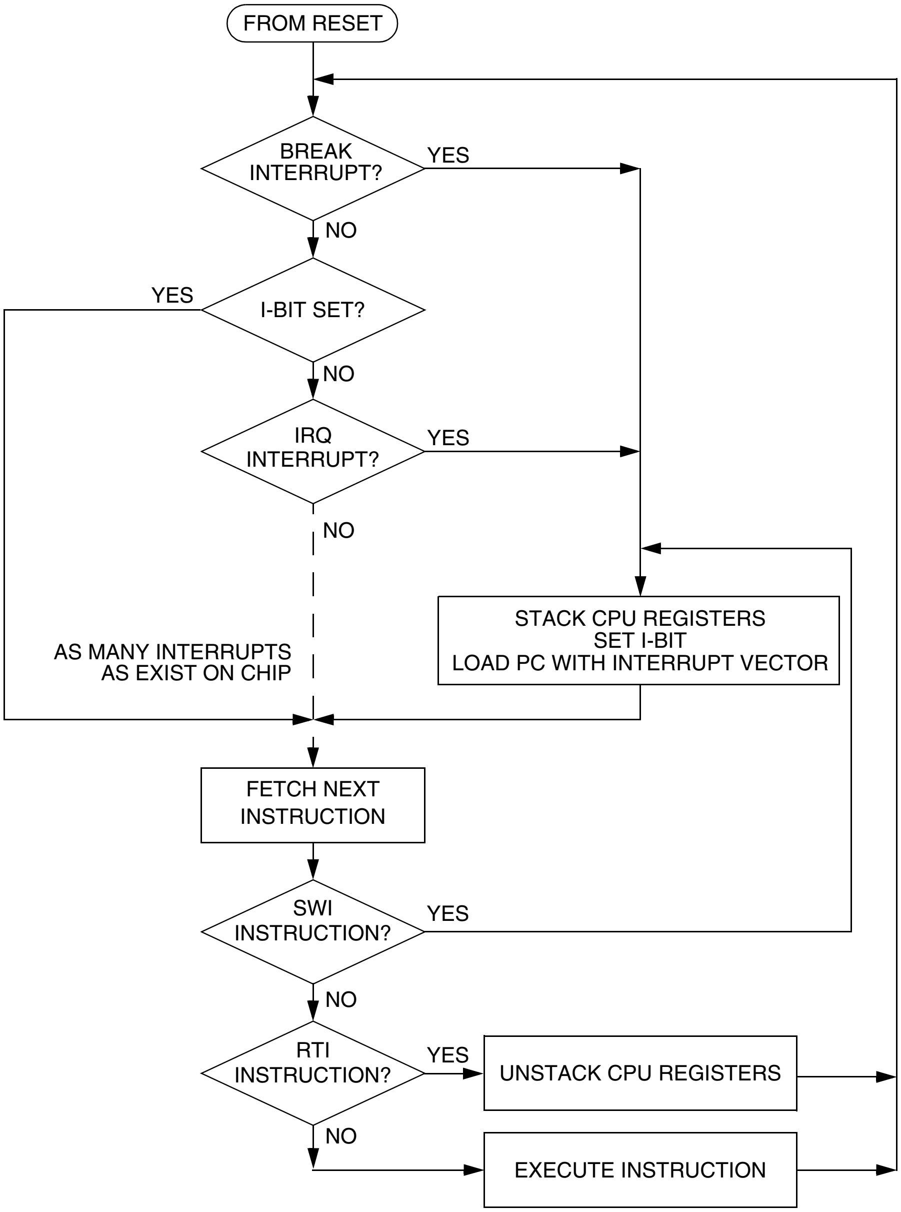


Figure 4-10. Interrupt Processing

4.5.1.1 Hardware Interrupts

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

If more than one interrupt is pending at the end of an instruction execution, the highest priority interrupt is serviced first. Figure 4-11 demonstrates what happens when two interrupts are pending. If an interrupt is pending upon exit from the original interrupt service routine, the pending interrupt is serviced before the LDA instruction is executed.

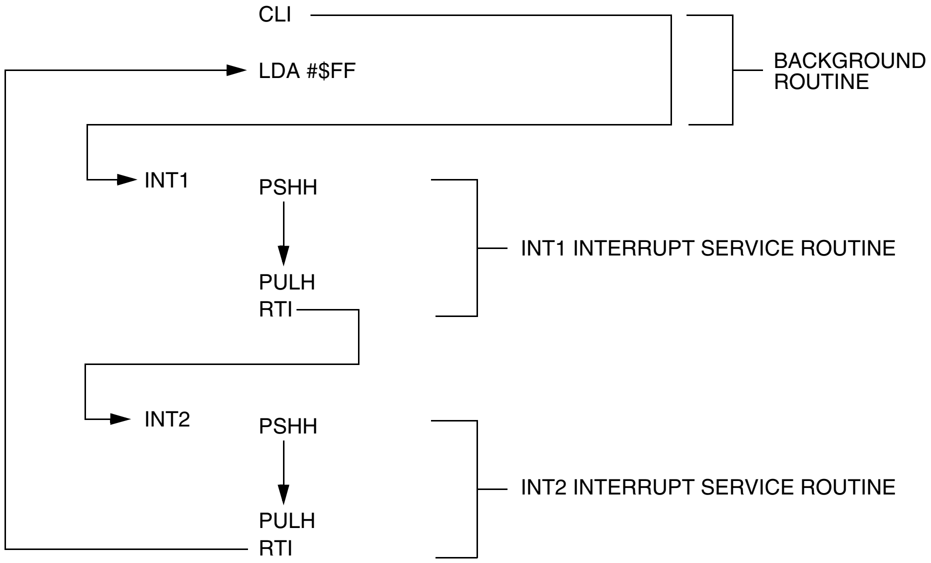


Figure 4-11. Interrupt Recognition Example

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

NOTE

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

4.5.1.2 SWI Instruction

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

NOTE

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

4.5.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources.

Table 2-1 summarizes the interrupt sources and the interrupt status register flags that they set.

4.5.2.1 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 4-12. Interrupt Status Register 1 (INT1)

IF6–IF1 — Interrupt Flags 1–6

These flags indicate the presence of interrupt requests from the sources shown in Table 2-1.

1 = Interrupt request present
0 = No interrupt request present

Bit 0 and Bit 1 — Always read 0

4.5.2.2 Interrupt Status Register 2

Address: $FE05

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	IF9	IF8	IF7
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 4-13. Interrupt Status Register 2 (INT2)

IF9–IF7 — Interrupt Flags 9–7

These flags indicate the presence of interrupt requests from the sources shown in Table 2-1.

1 = Interrupt request present
0 = No interrupt request present

4.5.2.3 Interrupt Status Register 3

Address: $FE06

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	IF17	IF16	0
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 4-14. Interrupt Status Register 3 (INT3)

IF17–IF16 — Interrupt Flags 17–16

These flags indicate the presence of an interrupt request from the source shown in Table 2-1.

1 = Interrupt request present
0 = No interrupt request present

4.5.3 Reset

All reset sources always have equal and highest priority and cannot be arbitrated.

4.5.4 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See Chapter 16 Development Support.) The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

4.5.5 Status Flag Protection in Break Mode

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

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

Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a 2-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.

4.6 Low-Power Modes

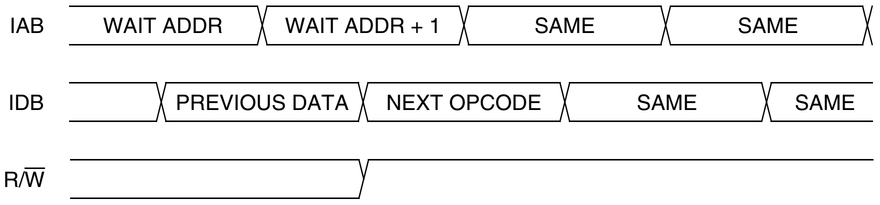
Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described in the following subsections. Both STOP and WAIT clear the interrupt mask (I) in the condition code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

4.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 4-15 shows the timing for wait mode entry.

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

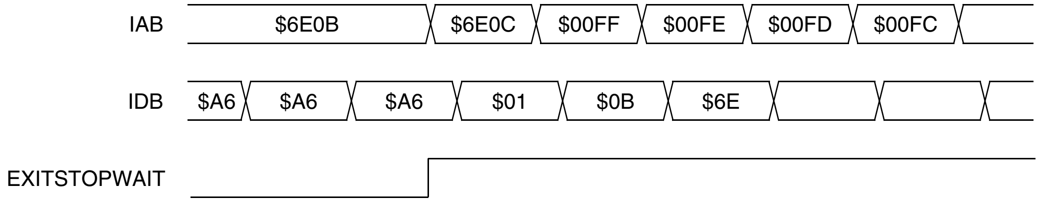
Wait mode also can be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is logic 0, then the computer operating properly module (COP) is enabled and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



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

Figure 4-15. Wait Mode Entry Timing

Figure 4-16 and Figure 4-17 show the timing for WAIT recovery.



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

Figure 4-16. Wait Recovery from Interrupt or Break

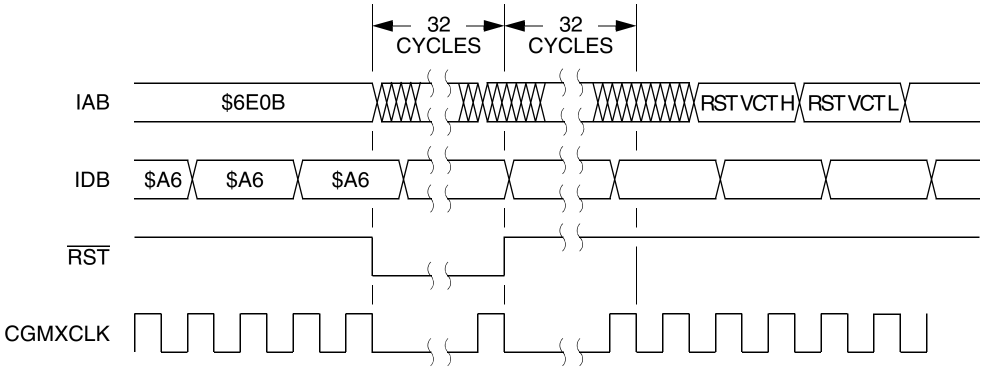


Figure 4-17. Wait Recovery from Internal Reset

4.6.2 Stop Mode

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

The SIM disables the clock generator module output (CGMOUT) in stop mode, stopping the CPU and peripherals. Stop recovery time is selectable using the SSREC bit in the configuration register 1 (CONFIG1). If SSREC is set, stop recovery is reduced from the normal delay of 4096 CGMXCLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode.

NOTE

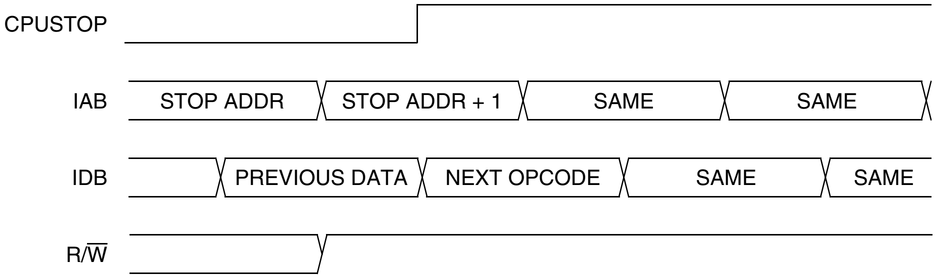
External crystal applications should use the full stop recovery time by clearing the SSREC bit.

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

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 4-18 shows stop mode entry timing.

NOTE

To minimize stop current, all pins configured as inputs should be driven to a logic 1 or logic 0.



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

Figure 4-18. Stop Mode Entry Timing

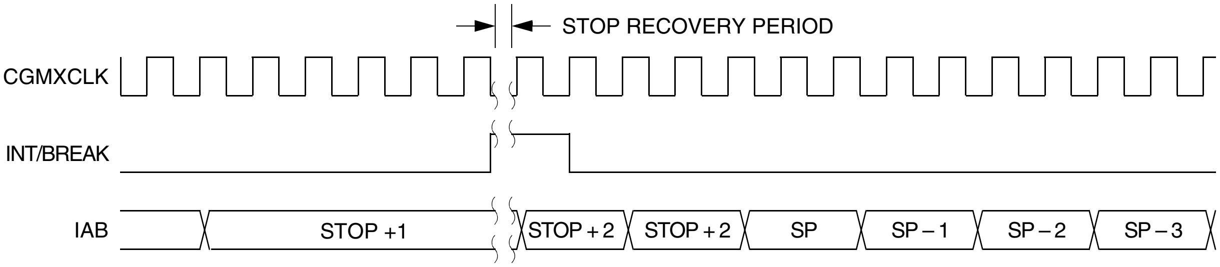


Figure 4-19. Stop Mode Recovery from Interrupt or Break

4.7 SIM Registers

The SIM has three memory-mapped registers:

- SIM Break Status Register (SBSR)
- SIM Reset Status Register (SRSR)
- SIM Break Flag Control Register (SBFCR)

4.7.1 SIM Break Status Register

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

Address: $FE00

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	R	R	R	R	R	SBSW	R
Write:							Note(1)	
Reset:							0	

R = Reserved

1. Writing a clears SBSW.

Figure 4-20. Break Status Register (BSR)

SBSW — Break Wait Bit

This status bit is set when a break interrupt causes an exit from wait mode or stop mode. Clear SBSW by writing a logic 0 to it. Reset clears SBSW.

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

SBSW can be read within the break 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 has been pushed onto the stack in the break
service routine software. This code should be executed at the end of the break
service routine software.

HIBYTE   EQU

LOBYTE   EQU

         If not SBSW, do RTI

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

         TST     LOBYTE,SP            ;If RETURNLO is not zero,

         BNE     DOLO                 ;then just decrement low byte.

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

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

RETURN   PULH                         ;Restore H register.
         RTI
```

4.7.2 SIM Reset Status Register

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

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 4-21. Reset Status Register (RSR)

POR — Power-On Reset Bit
1 = Last reset caused by POR circuit
0 = Read of SRSR

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

COP — Computer Operating Properly Reset Bit
1 = Last reset caused by COP counter
0 = POR or read of SRSR

ILOP — Illegal Opcode Reset Bit
1 = Last reset caused by an illegal opcode
0 = POR or read of SRSR

ILAD — Illegal Address Reset Bit (opcode fetches only)
1 = Last reset caused by an opcode fetch from an illegal address
0 = POR or read of SRSR

LVI — Low-Voltage Inhibit Reset Bit
1 = Last reset caused by the LVI circuit
0 = POR or read of SRSR

4.7.3 SIM Break Flag Control Register

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 4-22. Break Flag Control Register (BFCR)

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

BCFE — Break Clear Flag Enable Bit
This read/write bit enables software to clear status bits by accessing status registers while the MCU is in a break state. To clear status bits during the break state, the BCFE bit must be set.
1 = Status bits clearable during break
0 = Status bits not clearable during break

Chapter 5
Clock Generator Module (CGM)

5.1 Introduction

This section describes the clock generator module (CGM). The CGM generates the base clock signal, CGMOUT, which is based on either the oscillator clock divided by two or the phase-locked loop (PLL) clock, CGMVCLK, divided by two. CGMOUT is the clock from which the SIM derives the system clocks, including the bus clock, which is at a frequency of CGMOUT÷2.

The PLL is a frequency generator designed for use with a low frequency crystal (typically 32.768 kHz) to generate a base frequency and dividing to a maximum bus frequency of 8 MHz.

5.2 Features

Features of the CGM include:
- Phase-locked loop with output frequency in integer multiples of an integer dividend of the crystal reference
- Low-frequency crystal operation with low-power operation and high-output frequency resolution
- Programmable prescaler for power-of-two increases in frequency
- 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
- Configuration register bit to allow oscillator operation during stop mode

5.3 Functional Description

The CGM consists of three major sub-modules:
- Crystal oscillator module — The crystal oscillator module generates the constant reference frequency clock, CGMRCLK (buffered CGMXCLK).
- Phase-locked loop (PLL) — The PLL generates the programmable VCO frequency clock, CGMVCLK.
- Base clock selector circuit — This software-controlled circuit selects either CGMXCLK divided by two or the VCO clock, CGMVCLK, divided by two as the base clock, CGMOUT. The SIM derives the system clocks from either CGMOUT or CGMXCLK.

Figure 5-1 shows the structure of the CGM.

Figure 5-2 is a summary of the CGM registers.

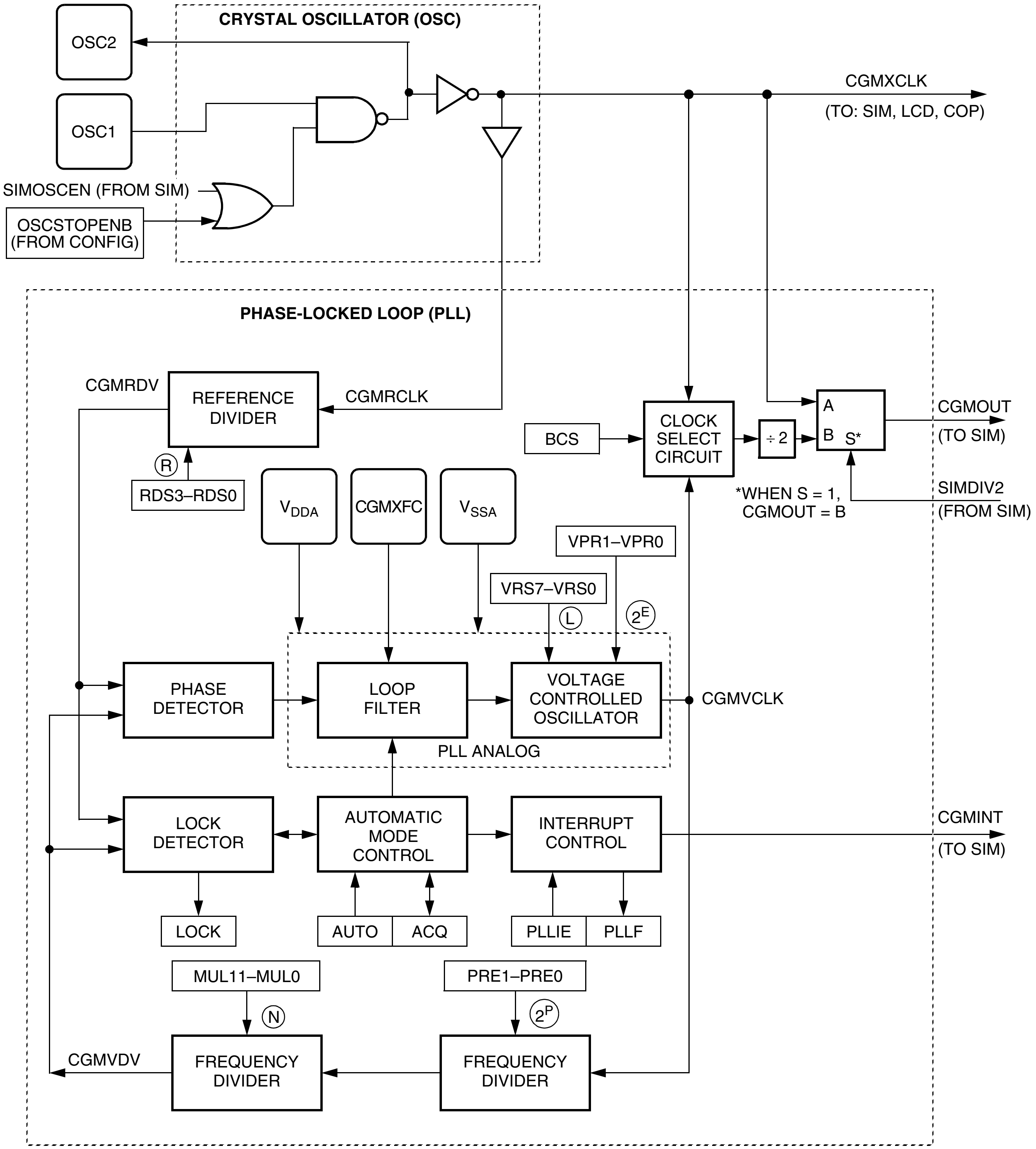


Figure 5-1. CGM Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0036	PLL Control Register (PTCL)	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		Write:		▨						
		Reset:	0	0	1	0	0	0	0	0
$0037	PLL Bandwidth Control Register (PBWC)	Read:	AUTO	LOCK	$\overline{\text{ACQ}}$	0	0	0	0	R
		Write:		▨		▨	▨	▨	▨	
		Reset:	0	0	0	0	0	0	0	0
$0038	PLL Multiplier Select Register High (PMSH)	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	0
$0039	PLL Multiplier Select Register Low (PMSL)	Read:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
		Write:								
		Reset:	0	1	0	0	0	0	0	0
$003A	PLL VCO Range Select Register (PMRS)	Read:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
		Write:								
		Reset:	0	1	0	0	0	0	0	0
$003B	PLL Reference Divider Select Register (PMDS)	Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	0	0	0	1

▨ = Unimplemented R = Reserved

NOTES:
1. When AUTO = 0, PLLIE is forced clear and is read-only.
2. When AUTO = 0, PLLF and LOCK read as clear.
3. When AUTO = 1, $\overline{\text{ACQ}}$ is read-only.
4. When PLLON = 0 or VRS7:VRS0 = $0, BCS is forced clear and is read-only.
5. When PLLON = 1, the PLL programming register is read-only.
6. When BCS = 1, PLLON is forced set and is read-only.

Figure 5-2. CGM I/O Register Summary

5.3.1 Crystal Oscillator Circuit

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

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

5.3.2 Phase-Locked Loop Circuit (PLL)

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

5.3.3 PLL Circuits

The PLL consists of these circuits:

- Voltage-controlled oscillator (VCO)
- Reference divider
- Frequency prescaler
- 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}, (38.4 kHz) times a linear factor, L, and a power-of-two factor, E, or $(L \times 2^E)f_{NOM}$.

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 programmable modulo reference divider, which divides f_{RCLK} by a factor, R. The divider's output is the final reference clock, CGMRDV, running at a frequency, $f_{RDV} = f_{RCLK}/R$. With an external crystal (30kHz–100kHz), always set R = 1 for specified performance. With an external high-frequency clock source, use R to divide the external frequency to between 30kHz and 100kHz.

The VCO's output clock, CGMVCLK, running at a frequency, f_{VCLK}, is fed back through a programmable pre-scaler divider and a programmable modulo divider. The pre-scaler divides the VCO clock by a power-of-two factor P and the modulo divider reduces the VCO clock by a factor, N. The dividers' output is the VCO feedback clock, CGMVDV, running at a frequency, $f_{VDV} = f_{VCLK}/(N \times 2^P)$. (See 5.3.6 Programming the PLL for more information.)

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external capacitor connected to CGMXFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in 5.3.4 Acquisition and Tracking Modes. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

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

5.3.4 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 start up 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 5.5.2 PLL Bandwidth Control Register.)
- Tracking mode — In tracking mode, the filter makes only small corrections to the frequency of the VCO. PLL jitter is much lower in tracking mode, but the response to noise is also slower. The PLL enters tracking mode when the VCO frequency is nearly correct, such as when the PLL is selected as the base clock source. (See 5.3.8 Base Clock Selector Circuit.) The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{\text{ACQ}}$ bit is set.

5.3.5 Manual and Automatic PLL Bandwidth Modes

The PLL can change the bandwidth or operational mode of the loop filter manually or automatically. Automatic mode is recommended for most users.

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. (See 5.5.2 PLL Bandwidth Control Register.) If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (during PLL start-up, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. (See 5.3.8 Base Clock Selector Circuit.) If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. (See 5.6 Interrupts for information and precautions on using interrupts.)

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

- The $\overline{\text{ACQ}}$ bit (See 5.5.2 PLL Bandwidth Control Register.) is a read-only indicator of the mode of the filter. (See 5.3.4 Acquisition and Tracking Modes.)
- The $\overline{\text{ACQ}}$ bit is set when the VCO frequency is within a certain tolerance and is cleared when the VCO frequency is out of a certain tolerance. (See 5.8 Acquisition/Lock Time Specifications for more information.)
- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance and is cleared when the VCO frequency is out of a certain tolerance. (See 5.8 Acquisition/Lock Time Specifications for more information.)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See 5.5.1 PLL Control Register.)

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

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 5.8 Acquisition/Lock Time Specifications.), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time, t_{AL}, after entering tracking mode before selecting the PLL as the clock source to CGMOUT (BCS = 1).
- The LOCK bit is disabled.
- CPU interrupts from the CGM are disabled.

5.3.6 Programming the PLL

The following procedure shows how to program the PLL.

NOTE

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

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

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

3. Choose a practical PLL (crystal) reference frequency, f_{RCLK}, and the reference clock divider, R. Typically, the reference crystal is 32.768 kHz and R = 1.

   Frequency errors to the PLL are corrected at a rate of f_{RCLK}/R. For stability and lock time reduction, this rate must be as fast as possible. The VCO frequency must be an integer multiple of this rate. The relationship between the VCO frequency, f_{VCLK}, and the reference frequency, f_{RCLK}, is

$$f_{VCLK} = \frac{2^P N}{R}(f_{RCLK})$$

   P, the power of two multiplier, and N, the range multiplier, are integers.

   In cases where desired bus frequency has some tolerance, choose f_{RCLK} to a value determined either by other module requirements (such as modules which are clocked by CGMXCLK), cost requirements, or ideally, as high as the specified range allows. See Section 23. Electrical Specifications. Choose the reference divider, R = 1. After choosing N and P, the actual bus frequency can be determined using equation in 2 above.

   When the tolerance on the bus frequency is tight, choose f_{RCLK} to an integer divisor of f_{BUSDES}, and R = 1. If f_{RCLK} cannot meet this requirement, use the following equation to solve for R with practical choices of f_{RCLK}, and choose the f_{RCLK} that gives the lowest R.

$$R = \text{round}\left[R_{MAX} \times \left\{\left(\frac{f_{VCLKDES}}{f_{RCLK}}\right) - \text{integer}\left(\frac{f_{VCLKDES}}{f_{RCLK}}\right)\right\}\right]$$

4. Select a VCO frequency multiplier, N.

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

Reduce N/R to the lowest possible R.

5. If N is < N_{max}, use P = 0. If N > N_{max}, choose P using this table:

Current N Value	P
$0 < N \le N_{max}$	0
$N_{max} < N \le N_{max} \times 2$	1
$N_{max} \times 2 < N \le N_{max} \times 4$	2
$N_{max} \times 2 < N \le N_{max} \times 4$	3

Then recalculate N:

$$N = \mathrm{round}\left(\frac{R \times f_{VCLKDES}}{f_{RCLK} \times 2^P}\right)$$

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

$$f_{VCLK} = (2^P \times N / R) \times f_{RCLK}$$

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

7. Select the VCO's power-of-two range multiplier E, according to this table:

Frequency Range	E
$0 < f_{VCLK} < 9{,}830{,}400$	0
$9{,}830{,}400 \le f_{VCLK} < 19{,}660{,}800$	1
$19{,}660{,}800 \le f_{VCLK} < 39{,}321{,}600$	2

NOTE: Do not program E to a value of 3.

8. Select a VCO linear range multiplier, L, where f_{NOM} = 38.4 kHz

$$L = \mathrm{round}\left(\frac{f_{VCLK}}{2^E \times f_{NOM}}\right)$$

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

$$f_{VRS} = (L \times 2^E)f_{NOM}$$

For proper operation,

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

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

NOTE

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

NOTE

11. Program the PLL registers accordingly:
    a. In the PRE bits of the PLL control register (PCTL), program the binary equivalent of P.
    b. In the VPR bits of the PLL control register (PCTL), program the binary equivalent of E.
    c. In the PLL multiplier select register low (PMSL) and the PLL multiplier select register high (PMSH), program the binary equivalent of N.
    d. In the PLL VCO range select register (PMRS), program the binary coded equivalent of L.
    e. In the PLL reference divider select register (PMDS), program the binary coded equivalent of R.

NOTE

The values for P, E, N, L, and R can only be programmed when the PLL is off (PLLON = 0).

Table 5-1 provides numeric examples (numbers are in hexadecimal notation):

Table 5-1. Numeric Example

f_{BUS}	f_{RCLK}	R	N	P	E	L
2.0 MHz	32.768 kHz	1	F5	0	0	D1
2.4576 MHz	32.768 kHz	1	12C	0	1	80
2.5 MHz	32.768 kHz	1	132	0	1	83
4.0 MHz	32.768 kHz	1	1E9	0	1	D1
4.9152 MHz	32.768 kHz	1	258	0	2	80
5.0 MHz	32.768 kHz	1	263	0	2	82
7.3728 MHz	32.768 kHz	1	384	0	2	C0
8.0 MHz	32.768 kHz	1	3D1	0	2	D0

5.3.7 Special Programming Exceptions

The programming method described in 5.3.6 Programming the PLL does not account for three possible exceptions. A value of 0 for R, N, or L is meaningless when used in the equations given. To account for these exceptions:

- A 0 value for R or N is interpreted exactly the same as a value of 1.
- A 0 value for L disables the PLL and prevents its selection as the source for the base clock.

(See 5.3.8 Base Clock Selector Circuit.)

5.3.8 Base Clock Selector Circuit

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

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

5.3.9 CGM External Connections

In its typical configuration, the CGMC requires up to nine external components. Five of these are for the crystal oscillator and two or four are for the PLL.

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in Figure 5-3. Figure 5-3 shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

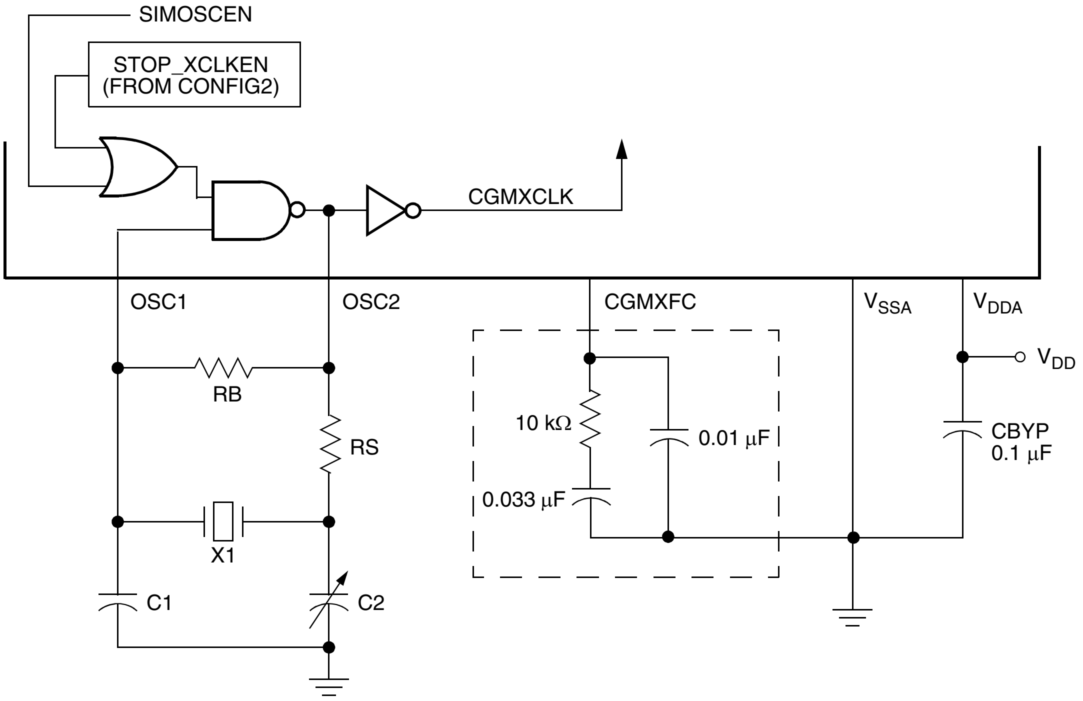
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S

The series resistor (R_S) is included in the diagram to follow strict Pierce oscillator guidelines. Refer to the crystal manufacturer’s data for more information regarding values for C1 and C2.

Figure 5-3 also shows the external components for the PLL:

- Bypass capacitor, C_{BYP}
- Filter network

Care should be taken with PCB routing in order to minimize signal cross talk and noise. (See 17.11.2 CGM Electrical Specifications for capacitor and resistor values.)



Note: Filter network in box can be replaced with a 0.47 µF capacitor, but will degrade stability.

Figure 5-3. CGMC External Connections

5.4 I/O Signals

The following paragraphs describe the CGM I/O signals.

5.4.1 Crystal Amplifier Input Pin (OSC1)

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

5.4.2 Crystal Amplifier Output Pin (OSC2)

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

5.4.3 External Filter Capacitor Pin (CGMXFC)

The CGMXFC pin is required by the loop filter to filter out phase corrections. An external filter network is connected to this pin. (See Figure 5-3.)

NOTE

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

5.4.4 PLL Analog Power Pin (V_{DDA})

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

NOTE

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

5.4.5 PLL Analog Ground Pin (V_{SSA})

V_{SSA} is a ground pin used by the analog portions of the PLL. Connect the V_{SSA} pin to the same voltage potential as the V_{SS} pin.

NOTE

Route V_{SSA} carefully for maximum noise immunity and place bypass capacitors as close as possible to the package.
On this MCU, the V_{SSA} is physically bonded to the V_{SS} pin.

5.4.6 Oscillator Output Frequency Signal (CGMXCLK)

CGMXCLK is the oscillator output signal. It runs at the full speed of the oscillator, and is generated directly from the crystal oscillator circuit, the RC oscillator circuit, or the internal oscillator circuit.

5.4.7 CGM Reference Clock (CGMRCLK)

CGMRCLK is a buffered version of CGMXCLK, this clock is the reference clock for the phase-locked-loop circuit.

5.4.8 CGM VCO Clock Output (CGMVCLK)

CGMVCLK is the clock output from the VCO.

5.4.9 CGM Base Clock Output (CGMOUT)

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

5.4.10 CGM CPU Interrupt (CGMINT)

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

5.5 CGM Registers

The following registers control and monitor operation of the CGM:

- PLL control register (PCTL)
  (See 5.5.1 PLL Control Register.)
- PLL bandwidth control register (PBWC)
  (See 5.5.2 PLL Bandwidth Control Register.)
- PLL multiplier select registers (PMSH and PMSL)
  (See 5.5.3 PLL Multiplier Select Registers.)
- PLL VCO range select register (PMRS)
  (See 5.5.4 PLL VCO Range Select Register.)
- PLL reference divider select register (PMDS)
  (See 5.5.5 PLL Reference Divider Select Register.)

5.5.1 PLL Control Register

The PLL control register (PCTL) contains the interrupt enable and flag bits, the on/off switch, the base clock selector bit, the prescaler bits, and the VCO power-of-two range selector bits.

Address: $0036

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
Write:		▨						
Reset:	0	0	1	0	0	0	0	0

▨ = Unimplemented

Figure 5-4. PLL Control Register (PCTL)

PLLIE — PLL Interrupt Enable Bit

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

1 = PLL interrupts enabled
0 = PLL interrupts disabled

PLLF — PLL Interrupt Flag Bit

This read-only bit is set whenever the LOCK bit toggles. PLLF generates an interrupt request if the PLLIE bit also is set. PLLF always reads as logic 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 5.3.8 Base Clock Selector Circuit.) Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

BCS — Base Clock Select Bit

This read/write bit selects either the oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS, it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See 5.3.8 Base Clock Selector Circuit.) Reset clears the BCS bit.

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

NOTE

PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. (See 5.3.8 Base Clock Selector Circuit.)

PRE1 and PRE0 — Prescaler Program Bits

These read/write bits control a prescaler that selects the prescaler power-of-two multiplier, P. (See 5.3.3 PLL Circuits and 5.3.6 Programming the PLL.) PRE1 and PRE0 cannot be written when the PLLON bit is set. Reset clears these bits.

These prescaler bits affects the relationship between the VCO clock and the final system bus clock.

Table 5-2. PRE1 and PRE0 Programming

PRE1 and PRE0	P	Prescaler Multiplier
00	0	1
01	1	2
10	2	4
11	3	8

VPR1 and VPR0 — VCO Power-of-Two Range Select Bits

These read/write bits control the VCO's hardware power-of-two range multiplier E that, in conjunction with L (See 5.3.3 PLL Circuits, 5.3.6 Programming the PLL, and 5.5.4 PLL VCO Range Select Register.) controls the hardware center-of-range frequency, f_{VRS}. VPR1:VPR0 cannot be written when the PLLON bit is set. Reset clears these bits.

Table 5-3. VPR1 and VPR0 Programming

VPR1 and VPR0	E	VCO Power-of-Two Range Multiplier
00	0	1
01	1	2
10	2	4

NOTE: Do not program E to a value of 3.

5.5.2 PLL Bandwidth Control Register

The PLL bandwidth control register (PBWC):

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

Address: $0037

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

▨ = Unimplemented R = Reserved

Figure 5-5. PLL Bandwidth Control Register (PBWCR)

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. The write one function of this bit is reserved for test, so this bit must *always* be written a 0. 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

5.5.3 PLL Multiplier Select Registers

The PLL multiplier select registers (PMSH and PMSL) contain the programming information for the modulo feedback divider.

Address: $0038

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
Write:	▨	▨	▨	▨				
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 5-6. PLL Multiplier Select Register High (PMSH)

Address: $0039

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
Reset:	0	1	0	0	0	0	0	0

Figure 5-7. PLL Multiplier Select Register Low (PMSL)

MUL[11:0] — Multiplier Select Bits

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier N. (See 5.3.3 PLL Circuits and 5.3.6 Programming the PLL.) A value of $0000 in the multiplier select registers configure the modulo feedback divider the same as a value of $0001. Reset initializes the registers to $0040 for a default multiply value of 64.

NOTE

The multiplier select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1).

5.5.4 PLL VCO Range Select Register

The PLL VCO range select register (PMRS) contains the programming information required for the hardware configuration of the VCO.

Address: $003A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
Reset:	0	1	0	0	0	0	0	0

Figure 5-8. PLL VCO Range Select Register (PMRS)

VRS[7:0] — VCO Range Select Bits

These read/write bits control the hardware center-of-range linear multiplier L which, in conjunction with E (See 5.3.3 PLL Circuits, 5.3.6 Programming the PLL, and 5.5.1 PLL Control Register.), controls the hardware center-of-range frequency, f_{VRS}. VRS[7:0] cannot be written when the PLLON bit in the PCTL is set. (See 5.3.7 Special Programming Exceptions.) A value of $00 in the VCO range select register disables the PLL and clears the BCS bit in the PLL control register (PCTL). (See 5.3.8 Base Clock Selector Circuit and 5.3.7 Special Programming Exceptions.). Reset initializes the register to $40 for a default range multiply value of 64.

NOTE

The VCO range select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1) and such that the VCO clock cannot be selected as the source of the base clock (BCS = 1) if the VCO range select bits are all clear.

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

5.5.5 PLL Reference Divider Select Register

The PLL reference divider select register (PMDS) contains the programming information for the modulo reference divider.

Address: $003B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
Write:	▨	▨	▨	▨				
Reset:	0	0	0	0	0	0	0	1

▨ = Unimplemented

Figure 5-9. PLL Reference Divider Select Register (PMDS)

RDS[3:0] — Reference Divider Select Bits

These read/write bits control the modulo reference divider that selects the reference division factor, R. (See 5.3.3 PLL Circuits and 5.3.6 Programming the PLL.) RDS[3:0] cannot be written when the PLLON bit in the PCTL is set. A value of $00 in the reference divider select register configures the reference divider the same as a value of $01. (See 5.3.7 Special Programming Exceptions.) Reset initializes the register to $01 for a default divide value of 1.

NOTE

The reference divider select bits have built-in protection such that they cannot be written when the PLL is on (PLLON = 1).

NOTE

The default divide value of 1 is recommended for all applications.

5.6 Interrupts

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

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

NOTE

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

5.7 Special Modes

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

5.7.1 Wait Mode

The WAIT instruction does not affect the CGM. Before entering wait mode, software can disengage and turn off the PLL by clearing the BCS and PLLON bits in the PLL control register (PCTL) to save power. Less power-sensitive applications can disengage the PLL without turning it off, so that the PLL clock is immediately available at WAIT exit. This would be the case also when the PLL is to wake the MCU from wait mode, such as when the PLL is first enabled and waiting for LOCK or LOCK is lost.

5.7.2 Stop Mode

If the oscillator stop mode enable bit (STOP_XCLKEN in CONFIG2 register) is configured to disabled the oscillator in stop mode, then the STOP instruction disables the CGM (oscillator and phase locked loop) 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.

If the oscillator stop mode enable bit is configured for continuous oscillator operation in stop mode, then the phase locked loop is shut off but the CGMXCLK will continue to drive the SIM and other MCU sub-systems.

5.7.3 CGM During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See 4.7.3 SIM Break Flag Control Register.)

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

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

5.8 Acquisition/Lock Time Specifications

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

5.8.1 Acquisition/Lock Time Definitions

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

5.8.2 Parametric Influences on Reaction Time

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

The most critical parameter which affects the reaction times of the PLL is the reference frequency, f_{RDV}. This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these corrections. This parameter is under user control via the choice of crystal frequency f_{XCLK} and the R value programmed in the reference divider. (See 5.3.3 PLL Circuits, 5.3.6 Programming the PLL, and 5.5.5 PLL Reference Divider Select Register.)

Another critical parameter is the external filter network. The PLL modifies the voltage on the VCO by adding or subtracting charge from capacitors in this network. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitance. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. (See 5.8.3 Choosing a Filter.)

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

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

5.8.3 Choosing a Filter

As described in 5.8.2 Parametric Influences on Reaction Time, the external filter network is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage.

Either of the filter networks in Figure 5-10 is recommended when using a 32.768 kHz reference clock (CGMRCLK). Figure 5-10 **(a)** is used for applications requiring better stability. Figure 5-10 **(b)** is used in low-cost applications where stability is not critical.

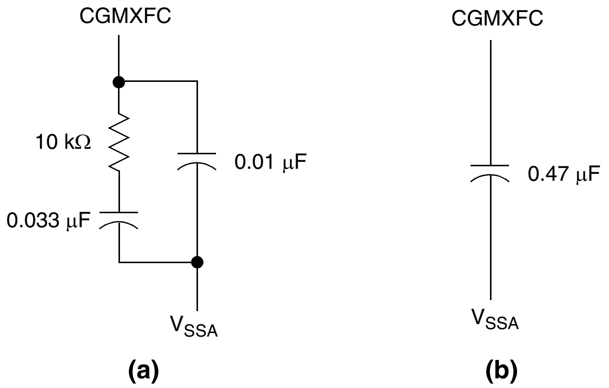


Figure 5-10. PLL Filter

Chapter 6
Timer Interface Module (TIM)

6.1 Introduction

This section describes the timer interface (TIM) module. The TIM is a two-channel timer that provides a timing reference with Input capture, output compare, and pulse-width-modulation functions. Figure 6-1 is a block diagram of the TIM.

This particular MCU has two timer interface modules which are denoted as TIM1 and TIM2.

6.2 Features

Features of the TIM include:

- Two input capture/output compare channels:
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
- Buffered and unbuffered pulse-width-modulation (PWM) signal generation
- Programmable TIM clock input with 7-frequency internal bus clock prescaler selection
- Free-running or modulo up-count operation
- Toggle any channel pin on overflow
- TIM counter stop and reset bits

6.3 Pin Name Conventions

The text that follows describes both timers, TIM1 and TIM2. The TIM input/output (I/O) pin names are T[1,2]CH0 (timer channel 0) and T[1,2]CH1 (timer channel 1), where “1” is used to indicate TIM1 and “2” is used to indicate TIM2. The two TIMs share four I/O pins with four I/O port pins. The external clock input for TIM2 is shared with the an ADC channel pin. The full names of the TIM I/O pins are listed in Table 6-1. The generic pin names appear in the text that follows.

Table 6-1. Pin Name Conventions

TIM Generic Pin Names:		**T[1,2]CH0**	**T[1,2]CH1**
Full TIM Pin Names:	TIM1	PTB2/T1CH0/PPIECK	PTB3/T1CH1
	TIM2	PTB4/T2CH0	PTB5/T2CH1

NOTE

References to either timer 1 or timer 2 may be made in the following text by omitting the timer number. For example, TCH0 may refer generically to T1CH0 and T2CH0, and TCH1 may refer to T1CH1 and T2CH1.

6.4 Functional Description

Figure 6-1 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 two TIM channels (per timer) are programmable independently as input capture or output compare channels.

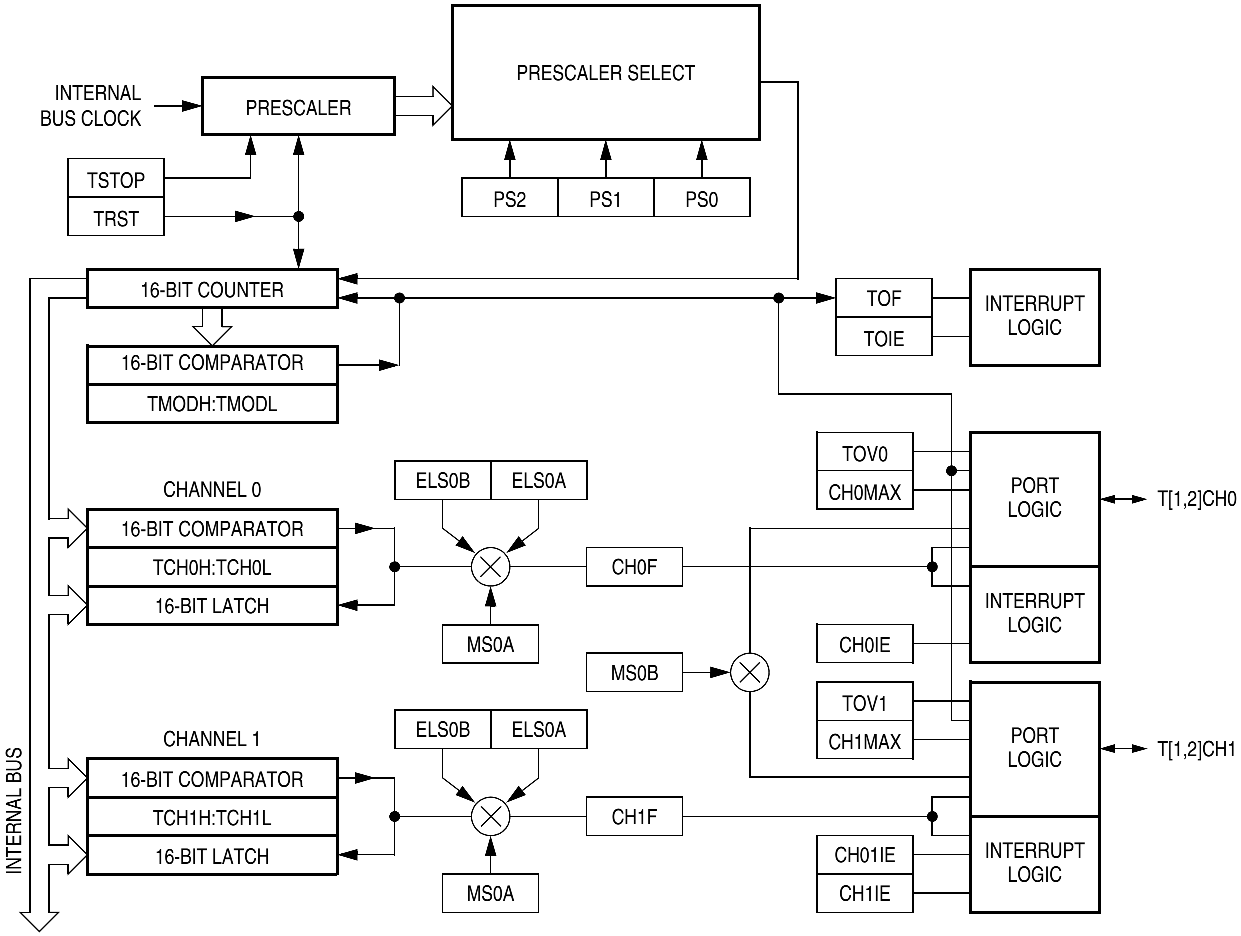


Figure 6-1. TIM Block Diagram

Figure 6-2 summarizes the timer registers.

NOTE

References to either timer 1 or timer 2 may be made in the following text by omitting the timer number. For example, TSC may generically refer to both T1SC and T2SC.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
	TIM1 Status and Control	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
$0020	Register	Write:	0			TRST	▨			
	(T1SC)	Reset:	0	0	1	0	0	0	0	0
	TIM1 Counter Register	Read:	Bit 15	14	13	12	11	10	9	Bit 8
$0021	High	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	(T1CNTH)	Reset:	0	0	0	0	0	0	0	0
	TIM1 Counter Register	Read:	Bit 7	6	5	4	3	2	1	Bit 0
$0022	Low	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	(T1CNTL)	Reset:	0	0	0	0	0	0	0	0
	TIM Counter Modulo	Read:	Bit 15	14	13	12	11	10	9	Bit 8
$0023	Register High	Write:								
	(TMODH)	Reset:	1	1	1	1	1	1	1	1
	TIM1 Counter Modulo	Read:	Bit 7	6	5	4	3	2	1	Bit 0
$0024	Register Low	Write:								
	(T1MODL)	Reset:	1	1	1	1	1	1	1	1
	TIM1 Channel 0 Status	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
$0025	and Control Register	Write:	0							
	(T1SC0)	Reset:	0	0	0	0	0	0	0	0
	TIM1 Channel 0	Read:	Bit 15	14	13	12	11	10	9	Bit 8
$0026	Register High	Write:								
	(T1CH0H)	Reset:	Indeterminate after reset							
	TIM1 Channel 0	Read:	Bit 7	6	5	4	3	2	1	Bit 0
$0027	Register Low	Write:								
	(T1CH0L)	Reset:	Indeterminate after reset							
	TIM1 Channel 1 Status	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
$0028	and Control Register	Write:	0		▨					
	(T1SC1)	Reset:	0	0	0	0	0	0	0	0
	TIM1 Channel 1	Read:	Bit 15	14	13	12	11	10	9	Bit 8
$0029	Register High	Write:								
	(T1CH1H)	Reset:	Indeterminate after reset							
	TIM1 Channel 1	Read:	Bit 7	6	5	4	3	2	1	Bit 0
$002A	Register Low	Write:								
	(T1CH1L)	Reset:	Indeterminate after reset							
	TIM2 Status and Control	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
$002B	Register	Write:	0			TRST	▨			
	(T2SC)	Reset:	0	0	1	0	0	0	0	0
	TIM2 Counter Register	Read:	Bit 15	14	13	12	11	10	9	Bit 8
$002C	High	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	(T2CNTH)	Reset:	0	0	0	0	0	0	0	0
	TIM2 Counter Register	Read:	Bit 7	6	5	4	3	2	1	Bit 0
$002D	Low	Write:	▨	▨	▨	▨	▨	▨	▨	▨
	(T2CNTL)	Reset:	0	0	0	0	0	0	0	0
	TIM2 Counter Modulo	Read:	Bit 15	14	13	12	11	10	9	Bit 8
$002E	Register High	Write:								
	(T2MODH)	Reset:	1	1	1	1	1	1	1	1

▨ = Unimplemented

Figure 6-2. TIM I/O Register Summary (Sheet 1 of 2)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$002F	TIM2 Counter Modulo Register Low (T2MODL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0030	TIM2 Channel 0 Status and Control Register (T2SC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0031	TIM2 Channel 0 Register High (T2CH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0032	TIM2 Channel 0 Register Low (T2CH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0033	TIM2 Channel 1 Status and Control Register (T2SC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0		▨					
		Reset:	0	0	0	0	0	0	0	0
$0034	TIM2 Channel 1 Register High (T2CH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0035	TIM2 Channel 1 Register Low (T2CH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

▨ = Unimplemented

Figure 6-2. TIM I/O Register Summary (Sheet 2 of 2)

6.4.1 TIM Counter Prescaler

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

6.4.2 Input Capture

With the input capture function, the TIM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input capture channel, the TIM latches the contents of the TIM counter into the TIM channel registers, TCHxH:TCHxL. The polarity of the active edge is programmable. Input captures can generate TIM CPU interrupt requests.

6.4.3 Output Compare

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

6.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 6.4.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

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

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

- When changing to a smaller value, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current output compare pulse. The interrupt routine has until the end of the counter overflow period to write the new value.
- When changing to a larger output compare value, enable 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.

6.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the 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.

NOTE

In buffered output compare operation, do not write new output compare values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered output compares.

6.4.4 Pulse Width Modulation (PWM)

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

As Figure 6-3 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.

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 6.9.1 TIM Status and Control Register.

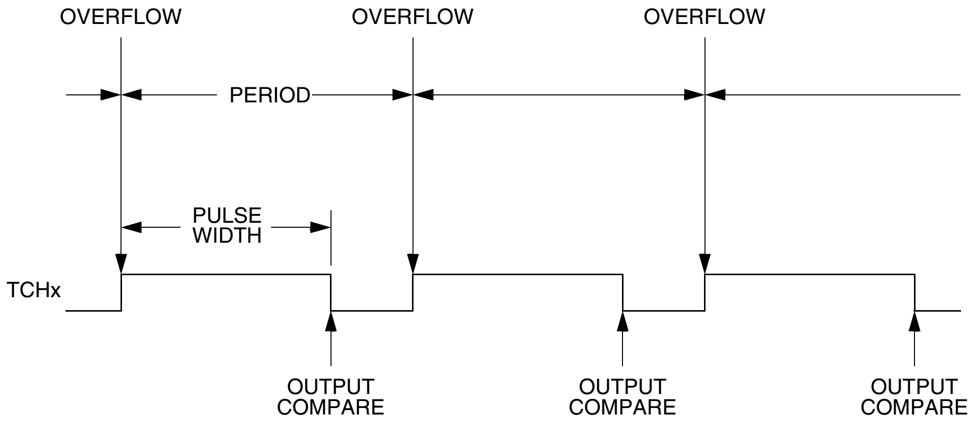


Figure 6-3. PWM Period and Pulse Width

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%.

6.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 6.4.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

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

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

- When changing to a shorter pulse width, enable channel x output compare interrupts and write the new value in the output compare interrupt routine. The output compare interrupt occurs at the end of the current pulse. The interrupt routine has until the end of the PWM period to write the new value.
- When changing to a longer pulse width, enable 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.

6.4.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the 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.

NOTE

In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. User software should track the currently active channel to prevent writing a new value to the active channel. Writing to the active channel registers is the same as generating unbuffered PWM signals.

6.4.4.3 PWM Initialization

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

1. In the TIM status and control register (TSC):
   a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
   b. Reset the TIM counter and prescaler by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. (See Table 6-3.)
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB:ELSxA. The output action on compare must force the output to the complement of the pulse width level. (See Table 6-3.)

NOTE

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

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

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

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

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output. (See 6.9.4 TIM Channel Status and Control Registers.)

6.5 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter reaches the modulo value programmed in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH1F:CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE. Channel x TIM CPU interrupt requests are enabled when CHxIE = 1. CHxF and CHxIE are in the TIM channel x status and control register.

6.6 Low-Power Modes

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

6.6.1 Wait Mode

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

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

6.6.2 Stop Mode

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

6.7 TIM During Break Interrupts

A break interrupt stops the TIM counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See 21.5.4 SIM Break Flag Control Register.)

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

To protect status bits during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), software can read and write I/O registers during the break state without affecting status bits. Some status bits have a 2-step read/write clearing procedure. If software does the first step on such a bit before the break, the bit cannot change during the break state as long as BCFE is at logic 0. After the break, doing the second step clears the status bit.

6.8 I/O Signals

Port B shares four of its pins with the TIM. The four TIM channel I/O pins are T1CH0, T1CH1, T2CH0, and T2CH1 as described in 6.3 Pin Name Conventions.

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. T1CH0 and T2CH0 can be configured as buffered output compare or buffered PWM pins.

6.9 I/O Registers

NOTE

References to either timer 1 or timer 2 may be made in the following text by omitting the timer number. For example, TSC may generically refer to both T1SC AND T2SC.

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

- TIM status and control register (TSC)
- TIM counter registers (TCNTH:TCNTL)
- TIM counter modulo registers (TMODH:TMODL)
- TIM channel status and control registers (TSC0, TSC1)
- TIM channel registers (TCH0H:TCH0L, TCH1H:TCH1L)

6.9.1 TIM Status and Control Register

The TIM status and control register (TSC):

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

Address: T1SC, $0020 and T2SC, $002B

	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 6-4. TIM Status and Control Register (TSC)

TOF — TIM Overflow Flag Bit

This read/write flag is set when the TIM counter reaches the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TIM status and control register when TOF is set and then writing a 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 = TIM counter has reached modulo value
0 = TIM counter has not reached modulo value

TOIE — TIM Overflow Interrupt Enable Bit

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

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

TSTOP — TIM Stop Bit

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

1 = TIM counter stopped
0 = TIM counter active

NOTE

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

TRST — TIM Reset Bit

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

1 = Prescaler and TIM counter cleared
0 = No effect

NOTE

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

PS[2:0] — Prescaler Select Bits

These read/write bits select one of the seven prescaler outputs as the input to the TIM counter as Table 6-2 shows. Reset clears the PS[2:0] bits.

Table 6-2. Prescaler Selection

PS2	PS1	PS0	TIM Clock Source
0	0	0	Internal bus clock ÷ 1
0	0	1	Internal bus clock ÷ 2
0	1	0	Internal bus clock ÷ 4
0	1	1	Internal bus clock ÷ 8
1	0	0	Internal bus clock ÷ 16
1	0	1	Internal bus clock ÷ 32
1	1	0	Internal bus clock ÷ 64
1	1	1	Not available

6.9.2 TIM Counter Registers

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

NOTE

If 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: T1CNTH, $0021 and T2CNTH, $002C

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

Figure 6-5. TIM Counter Registers High (TCNTH)

Address: T1CNTL, $0022 and T2CNTL, $002D

	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

Figure 6-6. TIM Counter Registers Low (TCNTL)

6.9.3 TIM Counter Modulo Registers

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

Address: T1MODH, $0023 and T2MODH, $002E

	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

Figure 6-7. TIM Counter Modulo Register High (TMODH)

Address: T1MODL, $0024 and T2MODL, $002F

	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 6-8. TIM Counter Modulo Register Low (TMODL)

NOTE

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

6.9.4 TIM Channel Status and Control Registers

Each of the TIM channel status and control registers:

- Flags input captures and output compares
- Enables input capture and output compare interrupts
- Selects input capture, output compare, or PWM operation
- Selects high, low, or toggling output on output compare
- Selects rising edge, falling edge, or any edge as the active input capture trigger
- Selects output toggling on TIM overflow
- Selects 0% and 100% PWM duty cycle
- Selects buffered or unbuffered output compare/PWM operation

Address: T1SC0, $0025 and T2SC0, $0030

	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 6-9. TIM Channel 0 Status and Control Register (TSC0)

Address: T1SC1, $0028 and T2SC1, $0033

	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

Figure 6-10. TIM Channel 1 Status and Control Register (TSC1)

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 = 1), 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.

Reset clears the CHxF bit. Writing a logic 1 to CHxF has no effect.

1 = Input capture or output compare on channel x
0 = No input capture or output compare on channel x

CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM CPU interrupt service requests on channel x.

Reset clears the CHxIE bit.

1 = Channel x CPU interrupt requests enabled
0 = Channel x CPU interrupt requests disabled

MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIM1 channel 0 and TIM2 channel 0 status and control registers.

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

Reset clears the MSxB bit.

1 = Buffered output compare/PWM operation enabled
0 = Buffered output compare/PWM operation disabled

MSxA — Mode Select Bit A

When ELSxB:ELSxA ≠ 0:0, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation.

See Table 6-3.

1 = Unbuffered output compare/PWM operation
0 = Input capture operation

When ELSxB:ELSxA = 0:0, this read/write bit selects the initial output level of the TCHx pin.

See Table 6-3. Reset clears the MSxA bit.

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

NOTE

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

ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to an I/O port, and pin TCHx is available as a general-purpose I/O pin. Table 6-3 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 6-3. Mode, Edge, and Level Selection

MSxB:MSxA	ELSxB:ELSxA	Mode	Configuration
X0	00	Output preset	Pin under port control; initial output level high
X1	00		Pin under port control; initial output level low
00	01	Input capture	Capture on rising edge only
00	10		Capture on falling edge only
00	11		Capture on rising or falling edge
01	01	Output compare or PWM	Toggle output on compare
01	10		Clear output on compare
01	11		Set output on compare
1X	01	Buffered output compare or buffered PWM	Toggle output on compare
1X	10		Clear output on compare
1X	11		Set output on compare

NOTE

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

TOVx — Toggle On Overflow Bit

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

Reset clears the TOVx bit.

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

NOTE

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

CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 6-11 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.

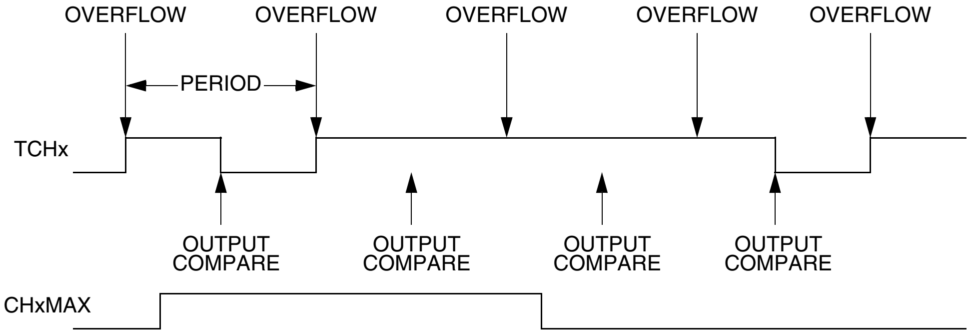


Figure 6-11. CHxMAX Latency

6.9.5 TIM Channel Registers

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

In input capture mode (MSxB:MSxA = 0:0), reading the high byte of the TIM channel x registers (TCHxH) inhibits input captures until the low byte (TCHxL) is read.

In output compare mode (MSxB:MSxA ≠ 0:0), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

Address: T1CH0H, $0026 and T2CH0H, $0031

	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							

Figure 6-12. TIM Channel 0 Register High (TCH0H)

Address: T1CH0L, $0027 and T2CH0L $0032

	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 6-13. TIM Channel 0 Register Low (TCH0L)

Address: T1CH1H, $0029 and T2CH1H, $0034

	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							

Figure 6-14. TIM Channel 1 Register High (TCH1H)

Address: T1CH1L, $002A and T2CH1L, $0035

	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 6-15. TIM Channel 1 Register Low (TCH1L)

Chapter 7
Programmable Periodic Interrupt (PPI)

7.1 Introduction

This section describes the programmable periodic interrupt (PPI) module. The PPI will generate periodic interrupts at user selectable rates using a counter clocked by the selected clock.

7.2 Features

Features of the PPI include:
- Seven user selectable periodic interrupts
- User selectable clock source:
  - Internal 32kHz
  - CGMXCLK output from CGM module
  - External clock from PPIECK pin

7.3 Functional Description

The PPI module generates periodic interrupt requests to the CPU. The interrupt request is treated as a regular keyboard interrupt request, with the difference that instead of a pin, the interrupt signal is generated by internal logic.

When PPI counter reaches the defined count, it generates an interrupt request. The latched status of interrupt generation of the PPI can be read directly from the PPI1L bit. This is a read-only status bit which is occupies a bit position in the register. The latch can be cleared by writing to the ACKK bit in the KBSCR register.

The PPI counter can count and generate interrupts even when the MCU is in stop mode if the corresponding clock source is enabled.

Figure 7-1 is a block diagram of the PPI.

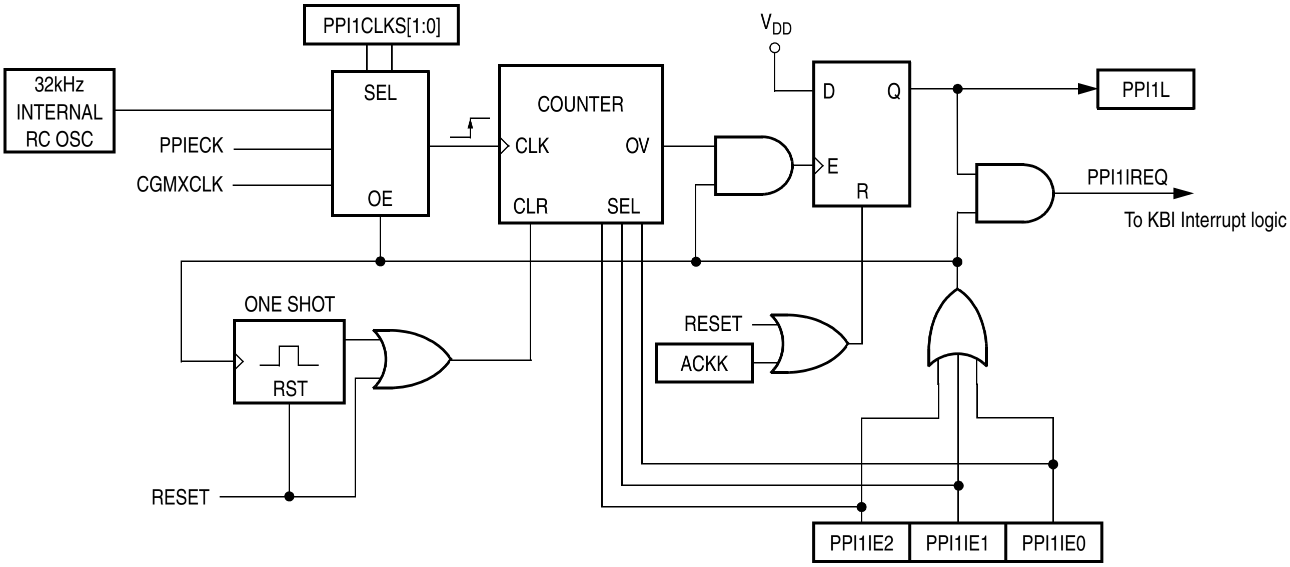


Figure 7-1. Programmable Periodic Interrupt Block Diagram

7.4 I/O Pins

The external clock input option of the PPI is from the PPIECK pin and is selected by the clock select bits, PPI1CLKS[1:0]. The maximum PPIECK frequency is four times the bus frequency.

7.5 Low-Power Modes

The PPI module remains active (crystal clock source is not affected if crystal clock is enabled in stop mode; counter can count and can generate interrupts) in wait and stop mode if proper clocking source is supplied.

7.6 PPI I/O Registers

The PPI module does not have dedicated registers, instead its control bits are located in other registers.

7.6.1 PPI Clock Source Select and Interrupt Latch

The control bits for selecting the PPI input clock source and the interrupt latch status bit is located in the port B high current drive control register (HDB).

Address: $000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	PPI1L	HDB5	HDB4	HDB3	HDB2	PPI1CLKS1	PPI1CLKS0
Write:		▨						
Reset:	0	0	0	0	0	0	0	0

Figure 7-2. Port B High Current Drive Control Register (HDB)

PPI1L — PPI1 Pending for Acknowledgement

This read-only status bit indicates a interrupt request is generated by PPI1 and is pending for acknowledgement. This bit does not generate an interrupt to the CPU, instead, the interrupt is generated by the KBI module. In the KBI interrupt service routine, the PPI1L bit should be read to determine if the interrupt was generated by the PPI. The PPI1L bit is cleared by writing logic 1 to the ACKK bit in the Keyboard Status and Control Register.

1 = PPI Interrupt request is pending
0 = No PPI interrupt request is pending

HDB[5:2] — Port B High Current Drive Enable Bits

(See 10.3.3 Port B High Current Drive Control Register (HDB).)

PPI1CLKS[1:0] — PPI1 Clock Source Select

These two bits select the clock source for the PPI.

Table 7-1. PPI1 Clock Source Selection

PPI1CLKS[1:0]	Clock Source for PPI1
00	32 kHz internal RC clock
01	External clock from PPIECK pin
10	CGMXCLK from CGM module
11	Reserved

7.6.2 PPI Interrupt Period Select

The interrupt period from the PPI is selected using bits, PPI1IE[2:0], in the keyboard interrupt enable register (KBIER).

Address: $001C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	PPI1IE2	PPI1IE1	PPI1IE0	KBIE3	KBIE2	KBIE1	KBIE0
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 7-3. Keyboard Interrupt Enable Register (KBIER)

PPI1IE[2:0] —PPI1 Interrupt Period Select Bits

These three bits select the PPI interrupt period. The PPI is disabled when PPI1IE[2:0] are zero and no interrupts are generated.

Table 7-2. PPI1 interrupt period selection

PPI1IE[2:0]	Interrupt Period
000	PPI and its associated interrupts are disabled
001	512 PPI counts
010	1,024 PPI counts
011	2,048 PPI counts
100	4,096 PPI counts
101	8,192 PPI counts
110	16,384 PPI counts
111	32,768 PPI counts

KBIE[3:0] — Keyboard Interrupt Enable Bits
(See Chapter 12 Keyboard Interrupt Module (KBI).)

7.6.3 PPI Interrupt Acknowledge

The PPI interrupt latch, PPI1L, is cleared using the ACCK bit in the keyboard status and control register (KBSCR).

Address: $001B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
Write:	▨	▨	▨	▨	▨	ACKK		
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 7-4. Keyboard Status and Control Register (KBSCR)

KEYF — Keyboard Flag Bit
(See Chapter 12 Keyboard Interrupt Module (KBI).)

ACKK — Keyboard Acknowledge Bit
Writing a logic 1 to this write-only bit clears the PPI interrupt latch, PPI1L. Writing a logic 1 also clears the keyboard interrupt request. ACKK always reads as logic 0. Reset clears ACKK.

IMASKK— Keyboard Interrupt Mask Bit
(See Chapter 12 Keyboard Interrupt Module (KBI).)

MODEK — Keyboard Triggering Sensitivity Bit
This bit should be set to logic 1 (edge and level trigger) when the PPI is enabled together with any of the KBI enabled. Logic 0 can be selected if the PPI is enabled with no KBI's enabled.

This read/write bit controls the triggering sensitivity of the keyboard interrupt pins on port A. Reset clears MODEK.

1 = Keyboard interrupt requests on falling edges and low levels
0 = Keyboard interrupt requests on falling edges only

KBIE3–KBIE0 — Keyboard Interrupt Enable Bits

(See 12.5.2 Keyboard Interrupt Enable Register.)

7.7 Using the PPI

As the PPI and KBI interrupts can appear asynchronously and share the same internal circuit to generate interrupts to the CPU, an edge-only KBI trigger may not able to detect all possible asynchronous interrupts from the PPI and KBI. Therefore, when enabling PPI interrupt together with any KBI interrupts, the KBI interrupt trigger sensitivity should be set to edge and level sensitive (MODEK = 1). When edge and level sensitive is selected, any KBI input should disable itself while the associated KBI pin is held at low and re-enable when associated KBI pin gets high.

The code below shows an example of a system having PPI and KBI3 enabled, with each running from a different asynchronous clock source.

```
KBIER          EQU
PPICLKS0_R     EQU
ACKK           EQU
IMASK          EQU
IMASK_R        EQU
PTA            EQU

RESETINIT:
.
.
.
* SETUP PPI1 AND KBI
               MOV     #$03,KBSCR             ; MODEK = 1 !
               MOV     #$78,KBIER
               LDA     HDB
               AND     #~(3)
               STA     HDB
               BSET    ACKK
               BCLR    IMASK,IMASK_R          ; ENABLE KBI INT
.
.
.
               CLI                            ; ENABLE ALL INT
LOOP:
               STOP                           ; PUT MCU IN STOP
               BRCLR   3,PTA,*                ; WAIT KBI3
               BSET    KBIE3,KBIE3_R          ; ENABLE WHEN HIGH
               BRA     LOOP                   ; LOOP AGAIN
.
.
.
KBI_ISR:
               BRSET   3,PTA,KBI_ISR1
               BCLR    KBIE3,KBIE3_R          ; DISABLE KBI3
                                              ; DO KBI3 SERVICES HERE

KBI_ISR1:
               BRCLR   PPI1L,PPI1L_R,KBI_ISR_X
KBI_ISR21:
                                              ; DO PPI1 SERVICES HERE
               BSET    ACKK,ACKK_R            ; CLEAR ALL FLAGS
KBI_ISR_X:
               RTI
```

Chapter 8
Analog-to-Digital Converter (ADC)

8.1 Introduction

This section describes the 10-bit successive approximation analog-to-digital converter (ADC10).

The ADC10 on this MCU uses V_{DD} and V_{SS} as its supply and reference pins. This MCU uses CGMXCLK as its alternate clock source for the ADC. This MCU does not have a hardware conversion trigger.

8.2 Features

Features of the ADC10 module include:

- Linear successive approximation algorithm with 10-bit resolution
- Output formatted in 10- or 8-bit right-justified format
- Single or continuous conversion (automatic power-down in single conversion mode)
- Configurable sample time and conversion speed (to save power)
- Conversion complete flag and interrupt
- Input clock selectable from up to three sources
- Operation in wait and stop modes for lower noise operation
- Selectable asynchronous hardware conversion trigger

Figure 8-1 provides a summary of the input/output (I/O) registers.

Addr.	**Register Name**		Bit 7	6	5	4	3	2	1	Bit 0
$003C	ADC Status and Control Register (ADCSC)	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$003D	ADC10 Data Register High 8/10-Bit Mode (ADRH)	Read:	0	0	0	0	0	0	0/AD9	0/AD8
		Write:	Reserved							
		Reset:	0	0	0	0	0	0	0	0
$003E	ADC10 Data Register Low (ADRL)	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	Reserved							
		Reset:	0	0	0	0	0	0	0	0
$003F	ADC10 Clock Register (ADCLK)	Read:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ACLKEN
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Figure 8-1. ADC I/O Register Summary

8.3 Functional Description

The ADC10 uses successive approximation to convert the input sample taken from ADVIN to a digital representation. The approximation is taken and then rounded to the nearest 10- or 8-bit value to provide greater accuracy and to provide a more robust mechanism for achieving the ideal code-transition voltage.

Figure 8-2 shows a block diagram of the ADC10.

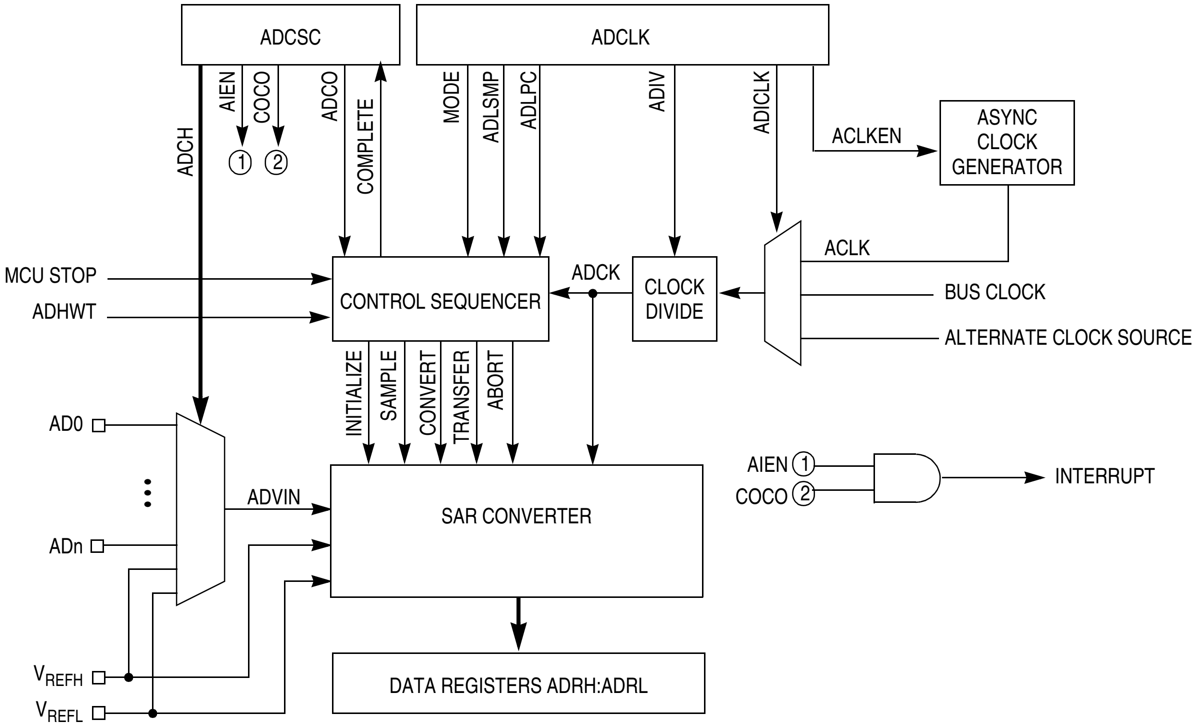


Figure 8-2. ADC10 Block Diagram

For proper conversion, the voltage on ADVIN must fall between V_{REFH} and V_{REFL}. If ADVIN is equal to or exceeds V_{REFH}, the converter circuit converts the signal to \$3FF for a 10-bit representation or \$FF for a 8-bit representation. If ADVIN is equal to or less than V_{REFL}, the converter circuit converts it to \$000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions.

NOTE

Input voltage must not exceed the analog supply voltages.

The ADC10 can perform an analog-to-digital conversion on one of the software selectable channels. The output of the input multiplexer (ADVIN) is converted by a successive approximation algorithm into a 10-bit digital result. When the conversion is completed, the result is placed in the data registers (ADRH and ADRL). In 8-bit mode, the result is rounded to 8 bits and placed in ADRL. The conversion complete flag is then set and an interrupt is generated if the interrupt has been enabled.

8.3.1 Clock Select and Divide Circuit

The clock select and divide circuit selects one of three clock sources and divides it by a configurable value to generate the input clock to the converter (ADCK). The clock can be selected from one of the following sources:

- The asynchronous clock source (ACLK) — This clock source is generated from a dedicated clock source which is enabled when the ADC10 is converting and the clock source is selected by setting the ACLKEN bit. When the ADLPC bit is clear, this clock operates from 1–2 MHz; when ADLPC is set it operates at 0.5–1 MHz. This clock is not disabled in STOP and allows conversions in stop mode for lower noise operation.
- Alternate Clock Source — This clock source is equal to the external oscillator clock or a four times the bus clock. The alternate clock source is MCU specific, see Table 8-1 to determine source and availability of this clock source option. This clock is selected when ADICLK and ACLKEN are both low.
- The bus clock — This clock source is equal to the bus frequency. This clock is selected when ADICLK is high and ACLKEN is low.

Whichever clock is selected, its frequency must fall within the acceptable frequency range for ADCK. If the available clocks are too slow, the ADC10 will not perform according to specifications. If the available clocks are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV[1:0] bits and can be divide-by 1, 2, 4, or 8.

8.3.2 Input Select and Pin Control

Only one analog input may be used for conversion at any given time. The channel select bits in ADCSC are used to select the input signal for conversion.

8.3.3 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC10 module can be configured for low power operation, long sample time, and continuous conversion.

8.3.3.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

8.3.3.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADRH and ADRL. This is indicated by the setting of the COCO bit. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADRH and ADRL if the previous data is in the process of being read while in 10-bit mode (ADRH has been read but ADRL has not). In this case the data transfer is blocked, COCO is not set, and the new result is lost. When a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled). If single conversions are enabled, this could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

8.3.3.3 Aborting Conversions

Any conversion in progress will be aborted when:

- A write to ADCSC occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCLK occurs.
- The MCU is reset.
- The MCU enters stop mode with ACLK not enabled.

When a conversion is aborted, the contents of the data registers, ADRH and ADRL, are not altered but continue to be the values transferred after the completion of the last successful conversion. In the case that the conversion was aborted by a reset, ADRH and ADRL return to their reset states.

Upon reset or when a conversion is otherwise aborted, the ADC10 module will enter a low power, inactive state. In this state, all internal clocks and references are disabled. This state is entered asynchronously and immediately upon aborting of a conversion.

8.3.3.4 Total Conversion Time

The total conversion time depends on many factors such as sample time, bus frequency, whether ACLKEN is set, and synchronization time. The total conversion time is summarized in Table 8-1.

Table 8-1. Total Conversion Time versus Control Conditions

Conversion Mode	ACLKEN	Maximum Conversion Time
8-Bit Mode (short sample — ADLSMP = 0): Single or 1st continuous Single or 1st continuous Subsequent continuous ($f_{Bus} \geq f_{ADCK}$)	 0 1 X	 18 ADCK + 3 bus clock 18 ADCK + 3 bus clock + 5 µs 16 ADCK
8-Bit Mode (long sample — ADLSMP = 1): Single or 1st continuous Single or 1st continuous Subsequent continuous ($f_{Bus} \geq f_{ADCK}$)	 0 1 X	 38 ADCK + 3 bus clock 38 ADCK + 3 bus clock + 5 µs 36 ADCK
10-Bit Mode (short sample — ADLSMP = 0): Single or 1st continuous Single or 1st continuous Subsequent continuous ($f_{Bus} \geq f_{ADCK}$)	 0 1 X	 21 ADCK + 3 bus clock 21 ADCK + 3 bus clock + 5 µs 19 ADCK
10-Bit Mode (long sample — ADLSMP = 1): Single or 1st continuous Single or 1st continuous Subsequent continuous ($f_{Bus} \geq f_{ADCK}$)	 0 1 X	 41 ADCK + 3 bus clock 41 ADCK + 3 bus clock + 5 µs 39 ADCK

The maximum total conversion time for a single conversion or the first conversion in continuous conversion mode is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK and ACLKEN bits, and the divide ratio is specified by the ADIV bits. For example, if the alternate clock source is 16 MHz and is selected as the input clock source, the input clock divide-by-8 ratio is selected and the bus frequency is 4 MHz, then the conversion time for a single 10-bit conversion is:

$$\text{Maximum Conversion time} = \frac{\text{21 ADCK cycles}}{\text{16 MHz/8}} + \frac{\text{3 bus cycles}}{\text{4 MHz}} = 11.25\ \mu\text{s}$$

$$\text{Number of bus cycles} = 11.25\ \mu\text{s} \times 4\ \text{MHz} = 45\ \text{cycles}$$

NOTE

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet A/D specifications.

8.3.4 Sources of Error

Several sources of error exist for ADC conversions. These are discussed in the following sections.

8.3.4.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 15 kΩ and input capacitance of approximately 10 pF, sampling to within

1/4LSB (at 10-bit resolution) can be achieved within the minimum sample window (3.5 cycles / 2 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 10 kΩ. Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

8.3.4.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than V_{ADVIN} / (4096*I_{Leak}) for less than 1/4LSB leakage error (at 10-bit resolution).

8.3.4.3 Noise-Induced Errors

System noise which occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC10 accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1µF low-ESR capacitor from V_{REFH} to V_{REFL} (if available).
- There is a 0.1µF low-ESR capacitor from V_{DDA} to V_{SSA} (if available).
- If inductive isolation is used from the primary supply, an additional 1µF capacitor is placed from V_{DDA} to V_{SSA} (if available).
- V_{SSA} and V_{REFL} (if available) is connected to V_{SS} at a quiet point in the ground plane.
- The MCU is placed in wait mode immediately after initiating the conversion (next instruction after write to ADCSC).
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC10. In these cases, or when the MCU cannot be placed in wait or I/O activity cannot be halted, the following recommendations may reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor on the selected input channel to V_{REFL} or V_{SSA} (if available). This will improve noise issues but will affect sample rate based on the external analog source resistance.
- Operate the ADC10 in stop mode by setting ACLKEN, selecting the channel in ADCSC, and executing a STOP instruction. This will reduce V_{DD} noise but will increase effective conversion time due to stop recovery.
- Average the input by converting the output many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ACLKEN=1) and averaging. Noise that is synchronous to the ADCK cannot be averaged out.

8.3.4.4 Code Width and Quantization Error

The ADC10 quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points from one code to the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1\text{LSB} = (V_{REFH} - V_{REFL}) / 2^N$$

Because of this quantization, there is an inherent quantization error. Because the converter performs a conversion and then rounds to 8 or 10 bits, the code will transition when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2LSB in 8- or 10-bit mode. As a consequence, however, the code width of the first ($000) conversion is only 1/2LSB and the code width of the last ($FF or $3FF) is 1.5LSB.

8.3.4.5 Linearity Errors

The ADC10 may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the user should be aware of them because they affect overall accuracy. These errors are:

- Zero-Scale Error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2LSB). Note, if the first conversion is $001, then the difference between the actual $001 code width and its ideal (1LSB) is used.
- Full-Scale Error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5LSB). Note, if the last conversion is $3FE, then the difference between the actual $3FE code width and its ideal (1LSB) is used.
- Differential Non-Linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral Non-Linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total Unadjusted Error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

8.3.4.6 Code Jitter, Non-Monotonicity and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

- Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 LSB but will increase with noise.
- Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage.
- Missing codes are those which are never converted for any input value. In 8-bit or 10-bit mode, the ADC10 is guaranteed to be monotonic and to have no missing codes.

8.4 Interrupts

When AIEN is set, the ADC10 is capable of generating a CPU interrupt after each conversion. A CPU interrupt is generated when the conversion completes (indicated by COCO being set). COCO will set at the end of a conversion regardless of the state of AIEN.

8.5 Low-Power Modes

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

8.5.1 Wait Mode

The ADC10 will continue the conversion process and will generate an interrupt following a conversion if AIEN is set. If the ADC10 is not required to bring the MCU out of wait mode, ensure that the ADC10 is not in continuous conversion mode by clearing ADCO in the ADC10 status and Control Register before executing the WAIT instruction. In single conversion mode the ADC10 automatically enters a low-power state when the conversion is complete. It is not necessary to set the channel select bits (ADCH[4:0]) to all 1s to enter a low power state.

8.5.2 Stop Mode

If ACLKEN is clear, executing a STOP instruction will abort the current conversion and place the ADC10 in a low-power state. Upon return from stop mode, a write to ADCSC is required to resume conversions, and the result stored in ADRH and ADRL will represent the last completed conversion until the new conversion completes.

If ACLKEN is set, the ADC10 continues normal operation during stop mode. The ADC10 will continue the conversion process and will generate an interrupt following a conversion if AIEN is set. If the ADC10 is not required to bring the MCU out of stop mode, ensure that the ADC10 is not in continuous conversion mode by clearing ADCO in the ADC10 status and Control Register before executing the STOP instruction. In single conversion mode the ADC10 automatically enters a low-power state when the conversion is complete. It is not necessary to set the channel select bits (ADCH[4:0]) to all 1s to enter a low-power state.

If ACLKEN is set, a conversion can be initiated while in stop using the external hardware trigger ADEXTCO when in external convert mode. The ADC10 will operate in a low-power mode until the trigger is asserted, at which point it will perform a conversion and assert the interrupt when complete (if AIEN is set).

8.6 ADC10 During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. BCFE in the break flag control register (BFCR) enables software to clear status bits during the break state. See BFCR in the SIM section of this data sheet.

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

To protect status bits during the break state, write a 0 to BCFE. With BCFE cleared (its default state), software can read and write 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 cleared. After the break, doing the second step clears the status bit.

8.7 Input/Output Signals

The ADC10 module shares its pins with general-purpose input/output (I/O) port pins. The ADC10 on this MCU uses V_{DD} and V_{SS} as its supply and reference pins. This MCU does not have an external trigger source.

8.7.1 ADC10 Analog Power Pin (V_{DDA})

The ADC10 analog portion uses V_{DDA} as its power pin. In some packages, V_{DDA} is connected internally to V_{DD}. If externally available, connect the V_{DDA} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDA} for good results.

NOTE

If externally available, route V_{DDA} carefully for maximum noise immunity and place bypass capacitors as near as possible to the package.

8.7.2 ADC10 Analog Ground Pin (V_{SSA})

The ADC10 analog portion uses V_{SSA} as its ground pin. In some packages, V_{SSA} is connected internally to V_{SS}. If externally available, connect the V_{SSA} pin to the same voltage potential as V_{SS}.

In cases where separate power supplies are used for analog and digital power, the ground connection between these supplies should be at the V_{SSA} pin. This should be the only ground connection between these supplies if possible. The V_{SSA} pin makes a good single point ground location.

8.7.3 ADC10 Voltage Reference High Pin (V_{REFH})

V_{REFH} is the power supply for setting the high-reference voltage for the converter. In some packages, V_{REFH} is connected internally to V_{DDA}. If externally available, V_{REFH} may be connected to the same potential as V_{DDA}, or may be driven by an external source that is between the minimum V_{DDA} spec and the V_{DDA} potential (V_{REFH} must never exceed V_{DDA}).

NOTE

> *Route V_{REFH} carefully for maximum noise immunity and place bypass capacitors as near as possible to the package.*

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as close as possible to the package pins. Resistance in the path is not recommended because the current will cause a voltage drop which could result in conversion errors. Inductance in this path must be minimum (parasitic only).

8.7.4 ADC10 Voltage Reference Low Pin (V_{REFL})

V_{REFL} is the power supply for setting the low-reference voltage for the converter. In some packages, V_{REFL} is connected internally to V_{SSA}. If externally available, connect the V_{REFL} pin to the same voltage potential as V_{SSA}. There will be a brief current associated with V_{REFL} when the sampling capacitor is charging. If externally available, connect the V_{REFL} pin to the same potential as V_{SSA} at the single point ground location.

8.7.5 ADC10 Channel Pins (ADn)

The ADC10 has multiple input channels. Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. 0.01 μF capacitors with good high-frequency characteristics are sufficient. These capacitors are not necessary in all cases, but when used they must be placed as close as possible to the package pins and be referenced to V_{SSA}.

8.8 Registers

These registers control and monitor operation of the ADC10:

- ADC10 status and control register, ADCSC
- ADC10 data registers, ADRH and ADRL
- ADC10 clock register, ADCLK

8.8.1 ADC10 Status and Control Register

This section describes the function of the ADC10 status and control register (ADCSC). Writing ADCSC aborts the current conversion and initiates a new conversion (if the ADCH[4:0] bits are equal to a value other than all 1s).

Address: $003C

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

▨ = Unimplemented

Figure 8-3. ADC10 Status and Control Register (ADCSC)

COCO — Conversion Complete Bit

The COCO bit is a read-only bit which is set each time a conversion is completed. This bit is cleared whenever the status and control register is written or whenever the data register (low) is read.

1 = Conversion completed
0 = Conversion not completed

AIEN — ADC10 Interrupt Enable Bit

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

1 = ADC10 interrupt enabled
0 = ADC10 interrupt disabled

ADCO — ADC10 Continuous Conversion Bit

When written high, the ADC10 will begin to convert samples continuously (continuous conversion mode) and update the result registers at the end of each conversion, provided the ADCH[4:0] bits do not decode to all 1s. The ADC10 will continue to convert until the MCU enters reset, the MCU enters stop mode (if ACLKEN is clear), the ADCLK register is written, or until the ADCSC is written again. If Stop is entered (with ACLKEN low), continuous conversions will cease and can only be restarted with a write to the ADCSC. Any write to the ADCSC with the ADCO bit set and the ADCH bits not all 1s will abort the current conversion and begin continuous conversions.

If the bus frequency is less than the ADCK frequency, precise sample time for continuous conversions cannot be guaranteed in short-sample mode (ADLSMP = 0). If the bus frequency is less than 1/11th of the ADCK frequency, precise sample time for continuous conversions cannot be guaranteed in long-sample mode (ADLSMP = 1).

When clear, the ADC10 will perform a single conversion (single conversion mode) each time the ADCSC is written (assuming the ADCH[4:0] bits do not decode all 1s). Reset clears the ADCO bit.

1 = Continuous conversion following a write to the ADCSC
0 = One conversion following a write to the ADCSC

ADCH[4:0] — Channel Select Bits

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of the input channels. The input channels are detailed in Table 8-2.

The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC10 and isolation of the input channel from the I/O pad. Terminating continuous convert mode this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC10 in a low-power state, however, because the module is automatically placed in a low-power state when a conversion completes.

Table 8-2. Input Channel Select[1]

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	Input Select[2]
0	0	0	0	0	AD0
0	0	0	0	1	AD1
0	0	0	1	0	AD2
0	0	0	1	1	AD3
0	0	1	0	0	AD4
0	0	1	0	1	AD5
0	0	1	1	0	Unused
Continuing to:					Unused
1	1	0	0	1	Unused
1	1	0	1	0	BANDGAP REF[3]
1	1	0	1	1	Reserved
1	1	1	0	0	Reserved
1	1	1	0	1	V_{REFH}
1	1	1	1	0	V_{REFL}
1	1	1	1	1	Low-power state

1. Accuracy is guaranteed for conversions on the selected channel only if V_{DDA} falls in the specified range.
2. If any unused or reserved channels are selected, the resulting conversion will be unknown.
3. Requires LVI to be powered (LVIPWRD = 0 in CONFIG1).

8.8.2 ADC10 Result High Register (ADRH)

This register holds the MSB's of the result and is updated each time a conversion completes. All other bits read as 0s. Reading ADRH prevents the ADC10 from transferring subsequent conversion results into the results registers until ADRL is read. If ADRL is not read until the after next conversion is completed, then the intermediate conversion results will be lost. In 8-bit mode, this register contains no interlocking with ADRL.

Address: $003D

	Bit 7	6	5	4	3	2	1	Bit 0
Read: (8-bit mode)	0	0	0	0	0	0	0	0
Read: (10-bit mode)	0	0	0	0	0	0	AD9	AD8
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 8-4. ADC10 Data Register High (ADRH)

8.8.3 ADC10 Result Low Register (ADRL)

This register holds the LSB's of the result. This register is updated each time a conversion completes. Reading ADRH prevents the ADC10 from transferring subsequent conversion results into the results registers until ADRL is read. If ADRL is not read until the after next conversion is completed, then the intermediate conversion results will be lost. In 8-bit mode, there is no interlocking with ADRH.

Address: $003E

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

R = Reserved

Figure 8-5. ADC10 Data Register Low (ADRL)

8.8.4 ADC10 Clock Register (ADCLK)

This register selects the clock frequency for the ADC10 and the modes of operation.

Address: $003F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	ADLPC	ADIV1	ADIV0	ADICLK	MODE1	MODE0	ADLSMP	ACLKEN
Reset:	0	0	0	0	0	0	0	0

Figure 8-6. ADC10 Clock Register (ADCLK)

ADLPC — ADC10 Low-Power Configuration Bit

ADLPC controls the speed and power configuration of the successive approximation converter. This is used to optimize power consumption when higher sample rates are not required.

1 = Low-power configuration: The power is reduced at the expense of maximum clock speed.
0 = High-speed configuration

ADIV[1:0] — ADC10 Clock Divider Bits

ADIV1 and ADIV0 select the divide ratio used by the ADC10 to generate the internal clock ADCK. Table 8-3 shows the available clock configurations.

Table 8-3. ADC10 Clock Divide Ratio

ADIV1	ADIV0	Divide Ratio (ADIV)	Clock Rate
0	0	1	Input clock ÷ 1
0	1	2	Input clock ÷ 2
1	0	4	Input clock ÷ 4
1	1	8	Input clock ÷ 8

ADICLK — Input Clock Select Bit

If ACLKEN is clear, ADICLK selects either the bus clock or an alternate clock source as the input clock source to generate the internal clock ADCK. If the alternate clock source is less than the minimum clock speed, use the internally-generated bus clock as the clock source. As long as the internal clock ADCK, which is equal to the selected input clock divided by ADIV, is at a frequency (f_{ADCK}) between the minimum and maximum clock speeds (considering ALPC), correct operation can be guaranteed.

1 = The internal bus clock is selected as the input clock source
0 = The alternate clock source IS SELECTED

MODE[1:0] — 10- or 8-Bit or External-Triggered Mode Selection

This bit selects between 10- or 8-bit operation. The successive approximation converter generates a result which is rounded to 8- or 10-bit value based on the mode selection. This rounding process sets the transfer function to transition at the midpoint between the ideal code voltages, causing a quantization error of 1/2LSB. Reset returns 8-bit mode.

Table 8-4. Mode Selection

MODE1	MODE0	Mode
0	0	8-bit, right-justified, ADCSC write-triggered mode enabled
0	1	10-bit, right-justified, ADCSC write-triggered mode enabled
1	0	Reserved.
1	1	Reserved.

ADLSMP — Long Sample Time Configuration

This bit configures the sample time of the ADC10 to either 3.5 or 23.5 ADCK clock cycles. This adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption in continuous conversion mode if high conversion rates are not required.

1 = Long sample time (23.5 cycles)
0 = Short sample time (3.5 cycles)

ACLKEN — Asynchronous Clock Source Enable

This bit enables the asynchronous clock source as the input clock to generate the internal clock ADCK, and allows operation in stop mode. The asynchronous clock source will operate between 1 MHz and 2 MHz if the ADLPC bit is clear, and between 0.5 MHz and 1 MHz if the ADLPC bit is set. As long as the internal clock ADCK, which is equal to the selected input clock divided by ADIV, is at a frequency (f_{ADCK}) between the minimum and maximum required clock frequencies (considering ALPC), correct operation is guaranteed.

1 = The asynchronous clock is selected as the input clock source (the clock generator is only enabled during the conversion)
0 = The ADICLK bit specifies the input clock source and conversions will not continue in stop mode

Chapter 9
Liquid Crystal Display (LCD) Driver

9.1 Introduction

This section describes the liquid crystal display (LCD) driver module. The LCD driver module can drive a maximum of 25 frontplanes and 4 backplanes, depending on the LCD duty selected.

9.2 Features

Features of the LCD driver module include the following:

- Software programmable driver segment configurations:
  - 24 frontplanes × 4 backplanes (96 segments)
  - 25 frontplanes × 3 backplanes (75 segments)
  - 25 frontplanes × 1 backplane (25 segments)
- LCD bias voltages generated by internal resistor ladder
- Software programmable contrast control

9.3 Pin Name Conventions and I/O Register Addresses

Three dedicated I/O pins are for the backplanes, BP0–BP2; twenty four frontplanes, FP1–FP24, are shared with port B, C, D, and E pins. FP0 and BP3 shares the same pin and configured by the DUTY[1:0] bits in the LCD clock register.

The full names of the LCD output pins are shown in Table 9-1. The generic pin names appear in the text that follows.

Table 9-1. Pin Name Conventions

LCD Generic Pin Name	Full MCU Pin Name	Pin Selected for LCD Function by:
FP0/BP3	FP0/BP3	—
BP0–BP2	BP0–BP2	—
FP1–FP2	PTB6/FP1–PTB7/FP2	LCDE in LCDCR
FP3–FP10	PTE0/FP3–PTE7/FP10	PEE in CONFIG2 LCDE in LCDCR
FP11–FP18	PTD0/FP11–PTD7/FP18	PDE in CONFIG2 LCDE in LCDCR
FP19–FP24	PTC0/FP19–PTC5/FP24	PCEL:PCEH in CONFIG2 LCDE in LCDCR

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$004F	LCD Clock Register (LCDCLK)	Read:	0	FCCTL1	FCCTL0	DUTY1	DUTY0	LCLK2	LCLK1	LCLK0
		Write:	▨							
		Reset:	0	0	0	0	0	0	0	0
$0051	LCD Control Register (LCDCR)	Read:	LCDE	0	FC	LC	LCCON3	LCCON2	LCCON1	LCCON0
		Write:		▨						
		Reset:	0	0	0	0	0	0	0	0
$0052	LCD Data Register 1 (LDAT1)	Read:	F1B3	F1B2	F1B1	F1B0	F0B3	F0B2	F0B1	F0B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0053	LCD Data Register 2 (LDAT2)	Read:	F3B3	F3B2	F3B1	F3B0	F2B3	F2B2	F2B1	F2B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0054	LCD Data Register 3 (LDAT3)	Read:	F5B3	F5B2	F5B1	F5B0	F4B3	F4B2	F4B1	F4B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0055	LCD Data Register 4 (LDAT4)	Read:	F7B3	F7B2	F7B1	F7B0	F6B3	F6B2	F6B1	F6B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0056	LCD Data Register 5 (LDAT5)	Read:	F9B3	F9B2	F9B1	F9B0	F8B3	F8B2	F8B1	F8B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0057	LCD Data Register 6 (LDAT6)	Read:	F11B3	F11B2	F11B1	F11B0	F10B3	F10B2	F10B1	F10B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0058	LCD Data Register 7 (LDAT7)	Read:	F13B3	F13B2	F13B1	F13B0	F12B3	F12B2	F12B1	F12B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$0059	LCD Data Register 8 (LDAT8)	Read:	F15B3	F15B2	F15B1	F15B0	F14B3	F14B2	F14B1	F14B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005A	LCD Data Register 9 (LDAT9)	Read:	F17B3	F17B2	F17B1	F17B0	F16B3	F16B2	F16B1	F16B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005B	LCD Data Register 10 (LDAT10)	Read:	F19B3	F19B2	F19B1	F19B0	F18B3	F18B2	F18B1	F18B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005C	LCD Data Register 11 (LDAT11)	Read:	F21B3	F21B2	F21B1	F21B0	F20B3	F20B2	F20B1	F20B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005D	LCD Data Register 12 (LDAT12)	Read:	F23B3	F23B2	F23B1	F23B0	F22B3	F22B2	F22B1	F22B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005E	LCD Data Register 13 (LDAT13)	Read:	0	0	0	0	F24B3	F24B2	F24B1	F24B0
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	U	U	U	U

U = Unaffected ▨ = Unimplemented

Figure 9-1. LCD I/O Register Summary

9.4 Functional Description

Figure 9-2 shows a block diagram of the LCD driver module, and Figure 9-3 shows a simplified schematic of the LCD system.

The LCD driver module uses a 1/3 biasing method. The LCD power is supplied by the V_{LCD} pin. Voltages V_{LCD1}, V_{LCD2}, and V_{LCD3} are generated by an internal resistor ladder.

The LCD data registers, LDAT1–LDAT13, control the LCD segments' ON/OFF, with each data register controlling two frontplanes. When a logic 1 is written to a FxBx bit in the data register, the corresponding frontplane-backplane segment will turn ON. When a logic 0 is written, the segment will turn OFF.

When the LCD driver module is disabled (LCDE = 0), the LCD display will be OFF, all backplane and frontplane drivers have the same potential as V_{DD}. The resistor ladder is disconnected from V_{DD} to reduce power consumption.

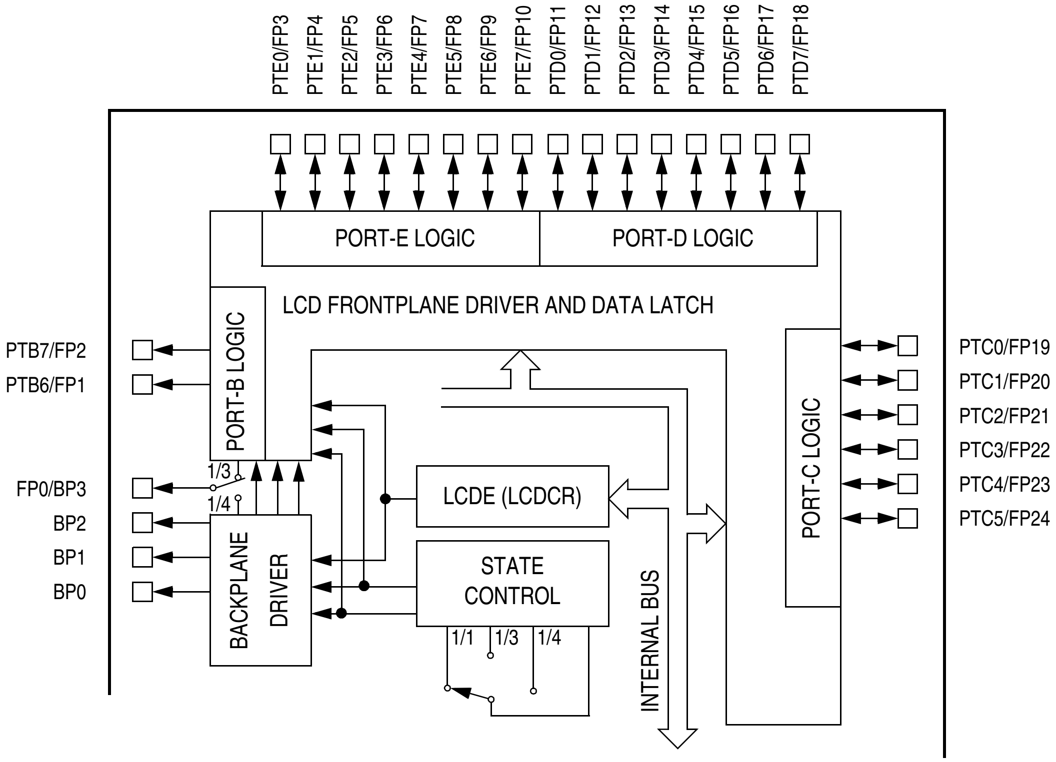


Figure 9-2. LCD Block Diagram

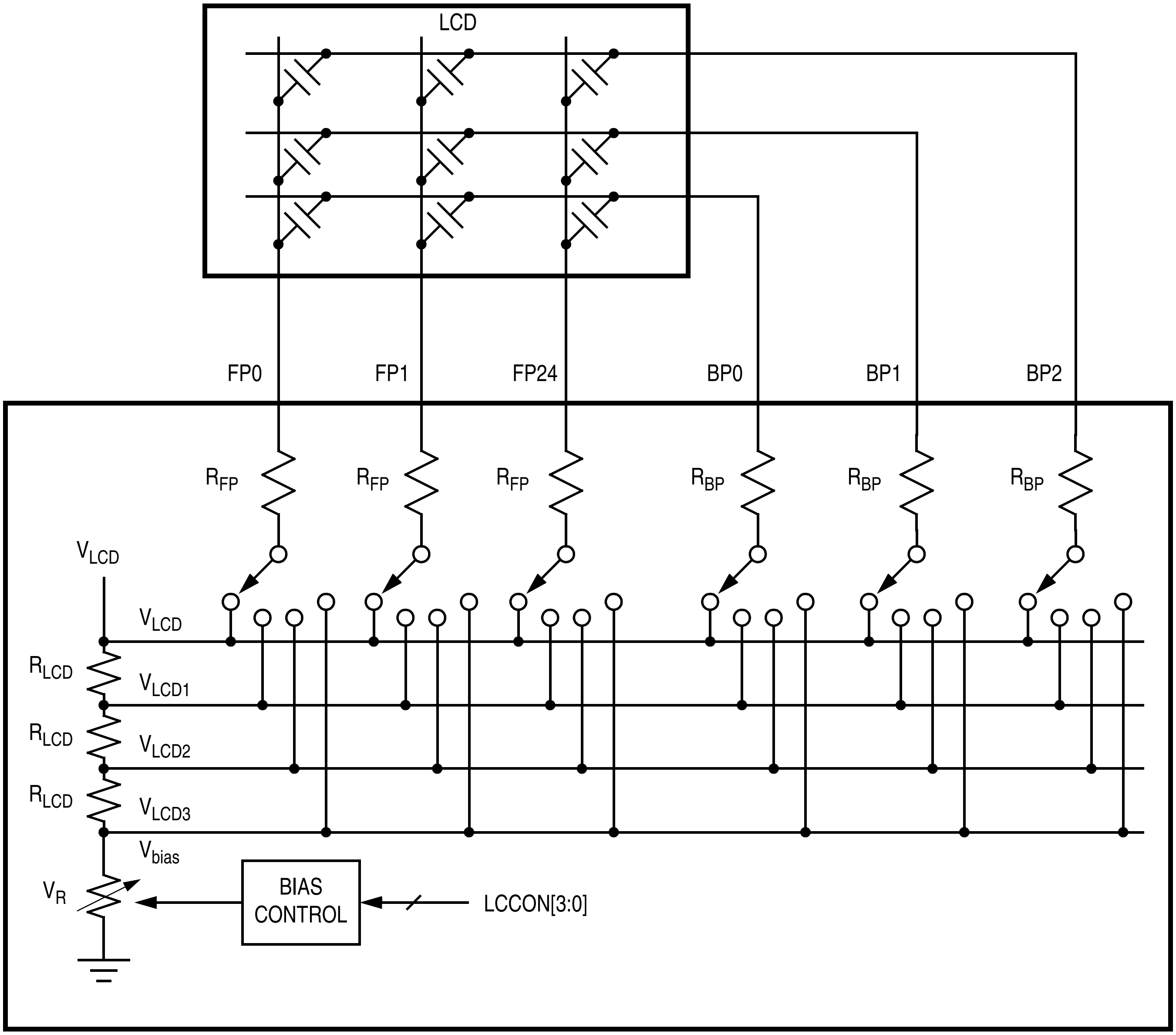


Figure 9-3. Simplified LCD Schematic (1/3 Duty, 1/3 Bias)

9.4.1 LCD Duty

The setting of the LCD output waveform duty is dependent on the number of backplane drivers required. Three LCD duties are available:

- Static duty — BP0 is used only
- 1/3 duty — BP0, BP1, and BP3 are used
- 1/4 duty — BP0, BP1, BP2, and BP3 are used

When the LCD driver module is enabled the backplane waveforms for the selected duty are driven out of the backplane pins. The backplane waveforms are periodic and are shown are shown in Figure 9-5, Figure 9-6, and Figure 9-7.

9.4.2 LCD Voltages (V_{LCD}, V_{LCD1}, V_{LCD2}, V_{LCD3})

The voltage V_{LCD} is from the V_{LCD} pin and must not exceed V_{DD}. V_{LCD1}, V_{LCD2}, and V_{LCD3} are internal bias voltages for the LCD driver waveforms. They are derived from V_{LCD} using a resistor ladder (see Figure 9-3).

The relative potential of the LCD voltages are:

- $V_{LCD} = V_{DD}$
- $V_{LCD1} = 2/3 \times (V_{LCD} - V_{bias})$
- $V_{LCD2} = 1/3 \times (V_{LCD} - V_{bias})$
- $V_{LCD3} = V_{bias}$

The V_{LCD3} bias voltage, V_{bias}, is controlled by the LCD contrast control bits, LCCON[2:0].

9.4.3 LCD Cycle Frame

The LCD driver module uses the CGMXCLK (see Chapter 5 Clock Generator Module (CGM)) as the input reference clock. This clock is divided to produce the LCD waveform base clock, LCDCLK, by configuring the LCLK[2:0] bits in the LCD clock register. The LCDCLK clocks the backplane and the frontplane output waveforms.

The LCD cycle frame is determined by the equation:

$$\text{LCD CYCLE FRAME} = \frac{1}{\text{LCD WAVEFORM BASE CLOCK} \times \text{DUTY}}$$

For example, for 1/3 duty and 256 Hz waveform base clock:

$$\begin{aligned}\text{LCD CYCLE FRAME} &= \frac{1}{256 \times (1/3)} \\ &= 11.72\ \text{ms}\end{aligned}$$

9.4.4 Fast Charge and Low Current

The default value for each of the bias resistors (see Figure 9-3), R_{LCD}, in the resistor ladder is approximately 37 kΩ at $V_{LCD} = 3\,V$. The relatively high current drain through the 37 kΩ resistor ladder may not be suitable for some LCD panel connections. Lowering this current is possible by setting the LC bit in the LCD control register, switching the R_{LCD} value to 146 kΩ.

Although the lower current drain is desirable, but in some LCD panel connections, the higher current is required to drive the capacitive load of the LCD panel. In most cases, the higher current is only required when the LCD waveforms change state (the rising and falling edges in the LCD output waveforms). The fast charge option is designed to have the high current for the switching and the low current for the steady state. Setting the FC bit in the LCD control register selects the fast charge option. The R_{LCD} value is set to 37 kΩ (for high current) for a fraction of time for each LCD waveform switching edge, and then back to 146 kΩ for the steady state period. The duration of the fast charge time is set by configuring the FCCTL[1:0] bits in the LCD clock register, and can be LCDCLK/32, LCDCLK/64, or LCDCLK/128. Figure 9-4 shows the fast charge clock relative to the BP0 waveform.

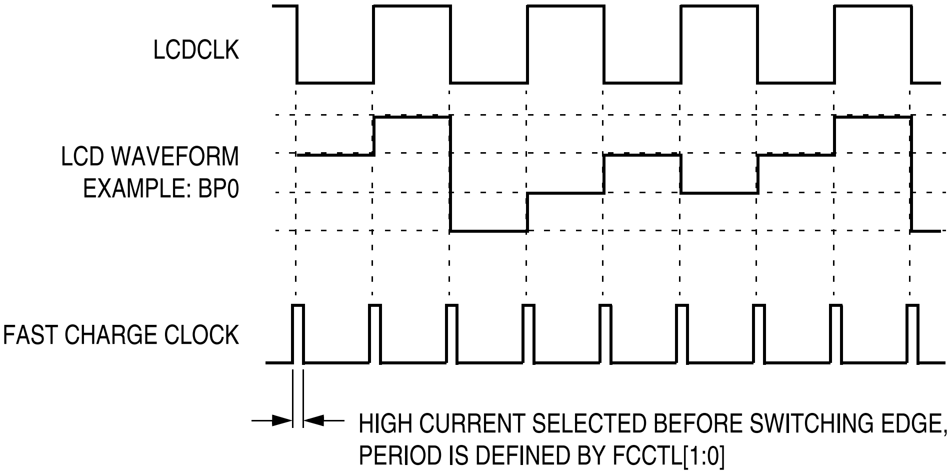


Figure 9-4. Fast Charge Timing

9.4.5 Contrast Control

The contrast of the connected LCD panel can be adjusted by configuring the LCCON[3:0] bits in the LCD control register. The LCCON[3:0] bits provide a 16-step contrast control, which adjusts the bias voltage in the resistor ladder for LCD voltage, V_{LCD3}. The relative voltages, V_{LCD1} and V_{LCD2}, are altered accordingly. For example, setting LCCON[3:0] = $F, the relative panel potential voltage ($V_{LCD} – V_{LCD3}$) is reduced from maximum 3.3V to approximate 2.45V.

The V_{LCD} voltage can be monitored by the ADC channel, ADC7, and then adjustments to the bias voltage by the user software to provide automatic contrast control.

9.5 Low-Power Modes

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

9.5.1 Wait Mode

The LCD driver module continues normal operation in wait mode. If the LCD is not required in wait mode, power down the LCD module by clearing the LCDE bit before executing the WAIT instruction.

9.5.2 Stop Mode

For continuous LCD module operation in stop mode, the oscillator stop mode enable bit (STOP_XCLKEN in CONFIG2 register) must be set before executing the STOP instruction. When STOP_XCLKEN is set, CGMXCLK continues to drive the LCD module.

If STOP_XCLKEN bit is cleared, the LCD module is inactive after the execution of a STOP instruction. The STOP instruction does not affect LCD register states. LCD module operation resumes after an external interrupt. To further reduce power consumption, the LCD module should be powered-down by clearing the LCDE bit before executing the STOP instruction.

9.6 I/O Signals

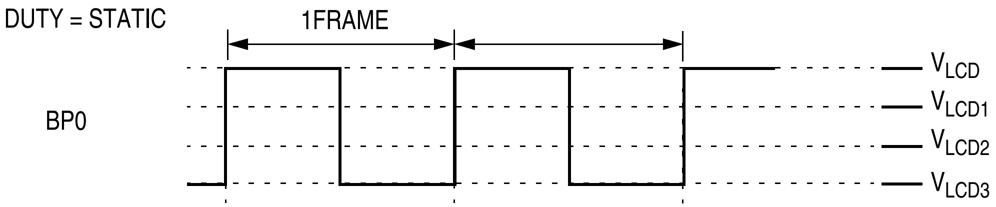
The LCD driver module has twenty-eight (28) output pins.

- FP0/BP3 (multiplexed; selected as FP0 or BP3 by DUTY[1:0])
- BP0–BP2
- FP1–FP2 (shared with port B)
- FP3–FP10 (shared with port E)
- FP11–FP18 (shared with port D)
- FP19–FP24 (shared with port C)

9.6.1 BP0–BP3 (Backplane Drivers)

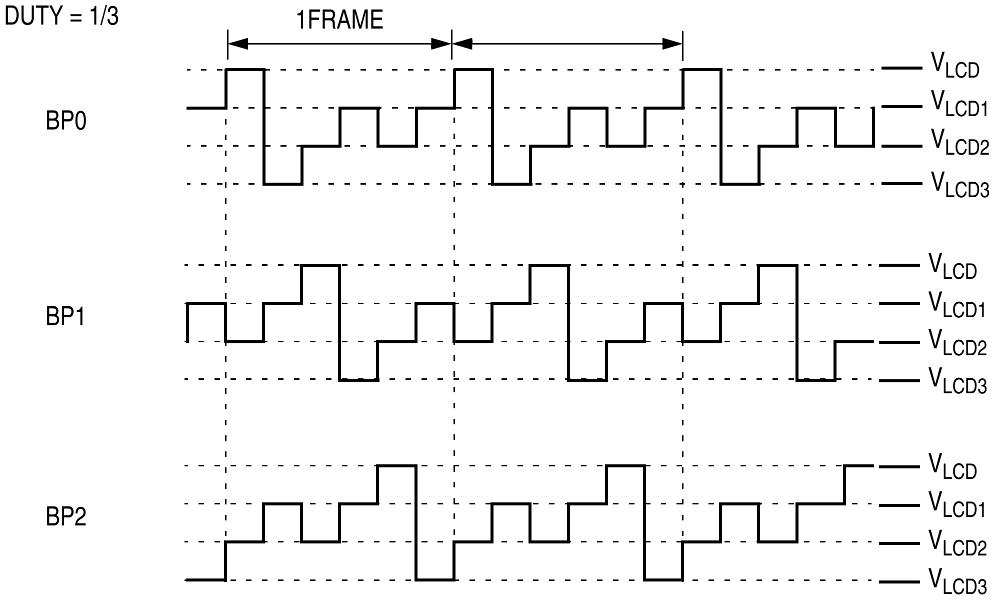
BP0–BP3 are the backplane driver output pins. These are connected to the backplane of the LCD panel. Depending on the LCD duty selected, the voltage waveforms in Figure 9-5, Figure 9-6, and Figure 9-7 appear on the backplane pins.

BP3 pin is only used when 1/4 duty is selected. The pin becomes FP0 for static and 1/3 duty operations.



NOTES:
1. BP1, BP2, and BP3 are not used.
2. At static duty, 1FRAME is equal to the cycle of LCD waveform base clock.

Figure 9-5. Static LCD Backplane Driver Waveform



NOTES:
1. BP3 is not used.
2. At 1/3 duty, 1FRAME has three times the cycle of LCD waveform base clock.

Figure 9-6. 1/3 Duty LCD Backplane Driver Waveforms

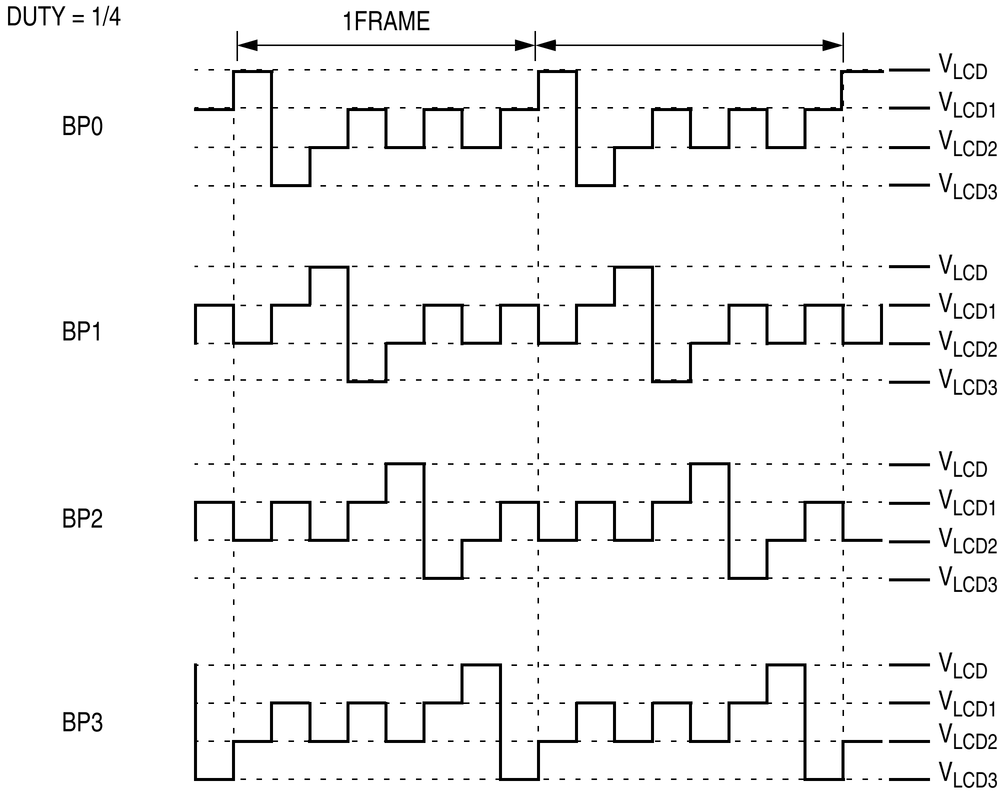


Figure 9-7. 1/4 Duty LCD Backplane Driver Waveforms

9.6.2 FP0–FP24 (Frontplane Drivers)

FP0–FP24 are the frontplane driver output pins. These are connected to the frontplane of the LCD panel. Depending on LCD duty selected and the contents in the LCD data registers, the voltage waveforms in Figure 9-8, Figure 9-9, Figure 9-10 and Figure 9-11 appear on the frontplane pins.

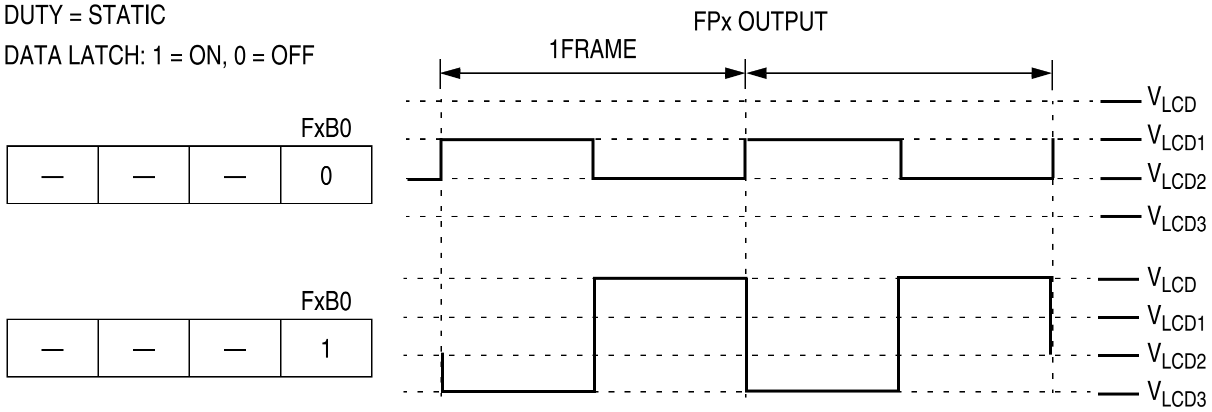


Figure 9-8. Static LCD Frontplane Driver Waveforms

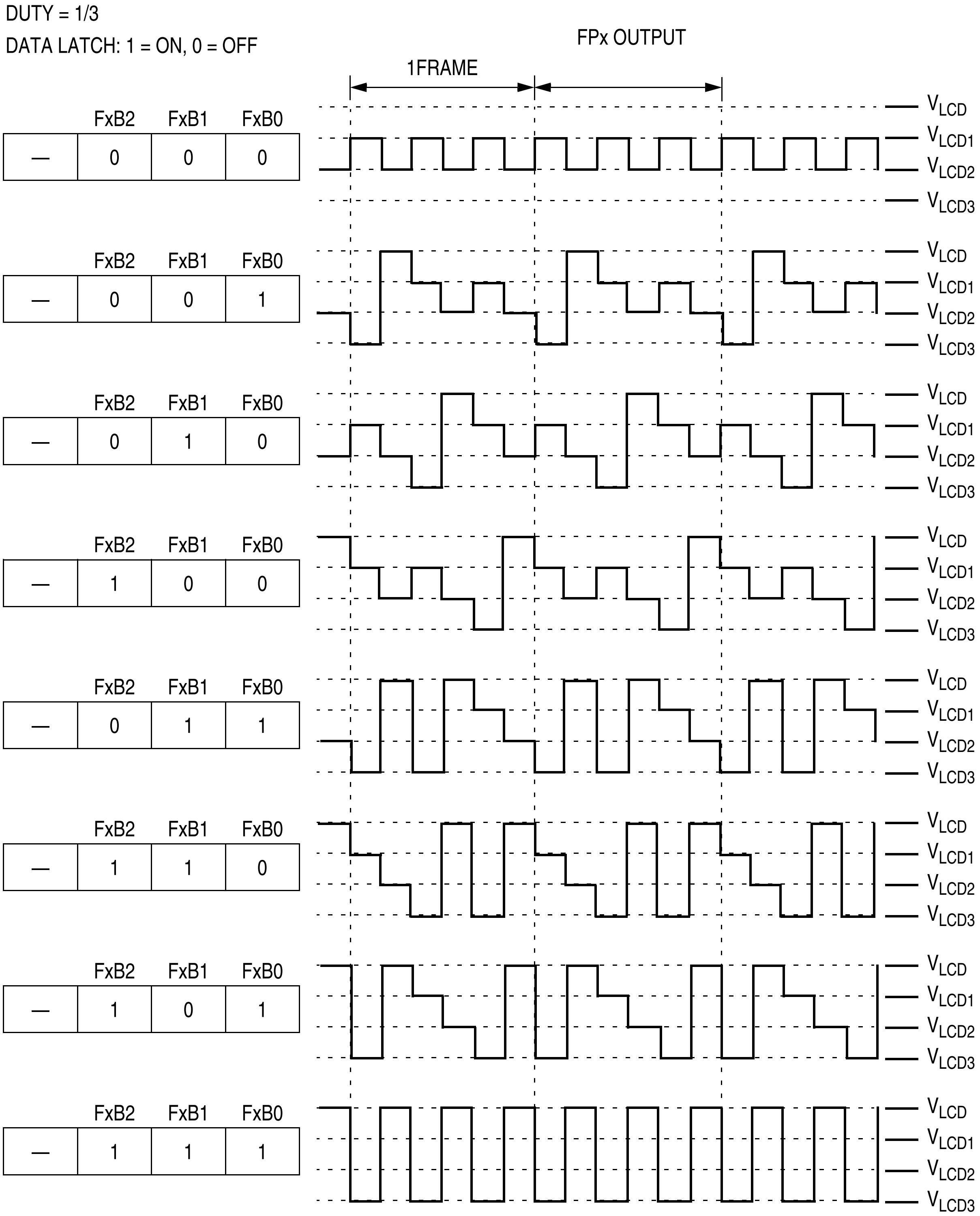


Figure 9-9. 1/3 Duty LCD Frontplane Driver Waveforms

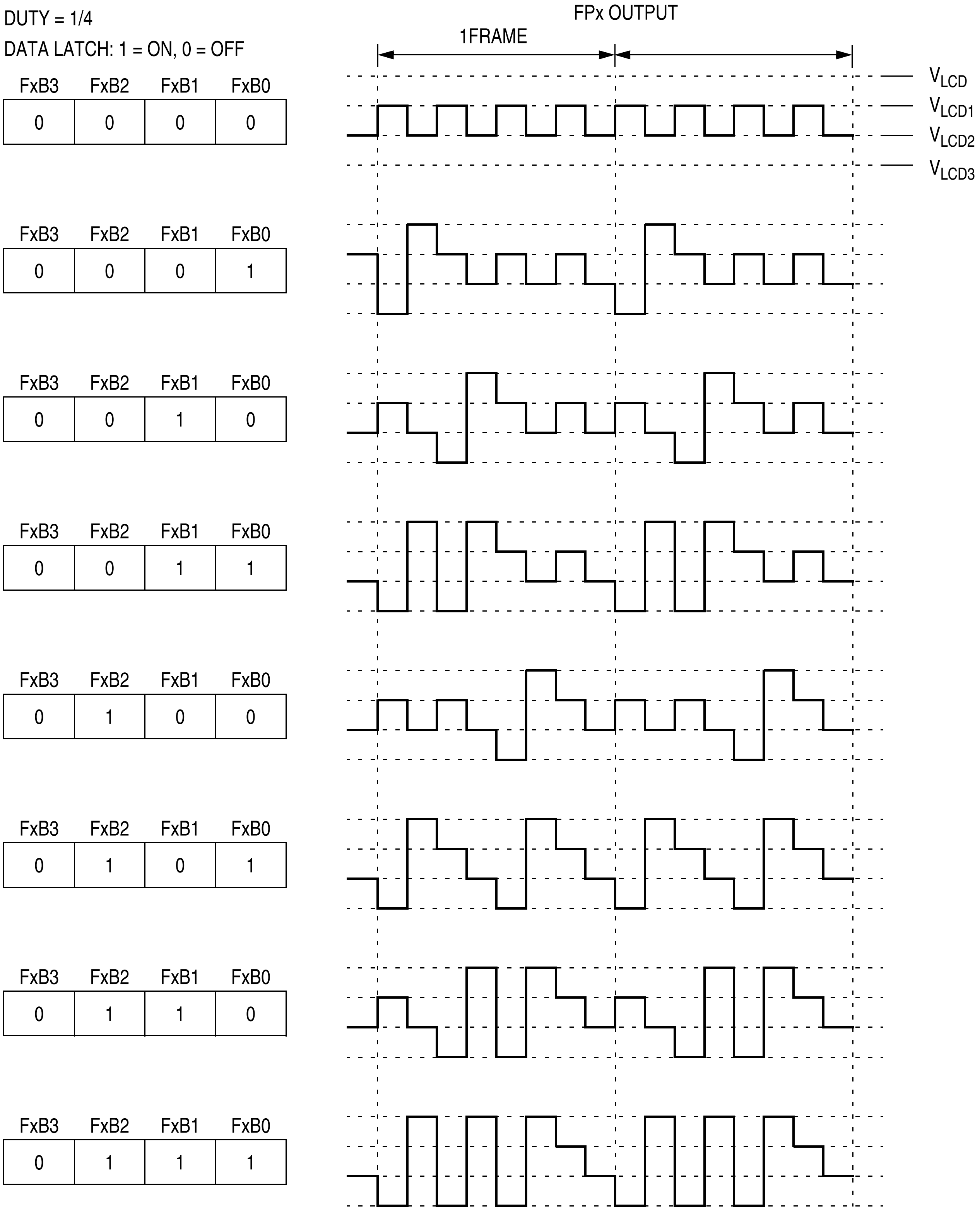


Figure 9-10. 1/4 Duty LCD Frontplane Driver Waveforms

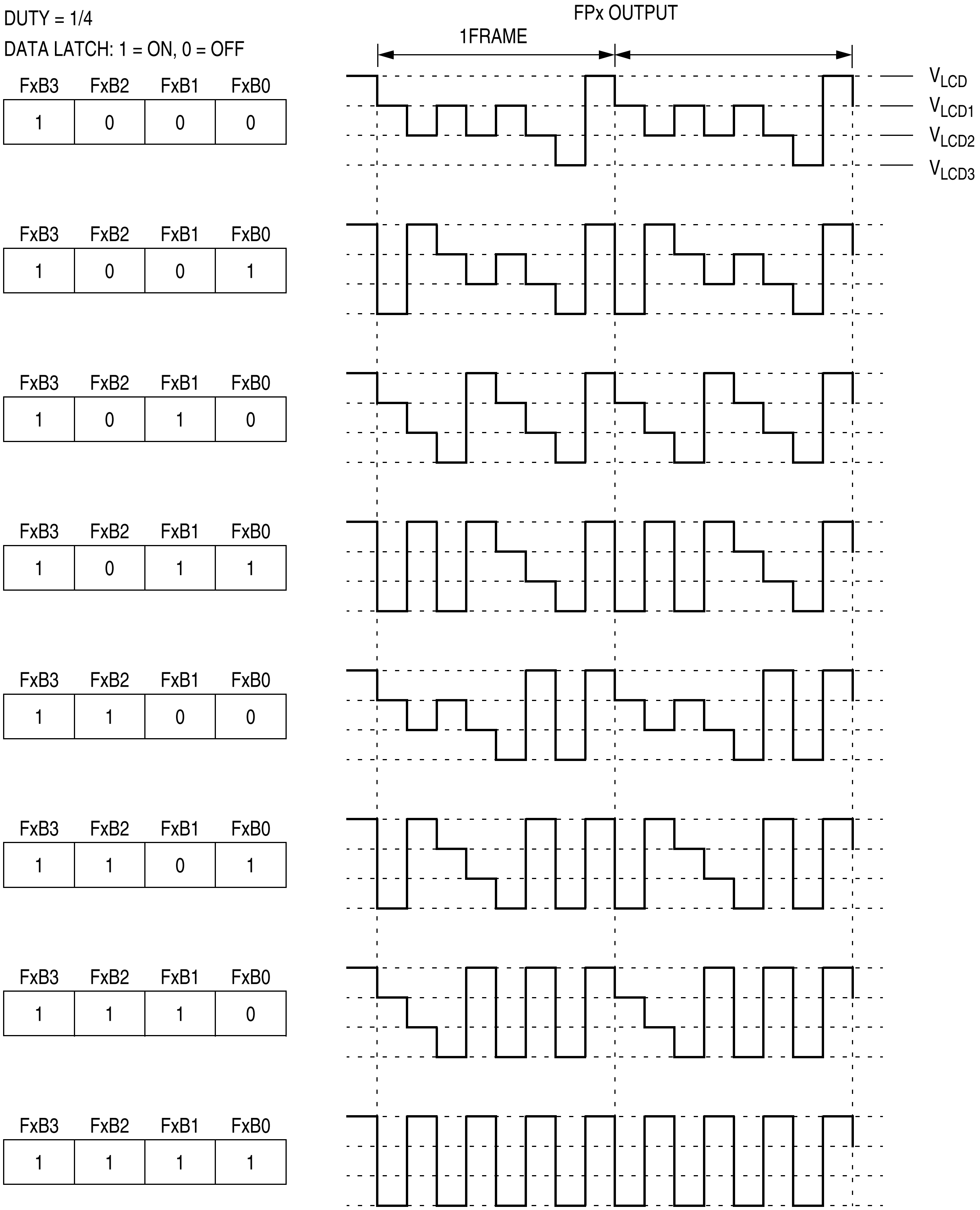
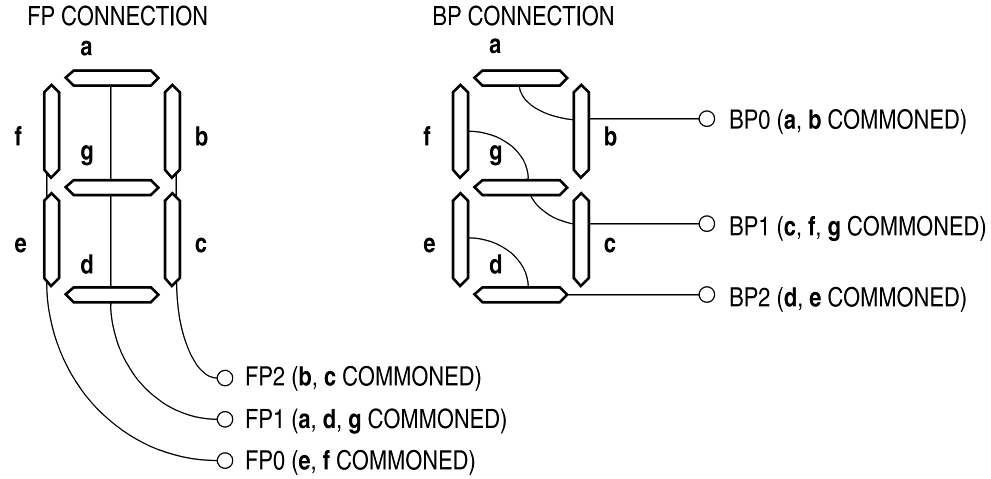


Figure 9-11. 1/4 Duty LCD Frontplane Driver Waveforms (continued)

9.7 Seven Segment Display Connection

The following shows an example for connecting a 7-segment LCD display to the LCD driver.

The example uses 1/3 duty cycle, with pins BP0, BP1, BP2, FP0, FP1, and FP2 connected as shown in Figure 9-12. The output waveforms are shown in Figure 9-13.



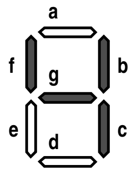
The segment assignments for each bit in the data registers are:

	F1B3	F1B2	F1B1	F1B0	F0B3	F0B2	F0B1	F0B0
LDAT1 $0052	—	d	g	a	—	e	f	—
	FP1				FP0			

	F3B3	F3B2	F3B1	F3B0	F2B3	F2B2	F2B1	F2B0
LDAT2 $0053	—	—	—	—	—	—	c	b
					FP2			

To display the character "4": LDAT1 = X010X01X, LDAT2 = XXXXXX11

LDAT1 $0052	X	0	1	0	X	0	1	X
LDAT2 $0053	X	X	X	X	X	X	1	1



X = don’t care

Figure 9-12. 7-Segment Display Example

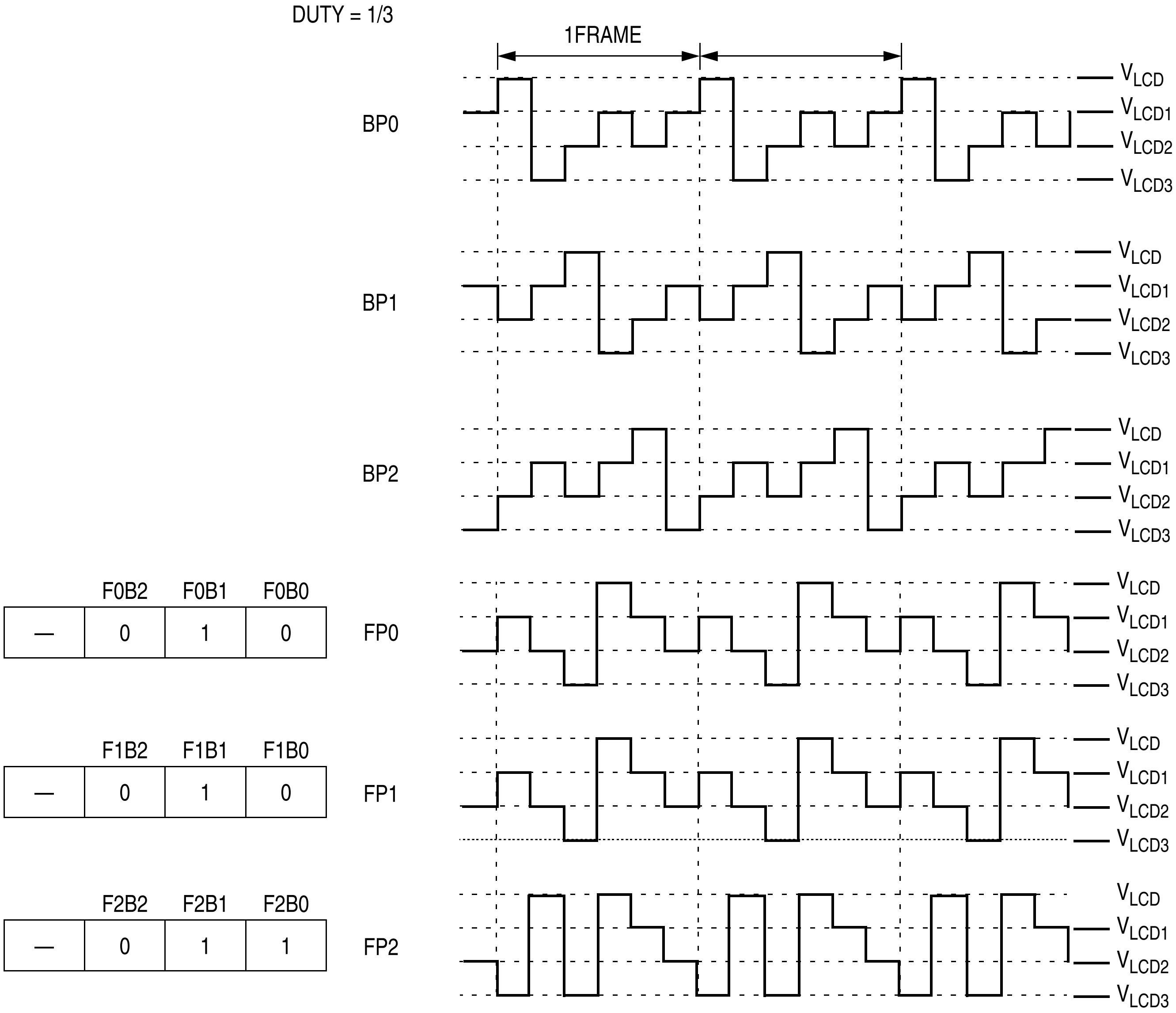


Figure 9-13. BP0–BP2 and FP0–FP2 Output Waveforms for 7-Segment Display Example

The voltage waveform across the "f" segment of the LCD (between BP1 and FP0) is illustrated in Figure 9-14. As shown in the waveform, the voltage peaks reach the LCD-ON voltage, V_{LCD}, therefore, the segment will be ON.

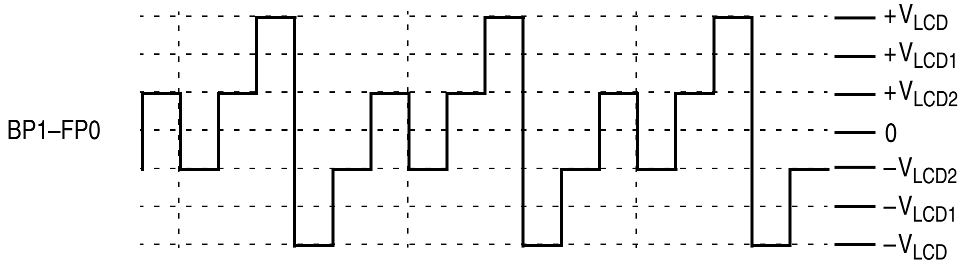


Figure 9-14. "f" Segment Voltage Waveform

The voltage waveform across the "e" segment of the LCD (between BP2 and FP0) is illustrated in Figure 9-15. As shown in the waveform, the voltage peaks do not reach the LCD-ON voltage, V_{LCD}, therefore, the segment will be OFF.

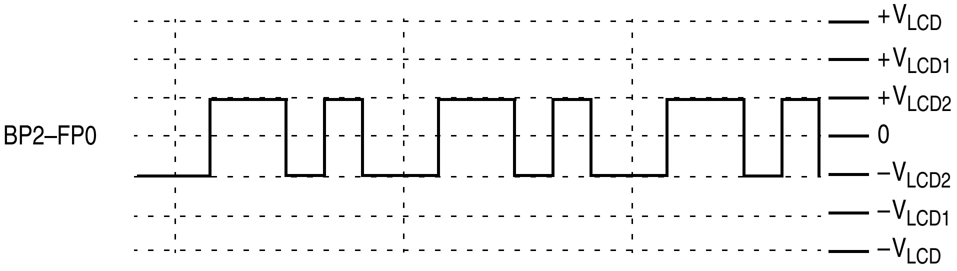


Figure 9-15. "e" Segment Voltage Waveform

9.8 I/O Registers

Fifteen (15) registers control LCD driver module operation:

- LCD control register (LCDCR)
- LCD clock register (LCDCLK)
- LCD data registers (LDAT1–LDAT13)

9.8.1 LCD Control Register (LCDCR)

The LCD control register (LCDCR):

- Enables the LCD driver module
- Selects bias resistor value and fast-charge control
- Selects LCD contrast

Address: $0051

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LCDE	0	FC	LC	LCCON3	LCCON2	LCCON1	LCCON0
Write:		▨						
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 9-16. LCD Control Register (LCDCR)

LCDE — LCD Enable

This read/write bit enables the LCD driver module; the backplane and frontplane drive LCD waveforms out of BPx and FPx pins. Reset clears the LCDE bit.

1 = LCD driver module enabled
0 = LCD driver module disabled

FC — Fast Charge
LC — Low Current

These read/write bits are used to select the value of the resistors in resistor ladder for LCD voltages. Reset clears the FC and LC bits.

Table 9-2. Resistor Ladder Selection

FC	LC	Action
X	0	Each resistor is approximately 37 kΩ (default)
0	1	Each resistor is approximately 146 kΩ
1	1	Fast charge mode

LCCON[3:0] — LCD Contrast Control

These read/write bits select the bias voltage, V_{bias}. This voltage controls the contrast of the LCD. Maximum contrast is set when LCCON[3:0] =%0000;
minimum contrast is set when LCCON[3:0] =%1111.

Table 9-3. LCD Bias Voltage Control

LCCON3	LCCON2	LCCON1	LCCON0	Bias Voltage (approximate % of V_{DD})
0	0	0	0	0.6
0	0	0	1	2.9
0	0	1	0	5.2
0	0	1	1	7.4
0	1	0	0	9.6
0	1	0	1	11.6
0	1	1	0	13.5
0	1	1	1	15.3
1	0	0	0	17.2
1	0	0	1	18.8
1	0	1	0	20.5
1	0	1	1	22.0
1	1	0	0	23.6
1	1	0	1	25.0
1	1	1	0	26.4
1	1	1	1	27.7

9.8.2 LCD Clock Register (LCDCLK)

The LCD clock register (LCDCLK):

- Selects the fast charge duty cycle
- Selects LCD driver duty cycle
- Selects LCD waveform base clock

Address: $004F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	FCCTL1	FCCTL0	DUTY1	DUTY0	LCLK2	LCLK1	LCLK0
Write:	▨							
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 9-17. LCD Clock Register (LCDCLK)

FCCTL[1:0] — Fast Charge Duty Cycle Select

These read/write bits select the duty cycle of the fast charge duration. Reset clears these bits. (See 9.4.4 Fast Charge and Low Current)

Table 9-4. Fast Charge Duty Cycle Selection

FCCTL1:FCCTL0	Fast Charge Duty Cycle
00	In each LCDCLK/2 period, each bias resistor is reduced to 37 kΩ for a duration of LCDCLK/32.
01	In each LCDCLK/2 period, each bias resistor is reduced to 37 kΩ for a duration of LCDCLK/64.
10	In each LCDCLK/2 period, each bias resistor is reduced to 37 kΩ for a duration of LCDCLK/128.
11	Not used

DUTY[1:0] — Duty Cycle Select

These read/write bits select the duty cycle of the LCD driver output waveforms. The multiplexed FP0/BP3 pin is controlled by the duty cycle selected. Reset clears these bits.

Table 9-5. LCD Duty Cycle Selection

DUTY1:DUTY0	Description
00	Static selected; FP0/BP3 pin function as FP0.
01	1/3 duty cycle selected; FP0/BP3 pin functions as FP0.
10	1/4 duty cycle selected; FP0/BP3 pin functions as BP3.
11	Not used

LCLK[2:0] — LCD Waveform Base Clock Select

These read/write bits selects the LCD waveform base clock. Reset clears these bits.

Table 9-6. LCD Waveform Base Clock Selection

LCLK2	LCLK1	LCLK0	Divide Ratio	LCD Waveform Base Clock Frequency LCDCLK (Hz)		LCD Frame Rate f_{XTAL}(1) = 32.768 kHz		LCD Frame Rate f_{XTAL} = 4.9152 MHz	
				f_{XTAL} = 32.768 kHz	f_{XTAL} = 4.9152 MHz	1/3 duty	1/4 duty	1/3 duty	1/4 duty
0	0	0	128	256	—	85.3	64	—	—
0	0	1	256	128	—	42.7	32	—	—
0	1	0	512	64	—	21.3	16	—	—
0	1	1	1024	32	—	10.7	8	—	—
1	0	0	16384	—	300	—	—	100	75
1	0	1	32768	—	150	—	—	50	37.5
1	1	0	65536	—	75	—	—	25	18.75
1	1	1	Reserved						

1. f_{XTAL} is the same as CGMXCLK (see Chapter 5 Clock Generator Module (CGM)).

9.8.3 LCD Data Registers (LDAT1–LDAT17)

The thirteen (13) LCD data registers enable and disable the drive to the corresponding LCD segments.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0052	LCD Data Register 1 (LDAT1)	Read: Write:	F1B3	F1B2	F1B1	F1B0	F0B3	F0B2	F0B1	F0B0
		Reset:	U	U	U	U	U	U	U	U
$0053	LCD Data Register 2 (LDAT2)	Read: Write:	F3B3	F3B2	F3B1	F3B0	F2B3	F2B2	F2B1	F2B0
		Reset:	U	U	U	U	U	U	U	U
$0054	LCD Data Register 3 (LDAT3)	Read: Write:	F5B3	F5B2	F5B1	F5B0	F4B3	F4B2	F4B1	F4B0
		Reset:	U	U	U	U	U	U	U	U
$0055	LCD Data Register 4 (LDAT4)	Read: Write:	F7B3	F7B2	F7B1	F7B0	F6B3	F6B2	F6B1	F6B0
		Reset:	U	U	U	U	U	U	U	U
$0056	LCD Data Register 5 (LDAT5)	Read: Write:	F9B3	F9B2	F9B1	F9B0	F8B3	F8B2	F8B1	F8B0
		Reset:	U	U	U	U	U	U	U	U
$0057	LCD Data Register 6 (LDAT6)	Read: Write:	F11B3	F11B2	F11B1	F11B0	F10B3	F10B2	F10B1	F10B0
		Reset:	U	U	U	U	U	U	U	U
$0058	LCD Data Register 7 (LDAT7)	Read: Write:	F13B3	F13B2	F13B1	F13B0	F12B3	F12B2	F12B1	F12B0
		Reset:	U	U	U	U	U	U	U	U

U = Unaffected ▨ = Unimplemented

Figure 9-18. LCD Data Registers 1–13 (LDAT1–LDAT13)

$0059	LCD Data Register 8 (LDAT8)	Read:	F15B3	F15B2	F15B1	F15B0	F14B3	F14B2	F14B1	F14B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005A	LCD Data Register 9 (LDAT9)	Read:	F17B3	F17B2	F17B1	F17B0	F16B3	F16B2	F16B1	F16B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005B	LCD Data Register 10 (LDAT10)	Read:	F19B3	F19B2	F19B1	F19B0	F18B3	F18B2	F18B1	F18B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005C	LCD Data Register 11 (LDAT11)	Read:	F21B3	F21B2	F21B1	F21B0	F20B3	F20B2	F20B1	F20B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005D	LCD Data Register 12 (LDAT12)	Read:	F23B3	F23B2	F23B1	F23B0	F22B3	F22B2	F22B1	F22B0
		Write:								
		Reset:	U	U	U	U	U	U	U	U
$005E	LCD Data Register 13 (LDAT13)	Read:	0	0	0	0	F24B3	F24B2	F24B1	F24B0
		Write:	▨	▨	▨	▨				
		Reset:	0	0	0	0	U	U	U	U

U = Unaffected ▨ = Unimplemented

Figure 9-18. LCD Data Registers 1–13 (LDAT1–LDAT13)

Chapter 10
Input/Output (I/O) Ports

10.1 Introduction

Forty (40) bidirectional input-output (I/O) pins form six parallel ports. All I/O pins are programmable as inputs or outputs.

NOTE

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC)	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$0008	Data Direction Register E (DDRE)	Read: Write:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2	DDRE1	DDRE0
		Reset:	0	0	0	0	0	0	0	0
$0009	Port E Data Register (PTE)	Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
		Reset:	Unaffected by reset							
$000C	Port-B High Current Drive Control Register (HDB)	Read:	R	PPI1L	HDB5	HDB4	HDB3	HDB2	PPI1CLKS1	PPI1CLKS0
		Write:		▨						
		Reset:			0	0	0	0	0	0

U = Unaffected

X = Indeterminate

▨ = Unimplemented

R = Reserved

Figure 10-1. I/O Port Register Summary

Table 10-1. Port Control Register Bits Summary (Sheet 1 of 2)

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
A	0	DDRA0	KBI	KBIER ($001C)	KBIE0	PTA0/KBI0
	1	DDRA1			KBIE1	PTA1/KBI1
	2	DDRA2			KBIE2	PTA2/KBI2
	3	DDRA3			KBIE3	PTA3/KBI3
	4	DDRA4	ADC	ADSCR ($003C)	ADCH[4:0]	PTA4/ADC0
	5	DDRA5				PTA5/ADC1
	6	DDRA6				PTA6/ADC2
	7	DDRA7				PTA7/ADC3
B	0	DDRB0	ADC	ADSCR ($003C)	ADCH[4:0]	PTB0/ADC4
	1	DDRB1				PTB1/ADC5
	2	DDRB2	TIM1	T1SC0 ($0025) HDB ($000C)	ELS0B:ELS0A PPI1CLKS[1:0]	PTB2/T1CH0/PPIECK
	3	DDRB3		T1SC1 ($0028)	ELS1B:ELS1A	PTB3/T1CH1
	4	DDRB4	TIM2	T2SC0 ($0030)	ELS0B:ELS0A	PTB4/T2CH0
	5	DDRB5		T2SC1 ($0033)	ELS1B:ELS1A	PTB5/T2CH1
	6	DDRB6	LCD	LCDCR ($0051)	LCDE	PTB6/FP1
	7	DDRB7				PTB7/FP2
C	0	DDRC0	LCD	CONFIG2 ($001D) LCDCR ($0051)	PCEL LCDE	PTC0/FP19
	1	DDRC1				PTC1/FP20
	2	DDRC2				PTC2/FP21
	3	DDRC3				PTC3/FP22
	4	DDRC4			PCEH LCDE	PTC4/FP23
	5	DDRC5				PTC5/FP24
	6	DDRC6	—	—	—	PTC6
	7	DDRC7	—	—	—	PTC7
D	0	DDRD0	LCD	CONFIG2 ($001D) LCDCR ($0051)	PDE LCDE	PTD0/FP11
	1	DDRD1				PTD1/FP12
	2	DDRD2				PTD2/FP13
	3	DDRD3				PTD3/FP14
	4	DDRD4				PTD4/FP15
	5	DDRD5				PTD5/FP16
	6	DDRD6				PTD6/FP17
	7	DDRD7				PTD7/FP18

Table 10-1. Port Control Register Bits Summary (Sheet 2 of 2)

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
E	0	DDRE0	LCD	CONFIG2 ($001D) LCDCR ($0051)	PEE LCDE	PTE0/FP3
	1	DDRE1				PTE1/FP4
	2	DDRE2				PTE2/FP5
	3	DDRE3				PTE3/FP6
	4	DDRE4				PTE4/FP7
	5	DDRE5				PTE5/FP8
	6	DDRE6				PTE6/FP9
	7	DDRE7				PTE7/FP10

10.2 Port A

Port A is an 8-bit special function port that shares four of its port pins with the analog-to-digital converter (ADC) module and four of its port pins with the keyboard interrupt module (KBI).

10.2.1 Port A Data Register (PTA)

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

Address: $0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
Reset:	Unaffected by Reset							
Alternative Function:	ADC3	ADC2	ADC1	ADC0	KBI3	KBI2	KBI1	KBI0

Figure 10-2. Port A Data Register (PTA)

PTA[7:0] — Port A Data Bits

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

KBI[3:0] — Keyboard Interrupt Channels 3 to 0

KBI[3:0] are pins used for the keyboard interrupt input. The corresponding input, KBI[3:0], can be enabled in the keyboard interrupt enable register, KBIER. Port pins used as KBI input will override any control from the port I/O logic. See Section 20. Keyboard Interrupt Module (KBI).

ADC[3:0] — ADC channels 0 to 3

ADC[3:0] are pins used for the input channels to the analog-to-digital converter module. The channel select bits, ADCH[4:0], in the ADC status and control register define which port pin will be used as an ADC input and overrides any control from the port I/O logic. See Section 16. Analog-to-Digital Converter (ADC).

NOTE

Care must be taken when reading port A while applying analog voltages to ADC[3:0] pins. If the appropriate ADC channel is not enabled, excessive current drain may occur if analog voltages are applied to the PTAx/ADCx pin, while PTA is read as a digital input. Those ports not selected as analog input channels are considered digital I/O ports.

10.2.2 Data Direction Register A (DDRA)

Data direction register A determines whether each port A pin is an input or an output. Writing a 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 10-3. Data Direction Register A (DDRA)

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

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

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

NOTE

Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1. Figure 10-4 shows the port A I/O logic.

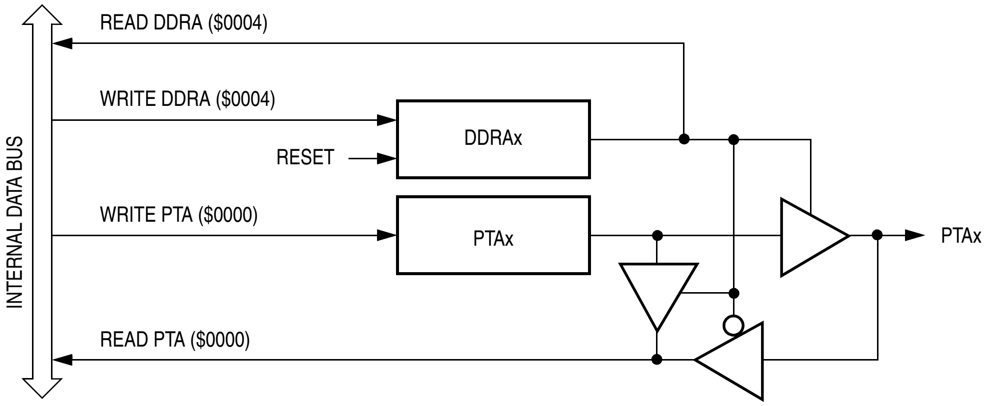


Figure 10-4. Port A I/O Circuit

When DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When 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 10-2 summarizes the operation of the port A pins.

Table 10-2. Port A Pin Functions

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

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

10.3 Port B

Port B is an 8-bit special function port that shares two of its port pins with the analog-to-digital converter (ADC) module, four of its port pins with the two timers (TIM1 and TIM2), and two of its ports pins with the liquid crystal display (LCD) driver module. Port pin, PTB2, is also shared with the external clock input of the programmable periodic interrupt (PPI) module.

10.3.1 Port B Data Register (PTB)

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

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Reset:	Unaffected by reset							
Alternative Functions:	FP2	FP1	T2CH1	T2CH0	T1CH1	T1CH0	ADC5	ADC4
						PPIECK		
Additional Functions:			High current sink					

Figure 10-5. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

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

ADC[5:4] — ADC Channels 5 and 4

ADC[5:4] are pins used for the input channels to the analog-to-digital converter module. The channel select bits, ADCH[4:0], in the ADC status and control register define which port pin will be used as an ADC input and overrides any control from the port I/O logic. See Section 16. Analog-to-Digital Converter (ADC).

NOTE

When a pin is to be used as an ADC channel, the user must make sure that any pin that is shared with another module is disabled and pin is configured as input port.

T1CH[1:0] — Timer 1 Channel I/O Bits

The T1CH1 and T1CH0 pins are the TIM1 input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTB2/T1CH0 and PTB3/T1CH1 pins are timer channel I/O pins or general-purpose I/O pins. See Chapter 6 Timer Interface Module (TIM).

T2CH[1:0] — Timer 2 Channel I/O Bits

The T2CH1 and T2CH0 pins are the TIM1 input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTB4/T2CH0 and PTB5/T2CH1 pins are timer channel I/O pins or general-purpose I/O pins. See Chapter 6 Timer Interface Module (TIM).

PPIECK — External Clock Source Input for PPI

The PPIECK pin is the external clock input to the PPI module. It is selected by setting the bits PPI1CLKS[1:0] = 01 in the port B high current drive control register. See 7.6.1 PPI Clock Source Select and Interrupt Latch.

FP[2:1] — LCD Driver Frontplanes 2–1

FP[2:1] are pins used for the frontplane output of the LCD driver module. The enable bit, LCDE, in the LCDCR register determine whether the PTB7/FP2–PTB6/FP1 pins are LCD frontplane driver pins or general-purpose I/O pins. See Chapter 9 Liquid Crystal Display (LCD) Driver.

10.3.2 Data Direction Register B (DDRB)

Data direction register B determines whether each port B pin is an input or an output. Writing a logic 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 10-6. Data Direction Register B (DDRB)

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

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

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

NOTE

Avoid glitches on port B pins by writing to the port B data register before changing data direction register B bits from 0 to 1. Figure 10-7 shows the port B I/O logic.

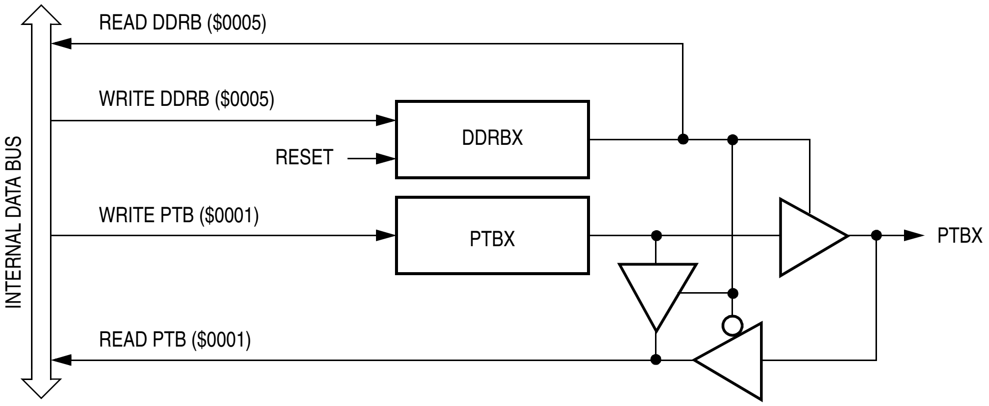


Figure 10-7. Port B I/O Circuit

When DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When 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 10-3 summarizes the operation of the port B pins.

Table 10-3. Port B Pin Functions

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

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

10.3.3 Port B High Current Drive Control Register (HDB)

The port-B high current drive control register (HDB) controls the high current drive capability on PTB[5:2]. Each bit is individually configurable and requires that the data direction register, DDRB, bit be configured as an output.

Address: $000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	PPI1L	HDB5	HDB4	HDB3	HDB2	PPI1CLKS1	PPI1CLKS0
Write:		▨						
Reset:	0	0	0	0	0	0	0	0

Figure 10-8. Port B High Current Drive Control Register (HDB)

PPI1L — PPI1 Pending for Acknowledgement

See Chapter 7 Programmable Periodic Interrupt (PPI).

HDB[5:2] — Port B High Current Drive Enable Bits

These read/write bits are software programmable to enable the direct LED drive on an output port pin.

1 = Corresponding port B pin is configured to high current sink direct LED drive.
0 = Corresponding port B pin is configured to standard drive

PPI1CLKS[1:0] — PPI1 Clock Source Select

See Chapter 7 Programmable Periodic Interrupt (PPI).

10.4 Port C

Port C is an 8-bit special function port that shares five of its port pins with the liquid crystal display (LCD) driver module.

10.4.1 Port C Data Register (PTC)

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

Address: $0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
Reset:	Unaffected by reset							
Alternative Function:			FP24	FP23	FP22	FP21	FP20	FP19

Figure 10-9. Port C Data Register (PTC)

PTC[7:0] — Port C Data Bits

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

FP[24:19] — LCD Driver Frontplanes 24–19

FP[24:19] are pins used for the frontplane output of the LCD driver module. The enable bits, PCEH and PCEL, in the CONFIG2 register, and LCDE bit in the LCDCR register determine whether the PTC5/FP24–PTC4/FP23 and PTC3/FP22–PTC0/FP19 pins are LCD frontplane driver pins or general-purpose I/O pins. See Chapter 9 Liquid Crystal Display (LCD) Driver.

10.4.2 Data Direction Register C (DDRC)

Data direction register C determines whether each port C pin is an input or an output. Writing a 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	DDRC1	DDRC0
Reset:	0	0	0	0	0	0	0	0

Figure 10-10. Data Direction Register C (DDRC)

DDRC[7:0] — Data Direction Register C Bits

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

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

NOTE

Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1. Figure 10-11 shows the port C I/O logic.

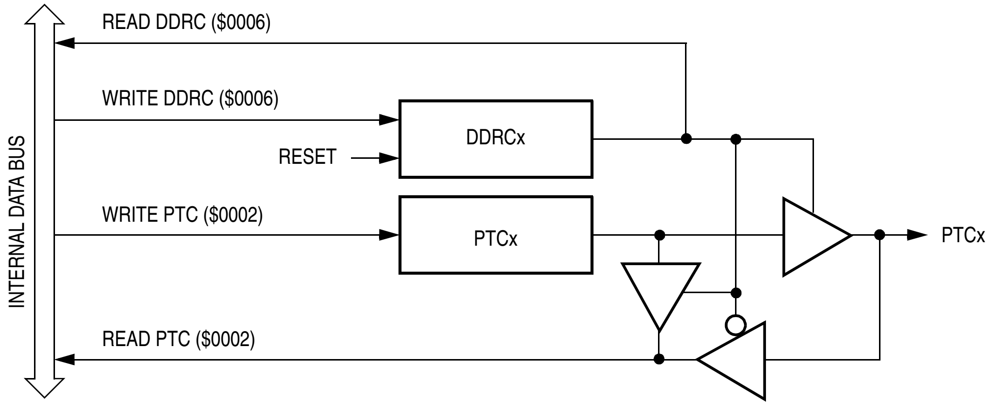


Figure 10-11. Port C I/O Circuit

When DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When 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 10-4 summarizes the operation of the port C pins.

Table 10-4. Port C Pin Functions

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

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

10.5 Port D

Port D is an 8-bit special function port that shares all of its port pins with the liquid crystal display (LCD) driver module.

10.5.1 Port D Data Register (PTD)

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

Address: $0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Reset:	Unaffected by reset							
Alternative Function:	FP18	FP17	FP16	FP15	FP14	FP13	FP12	FP11

Figure 10-12. Port D Data Register (PTD)

PTD[7:0] — 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.

FP[18:11] — LCD Driver Frontplanes 18–11

FP[18:11] are pins used for the frontplane output of the LCD driver module. The enable bit, PDE, in the CONFIG2 register and LCDE bit in the LCDCR register, determines whether the PTD7/FP18–PTD0/FP11 pins are LCD frontplane driver pins or general-purpose I/O pins. See Chapter 9 Liquid Crystal Display (LCD) Driver.

10.5.2 Data Direction Register D (DDRD)

Data direction register D determines whether each port D pin is an input or an output. Writing a 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 10-13. Data Direction Register D (DDRD)

DDRD[7:0] — Data Direction Register D Bits

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

1 = Corresponding port D pin configured as output

0 = Corresponding port D pin configured as input

NOTE

Avoid glitches on port D pins by writing to the port D data register before changing data direction register D bits from 0 to 1. Figure 10-14 shows the port D I/O logic.

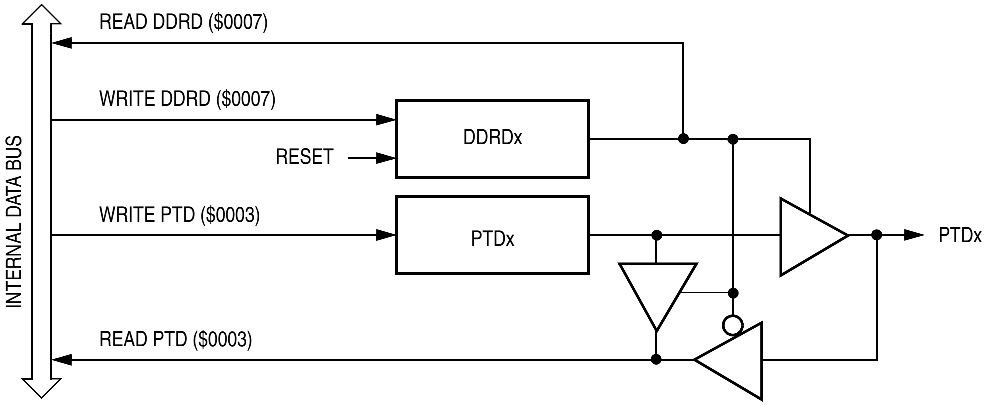


Figure 10-14. Port D I/O Circuit

When DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When 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 10-5 summarizes the operation of the port D pins.

Table 10-5. Port D Pin Functions

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

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

10.6 Port E

Port E is an 8-bit special function port that shares all of its port pins with the liquid crystal display (LCD) driver module.

10.6.1 Port E Data Register (PTE)

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

Address: $0009

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
Reset:	Unaffected by reset							
Alternative Function:	FP10	FP9	FP8	FP7	FP6	FP5	FP4	FP3

Figure 10-15. Port E Data Register (PTE)

PTE[7:0] — Port E Data Bits

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

FP[10:3] — LCD Driver Frontplanes 10–3

FP[10:3] are pins used for the frontplane output of the LCD driver module. The enable bit, PEE, in the CONFIG2 register and LCDE bit in the LCDCR register, determines whether the PTE7/FP10–PTE0/FP3 pins are LCD frontplane driver pins or general-purpose I/O pins. See Chapter 9 Liquid Crystal Display (LCD) Driver.

10.6.2 Data Direction Register E (DDRE)

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

Address: $0008

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

Figure 10-16. Data Direction Register E (DDRE)

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

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

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

NOTE

Avoid glitches on port E pins by writing to the port E data register before changing data direction register E bits from 0 to 1. Figure 10-14 shows the port E I/O logic.

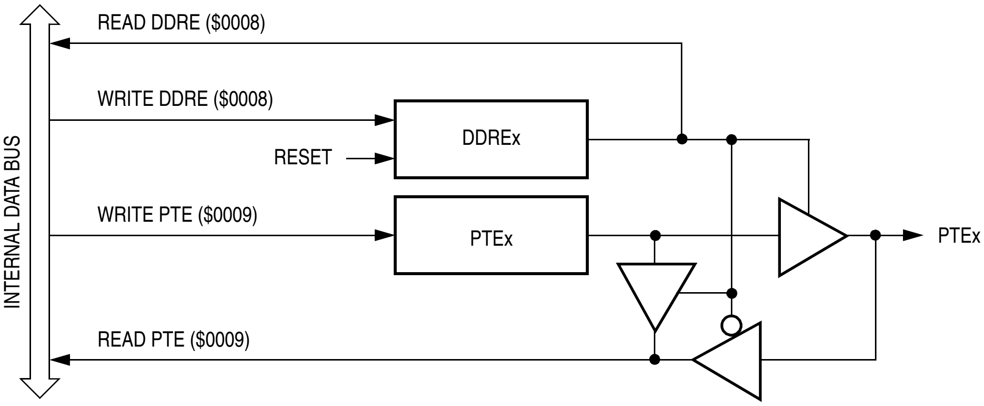


Figure 10-17. Port E I/O Circuit

When DDREx is a logic 1, reading address $0009 reads the PTEx data latch. When DDREx 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 10-5 summarizes the operation of the port E pins.

Table 10-6. Port E Pin Functions

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

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

Chapter 11
External Interrupt (IRQ)

11.1 Introduction

The external interrupt (IRQ) module provides a maskable interrupt input.

11.2 Features

Features of the IRQ module include the following:

- A dedicated external interrupt pin ($\overline{\text{IRQ}}$)
- IRQ interrupt control bits
- Hysteresis buffer
- Spike filter
- Programmable edge-only or edge and level interrupt sensitivity
- Automatic interrupt acknowledge
- Selectable internal pullup resistor

11.3 Functional Description

A logic zero applied to the external interrupt pin can latch a CPU interrupt request. Figure 11-1 shows the structure of the IRQ module.

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

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the IRQ latch.
- Software clear — Software can clear the interrupt latch by writing to the acknowledge bit in the interrupt status and control register (INTSCR). Writing a logic one to the ACK bit clears the IRQ latch.
- Reset — A reset automatically clears the interrupt latch.

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

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

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

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

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

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

NOTE

The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See 4.5 Exception Control.)

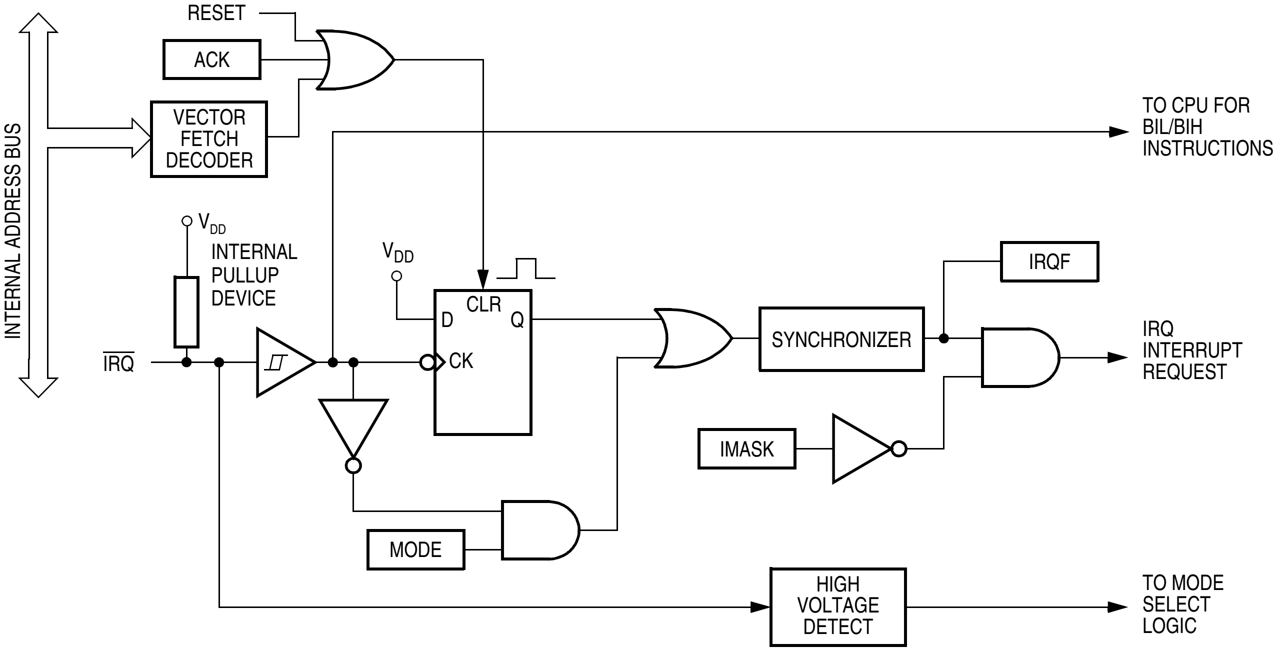


Figure 11-1. IRQ Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001E	IRQ Status and Control Register (INTSCR)	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:	▨	▨	▨	▨	▨	ACK		
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 11-2. IRQ I/O Register Summary

11.3.1 $\overline{\text{IRQ}}$ Pin

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

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

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

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

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

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

Use the BIH or BIL instruction to read the logic level on the $\overline{\text{IRQ}}$ pin.

NOTE

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

NOTE

An internal pull-up resistor to V_{DD} is connected to the $\overline{\text{IRQ}}$ pin; this can be disabled by setting the IRQPUD bit in the CONFIG2 register (\$001E).

11.4 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ latch can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear the latches during the break state. (See Chapter 4 System Integration Module (SIM).)

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

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

11.5 IRQ Status and Control Register (INTSCR)

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

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

Address: $001E

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

▨ = Unimplemented

Figure 11-3. IRQ Status and Control Register (INTSCR)

IRQF — IRQ Flag Bit

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

ACK — IRQ Interrupt Request Acknowledge Bit

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

IMASK — IRQ Interrupt Mask Bit

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

MODE — IRQ Edge/Level Select Bit

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

Chapter 12
Keyboard Interrupt Module (KBI)

12.1 Introduction

The keyboard interrupt module (KBI) provides four independently maskable external interrupts which are accessible via PTA0–PTA3. When a port pin is enabled for keyboard interrupt function, an internal pull-up device is also enabled on the pin.

12.2 Features

Features of the keyboard interrupt module include the following:

- Four keyboard interrupt pins with pull-up devices
- Separate keyboard interrupt enable bits and one keyboard interrupt mask
- Programmable edge-only or edge- and level- interrupt sensitivity
- Exit from low-power modes

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001B	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
$001C	Keyboard Interrupt Enable Register (KBIER)	Read:	0	PPI1IE2	PPI1IE1	PPI1IE0	KBIE3	KBIE2	KBIE1	KBIE0
		Write:	▨							
		Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 12-1. KBI I/O Register Summary

12.3 I/O Pins

The eight keyboard interrupt pins are shared with standard port I/O pins. The full name of the KBI pins are listed in Table 12-1. The generic pin name appear in the text that follows.

Table 12-1. Pin Name Conventions

KBI Generic Pin Name	Full MCU Pin Name	Pin Selected for KBI Function by KBIEx Bit in KBIER
KBI0–KBI3	PTA0/KBI0–PTA3/KBI3	KBIE0–KBIE3

12.4 Functional Description

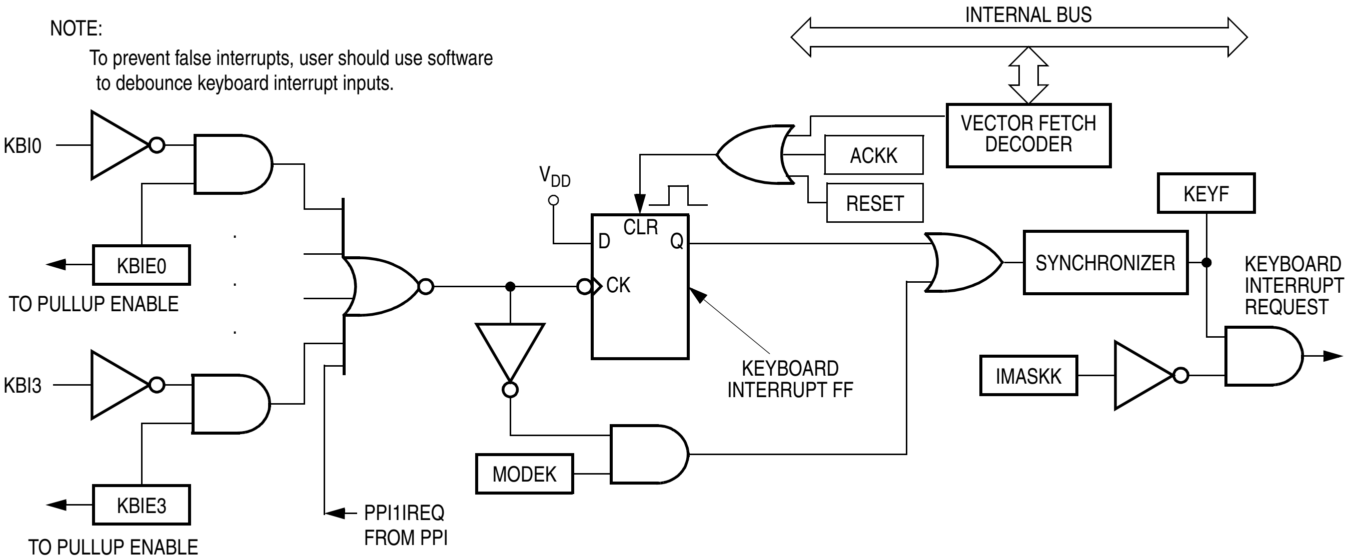


Figure 12-2. Keyboard Interrupt Block Diagram

Writing to the KBIE3–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port A pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin in port A also enables its internal pull-up device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

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

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

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

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register KBSCR. The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine 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 $FFE0 and $FFE1.
- 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 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, disable the pull-up device, use the data direction register to configure the pin as an input and then read the data register.

NOTE

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

12.4.1 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pull-up 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 DDRA bits in the data direction register A.
2. Write logic 1’s to the appropriate port A data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

12.5 Keyboard Interrupt Registers

Two registers control the operation of the keyboard interrupt module:

- Keyboard status and control register
- Keyboard interrupt enable register

12.5.1 Keyboard Status and Control Register

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

Address: $001B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	KEYF	0	IMASKK	MODEK
Write:	▨	▨	▨	▨	▨	ACKK		
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 12-3. Keyboard Status and Control Register (KBSCR)

KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending on port A. 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 on port A. 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 on port A. 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 on port A. Reset clears MODEK.

1 = Keyboard interrupt requests on falling edges and low levels
0 = Keyboard interrupt requests on falling edges only

12.5.2 Keyboard Interrupt Enable Register

The port-A keyboard interrupt enable register enables or disables each port-A pin to operate as a keyboard interrupt pin.

Address: $001C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	PPI1IE2	PPI1IE1	PPI1IE0	KBIE3	KBIE2	KBIE1	KBIE0
Write:	▨							
Reset:	0	0	0	0	0	0	0	0

Figure 12-4. Keyboard Interrupt Enable Register (KBIER)

PPI1IE[2:0] —PPI1 Interrupt Period Select Bits

(See Chapter 7 Programmable Periodic Interrupt (PPI).)

KBIE3–KBIE0 — Port-A Keyboard Interrupt Enable Bits

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

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

12.6 Low-Power Modes

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

12.6.1 Wait Mode

The keyboard modules remain 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.

12.6.2 Stop Mode

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

12.7 Keyboard Module During Break Interrupts

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

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

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

Chapter 13
Computer Operating Properly (COP)

13.1 Introduction

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

13.2 Functional Description

Figure 13-1 shows the structure of the COP module.

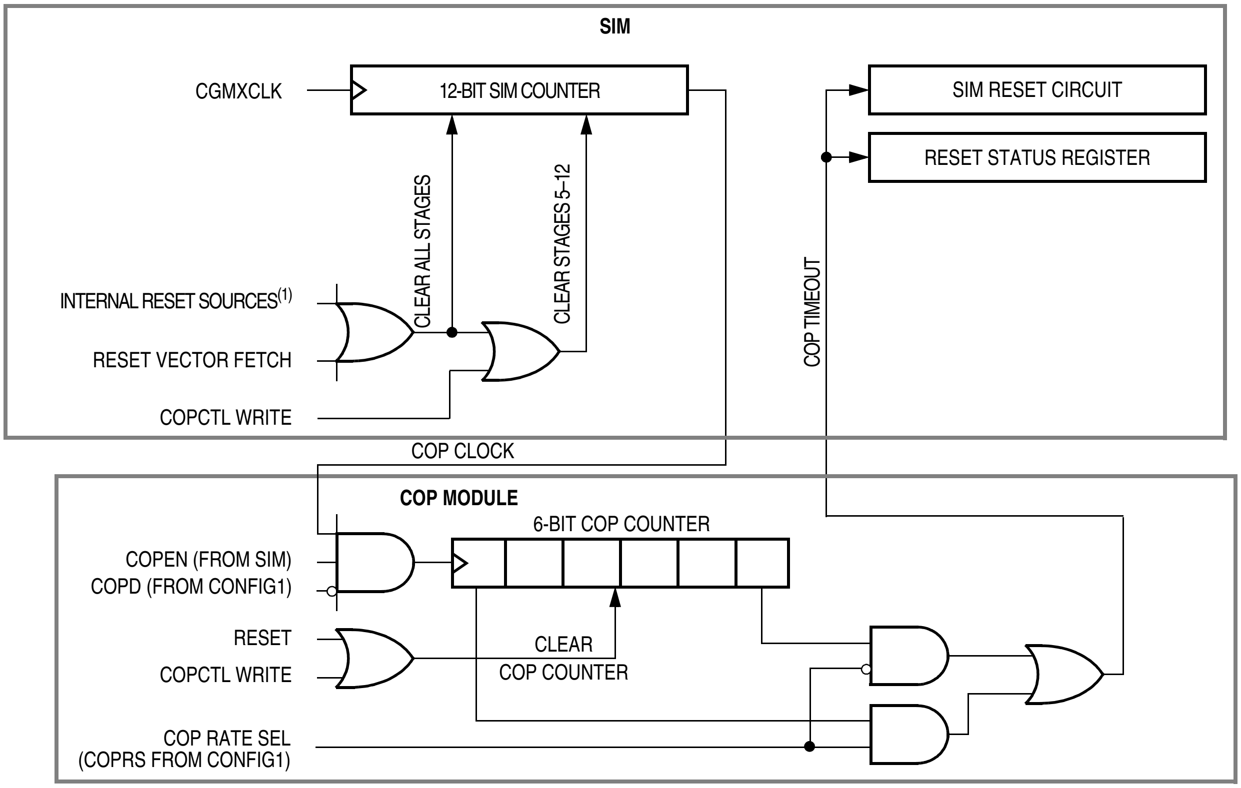


Figure 13-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by the 12-bit system integration module (SIM) counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after $2^{18} – 2^4$ or $2^{13} – 2^4$ CGMXCLK cycles; depending on the state of the COP rate select bit, COPRS, in CONFIG1 register. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 12 through 5 of the SIM counter.

NOTE

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

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 × CGMXCLK cycles and sets the COP bit in the reset status register (RSR). (See 4.7.2 Reset Status Register (RSR).).

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.

13.3 I/O Signals

The following paragraphs describe the signals shown in Figure 13-1.

13.3.1 CMGXCLK

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

13.3.2 COPCTL Write

Writing any value to the COP control register (COPCTL) (see 13.4 COP Control Register) clears the COP counter and clears bits 12 through 5 of the SIM counter. Reading the COP control register returns the low byte of the reset vector.

13.3.3 Power-On Reset

The power-on reset (POR) circuit in the SIM clears the SIM counter 4096 × CGMXCLK cycles after power-up.

13.3.4 Internal Reset

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

13.3.5 Reset Vector Fetch

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

13.3.6 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the CONFIG1 register. (See 3.3 Configuration Register 1 (CONFIG1).)

13.3.7 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register 1. (See 3.3 Configuration Register 1 (CONFIG1).)

13.4 COP Control Register

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

Address: $FFFF

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

Figure 13-2. COP Control Register (COPCTL)

13.5 Interrupts

The COP does not generate CPU interrupt requests.

13.6 Monitor Mode

When monitor mode is entered with V_{TST} on the $\overline{\text{IRQ}}$ pin, the COP is disabled as long as V_{TST} remains on the $\overline{\text{IRQ}}$ pin or the $\overline{\text{RST}}$ pin. When monitor mode is entered by having blank reset vectors and not having V_{TST} on the $\overline{\text{IRQ}}$ pin, the COP is automatically disabled until a POR occurs.

13.7 Low-Power Modes

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

13.7.1 Wait Mode

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

13.7.2 Stop Mode

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

To prevent inadvertently turning off the COP with a STOP instruction, a configuration option is available that disables the STOP instruction. When the STOP bit in the configuration register has the STOP instruction is disabled, execution of a STOP instruction results in an illegal opcode reset.

13.8 COP Module During Break Mode

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

Chapter 14
Low-Voltage Inhibit (LVI)

14.1 Introduction

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

14.2 Features

Features of the LVI module include:

- Programmable LVI interrupt and reset
- Selectable LVI trip voltage
- Programmable stop mode operation

14.3 Functional Description

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

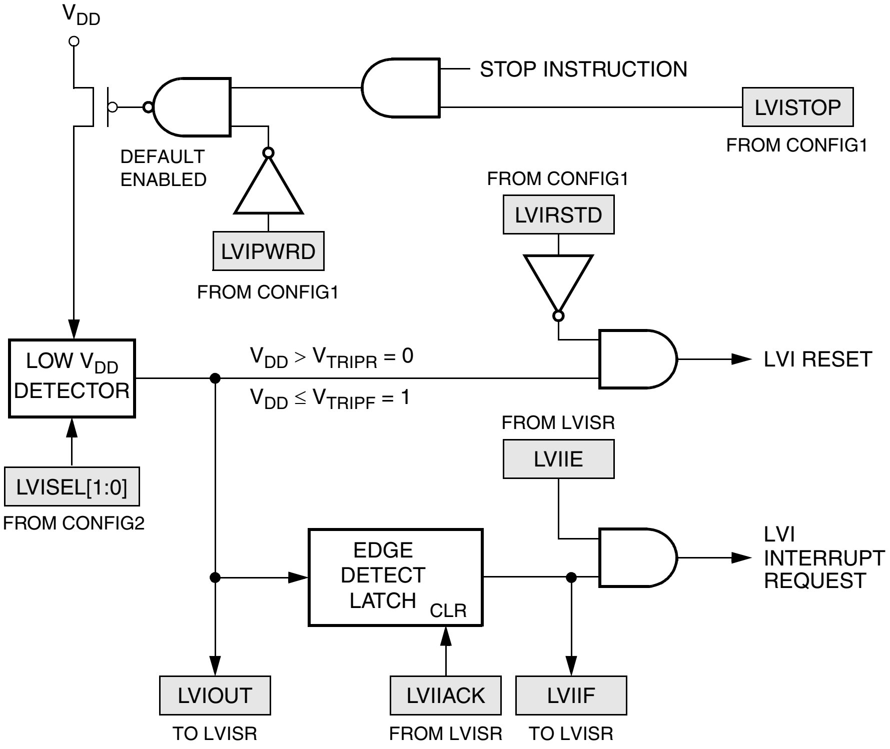


Figure 14-1. LVI Module Block Diagram

The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. Clearing the LVI power disable bit, LVIPWRD, enables the LVI to monitor V_{DD} voltage. Clearing the LVI reset disable bit, LVIRSTD, enables the LVI module to generate a reset when V_{DD} falls below a voltage, V_{TRIPF}. Setting the LVI enable in stop mode bit, LVISTOP, enables the LVI to operate in stop mode.

The LVI trip point selection bits, LVISEL[1:0], select the trip point voltage, V_{TRIPF}, to be configured for 5 V or 3 V operation. The actual trip points are shown in Chapter 17 Electrical Specifications.

Setting LVI interrupt enable bit, LVIIE, enables LVI interrupts whenever the LVIOUT bit toggles (from logic 0 to logic 1, or from logic 1 to logic 0).

NOTE

After a power-on reset (POR) the LVI's default mode of operation is 3 V. If a 5 V system is used, the user must modified the LVISEL[1:0] bits to raise the trip point to 5 V operation. Note that this must be done after every power-on reset since the default will revert back to 3 V mode after each power-on reset. If the V_{DD} *supply is below the 3V mode trip voltage when POR is released, the MCU will immediately go into reset. The LVI in this case will hold the MCU in reset until either* V_{DD} *goes above the rising 3 V trip point,* V_{TRIPR}*, which will release reset or* V_{DD} *decreases to approximately 0 V which will re-trigger the power-on reset.*

LVISTOP, LVIPWRD, LVIRSTD, and LVISEL[1:0] are in the configuration registers. See Section 5. Configuration Registers (CONFIG) for details of the LVI's configuration bits. Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above a voltage, V_{TRIPR}, which causes the MCU to exit reset. See 4.3.2.5 Low-Voltage Inhibit (LVI) Reset for details of the interaction between the SIM and the LVI. The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR). The LVIIE, LVIIF, and LVIIACK bits in the LVISR control LVI interrupt functions.

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

14.3.1 Polled LVI Operation

In applications that can operate at V_{DD} levels below the V_{TRIPF} level, software can monitor V_{DD} by polling the LVIOUT bit, or by setting the LVI interrupt enable bit, LVIIE, to enable interrupt requests. In the configuration register 1 (CONFIG1), the LVIPWRD bit must be at logic 0 to enable the LVI module, and the LVIRSTD bit must be at logic 1 to disable LVI resets.

The LVI interrupt flag, LVIIF, is set whenever the LVIOUT bit changes state (toggles). When LVIF is set, a CPU interrupt request is generated if the LVIIE is also set. In the LVI interrupt service subroutine, LVIIF bit can be cleared by writing a logic 1 to the LVI interrupt acknowledge bit, LVIIACK.

14.3.2 Forced Reset Operation

In applications that require V_{DD} to remain above the V_{TRIPF} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls below the V_{TRIPF} level. In the configuration register 1 (CONFIG1), the LVIPWRD and LVIRSTD bits must be at logic 0 to enable the LVI module and to enable LVI resets.

If LVIIE is set to enable LVI interrupts when LVIRSTD is cleared, LVI reset has a higher priority over LVI interrupt. In this case, when V_{DD} falls below the V_{TRIPF} level, an LVI reset will occur, and the LVIIE bit will be cleared.

14.3.3 Voltage Hysteresis Protection

Once the LVI has triggered (by having V_{DD} fall below V_{TRIPF}), the LVI will maintain a reset condition until V_{DD} rises above the rising trip point voltage, V_{TRIPR}. This prevents a condition in which the MCU is continually entering and exiting reset if V_{DD} is approximately equal to V_{TRIPF}. V_{TRIPR} is greater than V_{TRIPF} by the hysteresis voltage, V_{HYS}.

14.3.4 LVI Trip Selection

The trip point selection bits, LVISEL[1:0], in the CONFIG2 register select whether the LVI is configured for 5 V or 3 V operation. (See Chapter 3 Configuration Register (CONFIG).)

NOTE

The MCU is guaranteed to operate at a minimum supply voltage. The trip point (V_{TRIPF} [5 V] or V_{TRIPF} [3 V]) may be lower than this. (See Chapter 17 Electrical Specifications for the actual trip point voltages.)

14.4 LVI Status Register

The LVI status register (LVISR) controls LVI interrupt functions and indicates if the V_{DD} voltage was detected below the V_{TRIPF} level.

Address: $FE0F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	LVIOUT	LVIIE	LVIIF	0	0	0	0	0
Write:	▨		▨	LVIIACK	▨	▨	▨	▨
Reset:	0	0	0	0	0	0	0	0

▨ = Unimplemented

Figure 14-2. LVI Status Register (LVISR)

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} voltage falls below the V_{TRIPF} trip voltage (see Table 14-1). Reset clears the LVIOUT bit.

Table 14-1. LVIOUT Bit Indication

V_{DD}	LVIOUT
$V_{DD} > V_{TRIPR}$	0
$V_{DD} < V_{TRIPF}$	1
$V_{TRIPF} < V_{DD} < V_{TRIPR}$	Previous value

LVIIE — LVI Interrupt Enable Bit

This read/write bit enables the LVIIF bit to generate CPU interrupt requests. Reset clears the LVIIE bit.

1 = LVIIF can generate CPU interrupt requests
0 = LVIIF cannot generate CPU interrupt requests

LVIIF — LVI Interrupt Flag

This clearable, read-only flag is set whenever the LVIOUT bit toggles. Reset clears the LVIIF bit.

1 = LVIOUT has toggled
0 = LVIOUT has not toggled

LVIIACK — LVI Interrupt Acknowledge Bit

Writing a logic 1 to this write-only bit clears the LVI interrupt flag, LVIIF. LVIIACK always reads as logic 0.

1 = Clears LVIIF bit
0 = No effect

14.5 Low-Power Modes

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

14.5.1 Wait Mode

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

14.5.2 Stop Mode

If enabled in stop mode (LVISTOP = 1), the LVI module remains active in stop mode. If enabled to generate resets or interrupts, the LVI module can generate a reset or an interrupt and bring the MCU out of stop mode.

NOTE

If enabled to generate both resets and interrupts, there will be no LVI interrupts, as resets have a higher priority.

Chapter 15
Central Processor Unit (CPU)

15.1 Introduction

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

15.2 Features

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

15.3 CPU Registers

Figure 15-1 shows the five CPU registers. CPU registers are not part of the memory map.

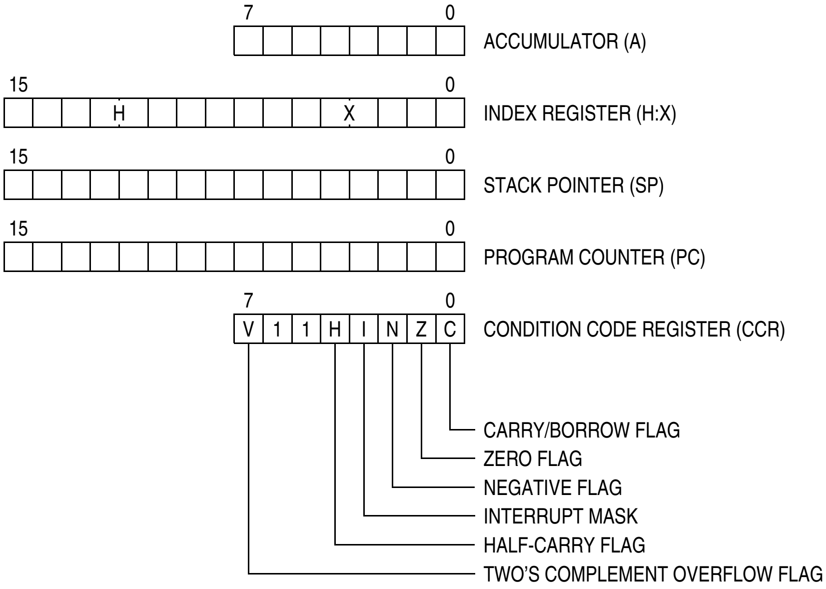


Figure 15-1. CPU Registers

15.3.1 Accumulator

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

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

Figure 15-2. Accumulator (A)

15.3.2 Index Register

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

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

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

	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 15-3. Index Register (H:X)

15.3.3 Stack Pointer

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

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

	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 15-4. Stack Pointer (SP)

NOTE

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

15.3.4 Program Counter

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

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

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

	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 15-5. Program Counter (PC)

15.3.5 Condition Code Register

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

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

X = Indeterminate

Figure 15-6. Condition Code Register (CCR)

V — Overflow Flag

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

1 = Overflow
0 = No overflow

H — Half-Carry Flag

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

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

I — Interrupt Mask

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

1 = Interrupts disabled
0 = Interrupts enabled

NOTE

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

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

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

N — Negative Flag

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

1 = Negative result
0 = Non-negative result

Z — Zero Flag

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

1 = Zero result
0 = Non-zero result

C — Carry/Borrow Flag

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

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

15.4 Arithmetic/Logic Unit (ALU)

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

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

15.5 Low-Power Modes

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

15.5.1 Wait Mode

The 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

15.5.2 Stop Mode

The STOP instruction:

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

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

15.6 CPU During Break Interrupts

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

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

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

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

15.7 Instruction Set Summary

Table 15-1 provides a summary of the M68HC08 instruction set.

Table 15-1. Instruction Set Summary (Sheet 1 of 6)

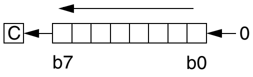
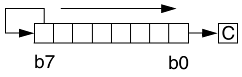
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9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

Table 15-1. Instruction Set Summary (Sheet 2 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{IRQ}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{IRQ}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{IRQ}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{IRQ}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr*	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM	A5	ii	2
BIT *opr*									DIR	B5	dd	3
BIT *opr*									EXT	C5	hh ll	4
BIT *opr*,X									IX2	D5	ee ff	4
BIT *opr*,X									IX1	E5	ff	3
BIT ,X									IX	F5		2
BIT *opr*,SP									SP1	9EE5	ff	4
BIT *opr*,SP									SP2	9ED5	ee ff	5
BLE *opr*	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) | (N ⊕ V) = 1	–	–	–	–	–	–	REL	93	rr	3
BLO *rel*	Branch if Lower (Same as BCS)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BLS *rel*	Branch if Lower or Same	PC ← (PC) + 2 + *rel* ? (C) | (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *opr*	Branch if Less Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n*,*opr*,*rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0)	01	dd rr	5
									DIR (b1)	03	dd rr	5
									DIR (b2)	05	dd rr	5
									DIR (b3)	07	dd rr	5
									DIR (b4)	09	dd rr	5
									DIR (b5)	0B	dd rr	5
									DIR (b6)	0D	dd rr	5
									DIR (b7)	0F	dd rr	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)	00	dd rr	5
									DIR (b1)	02	dd rr	5
									DIR (b2)	04	dd rr	5
									DIR (b3)	06	dd rr	5
									DIR (b4)	08	dd rr	5
									DIR (b5)	0A	dd rr	5
									DIR (b6)	0C	dd rr	5
									DIR (b7)	0E	dd rr	5
BSET *n*,*opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0)	10	dd	4
									DIR (b1)	12	dd	4
									DIR (b2)	14	dd	4
									DIR (b3)	16	dd	4
									DIR (b4)	18	dd	4
									DIR (b5)	1A	dd	4
									DIR (b6)	1C	dd	4
									DIR (b7)	1E	dd	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*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR	31	dd rr	5
CBEQA #*opr,rel*		PC ← (PC) + 3 + rel ? (A) – (M) = $00							IMM	41	ii rr	4
CBEQX #*opr,rel*		PC ← (PC) + 3 + rel ? (X) – (M) = $00							IMM	51	ii rr	4
CBEQ *opr*,X+,*rel*		PC ← (PC) + 3 + rel ? (A) – (M) = $00							IX1+	61	ff rr	5
CBEQ X+,*rel*		PC ← (PC) + 2 + rel ? (A) – (M) = $00							IX+	71	rr	4
CBEQ *opr*,SP,*rel*		PC ← (PC) + 4 + rel ? (A) – (M) = $00							SP1	9E61	ff rr	6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2

Table 15-1. Instruction Set Summary (Sheet 3 of 6)

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

Table 15-1. Instruction Set Summary (Sheet 4 of 6)

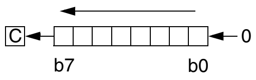
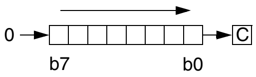
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR ,X	Jump to Subroutine	PC ← (PC) + *n* (*n* = 1, 2, or 3) Push (PCL); SP ← (SP) – 1 Push (PCH); SP ← (SP) – 1 PC ← Unconditional Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BD CD DD ED FD	dd hh ll ee ff ff	4 5 6 5 4
LDA #*opr* LDA *opr* LDA *opr* LDA *opr*,X LDA *opr*,X LDA ,X LDA *opr*,SP LDA *opr*,SP	Load A from M	A ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A6 B6 C6 D6 E6 F6 9EE6 9ED6	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LDHX #*opr* LDHX *opr*	Load H:X from M	H:X ← (M:M + 1)	0	–	–	↕	↕	–	IMM DIR	45 55	ii jj dd	3 4
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X LDX *opr*,SP LDX *opr*,SP	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
LSL *opr* LSLA LSLX LSL *opr*,X LSL ,X LSL *opr*,SP	Logical Shift Left (Same as ASL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
LSR *opr* LSRA LSR*X* LSR *opr*,X LSR ,X LSR *opr*,SP	Logical Shift Right		↕	–	–	0	↕	↕	DIR INH INH IX1 IX SP1	34 44 54 64 74 9E64	dd ff ff	4 1 1 4 3 5
MOV *opr,opr* MOV *opr*,X+ MOV #*opr,opr* MOV X+,*opr*	Move	$(M)_{Destination} \leftarrow (M)_{Source}$ H:X ← (H:X) + 1 (IX+D, DIX+)	0	–	–	↕	↕	–	DD DIX+ IMD IX+D	4E 5E 6E 7E	dd dd dd ii dd dd	5 4 4 4
MUL	Unsigned multiply	X:A ← (X) × (A)	–	0	–	–	–	0	INH	42		5
NEG *opr* NEGA NEGX NEG *opr*,X NEG ,X NEG *opr*,SP	Negate (Two's Complement)	M ← –(M) = $00 – (M) A ← –(A) = $00 – (A) X ← –(X) = $00 – (X) M ← –(M) = $00 – (M) M ← –(M) = $00 – (M)	↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60 70 9E60	dd ff ff	4 1 1 4 3 5
NOP	No Operation	None	–	–	–	–	–	–	INH	9D		1
NSA	Nibble Swap A	A ← (A[3:0]:A[7:4])	–	–	–	–	–	–	INH	62		3
ORA #*opr* ORA *opr* ORA *opr* ORA *opr*,X ORA *opr*,X ORA ,X ORA *opr*,SP ORA *opr*,SP	Inclusive OR A and M	A ← (A) | (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AA BA CA DA EA FA 9EEA 9EDA	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
PSHA	Push A onto Stack	Push (A); SP ← (SP) – 1	–	–	–	–	–	–	INH	87		2
PSHH	Push H onto Stack	Push (H); SP ← (SP) – 1	–	–	–	–	–	–	INH	8B		2
PSHX	Push X onto Stack	Push (X); SP ← (SP) – 1	–	–	–	–	–	–	INH	89		2

Table 15-1. Instruction Set Summary (Sheet 5 of 6)

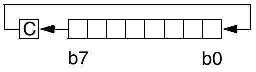
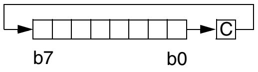
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X ROL *opr*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	39 49 59 69 79 9E69	dd ff ff	4 1 1 4 3 5
ROR *opr* RORA RORX ROR *opr*,X ROR ,X ROR *opr*,SP	Rotate Right through Carry		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	36 46 56 66 76 9E66	dd ff ff	4 1 1 4 3 5
RSP	Reset Stack Pointer	SP ← $FF	–	–	–	–	–	–	INH	9C		1
RTI	Return from Interrupt	SP ← (SP) + 1; Pull (CCR) SP ← (SP) + 1; Pull (A) SP ← (SP) + 1; Pull (X) SP ← (SP) + 1; Pull (PCH) SP ← (SP) + 1; Pull (PCL)	↕	↕	↕	↕	↕	↕	INH	80		7
RTS	Return from Subroutine	SP ← SP + 1; Pull (PCH) SP ← SP + 1; Pull (PCL)	–	–	–	–	–	–	INH	81		4
SBC #*opr* SBC *opr* SBC *opr* SBC *opr*,X SBC *opr*,X SBC ,X SBC *opr*,SP SBC *opr*,SP	Subtract with Carry	A ← (A) – (M) – (C)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A2 B2 C2 D2 E2 F2 9EE2 9ED2	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SEC	Set Carry Bit	C ← 1	–	–	–	–	–	1	INH	99		1
SEI	Set Interrupt Mask	I ← 1	–	–	1	–	–	–	INH	9B		2
STA *opr* STA *opr* STA *opr*,X STA *opr*,X STA ,X STA *opr*,SP STA *opr*,SP	Store A in M	M ← (A)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	B7 C7 D7 E7 F7 9EE7 9ED7	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
STHX *opr*	Store H:X in M	(M:M + 1) ← (H:X)	0	–	–	↕	↕	–	DIR	35	dd	4
STOP	Enable Interrupts, Stop Processing, Refer to MCU Documentation	I ← 0; Stop Processing	–	–	0	–	–	–	INH	8E		1
STX *opr* STX *opr* STX *opr*,X STX *opr*,X STX ,X STX *opr*,SP STX *opr*,SP	Store X in M	M ← (X)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X SUB *opr*,SP SUB *opr*,SP	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5

Table 15-1. Instruction Set Summary (Sheet 6 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		9
TAP	Transfer A to CCR	CCR ← (A)	↕	↕	↕	↕	↕	↕	INH	84		2
TAX	Transfer A to X	X ← (A)	–	–	–	–	–	–	INH	97		1
TPA	Transfer CCR to A	A ← (CCR)	–	–	–	–	–	–	INH	85		1
TST *opr* TSTA TSTX TST *opr*,X TST ,X TST *opr*,SP	Test for Negative or Zero	(A) – $00 or (X) – $00 or (M) – $00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D 6D 7D 9E6D	dd ff ff	3 1 1 3 2 4
TSX	Transfer SP to H:X	H:X ← (SP) + 1	–	–	–	–	–	–	INH	95		2
TXA	Transfer X to A	A ← (X)	–	–	–	–	–	–	INH	9F		1
TXS	Transfer H:X to SP	(SP) ← (H:X) – 1	–	–	–	–	–	–	INH	94		2
WAIT	Enable Interrupts; Wait for Interrupt	I bit ← 0; Inhibit CPU clocking until interrupted	–	–	0	–	–	–	INH	8F		1

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

15.8 Opcode Map

See Table 15-2.

Table 15-2. Opcode Map

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

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

*Pre-byte for stack pointer indexed instructions

	MSB / LSB	0	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

Chapter 16
Development Support

16.1 Introduction

This section describes the break module, the monitor module (MON), and the monitor mode entry methods.

16.2 Break Module (BRK)

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

Features include:

- Accessible input/output (I/O) registers during the break Interrupt
- Central processor unit (CPU) generated break interrupts
- Software-generated break interrupts
- Computer operating properly (COP) disabling during break interrupts

16.2.1 Functional Description

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

The following events can cause a break interrupt to occur:

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

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

Figure 16-1 shows the structure of the break module.

When the internal address bus matches the value written in the break address registers or when software writes a 1 to the BRKA bit in the break status and control register, the CPU starts a break interrupt by:

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

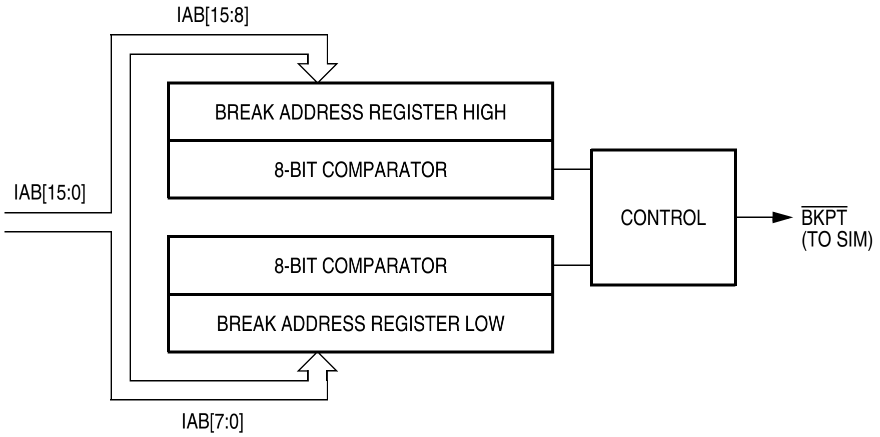


Figure 16-1. Break Module Block Diagram

The break interrupt timing is:

- When a break address is placed at the address of the instruction opcode, the instruction is not executed until after completion of the break interrupt routine.
- When a break address is placed at an address of an instruction operand, the instruction is executed before the break interrupt.
- When software writes a 1 to the BRKA bit, the break interrupt occurs just before the next instruction is executed.

By updating a break address and clearing the BRKA bit in a break interrupt routine, a break interrupt can be generated continuously.

CAUTION

A break address should be placed at the address of the instruction opcode. When software does not change the break address and clears the BRKA bit in the first break interrupt routine, the next break interrupt will not be generated after exiting the interrupt routine even when the internal address bus matches the value written in the break address registers.

16.2.1.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See 4.7.3 SIM Break Flag Control Register and the “Break Interrupts” subsection for each module.)

16.2.1.2 TIM During Break Interrupts

A break interrupt stops the timer counter.

16.2.1.3 COP During Break Interrupts

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

16.2.2 Break Module Registers

These registers control and monitor operation of the break module:

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

16.2.2.1 Break Status and Control Register (BRKSCR)

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

Address: $FE0E

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

▨ = Unimplemented

Figure 16-2. Break Status and Control Register (BRKSCR)

BRKE — Break Enable Bit

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

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

BRKA — Break Active Bit

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

1 = Break address match
0 = No break address match

16.2.2.2 Break Address Registers

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

Address: $FE0C

	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

Figure 16-3. Break Address Register High (BRKH)

Address: $FE0D

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

Figure 16-4. Break Address Register Low (BRKL)

16.2.2.3 Break Status Register

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

Address: $FE00

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	R	R	R	R	R	SBSW	R
Write:							Note[1]	
Reset:							0	

R = Reserved

1. Writing a logic zero clears SBSW.

Figure 16-5. Break Status Register (BSR)

SBSW — SIM Break Stop/Wait

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

1 = Stop mode or wait mode was exited by break interrupt

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

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it.

16.2.2.4 Break Flag Control Register (BFCR)

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

Address: $FE03

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

R = Reserved

Figure 16-6. 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

16.2.3 Low-Power Modes

The WAIT and STOP instructions put the MCU in low power-consumption standby modes. If enabled, the break module will remain enabled in wait and stop modes. However, since the internal address bus does not increment in these modes, a break interrupt will never be triggered.

16.3 Monitor Module (MON)

The monitor module allows debugging and programming of the microcontroller unit (MCU) through a single-wire interface with a host computer. Monitor mode entry can be achieved without use of the higher test voltage, V_{TST}, as long as vector addresses $FFFE and $FFFF are blank, thus reducing the hardware requirements for in-circuit programming.

Features of the monitor module include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between MCU and host computer
- Standard non-return-to-zero (NRZ) communication with host computer
- Standard communication baud rate (9600 @ 2.4576-MHz internal operating frequency)
- Execution of code in random-access memory (RAM) or FLASH
- FLASH memory security feature(1)
- FLASH memory programming interface
- Use of external 32.768kHz, 4.9152MHz or 9.8304MHz oscillator to generate internal operating frequency of 2.4576 MHz
- Monitor mode entry without high voltage, V_{TST}, if reset vector is blank ($FFFE and $FFFF contain $FF)
- Normal monitor mode entry if V_{TST} is applied to $\overline{\text{IRQ}}$

16.3.1 Functional Description

Figure 16-7 shows a simplified diagram of monitor mode entry.

The monitor module receives and executes commands from a host computer. Figure 16-8 and Figure 16-9 show example circuits used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute code downloaded into RAM by a host computer while most MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

Table 16-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a power-on-reset (POR) and will allow communication at 9600 baud provided one of the following sets of conditions is met.

- If $FFFE and $FFFF are erased or programmed:
  - The external clock is 4.9152 MHz
  - PTC3 = low
  - $\overline{\text{IRQ}} = V_{TST}$
- If $FFFE and $FFFF are erased or programmed:
  - The external clock is 9.8304 MHz
  - PTC3 = high
  - $\overline{\text{IRQ}} = V_{TST}$
- If $FFFE and $FFFF contain $FF (erased state):

---

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

- The external clock is 32.768kHz (PLL turns on automatically to generate an internal operating frequency of 2.4576 MHz)
- $\overline{IRQ} = V_{SS}$

The last two conditions are the forced monitor mode.

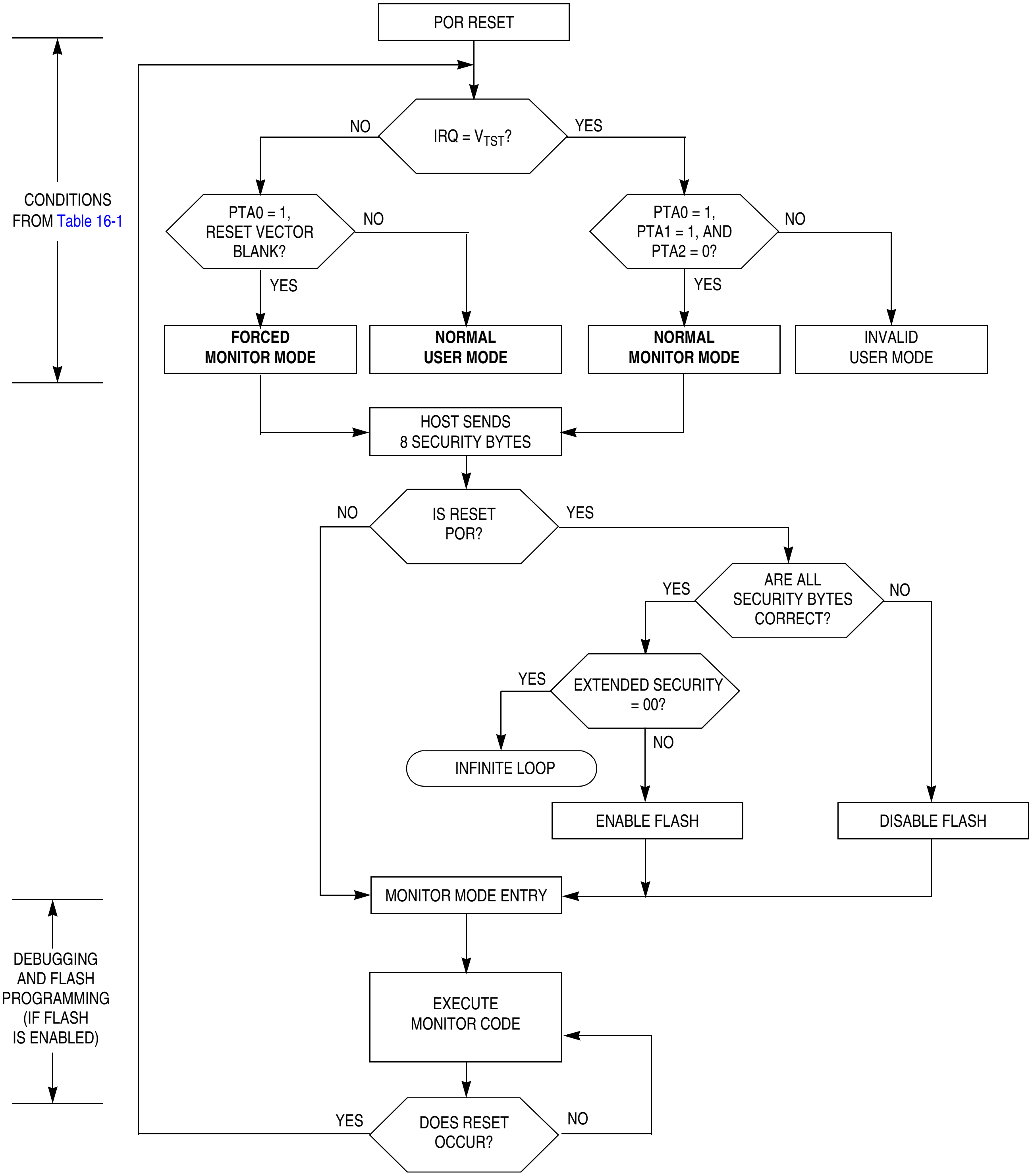


Figure 16-7. Simplified Monitor Mode Entry Flowchart

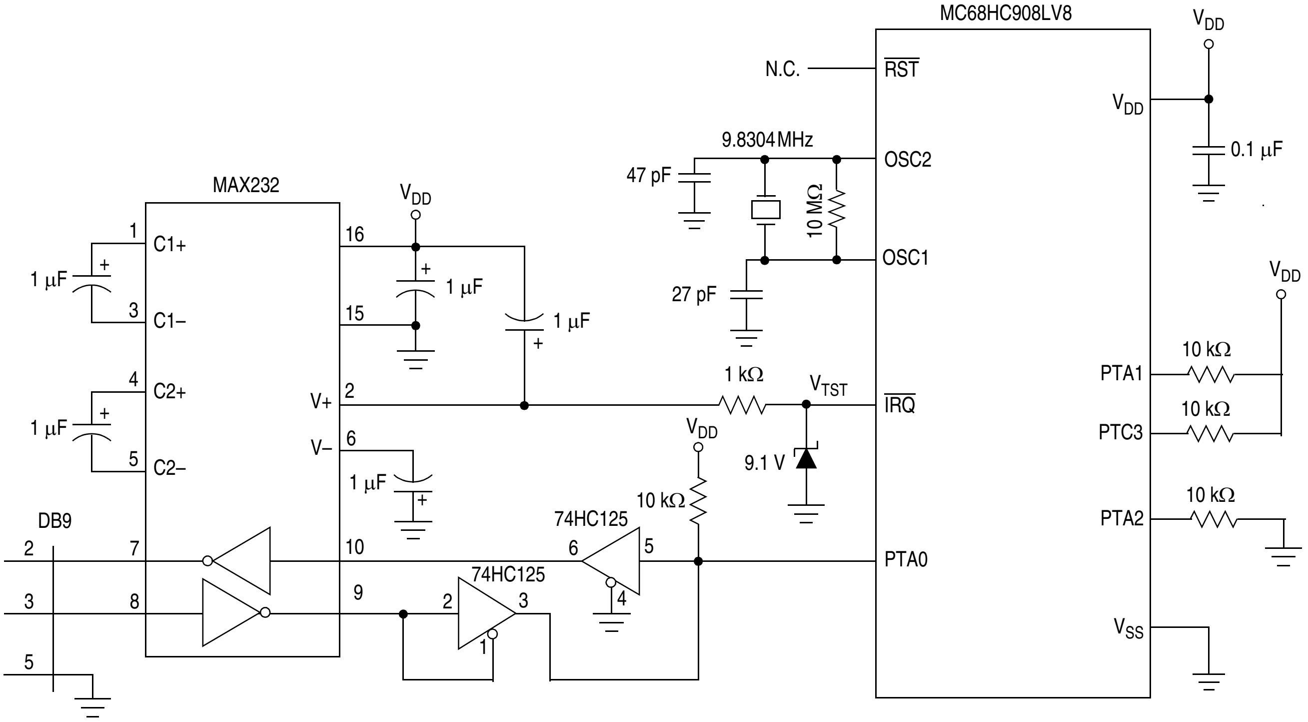


Figure 16-8. Normal Monitor Mode Circuit

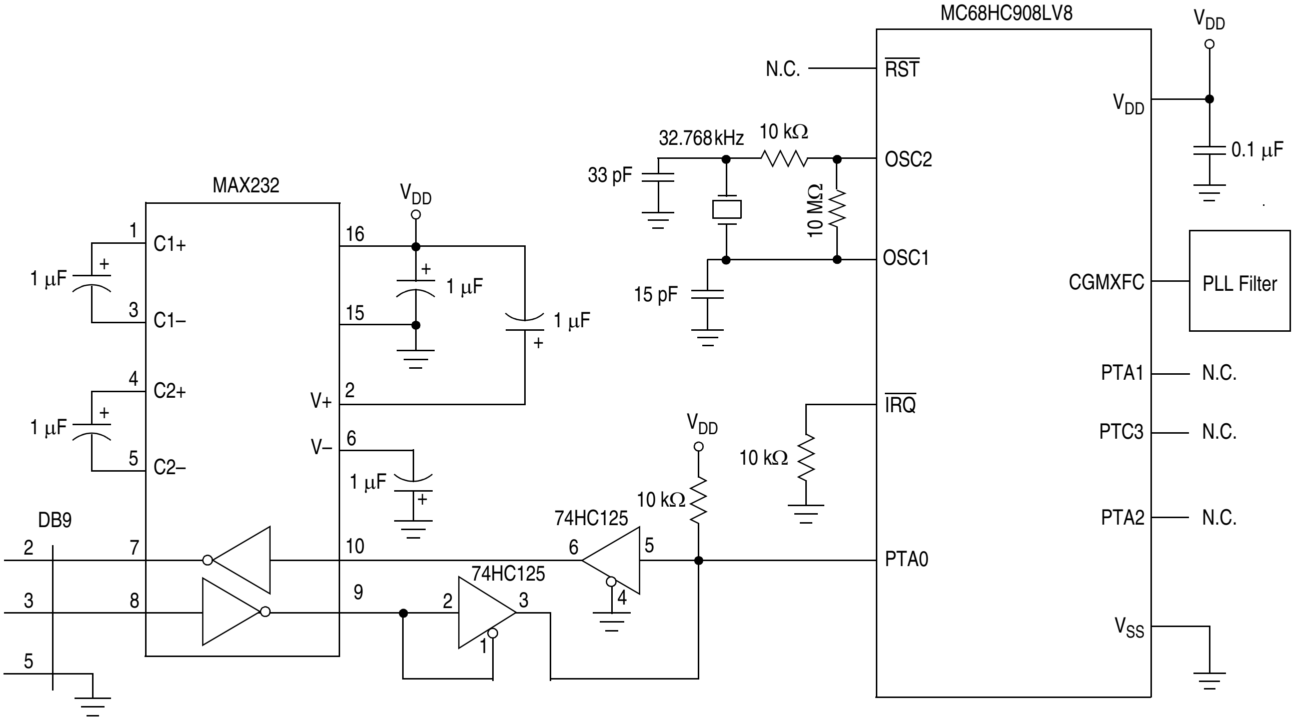


Figure 16-9. Forced Monitor Circuit ($V_{\overline{IRQ}} = V_{SS}$)

Table 16-1. Monitor Mode Signal Requirements and Options

Mode	$\overline{\text{IRQ}}$	$\overline{\text{RST}}$	Reset Vector	Serial Communication	Mode Selection		Divider	PLL	COP	Communication Speed		
				PTA0	PTA1	PTA2	PTC3			External Clock	f_{op}	Baud Rate
Normal Monitor	V_{TST}	V_{DD} or V_{TST}	X	1	1	0	0	OFF	Disabled	4.9152 MHz	2.4576 MHz	9600
			X	1	1	0	1	OFF	Disabled	9.8304 MHz	2.4576 MHz	9600
Forced Monitor	V_{SS}	V_{DD}	\$FFFF (blank)	1	X	X	X	ON	Disabled	32.768 kHz Crystal	2.4576 MHz	9600
User	V_{DD} or V_{SS}	V_{DD} or V_{TST}	Not \$FFFF	X	X	X	X	X	Enabled	X	—	—
MON08 Function [Pin No.]	V_{TST} [6]	RST [5]	—	COM [8]	MOD0 [12]	MOD1 [14]	DIV4 [16]	—	—	OSC1 [13]	—	—

1. PTA0 must have a pullup resistor to V_{DD} in monitor mode.
2. Communication speed in the table is an example to obtain a baud rate of 9600. Baud rate using external oscillator is internal operating frequency / 256.
3. External clock is a 32.768kHz, 4.9152MHz or 9.8304 MHz crystal on OSC1 and OSC2 or a 32.768kHz, 4.9152MHz or 9.8304 MHz canned oscillator on OSC1.
4. X = don't care
5. $\overline{\text{RST}}$ column indicates the state of $\overline{\text{RST}}$ after the monitor entry.
6. MON08 pin refers to P&E Microcomputer Systems' MON08-Cyclone 2 by 8-pin connector.

NC	1	2	GND
NC	3	4	$\overline{\text{RST}}$
NC	5	6	$\overline{\text{IRQ}}$
NC	7	8	PTA0
NC	9	10	NC
NC	11	12	PTA1
OSC1	13	14	PTA2
V_{DD}	15	16	PTC3

Enter monitor mode with pin configuration shown in Table 16-1 by pulling $\overline{\text{RST}}$ low and then high. The rising edge of the $\overline{\text{RST}}$ latches monitor mode. Once monitor mode is latched, the levels on the port pins except PTA0 can change.

Once out of reset, the MCU waits for the host to send eight security bytes (see 16.3.2 Security). After the security bytes, the MCU sends a break signal (10 consecutive 0s) to the host, indicating that it is ready to receive a command.

16.3.1.1 Normal Monitor Mode

If V_{TST} is applied to $\overline{\text{IRQ}}$ and PTC3 is low upon monitor mode entry, the internal operating frequency is a divide-by-two of the input clock. If PTC3 is high with V_{TST} applied to $\overline{\text{IRQ}}$ upon monitor mode entry, the internal operating frequency will be a divide-by-four of the input clock. Holding the PTC3 pin low when entering monitor mode causes a by pass of a divide-by-two stage at the oscillator only if V_{TST} is applied to $\overline{\text{IRQ}}$. In this event, the CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum internal operating frequency.

When monitor mode was entered with V_{TST} on $\overline{\text{IRQ}}$, the computer operating properly (COP) is disabled as long as V_{TST} is applied to either $\overline{\text{IRQ}}$ or $\overline{\text{RST}}$. This condition states that as long as V_{TST} is maintained on the $\overline{\text{IRQ}}$ pin after entering monitor mode, or if V_{TST} is applied to $\overline{\text{RST}}$ after the initial reset to get into monitor mode (when V_{TST} was applied to $\overline{\text{IRQ}}$), then the COP will be disabled. In the latter situation, after V_{TST} is applied to the $\overline{\text{RST}}$ pin, V_{TST} can be removed from the $\overline{\text{IRQ}}$ pin in the interest of freeing the $\overline{\text{IRQ}}$ for normal functionality in monitor mode.

16.3.1.2 Forced Monitor Mode

If entering monitor mode without high voltage on $\overline{\text{IRQ}}$, then startup port pin requirements and conditions, (PTA1/PTA2/PTC3) are not in effect. This is to reduce circuit requirements when performing in-circuit programming.

If the reset vector is blank and monitor mode is entered without V_{TST} on $\overline{\text{IRQ}}$, the MCU will see an additional reset cycle after the initial power-on reset (POR). The MCU will initially come out of reset in user mode. Internal circuitry monitors the reset vector fetches and will assert an internal reset if it detects the reset vector is erased ($FFFF).

Once the MCU enters this mode any reset other than a POR will automatically force the MCU to come back to the forced monitor mode. Exiting the forced monitor mode requires a POR. Pulling $\overline{\text{RST}}$ low will not exit monitor mode in this situation. Once the reset vector has been programmed, the traditional method of applying a voltage, V_{TST}, to $\overline{\text{IRQ}}$ must be used to re-enter monitor mode after the next POR.

When the forced monitor mode is entered the COP is always disabled regardless of the state of $\overline{\text{IRQ}}$ or $\overline{\text{RST}}$.

With V_{SS} on $\overline{\text{IRQ}}$ at the monitor entry, the PLL turns on and an internal operating frequency is generated with a 32.768 kHz crystal.

16.3.1.3 Monitor Vectors

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

Table 16-2 summarizes the differences between user mode and monitor mode regarding vectors.

Table 16-2. Mode Difference

Modes	Functions					
	Reset Vector High	Reset Vector Low	Break Vector High	Break Vector Low	SWI Vector High	SWI Vector Low
User	$FFFE	$FFFF	$FFFC	$FFFD	$FFFC	$FFFD
Monitor	$FEFE	$FEFF	$FEFC	$FEFD	$FEFC	$FEFD

16.3.1.4 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.

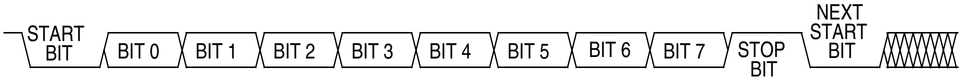


Figure 16-10. Monitor Data Format

16.3.1.5 Break Signal

A start bit (logic 0) followed by nine logic 0 bits is a break signal. When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits and then echoes back the break signal.

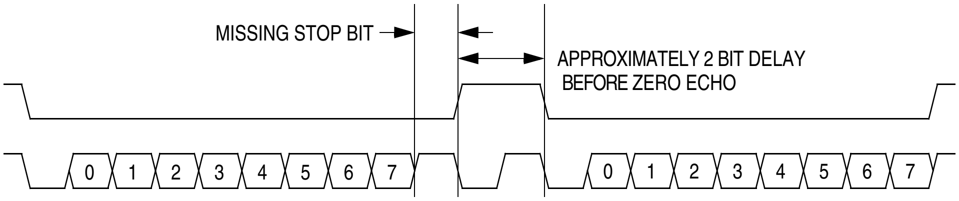


Figure 16-11. Break Transaction

16.3.1.6 Baud Rate

The monitor communication baud rate is controlled by the frequency of the external oscillator and the state of the appropriate pins as shown in Table 16-1.

Table 16-1 also lists the internal operating frequencies to achieve standard baud rates. The effective baud rate is the internal operating frequency divided by 256 when using an external oscillator. If using a crystal as the clock source, be aware of the upper frequency limit that the internal clock module can handle. See 17.7 5-V Control Timing for this limit.

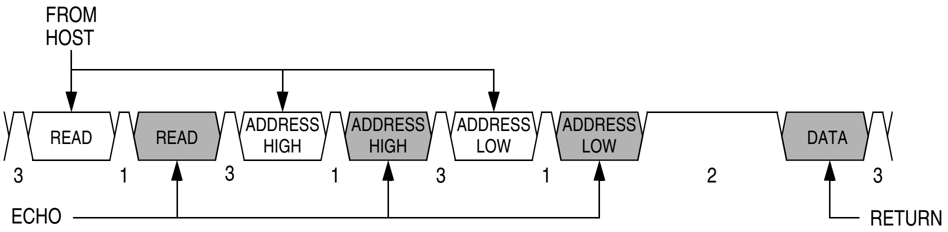
16.3.1.7 Commands

- The monitor ROM firmware uses these commands:
- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. A delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

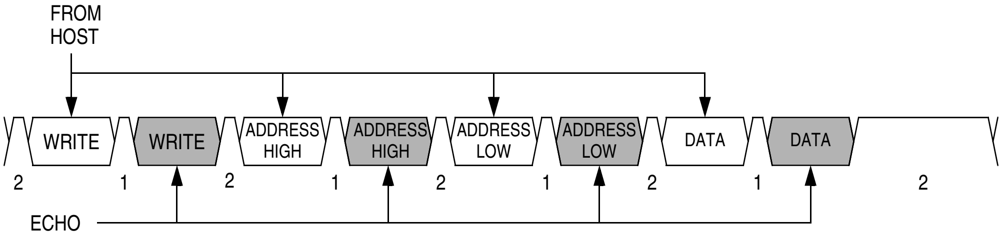
NOTE

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



Notes:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
3 = Wait approximately 1 bit time before sending next byte

Figure 16-12. Read Transaction



Notes:
1 = Echo delay, approximately 2 bit times
2 = Wait approximately 1 bit time before sending next byte

Figure 16-13. Write Transaction

A brief description of each monitor mode command is given in Table 16-3 through Table 16-8.

Table 16-3. READ (Read Memory) Command

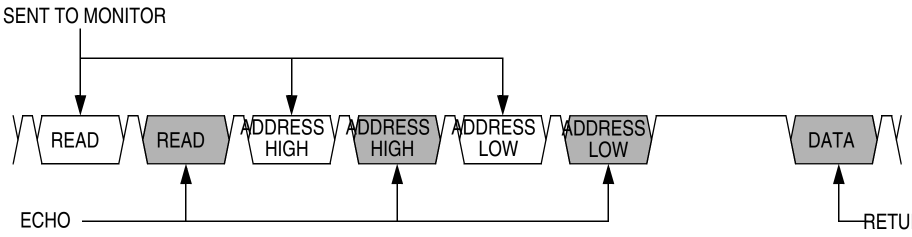
Description	Read byte from memory
Operand	2-byte address in high-byte:low-byte order
Data Returned	Returns contents of specified address
Opcode	$4A
Command Sequence	

Table 16-4. WRITE (Write Memory) Command

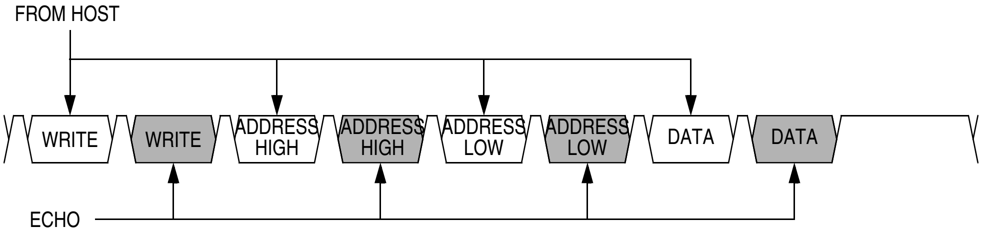
Description	Write byte to memory
Operand	2-byte address in high-byte:low-byte order; low byte followed by data byte
Data Returned	None
Opcode	$49
Command Sequence	

Table 16-5. IREAD (Indexed Read) Command

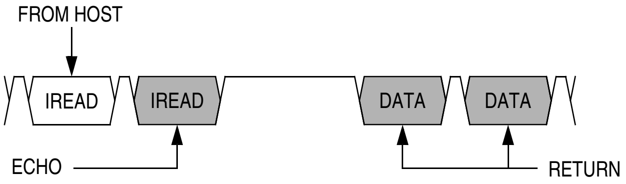
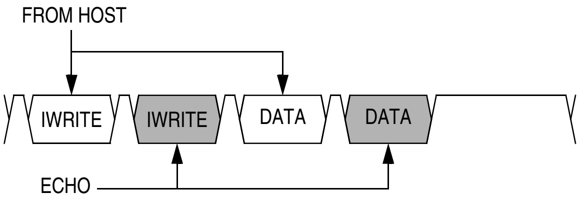
Description	Read next 2 bytes in memory from last address accessed
Operand	None
Data Returned	Returns contents of next two addresses
Opcode	$1A
Command Sequence	

Table 16-6. 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 16-7. 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

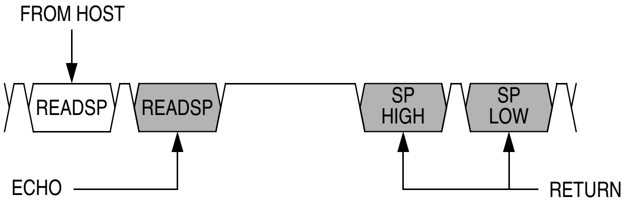
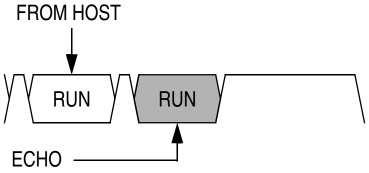


Table 16-8. RUN (Run User Program) Command

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

Command Sequence



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

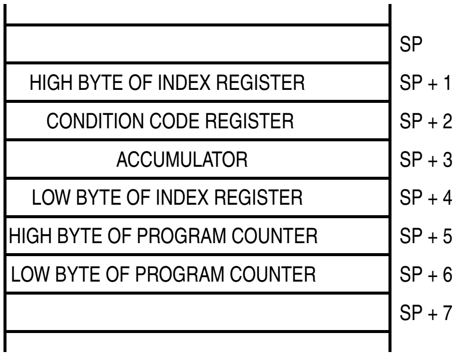


Figure 16-14. Stack Pointer at Monitor Mode Entry

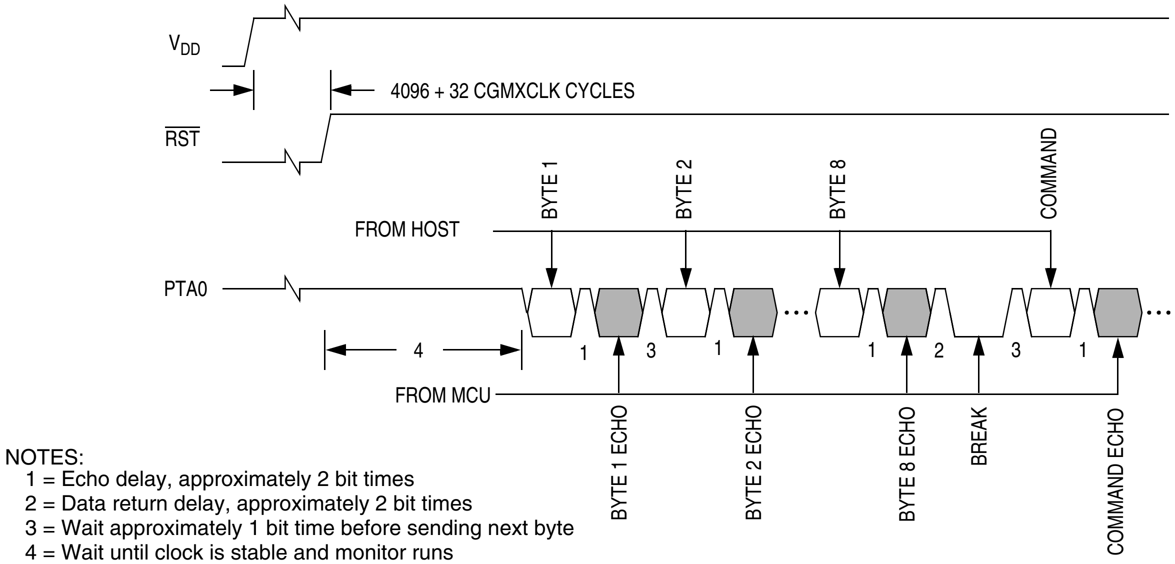
16.3.2 Security

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

NOTE

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

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations \$FFF6–\$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. See Figure 16-15.



NOTES:
1 = Echo delay, approximately 2 bit times
2 = Data return delay, approximately 2 bit times
3 = Wait approximately 1 bit time before sending next byte
4 = Wait until clock is stable and monitor runs

Figure 16-15. Monitor Mode Entry Timing

Upon power-on reset, if the received bytes of the security code do not match the data at locations \$FFF6–\$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

NOTE

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

To determine whether the security code entered is correct, check to see if bit 6 of RAM address \$80 is set. If it is, then the correct security code has been entered and FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become \$FF (blank).

16.3.3 Extended Security

In addition to the above security, a more secure feature called extended security is implemented in the MCU to further protect FLASH contents. Once this extended security is enabled, the MCU does not allow any user to enter the monitor mode even when all 8 security bytes are matched correctly. The extended security feature can be enabled by programming address $FDFF located in the user FLASH memory with data $00.

To unlock the extended security feature, the MCU must enter the monitor mode by failing the 8 byte security check. Then the FLASH must be mass-erased. This unlock process will erase the FLASH contents completely.

NOTE

To avoid enabling the extended security unintentionally, the user must make sure that the user software does not contain data $00 at address $FDFF.

16.4 Routines Supported in ROM

In the ROM, five routines are supported. Because the ROM has a jump table, the user does not call the routines with direct addresses. Therefore, the calling addresses will not change—even when the ROM code is updated in the future.

This section introduces each routine briefly. Details are discussed in later sections.

- **GetByte** — This routine is used to receive a byte serially on the general-purpose I/O PTA0. The receiving baud rate is the same as the baud rate used in monitor mode.
- **PutByte** — This routine is used to send a byte serially on the general-purpose I/O PTA0. The sending baud rate is the same as the baud rate specified in monitor mode.
- **Copy2RAM** — This routine is used to copy data in a contiguous range of FLASH locations to the DATA array assigned in RAM. The user can choose the DATA array locations and the variable locations required for this routine in the user software.
- **rErase** — This routine is used to erase either a page (64 bytes) or the whole array of FLASH. The user can choose the variable locations required for this routine in the user software. It can be used when the internal operating frequency (f_{op}) is between 1.5 MHz and 8.0 MHz.
- **rProgram** — This routine is used to program a contiguous range of FLASH locations. Programming data is first loaded into the DATA array assigned in RAM. The user can choose the DATA array locations and the variable locations required for this routine in the user software. rProgram can be used when the internal operating frequency (f_{op}) is between 1.5 MHz and 8.0 MHz.

16.4.1 Variables Used in the Routines

The Copy2RAM, rErase, and rProgram routines require certain registers and/or RAM locations to be initialized before calling the routines in the user software. Table 16-9 shows variables used in the routines and their locations.

Table 16-9. Variables and Their Locations

Location	Variable Name	Size (Bytes)	Description
H:X	CPUSPD	1	CPUSPD — the nearest integer of f_{op} (in MHz) × 4; for example, if f_{op} = 2.4576 MHz, CPUSPD = 10
H:X + 1	DATASIZE	1	Byte number to be programmed to FLASH or copied to RAM
H:X + 2 : H:X + 3	ADDR	2	Start address of a 16-bit range
H:X + 4	DATA	Varies	First location of DATA array; DATA array size must match a programming or verifying range

- Registers H:X — Register H:X are initialized with a 16 bit value representing the start address of a RAM block which contains variables CPUSPD, DATASIZE, ADDR and DATA array. The variables are used for Copy2RAM, rProgram and rErase routines. The RAM block start address must be set by the user software.
- CPUSPD — To set up proper delays used in the rProgram and rErase routines, a value indicating the internal operating frequency (f_{op}) must be stored at CPUSPD, which is an address specified by H:X registers. The CPUSPD value is the nearest integer of f_{op} (in MHz) times 4. For example, if f_{op} is 4.2 MHz, the CPUSPD value is 17 ($11). If f_{op} is 2.1 MHz, the CPUSPD value is 8. Setting a correct CPUSPD value is very important to program or erase the FLASH successfully.
- DATASIZE — DATASIZE is used in the Copy2RAM and rProgram routines. It is initialized with an 8 bit value representing the number of bytes to be programmed in FLASH or copied to RAM. The location is specified in (H:X+1).
- ADDR — The 16-bit value in RAM addresses specified in (H:X+2) and (H:X+3) holds the start address of the range in FLASH to be copied or programmed. The addresses (H:X+2) and (H:X+3) are the high and low bytes of the start address, respectively. In Copy2RAM and rProgram routines, ADDR is initialized with a 16 bit value representing the first address of the range. In rErase routine, ADDR is initialized with an address which is within the page to be erased or with the address of the Block Protect Register (FLBPR) if the entire array to be erased.
- DATA — DATA is the first location of the DATA array and the location is specified by (H:X+4). The array contains programming data or data read from FLASH. The DATA array size must match the size of the range to be programmed or copied.

16.4.2 How to Use the Routines

This section describes the details of each routine. Table 16-10 provides necessary addresses used in the on-chip FLASH routines.

Table 16-10. Summary of On-Chip FLASH Support Routines

	GetByte	PutByte	Copy2RAM	rErase	rProgram
Jump Table Address	$FF7F	$FF82	$FF85	$FF88	$FF8B
Routine Description	Receive a data byte serially through PTA0	Send a data byte serially through PTA0	Copy a FLASH range to RAM	Erase a PAGE or entire array	Program a FLASH range
Internal Operating Frequency (f_{op})	N/A	N/A	N/A	1.5 MHz to 8.0 MHz	1.5 MHz to 8.0 MHz
Hardware Requirement	Pullup on PTA0	Pullup on PTA0	N/A	N/A	N/A
Entry Conditions	PTA0: Input (DDRA0 = 0)	PTA0: Input and 0 data bit (DDRA0 = 0, PTA0 = 0) A: data to be sent	H:X: RAM block start address (CPUSPD location) DATASIZE: number of bytes to be copied to DATA array ADDR: Start address of FLASH range	H:X: RAM block start address (CPUSPD location) CPUSPD: the nearest integer f_{op} (in MHz) times 4 ADDR: Page erase - an address of within the page Mass erase - FLBPR address	H:X: RAM block start address (CPUSPD location) CPUSPD: the nearest integer f_{op} (in MHz) times 4 DATASIZE: number of bytes to be program to FLASH ADDR: Start address of FLASH range Data array: Load data to be programmed
Exit Conditions	A: Data received through PTA0 C-bit: Framing error indicator (error: C = 0)	A, X: No change PTA0: Input and 0 data bit (DDRA0 = 0, PTA0 = 0)	H:X, CPUSPD, DATASIZE, ADDR: No change DATA array: Data placed with data read from FLASH	H:X, CPUSPD, ADDR: No change	H:X, CPUSPD, DATASIZE, ADDR, DATA array: No change
I Bit	I bit is preserved	I bit is preserved	I bit is preserved	I bit is set, then restored to entry condition on exit	I bit is set, then restored to entry condition on exit
COP	Not Serviced	Not Serviced	Not Serviced	Serviced	Serviced
RAM Variable	N/A	N/A	DATASIZE, ADDR (2 bytes), DATA array	CPUSPD, DATASIZE, ADDR (2 bytes), DATA array	CPUSPD, ADDR (2 bytes)
Stack Used (Including the Routine's Call)	6 bytes	6 bytes	11 bytes	10 bytes	15 bytes

16.4.2.1 GetByte

GetByte is a routine that receives a byte on the general-purpose I/O PTA0, and the received value is returned to the calling routine in the accumulator (A). This routine is also used in monitor mode so that it expects the same non-return-to-zero (NRZ) communication protocol and baud rates.

This routine detects a framing error when a STOP bit is not detected. If the carry (C) bit of the condition control register (CCR) is cleared after returning from this routine, a framing error occurred during the data receiving process. Therefore, the data in A is not reliable. The user software is responsible for handling such errors.

Interrupts are not masked (the I bit is not set) and the COP is not serviced in the GetByte routine. User software should ensure that interrupts are blocked during character reception.

In the GetByte routine, the baud rate is f_{op} divided by 256. When the internal operating frequency is 2.4576 MHz, the baud rate is 2.4576 MHz/256 = 9600.

To use this routine, some hardware setup is required. The general-purpose I/O PTA0 must be pulled up. For more information, refer to 16.3 Monitor Module (MON).

Entry Condition

PTA0 must be configured as an input and pulled up in hardware.

Exit Condition

A — Contains data received from PTA0.

C bit — Usually the C bit is set, indicating proper reception of the STOP bit. However, if the C bit is clear, a framing error occurred. Therefore, the received byte in A is not reliable. Example 16-1 shows how to receive a byte serially on PTA0.

Example 16-1. Receiving a Byte Serially

```
GetByte:       equu     $FF7F         ;GetByte jump address

               bclr     0,DDRA0       ;Configure port A bit 0 as an input

               jsr      GetByte       ;Call GetByte routine
               bcc      FrameError    ;If C bit is clear, framing error occurred.
                                      ; Take a proper action
```

NOTE

After GetByte is called, the program will remain in this routine until a START bit (0) is detected and a complete character is received.

16.4.2.2 PutByte

PutByte is a routine that receives a byte on the general-purpose I/O PTA0. The sent value must be loaded into the accumulator (A) before calling this routine. This routine is also used in the monitor mode. Therefore, it uses the same non-return-to-zero (NRZ) communication protocol. The communication baud rates are the same as those described in GetByte.

To use this routine, some hardware setup is required. The general-purpose I/O PTA0 must be pulled up and configured as an input and the PTA0 data bit must be initialized to 0.

Interrupts are not masked and the COP is not serviced in the PutByte routine. User software should ensure that interrupts are blocked during character transmission.

Entry Condition

A — Contains data sent from PTA0

PTA0 — This pin must be configured as an input and pulled up in hardware and the PTA0 data bit must be initialized to 0.

Exit Condition

A and X are restored with entry values.

Example 16-2 shows how to send a byte ($55) serially on PTA0.

Example 16-2. Sending a Byte Serially

```
PutByte:       equu     $FF82        ;PutByte jump address

               bclr     0,DDRA       ;Configure port A bit 0 as an input
               bclr     0,PTA        ;Initialize data bit to zero PTA0=0
               lda      #$55         ;Load sent data $55 to A

               jsr      PutByte      ;Call PutByte routine
```

16.4.2.3 Copy2RAM

Copy2RAM is a routine in which FLASH data in a contiguous range of FLASH locations is copied to the DATA array in RAM. The size of the DATA array must match the number of bytes specified in DATASIZE. The start address of the RAM block must be specified by H:X registers in the user software. The number of bytes to be copied to the DATA array and the first address of a FLASH range must be stored at DATASIZE and ADDR in the user software, respectively.

Interrupts are not masked and the COP is not serviced in the Copy2RAM routine.

Entry Condition

H:X — Contains the start address of the RAM block, which must point to the location of CPUSPD.

DATASIZE — Contains the number of bytes to be copied to the DATA array.

ADDR — Contains the first address in a FLASH range.

Exit Condition

DATA array — Replaced with data read from FLASH.

Example 16-3 shows how to use the Copy2RAM routine. In this example, the RAM block start address is $0100. 64 byte FLASH data in locations $FF00–$FF3F is copied to DATA array $0104–$0143.

Example 16-3. Copy data in $FF00–$FF3F to RAM

```
Copy2RAM:       equ     $FF85        ;Copy2RAM jump address

RAMblock:       equ     $0100        ;In this example, RAM block start address $0100.
DATASIZE:       equ     RAMblock+1   ;DATASIZE location
ADDR:           equ     RAMblock+2   ;ADDR location (2 bytes)

DATA:           equ     RAMblock+4   ;DATA array start address

                lda     #$40         ;Load 64 bytes to DATASIZE
                sta     DATASIZE

                ldhx    #$FF00       ;Load first address of the range to ADDR
                sthx    ADDR

                ldhx    #RAMblock    ;Load RAM block start address to H:X

                jsr     Copy2RAM     ;Call Copy2RAM routine
                               ;DATA array ($0104-$0143) contains data read from FLASH
```

16.4.2.4 rErase

rErase can be called to erase a page (64 bytes) or a whole array of FLASH. When the address of the FLASH block protect register is passed to rErase, the entire array is erased (MASS). Any other valid FLASH address selects the page erase. This routine supports an internal operating frequency between 1.5 MHz and 8.0 MHz.

NOTE

Mass erase using rErase is not allowed without V_{TST} *applied on* $\overline{IRQ}$ *pin*

In this routine, both PAGE erase time (t_{Erase}) and MASS erase time (t_{MErase}) are set between 4 ms and 5.5 ms. The CPUSPD value is the nearest integer of f_{op} (in MHz) times 4. For example if f_{op} is 3.1 MHz, the CPUSPD is 12 ($0C). If f_{op} is 4.9152 MHz, the CPUSPD is 20 ($14).

Interrupts are masked (I bit is set) during an erasing operation. When returning from this routine, I bit is restored to the entry condition, and the COP is serviced in rErase. The first COP is serviced on (72+3xCPUSPD) bus cycles after this routine is called in the user software.

A speed parameter and an address which is within the page to be erased or with the address of the FLBPR must be stored at CPUSPD and ADDR in the user software, respectively.

Entry Condition

H:X — Contains the start address of the RAM block, which must point to the location of CPUSPD.

CPUSPD — Contains the nearest integer value of f_{op} (in MHz) times 4.

ADDR — Contains an address within a desired erase page or FLBPR for mass erase.

Exit Condition

None

Example 16-4 shows how to erase an entire array. In this example, the RAM block start address is $0090.

Example 16-4. Erasing an Entire Array

```
rErase:        equ     $FF88        ;rErase jump address

RAMblock:      equ     $0090        ;In this example, RAM block start address $0090
CPUSPD:        equ     RAMblock     ;CPUSPD location
DATASIZE:      equ     RAMblock+1   ;DATASIZE location
ADDR:          equ     RAMblock+2   ;ADDR location (2 bytes)
DATA:          equ     RAMblock+4   ;DATA array start address

               mov     #$8,CPUSPD   ;fop = 2.0MHz in this example

               ldhx    #FLBPR       ;Load FLBPR address to H:X
               sthx    ADDR

               ldhx    #RAMblock    ;Load RAM block start address to H:X

               jsr     rErase       ;Call rErase routine
```

Example 16-5 shows how to erase a page from $E100 through $E13F. In this example, the RAM block start address is $0120.

Example 16-5. Erasing a Page

```
rErase:        equ     $FF88        ;rErase jump address

RAMblock:      equ     $0120        ;In this example, RAM block start address $0120
CPUSPD:        equ     RAMblock     ;CPUSPD location
DATASIZE:      equ     RAMblock+1   ;DATASIZE location
ADDR:          equ     RAMblock+2   ;ADDR location (2 bytes)
DATA:          equ     RAMblock+4   ;DATA array start address

               lda     #$14         ;fop = 4.9152MHz in this example
               sta     CPUSPD

               ldhx    #$E121       ;Load any address within the page to ADDR
               sthx    ADDR

               ldhx    #RAMblock    ;Load RAM start address to H:X

               jsr     rErase       ;Call rErase routine
```

If the FLASH locations that you want to erase are protected due to the value in the FLASH block protect register (FLBPR), the erase operation will not be successful. However when a high voltage (V_{TST}) is applied to the $\overline{\text{IRQ}}$ pin, the block protection is bypassed.

When the FLASH security check fails in the normal monitor mode, the FLASH can be re-accessed by erasing the entire FLASH array. To override the FLASH security mechanism and erase the FLASH array using this routine, registers H and X must contain the address of the FLASH block protect register (FLBPR).

16.4.2.5 rProgram

rProgram is used to program a range of FLASH locations with data loaded into the DATA array. Programming data is passed to rProgram in the DATA array. The DATA array locations are re-locatable so that the user can specify the locations in the user software. The size of the DATA array must match the size of a specified programming range. This routine supports an internal operating frequency between 1.5 MHz and 8.0 MHz.

For this split-gate FLASH, the programming algorithm requires a programming time (t_{prog}) between 30 µs and 40 µs. Table 16-11 shows how t_{prog} is adjusted by a CPUSPD value in this routine. The CPUSPD value is the nearest integer of f_{op} (in MHz) multiplied by 4. For example, if f_{op} is 2.4576 MHz, the CPUSPD value is 10 ($0A). If f_{op} is 8.0 MHz, the CPUSPD value is 32 ($20).

Table 16-11. t_{prog} vs. Internal Operating Frequency

	Internal Operating Freq. (f_{op})	CPUSPD	t_{prog} (Cycles)	t_{prog}
Case 1	1.50 MHz $\leq f_{op} <$ 1.625 MHz	6, 7	57	35.1 µs $< t_{prog} \leq$ 38.0 µs
Case 2	1.625MHz $\leq f_{op} \leq$ 8.0 MHz	7 to 32	8 x CPUSPD+ 8	33.0 µs $\leq t_{prog} \leq$ 39.4 µs

All programming is done using one programming algorithm. The algorithm allows for programming a single byte in each pass through it (one-byte programming method). Or, a whole row may be programmed by looping within the algorithm to write all the values in the row (row programming method).

- When the COPD bit in CONFIG1 is cleared and COP is therefore enabled, care must be taken to keep any programming operation from interfering with the servicing of the COP. In this case, each FLASH byte in the range is programmed using the one-byte programming method. Therefore, there are no limitations on range size and row/page boundary, but the total time to program multiple bytes is longer than the row programming method.
- When COPD bit is set (COP is disabled) and all programming addresses are in the same row, all FLASH bytes can be programmed at the same time using the row programming method. In this way, the FLASH can be programmed quickly.
- When COPD bit is set (COP is disabled) and a program range extends beyond a row or page, each FLASH byte in the range is programmed with the one-byte programming method until the beginning of the last row is reached. Then the bytes in the last row are programmed using the row programming method.

In rProgram, the high programming voltage time is enabled for less than 125 µs when programming a single byte at any internal operating frequency between 1.5 MHz and 8.0 MHz. Therefore, even when a row is programmed by 32 separate single-byte programming operations, the cumulative high voltage programming time is less than the maximum t_{HV} (4 ms). The t_{HV} is defined as the cumulative high voltage programming time to the same row before the next erase. For more information, refer to the memory characteristics in Chapter 17 Electrical Specifications.

This routine does not confirm that all bytes in the specified range are erased prior to programming. Nor does this routine perform a verification after programming, so there is no return confirmation that programming was successful. To program data successfully, the user software is responsible for these verifying operations.

Interrupts are masked (I bit is set) during a programming operation. When returning from this routine, I bit is restored to the entry condition. If the COP is enabled (COPD = 0), the COP is serviced in this routine.

The first COP is serviced at 61 bus cycles after this routine is called in the user software. When the COP is disabled (COPD = 1), this row programming method is the fastest way to program the FLASH.

The size of the DATA array must match the byte number specified in DATASIZE. The RAM block start address must be specified by H:X registers in the user software. A speed parameter, the number of bytes to be programmed to FLASH and the first address of a FLASH range must be stored at CPUSPD, DATASIZE and ADDR in the user software, respectively.

Entry Condition

H:X — Contains the start address of the RAM block, which must point to the location of CPUSPD.

CPUSPD — Contains the nearest integer value of f_{op} (in MHz) times 4.

DATASIZE — Contains the number of bytes to be programmed to FLASH.

ADDR — Contains the first address in a range.

DATA array — Contains the data values to be programmed into FLASH.

Exit Condition

H:X — Contains the address of the next byte after the range just programmed.

Example 16-6 shows how to program one full 32-byte row. In this example, the RAM block start address is $0080. 32 byte data is programmed to location $C000–$C01F.

Example 16-6. Programming a Row

```
rProgram:     equ     $FF8B         ;rProgram jump address

RAMblock:     equ     $0080         ;In this example, RAM block start address $0080
CPUSPD:       equ     RAMblock      ;CPUSPD location
DATASIZE:     equ     RAMblock+1    ;DATASIZE location
ADDR:         equ     RAMblock+2    ;ADDR location (2 bytes)
DATA:         equ     RAMblock+4    ;DATA array start address

              ldhx    #$0000        ;Index offset into DATA array
              lda     #$AA          ;Initial data value (inverted)
Data_load:
              coma                  ;Alternate between $55 and $AA. Fill DATA array,
              sta     DATA,x        ;32 bytes data, values to program into FLASH
              aix     #1            ;(ie. 55, AA, 55, AA....)
              cphx    #$20
              bne     Data_load

              mov     #$0A,CPUSPD   ;fop = 2.4576MHz in this example

              mov     #$20,DATASIZE;Load 32 bytes to DATASIZE

              ldhx    #$C000        ;Load first address of the row to ADDR
              sthx    ADDR

              ldhx    #RAMblock     ;Load RAM block start address to H:X

              jsr     rProgram      ;Call rProgram routine
```

rProgram can be used to program a range less than 32 bytes. Example 16-7 shows how to program $55 and $AA at location $E004 and $E005, respectively. In this example, a RAM block start address is $115.

Example 16-7. Programming a Range Smaller than a Row

```
rProgram:      equ     $FF8B         ;rProgram jump address

RAMblock:      equ     $0115         ;In this example, RAM block start address $0115
CPUSPD:        equ     RAMblock      ;CPUSPD location
DATASIZE:      equ     RAMblock+1    ;DATASIZE location
ADDR:          equ     RAMblock+2    ;ADDR location (2 bytes)
DATA:          equ     RAMblock+4    ;DATA array start address

               ldhx    #$55AA        ;Load data to DATA array
               sthx    DATA

               lda     #$18          ;fop = 6.0MHz in this example
               sta     CPUSPD

               lda     #2            ;Load 2 bytes to DATASIZE
               sta     DATASIZE

               ldhx    #$E005        ;Load last address to ADDR
               sthx    ADDR

               ldhx    #RAMblock     ;Load RAM block start address to H:X

               jsr     rProgram      ;Call rProgram routine
```

rProgram can also program a range beyond a page. Example 16-8 shows how to program 70-byte data to FLASH. In this example, the RAM block start address is $0100. The data is programmed to locations $C0F0–$C135.

Example 16-8. Programming a Range Bigger than a Page

```
rProgram:      equ     $FF8B         ;rProgram jump address

RAMblock:      equ     $0100         ;In this example, RAM block start address $0100
CPUSPD:        equ     RAMblock      ;CPUSPD location
DATASIZE:      equ     RAMblock+1    ;DATASIZE location
ADDR:          equ     RAMblock+2    ;ADDR location (2 bytes)
DATA:          equ     RAMblock+4    ;DATA array start address

               ldhx    #$0000        ;Index offset into DATA array
               lda     #$AA          ;Initial data value (inverted)
Data_load:
               coma                  ;Alternate between $55 and $AA. Fill DATA array,
               sta     DATA,x        ;70 bytes data, values to program into FLASH
               aix     #1            ;(ie. 55, AA, 55, AA....)
               cphx    #$46
               bne     Data_load
```

```
lda      #$18          ;fop = 8 MHz in this example
sta      CPUSPD

lda      #$46          ;Load 70 bytes to DATASIZE
sta      DATASIZE

ldhx     #$C0F0        ;Load first address of the range to ADDR
sthx     ADDR

ldhx     #RAMblock     ;Load RAM block start address to H:X

jsr      rProgram      ;Call rProgram routine
```

Chapter 17
Electrical Specifications

17.1 Introduction

This section contains electrical and timing specifications.

17.2 Absolute Maximum Ratings

Maximum ratings are the extreme limits to which the microcontroller unit (MCU) can be exposed without permanently damaging it.

NOTE

This device is not guaranteed to operate properly at the maximum ratings. Refer to 17.5 5-V DC Electrical Characteristics and 17.6 3-V DC Electrical Characteristics for guaranteed operating conditions.

Table 17-1. Absolute Maximum Ratings

Characteristic[1]	Symbol	Value	Unit
Supply voltage	V_{DD}	−0.3 to +6.0	V
LCD voltage	V_{LCD}	V_{SS} to +6.0	
Input voltage	V_{IN}	V_{SS}−0.3 to V_{DD} +0.3	V
Mode entry voltage, $\overline{\text{IRQ}}$ pin	V_{TST}	V_{SS}−0.3 to +8.5	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}.

NOTE

This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that V_{IN} and V_{OUT} be constrained to the range $V_{SS} \leq (V_{IN}$ or $V_{OUT}) \leq V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}.)

17.3 Functional Operating Range

Table 17-2. Operating Range

Characteristic	Symbol	Value	Unit
Operating temperature range	T_A (T_L to T_H)	− 40 to +85	°C
Operating voltage range	V_{DD}	2.7 to 5.5	V
LCD voltage	V_{LCD}	V_{SS} to V_{DD}	V

17.4 Thermal Characteristics

Table 17-3. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 52-pin LQFP	θ_{JA}	85	°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	$P_D \text{ x } (T_A + 273\ °C) + P_D^2 \times \theta_{JA}$	W/°C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C

1. Power dissipation is a function of temperature.
2. K 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.

17.5 5-V DC Electrical Characteristics

Table 17-4. DC Electrical Characteristics (5V)

Characteristic(1)	Symbol	Min	Typ(2)	Max	Unit
Output high voltage (I_{LOAD} = –2.5 mA) All ports	V_{OH}	V_{DD}–0.4	—	—	V
Output low voltage (I_{LOAD} = 2.5 mA) All ports except PTB2–PTB5 (I_{LOAD} = 15 mA) PTB2–PTB5	V_{OL}	—	—	0.4	V
Input high voltage All ports, $\overline{RST}$, $\overline{IRQ}$, OSC1	V_{IH}	0.7 × V_{DD}	—	V_{DD}	V
Input low voltage All ports, $\overline{RST}$, $\overline{IRQ}$, OSC1	V_{IL}	V_{SS}	—	0.3 × V_{DD}	V
V_{DD} supply current, f_{OP} = 8 MHz	I_{DD}				
Run(3) with all modules on		—	15	17	mA
Wait(4) with all modules off		—	5	6.5	mA
Stop(5)					
(−40°C to 85°C)		—	0.75	1	µA
25°C incremental current with XCLK enabled		—	15	—	µA
25°C incremental current with LCD enabled		—	20	—	µA
25°C incremental current with LVI enabled		—	150	—	µA
Digital 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
POR rearm voltage(6)	V_{POR}	750	—	—	mV
POR rise time ramp rate(7)	R_{POR}	0.035	—	—	V/ms
Monitor mode entry voltage	V_{TST}	V_{DD} + 2.5	—	9.1	V
Pullup resistors(8) PTA0–PTA3 as KBI0–KBI3, $\overline{RST}$, $\overline{IRQ}$	R_{PU}	17	25	28	kΩ
Low-voltage inhibit, trip falling voltage	V_{TRIPF}	4.00	4.20	4.30	V
Low-voltage inhibit, trip rising voltage	V_{TRIPR}	4.10	4.30	4.40	V
Low-voltage inhibit reset/recovery hysteresis	V_{HYS}	—	75	—	mV

1. V_{DD} = 4.5 to 5.5 Vdc, 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_{OP} = 8 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled.
4. Wait I_{DD} measured using external square wave clock source (f_{OP} = 8 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}.
5. Stop I_{DD} measured with OSC1 grounded; no port pins sourcing current.
6. Maximum is highest voltage that POR is guaranteed.
7. 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.
8. R_{PU} is measured at V_{DD} = 5.0 V.

17.6 3-V DC Electrical Characteristics

Table 17-5. DC Electrical Characteristics (3 V)

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage (I_{LOAD} = –2.5 mA) All ports	V_{OH}	$V_{DD} - 0.4$	—	—	V
Output low voltage (I_{LOAD} = 2.5 mA) All ports except PTB2–PTB5 (I_{LOAD} = 10 mA) PTB2–PTB5	V_{OL}	—	—	0.4	V
Input high voltage All ports, $\overline{\text{RST}}$, $\overline{\text{IRQ}}$, OSC1	V_{IH}	$0.7 \times V_{DD}$	—	V_{DD}	V
Input low voltage All ports, $\overline{\text{RST}}$, $\overline{\text{IRQ}}$, OSC1	V_{IL}	V_{SS}	—	$0.3 \times V_{DD}$	V
V_{DD} supply current, f_{OP} = 4 MHz	I_{DD}				
Run[3] with all modules on		—	5	6	mA
Wait[4] with all modules off		—	1.5	2.5	mA
Stop[5]					
(−40°C to 85°C)		—	0.65	0.8	µA
25°C incremental current with XCLK enabled		—	2	—	µA
25°C incremental current with LCD enabled		—	7	—	µA
25°C incremental current with LVI enabled		—	135	—	µA
Digital 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
POR rearm voltage[6]	V_{POR}	750	—	—	mV
POR rise time ramp rate[7]	R_{POR}	0.02	—	—	V/ms
Monitor mode entry voltage	V_{TST}	$V_{DD} + 2.5$	—	9.1	V
Pullup resistors[8] PTA0–PTA3 as KBI0–KBI3, $\overline{\text{RST}}$, $\overline{\text{IRQ}}$	R_{PU}	17	25	28	kΩ
Low-voltage inhibit, trip falling voltage	V_{TRIPF}	2.40	2.55	2.60	V
Low-voltage inhibit, trip rising voltage	V_{TRIPR}	2.46	2.61	2.66	V
Low-voltage inhibit reset/recovery hysteresis	V_{HYS}	—	55	—	mV

1. V_{DD} = 2.7 to 3.3 Vdc, 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_{OP} = 4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C_L = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I_{DD}. Measured with all modules enabled.
4. Wait I_{DD} measured using external square wave clock source (f_{OP} = 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}.
5. Stop I_{DD} measured with OSC1 grounded; no port pins sourcing current.
6. Maximum is highest voltage that POR is guaranteed.
7. If minimum V_{DD} is not reached before the internal POR reset is released, $\overline{\text{RST}}$ must be driven low externally until minimum V_{DD} is reached.
8. R_{PU} is measured at V_{DD} = 5.0 V.

17.7 5-V Control Timing

Table 17-6. Control Timing (5V)

Characteristic(1)	Symbol	Min	Max	Unit
Internal operating frequency	f_{OP}	—	8	MHz
$\overline{\text{RST}}$ input pulse width low(2)	t_{IRL}	50	—	ns
$\overline{\text{IRQ}}$ interrupt pulse width low (edge-triggered)(3)	t_{ILIH}	50	—	ns
$\overline{\text{IRQ}}$ interrupt pulse period(3)	t_{ILIL}	Note(4)	—	t_{CYC}

1. V_{DD} = 4.5 to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H; timing shown with respect to 20% V_{DD} and 70% V_{SS}, unless otherwise noted.
2. Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.
3. Values are based on characterization results, not tested in production.
4. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 t_{CYC}.

17.8 3-V Control Timing

Table 17-7. Control Timing (3V)

Characteristic(1)	Symbol	Min	Max	Unit
Internal operating frequency(2)	f_{OP}	—	4	MHz
$\overline{\text{RST}}$ input pulse width low(3)	t_{IRL}	125	—	ns
$\overline{\text{IRQ}}$ input pulse width low(3)	t_{IIL}	125	—	ns
TIM2 external clock input	f_{T2CLK}	—	2	MHz

1. V_{DD} = 2.7 to 3.3 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H; timing shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.
2. Some modules may require a minimum frequency greater than dc for proper operation; see appropriate table for this information.
3. Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.

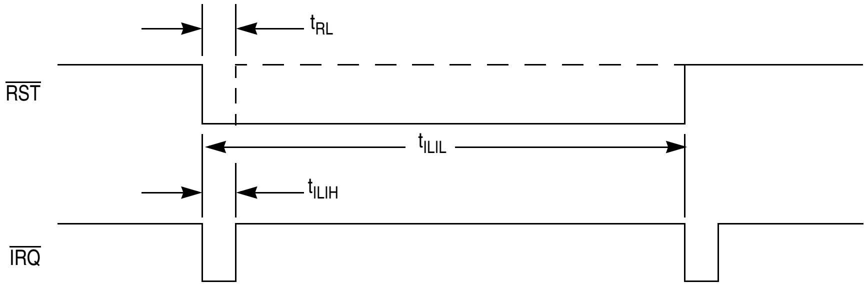


Figure 17-1. $\overline{\text{RST}}$ and $\overline{\text{IRQ}}$ Timing

17.9 Timer Interface Module Characteristics

Table 17-8. Timer Interface Module Characteristics (5V and 3V)

Characteristic	Symbol	Min	Max	Unit
Input capture pulse width	t_{TIH}, t_{TIL}	$1/f_{OP}$	—	

17.10 ADC10 Characteristics

Table 17-9. ADC10 Characteristics

Characteristic	Conditions	Symbol	Min	Typ[(1)]	Max	Unit	Comment
Supply voltage	Absolute	V_{DD}	2.7	—	5.5	V	
Supply Current ALPC = 1 ALSMP = 1 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}[(2)]	—	55	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	75	—		
Supply current ALPC = 1 ALSMP = 0 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}[(2)]	—	120	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	175	—		
Supply current ALPC = 0 ALSMP = 1 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}[(2)]	—	140	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	180	—		
Supply current ALPC = 0 ALSMP = 0 ADCO = 1	$V_{DD} \leq 3.3$ V (3.0 V Typ)	I_{DD}[(2)]	—	340	—	µA	
	$V_{DD} \leq 5.5$ V (5.0 V Typ)		—	440	—		
ADC internal clock	High speed (ALPC = 0)	f_{ADCK}	0.40[(3)]	—	2.00	MHz	$t_{ADCK} = 1/f_{ADCK}$
	Low power (ALPC = 1)		0.40[(3)]	—	1.00		
10-Bit Mode Conversion time	Short sample (ALSMP = 0)	t_{ADC}	19	19	21	t_{ADCK} cycles	$t_{Bus} = 1/f_{Bus}$ cycles
	Long sample (ALSMP = 1)		39	39	41		
8-Bit Mode Conversion time	Short sample (ALSMP = 0)	t_{ADC}	16	16	18	t_{ADCK} cycles	$t_{Bus} = 1/f_{Bus}$ cycles
	Long sample (ALSMP = 1)		36	36	38		
Sample time	Short sample (ALSMP = 0)	t_{ADS}	4	4	4	t_{ADCK} cycles	
	Long sample (ALSMP = 1)		24	24	24		
Input voltage		V_{ADIN}	V_{SS}	—	V_{DD}	V	
Input capacitance		C_{ADIN}	—	7	10	pF	Not tested
Input impedance		R_{ADIN}	—	5	15	kΩ	Not tested
Analog source impedance		R_{AS}	—	—	10	kΩ	External to MCU
Ideal resolution (1 LSB)	10-bit mode	RES	1.758	5	5.371	mV	$V_{REFH}/2^N$
	8-bit mode		7.031	20	21.48		
Total unadjusted error	10-bit mode	E_{TUE}	0	±2.0	±2.5	LSB	Includes quantization
	8-bit mode		0	±0.7	±1.0		

— Continued on next page

Table 17-9. ADC10 Characteristics

Characteristic	Conditions	Symbol	Min	Typ(1)	Max	Unit	Comment
Differential non-linearity	10-bit mode	DNL	0	±0.5	—	LSB	
	8-bit mode		0	±0.3	—		
	Monotonicity and no-missing-codes guaranteed						
Integral non-linearity	10-bit mode	INL	0	±0.5	—	LSB	
	8-bit mode		0	±0.3	—		
Zero-scale error	10-bit mode	E_{ZS}	0	±0.5	—	LSB	$V_{ADIN} = V_{SS}$
	8-bit mode		0	±0.3	—		
Full-scale error	10-bit mode	E_{FS}	0	±2.0	—	LSB	$V_{ADIN} = V_{DD}$
	8-bit mode		0	±0.3	—		
Quantization error	10-bit mode	E_Q	—	—	±0.5	LSB	8-bit mode is not truncated
	8-bit mode		—	—	±0.5		
Input leakage error	10-bit mode	E_{IL}	0	±0.2	±5	LSB	Pad leakage(4) * R_{AS}
	8-bit mode		0	±0.1	±1.2		
Bandgap voltage input(5)		V_{BG}	1.17	1.245	1.32	V	

1. Typical values assume V_{DD} = 5.0 V, temperature = 25°C, f_{ADCK} = 1.0 MHz unless otherwise stated. Typical values are for reference only and are not tested in production.
2. Incremental I_{DD} added to MCU mode current.
3. Values are based on characterization results, not tested in production.
4. Based on typical input pad leakage current.
5. LVI must be enabled, (LVID = 0, in CONFIG1). Voltage input to ADCH4:0 = $1A, an ADC conversion on this channel allows user to determine supply voltage.

17.11 Clock Generation Module Characteristics

17.11.1 CGM Component Specifications

Table 17-10. CGM Component Specifications

Characteristic	Symbol	Min	Typ	Max	Unit
External reference clock to OSC1 (1)	f_{OSC}	dc	32.768	—	kHz
Crystal reference frequency	$f_{XTALCLK}$	30	32.768	100	kHz
Crystal load capacitance (2)	C_L	—	12.5	—	pF
Crystal fixed capacitance (3)	C_1	—	15	—	pF
Crystal tuning capacitance (4)	C_2	—	15	—	pF
Feedback bias resistor	R_B	1	10	22	MΩ
Series resistor (3)	R_S	100	330	470	kΩ

1. No more than 10% duty cycle deviation from 50%. The max. frequency is limited by an EMC filter.
2. Crystal manufacturer value.
3. Capacitor on OSC1 pin. Does not include parasitic capacitance due to package, pin, and board.
4. Capacitor on OSC2 pin. Does not include parasitic capacitance due to package, pin, and board.

17.11.2 CGM Electrical Specifications

Table 17-11. CGM Electrical Specifications

Description	Symbol	Min	Typ	Max	Unit
Operating voltage	V_{DD}	2.7	—	5.5	V
Operating temperature	T	–40	25	85	°C
Reference frequency	f_{RDV}	30	32.768	100	kHz
Range nominal multiplier	f_{NOM}	—	38.4	—	kHz
VCO center-of-range frequency(1)	f_{VRS}	38.4 k	—	40.0 M	Hz
Medium-voltage VCO center-of-range frequency(2)	f_{VRS}	38.4 k	—	40.0 M	Hz
VCO range linear range multiplier	L	1	—	255	
VCO power-of-two range multiplier	2^E	1	—	4	
VCO multiply factor	N	1	—	4095	
VCO prescale multiplier	2^P	1	1	8	
Reference divider factor	R	1	1	15	
VCO operating frequency	f_{VCLK}	38.4 k	—	40.0 M	Hz
Bus operating frequency(1)	f_{BUS}	—	—	8.2	MHz
Bus frequency @ medium voltage(2)	f_{BUS}	—	—	4.1	MHz
Manual acquisition time	t_{Lock}	—	20	50	ms
Automatic lock time	t_{Lock}	—	20	50	ms
PLL jitter(3)	f_J	0	—	$f_{RCLK} \times 0.025\% \times 2^P N/4$	Hz
External clock input frequency PLL disabled	f_{OSC}	Not allowed			Hz
External clock input frequency PLL enabled	f_{OSC}	Not allowed			Hz

1. 5.0 V ± 10% V_{DD}
2. 3.0 V ± 10% V_{DD}
3. Deviation of average bus frequency over 2 ms. N = VCO multiplier.

17.12 Memory Characteristics

Table 17-12. Memory Characteristics

Characteristic	Symbol	Min.	Max.	Unit
Data Retention Voltage	V_{RDR}	1.3	—	V
Number of row per page	—	2	2	Rows
Number of byte per page	—	64	64	Bytes
Read bus clock frequency	F_{read}(1)	32k	8M	Hz
Page Erase time	T_{erase}(2)	1	—	ms
Mass erase time	T_{merase}(3)	4	—	ms
Flash PGM/ERASE to NVSTR setup time	t_{NVS}	10	—	µs
High-voltage hold time	t_{nvh}	5	—	µs
High-voltage hold time (mass erase)	t_{nvhl}	100	—	µs
Program hold time	T_{pgs}	5	—	µs
Program time	T_{prog}	30	40	µs
address/data setup time	T_{ads}	—	30	ns
address/data hold time	T_{adh}	—	30	ns
Recovery time	T_{rcv}(4)	1	—	µs
Flash cumulative HV period	T_{hv}(5)	—	25	ms
Row erase endurance(6)	—	10k	—	cycles
Row program endurance (7)	—	10k	—	cycles
Data retention time(8)	—	10	—	Years

1. F_{read} is defined as the frequency range for which the Flash memory can be read.
2. If the page erase time is longer than T_{erase} (Min.), there is no erase-disturb, but it reduces the endurance of the Flash memory.
3. If the mass erase time is longer than T_{me} (Min.), there is no erase-disturb, but is reduces the endurance of the Flash memory.
4. It is defined as the time it needs before the Flash can be read after turning off the high voltage charge pump, by clearing HVEN to logic "0".
5. T_{hv} is the cumulative high voltage programming time to the same row before next erase, and the same address can not be programmed twice before next erase.
6. The minimum row endurance value specifies each row of the Flash memory is guaranteed to work for at least this many erase/program cycles.
7. The minimum row endurance value specifies each row of the Flash memory is guaranteed to work for at least this many erase/program cycle.
8. The Flash is guaranteed to retain data over the entire operating temperature range for at least the minimum time specified.

Chapter 18
Ordering Information and Mechanical Specifications

18.1 Introduction

This section contains order numbers for the MC68HC908LV8. Dimensions are given for:

- 52-pin low-profile quad flat pack (LQFP)

18.2 MC Order Numbers

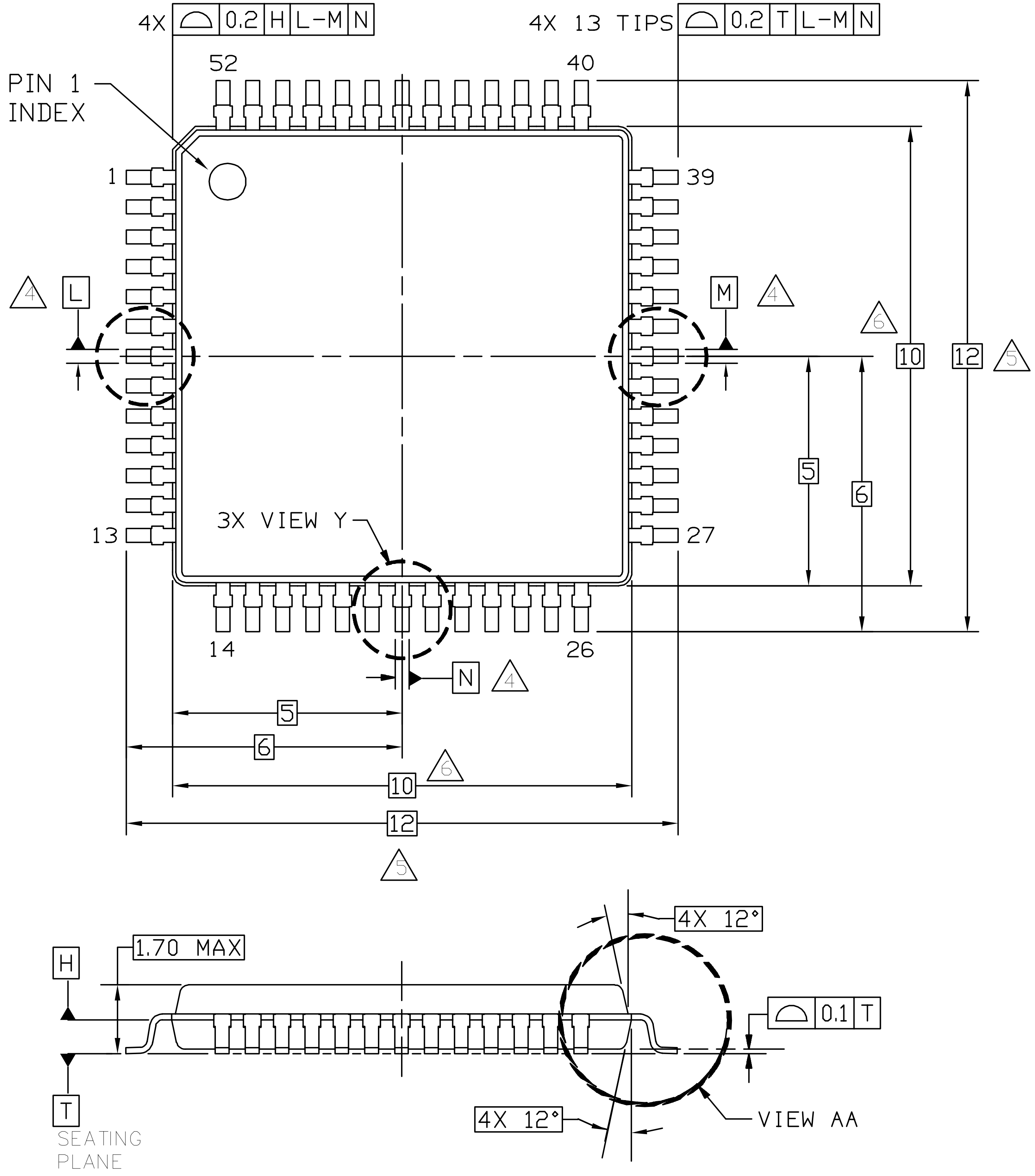
Table 18-1. MC Order Numbers

MC Order Number	Operating Temperature Range	Package
MC68HC908LV8CPBE	–40 to +85 °C	52-pin LQFP

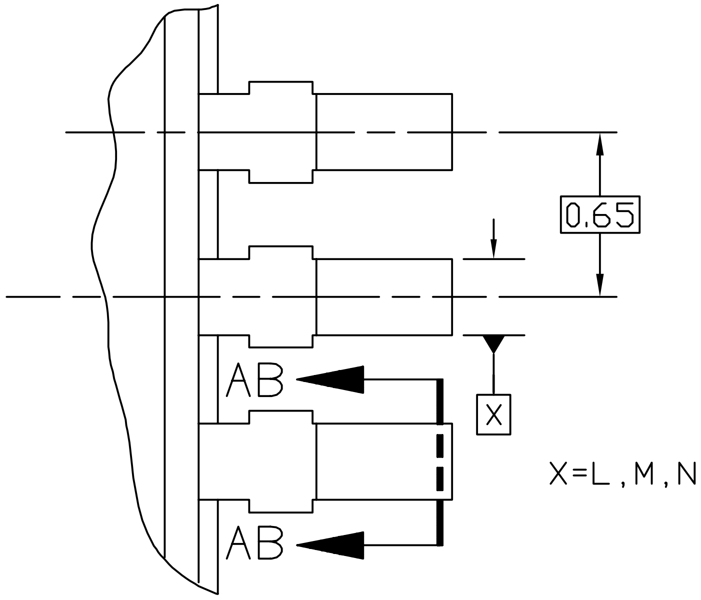
Temperature and package designators:
C = –40 to +85 °C
PB= Low-profile quad flat pack (LQFP)
E = RoHS

18.3 Package Dimensions

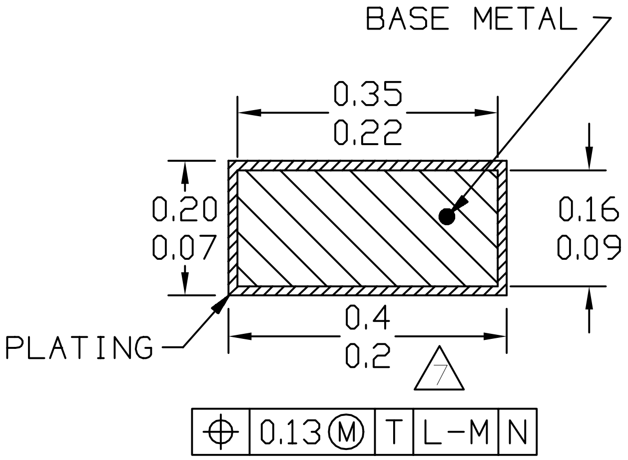
Refer to the following pages for detailed package dimensions.



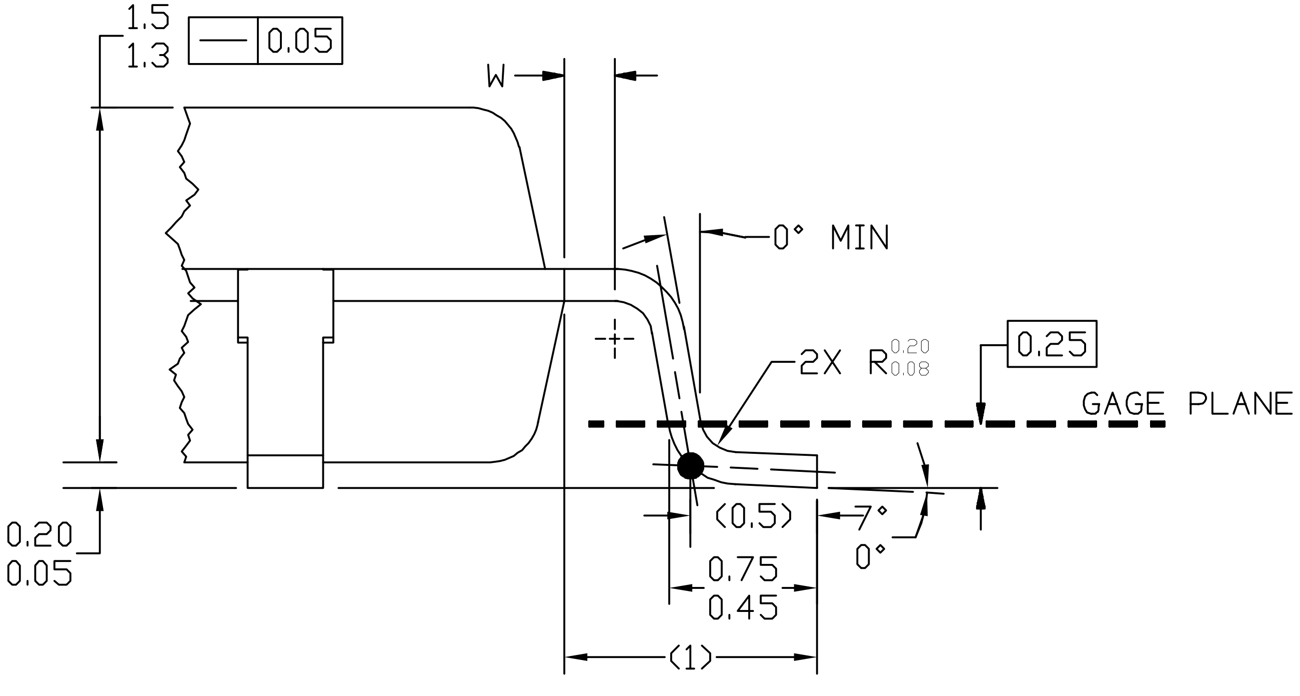
MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE	
TITLE: 52LD TQFP 10 X 10 PKG, 0.65 PITCH, 1.4 THICK	DOCUMENT NO: 98ASS23228W	REV: F
	CASE NUMBER: 848D-03	05 MAY 2005
	STANDARD: NON-JEDEC	



VIEW Y



SECTION AB-AB
ROTATED 90° CLOCKWISE



VIEW AA

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	**MECHANICAL OUTLINE**	PRINT VERSION NOT TO SCALE
TITLE: 52LD TQFP 10 X 10 PKG, 0.65 PITCH, 1.4 THICK	DOCUMENT NO: 98ASS23228W	REV: F
	CASE NUMBER: 848D-03	05 MAY 2005
	STANDARD: NON-JEDEC	

NOTES

1. DIMENSIONS AND TOLERANCING PER ASME Y14.5M. 1994.

2. CONTROLLING DIMENSION: MILLIMETER.

3. DATUM PLANE H IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.

4. DATUMS L, M, AND N TO BE DETERMINED AT DATUM PLANE H.

5. DIMENSIONS TO BE DETERMINED AT SEATING PLANE T.

6. DIMENSIONS DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 PER SIDE. DIMENSIONS DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE H.

7. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. DAMBAR PROTRUSION SHALL NOT CAUSE THE LEAD WIDTH TO EXCEED 0.46. MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD OR PROTRUSION 0.07.

MECHANICAL OUTLINE

PRINT VERSION NOT TO SCALE

TITLE:
52LD TQFP
10 X 10 PKG, 0.65 PITCH, 1.4 THICK

DOCUMENT NO: 98ASS23228W

REV: F

CASE NUMBER: 848D-03

05 MAY 2005

STANDARD: NON-JEDEC

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

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

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

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 that 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.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.
© Freescale Semiconductor, Inc. 2005. All rights reserved.

freescale™
semiconductor