MC68HC908JW32

Data Sheet

M68HC08
Microcontrollers

MC68HC908JW32
Rev. 5
10/2006

freescale.com

freescale™
semiconductor

MC68HC908JW32

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://www.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)
January, 2005	2	First general release.	—
March, 2005	3	Second general release. Cleaned typos.	—
October, 2006	4	Table 4-1. Instruction Set Summary — Updated definition for the STOP instruction and added WAIT instruction.	48
		5.4 I/O Signals — Removed subsections referring to V_{DDA} and V_{SSA}.	64
		Figure 5-1. CGM Block Diagram — Corrected references to V_{DDA} and V_{SSA} to V_{DD} and V_{SS}.	56
		Figure 5-3. CGM External Connections — Removed V_{DD} connection to V_{DDPLL}.	63
		Figure 5-10. PLL Filter — Corrected reference to V_{SSA} to V_{SS}.	72
		Figure 7-1. Monitor Mode Circuit — Corrected V_{DDPLL} connection.	92
		Chapter 20 Ordering Information and Mechanical Specifications — Combined ordering information and mechanical specifications. Updated package dimensions to the latest available at time of publication.	219
October, 2006	5	1.7.2 Analogy Power Supply (V_{DDPLL} and V_{SSPLL}) — Reworked for clarity.	21

List of Chapters

Chapter 1 General Description 17

Chapter 2 Memory 23

Chapter 3 Configuration Registers (CONFIG) 39

Chapter 4 Central Processor Unit (CPU) 43

Chapter 5 Clock Generator Module (CGM) 55

Chapter 6 System Integration Module (SIM) 73

Chapter 7 Monitor ROM (MON) 91

Chapter 8 Timer Interface Module (TIM) 105

Chapter 9 Timebase Module (TBM) 121

Chapter 10 Serial Peripheral Interface Module (SPI) 125

Chapter 11 USB 2.0 FS Module 145

Chapter 12 PS2 Clock Generator (PS2CLK) 161

Chapter 13 Input/Output (I/O) Ports 165

Chapter 14 External Interrupt (IRQ) 183

Chapter 15 Keyboard Interrupt Module (KBI) 189

Chapter 16 Computer Operating Properly (COP) 195

Chapter 17 Low-Voltage Inhibit (LVI) 199

Chapter 18 Break Module (BRK) 203

Chapter 19 Electrical Specifications 209

Chapter 20 Ordering Information and Mechanical Specifications 219

Table of Contents

Chapter 1
General Description

1.1 Introduction . . . 17
1.2 Features . . . 17
1.3 MCU Block Diagram . . . 18
1.4 Pin Assignments . . . 19
1.5 Clock Tree . . . 19
1.6 Power Management . . . 20
1.7 Pin Function . . . 21
1.7.1 Power Supply Pins (V_{DD} and V_{SS}) . . . 21
1.7.2 Analogy Power Supply (V_{DDPLL} and V_{SSPLL}) . . . 21
1.7.3 Internal Voltage Regulator Supply (REG25V, REG33V, and V_{SS33}) . . . 21
1.7.4 Oscillator Pins (OSC1 and OSC2) . . . 21
1.7.5 External Reset Pin ($\overline{RST}$) . . . 22
1.7.6 External Interrupt Pin ($\overline{IRQ}$) . . . 22
1.7.7 External Filter Capacitor Pin (CGMXFC) . . . 22
1.7.8 Port A Input/Output (I/O) Pins (PTA7–PTA0) . . . 22
1.7.9 Port B Input/Output (I/O) Pins (PTB5, PTB1, PTB0) . . . 22
1.7.10 Port C Input/Output (I/O) Pins (PTC3–PTC0) . . . 22
1.7.11 Port D Input/Output (I/O) Pins (PTD7–PTD0) . . . 22
1.7.12 Port E Input/Output (I/O) Pins (PTE7–PTE2) . . . 22

Chapter 2
Memory

2.1 Introduction . . . 23
2.2 Input/Output I/O Section . . . 23
2.3 Monitor ROM . . . 23
2.4 Random-Access Memory (RAM) . . . 33
2.5 FLASH Memory . . . 33
2.5.1 Functional Description . . . 33
2.5.2 FLASH Control Register . . . 34
2.5.3 FLASH Page Erase Operation . . . 34
2.5.4 FLASH Mass Erase Operation . . . 35
2.5.5 FLASH Program Operation . . . 35
2.5.6 FLASH Protection . . . 36
2.5.7 FLASH Block Protect Register . . . 38

Chapter 3
Configuration Registers (CONFIG)

3.1 Introduction . . . 39
3.2 Functional Description . . . 39
3.3 Configuration Register 1 (CONFIG1) . . . 40
3.4 Configuration Register 2 (CONFIG2) . . . 41

Chapter 4
Central Processor Unit (CPU)

4.1 Introduction . . . 43
4.2 Features . . . 43
4.3 CPU Registers . . . 43
4.3.1 Accumulator . . . 44
4.3.2 Index Register . . . 44
4.3.3 Stack Pointer . . . 45
4.3.4 Program Counter . . . 45
4.3.5 Condition Code Register . . . 46
4.4 Arithmetic/Logic Unit (ALU) . . . 47
4.5 Low-Power Modes . . . 47
4.5.1 Wait Mode . . . 47
4.5.2 Stop Mode . . . 47
4.6 CPU During Break Interrupts . . . 47
4.7 Instruction Set Summary . . . 48
4.8 Opcode Map . . . 53

Chapter 5
Clock Generator Module (CGM)

5.1 Introduction . . . 55
5.2 Features . . . 55
5.3 Functional Description . . . 55
5.3.1 Oscillator Module . . . 57
5.3.2 Phase-Locked Loop Circuit (PLL) . . . 57
5.3.3 PLL Circuits . . . 58
5.3.4 Acquisition and Tracking Modes . . . 59
5.3.5 Manual and Automatic PLL Bandwidth Modes . . . 59
5.3.6 Programming the PLL . . . 60
5.3.7 Special Programming Exceptions . . . 62
5.3.8 Base Clock Selector Circuit . . . 62
5.3.9 CGM External Connections . . . 63
5.4 I/O Signals . . . 64
5.4.1 Crystal Amplifier Input Pin (OSC1) . . . 64
5.4.2 Crystal Amplifier Output Pin (OSC2) . . . 64
5.4.3 External Filter Capacitor Pin (CGMXFC) . . . 64
5.4.4 Oscillator Output Frequency Signal (CGMXCLK) . . . 64
5.4.5 CGM Reference Clock (CGMRCLK) . . . 64

5.4.6 CGM VCO Clock Output (CGMVCLK) . . . 64
5.4.7 CGM Base Clock Output (CGMOUT) . . . 64
5.4.8 CGM CPU Interrupt (CGMINT) . . . 64
5.5 CGM Registers . . . 65
5.5.1 PLL Control Register . . . 65
5.5.2 PLL Bandwidth Control Register . . . 67
5.5.3 PLL Multiplier Select Registers . . . 68
5.5.4 PLL VCO Range Select Register . . . 68
5.5.5 PLL Reference Divider Select Register . . . 69
5.6 Interrupts . . . 69
5.7 Special Modes . . . 70
5.7.1 Wait Mode . . . 70
5.7.2 Stop Mode . . . 70
5.7.3 CGM During Break Interrupts . . . 70
5.8 Acquisition/Lock Time Specifications . . . 71
5.8.1 Acquisition/Lock Time Definitions . . . 71
5.8.2 Parametric Influences on Reaction Time . . . 71
5.8.3 Choosing a Filter . . . 72

Chapter 6
System Integration Module (SIM)

6.1 Introduction . . . 73
6.2 SIM Bus Clock Control and Generation . . . 75
6.2.1 Bus Timing . . . 75
6.2.2 Clock Start-up from POR or LVI Reset . . . 76
6.2.3 Clocks in Stop Mode and Wait Mode . . . 76
6.3 Reset and System Initialization . . . 76
6.3.1 External Pin Reset . . . 76
6.3.2 Active Resets from Internal Sources . . . 77
6.3.2.1 Power-On Reset . . . 77
6.3.2.2 Computer Operating Properly (COP) Reset . . . 78
6.3.2.3 Illegal Opcode Reset . . . 78
6.3.2.4 Illegal Address Reset . . . 79
6.3.2.5 Low-Voltage Inhibit (LVI) Reset . . . 79
6.3.2.6 Universal Serial Bus (USB) Reset . . . 79
6.4 SIM Counter . . . 79
6.4.1 SIM Counter During Power-On Reset . . . 79
6.4.2 SIM Counter During Stop Mode Recovery . . . 80
6.4.3 SIM Counter and Reset States . . . 80
6.5 Exception Control . . . 80
6.5.1 Interrupts . . . 80
6.5.1.1 Hardware Interrupts . . . 81
6.5.1.2 SWI Instruction . . . 82
6.5.2 Interrupt Status Registers . . . 82
6.5.2.1 Interrupt Status Register 1 . . . 83
6.5.2.2 Interrupt Status Register 2 . . . 83
6.5.2.3 Interrupt Status Register 3 . . . 83

6.5.3 Reset . . . 85
6.5.4 Break Interrupts . . . 85
6.5.5 Status Flag Protection in Break Mode . . . 85
6.6 Low-Power Modes . . . 85
6.6.1 Wait Mode . . . 85
6.6.2 Stop Mode . . . 87
6.7 SIM Registers . . . 88
6.7.1 SIM Break Status Register . . . 88
6.7.2 SIM Reset Status Register . . . 89
6.7.3 SIM Break Flag Control Register . . . 90

Chapter 7
Monitor ROM (MON)

7.1 Introduction . . . 91
7.2 Features . . . 91
7.3 Functional Description . . . 91
7.3.1 Entering Monitor Mode . . . 93
7.3.2 Data Format . . . 94
7.3.3 Break Signal . . . 95
7.3.4 Baud Rate . . . 95
7.3.5 Commands . . . 95
7.4 Security . . . 99
7.5 ROM-Resident Routines . . . 101
7.5.1 PRGRNGE . . . 102
7.5.2 ERARNGE . . . 103
7.5.3 LDRNGE . . . 104

Chapter 8
Timer Interface Module (TIM)

8.1 Introduction . . . 105
8.2 Features . . . 105
8.3 Pin Name Conventions . . . 105
8.4 Functional Description . . . 106
8.4.1 TIM Counter Prescaler . . . 107
8.4.2 Input Capture . . . 107
8.4.3 Output Compare . . . 108
8.4.3.1 Unbuffered Output Compare . . . 108
8.4.3.2 Buffered Output Compare . . . 108
8.4.4 Pulse Width Modulation (PWM) . . . 109
8.4.4.1 Unbuffered PWM Signal Generation . . . 109
8.4.4.2 Buffered PWM Signal Generation . . . 110
8.4.4.3 PWM Initialization . . . 110
8.5 Interrupts . . . 111
8.6 Low-Power Modes . . . 111
8.6.1 Wait Mode . . . 111
8.6.2 Stop Mode . . . 112

8.7 TIM During Break Interrupts . . . 112
8.8 I/O Signals . . . 112
8.8.1 TIM Clock Pin (PTC1/TCLK1) . . . 112
8.9 I/O Registers . . . 112
8.9.1 TIM Status and Control Register . . . 113
8.9.2 TIM Counter Registers . . . 114
8.9.3 TIM Counter Modulo Registers . . . 115
8.9.4 TIM Channel Status and Control Registers . . . 115
8.9.5 TIM Channel Registers . . . 118

Chapter 9
Timebase Module (TBM)

9.1 Introduction . . . 121
9.2 Features . . . 121
9.3 Functional Description . . . 121
9.4 Timebase Register Description . . . 122
9.5 Interrupts . . . 123
9.6 Low-Power Modes . . . 124
9.6.1 Wait Mode . . . 124
9.6.2 Stop Mode . . . 124

Chapter 10
Serial Peripheral Interface Module (SPI)

10.1 Introduction . . . 125
10.2 Features . . . 125
10.3 Pin Name Conventions and I/O Register Addresses . . . 125
10.4 Functional Description . . . 126
10.4.1 Master Mode . . . 126
10.4.2 Slave Mode . . . 128
10.5 Transmission Formats . . . 128
10.5.1 Clock Phase and Polarity Controls . . . 128
10.5.2 Transmission Format When CPHA = 0 . . . 129
10.5.3 Transmission Format When CPHA = 1 . . . 130
10.5.4 Transmission Initiation Latency . . . 130
10.6 Queuing Transmission Data . . . 131
10.7 Error Conditions . . . 132
10.7.1 Overflow Error . . . 133
10.7.2 Mode Fault Error . . . 134
10.8 Interrupts . . . 135
10.9 Resetting the SPI . . . 137
10.10 Low-Power Modes . . . 137
10.10.1 Wait Mode . . . 137
10.10.2 Stop Mode . . . 137
10.11 SPI During Break Interrupts . . . 137

10.12 I/O Signals . . . 138
10.12.1 MISO (Master In/Slave Out) . . . 138
10.12.2 MOSI (Master Out/Slave In) . . . 138
10.12.3 SPSCK (Serial Clock) . . . 138
10.12.4 $\overline{SS}$ (Slave Select) . . . 139
10.12.5 CGND (Clock Ground) . . . 140
10.13 I/O Registers . . . 140
10.13.1 SPI Control Register . . . 140
10.13.2 SPI Status and Control Register . . . 141
10.13.3 SPI Data Register . . . 143

Chapter 11
USB 2.0 FS Module

11.1 Introduction . . . 145
11.2 Features . . . 145
11.3 USB Module Architecture . . . 146
11.3.1 USB Transceiver . . . 146
11.3.2 USB Control Logic . . . 147
11.3.3 USB Endpoint Configuration . . . 147
11.3.4 USB Requestor Processor . . . 148
11.3.4.1 Configuration Process . . . 149
11.3.4.2 Control Endpoint 0 . . . 149
11.3.5 Endpoint Controller . . . 149
11.3.5.1 OUT endpoint Data Transfer . . . 150
11.3.5.2 IN endpoint Data Transfer . . . 150
11.4 Interrupt Source . . . 150
11.5 USB Module Registers . . . 151
11.5.1 USB Control Register (USBCR) . . . 153
11.5.2 USB Status Register (USBSR) . . . 154
11.5.3 USB Status Interrupt Mask Register (USIMR) . . . 155
11.5.4 USB Endpoint 0 Control/Status Register (UEP0CSR) . . . 156
11.5.5 USB Endpoint 1–4 Control Status Register (UEP1CSR–UEP4CSR) . . . 157
11.5.6 USB Endpoint 1–4 Data Size Register (UEP1DSR–UEP4DSR) . . . 159
11.5.7 USB Endpoint 1/2 and 3/4 Base Pointer Register (UEP12BPR–UEP34BPR) . . . 159
11.5.8 USB Interface Control Register (UINTFCR) . . . 160
11.5.9 USB Endpoint 0 Data Register 7–0 (UE0D7–UE0D0) . . . 160

Chapter 12
PS2 Clock Generator (PS2CLK)

12.1 Introduction . . . 161
12.2 Functional Description . . . 161
12.3 PS2 Clock Generator Control and Status Registers . . . 162

Chapter 13
Input/Output (I/O) Ports

13.1 Introduction . . . 165
13.2 Port A . . . 168
13.2.1 Port A Data Register . . . 168
13.2.2 Data Direction Register A . . . 168
13.3 Port B . . . 170
13.3.1 Port B Data Register . . . 170
13.3.2 Data Direction Register B . . . 170
13.4 Port C . . . 172
13.4.1 Port C Data Register . . . 172
13.4.2 Data Direction Register C . . . 173
13.5 Port D . . . 174
13.5.1 Port D Data Register . . . 174
13.5.2 Data Direction Register D . . . 174
13.6 Port E . . . 176
13.6.1 Port E Data Register . . . 176
13.6.2 Data Direction Register E . . . 178
13.7 Port Options . . . 180
13.7.1 Port Option Control Register 1 . . . 180
13.7.2 Port Option Control Register 2 . . . 180
13.7.3 Pullup Control Register (PULLCR) . . . 181

Chapter 14
External Interrupt (IRQ)

14.1 Introduction . . . 183
14.2 Features . . . 183
14.3 Functional Description . . . 183
14.4 $\overline{\text{IRQ}}$ Pin . . . 185
14.5 PTE3/D– Pin . . . 186
14.6 IRQ Module During Break Interrupts . . . 186
14.7 IRQ Status and Control Register . . . 186
14.8 IRQ Option Control Register . . . 187

Chapter 15
Keyboard Interrupt Module (KBI)

15.1 Introduction . . . 189
15.2 Features . . . 189
15.3 Pin Name Conventions . . . 189
15.4 Functional Description . . . 190
15.4.1 Keyboard Initialization . . . 191
15.5 I/O Registers . . . 192
15.5.1 Keyboard Status and Control Register . . . 192
15.5.2 Keyboard Interrupt Enable Register . . . 193

15.6 Low-Power Modes . . . 193
15.6.1 Wait Mode . . . 193
15.6.2 Stop Mode . . . 193
15.7 Keyboard Module During Break Interrupts . . . 193

Chapter 16
Computer Operating Properly (COP)

16.1 Introduction . . . 195
16.2 Functional Description . . . 195
16.3 I/O Signals . . . 196
16.3.1 CGMRCLK . . . 196
16.3.2 STOP Instruction . . . 196
16.3.3 COPCTL Write . . . 196
16.3.4 Power-On Reset . . . 196
16.3.5 Internal Reset . . . 196
16.3.6 Reset Vector Fetch . . . 197
16.3.7 COPD (COP Disable) . . . 197
16.3.8 COPRS (COP Rate Select) . . . 197
16.4 COP Control Register . . . 197
16.5 Interrupts . . . 197
16.6 Monitor Mode . . . 198
16.7 Low-Power Modes . . . 198
16.7.1 Wait Mode . . . 198
16.7.2 Stop Mode . . . 198
16.8 COP Module During Break Mode . . . 198

Chapter 17
Low-Voltage Inhibit (LVI)

17.1 Introduction . . . 199
17.2 Features . . . 199
17.3 Functional Description . . . 199
17.3.1 Low V_{DD} Detector . . . 200
17.3.2 Polled LVI Operation . . . 200
17.3.3 Forced Reset Operation . . . 200
17.3.4 Voltage Hysteresis Protection . . . 200
17.4 LVI Status Register . . . 201
17.5 LVI Interrupts . . . 201
17.6 Low-Power Modes . . . 201
17.6.1 Wait Mode . . . 201
17.6.2 Stop Mode . . . 201

Chapter 18
Break Module (BRK)

18.1 Introduction . . . 203
18.2 Features . . . 203
18.3 Functional Description . . . 204
18.3.1 Flag Protection During Break Interrupts . . . 204
18.3.2 CPU During Break Interrupts . . . 204
18.3.3 TIMI and TIM2 During Break Interrupts . . . 204
18.3.4 COP During Break Interrupts . . . 205
18.4 Low-Power Modes . . . 205
18.4.1 Wait Mode . . . 205
18.4.2 Stop Mode . . . 205
18.5 Break Module Registers . . . 205
18.5.1 Break Status and Control Register . . . 205
18.5.2 Break Address Registers . . . 206
18.5.3 SIM Break Status Register . . . 206
18.5.4 SIM Break Flag Control Register . . . 207

Chapter 19
Electrical Specifications

19.1 Introduction . . . 209
19.2 Absolute Maximum Ratings . . . 209
19.3 Functional Operating Range . . . 210
19.4 Thermal Characteristics . . . 210
19.5 DC Electrical Characteristics . . . 211
19.6 Control Timing . . . 212
19.7 Internal RC Clock Timing . . . 212
19.8 Crystal Oscillator Characteristics . . . 213
19.9 USB DC Electrical Characteristic . . . 213
19.10 Timer Interface Module Characteristics . . . 214
19.11 FLASH Program/Erase Timing . . . 214
19.12 CGM Electrical Specifications . . . 214
19.13 5.0V SPI Characteristics . . . 215

Chapter 20
Ordering Information and Mechanical Specifications

20.1 Introduction . . . 219
20.2 Ordering Information . . . 219
20.3 Package Dimensions . . . 219

Chapter 1
General Description

1.1 Introduction

The MC68HC908JW32 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 of the MC68HC908JW32 include:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- 8-MHz internal bus frequency
- 88-kHz internal RC clock for timebase wakeup
- 32-Kbytes of on-chip FLASH memory with security[1]
- 1-Kbytes of on-chip random-access memory (RAM)
- On-chip programming firmware for use with host PC computer
- Clock generation module (CGM)
- Up to 29 general-purpose 5V input/output (I/O) pins, including:
  - Keyboard interrupts on 8 pins
  - Direct drive for normal LED on 3 pins
  - High current drive for PS/2 connection on 2 pins (with USB module disabled)
- Serial peripheral interface module (SPI)
- PS2 clock generator module
- 16-bit, 2-channel timer interface module (TIM) with selectable rising and falling edges input capture, output compare, PWM capability on each channel, and external clock input option
- Full universal serial bus (USB) specification 2.0 full-speed functions:
  - 12 Mbps data rate
  - On-chip 3.3V regulator
  - Endpoint 0 with 8-byte transmit buffer and 8-byte receive buffer
  - 64 bytes endpoint buffer to share among endpoints 1–4
- System protection features:
  - Optional computer operating properly (COP) reset
  - Optional low-voltage detection with reset
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- Low-power design (fully static with stop and wait modes)
- Master reset pin with internal pull-up and power-on reset

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

- External asynchronous interrupt pin with internal pull-up ($\overline{IRQ}$)
- 48-pin quad flat non-leaded package (QFN)

1.3 MCU Block Diagram

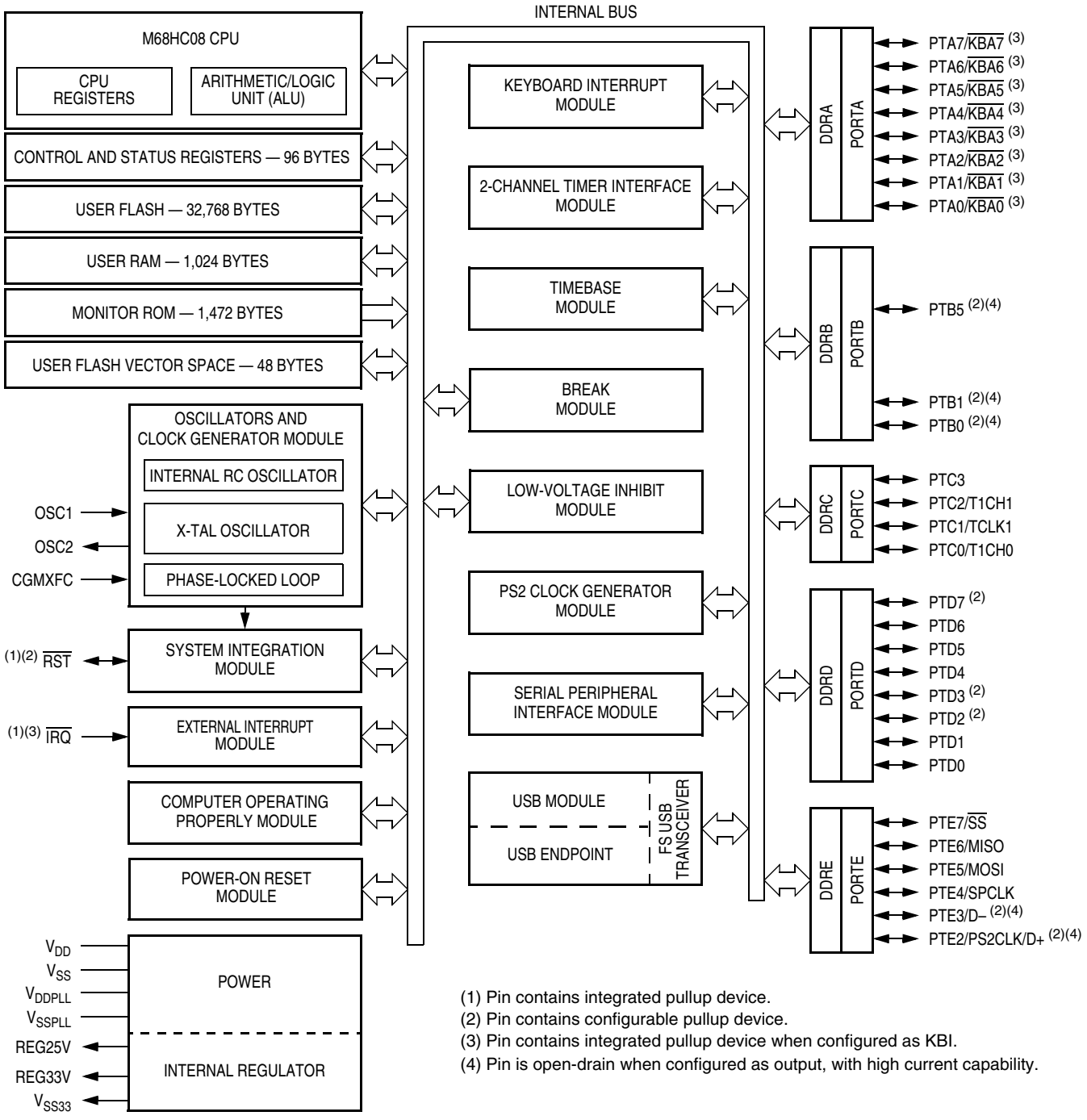


Figure 1-1. MC68HC908JW32 Block Diagram

1.4 Pin Assignments

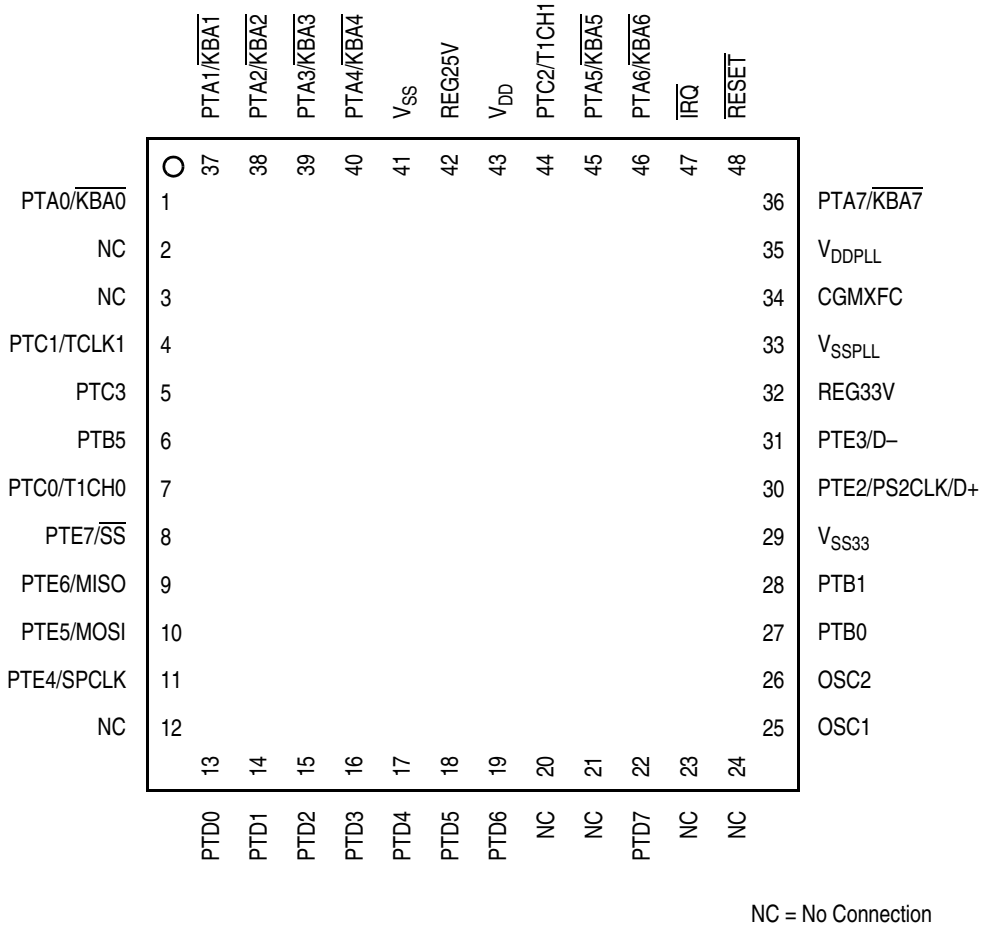


Figure 1-2. 48-Pin QFN Pin Assignment

1.5 Clock Tree

Figure 1-3 shows the clock tree diagram for the MC68HC908JW32.

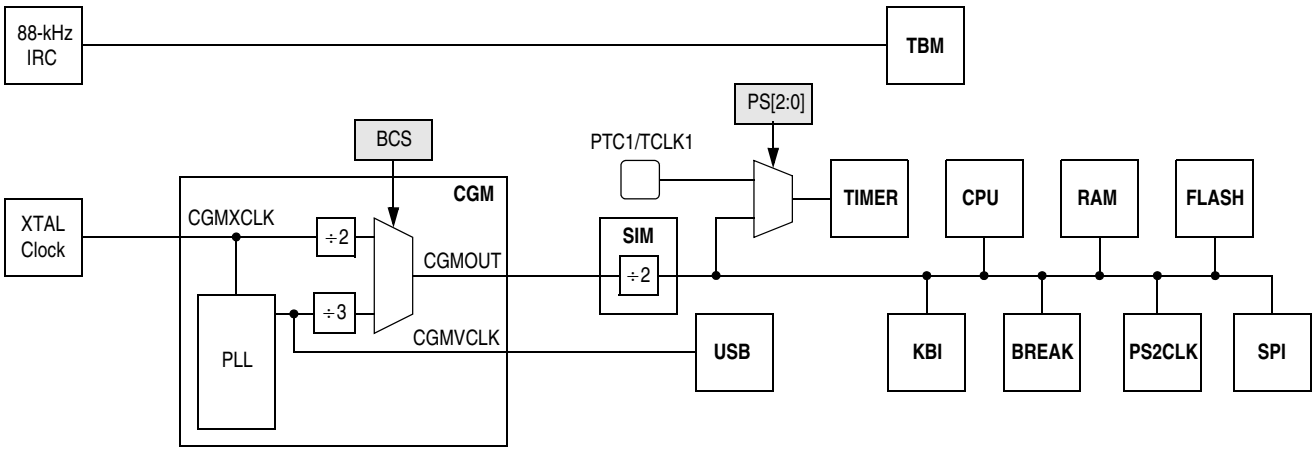


Figure 1-3. Clock Tree Diagram

1.6 Power Management

Figure 1-4 shows the power management diagram for MC68HC908JW32.

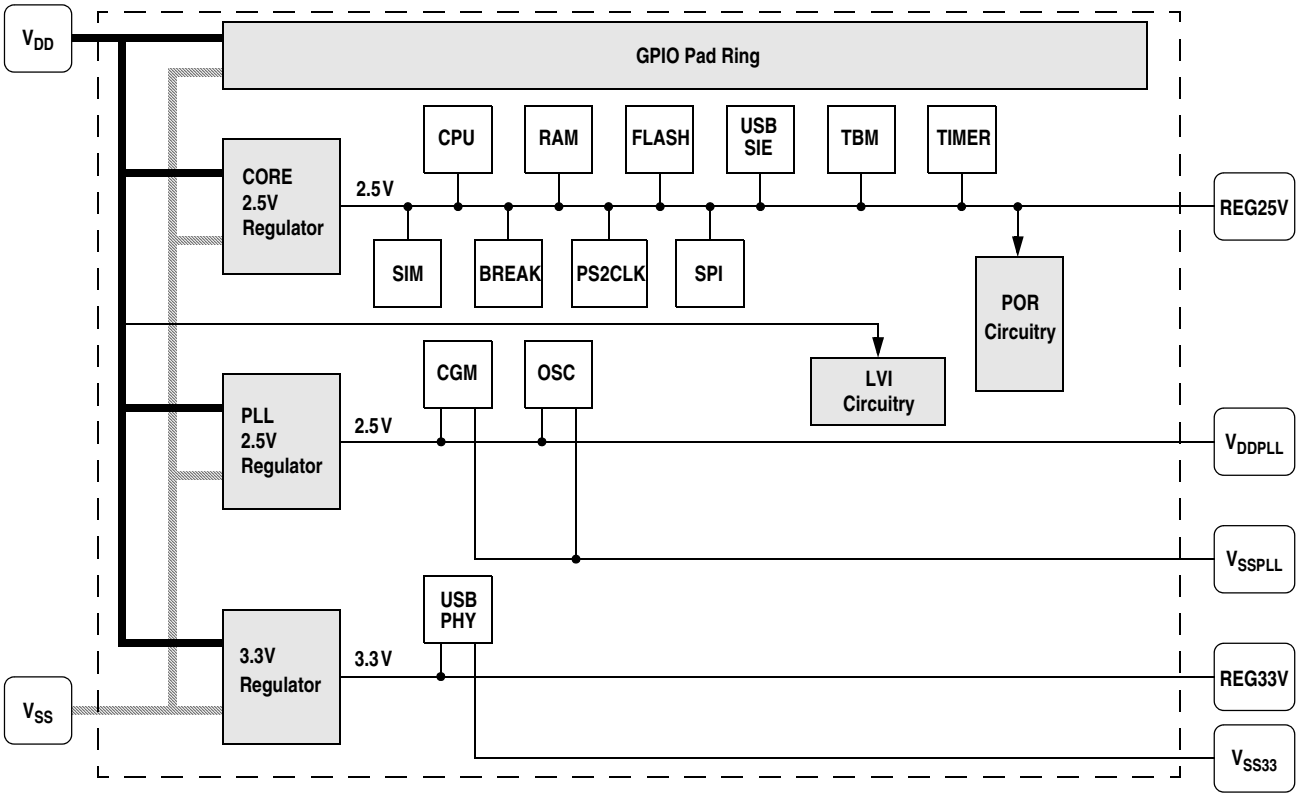


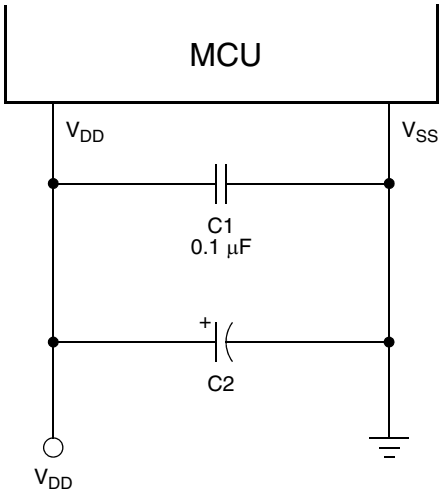
Figure 1-4. Power Management Diagram

1.7 Pin Function

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

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

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as Figure 1-5 shows. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency-response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.



NOTE: Component values shown represent typical applications.

Figure 1-5. Power Supply Bypassing

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

1.7.2 Analogy Power Supply (V_{DDPLL} and V_{SSPLL})

V_{DDPLL} is the internal voltage regulator supply for the CGM module of the device. It is recommended that a decoupling capacitor be connected between the V_{DDPLL} and V_{SSPLL} pins placing it as close to the pins as possible.

1.7.3 Internal Voltage Regulator Supply (REG25V, REG33V, and V_{SS33})

VREG25 is the internal core voltage regulator supply. VREG33 and VSS33 are the internal USB voltage regulator supply.

1.7.4 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit.

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

A logic 0 on the $\overline{\text{RST}}$ pin forces the MCU to a known start-up state. $\overline{\text{RST}}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. A schmitt-trigger and a spike filter is associated with this pin so that the device is more robust to EMC noise.This pin also contains an internal pullup resistor.

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

$\overline{\text{IRQ}}$ is an asynchronous external interrupt pin. This pin contains an internal pullup resistor.

1.7.7 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter connection for the on-chip PLL.

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

PTA7–PTA0 are special function, bidirectional ports pins. These pins are shared with KBI module.

1.7.9 Port B Input/Output (I/O) Pins (PTB5, PTB1, PTB0)

PTB5, PTB1–PTB0 are special function, bidirectional ports pins. These pins can be programmable as open-drain output with high current sourcing capability and has built-in programmable pull up resistor.

1.7.10 Port C Input/Output (I/O) Pins (PTC3–PTC0)

PTC0–PTC3 are bidirectional ports pins. PTC0–PTC2 are shared with TIMER channel 0, channel 1 and TCLK1 pins respectively.

1.7.11 Port D Input/Output (I/O) Pins (PTD7–PTD0)

PTD7–PTD0 are bidirectional ports pins. Pullup option are associated with PTD2, 3 and 7. The option is default enabled after reset.

1.7.12 Port E Input/Output (I/O) Pins (PTE7–PTE2)

PTE7–PTE2 are special function, bidirectional ports pins. PTE2–PTE3 are shared with USB 2.0 FS module. PTE2 is shared with PS2 clock module. PTE4–PTE7 are shared with SPI module.

Chapter 2
Memory

2.1 Introduction

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

- 32,768 bytes of user FLASH
- 1,024 bytes of RAM
- 64 bytes of USB buffer RAM
- 48 bytes of user-defined vectors
- 1,472 bytes of monitor ROM

2.2 Input/Output I/O Section

Addresses $0000–$005F, shown in Figure 2-2**,** contain most of the control, status, and data registers. Additional I/O registers have these addresses:

- $1090; PLL control registers, PTCL
- $1091; PLL bandwidth control register, PBWC
- $1092; PLL multiplier select register high, PMSH
- $1093; PLL multiplier select register low, PMSL
- $1094; PLL VCO range select register, PMRS
- $1095; PLL Reference divider select register, PMDS
- $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 2, INT3
- $FE07; Reserved
- $FE08; FLASH control register, FLCR
- $FE09; FLASH block protect register, FLBPR
- $FE0A; Reserved
- $FE0B; Reserved
- $FE0C; Break Address Register High, BRKH
- $FE0D; Break Address Register Low, BRKL
- $FE0E; Break status and control register, BRKSCR
- $FFFF; COP control register, COPCTL

2.3 Monitor ROM

The 1024 bytes at addresses $FA00–$FDFF and 448 bytes at addresses $FE10–$FFCF are reserved ROM addresses that contain the instructions for the monitor functions. (See Chapter 7 Monitor ROM (MON).)

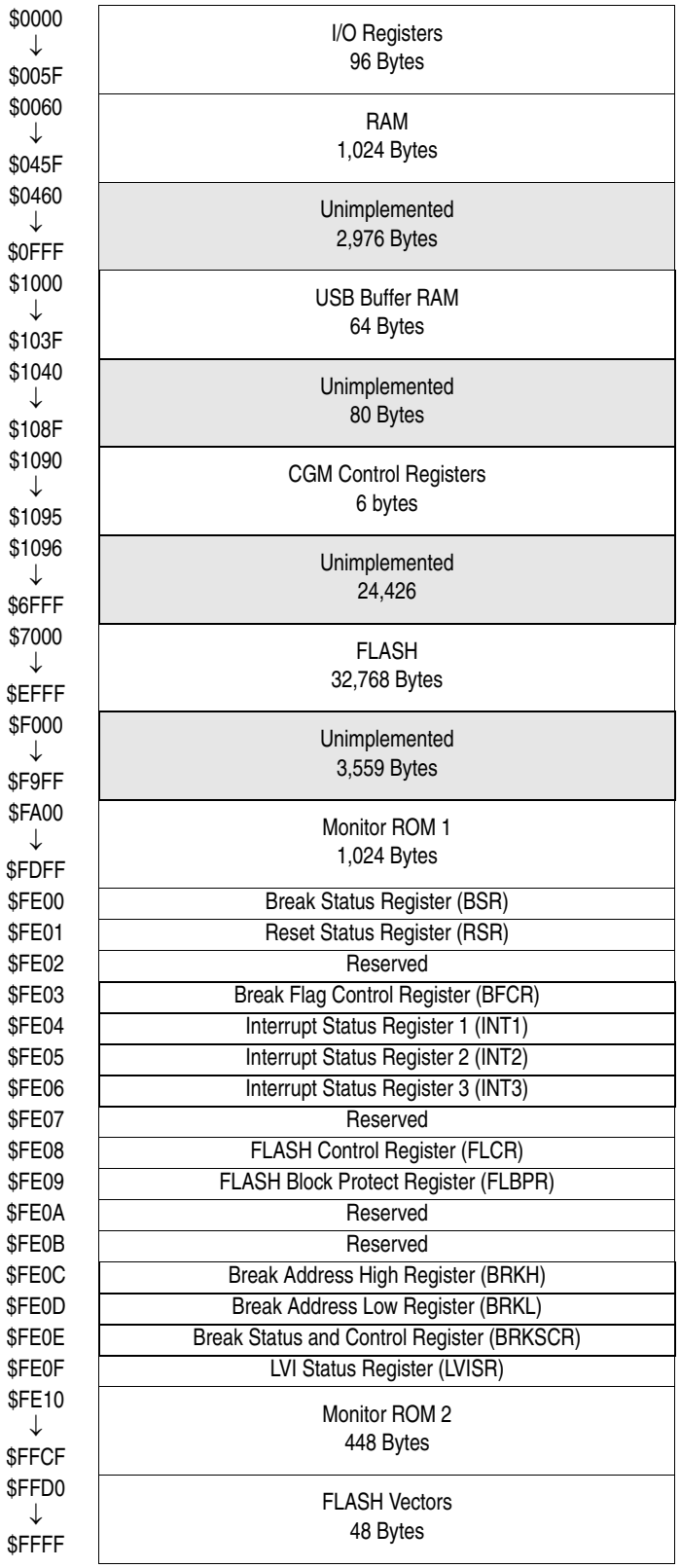


Figure 2-1. Memory Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read:			PTB5				PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read:					PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Write:								
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0*	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read:			DDRB5				DDRB1	DDRB0
		Write:								
		Reset:	0*	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC)	Read:					DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0008	Port E Data Register (PTE)	Read:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2		
		Write:								
		Reset:	Unaffected by reset							
$0009	Data Direction Register E (DDRE)	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2		
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000A	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
$000B	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$000C	Timer 1 Counter Register High (T1CNTH)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected by reset

Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000D	Timer 1 Counter Register Low (T1CNTL)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$000E	Timer 1 Counter Modulo Register High (T1MODH)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$000F	Timer 1 Counter Modulo Register Low (T1MODL)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	1	1	1	1	1	1	1	1
$0010	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
$0011	Timer 1 Channel 0 Register High (T1CH0H)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	Indeterminate after reset							
$0012	Timer 1 Channel 0 Register Low (T1CH0L)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	Indeterminate after reset							
$0013	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
$0014	Timer 1 Channel 1 Register High (T1CH1H)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	Indeterminate after reset							
$0015	Timer 1 Channel 1 Register Low (T1CH1L)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		Write:								
		Reset:	Indeterminate after reset							
$0016	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
$0017	Keyboard Interrupt Enable Register (KBIER)	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	Timebase Control Register (TBCR)	Read:	TBIF	TBR2	TBR1	TBR0	0	TBIE	TBON	R
		Write:					TACK			
		Reset:	0	0	0	0	0	0	0	0
$0019	PS2 Clock Generator Control and Satus Register (PS2CSR)	Read:	PSTATUS	PS2IF	PRE	CSEL1	CSEL0	PS2IEN	CLKEN	PS2EN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
			= Unimplemented			R	= Reserved		U = Unaffected by reset	

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	Port Option Control Register 1 (POCR1)	Read:			LEDB5				LEDB1	LEDB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001B	Port Option Control Register 2 (POCR2)	Read:	0	0	PTD7PD	PTD3PD	PTD2PD	DPPULLEN	PTE3P	PTE2P
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001C	IRQ Option Control Register (IOCR)	Read:	0	0	0	0	0	PTE3IF	PTE3IE	IRQPD
		Write:								
		Reset:								
$001D	Configuration Register 2 (CONFIG2)	Read:			STOP_ XCLKEN	STOP_RC CLKEN		R	VREG33D	URSTD
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001E	IRQ Status and Control Register (ISCR)	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001F	Configuration Register (CONFIG)†	Read:	COPRS	LVISTOP	LVIRSTD	LVIPWRD		SSREC	STOP	COPD
		Write:								
		Reset:	0	0	0	0	0	0	0	0
† One-time writable register after each reset.										
$0020 to $003D	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$003E	Pullup Control Register (PULLCR)	Read:			PULL5EN				PULL1EN	PULL0EN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$003F	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$0040	USB Endpoint 0 Data Register 0 (UE0D0)	Read:	UE0D07_OUT	UE0D06_OUT	UE0D05_OUT	UE0D04_OUT	UE0D03_OUT	UE0D02_OUT	UE0D01_OUT	UE0D00_OUT
		Write:	UE0D07_IN	UE0D06_IN	UE0D05_IN	UE0D04_IN	UE0D03_IN	UE0D02_IN	UE0D01_IN	UE0D00_IN
		Reset:	Unaffected by reset							
$0041	USB Endpoint 0 Data Register 1 (UE0D1)	Read:	UE0D17_OUT	UE0D16_OUT	UE0D15_OUT	UE0D14_OUT	UE0D13_OUT	UE0D12_OUT	UE0D11_OUT	UE0D10_OUT
		Write:	UE0D17_IN	UE0D16_IN	UE0D15_IN	UE0D14_IN	UE0D13_IN	UE0D12_IN	UE0D11_IN	UE0D10_IN
		Reset:	Unaffected by reset							
$0042	USB Endpoint 0 Data Register 2 (UE0D2)	Read:	UE0D27_OUT	UE0D26_OUT	UE0D25_OUT	UE0D24_OUT	UE0D23_OUT	UE0D22_OUT	UE0D21_OUT	UE0D20_OUT
		Write:	UE0D27_IN	UE0D26_IN	UE0D25_IN	UE0D24_IN	UE0D23_IN	UE0D22_IN	UE0D21_IN	UE0D20_IN
		Reset:	Unaffected by reset							
$0043	USB Endpoint 0 Data Register 3 (UE0D3)	Read:	UE0D37_OUT	UE0D36_OUT	UE0D35_OUT	UE0D34_OUT	UE0D33_OUT	UE0D32_OUT	UE0D31_OUT	UE0D30_OUT
		Write:	UE0D37_IN	UE0D36_IN	UE0D35_IN	UE0D34_IN	UE0D33_IN	UE0D32_IN	UE0D31_IN	UE0D30_IN
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved U = Unaffected by reset

Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0044	USB Endpoint 0 Data Register 4 (UE0D4)	Read:	UE0D47_OUT	UE0D46_OUT	UE0D45_OUT	UE0D44_OUT	UE0D43_OUT	UE0D42_OUT	UE0D41_OUT	UE0D40_OUT
		Write:	UE0D47_IN	UE0D46_IN	UE0D45_IN	UE0D44_IN	UE0D43_IN	UE0D42_IN	UE0D41_IN	UE0D40_IN
		Reset:	Unaffected by reset							
$0045	USB Endpoint 0 Data Register 5 (UE0D5)	Read:	UE0D57_OUT	UE0D56_OUT	UE0D55_OUT	UE0D54_OUT	UE0D53_OUT	UE0D52_OUT	UE0D51_OUT	UE0D50_OUT
		Write:	UE0D57_IN	UE0D56_IN	UE0D55_IN	UE0D54_IN	UE0D53_IN	UE0D52_IN	UE0D51_IN	UE0D50_IN
		Reset:	Unaffected by reset							
$0046	USB Endpoint 0 Data Register 6 (UE0D6)	Read:	UE0D67_OUT	UE0D66_OUT	UE0D65_OUT	UE0D64_OUT	UE0D63_OUT	UE0D62_OUT	UE0D61_OUT	UE0D60_OUT
		Write:	UE0D67_IN	UE0D66_IN	UE0D65_IN	UE0D64_IN	UE0D63_IN	UE0D62_IN	UE0D61_IN	UE0D60_IN
		Reset:	Unaffected by reset							
$0047	USB Endpoint 0 Data Register 7 (UE0D7)	Read:	UE0D77_OUT	UE0D76_OUT	UE0D75_OUT	UE0D74_OUT	UE0D73_OUT	UE0D72_OUT	UE0D71_OUT	UE0D70_OUT
		Write:	UE0D77_IN	UE0D76_IN	UE0D75_IN	UE0D74_IN	UE0D73_IN	UE0D72_IN	UE0D71_IN	UE0D70_IN
		Reset:	Unaffected by reset							
$0048	Unimplemented	Read:								
		Write:								
$0049	Unimplemented	Read:								
		Write:								
$004A	Unimplemented	Read:								
		Write:								
$004B	Unimplemented	Read:								
		Write:								
		Reset:								
$004C	SPI Control Register (SPCR)	Read:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:								
		Reset:	0	0	1	0	1	0	0	0
$004D	SPI Status and Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$004E	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$004F	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$0050	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								

= Unimplemented R = Reserved U = Unaffected by reset

Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0051	USB Control Register (USBCR)	Read:	USBEN	USBCLKEN	TFC4IE	TFC3IE	TFC2IE	TFC1IE	TFC0IE	0
		Write:								RESUME
		Reset:	0	0	0	0	0	0	0	0
$0052	USB Status Register (USBSR)	Read:	CONFIG		SETUP	SOF	CONFIG_CHG	USBRST	RESUMEF	SUSPND
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0053	USB Status Interrupt Mask Register (USIMR)	Read:		0	SETUPIE	SOFIE	CONFIG_ CHGIE	USBRE-SETIE	RESUME-FIE	SUSPNDIE
		Write:		EP0_STALL						
		Reset:	0	0	0	0	0	0	0	0
$0054	USB EPO Control/Status Register (UEP0CSR)	Read:	DSIZE3_OUT	DSIZE2_OUT	DSIZE1_OUT	DSIZE0_OUT	DVALID_IN	TFRC_IN	DVALID_OUT	TFRC_OUT
		Write:	DSIZE3_IN	DSIZE2_IN	DSIZE1_IN	DSIZE0_IN				
		Reset:	0	0	0	0	0	0	0	0
$0055	USB EP1 Control/Status Register (UEP1CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0056	USB EP2 Control/Status Register (UEP2CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0057	USB EP3 Control/Status Register (UEP3CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0058	USB EP4 Control/Status Register (UEP4CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0059	USB EP1 Data Size Register (UEP1DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005A	USB EP2 Data Size Register (UEP2DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005B	USB EP3 Data Size Register (UEP3DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005C	USB EP4 Data Size Register (UEP4DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005D	USB EP 1/2 Base Pointer Register (UEP12BPR)	Read:		BASE22	BASE21	BASE20		BASE12	BASE11	BASE10
		Write:								
		Reset:	0	0	0	0	0	0	0	0
			= Unimplemented			R	= Reserved		U = Unaffected by reset	

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 7)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$005E	USB EP 3/4 Base Pointer Register (UEP34BPR)	Read:		BASE42	BASE41	BASE40		BASE32	BASE31	BASE30
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005F	USB Interface Control Register (UINTFCR)	Read:		EP4INT		EP3INT		EP2INT		EP1INT
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$1090	PLL Control Register (PTCL)	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		Write:								
		Reset:	0	0	1	0	0	0	0	0
$1091	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	
$1092	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
$1093	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
$1094	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
$1095	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
$FE00	Break Status Register (BSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							See note	
		Reset:							0	
$FE01	Reset Status Register (RSR)	Read:	POR	PIN	COP	ILOP	ILAD	USB	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	1	0
$FE02	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$FE03	Break Flag Control Register (BFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							

Note: Writing a logic 0 clears SBSW.

= Unimplemented R = Reserved U = Unaffected by reset

Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 7)

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:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE06	Interrupt Status Register 3 (INT3)	Read:	0	0	0	0	0	0	0	IF15
		Write:	R	R	R	R	R	R	R	R
			0	0	0	0	0	0	0	0
$FE07	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$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	FLASH Block Protect Register (FLBPR)	Read:	BPR7	BPR6	BPR5	BPR4	BPR3	BPR2	BPR1	BPR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0A	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$FE0B	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
$FE0C	Break Address High Register (BRKH)	Read:	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address low Register (BRKL)	Read:	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		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	LVI Sttatusl Register (LVISR)	Read:	LVIOUT	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FFFF	COP Control Register (COPCTL)	Read:	Low byte of reset vector							
		Write:	Writing clears COP counter (any value)							
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved U = Unaffected by reset

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

Table 2-1 is a list of vector locations.

Table 2-1. Vector Addresses

Vector	Vector	Address	Vector
Lowest	IF15	$FFDE	Timebase Vector (High)
		$FFDF	Timebase Vector (Low)
	IF14	$FFE0	Keyboard Vector (High)
		$FFE1	Keyboard Vector (Low)
	IF13	$FFE2	SPI Transmit Vector (High)
		$FFE3	SPI Transmit Vector (Low)
	IF12	$FFE4	SPI Receive Vector (High)
		$FFE5	SPI Receive Vector (Low)
	IF11	$FFE6	Reserved
		$FFE7	Reserved
	IF10	$FFE8	Reserved
		$FFE9	Reserved
	IF9	$FFEA	Reserved
		$FFEB	Reserved
	IF8	$FFEC	PS2 Interrupt Vector (High)
		$FFED	PS2 Interrupt Vector (Low)
	IF7	$FFEE	Timer 1 Overflow Vector (High)
		$FFEF	Timer 1 Overflow Vector (Low)
	IF6	$FFF0	Timer 1 Channel 1 Vector (High)
		$FFF1	Timer 1 Channel 1 Vector (Low)
	IF5	$FFF2	Timer 1 Channel 0 Vector (High)
		$FFF3	Timer 1 Channel 0 Vector (Low)
	IF4	$FFF4	PLL Vector (High)
		$FFF5	PLL Vector (Low)
	IF3	$FFF6	IRQ Vector (High)
		$FFF7	IRQ Vector (Low)
	IF2	$FFF8	USB Endpoint Vector (High)
		$FFF9	USB Endpoint Vector (Low)
	IF1	$FFFA	USB System Vector (High)
		$FFFB	USB System Vector (Low)
	—	$FFFC	SWI Vector (High)
		$FFFD	SWI Vector (Low)
	—	$FFFE	Reset Vector (High)
Highest		$FFFF	Reset Vector (Low)

2.4 Random-Access Memory (RAM)

Addresses $0060 through $045F are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64k-byte memory space.

NOTE

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

Within page zero are 160 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 out of page zero, direct addressing mode instructions can efficiently access 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. This memory can be read, programmed, and erased from a single external supply. The program and erase operations are enabled through the use of an internal charge pump.

2.5.1 Functional Description

The FLASH memory consists of an array of 32,768 bytes for user memory plus a block of 48 bytes for user interrupt vectors and one byte for the mask option register. *An erased bit reads as logic 1 and a programmed bit reads as a logic 0.* The FLASH memory page size is defined as 512 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:

- $7000–$EFFF; user memory, 32,768 bytes
- $FFD0–$FFFF; user interrupt vectors, 48 bytes

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

NOTE

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

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

2.5.2 FLASH Control Register

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

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

2.5.3 FLASH Page Erase Operation

Use the following procedure to erase a page of FLASH memory. A page consists of 512 consecutive bytes starting from addresses $X000, $X200, $X400, $X600, $X800, $XA00, $XC00 or $XE00. *The 48-byte user interrupt vectors cannot be erased by the page erase operation because of security reasons. Mass erase is required to erase this page.*

1. Set the ERASE bit and clear the MASS bit in the FLASH control register.
2. Write any data to any FLASH location within the page address range desired.
3. Wait for a time, t_{nvs} (5 μs).
4. Set the HVEN bit.
5. Wait for a time t_{erase} (20 ms).
6. Clear the ERASE bit.

7. Wait for a time, t_{nvh} (5 μs).
8. Clear the HVEN bit
9. After time, t_{rcv} (1 μs), the memory can be accessed in read mode again.

NOTE

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

2.5.4 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. Write any data to any FLASH location within the FLASH memory address range.
3. Wait for a time, t_{nvs} (5 μs).
4. Set the HVEN bit.
5. Wait for a time t_{me} (200 ms).
6. Clear the ERASE bit.
7. Wait for a time, t_{nvh1} (100 μs).
8. Clear the HVEN bit.
9. After time, t_{rcv} (1 μs), the memory can be accessed in read mode again.

NOTE

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

2.5.5 FLASH Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 64 consecutive bytes starting from addresses $XX00, $XX40, $XX800 or $XXC00. Use the following procedure to program a row of FLASH memory. (Figure 2-4 shows a flowchart of the programming algorithm.)

1. Set the PGM bit. This configures the memory for program operation and enables the latching of address and data for programming.
2. Write any data to any FLASH location within the address range of the row to be programmed.
3. Wait for a time, t_{nvs} (5 μs).
4. Set the HVEN bit.
5. Wait for a time, t_{pgs} (10 μs).
6. Write data to the FLASH location to be programmed.
7. Wait for time, t_{prog} (20 μs to 40 μs).
8. Repeat steps 6 and 7 until all bytes within the row are programmed.
9. Clear the PGM bit.
10. Wait for time, t_{nvh} (5 μs).
11. Clear the HVEN bit.
12. After time, t_{rcv} (1 μs), the memory can be accessed in read mode again.

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

NOTE

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

NOTE

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

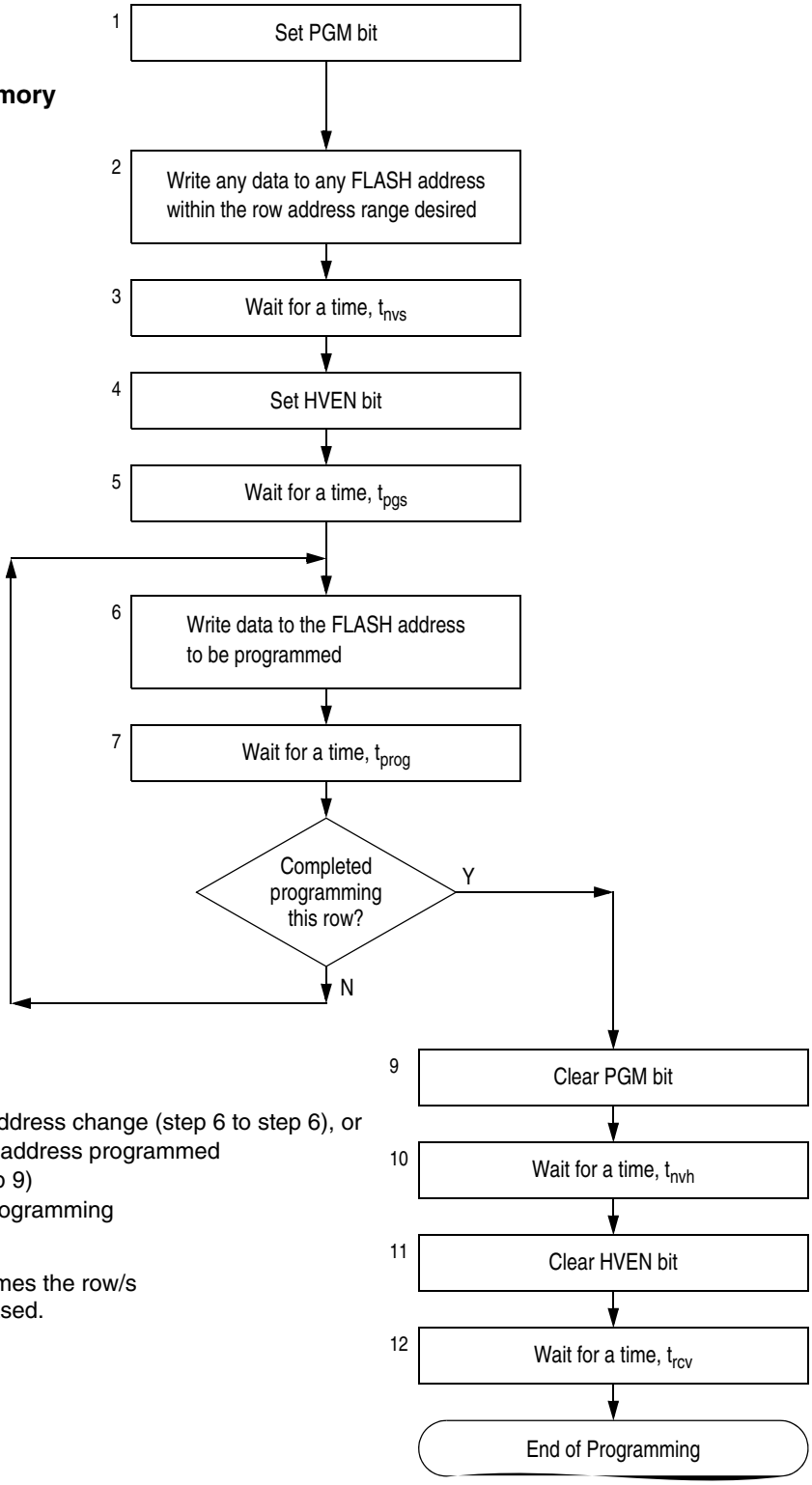
2.5.6 FLASH Protection

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

The 48 bytes of user interrupt vectors ($FFD0–$FFFF) are always protected, regardless of the value in the FLASH block protect register. A mass erase is required to erase these locations.

Algorithm for programming a row (64 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

2.5.7 FLASH Block Protect Register

The FLASH block protect register is implemented as an 8-bit I/O register. The value in this register determines the starting address of the protected range within the FLASH memory.

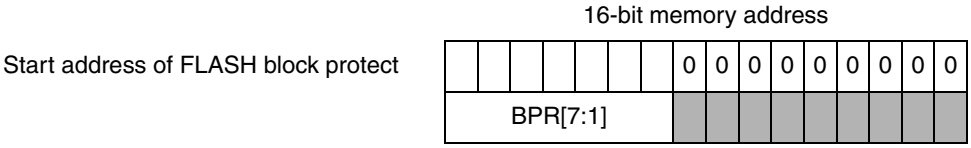
Address: $FE09

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

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

BPR[7:0] — FLASH Block Protect Bits

BPR[7:1] represent bits [15:9] of a 16-bit memory address. Bits [8:0] are logic 0's.



BPR0 is used only for BPR[7:0] = $FF, for no block protection.

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 X000, X200, etc. (at page boundaries — 512 bytes) within the FLASH memory.

Examples of protect start address:

Table 2-2 FLASH Block Protect Range

BPR[7:0]	Protected Range
$00–$70	The entire FLASH memory is protected.
$70 or $71 (**0111 000x**)	$7000 to $FFFF (The entire FLASH memory is protected.)
$72 or $73 (**0111 001x**)	$7200 to $FFFF
$74 or $75 (**0111 010x**)	$7400 to $FFFF
and so on...	
$EE or $EF (**1110 111x**)	$EE00 to $FFFF
$F0 - $FF	The entire FLASH memory is **NOT** protected.(1)

1. The 48-byte user vectors ($FFD0–$FFFF), which are always protected.

Chapter 3
Configuration Registers (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 (262,128 or 8176 CGMRCLK cycles)
- 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 CGMXCLK cycles or 4096 CGMXCLK cycles)
- Oscillator during stop mode

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001D	Configuration Register 2 (CONFIG2)	Read: Write:			STOP_ XCLKEN	STOP_RC CLKEN		R	VREG33D	URSTD
		Reset:	0	0	0	0	0	0	0	0
$001F	Configuration Register 1 (CONFIG1)†	Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD		SSREC	STOP	COPD
		Reset:	0	0	0	0	0	0	0	0

† One-time writable register after each reset.

= Unimplemented

Figure 3-1. CONFIG Registers Summary

3.2 Functional Description

The configuration registers are used in the initialization of various options. The configuration register (CONFIG1) can be written once after each reset but CONFIG2 register can perform multiple write. 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.

3.3 Configuration Register 1 (CONFIG1)

Address:	$001F							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD		SSREC	STOP	COPD
Reset:	0	0	0	0	0	0	0	0
	= Unimplemented							

Figure 3-2. Configuration Register 1 (CONFIG1)

COPRS — COP Rate Select

COPRS selects the COP time-out period. Reset clears COPRS. (See Chapter 16 Computer Operating Properly (COP).)

1 = COP time out period = 8176 CGMRCLK cycles
0 = COP time out period = 262,128 CGMRCLK cycles

LVISTOP — Low Voltage Inhibit Enable in STOP mode bit

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

1 = Low voltage inhibit enabled during stop mode
0 = Low voltage inhibit disable during stop mode

LVIRSTD — LVI Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module.

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

LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module completely. When it is set, LVI trip for V_{DD} is disabled.

1 = LVI module power and LVI trip for V_{DD} disabled
0 = LVI module power and LVI trip for V_{DD} is enabled

SSREC — Short Stop Recovery

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

NOTE

Exiting stop mode by pulling reset will result in the long stop recovery.

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

NOTE

When the LVISTOP is enabled, the system stabilization time for power on reset and long stop recovery (both 4096 CGMXCLK cycles) gives a delay longer than the enable time for the LVI. There is no period where the MCU is not protected from a low power condition. However, when using the short stop recovery configuration option, the 32 CGMXCLK delay is less than the LVI's turn-on time and there exists a period in start-up where the LVI is not protecting the MCU.

STOP — STOP Instruction Enable

STOP enables the STOP instruction.

1 = STOP instruction enabled

0 = STOP instruction treated as illegal opcode

COPD — COP Disable Bit

COPD disables the COP module. (See Chapter 16 Computer Operating Properly (COP).)

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	STOP_RC CLKEN		R	VREG33D	URSTD
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

†† Reset by POR only.

Figure 3-3. Configuration Register 2 (CONFIG2)

STOP_XCLKEN — Crystal Oscillator Stop Mode Enable

Setting STOP_XCLKEN enables the external crystal (XTAL) oscillator to continue operating during stop mode, in the other words, SIMOSCEN hold high during STOP mode. When this bit is cleared, the external XTAL oscillator will be disabled during stop mode. Reset clears this bit.

1 = XTAL oscillator enabled during stop mode

0 = XTAL oscillator disabled during stop mode

STOP_RCCLKEN — RC clock Stop Mode Enable

Setting STOP_RCCLKEN enables the internal RC clock to continue operating during STOP mode. When this bit is cleared, the internal RC clock will be disabled during STOP mode. Reset clears this bit.

1 = Internal RC clock enabled during stop mode

0 = Internal RC clock disable during stop mode

VREG33D — 3.3V USB Regulator Disable Bit

VREG33D disables the USB 3.3V regulator completely.

1 = VREG33 regulator is disabled

0 = VREG33 regulator is enabled

URSTD — USB Reset Disable Bit

URSTD disables the USB reset signal generating an internal reset to the CPU and internal registers. Instead, it will generate an interrupt request to CPU.

1 = USB reset generates a interrupt request to CPU

0 = USB reset generates a chip reset

Chapter 4
Central Processor Unit (CPU)

4.1 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.

4.2 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

4.3 CPU Registers

Figure 4-1 shows the five CPU registers. CPU registers are not part of the memory map.

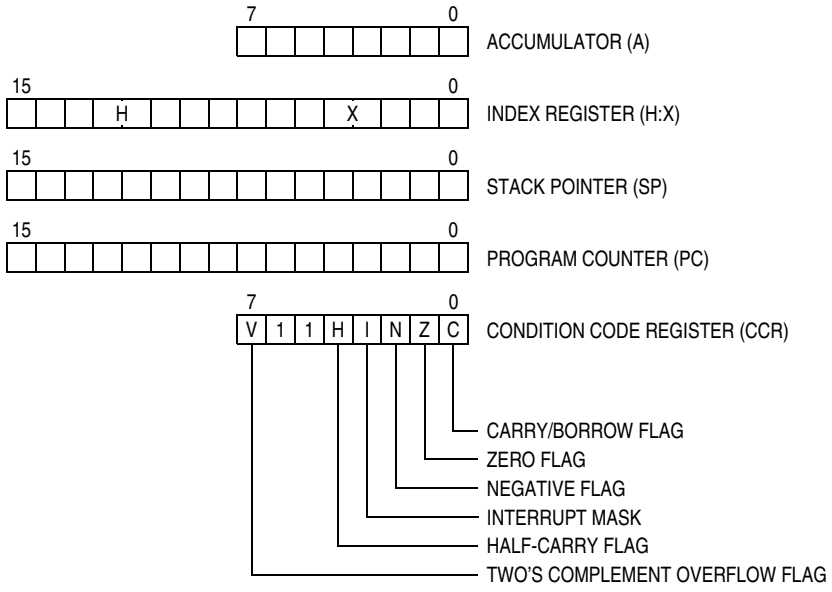


Figure 4-1. CPU Registers

4.3.1 Accumulator

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

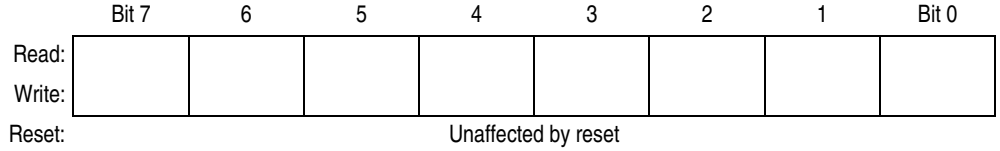


Figure 4-2. Accumulator (A)

4.3.2 Index Register

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

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

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

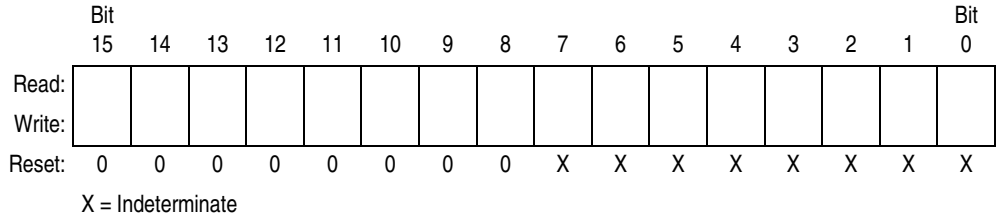


Figure 4-3. Index Register (H:X)

4.3.3 Stack Pointer

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

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

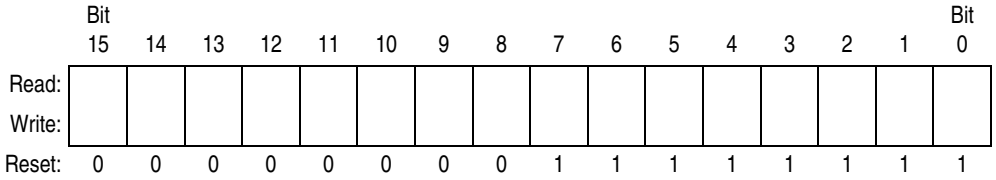


Figure 4-4. Stack Pointer (SP)

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.

4.3.4 Program Counter

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

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

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

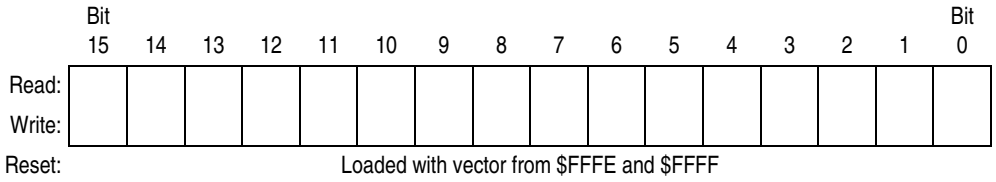


Figure 4-5. Program Counter (PC)

4.3.5 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 4-6. 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

4.4 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.

4.5 Low-Power Modes

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

4.5.1 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

4.5.2 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.

4.6 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.

4.7 Instruction Set Summary

Table 4-1 provides a summary of the M68HC08 instruction set.

Table 4-1. Instruction Set Summary (Sheet 1 of 6)

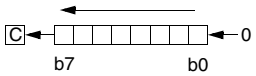
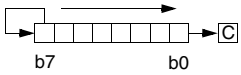
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
ADC #*opr* ADC *opr* ADC *opr* ADC *opr*,X ADC *opr*,X ADC ,X ADC *opr*,SP ADC *opr*,SP	Add with Carry	A ← (A) + (M) + (C)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A9 B9 C9 D9 E9 F9 9EE9 9ED9	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ADD #*opr* ADD *opr* ADD *opr* ADD *opr*,X ADD *opr*,X ADD ,X ADD *opr*,SP ADD *opr*,SP	Add without Carry	A ← (A) + (M)	↕	↕	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	AB BB CB DB EB FB 9EEB 9EDB	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
AIS #*opr*	Add Immediate Value (Signed) to SP	SP ← (SP) + (16 « M)	–	–	–	–	–	–	IMM	A7	ii	2
AIX #*opr*	Add Immediate Value (Signed) to H:X	H:X ← (H:X) + (16 « M)	–	–	–	–	–	–	IMM	AF	ii	2
AND #*opr* AND *opr* AND *opr* AND *opr*,X AND *opr*,X AND ,X AND *opr*,SP AND *opr*,SP	Logical AND	A ← (A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A4 B4 C4 D4 E4 F4 9EE4 9ED4	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X ASL *opr*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
ASR *opr* ASRA ASRX ASR *opr*,X ASR *opr*,X ASR *opr*,SP	Arithmetic Shift Right		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	4 1 1 4 3 5
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + 2 + rel ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BCLR *n*, *opr*	Clear Bit n in M	Mn ← 0	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + 2 + *rel* ? (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *opr*	Branch if Greater Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) = 0	–	–	–	–	–	–	REL	90	rr	3
BGT *opr*	Branch if Greater Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) ǀ (N ⊕ V) = 0	–	–	–	–	–	–	REL	92	rr	3
BHCC *rel*	Branch if Half Carry Bit Clear	PC ← (PC) + 2 + *rel* ? (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS *rel*	Branch if Half Carry Bit Set	PC ← (PC) + 2 + *rel* ? (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI *rel*	Branch if Higher	PC ← (PC) + 2 + *rel* ? (C) ǀ (Z) = 0	–	–	–	–	–	–	REL	22	rr	3

Table 4-1. Instruction Set Summary (Sheet 2 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
BHS *rel*	Branch if Higher or Same (Same as BCC)	PC ← (PC) + 2 + *rel* ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BIH *rel*	Branch if $\overline{\text{IRQ}}$ Pin High	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1	–	–	–	–	–	–	REL	2F	rr	3
BIL *rel*	Branch if $\overline{\text{IRQ}}$ Pin Low	PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0	–	–	–	–	–	–	REL	2E	rr	3
BIT #*opr* BIT *opr* BIT *opr* BIT *opr*,X BIT *opr*,X BIT ,X BIT *opr*,SP BIT *opr*,SP	Bit Test	(A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A5 B5 C5 D5 E5 F5 9EE5 9ED5	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
BLE *opr*	Branch if Less Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (Z) ǀ (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) ǀ (Z) = 1	–	–	–	–	–	–	REL	23	rr	3
BLT *opr*	Branch if Less Than (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) =1	–	–	–	–	–	–	REL	91	rr	3
BMC *rel*	Branch if Interrupt Mask Clear	PC ← (PC) + 2 + *rel* ? (I) = 0	–	–	–	–	–	–	REL	2C	rr	3
BMI *rel*	Branch if Minus	PC ← (PC) + 2 + *rel* ? (N) = 1	–	–	–	–	–	–	REL	2B	rr	3
BMS *rel*	Branch if Interrupt Mask Set	PC ← (PC) + 2 + *rel* ? (I) = 1	–	–	–	–	–	–	REL	2D	rr	3
BNE *rel*	Branch if Not Equal	PC ← (PC) + 2 + *rel* ? (Z) = 0	–	–	–	–	–	–	REL	26	rr	3
BPL *rel*	Branch if Plus	PC ← (PC) + 2 + *rel* ? (N) = 0	–	–	–	–	–	–	REL	2A	rr	3
BRA *rel*	Branch Always	PC ← (PC) + 2 + *rel*	–	–	–	–	–	–	REL	20	rr	3
BRCLR *n,opr,rel*	Branch if Bit *n* in M Clear	PC ← (PC) + 3 + *rel* ? (Mn) = 0	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BRN *rel*	Branch Never	PC ← (PC) + 2	–	–	–	–	–	–	REL	21	rr	3
BRSET *n,opr,rel*	Branch if Bit *n* in M Set	PC ← (PC) + 3 + *rel* ? (Mn) = 1	–	–	–	–	–	↕	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd rr	5 5 5 5 5 5 5 5
BSET *n,opr*	Set Bit *n* in M	Mn ← 1	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr,rel* CBEQA #*opr,rel* CBEQX #*opr,rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2

Table 4-1. Instruction Set Summary (Sheet 3 of 6)

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
CLR *opr* CLRA CLRX CLRH CLR *opr*,X CLR ,X CLR *opr*,SP	Clear	M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X CMP *opr*,SP CMP *opr*,SP	Compare A with M	(A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
COM *opr* COMA COMX COM *opr*,X COM ,X COM *opr*,SP	Complement (One's Complement)	M ← ($\overline{M}$) = $FF – (M) A ← ($\overline{A}$) = $FF – (M) X ← ($\overline{X}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M)	0	–	–	↕	↕	1	DIR INH INH IX1 IX SP1	33 43 53 63 73 9E63	dd ff ff	4 1 1 4 3 5
CPHX #*opr* CPHX *opr*	Compare H:X with M	(H:X) – (M:M + 1)	↕	–	–	↕	↕	↕	IMM DIR	65 75	ii ii+1 dd	3 4
CPX #*opr* CPX *opr* CPX *opr* CPX ,X CPX *opr*,X CPX *opr*,X CPX *opr*,SP CPX *opr*,SP	Compare X with M	(X) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A3 B3 C3 D3 E3 F3 9EE3 9ED3	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
DAA	Decimal Adjust A	$(A)_{10}$	U	–	–	↕	↕	↕	INH	72		2
DBNZ *opr,rel* DBNZA *rel* DBNZX *rel* DBNZ *opr*,X,*rel* DBNZ X,*rel* DBNZ *opr*,SP,*rel*	Decrement and Branch if Not Zero	A ← (A) – 1 or M ← (M) – 1 or X ← (X) – 1 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 3 + *rel* ? (result) ≠ 0 PC ← (PC) + 2 + *rel* ? (result) ≠ 0 PC ← (PC) + 4 + *rel* ? (result) ≠ 0	–	–	–	–	–	–	DIR INH INH IX1 IX SP1	3B 4B 5B 6B 7B 9E6B	dd rr rr rr ff rr rr ff rr	5 3 3 5 4 6
DEC *opr* DECA DECX DEC *opr*,X DEC ,X DEC *opr*,SP	Decrement	M ← (M) – 1 A ← (A) – 1 X ← (X) – 1 M ← (M) – 1 M ← (M) – 1 M ← (M) – 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3A 4A 5A 6A 7A 9E6A	dd ff ff	4 1 1 4 3 5
DIV	Divide	A ← (H:A)/(X) H ← Remainder	–	–	–	–	↕	↕	INH	52		7
EOR #*opr* EOR *opr* EOR *opr* EOR *opr*,X EOR *opr*,X EOR ,X EOR *opr*,SP EOR *opr*,SP	Exclusive OR M with A	A ← (A ⊕ M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A8 B8 C8 D8 E8 F8 9EE8 9ED8	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
INC *opr* INCA INCX INC *opr*,X INC ,X INC *opr*,SP	Increment	M ← (M) + 1 A ← (A) + 1 X ← (X) + 1 M ← (M) + 1 M ← (M) + 1 M ← (M) + 1	↕	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3C 4C 5C 6C 7C 9E6C	dd ff ff	4 1 1 4 3 5

Table 4-1. Instruction Set Summary (Sheet 4 of 6)

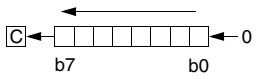
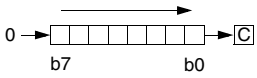
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
JMP *opr* JMP *opr* JMP *opr*,X JMP *opr*,X JMP ,X	Jump	PC ← Jump Address	–	–	–	–	–	–	DIR EXT IX2 IX1 IX	BC CC DC EC FC	dd hh ll ee ff ff	2 3 4 3 2
JSR *opr* JSR *opr* JSR *opr*,X JSR *opr*,X JSR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

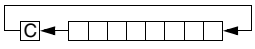
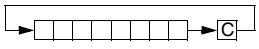
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULA	Pull A from Stack	SP ← (SP + 1); Pull (A)	–	–	–	–	–	–	INH	86		2
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	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 4-1. 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

4.8 Opcode Map

See Table 4-2.

Table 4-2. Opcode Map

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

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

*Pre-byte for stack pointer indexed instructions

MSB / LSB	0
Low Byte of Opcode in Hexadecimal	0

High Byte of Opcode in Hexadecimal

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

Chapter 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 divided phase-locked loop (PLL) clock, CGMVCLK, divided by three. 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 crystal (4MHz) to generate a base frequency and dividing to a maximum bus frequency of 8MHz.

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:

- Oscillator module — The 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 divided VCO clock, CGMVCLK, divided by three 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.

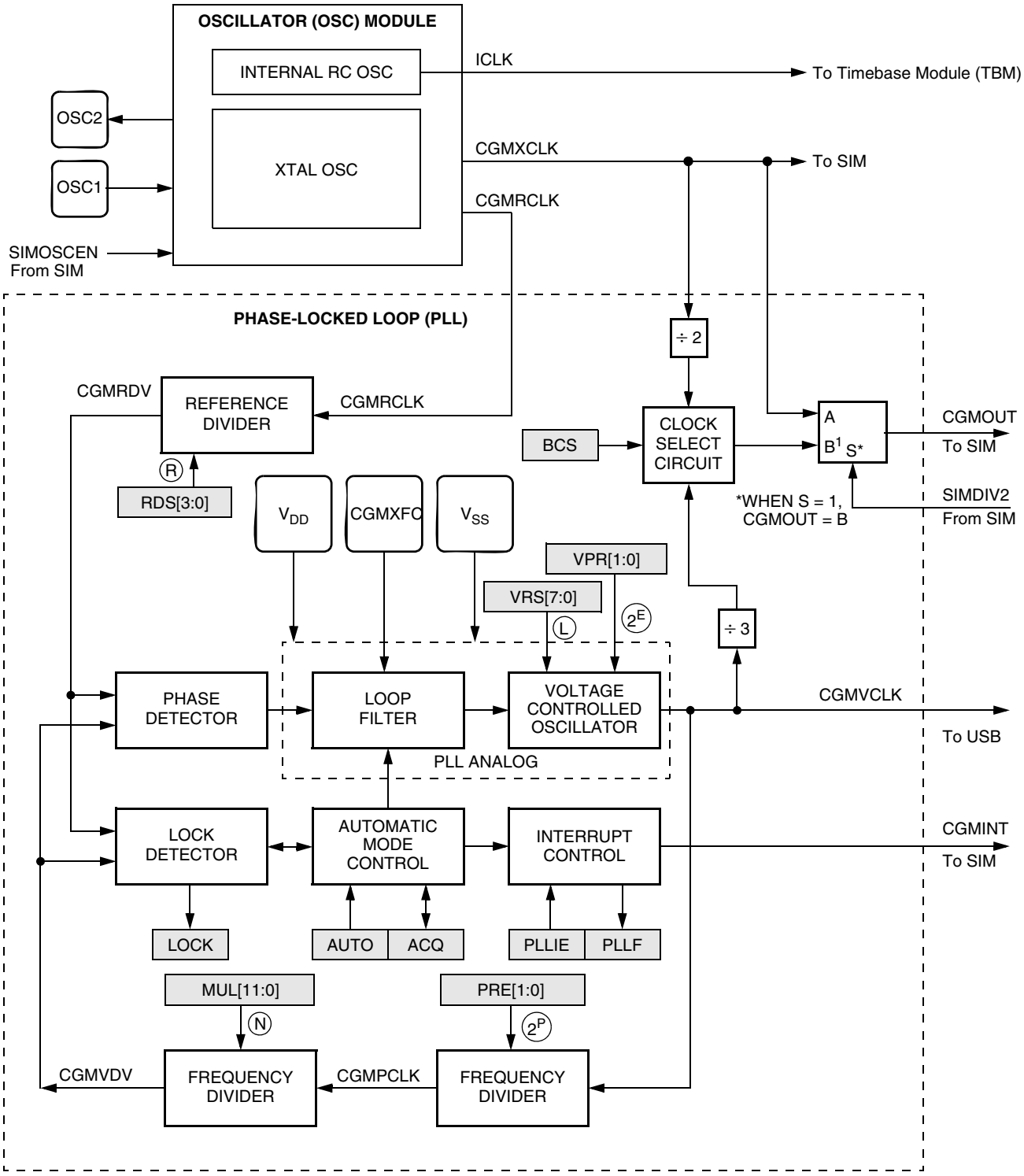


Figure 5-1. CGM Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$1090	PLL Control Register (PTCL)	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		Write:								
		Reset:	0	0	1	0	0	0	0	0
$1091	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	
$1092	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
$1093	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
$1094	1095PLL VCO Range Select Register (PMRS)	Read:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
		Write:								
		Reset:	0	1	0	0	0	0	0	0
$1095	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 Oscillator Module

The oscillator module provides two clock outputs CGMXCLK and CGMRCLK to the CGM module. CGMXCLK when selected, is driven to SIM module to generate the system bus clock. CGMRCLK is used by the phase-lock-loop to provide a higher frequency system bus clock. The oscillator module also provides the reference clock for the timebase module (TBM). See Chapter 9 Timebase Module (TBM) for detailed description on TBM.

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 pre-scaler
- 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}, (125 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 (4 MHz), always set R = 1 for specified performance. With an external high-frequency clock source, use R to divide the external frequency to between 1 MHz and 8 MHz.

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 (the CGMPCLK) 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 $\overline{ACQ}$ 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 $\overline{ACQ}$ 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 $\overline{ACQ}$ 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 $\overline{ACQ}$ 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:

- $\overline{ACQ}$ is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the $\overline{ACQ}$ bit must be clear.
- Before entering tracking mode ($\overline{ACQ}$ = 1), software must wait a given time, t_{ACQ} (See 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}, or the desired VCO frequency, $f_{VCLKDES}$; and then solve for the other.
   The relationship between f_{BUS} and f_{VCLK} is governed by the equation:
   $$f_{VCLK} = 6 \times f_{BUS}$$
2. Choose a practical PLL reference frequency, f_{RCLK}, and the reference clock divider, R. Typically, the reference is 4MHz 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
   $$f_{VCLK} = \frac{2^P N}{R}(f_{RCLK})$$
   where N is the integer range multiplier, between 1 and 4095.

   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 Chapter 19 Electrical Specifications.
   Choose the reference divider, R = 1.

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

3. Calculate N:

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

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

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

$$f_{BUS} = \frac{f_{VCLK}}{6}$$

5. 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.

6. Select a VCO linear range multiplier, L, where f_{NOM} = 125kHz

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

7. 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,

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

8. 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.

9. 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 Examples

CGMVCLK	CGMPCLK	f_{BUS}	f_{RCLK}	R	N	P	E	L
48 MHz	24 MHz	8 MHz	4 MHz	1	06	1	2	96

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 oscillator clock, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the base clock, CGMOUT. The CGMXCLK clock is divided by two while the CGMVCLK is divided by three to correct the duty cycle. The two divided clocks go through a transition control circuit that to change from one clock source to the other. During this time, CGMOUT is held in stasis. Therefore, the bus clock frequency, which is one-half of the base clock frequency, is either one-fourth the frequency of the selected clock (CGMXCLK) or one-sixth the frequency of the selected CGMVCLK clock.

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The divided 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 divided VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the divided VCO clock. The divided 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 oscillator 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:

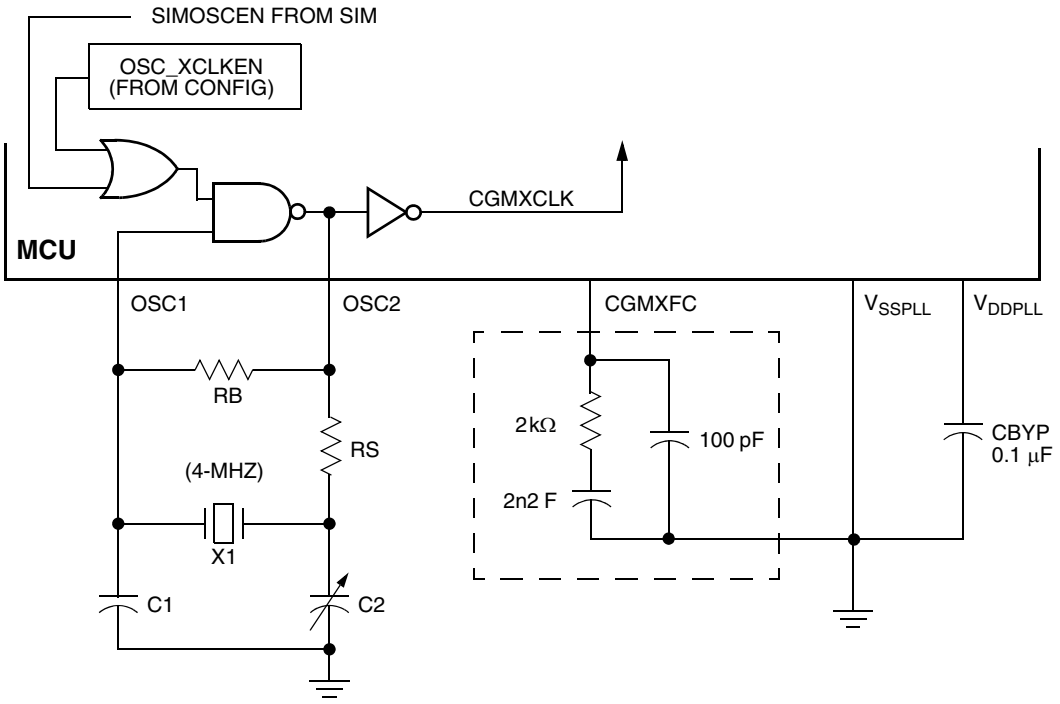
- Crystal, X_1
- Fixed capacitor, C_1
- Tuning capacitor, C_2 (can also be a fixed capacitor)
- Feedback resistor, R_B
- Series resistor, R_S

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 5.8 Acquisition/Lock Time Specifications for routing information, filter network and its effects on PLL performance.)



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

Figure 5-3. CGM 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 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.

5.4.5 CGM Reference Clock (CGMRCLK)

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

5.4.6 CGM VCO Clock Output (CGMVCLK)

CGMVCLK is the clock output from the VCO.

5.4.7 CGM Base Clock Output (CGMOUT)

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

5.4.8 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: $1090

	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 three 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

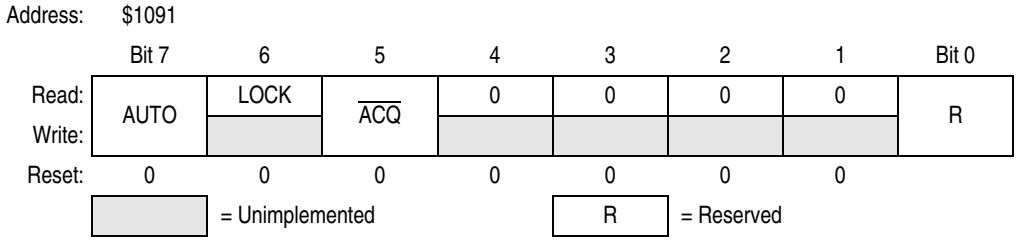


Figure 5-5. PLL Bandwidth Control Register (PBWCR)

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 $\overline{ACQ}$ 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

$\overline{ACQ}$ — Acquisition Mode Bit

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

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

1 = Tracking mode
0 = Acquisition mode

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: $1092

	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: $1093

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
Write:								
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: $1094

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
Write:								
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: $1095

	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 divided VCO clock, CGMVCLK, divided by three 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 three 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) for the selected oscillator 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 (CGMOUT, CGMVCLK, and CGMINT).

If the STOP instruction is executed with the VCO clock, CGMVCLK, divided by three driving CGMOUT, the PLL automatically clears the BCS bit in the PLL control register (PCTL), thereby selecting the oscillator 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 6.7.3 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.)

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 4 MHz 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.

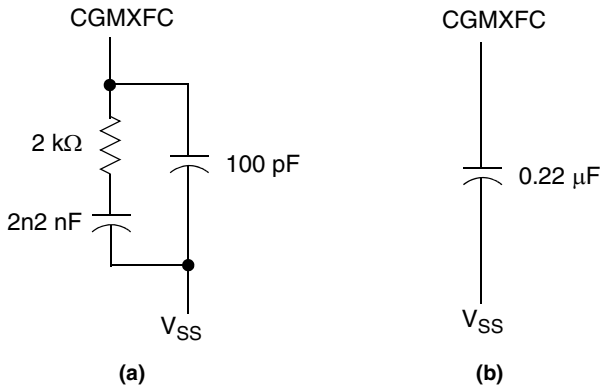


Figure 5-10. PLL Filter

Chapter 6
System Integration Module (SIM)

6.1 Introduction

This section describes the system integration module (SIM). Together with the CPU, the SIM controls all MCU activities. A block diagram of the SIM is shown in Figure 6-1. Figure 6-2 is a summary of the SIM input/output (I/O) registers. The SIM is a system state controller that coordinates CPU and exception timing. The SIM is responsible for:

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

Table 6-1 shows the internal signal names used in this section.

Table 6-1. Signal Name Conventions

Signal Name	Description
ICLK	Internal RC oscillator clock
CGMXCLK	Selected oscillator clock from oscillator module
CGMVCLK	PLL VCO output and the divided PLL output
CGMOUT	CGMVCLK-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/$\overline{W}$	Read/write signal

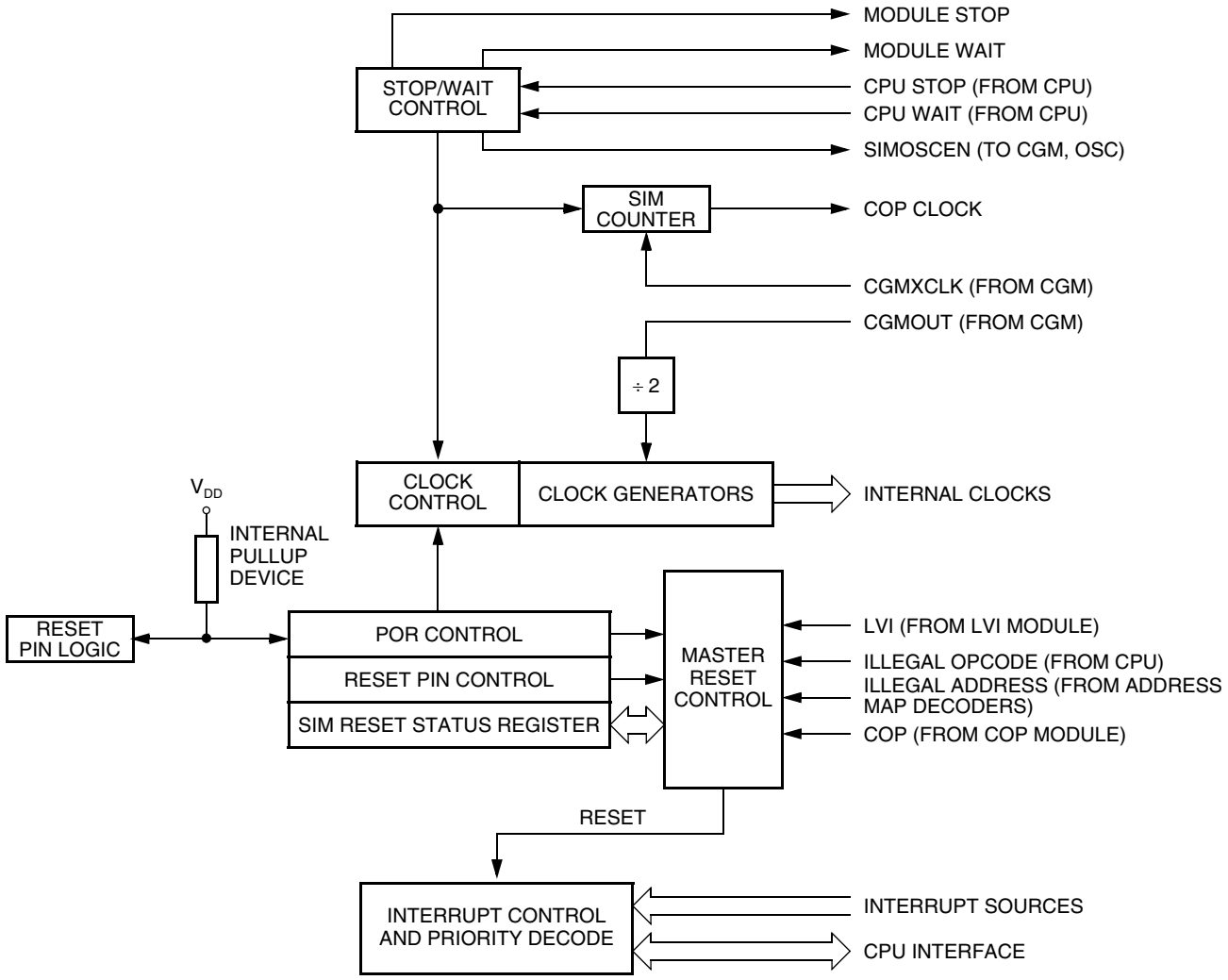


Figure 6-1. SIM Block Diagram

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

= Unimplemented R = Reserved

Figure 6-2. SIM I/O Register Summary

$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:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE06	Interrupt Status Register 3 (INT3)	Read:	0	0	0	0	0	0	0	IF15
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 6-2. SIM I/O Register Summary

6.2 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 6-3. This clock can come from either an external oscillator or from the on-chip PLL. (See Chapter 5 Clock Generator Module (CGM).)

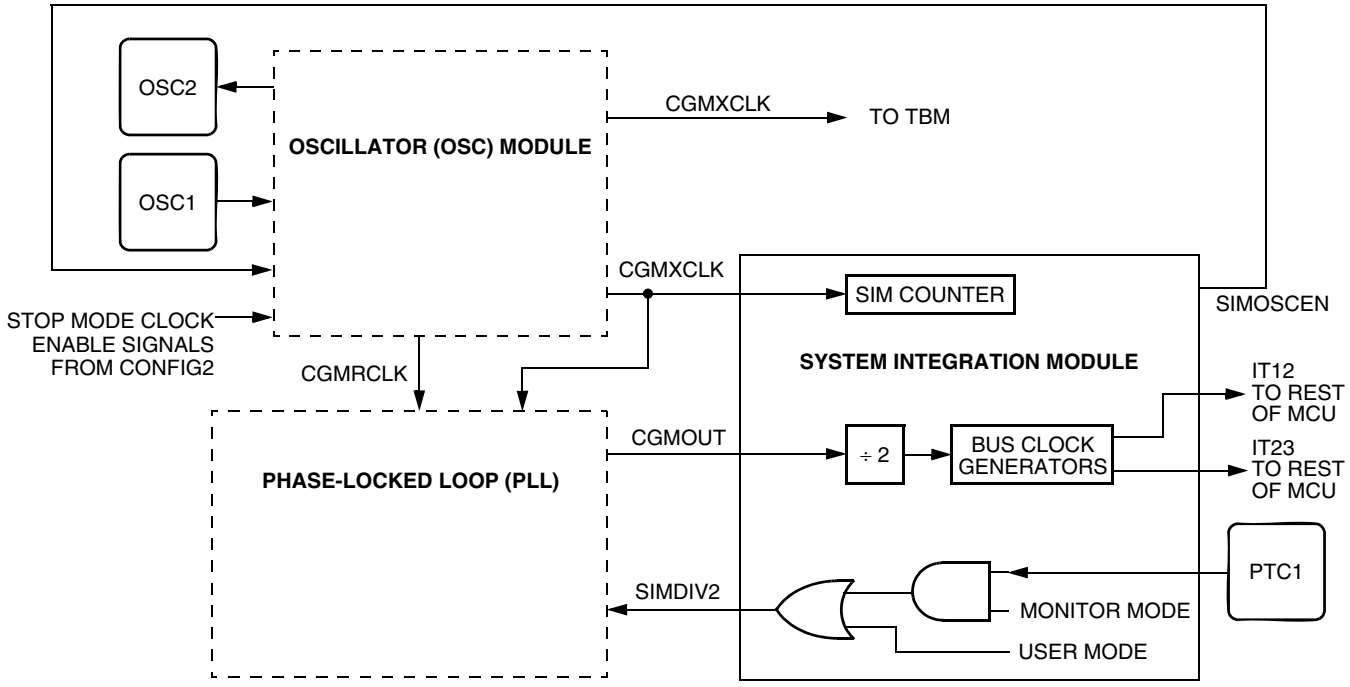


Figure 6-3. CGM Clock Signals

6.2.1 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 six.

6.2.2 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.

6.2.3 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 6.6.2 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.

6.3 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
- Universal serial bus module (USB)

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 6.4 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See 6.7 SIM Registers.)

6.3.1 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 6-2 for details. Figure 6-4 shows the relative timing.

Table 6-2. PIN Bit Set Timing

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

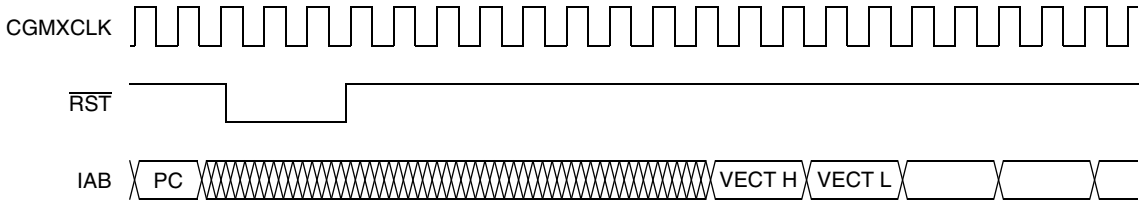


Figure 6-4. External Reset Timing

6.3.2 Active Resets from Internal Sources

All internal reset sources actively pull the 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 6-5). An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR (see Figure 6-6).

NOTE

For LVI or POR resets, the SIM cycles through 4096 + 32 CGMXCLK cycles during which the SIM forces the RST pin low. The internal reset signal then follows the sequence from the falling edge of RST shown in Figure 6-5.

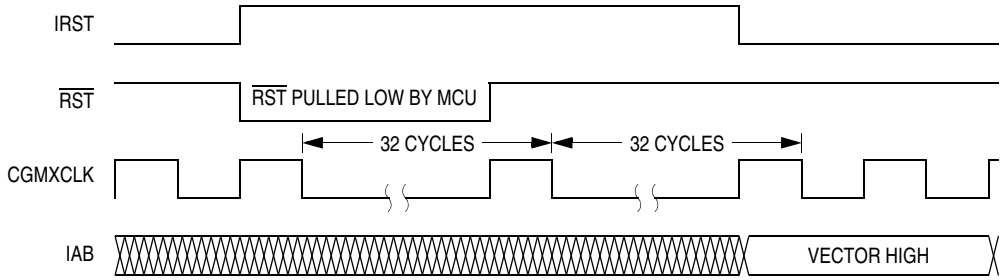


Figure 6-5. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

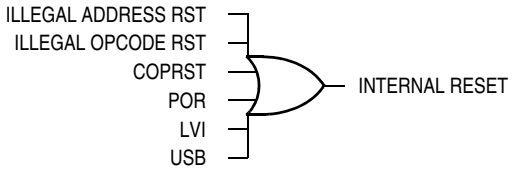


Figure 6-6. Sources of Internal Reset

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

6.3.2.1 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 (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 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.

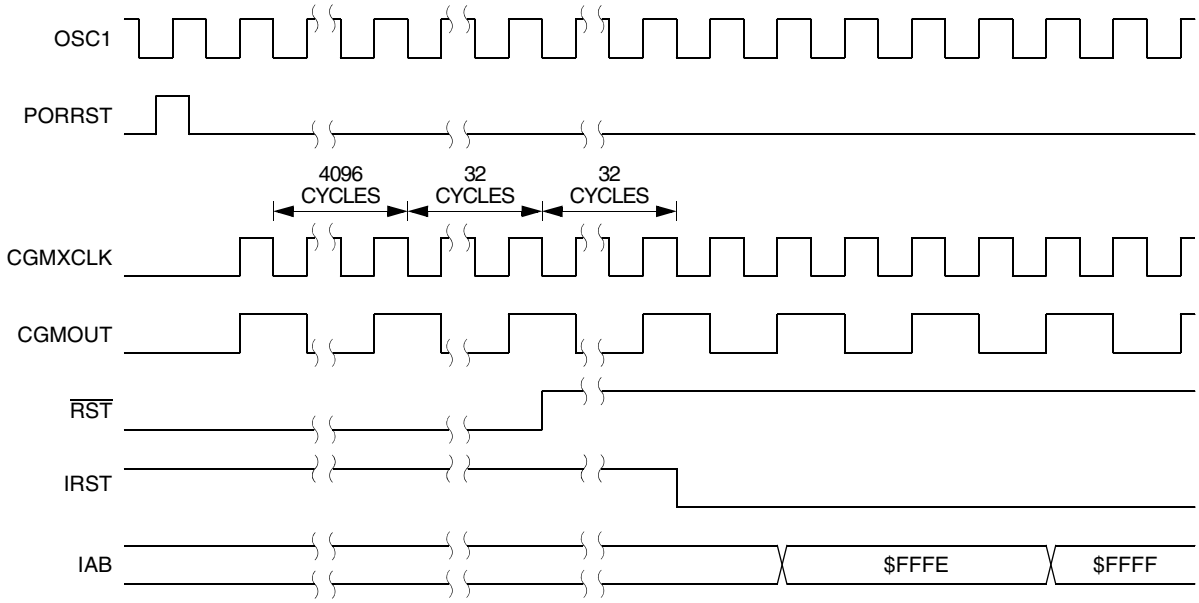


Figure 6-7. POR Recovery

6.3.2.2 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 8176 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 $\overline{\text{IRQ1}}$ 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 $\overline{\text{IRQ1}}$ 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.

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

6.3.2.4 Illegal Address Reset

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

6.3.2.5 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 $LVI_{\overline{TRIPF}}$ voltage. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{RST}$) 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 $\overline{RST}$ pin for all internal reset sources.

6.3.2.6 Universal Serial Bus (USB) Reset

The USB module will detect a reset signaled on the bus by the presence of an extended SE0 at the USB data pins of a device. The MCU seeing a single-ended 0 on its USB data inputs for more than 2.5μs treats that signal as a reset. After the reset is removed, the device will be in the attached, but not yet addressed or configured, state (refer to Section 9.1 USB Devices of the *Universal Serial Bus Specification Rev. 2.0*). The device must be able to accept the device address via a SET_ADDRESS command (refer to Section 9.4 of the *Universal Serial Bus Specification Rev. 2.0*) no later than 10ms after the reset is removed.

USB reset can be disabled to generate an internal reset. It can be configured to generate IRQ interrupt. (See Chapter 3 Configuration Registers (CONFIG).)

NOTE

USB reset is disabled when the USB module is disabled by clearing the USBEN bit of the USB address register (UADDR).

6.4 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 13 bits long and is clocked by the falling edge of CGMXCLK.

6.4.1 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.

6.4.2 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 mask option register. 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.

6.4.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See 6.6.2 Stop Mode for details.) The SIM counter is free-running after all reset states. (See 6.3.2 Active Resets from Internal Sources for counter control and internal reset recovery sequences.)

6.5 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

6.5.1 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 6-8 shows interrupt entry timing, and Figure 6-9 shows interrupt recovery timing.

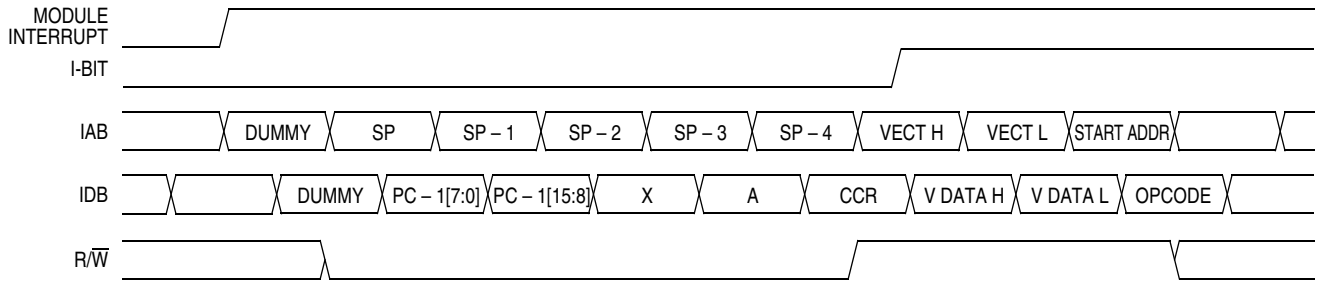


Figure 6-8. Interrupt Entry Timing

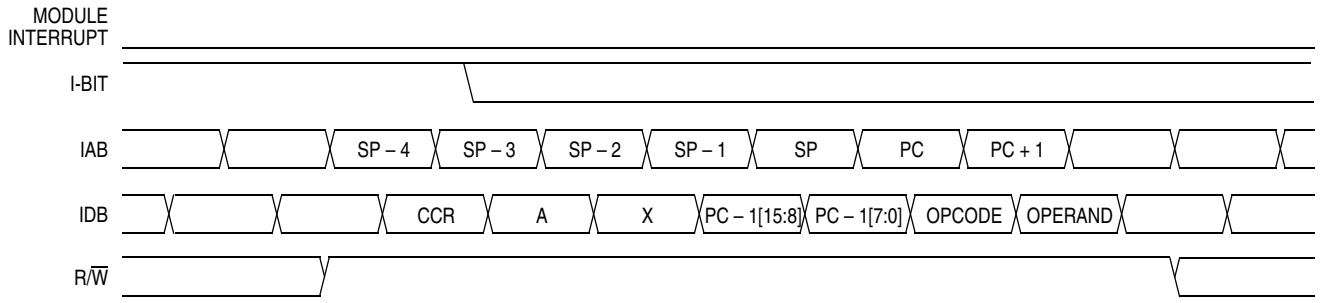


Figure 6-9. Interrupt Recovery Timing

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 6-10.)

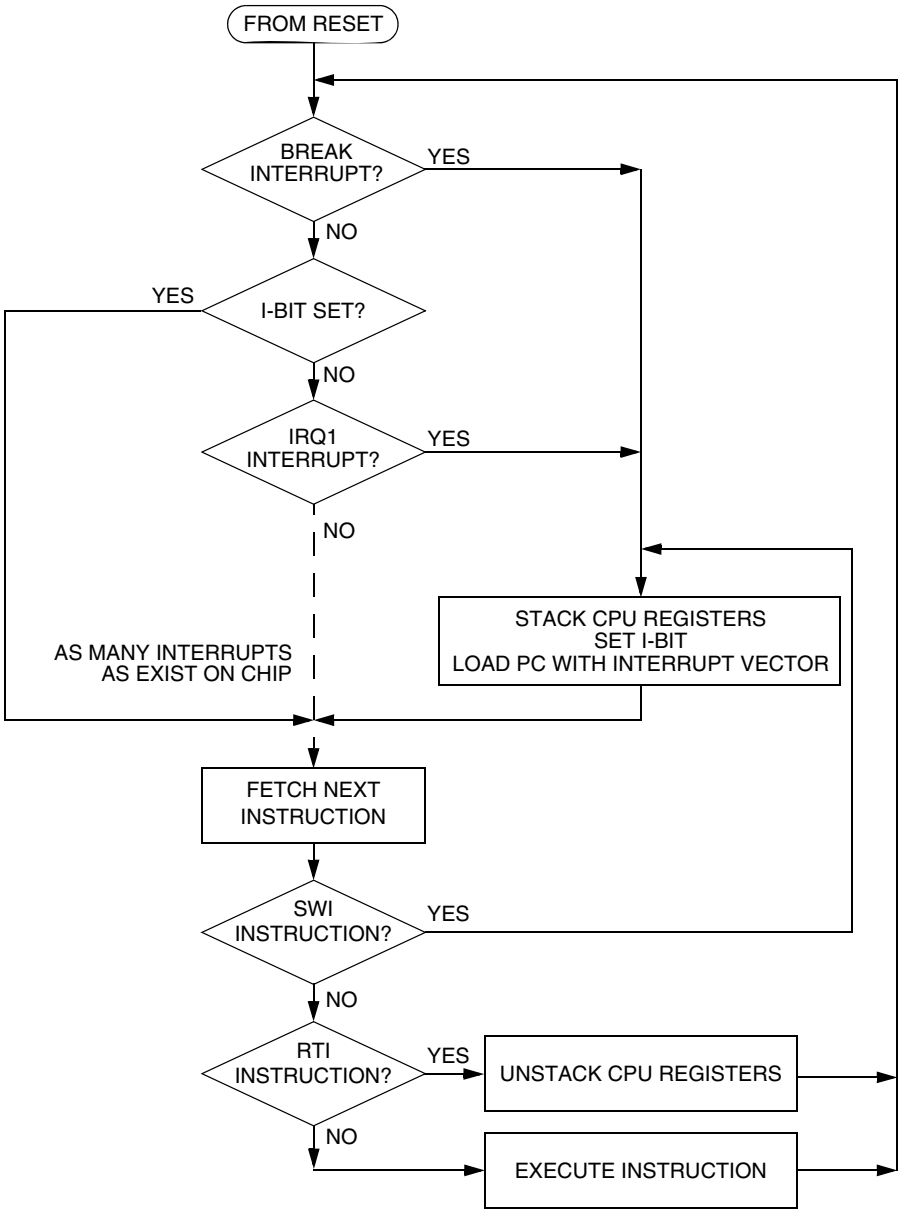


Figure 6-10. Interrupt Processing

6.5.1.1 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 6-11 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.

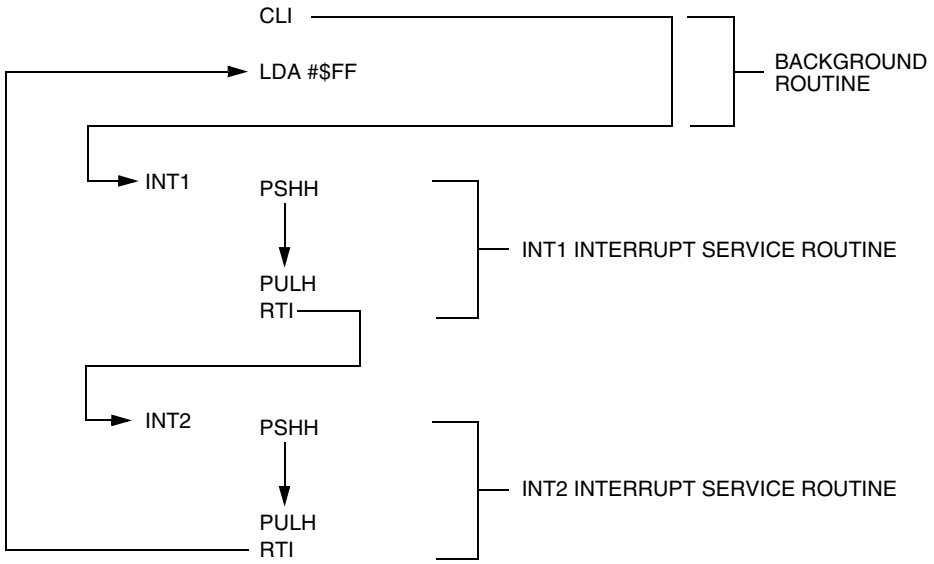


Figure 6-11. Interrupt Recognition Example

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.

6.5.1.2 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.

6.5.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. Table 6-3 summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

6.5.2.1 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 6-12. Interrupt Status Register 1 (INT1)

IF6–IF1 — Interrupt Flags 6–1

These flags indicate the presence of interrupt requests from the sources shown in Table 6-3.

1 = Interrupt request present
0 = No interrupt request present

Bit 0 and Bit 1 — Always read 0

6.5.2.2 Interrupt Status Register 2

Address: $FE05

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

R = Reserved

Figure 6-13. Interrupt Status Register 2 (INT2)

IF14–IF7 — Interrupt Flags 14–7

These flags indicate the presence of interrupt requests from the sources shown in Table 6-3.

1 = Interrupt request present
0 = No interrupt request present

6.5.2.3 Interrupt Status Register 3

Address: $FE06

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

R = Reserved

Figure 6-14. Interrupt Status Register 3 (INT3)

IF15 — Interrupt Flag 15

This flag indicates the presence of an interrupt request from the source shown in Table 6-3.

1 = Interrupt request present
0 = No interrupt request present

Table 6-3. Interrupt Sources

Priority	INT Flag	Vector Address	Interrupt Source
Lowest	IF15	$FFDE	Timebase
		$FFDF	
	IF14	$FFE0	Keyboard
		$FFE1	
	IF13	$FFE2	SPI Transmit
		$FFE3	
	IF12	$FFE4	SPI Receive
		$FFE5	
	IF11	$FFE6	Reserved
		$FFE7	
	IF10	$FFE8	Reserved
		$FFE9	
	IF9	$FFEA	Reserved
		$FFEB	
	IF8	$FFEC	PS2 Interrupt
		$FFED	
	IF7	$FFEE	TIM1 Overflow
		$FFEF	
	IF6	$FFF0	TIM1 Channel 1
		$FFF1	
	IF5	$FFF2	TIM1 Channel 0
		$FFF3	
	IF4	$FFF4	PLL
		$FFF5	
	IF3	$FFF6	$\overline{\text{IRQ}}$
		$FFF7	
	IF2	$FFF8	USB Endpoint
		$FFF9	
	IF1	$FFFA	USB System
		$FFFB	
	—	$FFFC	SWI
		$FFFD	
	—	$FFFE	Reset
Highest		$FFFF	

6.5.3 Reset

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

6.5.4 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See Chapter 18 Break Module (BRK).) The SIM puts the CPU into the break state by forcing it to the SWI vector location. Refer to the break interrupt subsection of each module to see how each module is affected by the break state.

6.5.5 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 initializing the break clear flag enable bit (BCFE) in the SIM break flag control register (SBFCR).

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

Setting the BCFE bit enables the clearing mechanisms. Once cleared in break mode, a flag remains cleared even when break mode is exited. Status flags with a 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.

6.6 Low-Power Modes

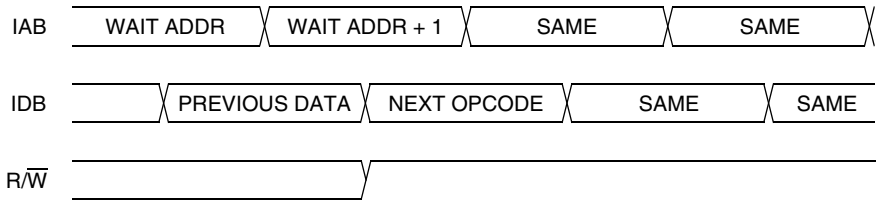
Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described in the following subsections. Both STOP and WAIT clear the interrupt mask (I) in the condition code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

6.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 6-15 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.

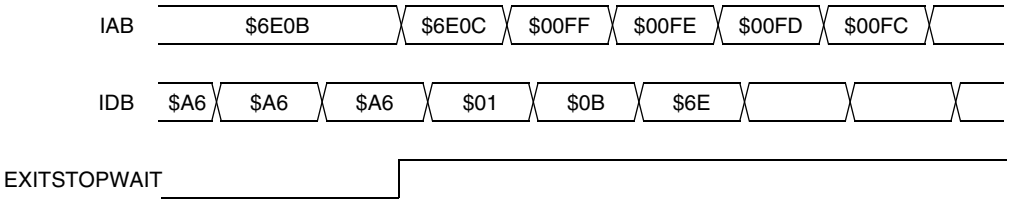
Wait mode also can be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is logic 0, then the computer operating properly module (COP) is enabled and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



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

Figure 6-15. Wait Mode Entry Timing

Figure 6-16 and Figure 6-17 show the timing for WAIT recovery.



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

Figure 6-16. Wait Recovery from Interrupt or Break

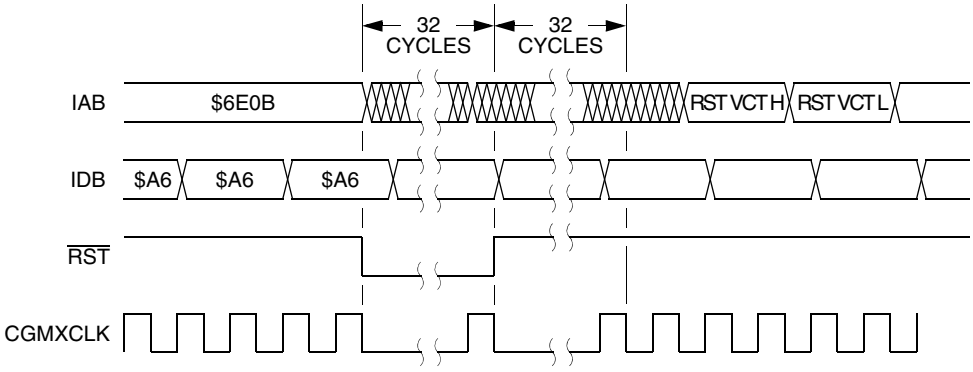


Figure 6-17. Wait Recovery from Internal Reset

6.6.2 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

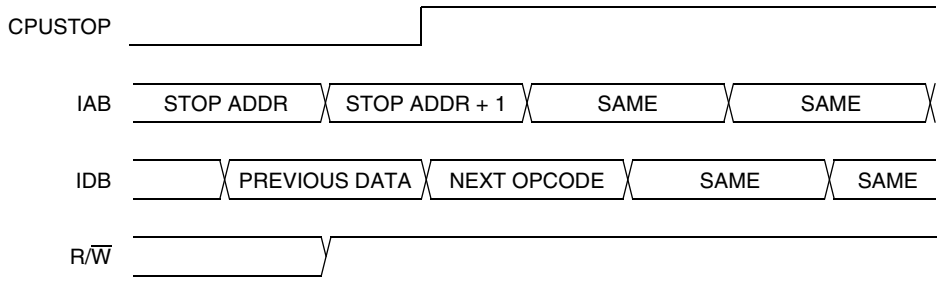
External crystal applications should use the full stop recovery time by clearing the SSREC bit.

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 6-18 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 6-18. Stop Mode Entry Timing

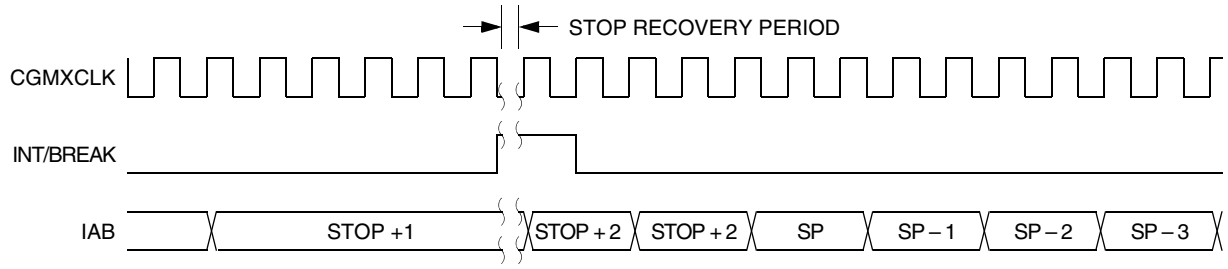


Figure 6-19. Stop Mode Recovery from Interrupt or Break

6.7 SIM Registers

The SIM has three memory-mapped registers:

- SIM Break Status Register (SBSR) — $FE00
- SIM Reset Status Register (SRSR) — $FE01
- SIM Break Flag Control Register (SBFCR) — $FE03

6.7.1 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	
Reset:							0	

Note: Writing a logic 0 clears SBSW. R = Reserved

Figure 6-20. SIM Break Status Register (SBSR)

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.

6.7.2 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	USB	LVI	0
Write:								
Reset:	1	0	0	0	0	0	0	0

= Unimplemented

Figure 6-21. SIM Reset Status Register (SRSR)

POR — Power-On Reset Bit

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

PIN — External Reset Bit

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

COP — Computer Operating Properly Reset Bit

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

ILOP — Illegal Opcode Reset Bit

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

ILAD — Illegal Address Reset Bit (opcode fetches only)

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

USB — USB Reset Bit

1 = Last reset caused by USB reset.
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

6.7.3 SIM Break Flag Control Register

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

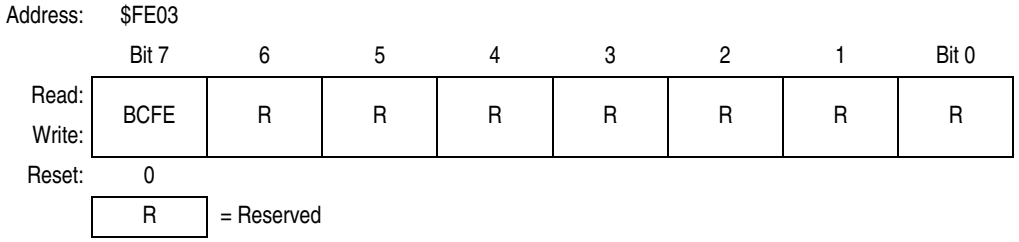


Figure 6-22. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

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

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

Chapter 7
Monitor ROM (MON)

7.1 Introduction

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

7.2 Features

Features of the monitor ROM include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- Execution of code in RAM or ROM
- ROM memory security feature[1]
- 960 bytes monitor ROM code size ($FC00–$FDFF and $FE10–$FFCE)
- Standard monitor mode entry if high voltage, V_{TST}, is applied to $\overline{IRQ}$

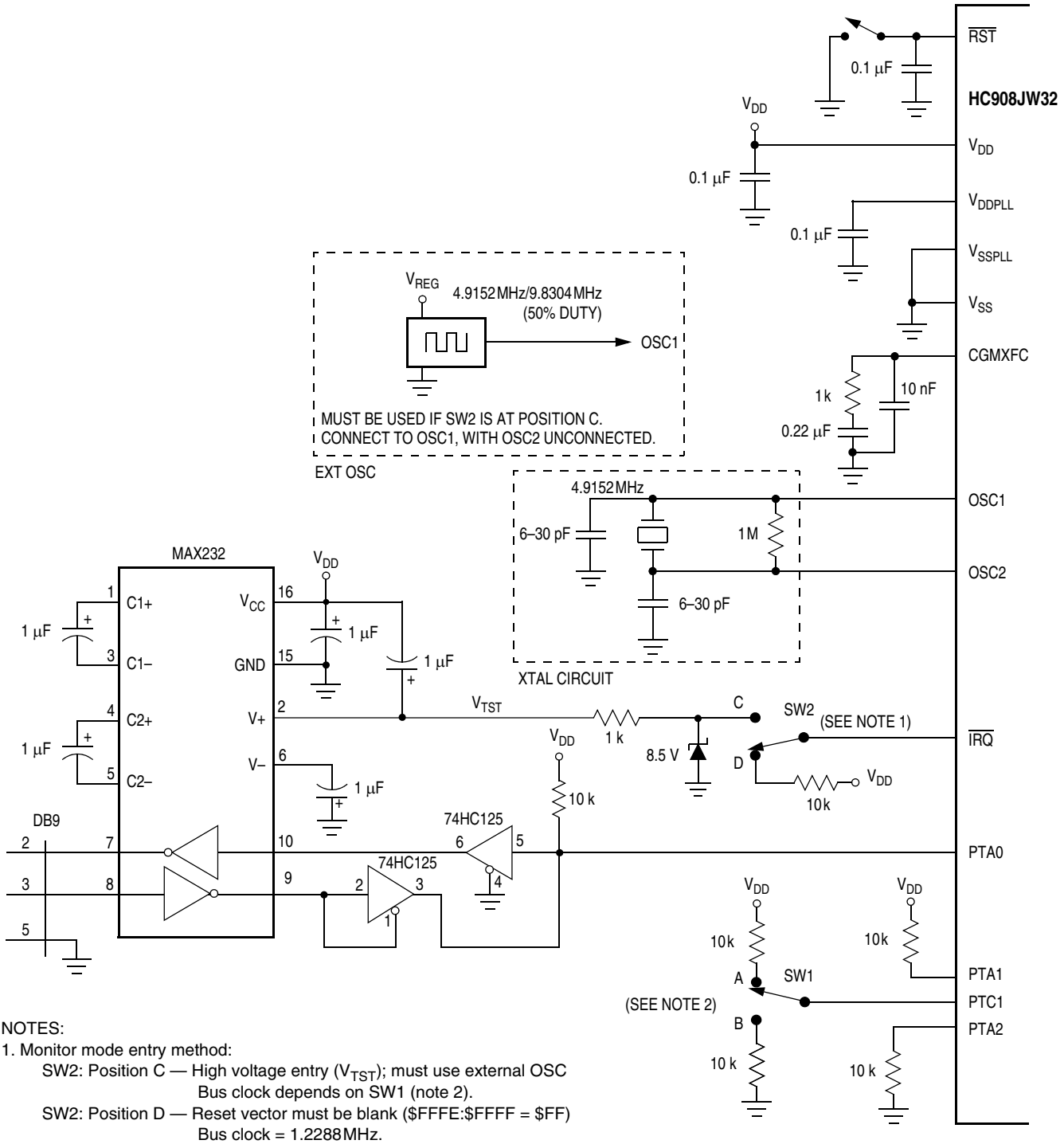
7.3 Functional Description

The monitor ROM receives and executes commands from a host computer. Figure 7-1 shows an example circuit 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.

The monitor code allows enabling the PLL to generate the internal clock, provided the reset vector is blank, when the device is being clocked by a low-frequency crystal. This entry method, which is enabled when $\overline{IRQ}$ is held low out of reset, is intended to support serial communication/ programming at 9600 baud in monitor mode.

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



NOTES:

1. Monitor mode entry method:
   SW2: Position C — High voltage entry (V_{TST}); must use external OSC
   Bus clock depends on SW1 (note 2).
   SW2: Position D — Reset vector must be blank ($FFFE:$FFFF = $FF)
   Bus clock = 1.2288MHz.
2. Affects high voltage entry to monitor mode only (SW2 at position C):
   SW1: Position A — Bus clock = OSC1 ÷ 4
   SW1: Position B — Bus clock = OSC1 ÷ 2
5. See Table 19-4. DC Electrical Characteristics for V_{TST} voltage level requirements.

Figure 7-1. Monitor Mode Circuit

7.3.1 Entering Monitor Mode

Table 7-1 shows the pin conditions for entering monitor mode. As specified in the table, monitor mode may be entered after a POR and will allow communication at 9600 baud provided one of the following sets of conditions is met:

1. $\overline{IRQ}$ = V_{TST} (PLL off):
   – The external clock is 4.9152 MHz with PTC1 low
2. $\overline{IRQ}$ = V_{TST} (PLL off):
   – The external clock is 9.8304 MHz with PTC1 high

If V_{TST} is applied to $\overline{IRQ}$ and PTC1 is low upon monitor mode entry (above condition set 1), the bus frequency is a divide-by-two of the input clock. If PTC1 is high with V_{TST} applied to $\overline{IRQ}$ upon monitor mode entry, the bus frequency will be a divide-by-four of the input clock. Holding the PTC1 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator only if V_{TST} is applied to $\overline{IRQ}$. 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 bus frequency.

If entering monitor mode without high voltage on $\overline{IRQ}$ (above condition set 2 or 3, where applied voltage is either V_{DD} or V_{SS}), then all port A pin requirements and conditions, including the PTC1 frequency divisor selection, are not in effect. This is to reduce circuit requirements when performing in-circuit programming.

Table 7-1. Monitor Mode Signal Requirements and Options

$\overline{IRQ}$	$\overline{RST}$	PTA2	PTA1	PTA0 (1)	PTC1	External Clock(2)	Bus Freq.	PLL	COP	Baud Rate	Comment
X	GND	X	X	X	X	X	0	X	Disabled	0	No operation until reset goes high
V_{TST} (3)	V_{DD} or V_{TST}	0	1	1	0	4.9152 MHz	2.4576 MHz	OFF	Disabled	9600	PTA1 and PTA2 voltages only required if $\overline{IRQ}$ = V_{TST}; PTC1 determines frequency divider
V_{TST}(3)	V_{DD} or V_{TST}	0	1	1	1	9.8304 MHz	2.4576 MHz	OFF	Disabled	9600	PTA1 and PTA2 voltages only required if $\overline{IRQ}$ = V_{TST}; PTC1 determines frequency divider
V_{DD} or GND	V_{DD} or V_{TST}	X	X	X	X	X	—	OFF	Enabled	—	Enters user mode

1. PTA0 = 1 if serial communication; PTA0 = 0 if parallel communication
2. External clock is derived by a 4.9152/9.8304 MHz off-chip oscillator
3. Monitor mode entry by $\overline{IRQ}$ = V_{TST}, a 4.9152/9.8304 MHz off-chip oscillator must be used. The MCU internal crystal oscillator circuit is bypassed.

The COP module is disabled in monitor mode based on these conditions:

- If monitor mode was entered as a result of the reset vector being blank (above condition set 2 or 3), the COP is always disabled regardless of the state of $\overline{IRQ}$ or $\overline{RST}$.
- If monitor mode was entered with V_{TST} on $\overline{IRQ}$ (condition set 1), then the COP is disabled as long as V_{TST} is applied to either $\overline{IRQ}$ or $\overline{RST}$.

The second condition states that as long as V_{TST} is maintained on the $\overline{IRQ}$ pin after entering monitor mode, or if V_{TST} is applied to $\overline{RST}$ after the initial reset to get into monitor mode (when V_{TST} was applied to $\overline{IRQ}$), then the COP will be disabled. In the latter situation, after V_{TST} is applied to the $\overline{RST}$ pin, V_{TST} can be removed from the $\overline{IRQ}$ pin in the interest of freeing the $\overline{IRQ}$ for normal functionality in monitor mode.

Enter monitor mode with pin configuration shown in Figure 7-1 by pulling $\overline{RST}$ low and then high. The rising edge of $\overline{RST}$ latches monitor mode. Once monitor mode is latched, the values on the specified pins can change (except for PTA1, where it should be held until after security, see 7.4 Security).

Once out of reset, the MCU waits for the host to send eight security bytes. (See 7.4 Security.) After the security bytes, the MCU sends a break signal (10 consecutive logic 0s) to the host, indicating that it is ready to receive a command.

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.

NOTE

Exiting monitor mode after it has been initiated by having a blank reset vector requires a power-on reset (POR). Pulling $\overline{RST}$ low will not exit monitor mode in this situation.

Table 7-2 summarizes the differences between user mode and monitor mode vectors.

Table 7-2. Mode Differences (Vectors)

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

7.3.2 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.

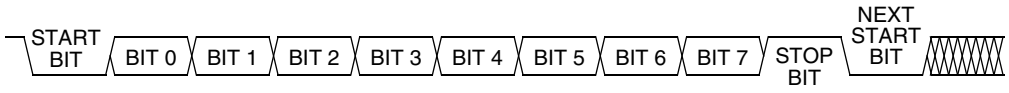


Figure 7-2. Monitor Data Format

7.3.3 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.

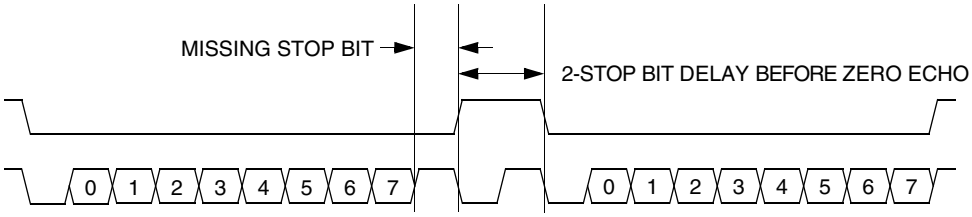


Figure 7-3. Break Transaction

7.3.4 Baud Rate

The communication baud rate is controlled by the crystal frequency and the state of the PTB0 pin (when $\overline{IRQ1}$ is set to V_{TST}) upon entry into monitor mode. When PTB0 is high, the divide by ratio is 1024. If the PTB0 pin is at logic 0 upon entry into monitor mode, the divide by ratio is 512.

If monitor mode was entered with V_{DD} on $\overline{IRQ1}$, then the divide by ratio is set at 1024, regardless of PTB0. This condition for monitor mode entry requires that the reset vector is blank.

Table 7-3 lists external frequencies required to achieve a standard baud rate of 9600 BPS. Other standard baud rates can be accomplished using proportionally higher or lower frequency generators. If using a crystal as the clock source, be aware of the upper frequency limit that the internal clock module can handle.

Table 7-3. Monitor Baud Rate Selection

External Frequency	$\overline{IRQ1}$	PTB0	Internal Frequency	Baud Rate (BPS)
4.9152 MHz	V_{TST}	0	2.4576 MHz	9600
9.8304 MHz	V_{TST}	1	2.4576 MHz	9600
9.8304 MHz	V_{DD}	X	2.4576 MHz	9600

7.3.5 Commands

The monitor ROM firmware uses these commands:

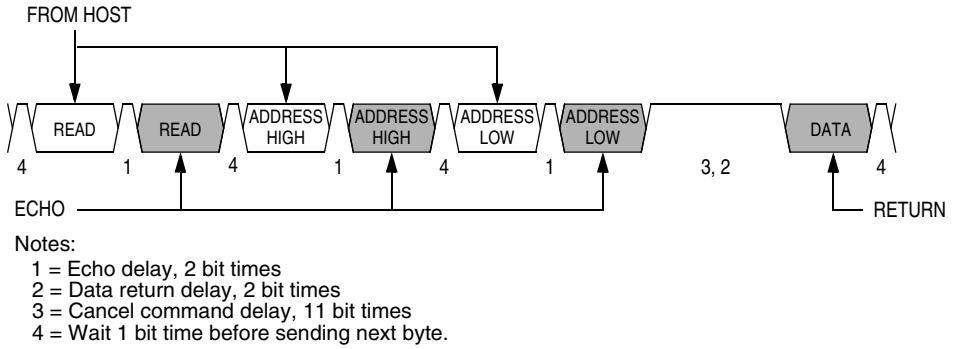
- READ (read memory)
- WRITE (write memory)
- IREAD (indexed read)
- IWRITE (indexed write)
- READSP (read stack pointer)
- RUN (run user program)

The monitor ROM firmware echoes each received byte back to the PTA0 pin for error checking. An 11-bit delay at the end of each command allows the host to send a break character to cancel the command. A

delay of two bit times occurs before each echo and before READ, IREAD, or READSP data is returned. The data returned by a read command appears after the echo of the last byte of the command.

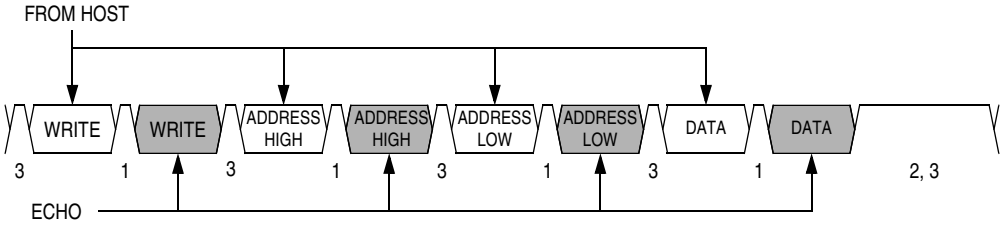
NOTE

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



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

Figure 7-4. Read Transaction



Notes:
1 = Echo delay, 2 bit times
2 = Cancel command delay, 11 bit times
3 = Wait 1 bit time before sending next byte.

Figure 7-5. Write Transaction

A brief description of each monitor mode command is given in Table 7-4 through Table 7-9.

Table 7-4. READ (Read Memory) Command

Description	Read byte from memory
Operand	2-byte address in high-byte:low-byte order
Data Returned	Returns contents of specified address
Opcode	$4A
Command Sequence	

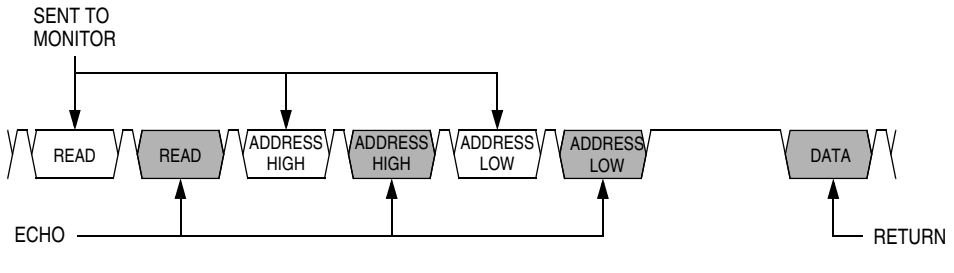


Table 7-5. WRITE (Write Memory) Command

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

Command Sequence

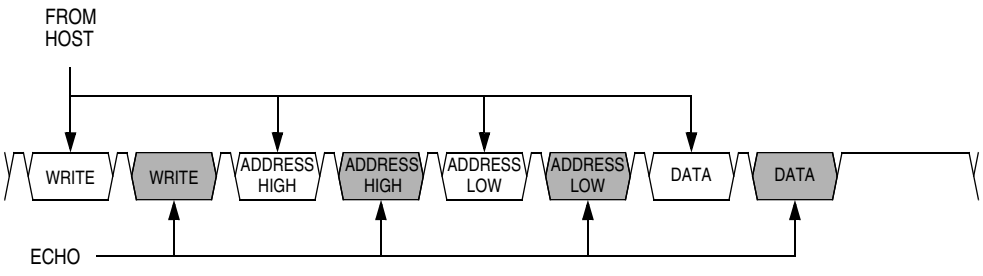


Table 7-6. IREAD (Indexed Read) Command

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

Command Sequence

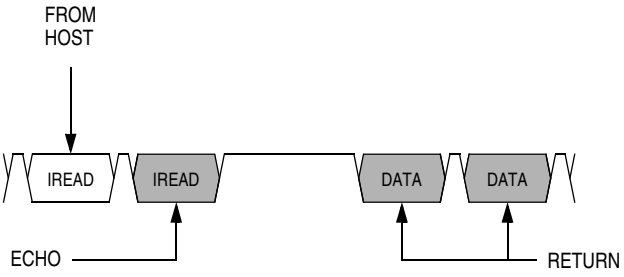
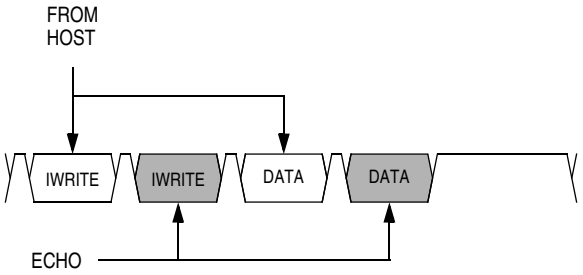


Table 7-7. IWRITE (Indexed Write) Command

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

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

Table 7-8. READSP (Read Stack Pointer) Command

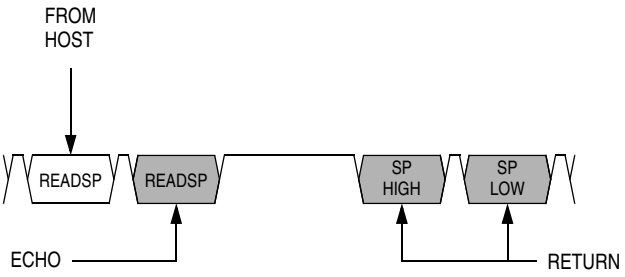
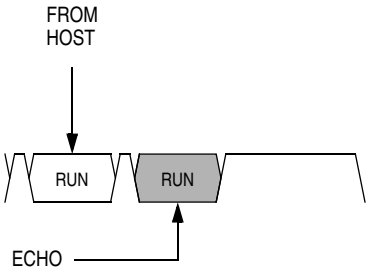
Description	Reads stack pointer
Operand	None
Data Returned	Returns incremented stack pointer value (SP + 1) in high-byte:low-byte order
Opcode	$0C
Command Sequence	

Table 7-9. RUN (Run User Program) Command

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

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

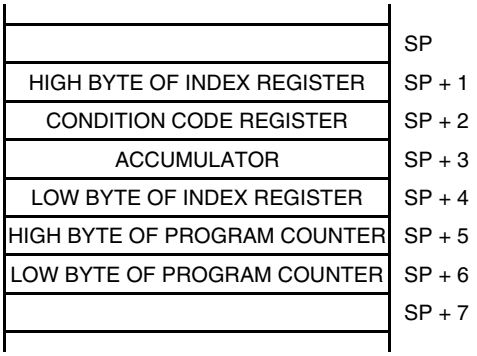


Figure 7-6. Stack Pointer at Monitor Mode Entry

7.4 Security

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

NOTE

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

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all ROM locations and execute code from ROM. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. (See Figure 7-7.)

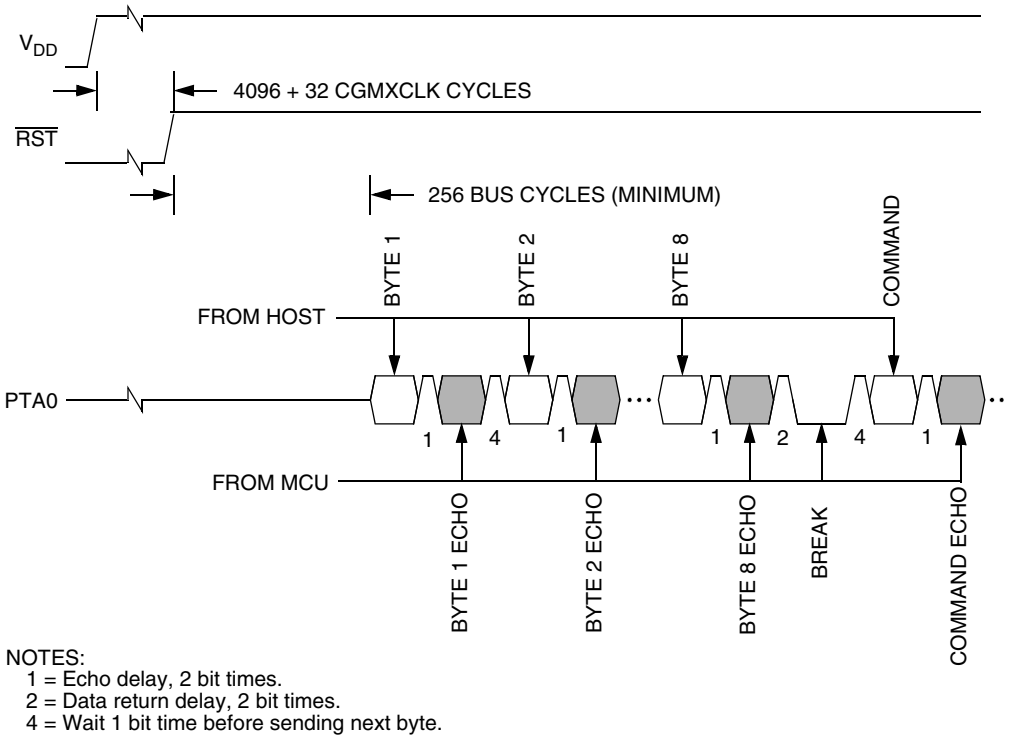


Figure 7-7. Monitor Mode Entry Timing

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 ROM location returns an invalid value and trying to execute code from ROM 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 bits.

To determine whether the security code entered is correct, check to see if bit 6 of RAM address $60 is set. If it is, then the correct security code has been entered and ROM can be accessed.

7.5 ROM-Resident Routines

Five routines stored in the monitor ROM area (thus ROM-resident) are provided for FLASH memory manipulation. They are intended to simplify FLASH program, erase and load operations. Table 7-10 shows a summary of the ROM-resident routines.

Table 7-10. Summary of ROM-Resident Routines

Routine Name	Routine Description	Call Address	Stack Used (bytes)
PRGRNGE	Program a range of locations	$FE10	16
ERARNGE	Erase a page or the entire array	$FE13	10
LDRNGE	Loads data from a range of locations	$FA31	10
MON_PRGRNGE	Program a range of locations in monitor mode	$FF24	18
MON_ERARNGE	Erase a page or the entire array in monitor mode	$FF28	12

The routines are designed to be called as stand-alone subroutines in the user program or monitor mode. The parameters that are passed to a routine are in the form of a contiguous data block, stored in RAM. The index register (H:X) is loaded with the address of the first byte of the data block (acting as a pointer), and the subroutine is called (JSR). Using the start address as a pointer, multiple data blocks can be used, any area of RAM be used. A data block has the control and data bytes in a defined order, as shown in Figure 7-8.

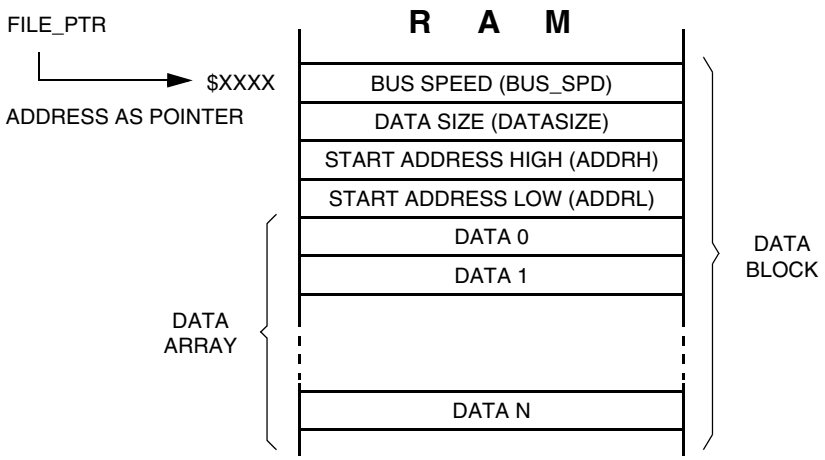


Figure 7-8. Data Block Format for ROM-Resident Routines

During the software execution, it does not consume any dedicated RAM location, the run-time heap will extend the system stack, all other RAM location will not be affected.

The control and data bytes are described below.

- **Bus speed** — This one byte indicates the operating bus speed of the MCU. The value of this byte should be equal to 4 times the bus speed. E.g. for a 4MHz bus, the value is 16 ($10). This control byte is useful where the MCU clock source is switched between the PLL clock and the crystal clock.
- **Data size** — This one byte indicates the number of bytes in the data array that are to be manipulated. The maximum data array size is 255. Routines ERARNGE and MON_ERARNGE do not manipulate a data array, thus, this data size byte has no meaning.
- **Start address** — These two bytes, high byte followed by low byte, indicate the start address of the FLASH memory to be manipulated.
- **Data array** — This data array contains data that are to be manipulated. Data in this array are programmed to FLASH memory by the programming routines: PRGRNGE, MON_PRGRNGE. For the read routines: LDRNGE and data is read from FLASH and stored in this array.

7.5.1 PRGRNGE

PRGRNGE is used to program a range of FLASH locations with data loaded into the data array.

Table 7-11. PRGRNGE Routine

Routine Name	PRGRNGE
Routine Description	Program a range of locations
Calling Address	$FE10
Stack Used	16 bytes
Data Block Format	Bus speed (BUS_SPD) Data size (DATASIZE) Start address high (ADDRH) Start address (ADDRL) Data 1 (DATA1) : Data N (DATAN)

The start location of the FLASH to be programmed is specified by the address ADDRH:ADDRL and the number of bytes from this location is specified by DATASIZE. The maximum number of bytes that can be programmed in one routine call is 255 bytes (max. DATASIZE is 255).

ADDRH:ADDRL do not need to be at a page boundary, the routine handles any boundary misalignment during programming. A check to see that all bytes in the specified range are erased is not performed by this routine prior programming. Nor does this routine do a verification after programming, so there is no return confirmation that programming was successful. User must assure that the range specified is first erased.

The coding example below is to program 64 bytes of data starting at FLASH location $EE00, with a bus speed of 4.9152 MHz. The coding assumes the data block is already loaded in RAM, with the address pointer, FILE_PTR, pointing to the first byte of the data block.

```
              ORG     RAM
       :
FILE_PTR:
BUS_SPD       DS.B    1     ; Indicates 4x bus frequency
DATASIZE      DS.B    1     ; Data size to be programmed
START_ADDR    DS.W    1     ; FLASH start address
DATAARRAY     DS.B    64    ; Reserved data array

PRGRNGE       EQU     $FE10
FLASH_START   EQU     $EE00

              ORG     FLASH
INITIALISATION:
       MOV    #20,    BUS_SPD
       MOV    #64,    DATASIZE
       LDHX   #FLASH_START
       STHX   START_ADDR
       RTS
MAIN:
       BSR    INITIALISATION
       :
       :
       LDHX   #FILE_PTR
       JSR    PRGRNGE
```

7.5.2 ERARNGE

ERARNGE is used to erase a range of locations in FLASH.

Table 7-12. ERARNGE Routine

Routine Name	ERARNGE
Routine Description	Erase a page or the entire array
Calling Address	$FE13
Stack Used	10 bytes
Data Block Format	Bus speed (BUS_SPD) Data size (DATASIZE) Starting address (ADDRH) Starting address (ADDRL)

There are two sizes of erase ranges: a page or the entire array. The ERARNGE will erase the page (512 consecutive bytes) in FLASH specified by the address ADDRH:ADDRL. This address can be any address within the page. Calling ERARNGE with ADDRH:ADDRL equal to $FFFF will erase the entire FLASH array (mass erase). Therefore, care must be taken when calling this routine to prevent an accidental mass erase.

The ERARNGE routine do not use a data array. The DATASIZE byte is a dummy byte that is also not used.

The coding example below is to perform a page erase, from $EE00–$EFFF. The Initialization subroutine is the same as the coding example for PRGRNGE (see 7.5.1 PRGRNGE).

```
ERARNGE         EQU     $FE13
MAIN:
        BSR     INITIALISATION
        :
        :
        LDHX    #FILE_PTR
        JSR     ERARNGE
        :
```

7.5.3 LDRNGE

LDRNGE is used to load the data array in RAM with data from a range of FLASH locations.

Table 7-13. LDRNGE Routine

Routine Name	LDRNGE
Routine Description	Loads data from a range of locations
Calling Address	$FA31
Stack Used	10 bytes
Data Block Format	Bus speed (BUS_SPD) Data size (DATASIZE) Starting address (ADDRH) Starting address (ADDRL) Data 1 : Data N

The start location of FLASH from where data is retrieved is specified by the address ADDRH:ADDRL and the number of bytes from this location is specified by DATASIZE. The maximum number of bytes that can be retrieved in one routine call is 255 bytes. The data retrieved from FLASH is loaded into the data array in RAM. Previous data in the data array will be overwritten. User can use this routine to retrieve data from FLASH that was previously programmed.

The coding example below is to retrieve 64 bytes of data starting from $EE00 in FLASH. The Initialization subroutine is the same as the coding example for PRGRNGE (see 7.5.1 PRGRNGE).

```
LDRNGE          EQU     $FA31
MAIN:
        BSR     INITIALIZATION
        :
        :
        LDHX    #FILE_PTR
        JSR     LDRNGE
        :
```

Chapter 8
Timer Interface Module (TIM)

8.1 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 8-1 is a block diagram of the TIM.

This particular MCU has a single timer interface modules which are denoted as TIM1.

8.2 Features

Features of the TIM include:

- One input capture/output compare channel:
  - 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
  - External TIM clock input (bus frequency / 2 maximum)
- Free-running or modulo up-count operation
- Toggle channel pin on overflow
- TIM counter stop and reset bits
- Modular architecture expandable to eight channels

8.3 Pin Name Conventions

The text that follows describes the timer, TIM1. The TIM input/output (I/O) pin names are T1CH01 (timer channel 01). The TIMER shares three I/O pins with three port C I/O port pins. The timer clock input is used by TIM1 modules. The full names of the TIM I/O pins are listed in Table 8-1. The generic pin names appear in the text that follows.

Table 8-1. Pin Name Conventions

TIM Generic Pin Names:	T1CH0	T1CH1	TCLK1
Full TIM Pin Names:	PTC0/T1CH0	PTC2/T1CH1	PTC1/TCLK1

NOTE

References to timer 1 may be made in the following text by omitting the timer number. For example, TCH01 may refer generically to T1CH01.

8.4 Functional Description

Figure 8-1 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 TIM channel (per timer) is programmable independently as input capture or output compare channel. If a channel is configured as input capture, then an internal pullup device may be enabled for that channel. (See Chapter 13 Input/Output (I/O) Ports.)

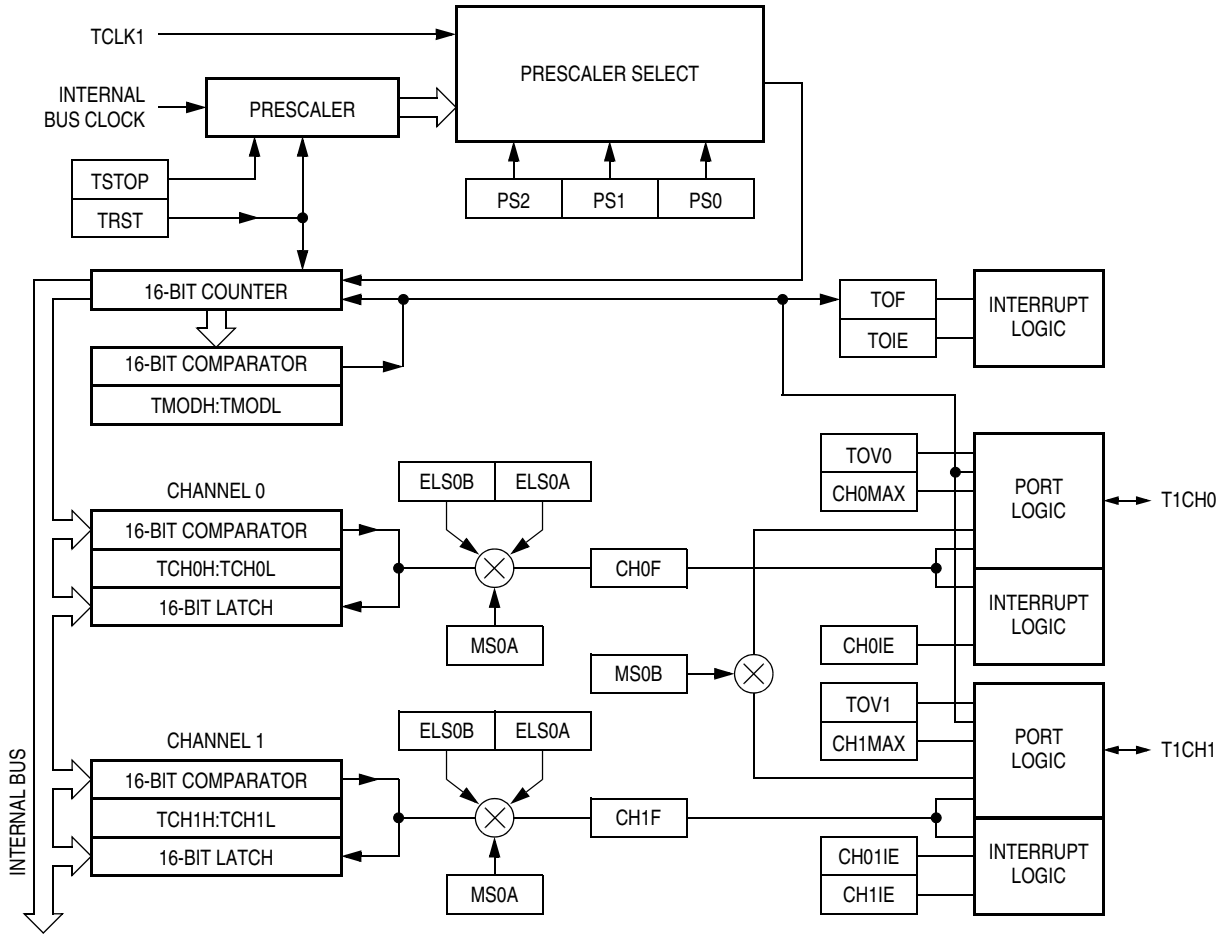


Figure 8-1. TIM Block Diagram

Figure 8-2 summarizes the timer registers.

NOTE

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000A	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
$000C	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
$000D	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
$000E	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
$000F	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
$0010	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
$0011	Timer 1 Channel 0 Register High (T1CH0H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0012	Timer 1 Channel 0 Register Low (T1CH0L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							
$0013	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
$0014	Timer 1 Channel 1 Register High (T1CH1H)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	Indeterminate after reset							
$0015	Timer 1 Channel 1 Register Low (T1CH1L)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	Indeterminate after reset							

= Unimplemented

Figure 8-2. TIM I/O Register Summary

8.4.1 TIM Counter Prescaler

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

8.4.2 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.

8.4.3 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.

8.4.3.1 Unbuffered Output Compare

Output compare channel 0 can generate unbuffered output compare pulses as described in 8.4.3 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 0:

- When changing to a smaller value, enable channel 0 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.

8.4.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel. 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.

8.4.4 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 8-3 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 8.9.1 TIM Status and Control Register.

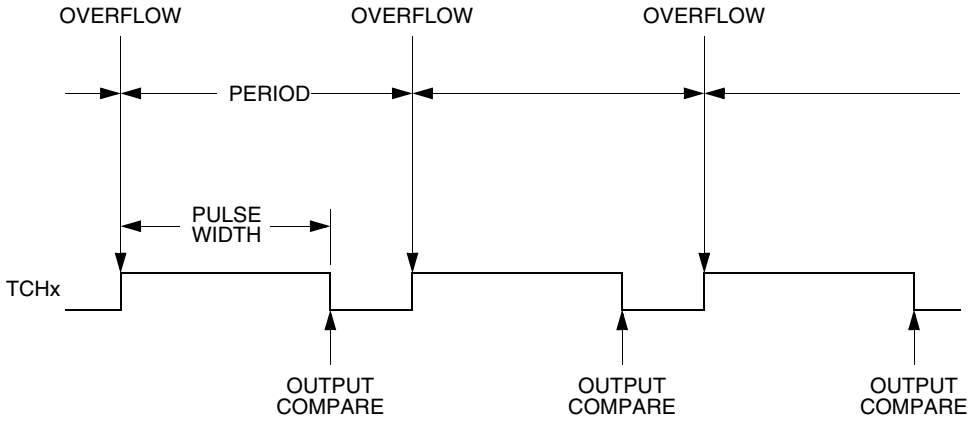


Figure 8-3. PWM Period and Pulse Width

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%.

8.4.4.1 Unbuffered PWM Signal Generation

Output compare channel 0 can generate unbuffered PWM pulses as described in 8.4.4 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 0:

- When changing to a shorter pulse width, enable channel 0 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.

8.4.4.2 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.

8.4.4.3 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 8-3.)
   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 8-3.)

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 8.9.4 TIM Channel Status and Control Registers.)

8.5 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.

8.6 Low-Power Modes

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

8.6.1 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.

8.6.2 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.

8.7 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 6.7.3 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 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.

8.8 I/O Signals

Port C shares three of its pins with the TIM. The two TIM channel I/O pins are PTC0/T1CH0 and PTC2/T1CH1; and the external clock input is PTC1/TCLK1.

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

8.8.1 TIM Clock Pin (PTC1/TCLK1)

PTC1/TCLK1 is an external clock input that can be the clock source for the TIM counter instead of the prescaled internal bus clock. Select the PTC1/TCLK1 input by writing logic 1's to the three prescaler select bits, PS[2:0]. (See 8.9.1 TIM Status and Control Register.) The minimum T2CLK pulse width, $TCLK1_{LMIN}$ or $TCLK1_{HMIN}$, is:

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

The maximum TCLK1 frequency is: bus frequency ÷ 2

8.9 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)

8.9.1 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: $000A

	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 8-4. 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 8-2 shows. Reset clears the PS[2:0] bits.

Table 8-2. 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	TCLK1

8.9.2 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: $000C

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

= Unimplemented

Figure 8-5. TIM Counter Registers High (TCNTH)

Address: $000D

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

= Unimplemented

Figure 8-6. TIM Counter Registers Low (TCNTL)

8.9.3 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: $000E

	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 8-7. TIM Counter Modulo Register High (TMODH)

Address: $000F

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

NOTE

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

8.9.4 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: $0010

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

Figure 8-9. TIM Channel 0 Status and Control Register (TSC0)

Address: $0013

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

= Unimplemented

Figure 8-10. 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 8-3.

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 8-3. 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 8-3 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 8-3. 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 8-11 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.

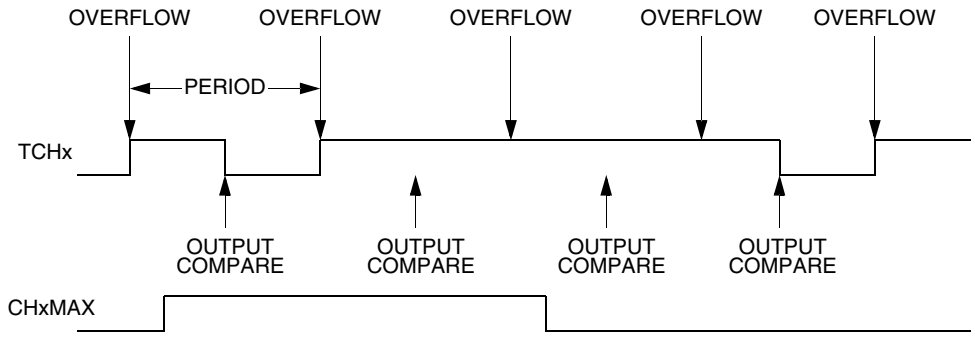


Figure 8-11. CHxMAX Latency

8.9.5 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: $0011

	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 8-12. TIM Channel 0 Register High (TCH0H)

Address: $0012

	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 8-13. TIM Channel 0 Register Low (TCH0L)

Address: $0014

	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 8-14. TIM Channel 1 Register High (TCH1H)

Address: $0015

	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 8-15. TIM Channel 1 Register Low (TCH1L)

Chapter 9
Timebase Module (TBM)

9.1 Introduction

This section describes the timebase module (TBM). The TBM will generate periodic interrupts at user selectable rates using a counter clocked by the selected OSCCLK clock from the oscillator module. This TBM version uses 18 divider stages, eight of which are user selectable.

9.2 Features

Features of the TBM module include:

- 88-kHz build-in RC clock.
- Software programmable ~3s, ~1.5s, ~745ms, ~372ms, ~186ms, ~93ms, ~47ms, and ~23ms periodic interrupt
- User selectable oscillator clock source enable during stop mode to allow periodic wake-up from stop

9.3 Functional Description

This module can generate a periodic interrupt by dividing the oscillator clock frequency, OSCCLK. The counter is initialized to all 0s when TBON bit is cleared. The counter, shown in Figure 9-1, starts counting when the TBON bit is set. When the counter overflows at the tap selected by TBR2:TBR0, the TBIF bit gets set. If the TBIE bit is set, an interrupt request is sent to the CPU. The TBIF flag is cleared by writing a 1 to the TACK bit. The first time the TBIF flag is set after enabling the timebase module, the interrupt is generated at approximately half of the overflow period. Subsequent events occur at the exact period.

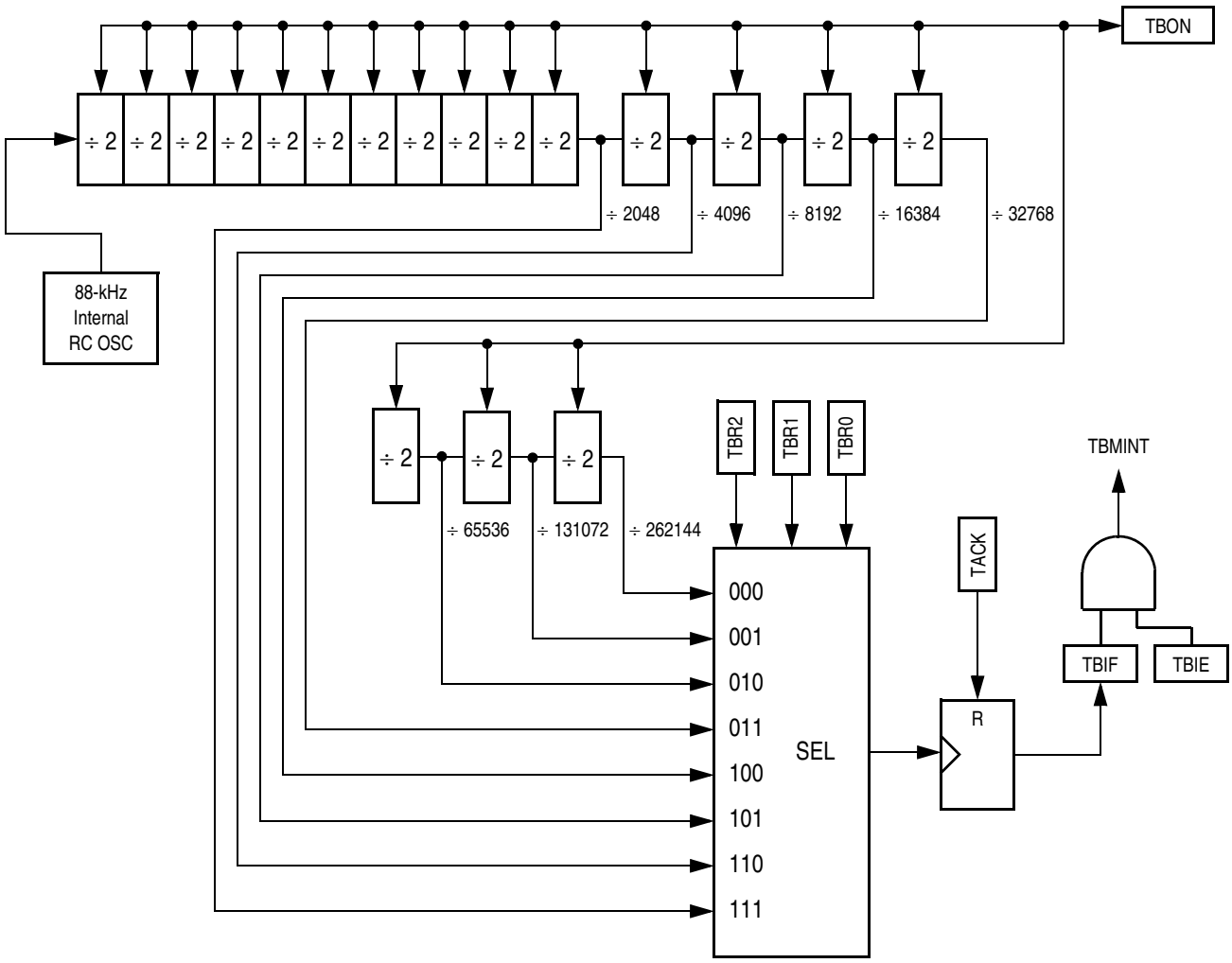


Figure 9-1. Timebase Block Diagram

9.4 Timebase Register Description

The timebase has one register, the TBCR, which is used to enable the timebase interrupts and set the rate.

Address: $0018

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	TBIF	TBR2	TBR1	TBR0	0	TBIE	TBON	R
Write:					TACK			
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 9-2. Timebase Control Register (TBCR)

TBIF — Timebase Interrupt Flag

This read-only flag bit is set when the timebase counter has rolled over.

1 = Timebase interrupt pending
0 = Timebase interrupt not pending

TBR2–TBR0 — Timebase Rate Selection

These read/write bits are used to select the rate of timebase interrupts as shown in Table 9-1.

Table 9-1. Timebase Rate Selection (88-kHz Reference)

TBR2	TBR1	TBR0	Divider	Timebase Interrupt Rate	
				Hz	ms
0	0	0	262144	~0.33	~2979
0	0	1	131072	~0.67	~1489
0	1	0	65536	~1.3	~745
0	1	1	32768	~2.7	~372
1	0	0	16384	~5.4	~186
1	0	1	8192	~10.7	~93
1	1	0	4096	~21.5	~47
1	1	1	2048	~43.0	~23

NOTE

Do not change TBR2–TBR0 bits while the timebase is enabled (TBON = 1).

TACK — Timebase ACKnowledge

The TACK bit is a write-only bit and always reads as 0. Writing a logic 1 to this bit clears TBIF, the timebase interrupt flag bit. Writing a logic 0 to this bit has no effect.

1 = Clear timebase interrupt flag
0 = No effect

TBIE — Timebase Interrupt Enabled

This read/write bit enables the timebase interrupt when the TBIF bit becomes set. Reset clears the TBIE bit.

1 = Timebase interrupt enabled
0 = Timebase interrupt disabled

TBON — Timebase Enabled

This read/write bit enables the timebase. Timebase may be turned off to reduce power consumption when its function is not necessary. The counter can be initialized by clearing and then setting this bit. Reset clears the TBON bit.

1 = Timebase enabled
0 = Timebase disabled and the counter initialized to 0s

9.5 Interrupts

The timebase module can interrupt the CPU on a regular basis with a rate defined by TBR2–TBR0. When the timebase counter chain rolls over, the TBIF flag is set. If the TBIE bit is set, enabling the timebase interrupt, the counter chain overflow will generate a CPU interrupt request. The interrupt vector is defined in Table 6-3. Interrupt Sources.

Interrupts must be acknowledged by writing a logic 1 to the TACK bit.

9.6 Low-Power Modes

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

9.6.1 Wait Mode

The timebase module remains active after execution of the WAIT instruction. In wait mode, the timebase register is not accessible by the CPU.

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

9.6.2 Stop Mode

The timebase module may remain active after execution of the STOP instruction if the oscillator has been enabled to operate during stop mode through the stop mode oscillator enable bit (STOP_RCLKEN) for the selected oscillator in the CONFIG2 register. The timebase module can be used in this mode to generate a periodic walk-up from stop mode.

If the oscillator has not been enabled to operate in stop mode, the timebase module will not be active during STOP mode. In stop mode the timebase register is not accessible by the CPU.

If the timebase functions are not required during stop mode, reduce the power consumption by stopping the timebase before enabling the STOP instruction.

Chapter 10
Serial Peripheral Interface Module (SPI)

10.1 Introduction

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

10.2 Features

Features of the SPI module include the following:

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

10.3 Pin Name Conventions and I/O Register Addresses

The text that follows describes the SPI. The SPI I/O pin names are $\overline{SS}$ (slave select), SPSCK (SPI serial clock), CGND (clock ground), MOSI (master out slave in), and MISO (master in/slave out). The SPI shares four I/O pins with four parallel I/O ports.

The full names of the SPI I/O pins are shown in Table 10-1. The generic pin names appear in the text that follows.

Table 10-1. Pin Name Conventions

SPI Generic Pin Names:		MISO	MOSI	$\overline{SS}$	SPSCK	CGND
Full SPI Pin Names:	SPI	PTE6/MISO	PTE5/MOSI	PTE7/$\overline{SS}$	PTE4/SPSCK	V_{SS}

Figure 10-1 summarizes the SPI I/O registers.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$004C	SPI Control Register (SPCR)	Read:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Write:								
		Reset:	0	0	1	0	1	0	0	0
$004D	SPI Status and Control Register (SPSCR)	Read:	SPRF	ERRIE	OVRF	MODF	SPTE	MODFEN	SPR1	SPR0
		Write:								
		Reset:	0	0	0	0	1	0	0	0
$004E	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							

= Unimplemented R = Reserved

Figure 10-1. SPI I/O Register Summary

10.4 Functional Description

Figure 10-2 shows the structure of the SPI module.

The SPI module allows full-duplex, synchronous, serial communication between the MCU and peripheral devices, including other MCUs. Software can poll the SPI status flags or SPI operation can be interrupt-driven.

The following paragraphs describe the operation of the SPI module.

10.4.1 Master Mode

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

NOTE

Configure the SPI modules as master or slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. (See 10.13.1 SPI Control Register.)

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the transmit data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE. The byte begins shifting out on the MOSI pin under the control of the serial clock. (See Figure 10-3.)

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See 10.13.2 SPI Status and Control Register.) Through the SPSCK pin, the baud rate generator of the master also controls the shift register of the slave peripheral.

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

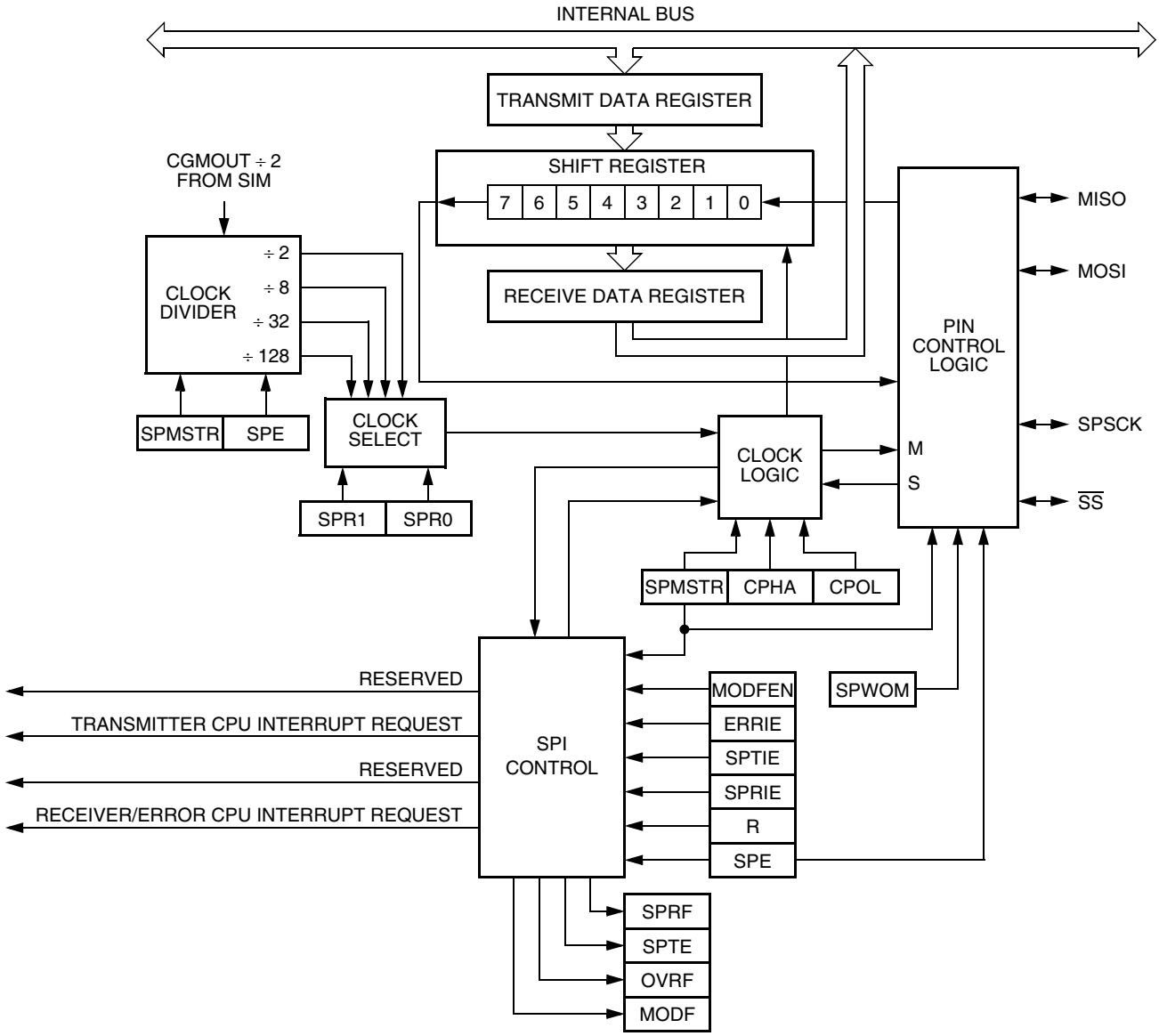


Figure 10-2. SPI Module Block Diagram

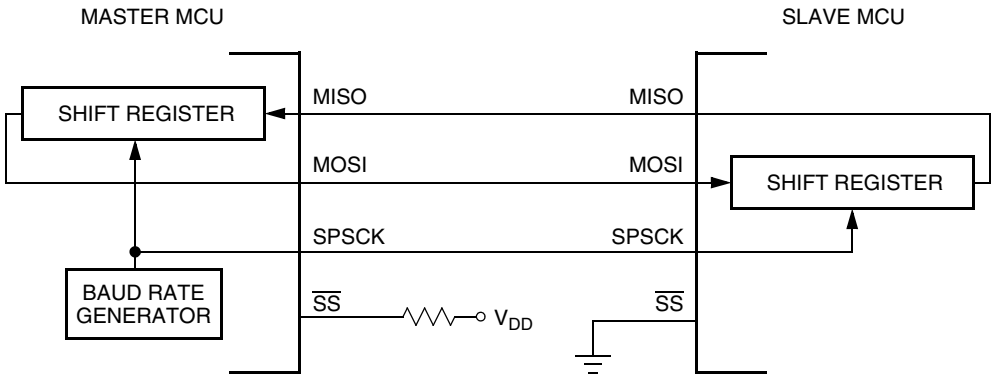


Figure 10-3. Full-Duplex Master-Slave Connections

10.4.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit is clear. In slave mode, the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave SPI must be at logic 0. $\overline{SS}$ must remain low until the transmission is complete. (See 10.7.2 Mode Fault Error.)

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

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

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

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. (See 10.5 Transmission Formats.)

NOTE

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

10.5 Transmission Formats

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

10.5.1 Clock Phase and Polarity Controls

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

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

NOTE

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

10.5.2 Transmission Format When CPHA = 0

Figure 10-4 shows an SPI transmission in which CPHA is logic 0. The figure should not be used as a replacement for data sheet parametric information.

Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 10.7.2 Mode Fault Error.) When CPHA = 0, the first SPSCK edge is the MSB capture strobe. Therefore, the slave must begin driving its data before the first SPSCK edge, and a falling edge on the $\overline{SS}$ pin is used to start the slave data transmission. The slave's $\overline{SS}$ pin must be toggled back to high and then low again between each byte transmitted as shown in Figure 10-5.

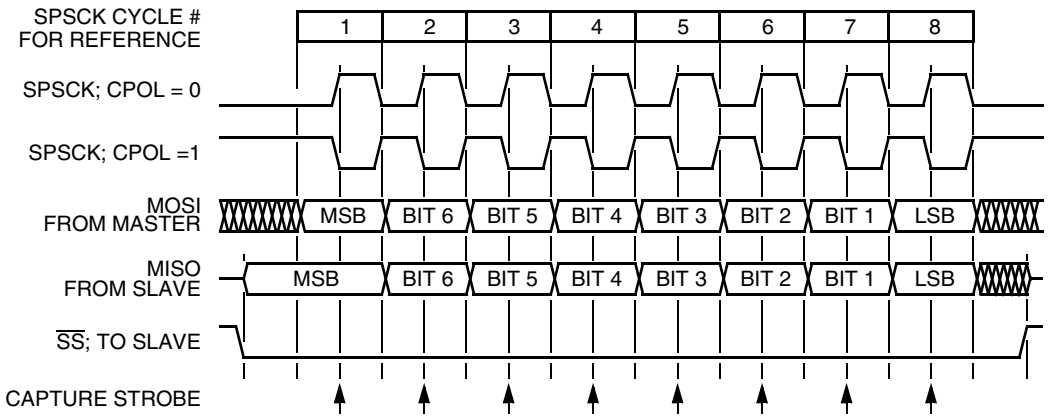


Figure 10-4. Transmission Format (CPHA = 0)

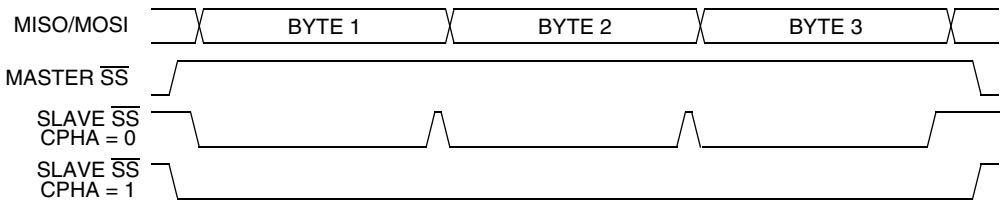


Figure 10-5. CPHA/$\overline{SS}$ Timing

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

10.5.3 Transmission Format When CPHA = 1

Figure 10-6 shows an SPI transmission in which CPHA is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SPSCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SPSCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is at logic 0, so that only the selected slave drives to the master. The $\overline{SS}$ pin of the master is not shown but is assumed to be inactive. The $\overline{SS}$ pin of the master must be high or must be reconfigured as general-purpose I/O not affecting the SPI. (See 10.7.2 Mode Fault Error.) When CPHA = 1, the master begins driving its MOSI pin on the first SPSCK edge. Therefore, the slave uses the first SPSCK edge as a start transmission signal. The $\overline{SS}$ pin can remain low between transmissions. This format may be preferable in systems having only one master and only one slave driving the MISO data line.

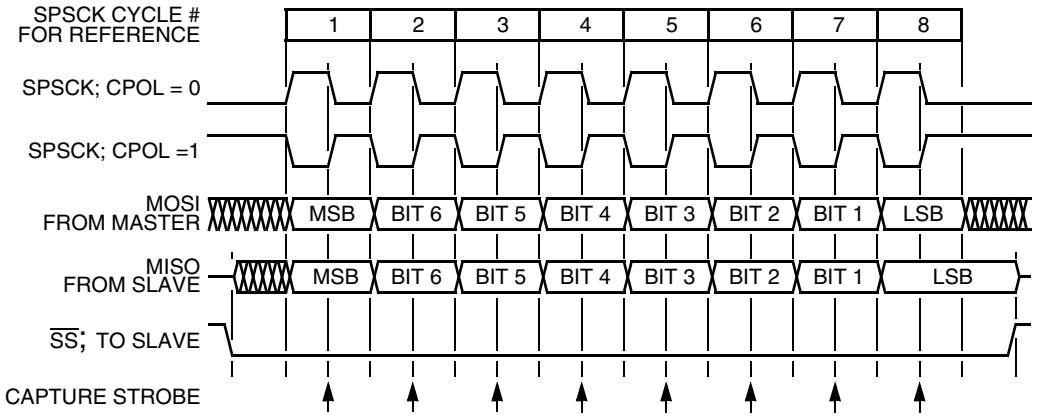


Figure 10-6. Transmission Format (CPHA = 1)

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

10.5.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), writing to the SPDR starts a transmission. CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SPSCK signal. When CPHA = 0, the SPSCK signal remains inactive for the first half of the first SPSCK cycle. When CPHA = 1, the first SPSCK cycle begins with an edge on the SPSCK line from its inactive to its active level. The SPI clock rate (selected by SPR1:SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See Figure 10-7.) The internal SPI clock in the master is a free-running derivative of the internal MCU clock. To conserve power, it is enabled only when both the SPE and SPMSTR bits are set. SPSCK edges occur halfway through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR occurs relative to the slower SPSCK. This uncertainty causes the variation in the initiation delay shown in Figure 10-7. This delay is no longer than a single SPI bit time. That is, the maximum delay is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

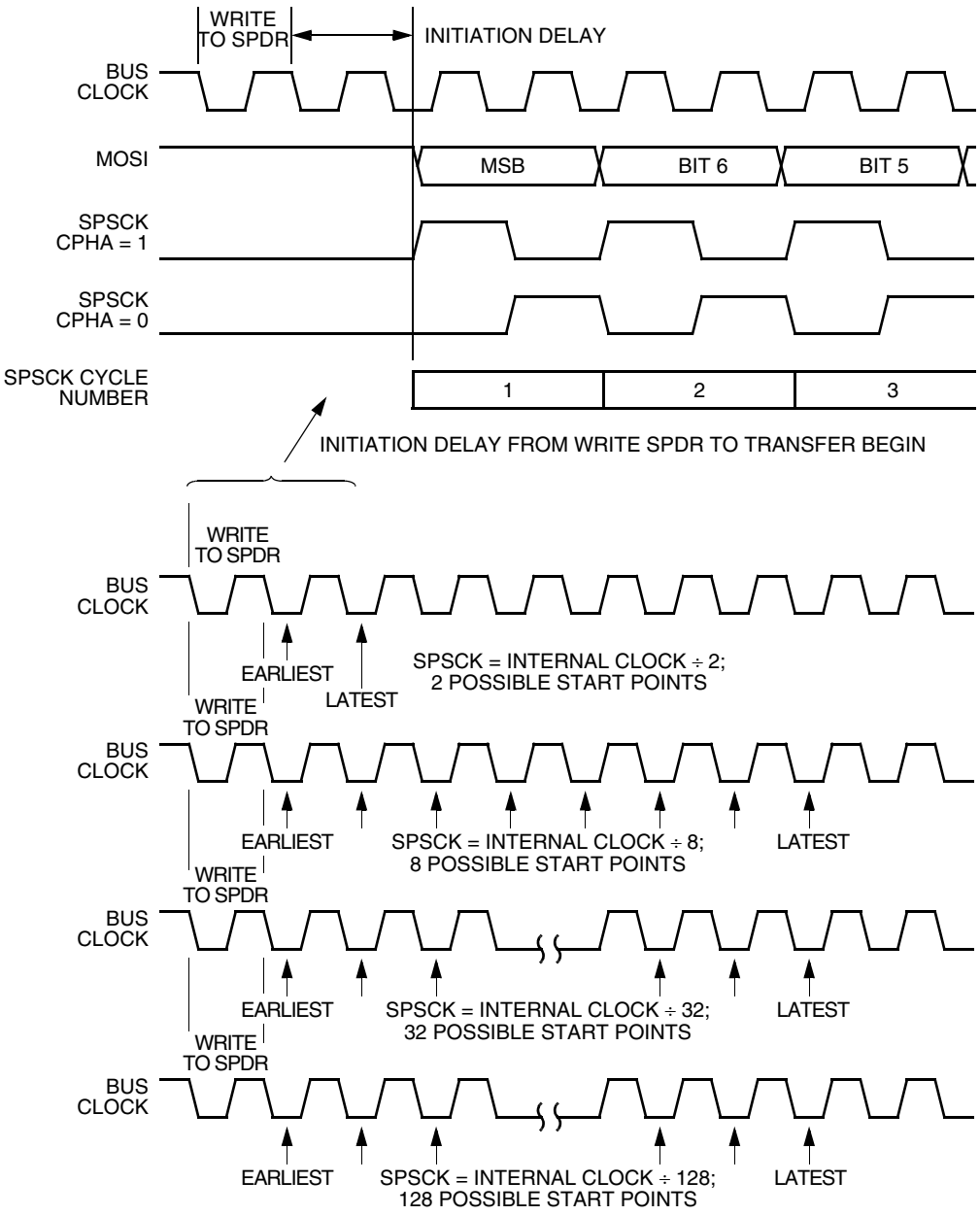


Figure 10-7. Transmission Start Delay (Master)

10.6 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE) indicates when the transmit data buffer is ready to accept new data. Write to the transmit data register only when the SPTE bit is high. Figure 10-8 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

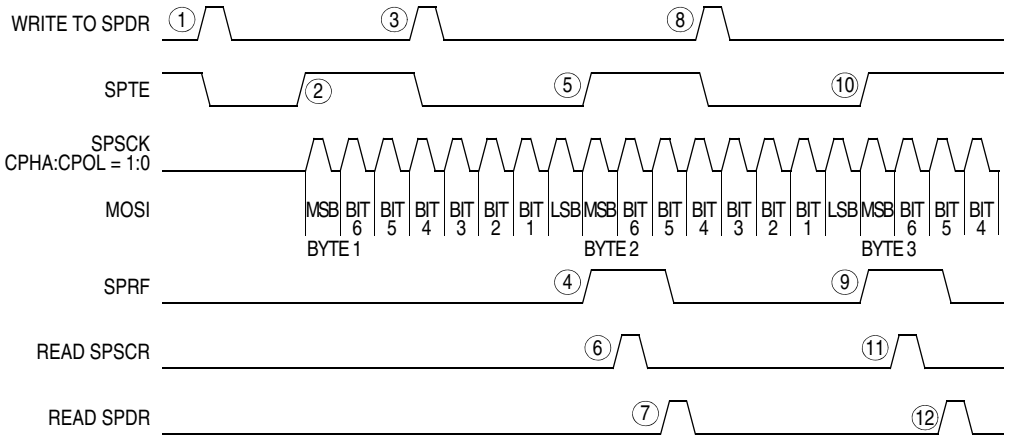


Figure 10-8. SPRF/SPTE CPU Interrupt Timing

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

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

10.7 Error Conditions

The following flags signal SPI error conditions:

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

10.7.1 Overflow Error

The overflow flag (OVRF) becomes set if the receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. The bit 1 capture strobe occurs in the middle of SPSCK cycle 7. (See Figure 10-4 and Figure 10-6.) If an overflow occurs, all data received after the overflow and before the OVRF bit is cleared does not transfer to the receive data register and does not set the SPI receiver full bit (SPRF). The unread data that transferred to the receive data register before the overflow occurred can still be read. Therefore, an overflow error always indicates the loss of data. Clear the overflow flag by reading the SPI status and control register and then reading the SPI data register.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. (See Figure 10-11.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. Figure 10-9 shows how it is possible to miss an overflow. The first part of Figure 10-9 shows how it is possible to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF bit can be set in between the time that SPSCR and SPDR are read.

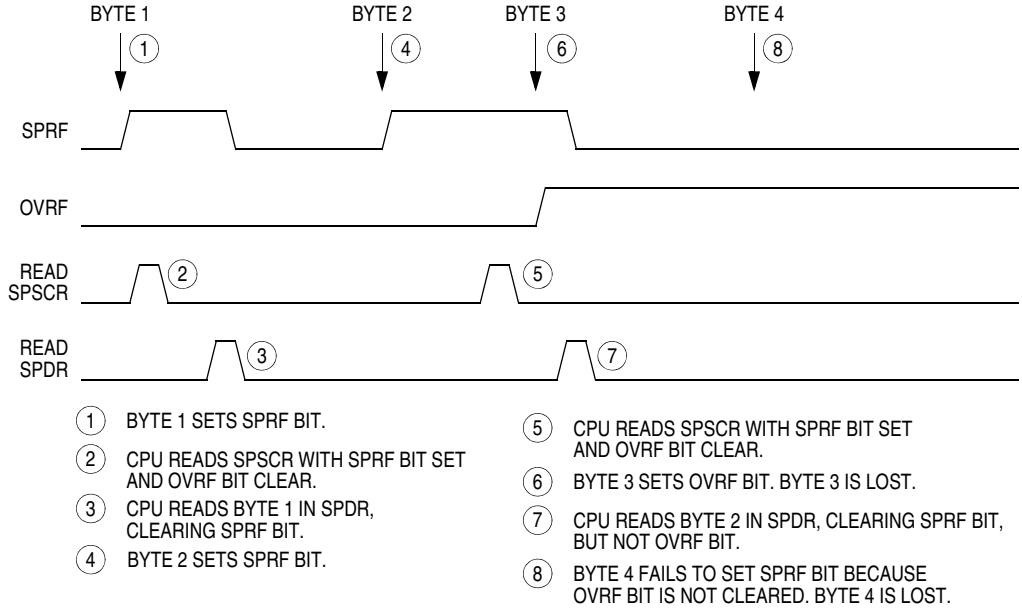


Figure 10-9. Missed Read of Overflow Condition

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

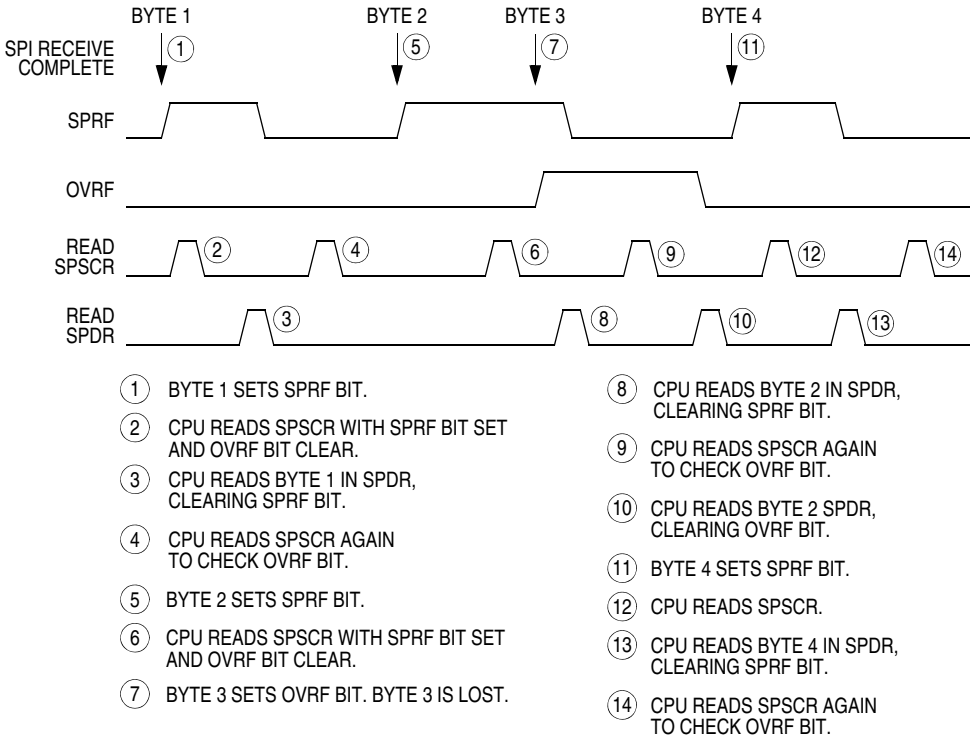


Figure 10-10. Clearing SPRF When OVRF Interrupt Is Not Enabled

10.7.2 Mode Fault Error

Setting the SPMSTR bit selects master mode and configures the SPSCK and MOSI pins as outputs and the MISO pin as an input. Clearing SPMSTR selects slave mode and configures the SPSCK and MOSI pins as inputs and the MISO pin as an output. The mode fault bit, MODF, becomes set any time the state of the slave select pin, $\overline{SS}$, is inconsistent with the mode selected by SPMSTR.

To prevent SPI pin contention and damage to the MCU, a mode fault error occurs if:

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

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. (See Figure 10-11.) It is not possible to enable MODF or OVRF individually to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

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

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

NOTE

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

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK goes back to its idle level following the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its idle level following the shift of the last data bit. (See 10.5 Transmission Formats.)

NOTE

Setting the MODF flag does not clear the SPMSTR bit. The SPMSTR bit has no function when SPE = 0. Reading SPMSTR when MODF = 1 shows the difference between a MODF occurring when the SPI is a master and when it is a slave.

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

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

NOTE

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

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

10.8 Interrupts

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

Table 10-2. SPI Interrupts

Flag	Request
SPTE Transmitter empty	SPI transmitter CPU interrupt request (SPTIE = 1, SPE = 1)
SPRF Receiver full	SPI receiver CPU interrupt request (SPRIE = 1)
OVRF Overflow	SPI receiver/error interrupt request (ERRIE = 1)
MODF Mode fault	SPI receiver/error interrupt request (ERRIE = 1)

Reading the SPI status and control register with SPRF set and then reading the receive data register clears SPRF. The clearing mechanism for the SPTE flag is always just a write to the transmit data register.

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

The SPI receiver interrupt enable bit (SPRIE) enables the SPRF bit to generate receiver CPU interrupt requests, regardless of the state of the SPE bit. (See Figure 10-11.)

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

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

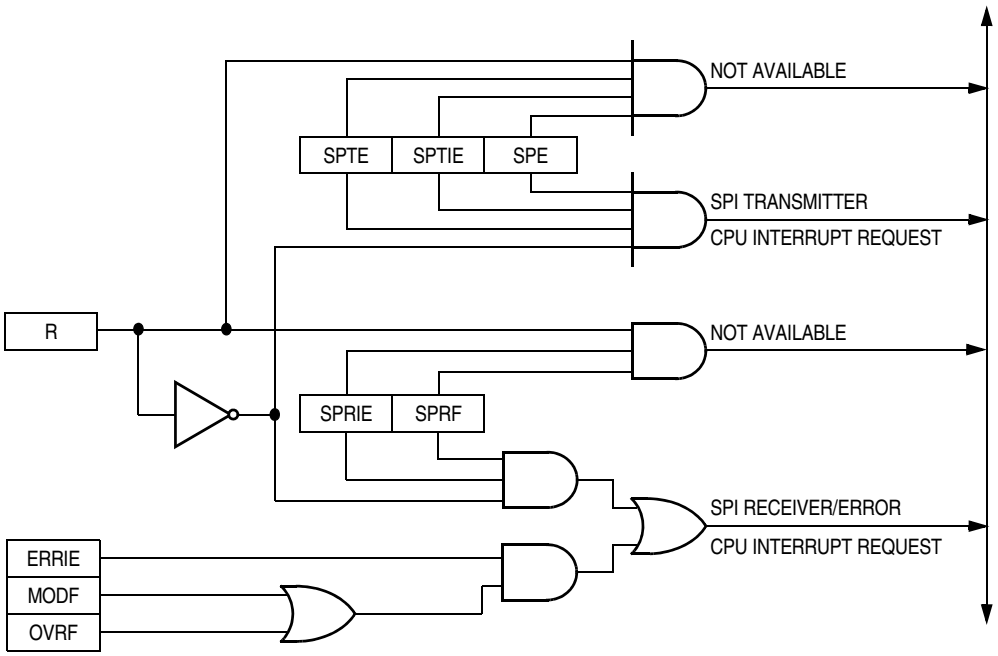


Figure 10-11. SPI Interrupt Request Generation

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

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

10.9 Resetting the SPI

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

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

These items are reset only by a system reset:

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

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

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

10.10 Low-Power Modes

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

10.10.1 Wait Mode

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

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

To exit wait mode when an overflow condition occurs, enable the OVRF bit to generate CPU interrupt requests by setting the error interrupt enable bit (ERRIE). (See 10.8 Interrupts.)

10.10.2 Stop Mode

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

10.11 SPI During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See Chapter 6 System Integration Module (SIM).)

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

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

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

10.12 I/O Signals

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

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select
- CGND — Clock ground (internally connected to V_{SS})

The SPI has limited inter-integrated circuit (I^2C) capability (requiring software support) as a master in a single-master environment. To communicate with I^2C peripherals, MOSI becomes an open-drain output when the SPWOM bit in the SPI control register is set. In I^2C communication, the MOSI and MISO pins are connected to a bidirectional pin from the I^2C peripheral and through a pullup resistor to V_{DD}.

10.12.1 MISO (Master In/Slave Out)

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

Slave output data on the MISO pin is enabled only when the SPI is configured as a slave. The SPI is configured as a slave when its SPMSTR bit is logic 0 and its $\overline{SS}$ pin is at logic 0. To support a multiple-slave system, a logic 1 on the $\overline{SS}$ pin puts the MISO pin in a high-impedance state.

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

10.12.2 MOSI (Master Out/Slave In)

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

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

10.12.3 SPSCK (Serial Clock)

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

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

10.12.4 $\overline{SS}$ (Slave Select)

The $\overline{SS}$ pin has various functions depending on the current state of the SPI. For an SPI configured as a slave, the $\overline{SS}$ is used to select a slave. For CPHA = 0, the $\overline{SS}$ is used to define the start of a transmission. (See 10.5 Transmission Formats.) Since it is used to indicate the start of a transmission, the $\overline{SS}$ must be toggled high and low between each byte transmitted for the CPHA = 0 format. However, it can remain low between transmissions for the CPHA = 1 format. See Figure 10-12.

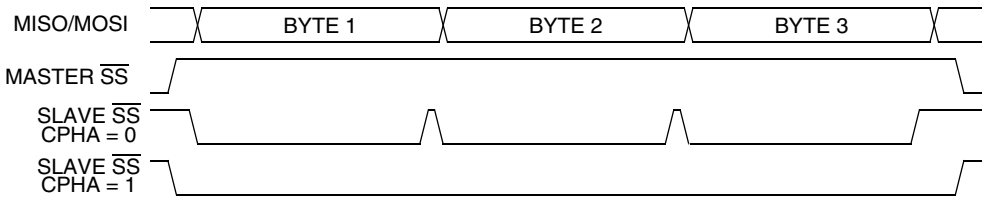


Figure 10-12. CPHA/$\overline{SS}$ Timing

When an SPI is configured as a slave, the $\overline{SS}$ pin is always configured as an input. It cannot be used as a general-purpose I/O regardless of the state of the MODFEN control bit. However, the MODFEN bit can still prevent the state of the $\overline{SS}$ from creating a MODF error. (See 10.13.2 SPI Status and Control Register.)

NOTE

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

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See 10.7.2 Mode Fault Error.) For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the port data register. (See Table 10-3.)

Table 10-3. SPI Configuration

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

Note 1. X = Don't care

10.12.5 CGND (Clock Ground)

CGND is the ground return for the serial clock pin, SPSCK, and the ground for the port output buffers. It is internally connected to V_{SS} as shown in Table 10-1.

10.13 I/O Registers

Three registers control and monitor SPI operation:

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

10.13.1 SPI Control Register

The SPI control register:

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

Address: $004C

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

R = Reserved

Figure 10-13. SPI Control Register (SPCR)

SPRIE — SPI Receiver Interrupt Enable Bit

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

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

SPMSTR — SPI Master Bit

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

1 = Master mode
0 = Slave mode

CPOL — Clock Polarity Bit

This read/write bit determines the logic state of the SPSCK pin between transmissions. (See Figure 10-4 and Figure 10-6.) To transmit data between SPI modules, the SPI modules must have identical CPOL values. Reset clears the CPOL bit.

CPHA — Clock Phase Bit

This read/write bit controls the timing relationship between the serial clock and SPI data. (See Figure 10-4 and Figure 10-6.) To transmit data between SPI modules, the SPI modules must have identical CPHA values. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic 1 between bytes. (See Figure 10-12.) Reset sets the CPHA bit.

SPWOM — SPI Wired-OR Mode Bit

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

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

SPE — SPI Enable

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI. (See 10.9 Resetting the SPI.) Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

SPTIE— SPI Transmit Interrupt Enable

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

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

10.13.2 SPI Status and Control Register

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

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

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

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

Address $004D

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

= Unimplemented

Figure 10-14. SPI Status and Control Register (SPSCR)

SPRF — SPI Receiver Full Bit

This clearable, read-only flag is set each time a byte transfers from the shift register to the receive data register. SPRF generates a CPU interrupt request if the SPRIE bit in the SPI control register is set also. During an SPRF CPU interrupt, the CPU clears SPRF by reading the SPI status and control register with SPRF set and then reading the SPI data register. Reset clears the SPRF bit.

1 = Receive data register full
0 = Receive data register not full

ERRIE — Error Interrupt Enable Bit

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

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

OVRF — Overflow Bit

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

1 = Overflow
0 = No overflow

MODF — Mode Fault Bit

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

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

SPTE — SPI Transmitter Empty Bit

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

NOTE

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

During an SPTE CPU interrupt, the CPU clears the SPTE bit by writing to the transmit data register. Reset sets the SPTE bit.

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

MODFEN — Mode Fault Enable Bit

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

If the MODFEN bit is set, then this pin is not available as a general-purpose I/O. When the SPI is enabled as a slave, the $\overline{SS}$ pin is not available as a general-purpose I/O regardless of the value of MODFEN. (See 10.12.4 SS (Slave Select).)

If the MODFEN bit is low, the level of the $\overline{SS}$ pin does not affect the operation of an enabled SPI configured as a master. For an enabled SPI configured as a slave, having MODFEN low only prevents the MODF flag from being set. It does not affect any other part of SPI operation. (See 10.7.2 Mode Fault Error.)

SPR1 and SPR0 — SPI Baud Rate Select Bits

In master mode, these read/write bits select one of four baud rates as shown in Table 10-4. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 10-4. SPI Master Baud Rate Selection

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

Use this formula to calculate the SPI baud rate:

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

where:

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

10.13.3 SPI Data Register

The SPI data register consists of the read-only receive data register and the write-only transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate registers that can contain different values. (See Figure 10-2.)

Address: $004E

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

Figure 10-15. SPI Data Register (SPDR)

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

NOTE

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

Chapter 11
USB 2.0 FS Module

11.1 Introduction

This section describes the universal serial bus (USB) module. The USB module is designed to serve as a full speed (FS) USB device per the *Universal Serial Bus Specification Rev 2.0.* Control and interrupt data transfers are supported. Endpoint 0 functions as a transmit/receive control endpoint; endpoint 1, 2, 3 and 4 functions are configurable as interrupt or bulk endpoints and support transmit or receive communication.

11.2 Features

Features of the USB module include:

- Full Universal Serial Bus Specification 2.0 full-speed functions
- 12Mbps data rate
- On-chip 3.3V regulator
- Endpoint 0 with 8-byte transmit buffer and 8-byte receive buffer
- 64 bytes programmable buffer to share with 4 data endpoint
- 4 data endpoints supports
- USB device controller with protocol control supports single configuration, 2 interfaces and no alternate settings for each interface
- Programmable endpoint type for four independent endpoints — interrupt or bulk
- USB data control logic:
  - Packet identification and decoding/generation
  - CRC generation and checking
  - NRZI (Non-Return-to Zero Inserted) encoding/decoding
  - Bit-stuffing
  - Sync detection
  - End-of-packet detection
- USB reset options:
  - Internal MCU reset generation
  - CPU interrupt request generation
- Suspend and resume operations, with remote wakeup support

11.3 USB Module Architecture

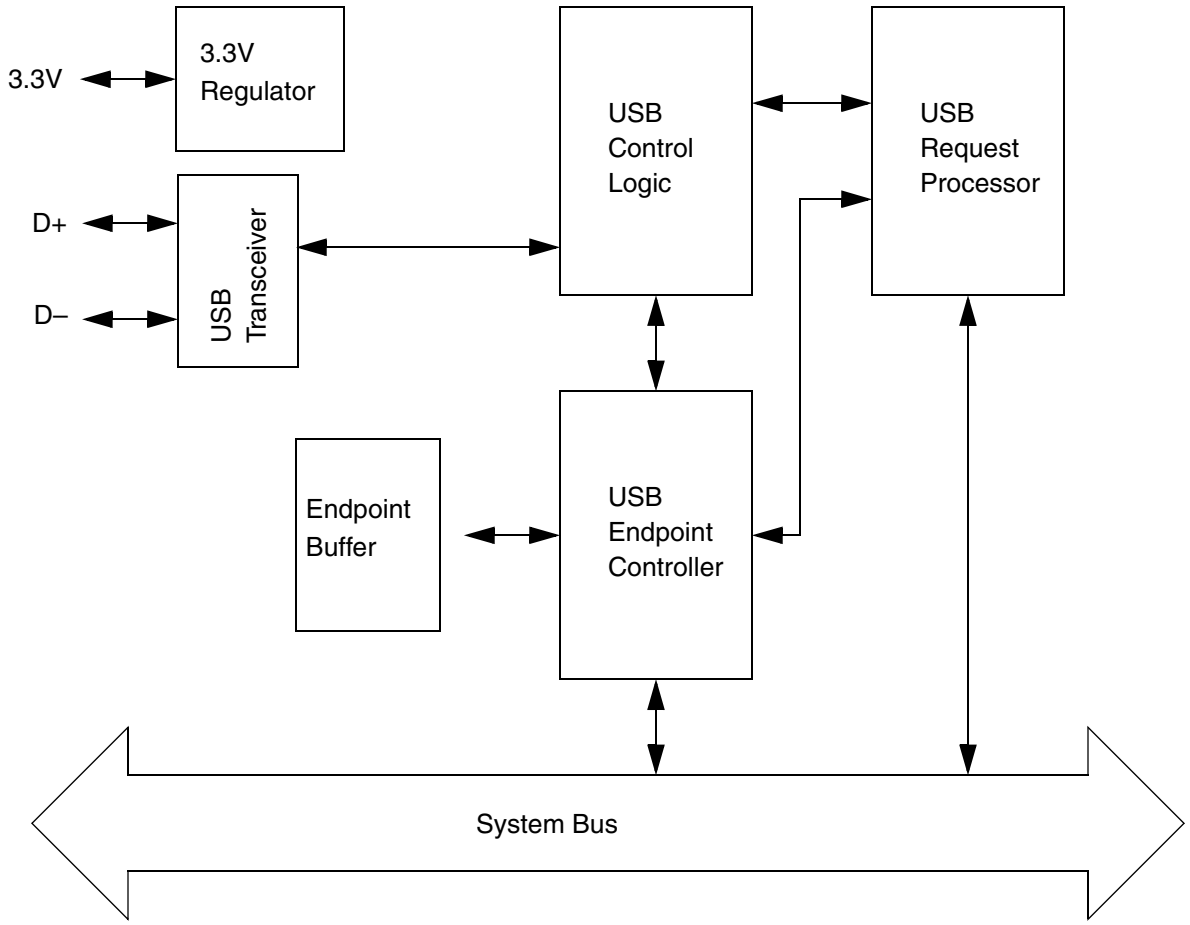


Figure 11-1. USB Module Block Diagram

The USB module block diagram is shown in Figure 11-1. The module involves six major blocks - USB transceiver, USB control logic, USB request processor, USB endpoint controller and USB Endpoint buffer.

11.3.1 USB Transceiver

The USB transceiver is electrically compliant to the *Universal Serial Bus Specification 2.0.* The on-chip 3.3V regulator provides a stable power source for the termination pull-up resistor. Full speed devices are terminated with the pull-up resistor on the D+ line.

11.3.2 USB Control Logic

The USB control logic handle the following functions:

- For transmit data
  - Packet creation
  - CRC generation
  - NRZI encoding
  - Bit stuffing
- For receive data
  - Sync detection
  - Packet Identification
  - End-of-packet (EOP) detection
  - CRC validation
  - NRZI decoding
  - Bit unstuffing
- For error detection
  - Bad CRC
  - Timeout detection for EOP
  - Bit stuffing violation

11.3.3 USB Endpoint Configuration

A single configuration and 2 interfaces are supported by the module. Endpoint 0 is always used as control endpoint. The interface number for endpoint 1 to 4 are programmable through USB interface control register (UINTFCR). The endpoint type and direction of all endpoint 1 to 4 is software programmable to either BULK or INTERRUPT and either IN or OUT respectively. The endpoint configuration is summarized in Table 11-1

Table 11-1. Endpoint Summary

Endpoint Number	Configuration Number	Interface Number	Direction	Type
0	—	—	IN/OUT	Control
1	1	EP1INT	IN/OUT	Bulk/ Interrupt
2	1	EP2INT	IN/OUT	Bulk/ Interrupt
3	1	EP3INT	IN/OUT	Bulk/ Interrupt
4	1	EP4INT	IN/OUT	Bulk/ Interrupt

11.3.4 USB Requestor Processor

The USB requestor processor automatically process some standard USB requests as listed in Table 11-2.

Table 11-2. USB Requests Handling

Request	Handling
CLEAR_FEATURE	Requestor processor clears the feature specified by the feature selector. For USB specification 2.0, only two features are specified - DEVICE_REMOTE_WAKEUP and ENDPOINT_HALT. The module stores the HALT status and remote wakeup status internally. No user attention is required.
GET_CONFIGURATION	Return the configuration number specified in CONFIG. No user notification is provided.
GET_DESCRIPTOR	This requests is not handled automatically by the request processor. User is notified by the SETUP flag and the TFRC_OUT flag being set. The request command can be decoded through the 8-byte endpoint 0 OUT data buffer. User must fill up the Endpoint 0 data IN buffer 8 bytes at a time manually with the corresponding descriptor requested by the host. When DVALID_IN bit is set and TFRC_IN flag is cleared, the requestor processor responses to the next IN packet with the data stored. Before the DVALID_IN bit is set, NAK is returned to all IN packet.
GET_INTERFACE	No alternate setting is support by the module, Alternative interface number zero is always return. No user notification is provided.
GET_STATUS	Returns the current status of the specified device, endpoint or interface. No user notification is provided.
SET_ADDRESS	Internal address register is modified. The control logic begins responding to the new address once the status stage of the request completes successfully. No user notification is provided.
SET_CONFIGURATION	If the configuration value is zero, the USB module is placed into the unconfigured state. If the device is successfully configured by the host, the new configuration value is specified by CONFIG bit. NOTE: User is notified if the request completes successfully. CONFIG_CHG flag of USB Status Register (USBSR) will be set upon a successful completion of the request where the configuration number is changed from zero to one. User can read the CONFIG flag for the corresponding changes.
SET_DESCRIPTOR	Not supported. STALL packet is returned to the host.
SET_FEATURE	Corresponding feature specified by the packet is enabled accordingly. No user attention is required.
SET_INTERFACE	No alternative setting is supported. If the alternative setting number is zero, ACK is returned to the host. No user notification is provided.
SYNC_FRAME	Passed to the user as a vendor specific request. NOTE: SETUP flag, TFRC_OUT flag and DVALID_OUT flag will be set. User should decode the request via reading the endpoint 0 data registers (UE0D0-UE0D7).

11.3.4.1 Configuration Process

All USB devices must be configured before used. The host will configure the device according to the configuration process defined by the USB specification 2.0 Chapter 9. During the process most of the USB commands issued by the host are responded automatically except GET_DESCRIPTOR, SYNC_FRAME, vendor specific and class specific commands where user interaction is required. These are known as the user commands. The number of configurations and interfaces supported is limited by the module. This module can support a single configuration and maximum of two interfaces. No alternate setting is allowed.

Upon the reception of the user commands, no module level decoding is done instead user is notified by the SETUP flag and TFRC_OUT flag. User can then decode the command through the dedicated 8-byte endpoint 0 buffer. For instance, when a valid GET_DESCRIPTOR command is detected, user is notified by the SETUP, TFRC_OUT flag and DVALID_OUT flag. User should decode the command via the 8-byte endpoint 0 OUT buffer. Corresponding return descriptor is written to the endpoint 0 IN buffer 8 bytes at a time. By setting the DVALID_IN bit, the data is sent to the host in the next IN packet. Otherwise, the module will return NAK to all IN packet. If ACK is not returned from the host, the data is re-sent automatically in the next IN packet until ACK is returned from the host, then transfer complete flag TFRC_IN is set, the next 8 bytes of data can be written to the endpoint 0 IN buffer. The process continues until the requested descriptor is sent completely.

NOTE

Please note the module will return ACK to all valid SETUP packet. No software attention is required.

Endpoint 0 buffer and endpoint 0 data size register (DSIZE) will be updated on every incoming SETUP packet. However, SETUP or TFRC_OUT will not be set unless the SETUP packet is a valid GET_DESCRIPTOR, SYNC_FRAME or class/vendor specific SETUP command.

11.3.4.2 Control Endpoint 0

Endpoint 0 is always treated as control endpoint. It has eight bytes dedicated buffer for device transmit (IN packet) and eight bytes dedicated buffer for device receive (OUT packet). Most of the host requests is handled by the requestor processor excepts the class/vendor specified request, GET_DESCRIPTOR request and the SYNC_FRAME request. If the user is notified by the module about the arrival of such requests, user can decode the request command by reading the endpoint 0 data register.

The SETUP flag will be set if the 8-byte setup packet is received without CRC/Token/EOP error for Vendor/Class/SetDescriptor/SynchFrame commands only.

NOTE

For any OUT data received in the 8-byte endpoint OUT buffer, they are only valid until the start of any SETUP packet addressed to the device, even if the packet is corrupted the 8-byte OUT buffer may still be overwritten by this new SETUP packet. There is no indication of the corruption built into this module.

11.3.5 Endpoint Controller

The module has four independent endpoint controllers that managed the data transfer between CPU and the USB host. Each of these endpoint can be configured to either one of the two modes - bulk or interrupt.

There are 64-byte RAM buffer to share between the four data endpoints. User is required to specify the buffer base address and the buffer size for each endpoint used. The buffer is separated in 8 bytes page, therefore, there are 8 pages in total. For example, if 16 bytes of buffer is required for endpoint 1 and 16 bytes is required for endpoint 2, the buffer base address for endpoint 1 can be specified as %000, while the buffer base address for endpoint 2 can be specified %010 and the buffer size SIZE[1:0] register should be defined as %01 and %01 respectively.

11.3.5.1 OUT endpoint Data Transfer

The buffer size assigned to the endpoint is required to match with the endpoint definition specified in the endpoint descriptor. On every packet of data transfer, data loaded to the endpoint buffers are started with the buffer base address. If the data is valid, the complete packet is downloaded to the buffer RAM and ACK is sent automatically. The packet size is reported to DSIZE register and the transfer complete flag (TFRC) is set. User should wait until the data valid bit (DVALID) to be set before reading the data from the buffers. Otherwise, if CRC error encountered, the data packet is ignored and no handshake is returned.

11.3.5.2 IN endpoint Data Transfer

When IN packet is received by the module and DVALID bit is cleared, NAK is returned. If the IN packet is corresponding to endpoint 0, user is required to write data to the dedicate 8 bytes registers, then DSIZE should be updated before setting DVALID bit to send data via the next IN packet.

If the IN packet is corresponding to other endpoint 1 to 4, user is required to write data to corresponding endpoint buffer indicated by the BASE pointer.

When the packet is transmitted successfully that ACK is returned from the host, DVALID bit is returned to zero. Transfer complete flag (TFRC) is set to notify user for the next transfer.

11.4 Interrupt Source

There are two interrupt source reserved for the USB module.

Table 11-3. Interrupt Source Table

Flag	Interrupt Source
TFRC0_IN	USB Endpoint Interrupt
TFRC0_OUT	USB Endpoint Interrupt
TFRC1	USB Endpoint Interrupt
TFRC2	USB Endpoint Interrupt
TFRC3	USB Endpoint Interrupt
TFRC4	USB Endpoint Interrupt
SETUP	USB System Interrupt
SOF	USB System Interrupt
CONFIG_CHG	USB System Interrupt
USBRST	USB System Interrupt
RESUMEF	USB System Interrupt
SUSPND	USB System Interrupt

11.5 USB Module Registers

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0051	USB Control Register (USBCR)	Read:	USBEN	USBCLKEN	TFC4IE	TFC3IE	TFC2IE	TFC1IE	TFC0IE	0
		Write:								RESUME
		Reset:	0	0	0	0	0	0	0	0
$0052	USB Status Register (USBSR)	Read:	CONFIG		SETUP	SOF	CONFIG_ CHG	USBRST	RESUMEF	SUSPND
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0053	USB Status Interrupt Mask Register (USIMR)	Read:	R	0	SETUPIE	SOFIE	CONFIG_ CHGIE	USBRE- SETIE	RESUME- FIE	SUSPNDIE
		Write:		EP0_STALL						
		Reset:	0	0	0	0	0	0	0	0
$0054	USB EPO Control/Status Register (UEP0CSR)	Read:	DSIZE3_ OUT	DSIZE2_ OUT	DSIZE1_ OUT	DSIZE0_ OUT	DVALID_IN	TFRC_IN	DVALID_ OUT	TFRC_OUT
		Write:	DSIZE3_IN	DSIZE2_IN	DSIZE1_IN	DSIZE0_IN				
		Reset:	0	0	0	0	0	0	0	0
$0055	USB EP1 Control/Status Register (UEP1CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0056	USB EP2 Control/Status Register (UEP2CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0057	USB EP3 Control/Status Register (UEP3CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0058	USB EP4 Control/Status Register (UEP4CSR)	Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
		Write:			STALL					
		Reset:	0	0	0	0	0	0	0	0
$0059	USB EP1 Data Size Register (UEP1DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005A	USB EP2 Data Size Register (UEP2DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005B	USB EP3 Data Size Register (UEP3DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005C	USB EP4 Data Size Register (UEP4DSR)	Read:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected by reset R = Reserved

Figure 11-2. USB Registers

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$005D	USB EP 1/2 Base Pointer Register (UEP12BPR)	Read:		BASE22	BASE21	BASE20		BASE12	BASE11	BASE10
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005E	USB EP 3/4 Base Pointer Register (UEP34BPR)	Read:		BASE42	BASE41	BASE40		BASE32	BASE31	BASE30
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$005F	USB Interface Control Register (UINTFCR)	Read:		EP4INT		EP3INT		EP2INT		EP1INT
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0040	USB Endpoint 0 Data Register 0 (UE0D0)	Read:	UE0D07_ OUT	UE0D06_ OUT	UE0D05_ OUT	UE0D04_ OUT	UE0D03_ OUT	UE0D02_ OUT	UE0D01_ OUT	UE0D00_ OUT
		Write:	UE0D07_IN	UE0D06_IN	UE0D05_IN	UE0D04_IN	UE0D03_IN	UE0D02_IN	UE0D01_IN	UE0D00_IN
		Reset:	Unaffected by reset							
$0041	USB Endpoint 0 Data Register 1 (UE0D1)	Read:	UE0D17_ OUT	UE0D16_ OUT	UE0D15_ OUT	UE0D14_ OUT	UE0D13_ OUT	UE0D12_ OUT	UE0D11_ OUT	UE0D10_ OUT
		Write:	UE0D17_IN	UE0D16_IN	UE0D15_IN	UE0D14_IN	UE0D13_IN	UE0D12_IN	UE0D11_IN	UE0D10_IN
		Reset:	Unaffected by reset							
$0042	USB Endpoint 0 Data Register 2 (UE0D2)	Read:	UE0D27_ OUT	UE0D26_ OUT	UE0D25_ OUT	UE0D24_ OUT	UE0D23_ OUT	UE0D22_ OUT	UE0D21_ OUT	UE0D20_ OUT
		Write:	UE0D27_IN	UE0D26_IN	UE0D25_IN	UE0D24_IN	UE0D23_IN	UE0D22_IN	UE0D21_IN	UE0D20_IN
		Reset:	Unaffected by reset							
$0043	USB Endpoint 0 Data Register 3 (UE0D3)	Read:	UE0D37_ OUT	UE0D36_ OUT	UE0D35_ OUT	UE0D34_ OUT	UE0D33_ OUT	UE0D32_ OUT	UE0D31_ OUT	UE0D30_ OUT
		Write:	UE0D37_IN	UE0D36_IN	UE0D35_IN	UE0D34_IN	UE0D33_IN	UE0D32_IN	UE0D31_IN	UE0D30_IN
		Reset:	Unaffected by reset							
$0043	USB Endpoint 0 Data Register 4 (UE0D4)	Read:	UE0D47_ OUT	UE0D46_ OUT	UE0D45_ OUT	UE0D44_ OUT	UE0D43_ OUT	UE0D42_ OUT	UE0D41_ OUT	UE0D40_ OUT
		Write:	UE0D47_IN	UE0D46_IN	UE0D45_IN	UE0D44_IN	UE0D43_IN	UE0D42_IN	UE0D41_IN	UE0D40_IN
		Reset:	Unaffected by reset							
$0044	USB Endpoint 0 Data Register 5 (UE0D5)	Read:	UE0D57_ OUT	UE0D56_ OUT	UE0D55_ OUT	UE0D54_ OUT	UE0D53_ OUT	UE0D52_ OUT	UE0D51_ OUT	UE0D50_ OUT
		Write:	UE0D57_IN	UE0D56_IN	UE0D55_IN	UE0D54_IN	UE0D53_IN	UE0D52_IN	UE0D51_IN	UE0D50_IN
		Reset:	Unaffected by reset							
$0045	USB Endpoint 0 Data Register 6 (UE0D6)	Read:	UE0D67_ OUT	UE0D66_ OUT	UE0D65_ OUT	UE0D64_ OUT	UE0D63_ OUT	UE0D62_ OUT	UE0D61_ OUT	UE0D60_ OUT
		Write:	UE0D67_IN	UE0D66_IN	UE0D65_IN	UE0D64_IN	UE0D63_IN	UE0D62_IN	UE0D61_IN	UE0D60_IN
		Reset:	Unaffected by reset							
$0046	USB Endpoint 0 Data Register 7 (UE0D7)	Read:	UE0D77_ OUT	UE0D76_ OUT	UE0D75_ OUT	UE0D74_ OUT	UE0D73_ OUT	UE0D72_ OUT	UE0D71_ OUT	UE0D70_ OUT
		Write:	UE0D77_IN	UE0D76_IN	UE0D75_IN	UE0D74_IN	UE0D73_IN	UE0D72_IN	UE0D71_IN	UE0D70_IN
		Reset:	Unaffected by reset							

U = Unaffected by reset R = Reserved

Figure 11-2. USB Registers (Continued)

11.5.1 USB Control Register (USBCR)

Address: $0051

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	USBEN	USBCLK-EN	TC4IE	TC3IE	TC2IE	TFC1IE	TC0IE	0
Write:								RESUME
Reset:	0	0	0	0	0	0	0	0

Figure 11-3. USB Control Register

USBEN — USB Module Enable

This read/write bit enables the USB module. Setting this bit updates the endpoint configuration according to the definition defined in UEPxCSR, UINTFCR, UEP12BPR and UEP34BPR registers. User must ensure the 48MHz clock source is ready before the module is enabled. When the USBEN bit is returned to zero, the module is reset and the device is returned to power state. User is recommended to reset all the status flags by software before enabling USBEN again. Reset clears this bit.

1 = USB module enabled
0 = USB module disabled

USBCLKEN — USB Clock Enable

This read/write bit enables the 48MHz clock source to the USB module. User must ensure this bit is set before setting USBEN bit. In USB suspend mode it is recommended to clear this bit for power saving. Reset clears this bit.

1 = USB clock enabled
0 = USB clock disabled

TCxIE — Transfer complete interrupt enable for endpoint x

The read/write bit enables CPU interrupt when transfer complete flag (TFRC) of corresponding endpoint is set. TC0IE is controlling both TFRC0_IN and TFRC0_OUT at the same time. Reset clears this bit.

1 = Transfer complete interrupt enabled
0 = Transfer complete interrupt disabled

RESUME — Force RESUME condition

This write-only bit forces a resume state (K or non-idle state) onto the USB bus data lines to initiate a remote wakeup. This bit generates RESUME only if the device is in SUSPEND mode and the remote wake up feature is enabled by the SET_FEATURE command. The USB control logic ensures the forced resume duration is greater than 3ms. Reading this bit always returns zero. Writing zero to the bit has no effect.

1 = Generates forced RESUME condition on the USB data lines
0 = Default value

11.5.2 USB Status Register (USBSR)

Address: $0052

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	CONFIG		SETUP	SOF	CONFIG_ CHG	USBRST	RESUMEF	SUSPND
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-4. USB Status Register

CONFIG — Configuration Number

This read-only bit specify the configuration number returned from the USB host. The module only supports a single configuration setting.

1 = Device configure to configuration number 1
0 = Device is unconfigured

SETUP — SETUP Request Received

This read/write bit indicates a valid GET_DESCRIPTOR command, SYNC_FRAME command or class/vendor specific request is detected. This bit only set when the packet is received without CRC/Token/EOP error. When this is set, user must read and decode the request from the endpoint 0 data registers (UE0D7-0). Writing zero to clear the bit. Writing one to the bit has no effect. Reset clears this bit.

1 = GET_DESCRIPTOR, SYNC_FRAME or Class/vendor specific requests received
0 = No vendor specific request received

SOF — Start of Frame Detection Flag

This read/write bit indicates a start-of-frame signal is detected from the USB data line. Writing zero to clear the bit. Writing one to the bit has no effect. Reset clears this bit.

1 = Start-of-frame is detected
0 = No start-of-frame is detected

CONFIG_CHG — Change of Configuration Detection Flag

This read/write bit indicates a change of device configuration request is received from the host. This bit will be set when new configuration is requested and accepted by the host. Writing zero to clear the bit. Writing one to the bit has no effect. Reset clears this bit.

1 = A change of configuration request is received
0 = No change of configuration request is received

USBRST — USB Reset Detection Flag

This read/write bit is set when a valid reset signal state is detected on the D+ and D- lines. If URSTD bit of the configuration register (CONFIG) is clear, this reset detection flag will generate a MCU internal reset signal to reset the CPU and its peripheral. Otherwise, if URSTD is set, a interrupt request will be generated instead. Writing zero to clear the bit. Writing one to the bit has no effect. Reset clears this bit.

1 = USB reset signal is detected
0 = No USB reset signal is detected

RESUMEF — RESUME Detection Flag

This read/write bit is set when bus activity is detected while the device is in SUSPEND mode. Writing zero to clear the bit. Writing one to the bit has no effect. Reset clears this bit.

1 = USB bus activity is detected while the device is in SUSPEND mode
0 = No USB bus activity is detected

SUSPND — SUSPEND Detection Flag

This read/write bit is set when the module detects a suspend state on the USB bus or the bus is idle for 3ms. The module will enter suspend mode when this bit is set. In order to reduce the power consumption, user is recommended to stop the USB module clock by clearing the USBCLKEN bit in USBCR register before putting the MCU in STOP mode. Writing zero to clear the bit. Writing one to the bit has no effect. Reset clears this bit.

1 = SUSPEND state is detected
0 = No suspend state is detected

11.5.3 USB Status Interrupt Mask Register (USIMR)

Address: $0053

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	0	SETUPIE	SOFIE	CONFIG_ CHGIE	USBRE- SETIE	RESUME- FIE	SUSP- NDIE
Write:		EP0_STALL						
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 11-5. USB Status Interrupt Mask Register

EP0_STALL — Forced EP0 STALL Handshake Enable

This write only bit is used to provide protocol STALL to the control endpoint. Writing one to the bit causes endpoint 0 to return STALL in response to any IN or OUT token issue by the USB host until the next SETUP transaction. The bit can only be erased by module hardware, writing zero to the bit has no effect. Reset also clears this bit.

1 = Send STALL handshake
0 = Do not response STALL handshaking

SETUPIE — SETUP Request Interrupt Mask

This read/write bit enables a CPU interrupt request when GET_DESCRIPTOR, SYNC_FRAME or class/vendor specific request is received or SETUP flag of USB status register (USBSR) is set. Reset clears this bit.

1 = CPU interrupt is enabled when SETUP flag in USBSR is set
0 = CPU interrupt is disabled when SETUP flag in USBSR is set

SOFIE — Start-of-frame Interrupt Mask

This read/write bit enables a CPU interrupt request when a start-of-frame signal is detected on the USB bus or SOF flag of USB status register (USBSR) is set. Reset clears this bit.

1 = SOF interrupt is enabled
0 = SOF interrupt is disabled

CONFIG_CHGIE — Configuration Change Interrupt Mask

This read/write bit enables a CPU interrupt request when a configuration change from zero to one is detected or CONFIG_CHG flag of USB status register (USBSR) is set. Reset clears this bit.

1 = CPU interrupt is enabled when CONFIG_CHG flag in USBSR is set
0 = CPU interrupt is disabled when CONFIG_CHG flag in USBSR is set

USBRSTIE — USB RESET Interrupt Mask

This read/write bit enables a CPU interrupt request when USBRST flag of USB status register (USBSR) and URSTD bit of configuration register (CONFIG) is set. Reset clears this bit.

1 = CPU interrupt request is enabled when USB reset signal is detected
0 = CPU interrupt request is disabled when USB reset signal is detected

RESUMEFIE — Resume Interrupt Mask

This read/write bit enables a CPU interrupt request when the USB bus activity is resumed or RESUMEF of USB status register (USBSR) is set. Reset clears this bit.

1 = CPU interrupt request is enabled when USB bus activity is resumed
0 = CPU interrupt request is disabled when USB bus activity is resumed

SUSPNDIE — Suspend Mode Interrupt Mask

This read/write bit enables a CPU interrupt request when the module entered suspend mode or SUSPND flag of USB status register (USBSR) is set. Reset clears this bit.

1 = CPU interrupt request is enabled when the module enters suspend mode
0 = CPU interrupt request is disabled when the module enters suspend mode

11.5.4 USB Endpoint 0 Control/Status Register (UEP0CSR)

Address: $0054

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	DSIZE3_OUT	DSIZE2_OUT	DSIZE1_OUT	DSIZE0_OUT	DVALID_IN	TFRC_IN	DVALID_OUT	TFRC_OUT
Write:	DSIZE3_IN	DSIZE2_IN	DSIZE1_IN	DSIZE0_IN				
Reset:	0	0	0	0	0	0	0	0

Figure 11-6. USB Endpoint 0 Control/Status Register

DSIZE[3:0]_OUT — Endpoint 0 Data Size for OUT packet

These bits specify the packet size received for the previous valid OUT packet. The bits are read only.

DSIZE[3:0]_IN — Endpoint 0 Data Size for IN packet

These bits indicates the packet size to be transmitted in response to the next IN packet. The bits are write only.

DVALID_IN — Data valid enable bit for IN packet

This read/write bit indicates the data in the endpoint buffer is valid and ready to send. Setting this bit triggers the data transmission. The bit will be cleared automatically by hardware when a successful IN packet transaction occurred or a valid SETUP packet is received. The bit can also be cleared by writing zero. When the bit is zero, all IN packets to endpoint zero will be responded by NAK. Reset also clears this bit.

1 = Data in the EP0 buffer is valid and ready to transmit
0 = Data in the EP0 buffer is not valid

TFRC_IN — Transfer Complete Flag for IN packet

This read/write bit indicates the data in the EP0 buffer is completely transferred to the host. When the bit is set, all successive IN packet is responded by NAK. Writing zero to clear this bit. Writing one to the bit has no effect.

1 = Endpoint data transfer completed
0 = Default status

DVALID_OUT — Data valid enable bit for OUT packet

This bit indicates valid data is stored in the endpoint buffer, CPU attention is required. User must clear this bit in order to receive the next OUT packet by writing zero to the bit, otherwise all successive OUT packet is NAK by the module. Writing one to the bit has no effect. Reset clears this bit.

1 = Data in the EP0 buffer is valid
0 = Data in the EP0 buffer is not valid

TFRC_OUT — Transfer Complete Flag for OUT packet

This read/write bit indicates the a valid OUT or SETUP packet is completely transferred to the EP0 data buffer. When the bit is set, all successive OUT packet will be responded by NAK. Writing zero to clear this bit. Writing one to the bit has no effect.

1 = Endpoint data transfer completed
0 = Default status

11.5.5 USB Endpoint 1–4 Control Status Register (UEP1CSR–UEP4CSR)

Address: $0055 to $0058

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MODE1	MODE0	0	DIR	SIZE1	SIZE0	DVALID	TFRC
Write:			STALL					
Reset:	0	0	0	0	0	0	0	0

Figure 11-7. USB Endpoint 1–4 Control Status Register

TFRC — Transfer Complete Flag

This read/write bit indicates the data transfer associated with the endpoint is completed. When the bit is set, all successive IN/OUT packet will be responded by NAK. Writing zero to clear this bit. Writing one to the bit has no effect.

1 = Endpoint data transfer completed
0 = Default status

DVALID — Data valid bit

When the endpoint is configured as IN endpoint, this bit indicates the data in the endpoint buffer is valid and ready to be sent. Setting this bit arms the data transmission otherwise all IN packets are returned by NAK. The bit will be cleared automatically by hardware when a successful IN packet transaction occurred.

When the endpoint is configured as OUT endpoint, this bit indicates valid received data is stored in the endpoint buffer, CPU attention is required. User must clear this bit in order to receive the next OUT packet, otherwise all successive OUT packet is responded NAK by the module. Reset clears this bit.

1 = Data in the endpoint buffer is valid
0 = Data in the endpoint buffer is not valid

SIZE[1:0] — Buffer size Selection bits

This read/write bits select the buffer size for the corresponding endpoint. When USBEN is set, these bits has no effect.

Table 11-4. Buffer Size Selection Table

SIZE[1:0]	Buffer Size
00	8 Bytes
01	16 Bytes
10	32 Bytes
11	64 Bytes

TFRCIE — Transfer Complete Interrupt Enable

This read/write bit enables the CPU interrupt associated with the TFRC flag.

1 = TFRC flag interrupt is enabled

0 = TFRC flag interrupt is disabled

DIR — Endpoint Direction Bit

Setting this bit enables the endpoint to become IN endpoint. Clearing this bit enables the endpoint to become OUT endpoint. Writing to this bit will have no effect when USBEN is set.

1 = IN endpoint is enabled

0 = OUT endpoint is enabled

STALL — Forced STALL Handshake Enable

This read/write bit causes endpoint 0 to return a STALL handshake when polled by either an IN or OUT token by the USB host. If the bit is set by software, when a data packet addressed to that endpoint is detected, this STALL status will be latched to the module, the bit is cleared automatically and the packet will be responded by STALL. Once the STALL status is latched, it can only be cleared by CLEAR_FEATURE command from the host. Software cannot clear this status. If there is no packet addressed to the endpoint after the bit is set, it can still be cleared by writing zero. Reset clears this bit.

1 = Send STALL handshake

0 = Do not response STALL handshaking

NOTE

When USB RESET is detected, explicitly writing zero to the STALL bit is recommended to ensure all unlatched STALL status is cleared.

MODE[1:0] — Endpoint Type selection

This bit selects the type of the endpoint. When USBEN is set, this bit has no effect.

Table 11-5. Mode selection for Endpoint type

MODE[1:0]	Endpoint Type
00	Endpoint Disable
01	—
10	Bulk
11	Interrupt

11.5.6 USB Endpoint 1–4 Data Size Register (UEP1DSR–UEP4DSR)

Address: $0059 to $005C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:		DSIZE6	DSIZE5	DSIZE4	DSIZE3	DSIZE2	DSIZE1	DSIZE0
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-8. USB Endpoint 1–4 Data Size Register

DSIZE[6:0] — Packet Data Size

When the corresponding endpoint is configured as IN endpoint, these bits indicates the packet size to be transmitted. When the corresponding endpoint is configured as OUT endpoint, these bits indicates the packet size received.

11.5.7 USB Endpoint 1/2 and 3/4 Base Pointer Register (UEP12BPR–UEP34BPR)

Address: $005D to $005E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: / Write:		BASE22	BASE21	BASE20		BASE12	BASE11	BASE10
Reset:	0	0	0	0	0	0	0	0
Read: / Write:		BASE42	BASE41	BASE40		BASE32	BASE31	BASE30
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-9. USB Endpoint 1-4 Data Pointer Register

BASEx[2:0] — Base Location Pointer

There are total 64 bytes of dedicated RAM space assigned to the module, the addressable space is from address $1000 to $103F. User must allocated appropriate buffer area to the endpoint which matches with the packet size reported to the host. This register indicates the base address pointer for the endpoint buffer area. The pointer location must align to the 8 bytes boundary. BASEx[2:0] specifies only the 3 address bits A5-A3. When USBEN is set, these bit have no effect.

Table 11-6. BASEx[2:0] Address Definition

%0001 0000	0	0	BASEx[2:0]			0	0	0
A15–A8	A7	A6	A5	A4	A3	A2	A1	A0

11.5.8 USB Interface Control Register (UINTFCR)

Address: $005F

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:		EP4INT		EP3INT		EP2INT		EP1INT
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 11-10. USB Interface Control Register

EP1INT — Endpoint 1 Interface number

This bit specifies the interface number for physical endpoint 1. The interface number is loaded to the USB module at the time when the USB module is enabled. When USBEN is set, this bit has no effect. Reset clears this bit.

1 = The interface number for endpoint 1 is one
0 = The interface number for endpoint 1 is zero

EP2INT — Endpoint 2 Interface number

This bit specifies the interface number for physical endpoint 2. The interface number is loaded to the USB module at the time when the USB module is enabled. When USBEN is set, this bit has no effect. Reset clears this bit.

1 = The interface number for endpoint 2 is one
0 = The interface number for endpoint 2 is zero

EP3INT — Endpoint 3 Interface number

This bit specifies the interface number for physical endpoint 3. The interface number is loaded to the USB module at the time when the USB module is enabled. When USBEN is set, this bit has no effect. Reset clears this bit.

1 = The interface number for endpoint 3 is one7
0 = The interface number for endpoint 3 is zero

EP4INT — Endpoint 4 Interface number

This bit specifies the interface number for physical endpoint 4. The interface number is loaded to the USB module at the time when the USB module is enabled. When USBEN is set, this bit has no effect. Reset clears this bit.

1 = The interface number for endpoint 4 is one
0 = The interface number for endpoint 4 is zero

11.5.9 USB Endpoint 0 Data Register 7–0 (UE0D7–UE0D0)

Address: $0040 to $0047

Read:	UE0Dx7_ OUT	UE0Dx6_ OUT	UE0Dx5_ OUT	UE0Dx4_ OUT	UE0Dx3_ OUT	UE0Dx2_ OUT	UE0Dx1_ OUT	UE0D0x_ OUT
Write:	UE0Dx7_ IN	UE0Dx6_ IN	UE0Dx5_ IN	UE0Dx4_ IN	UE0Dx3_ IN	UE0Dx2_ IN	UE0Dx1_ IN	UE0Dx0_ IN
Reset:	Unaffected by reset							

Figure 11-11. USB Endpoint 0 Data Register 7–0

These are the IN/OUT data buffer endpoint 0. The OUT buffer can be accessed by reading to the registers. The IN buffer can be accessed by writing to the registers.

Chapter 12
PS2 Clock Generator (PS2CLK)

12.1 Introduction

This module provides the capability to generate PS2 clock.

12.2 Functional Description

Figure 12-1 shows the block diagram for the PS2 clock generator. The module is enabled by setting PS2EN bit. When the module is enabled, the output port becomes an open-drain output. A two phase clock is created by the prescaler block, one is driving the clock generator unit and the other is driving the interrupt generator. The clock generator drives the output port directly if CLKEN bit is set, while the interrupt generator enables CPU interrupt at the different clock phase. The CPU interrupt can be masked by clearing the PS2IEN bit. The waveform diagram is shown in Figure 12-2.

When the module is enabled, the output port status is continuous monitored and stored in PSTATUS flag.

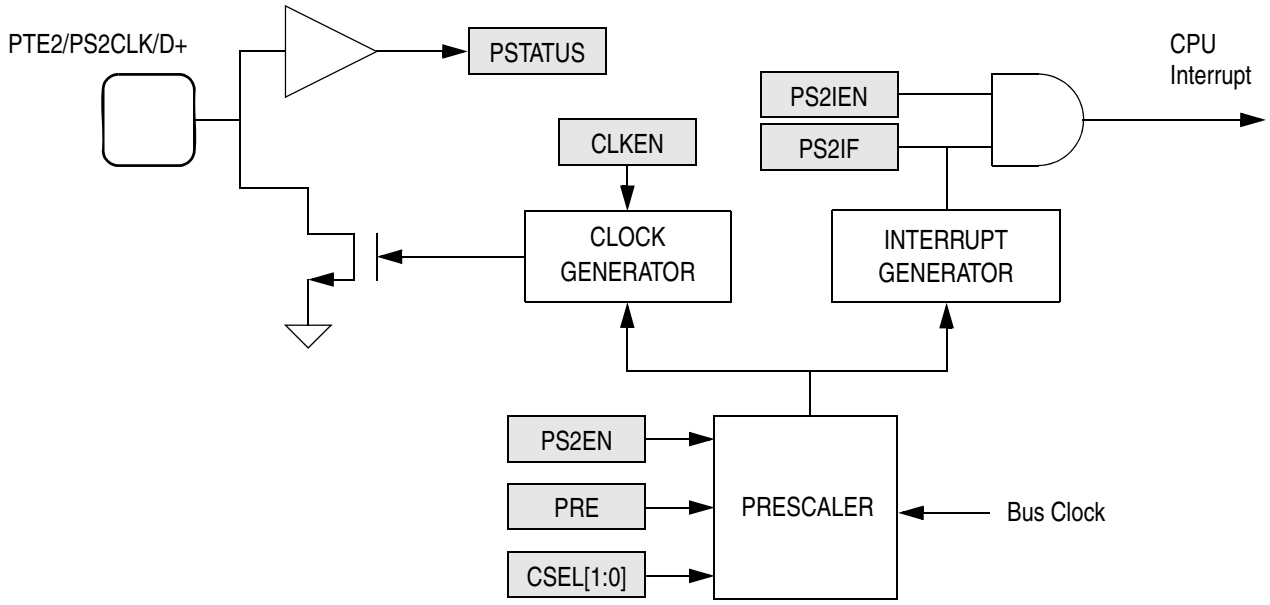


Figure 12-1. PS2 Clock Generator Block Diagram

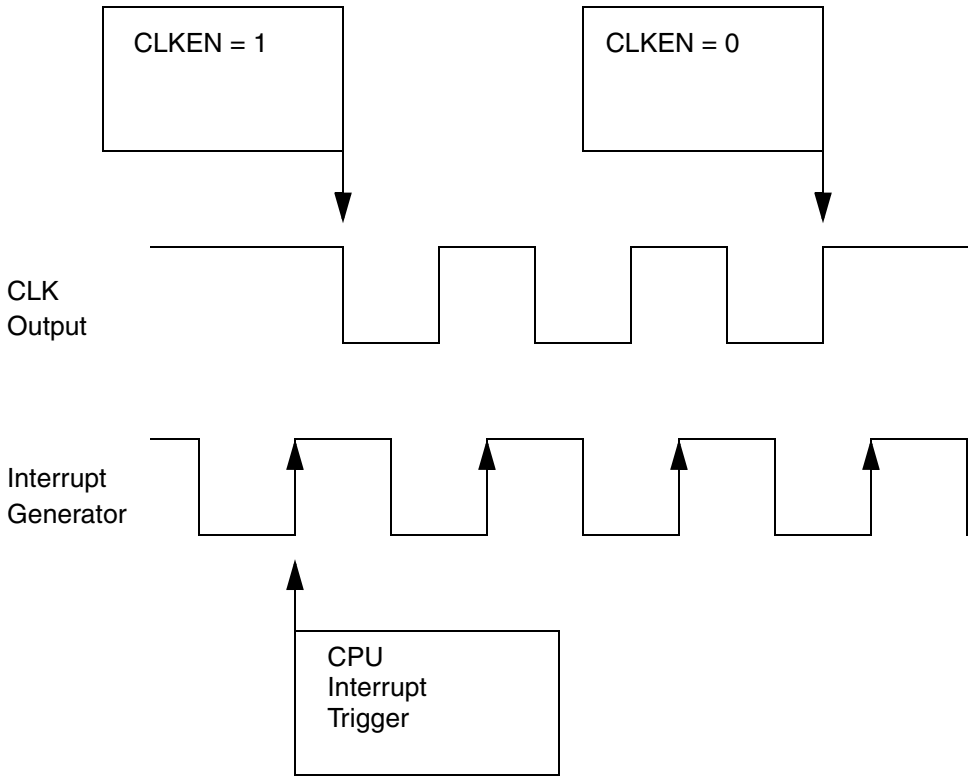


Figure 12-2. Clock Generator Output Waveform.

12.3 PS2 Clock Generator Control and Status Registers

Address: $0019

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	PSTATUS	PS2IF	PRE	CSEL1	CSEL0	PS2IEN	CLKEN	PS2EN
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 12-3. PS2 Clock Generator Control and Status Registers (PS2CSR)

PSTATUS — Port Status Flag

This read only bit reflects the port status when the module is enabled. Reset clears this bit.

1 = Port status is logic high
0 = Port status is logic low

PS2IF — PS2 Interrupt Flag.

This flag is set when PS2 interrupt is trigger by the interrupt generator. Writing one to this bit clears the flag. Reset clears this flag.

1 = CPU interrupt is pending
0 = CPU interrupt is not pending

PRE — Prescaler Selection

These bits select prescaler divider ratio. Reset clears this bit.

1 = Divide by 480 is selected
0 = Divide by 160 is selected

CSEL[1:0] — Clock Frequency Selection bits.

These bits selects the clock divider ratio to cater for different clock source frequency. Reset clears these bits.

Table 12-1. CSEL[1:0] Divider Ratio

CSEL[1:0]	Divider Ratio
00	1
01	2
10	4
11	Not used

Table 12-2. Clock Selection Summary

BUS Frequency	PRE (Divider Raito)	CSEL[1:0] (Divider Ratio)	Port Output Frequency
8-MHz	160	4	12.5 kHz
6-MHz	480	1	12.5 kHz
4-MHz	160	2	12.5 kHz

NOTE

Glitches may occur when CSEL[1:0] and PRE value can be altered while PS2EN is set.

PS2IEN — PS2 Interrupt Mask

This read/write bit enables the periodic PS2 interrupt. Reset clears this bit.

1 = PS2 interrupt is enabled
0 = PS2 interrupt is disabled

CLKEN — Clock Output Enable bit

This read/write bit enables the open drain clock output. Reset clears this bit.

1 = Open drain clock output is enabled
0 = Open drain clock output is disabled

PS2EN — PS2 Clock Generator Module Enable

This read/write bit enables the module clock source. Reset clears this bit.

1 = Module enabled
0 = Module disabled

Chapter 13
Input/Output (I/O) Ports

13.1 Introduction

Twenty-nine (29) bidirectional input-output (I/O) pins form five parallel ports. All I/O pins are programmable as inputs or outputs.

NOTE

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Write:								
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read:			PTB5				PTB1	PTB0
		Write:								
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read:					PTC3	PTC2	PTC1	PTC0
		Write:								
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Write:								
		Reset:	Unaffected by reset							
$0008	Port E Data Register (PTE)	Read:	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2		
		Write:								
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read:			DDRB5				DDRB1	DDRB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-1. I/O Port Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0006	Data Direction Register C (DDRC)	Read:					DDRC3	DDRC2	DDRC1	DDRC0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0009	Data Direction Register E (DDRE)	Read:	DDRE7	DDRE6	DDRE5	DDRE4	DDRE3	DDRE2		
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001A	Port Option Control Register 1 (POCR1)	Read:			LEDB5				LEDB1	LEDB0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001B	Port Option Control Register 2 (POCR2)	Read:	0	0	PTD7PD	PTD3PD	PTD2PD	DPPULLEN	PTE3P	PTE2P
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$003E	Pullup Control Register (PULLCR)	Read:			PULL5EN				PULL1EN	PULL0EN
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-1. I/O Port Register Summary (Continued)

Table 13-1. Port Control Register Bits Summary

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
A	0	DDRA0	KBI	KBIER ($17)	KBIE0	PTA0/$\overline{\text{KBA0}}$
	1	DDRA1			KBIE1	PTA1/$\overline{\text{KBA1}}$
	2	DDRA2			KBIE2	PTA2/$\overline{\text{KBA2}}$
	3	DDRA3			KBIE3	PTA3/$\overline{\text{KBA3}}$
	4	DDRA4			KBIE4	PTA4/$\overline{\text{KBA4}}$
	5	DDRA5			KBIE5	PTA5/$\overline{\text{KBA5}}$
	6	DDRA6			KBIE6	PTA6/$\overline{\text{KBA6}}$
	7	DDRA7			KBIE7	PTA7/$\overline{\text{KBA7}}$

Table 13-1. Port Control Register Bits Summary

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
B	0	DDRB0	LED	POCR1 ($1A)	LEDB0	PTB0
			PULLUP	PULLCR ($3E)	PULL0EN	
	1	DDRB1	LED	POCR1 ($1A)	LEDB1	PTB1
			PULLUP	PULLCR ($3E)	PULL1EN	
	5	DDRB5	LED	POCR1 ($1A)	LEDB5	PTB5
			PULLUP	PULLCR ($3E)	PULL5EN	
C	0	DDRC0	TIM1	T1SC0 ($10)	ELS0B:ELS0A	PTC0/T1CH0
	1	DDRC1		T1SC ($0A)	PS[2:0]	PTC1/TCLK1
	2	DDRC2		T1SC1 ($13)	ELS1B:ELS1A	PTC2/T1CH1
	3	DDRC3	—	—	—	PTC3
D	0	DDRD0	—	—	—	PTD0
	1	DDRD1	—	—	—	PTD1
	2	DDRD2	PULLUP	POCR2 ($1B)	PTD2PD	PTD2
	3	DDRD3			PTD3PD	PTD3
	4	DDRD4	—	—	—	PTD4
	5	DDRD5				PTD5
	6	DDRD6				PTD6
	7	DDRD7	PULLUP	POCR2 ($1B)	PTD7PD	PTD7
E	2	DDRE2	USB	USBCR ($51)	USBEN	PTE2/D+
			PULLUP	POCR2 ($1B)	PTE2P	
			PS2CLK	PS2CSR ($19)	PS2EN	
	3	DDRE3	USB	USBCR ($51)	USBEN	PTE3/D–
			PULLUP	POCR2 ($1B)	PTE3P	
			IRQ	IOCR ($1C)	PTE3IE	
	4	DDRE4	SPI	SPCR ($4C)	SPE	PTE4/SPSCK
	5	DDRE5				PTE5/MOSI
	6	DDRE6				PTE6/MISO
	7	DDRE7				PTE7/$\overline{SS}$

13.2 Port A

Port A is an 8-bit general-purpose bidirectional I/O port with software configurable pullups, and it shares its pins with the keyboard interrupt module (KBI).

13.2.1 Port A Data Register

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

Address:	$0000							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
Reset:	Unaffected by reset							
Alternative Function:	$\overline{\text{KBA7}}$	$\overline{\text{KBA6}}$	$\overline{\text{KBA5}}$	$\overline{\text{KBA4}}$	$\overline{\text{KBA3}}$	$\overline{\text{KBA2}}$	$\overline{\text{KBA1}}$	$\overline{\text{KBA0}}$
Additional Function:	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup	Optional pullup

Figure 13-2. 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.

$\overline{\text{KBA7}}$–$\overline{\text{KBA0}}$ — Keyboard Interrupts

The keyboard interrupt enable bits, $\overline{\text{KBA7}}$–$\overline{\text{KBA0}}$, in the keyboard interrupt enable register (KBIER), enable the port A pins as external interrupt pins and the internal pullup of the corresponding pin. (See Chapter 15 Keyboard Interrupt Module (KBI).)

13.2.2 Data Direction Register A

Data direction register A determines whether each port A pin is an input or an output. Writing a 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

* DDRA7 bit is reset by POR or LVI reset only.

Figure 13-3. 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 13-4 shows the port A I/O logic.

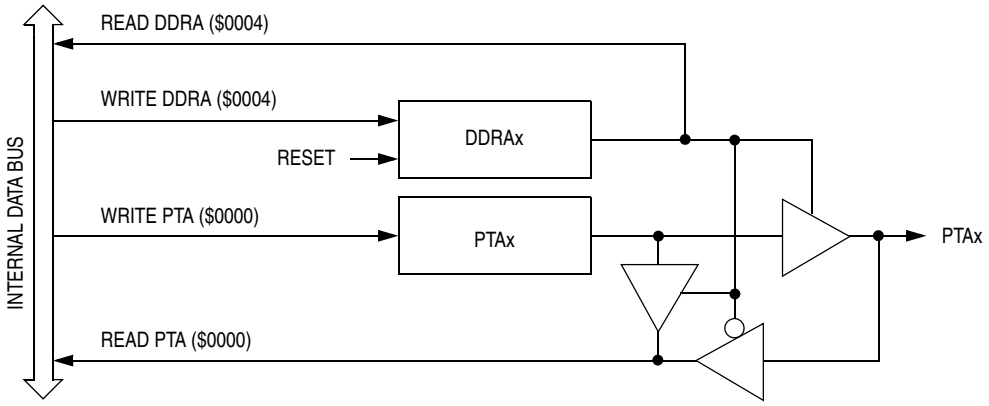


Figure 13-4. Port A I/O Circuit

When bit DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-2 summarizes the operation of the port A pins.

Table 13-2. 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

13.3 Port B

Port B is a 3-bit general-purpose bidirectional I/O port; open-drain when configured as output.

13.3.1 Port B Data Register

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

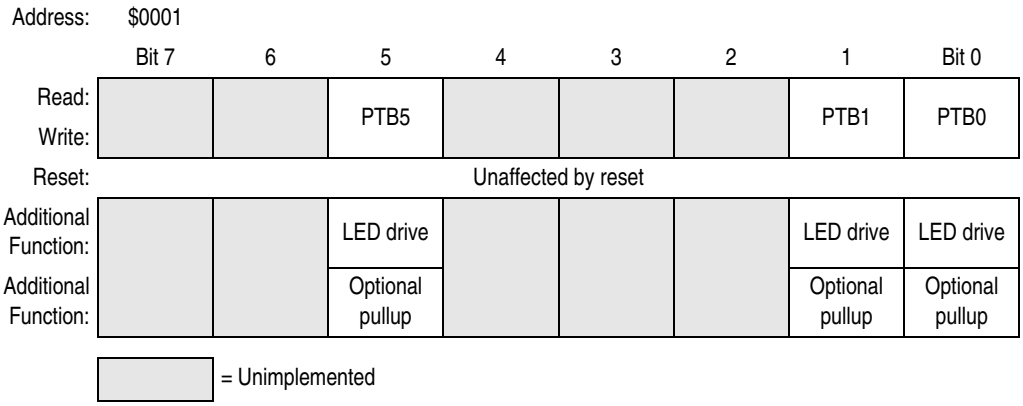


Figure 13-5. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

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

LED Drive — Direct LED Driver

The LED direct drive bit, LEDB[7:0], in the port option control register (POCR) controls the drive options for PTB[7:0]. (See 13.7 Port Options.)

Pullup — Programmable Pullup

The Pullup control bit, PULL[7:0]EN, in the pullup control register (PULLCR) controls the optional pullup for PTB[7:0]. (See 13.7 Port Options.)

13.3.2 Data Direction Register B

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

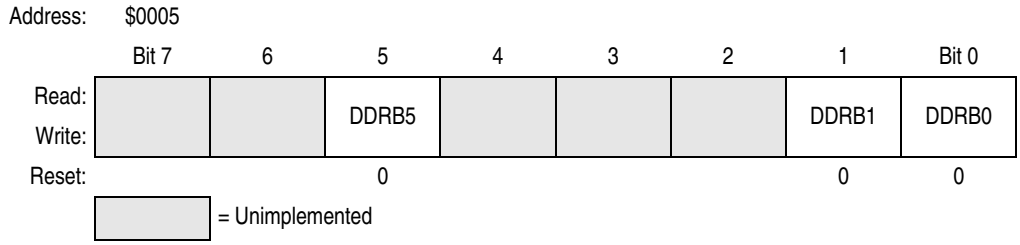


Figure 13-6. Data Direction Register D (DDRD)

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

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

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

NOTE

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

Figure 13-7 shows the port B I/O circuit logic.

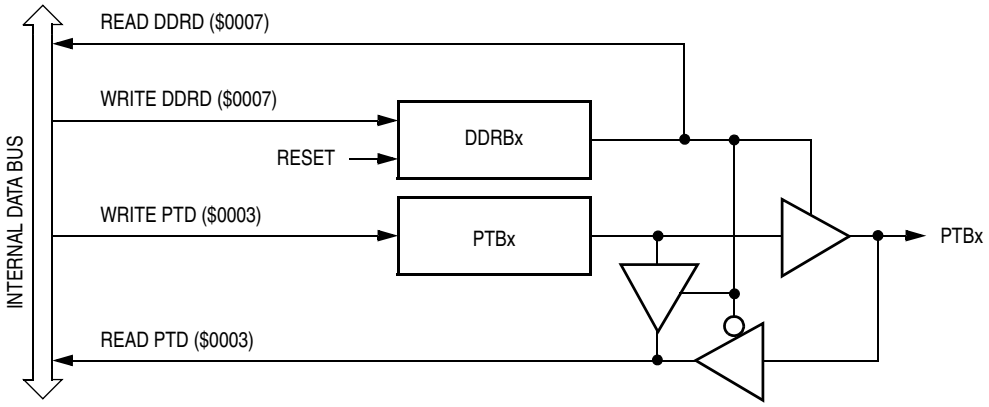


Figure 13-7. Port B I/O Circuit

When bit DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-3 summarizes the operation of the port B pins.

Table 13-3. 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)	DDRB5,1,0	Pin	PTB5,1,0(3)
1	X	Output	DDRB5,1,0	PTB5,1,0	PTB5,1,0

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

13.4 Port C

Port C is a 4-bit general-purpose bidirectional I/O port. PTC[3:0] are shared with Timer.

13.4.1 Port C Data Register

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

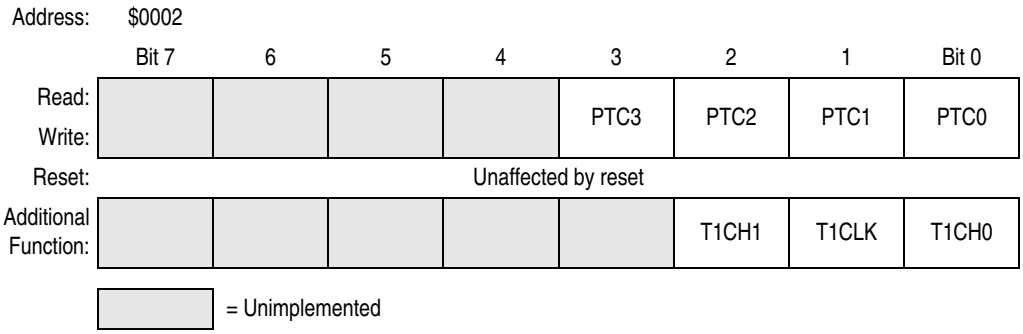


Figure 13-8. Port C Data Register (PTC)

Table 13-4 shows the port function priority table.

Table 13-4. Port C Priority Table

MSxB:MSxA	Feature
01/10/11	Timer function pins
00	Port logic control

PTC[3:0] — 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.

T1CH0, T1CH1 — Timer Channels I/O Bits

The PTC0/T1CH0, PTC2/T1CH1 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB and ELSxA, determine whether they are timer channel I/O pins or general-purpose I/O pins. (see Chapter 8 Timer Interface Module (TIM))

TCLK1 — Timer Clock Input

The PTC1/TCLK1 pin are the external clock input for the TIM. The prescaler select bits, PS[2:0], select PTC1/TCLK1 as the TIM clock input. When not selected as the TIM clock, they are available for general purpose I/O. (see Chapter 8 Timer Interface Module (TIM))

NOTE

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

13.4.2 Data Direction Register C

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

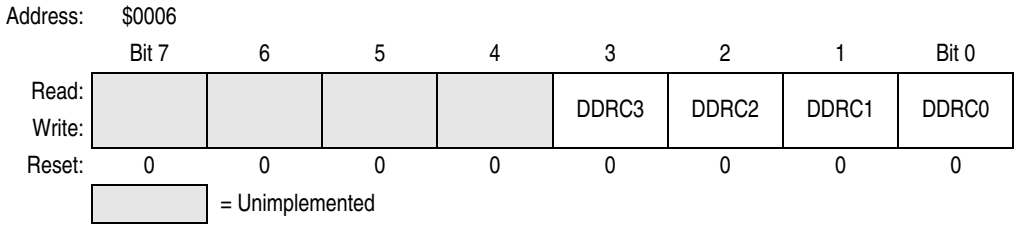


Figure 13-9. Data Direction Register C (DDRC)

DDRC[3:0] — Data Direction Register C Bits

These read/write bits control port C data direction. Reset clears DDRC[3:0], 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 3.

Figure 13-10 shows the port C I/O logic.

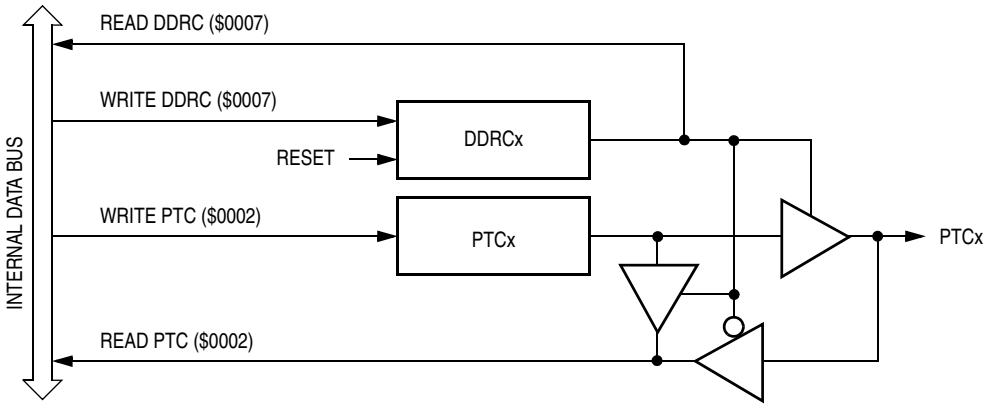


Figure 13-10. Port C I/O Circuit

When bit DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-5 summarizes the operation of the port C pins.

Table 13-5. 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[3:0]	Pin	PTC[3:0](3)
1	X	Output	DDRC[3:0]	PTC[3:0]	PTC[3:0]

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

13.5 Port D

Port D is a 8-bit general-purpose bidirectional I/O port.

13.5.1 Port D Data Register

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

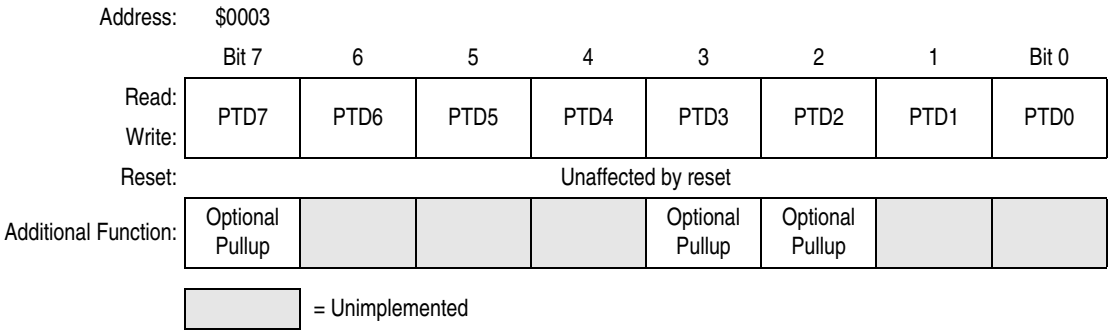


Figure 13-11. Port D Data Register (PTD)

PTD[7:0] — Port D Data Bits

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

PTD2, PTD3 and PTD7

There is programmable pullup associated with the pins. The pullup are default enabled and can be controlled via POCR2 register.

13.5.2 Data Direction Register D

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

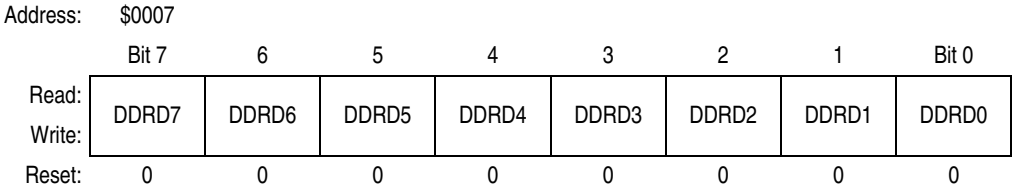


Figure 13-12. Data Direction Register D (DDRD)

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 13-13 shows the port D I/O circuit logic.

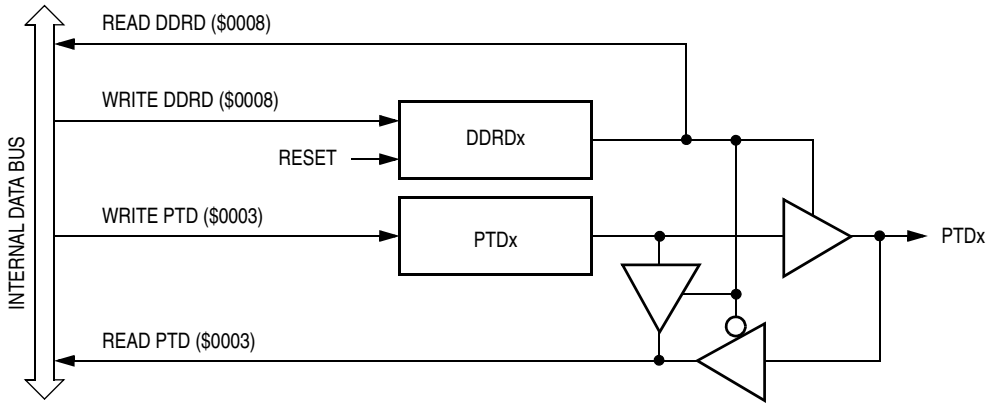


Figure 13-13. Port D I/O Circuit

When bit DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-6 summarizes the operation of the port D pins.

Table 13-6. 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
2. Hi-Z = high impedance
3. Writing affects data register, but does not affect input.

13.6 Port E

Port E is a 6-bit general-purpose bidirectional I/O port. PTE[3:2] are special-function pins that share with the USB data pin D+ and D–. Four of the pins PTE[7:4] are shared with serial peripheral interface (SPI) module.

13.6.1 Port E Data Register

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

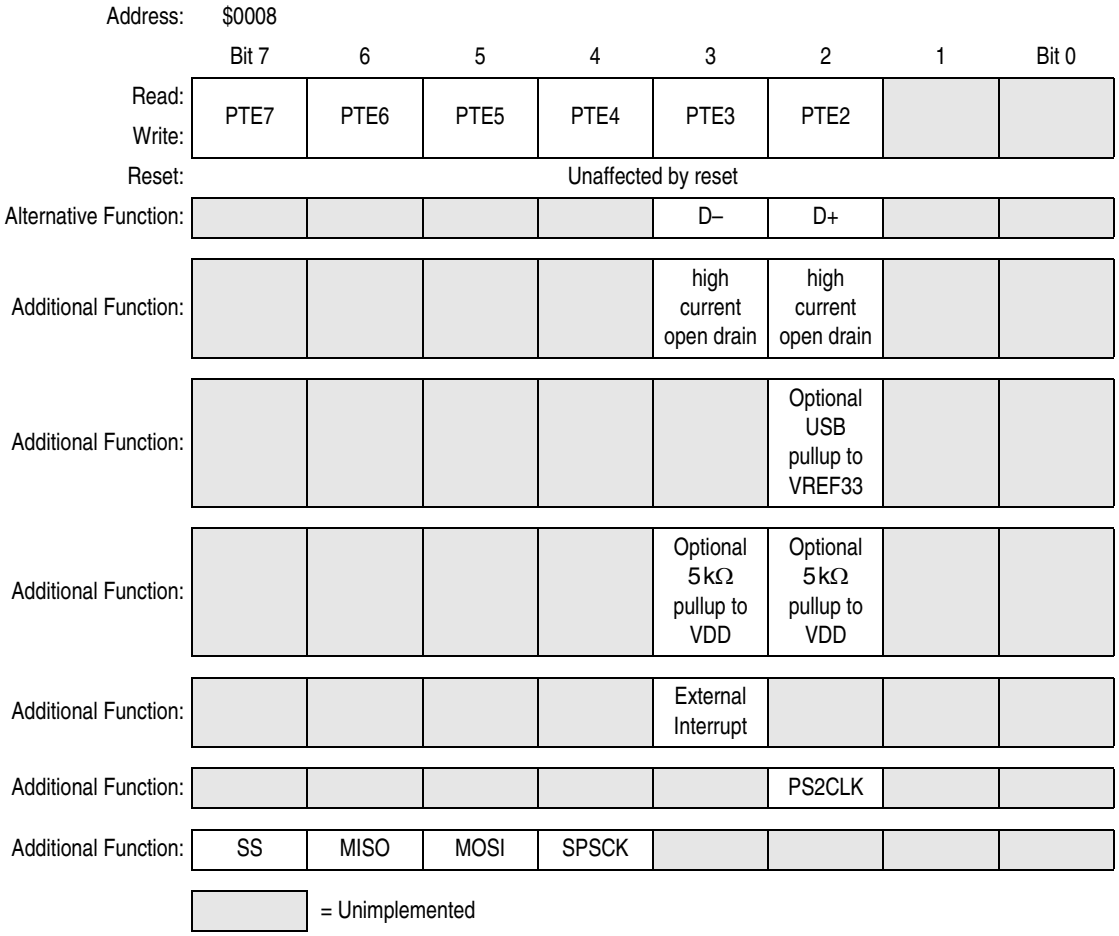


Figure 13-14. Port E Data Register (PTE)

Table 13-7 shows the priority table for PTE2/D+ pin.

Table 13-7. PTE2/D+ Priority Table

USB Module Enable (USBEN)	PS2 Clock Generator Enable (PS2EN)	Data Direction Control (DDRE2)	5k Pullup Enable (PTE2P)	USB D+ Pullup Enable (DPPULPEN)	Pin Function
1	X	X	X	1	D+ with pullup to VREF33.
1	X	X	X	0	D+ without pullup
0	1	X	1/0	0	PS2 Clock output (open-drain) with optional 5k pullup to VDD
0	1	X	0	1/0	PS2 Clock output (open-drain) with optional 1.2k pullup to VREG33
0	0	1	X	1	GPIO output (open-drain) with 1.2K pullup to REG33V
0	0	1	1	0	GPIO output (open-drain) with 5k pullup to VDD
0	0	0	X	1	GPIO input with 1.2K pullup to REG33V
0	0	0	1	0	GPIO input with 5K pullup to VDD

Table 13-8 shows the priority table for PTE3/D– pin.

Table 13-8. PTE3/D– Priority Table

USB Module Enable (USBEN)	PTE3 IRQ Enable (PT3IE)	Data Direction Control (DDRE3)	5K Pullup Enable (PTE3P)	Pin Function
1	X	X	X	USB D– pin
0	1	X	0/1	GPIO input with associated interrupt function and optional 5k pullup to VDD
0	0	0	0/1	GPIO input with optional 5k pullup to VDD
0	0	1	0/1	GPIO output (open-drain) with optional 5k pullup to VDD

PTE[7:2] — Port E Data Bits

PTE[7:2] are read/write, software-programmable bits. Data direction of each port E pin is under the control of the corresponding bit in data direction register E.

The PTE3 and PTE2 pullup enable bits, PTE3P and PTE2P, in the port option control register 2 (POCR2) enable 5kΩ pullups to VDD on PTE3 and PTE2 if the USB module is disabled. (See 13.7 Port Options.)

The PTE2 USB pullup enable bits, DPPULLEN, in the port option control register 2 (POCR2) enable USB pullups to VREF33 on PTE2 for USB operation. Either of PTE2P or DPPULLEN bit can be activated at the one time, DPPULLEN bit has higher priority, it will always override the setting of PTE2P bit.

PTE3 pin functions as an external interrupt when PTE3IE=1 in the IRQ option control register (IOCR) and USBEN=0 in the USB address register (USB disabled). (See 14.7 IRQ Status and Control Register.)

PTE2 pin also muxed with PS2 clock generator module. (See Chapter 12 PS2 Clock Generator (PS2CLK).)

D– and D+ — USB Data Pins

D– and D+ are the differential data lines used by the USB module. (See Chapter 11 USB 2.0 FS Module.)

When the USB module is enabled, PTE2/D+ and PTE3/D– function as USB data pins D– and D+. When the USB module is disabled, PTE2/D+ and PTE3/D– function as open drain high current pins for PS/2 clock and data use.

NOTE

PTE2/D+ pin has two programmable pullup resistors. One is used for PTE2 when the USB module is disable and another is used for D+ when the USB module is enabled.

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

$\overline{SS}$, MISO, MOSI, and SPSCK — SPI Functional Pins

These are the chip select, master-input-slave-output, master-output-slave-input and clock pins for the SPI module. The SPI enable bit, SPE, in the SPI control register, SPCR, enables these pins as the SPI functional pins and overrides any control from port I/O logic. See Chapter 10 Serial Peripheral Interface Module (SPI).

13.6.2 Data Direction Register E

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

Address: $0009

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

= Unimplemented

Figure 13-15. Data Direction Register E (DDRE)

DDRE[7:2] — Data Direction Register E Bits

These read/write bits control port E data direction. Reset clears DDRE[7:2], configuring all port E pins as inputs.

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

NOTE

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

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

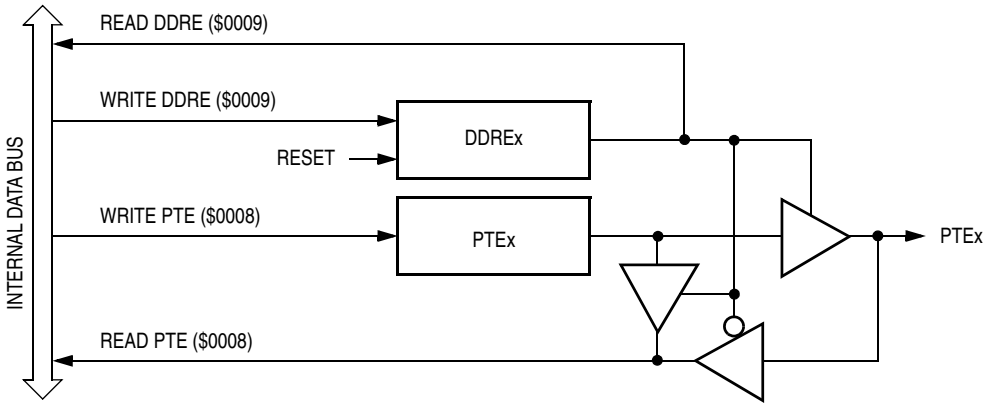


Figure 13-16. Port E I/O Circuit

When bit DDREx is a logic 1, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic 0, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 13-9 summarizes the operation of the port E pins.

Table 13-9. 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[7:2]	Pin	PTE[7:2](3)
1	X	Output	DDRE[7:2]	PTE[7:2]	PTE[7:2]

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

13.7 Port Options

All pins of port B have programmable pullup resistors and LED drive capability. Port pins have programmable high current drive capability.

13.7.1 Port Option Control Register 1

Address: $001A

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:			LEDB5				LEDB1	LEDB0
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Table 13-10. port Option Control Register 1 (POCR1)

LEDB[5,1,0] — Port B LED 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 configured for direct LED drive: high current sinking capability
0 = Corresponding port B pin configured for standard drive

13.7.2 Port Option Control Register 2

The port option control register controls the pullup options for port E.

Address: $001B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	PTD7PD	PTD3PD	PTD2PD	DPPULLEN	PTE3P	PTE2P
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Table 13-11. Port Option Control Register 2 (POCR2)

PTD7PD — Pin PTD7 Pullup Disable

This read/write bit disables the pullup option for pin PTD7. The pullup resistor is default enabled after reset.

1 = Pullup option disabled
0 = Pullup option enabled

PTD3PD — Pin PTD3 Pullup Disable

This read/write bit disables the pullup option for pin PTD3. The pullup resistor is default enabled after reset.

1 = Pullup option disabled
0 = Pullup option enabled

PTD2PD — Pin PTD2 Pullup Disable

This read/write bit disables the pullup option for pin PTD2. The pullup resistor is default enabled after reset.

1 = Pullup option disabled
0 = Pullup option enabled

DPPULLEN — D+ Pullup Enable

This read/write bit enables the USB D+ pullup option to VREF33 for pin PTE2.

1 = Configure PTE2 to have internal USB pullups to VREF33
0 = Disconnect PTE2 internal pullups

PTE3P — Pin PTE3 Pullup Enable

This read/write bit enables the pullup option for pin PTE3 if it is not configured as an USB D– pin or USB module is disabled.

1 = Configure PTE3 to have 5k internal pullups
0 = Disconnect PTE3 internal pullups

PTE2P — Pin PTE2 Pullup Enable

This read/write bit enables the pullup option for pin PTE2 if it is not configured as an USB D+ pin or USB module is disabled.

1 = Configure PTE2 to have 5k internal pullups
0 = Disconnect PTE2 internal pullups

NOTE

When the USB module is enabled, the pullup controlled by PTE2P and PTE3P are disconnected; PTE2/D+ pin functions as D+ which has a programmable pull-up resistor by setting the DPPULLEN bit.

13.7.3 Pullup Control Register (PULLCR)

The pullup control register enables the embedded pullup resistor associated with port B [5,1,0].

Address: $003E

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:			PULL5EN				PULL1EN	PULL0EN
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 13-17. Pullup Control Register (PULLCR)

PULL[5,1,0]EN — Pullup Enable Bit

This read/write bits enables the embedded pullup resistor associated with the corresponding port pin. Reset clears this bit.

1 = Pullup resistor is enabled
0 = Pullup resistor is disabled

Chapter 14
External Interrupt (IRQ)

14.1 Introduction

The IRQ module provides two external interrupt inputs: one dedicated $\overline{\text{IRQ}}$ pin and one shared port pin, PTE3/D–.

14.2 Features

Features of the IRQ module include:

- Two external interrupt pins, $\overline{\text{IRQ}}$ and PTE3/D–
- $\overline{\text{IRQ}}$ interrupt control bits
- Hysteresis buffer
- Programmable edge-only or edge and level interrupt sensitivity
- Automatic interrupt acknowledge
- Low leakage $\overline{\text{IRQ}}$ pin for external RC wake up input
- Selectable internal pullup resistor

14.3 Functional Description

A logic 0 applied to the external interrupt pin can latch a CPU interrupt request. Figure 14-1 shows the structure of the IRQ module.

Interrupt signals on the $\overline{\text{IRQ}}$ 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 (ISCR). Writing a logic 1 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 low-level-triggered. The MODE bit in the ISCR 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 1. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

When set, the IMASK bit in the ISCR 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 6.5 Exception Control.)

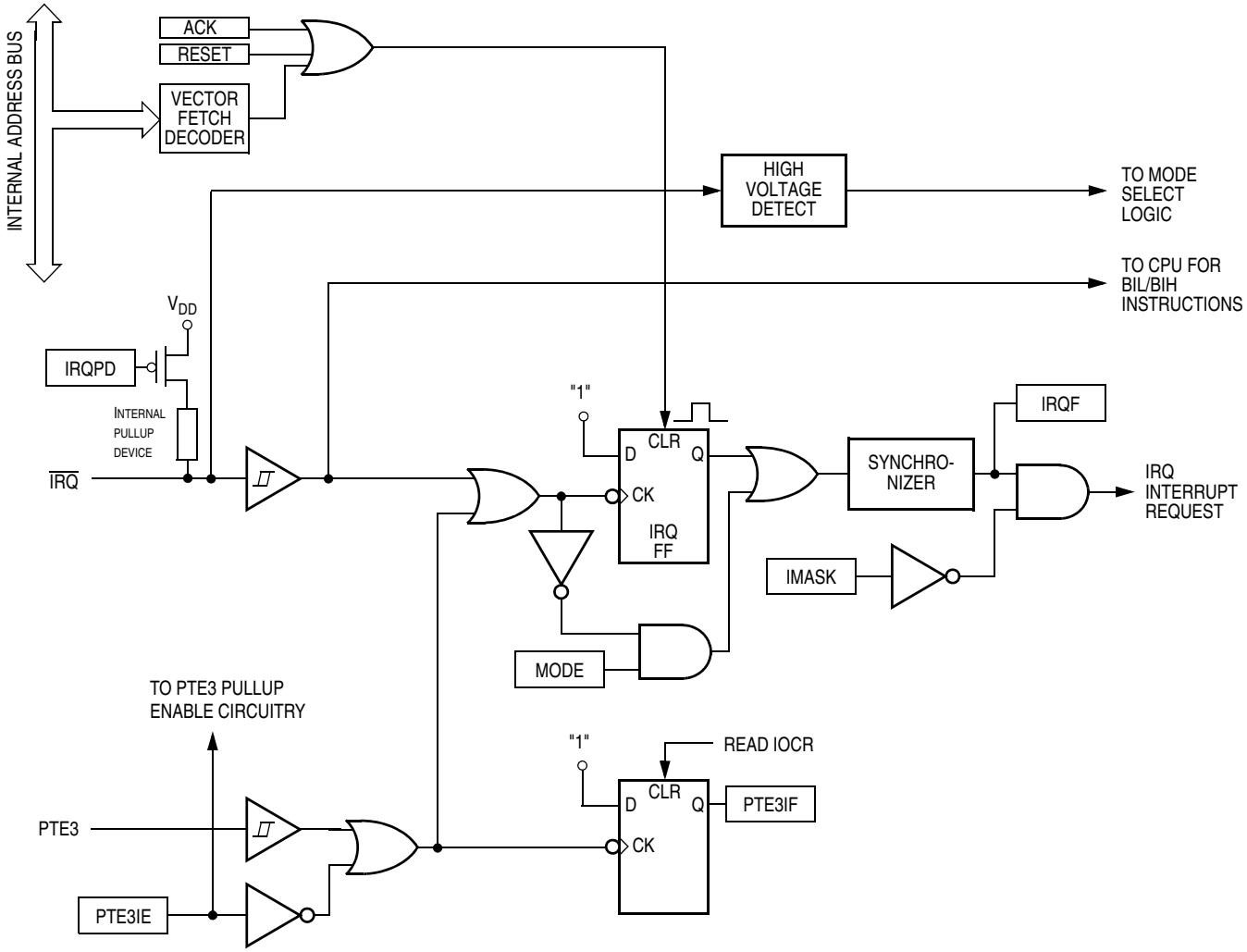


Figure 14-1. IRQ Module Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001C	IRQ Option Control Register (IOCR)	Read:	0	0	0	0	0	PTE3IF	PTE3IE	IRQPD
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001E	IRQ Status and Control Register (ISCR)	Read:	0	0	0	0	IRQF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-2. IRQ I/O Register Summary

14.4 $\overline{\text{IRQ}}$ Pin

The $\overline{\text{IRQ}}$ pin has a low leakage for input voltages ranging from 0V to V_{DD}; suitable for applications using RC discharge circuitry to wake up the MCU.

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

If the MODE bit is set, the $\overline{\text{IRQ}}$ pin is both falling-edge-sensitive and low-level-sensitive. With MODE set, both of 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 1 to the ACK bit in the interrupt status and control register (ISCR). The ACK bit is useful in applications that poll the $\overline{\text{IRQ}}$ pin and require software to clear the IRQ latch. Writing to the ACK bit 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 $FFF8 and $FFF9.
- 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 ISCR register can be used to check for pending interrupts. The IRQF bit is not affected by the IMASK bit, which makes it useful in applications where polling is preferred.

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

NOTE

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

NOTE

An internal pullup resistor to V_{DD} is connected to $\overline{\text{IRQ}}$ pin; this can be disabled by setting the IRQPD bit in the IRQ option control register ($001C).

14.5 PTE3/D– Pin

The PTE3 pin is configured as an interrupt input to trigger the IRQ interrupt when the following conditions are satisfied:

- The USB module is disabled
- PTE3 pin configured for external interrupt input (PTE3IE = 1)

Setting PTE3IE configures the PTE3 pin to an input pin with an internal pullup device. The PTE3 interrupt is "ORed" with the $\overline{IRQ}$ input to trigger the IRQ interrupt Figure 14-1. Therefore, the IRQ status and control register affects both the $\overline{IRQ}$ pin and the PTE pin. An interrupt on PTE3 also sets the PTE3 interrupt flag, PTE3IF, in the IRQ option control register (IOCR).

14.6 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 6 System Integration Module (SIM).)

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

14.7 IRQ Status and Control Register

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

- Shows the state of the IRQ flag
- Clears the IRQ latch
- Masks IRQ interrupt request
- Controls triggering sensitivity of the $\overline{IRQ}$ 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 14-3. IRQ Status and Control Register (ISCR)

IRQF — IRQ Flag

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

1 = IRQ interrupt pending
0 = IRQ interrupt not pending

ACK — IRQ Interrupt Request Acknowledge Bit

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

IMASK — IRQ Interrupt Mask Bit

Writing a logic 1 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 $\overline{IRQ}$ pin. Reset clears MODE.

1 = $\overline{IRQ}$ interrupt requests on falling edges and low levels
0 = $\overline{IRQ}$ interrupt requests on falling edges only

14.8 IRQ Option Control Register

The IRQ option control register controls and monitors the external interrupt function available on the PTE3 pin. It also disables/enables the pullup resistor on the $\overline{IRQ}$ pin.

- Controls pullup option on $\overline{IRQ}$ pin
- Enables PTE3 pin for external interrupts to IRQ
- Shows the state of the PTE3 interrupt flag

Address: $001C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	PTE3IF	PTE3IE	IRQPD
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-4. IRQ Option Control Register (IOCR)

PTE3IF — PTE3 Interrupt Flag

This read-only status bit is high when a falling edge on PTE3 pin is detected. PTE3IF bit clears when the IOCR is read.

1 = falling edge on PTE3 is detected and PTE3IE is set
0 = falling edge on PTE3 is not detected or PTE3IE is clear

PTE3IE — PTE3 Interrupt Enable

This read/write bit enables or disables the interrupt function on the PTE3 pin to trigger the IRQ interrupt. Setting the PTE3IE bit and clearing the USBEN bit in the USB address register configure the PTE3 pin for interrupt function to the IRQ interrupt. Setting PTE3IE also enables the internal pullup on PTE3 pin.

1 = PTE3 interrupt enabled; triggers IRQ interrupt
0 = PTE3 interrupt disabled

IRQPD — $\overline{IRQ}$ Pullup Disable

This read/write bit controls the pullup option for the $\overline{IRQ}$ pin.

1 = Internal pullup is disconnected
0 = Internal pull-up is connected between $\overline{IRQ}$ pin and V_{DD}

Chapter 15
Keyboard Interrupt Module (KBI)

15.1 Introduction

The keyboard interrupt module (KBI) provides eight independently maskable external interrupts which are accessible via PTA0–PTA7 pins.

15.2 Features

Features of the keyboard interrupt module include:

- Eight keyboard interrupt pins with separate keyboard interrupt enable bits and one keyboard interrupt mask
- Hysteresis buffers
- Programmable edge-only or edge- and level-interrupt sensitivity
- Exit from low-power modes

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0016	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
$0017	Keyboard Interrupt Enable Register (KBIER)	Read:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 15-1. KBI I/O Register Summary

15.3 Pin Name Conventions

The eight keyboard interrupt pins are shared with standard port I/O pins. The full name of the KBI pins are listed in Table 15-1. The generic pin name appear in the text that follows.

Table 15-1. Pin Name Conventions

KBI Generic Pin Name	Full MCU Pin Name	Pin Selected for KBI Function by KBIEx Bit in KBIER
$\overline{KBA0}$–$\overline{KBA7}$	PTA0/$\overline{KBA0}$–PTA7/$\overline{KBA7}$	KBIE0–KBIE7

15.4 Functional Description

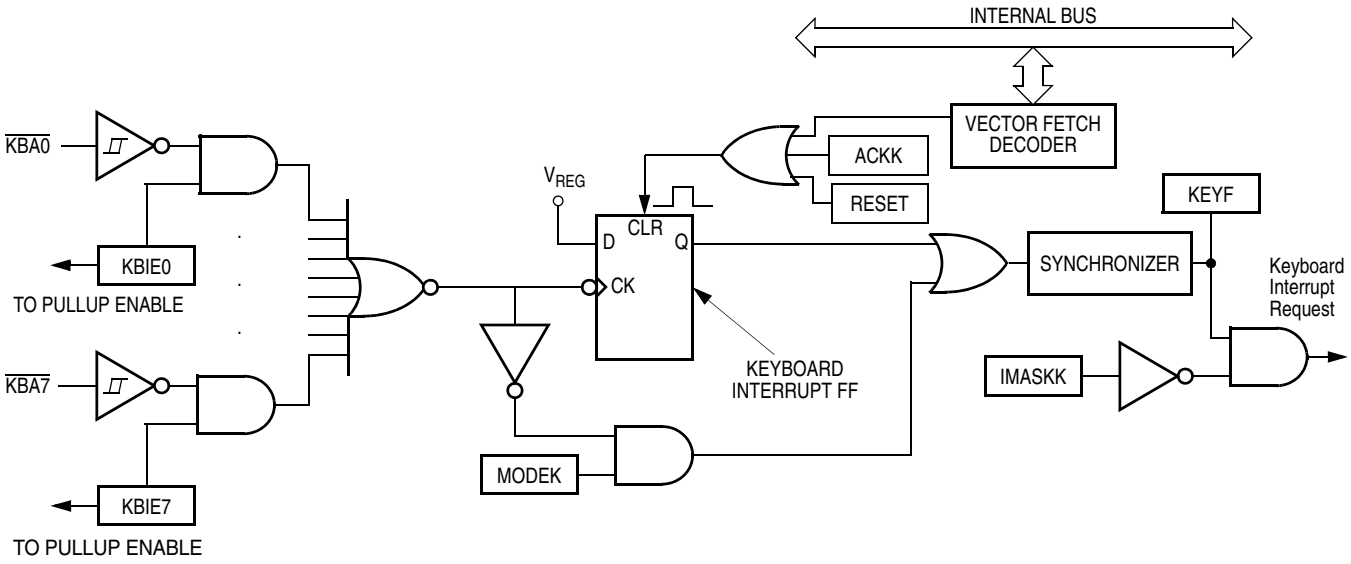


Figure 15-2. Keyboard Module Block Diagram

Writing to the KBIE7–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 also enables its internal pullup 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.
- If the keyboard interrupt is falling edge- and low level-sensitive, an interrupt request is present as long as any keyboard pin is low.

NOTE

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 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 also can 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, use the data direction register to configure the pin as an input and 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.

15.4.1 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the pullup device 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 data direction register A.
2. Write logic 1s 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.

15.5 I/O Registers

These registers control and monitor operation of the keyboard module:

- Keyboard status and control register (KBSCR)
- Keyboard interrupt enable register (KBIER)

15.5.1 Keyboard Status and Control Register

The keyboard status and control register:

- Flags keyboard interrupt requests
- Acknowledges keyboard interrupt requests
- Masks keyboard interrupt requests
- Controls keyboard interrupt triggering sensitivity

Address: $0016

	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 15-3. Keyboard Status and Control Register (KBSCR)

KEYF — Keyboard Flag Bit

This read-only bit is set when a keyboard interrupt is pending. Reset clears the KEYF bit.

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

ACKK — Keyboard Acknowledge Bit

Writing a logic 1 to this write-only bit clears the keyboard interrupt request. ACKK always reads as logic 0. Reset clears ACKK.

IMASKK — Keyboard Interrupt Mask Bit

Writing a logic 1 to this read/write bit prevents the output of the keyboard interrupt mask from generating interrupt requests. Reset clears the IMASKK bit.

1 = Keyboard interrupt requests masked
0 = Keyboard interrupt requests not masked

MODEK — Keyboard Triggering Sensitivity Bit

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

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

15.5.2 Keyboard Interrupt Enable Register

The keyboard interrupt enable register enables or disables each port A pin to operate as a keyboard interrupt pin.

Address: $0017

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
Reset:	0	0	0	0	0	0	0	0

Figure 15-4. Keyboard Interrupt Enable Register (KBIER)

KBIE7–KBIE0 — Keyboard Interrupt Enable Bits

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch interrupt requests. Reset clears the keyboard interrupt enable register.

1 = PTAx pin enabled as keyboard interrupt pin
0 = PTAx pin not enabled as keyboard interrupt pin

15.6 Low-Power Modes

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

15.6.1 Wait Mode

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

15.6.2 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.

15.7 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. (See Figure 15-3. Keyboard Status and Control Register (KBSCR).)

Chapter 16
Computer Operating Properly (COP)

16.1 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 CONFIG register.

16.2 Functional Description

Figure 16-1 shows the structure of the COP module.

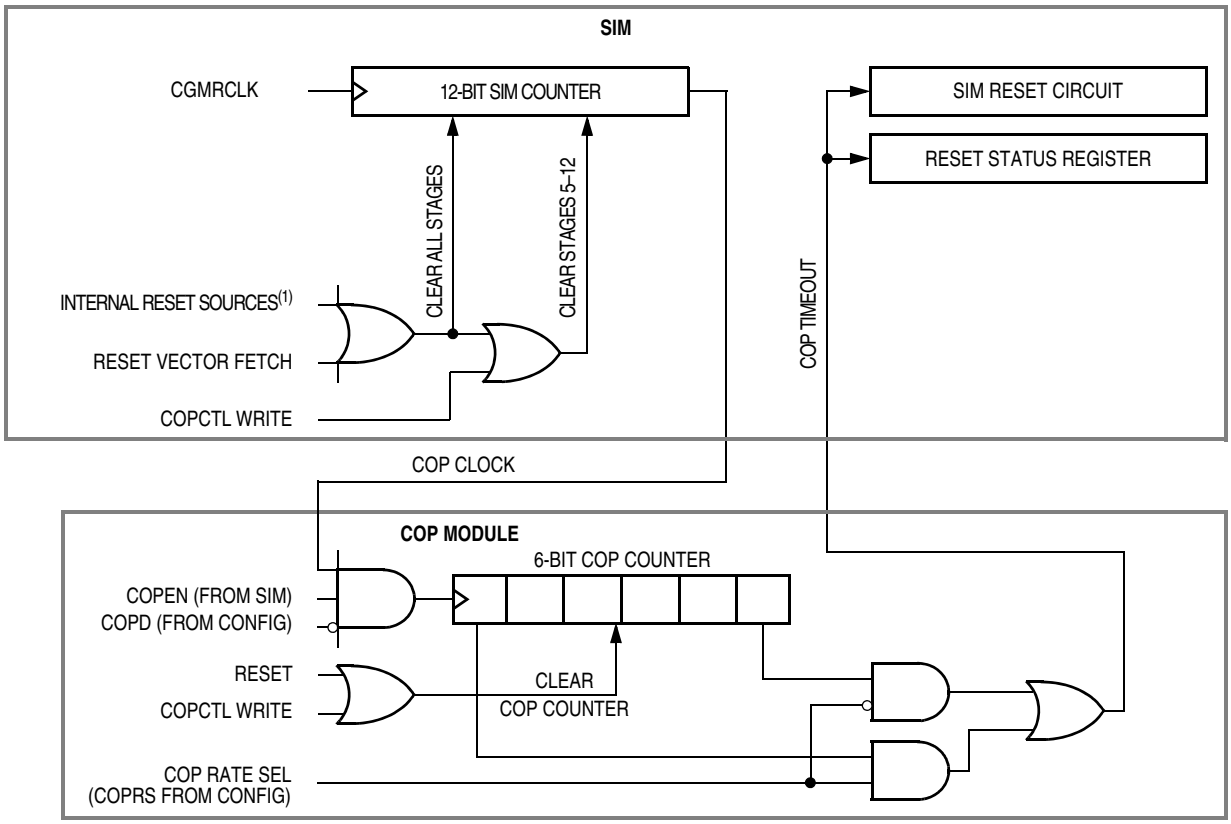


Figure 16-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by a 12-bit system integration module (SIM) counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after 262,128 or 8176 CGMRCLK cycles, depending on the state of the COP rate select bit, COPRS in the configuration register. With a 262,128 CGMRCLK cycle overflow option (COPRS = 0), a 4-MHz external clock source gives a COP timeout period of 66 ms. 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 CGMRCLK cycles and sets the COP bit in the reset status register (RSR).

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

NOTE

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

16.3 I/O Signals

The following paragraphs describe the signals shown in Figure 16-1.

16.3.1 CGMRCLK

CGMRCLK is the reference clock output from the OSC module. If a 4-MHz crystal is used, CGMRCLK is also 4-MHz.

16.3.2 STOP Instruction

The STOP instruction clears the COP prescaler.

16.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see 16.4 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.

16.3.4 Power-On Reset

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

16.3.5 Internal Reset

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

16.3.6 Reset Vector Fetch

A reset vector fetch occurs when the vector address appears on the data bus. A reset vector fetch clears the COP prescaler.

16.3.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register (CONFIG).

16.3.8 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register (CONFIG).

Address:	$001F							
	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD		SSREC	STOP	COPD
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 16-2. Configuration Register (CONFIG)

COPRS — COP Rate Select Bit

COPRS selects the COP timeout period. Reset clears COPRS.

1 = COP timeout period is (8176) × CGMRCLK cycles

0 = COP timeout period is (262,128) × CGMRCLK cycles

COPD — COP Disable Bit

COPD disables the COP module.

1 = COP module disabled

0 = COP module enabled

16.4 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 16-3. COP Control Register (COPCTL)

16.5 Interrupts

The COP does not generate CPU interrupt requests.

16.6 Monitor Mode

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

16.7 Low-Power Modes

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

16.7.1 Wait Mode

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

16.7.2 Stop Mode

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

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

16.8 COP Module During Break Mode

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

Chapter 17
Low-Voltage Inhibit (LVI)

17.1 Introduction

This section describes the low-voltage inhibit (LVI) module. The LVI module monitors the voltage on the V_{DD} pin, and can force a reset when V_{DD} voltage falls below V_{TRIPF1}.

17.2 Features

Features of the LVI module include:

- Independent voltage monitoring circuits for V_{DD}
- Independent LVI circuit disable for V_{DD}
- Programmable LVI reset
- Programmable stop mode operation

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE0F	LVI Status Register (LVISR)	Read:	LVIOUT	0	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-1. LVI I/O Register Summary

17.3 Functional Description

Figure 17-2 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains independent bandgap reference circuit and comparator for monitoring the V_{DD} voltage. An LVI reset performs a MCU internal reset and drives the $\overline{RST}$ pin low to provide low-voltage protection to external peripheral devices.

LVISTOP, LVIPWRD, LVIRSTD are in the CONFIG1 register. See Chapter 3 Configuration Registers (CONFIG) for details of the LVI configuration bits. Once an LVI reset occurs, the MCU remains in reset until V_{DD} rises above V_{TRIPR1} which causes the MCU to exit reset. The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

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

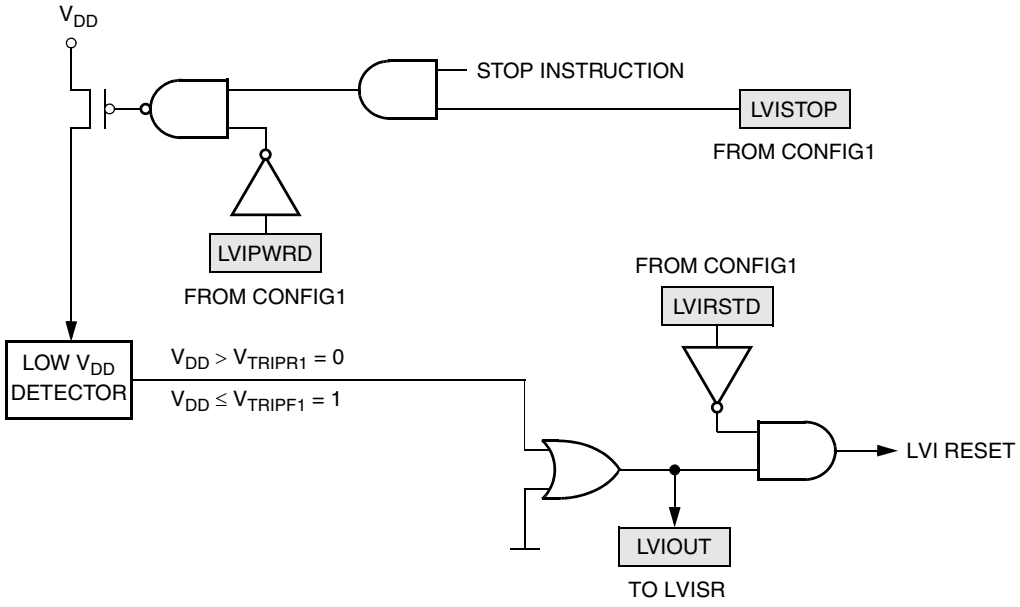


Figure 17-2. LVI Module Block Diagram

17.3.1 Low V_{DD} Detector

The low V_{DD} detector circuit monitors the V_{DD} voltage and forces a LVI reset when the V_{DD} voltage falls below the trip voltage, V_{TRIPF1}. The V_{DD} LVI circuit can be disabled by the setting the LVIPWRD bit in CONFIG1 register.

17.3.2 Polled LVI Operation

In applications that can operate at V_{DD} levels below the V_{TRIPF1} level, software can monitor V_{DD} by polling the LVIOUT bit. In the CONFIG1 register, 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.

17.3.3 Forced Reset Operation

In applications that require V_{DD} to remain above the V_{TRIPF1} level, enabling LVI resets allows the LVI module to reset the MCU when V_{DD} falls below the V_{TRIPF1} level. In the CONFIG1 register, the LVIPWRD and LVIRSTD bits must be at logic 0 to enable the LVI module and to enable LVI resets.

17.3.4 Voltage Hysteresis Protection

Once the LVI has triggered (by having V_{DD} fall below V_{TRIPF1}), the LVI will maintain a reset condition until V_{DD} rises above the rising trip point voltage, V_{TRIPR1}. This prevents a condition in which the MCU is continually entering and exiting reset if V_{DD} is approximately equal to V_{TRIPF1}. V_{TRIPR1} is greater than V_{TRIPF1} by the hysteresis voltage, V_{HYS}.

17.4 LVI Status Register

The LVI status register (LVISR) indicates if the V_{DD} voltage was detected below V_{TRIPF1}.

Address: $FE0F

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

= Unimplemented

Figure 17-3. LVI Status Register

LVIOUT — LVI Output Bit

This read-only flag becomes set when the V_{DD} or V_{REG} falls below their respective trip voltages. Reset clears the LVIOUT bit.

Table 17-1. LVIOUT Bit Indication

V_{DD}, V_{REG}	LVIOUT
$V_{DD} > V_{TRIPR1}$	0
$V_{DD} < V_{TRIPF1}$	1
$V_{TRIPF1} < V_{DD} < V_{TRIPR1}$	Previous value

17.5 LVI Interrupts

The LVI module does not generate interrupt requests.

17.6 Low-Power Modes

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

17.6.1 Wait Mode

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

17.6.2 Stop Mode

If enabled in stop mode (LVISTOP = 1), the LVI module remains active in stop mode. If enabled to generate resets (LVIRSTD = 0), the LVI module can generate a reset and bring the MCU out of stop mode.

Chapter 18
Break Module (BRK)

18.1 Introduction

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

18.2 Features

Features of the break module include:

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							Note	
		Reset:							0	
$FE03	SIM Break Flag Control Register (SBFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE0C	Break Address Register High (BRKH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address Register Low (BRKL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$FE0E	Break Status and Control Register (BRKSCR)	Read:	BRKE	BRKA	0	0	0	0	0	0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

Note: Writing a logic 0 clears BW.

= Unimplemented R = Reserved

Figure 18-1. Break Module I/O Register Summary

18.3 Functional Description

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

The following events can cause a break interrupt to occur:

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

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. Figure 18-2 shows the structure of the break module.

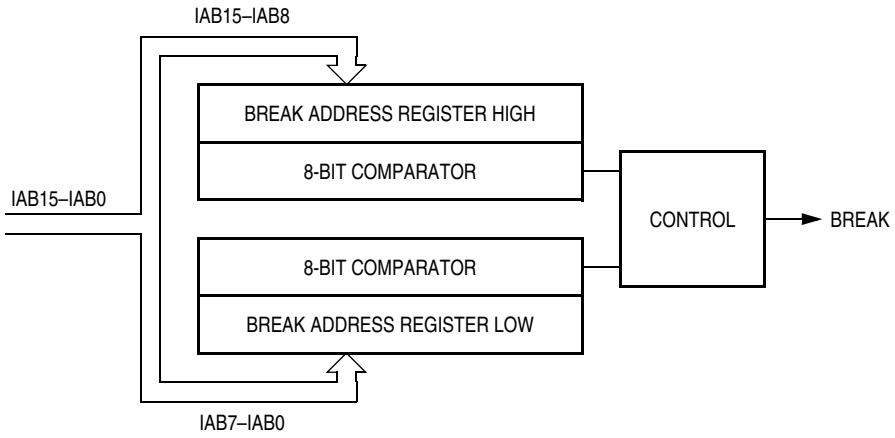


Figure 18-2. Break Module Block Diagram

18.3.1 Flag Protection During Break Interrupts

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

18.3.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with $FFFC and $FFFD ($FEFC and $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.

18.3.3 TIMI and TIM2 During Break Interrupts

A break interrupt stops the timer counters.

18.3.4 COP During Break Interrupts

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

18.4 Low-Power Modes

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

18.4.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if SBSW is set. (See Chapter 6 System Integration Module (SIM).) Clear the BW bit by writing logic 0 to it.

18.4.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the break status register.

18.5 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)
- SIM break status register (SBSR)
- SIM break flag control register (SBFCR)

18.5.1 Break Status and Control Register

The break status and control register (BRKSCR) 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 18-3. Break Status and Control Register (BRKSCR)

BRKE — Break Enable Bit

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

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

BRKA — Break Active Bit

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

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

18.5.2 Break Address Registers

The break address registers (BRKH and BRKL) 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 18-4. 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 18-5. Break Address Register Low (BRKL)

18.5.3 SIM Break Status Register

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

Address: $FE00

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

Note: Writing a logic 0 clears SBSW. R = Reserved

Figure 18-6. SIM Break Status Register (SBSR)

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.

18.5.4 SIM Break Flag Control Register

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

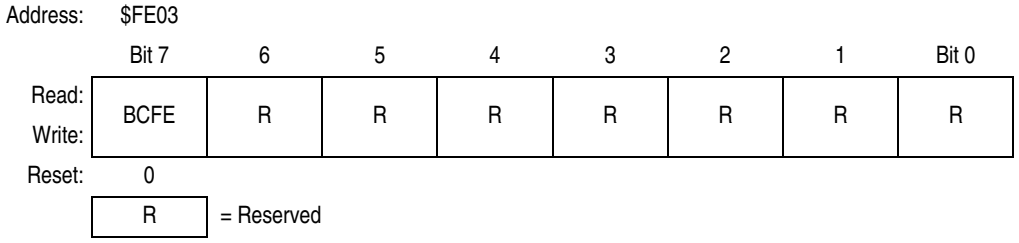


Figure 18-7. SIM Break Flag Control Register (SBFCR)

BCFE — Break Clear Flag Enable Bit

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

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

Chapter 19
Electrical Specifications

19.1 Introduction

This section contains electrical and timing specifications.

19.2 Absolute Maximum Ratings

Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

NOTE

This device is not guaranteed to operate properly at the maximum ratings. Refer to 19.5 DC Electrical Characteristics for guaranteed operating conditions.

Table 19-1. Absolute Maximum Ratings(1)

Characteristic	Symbol	Value	Unit
Supply voltage	V_{DD}	−0.3 to +6.0	V
Input voltage All pins (except $\overline{IRQ}$) $\overline{IRQ}$ pin	V_{IN}	V_{SS}−0.3 to V_{DD} +0.3 V_{SS}−0.3 to 8.5	V
Maximum current per pin excluding V_{DD} and V_{SS}	I	±25	mA
Maximum current out of V_{SS}	I_{MVSS}	100	mA
Maximum current into V_{DD}	I_{MVDD}	100	mA
Storage temperature	T_{STG}	−55 to +150	°C

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 $V_{SS} \le (V_{IN}$ or $V_{OUT}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}.)

19.3 Functional Operating Range

Table 19-2. Operating Range

Characteristic	Symbol	Value	Unit
Operating temperature range	T_A	0 to +70	°C
Operating voltage range	V_{DD}	3.5 to 5.5	V

19.4 Thermal Characteristics

Table 19-3. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 48-Pin QFN	θ_{JA}	84	°C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273\ °C)$	W
Constant(2)	K	$P_D \times (T_A + 273\ °C) + P_D^2 \times \theta_{JA}$	W/°C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C

1. Power dissipation is a function of temperature.
2. K 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.

19.5 DC Electrical Characteristics

Table 19-4. DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
2.5V Regulator Output Voltage	V_{REG25}	2.25	2.5	2.75	V
3.3V Regulator Output Voltage (At V_{DD} from 3.9V–5.5V)[3]	V_{REG33}	3.0	3.3	3.6	V
Output high voltage (I_{LOAD} = −2.0 mA) All ports	V_{OH}	V_{DD}–0.8	—	—	V
Output low voltage (I_{LOAD} = 1.6mA) All ports	V_{OL}	—	—	0.4	V
Output Sourcing Capability PTB0–PTB1 (V_{OL} = 0.4V) PTB5 (V_{OL} = 0.4V) PTE2–PTE3 (V_{OL} = 0.4V)	I_{LOAD}	 18 12 8	 26 18 12	 34 24 16	 mA mA mA
Input high voltage All ports, $\overline{RST}$, $\overline{IRQ}$	V_{IH}	0.7 × V_{DD}	—	V_{DD}	V
Input low voltage All ports, $\overline{RST}$, $\overline{IRQ}$	V_{IL}	V_{SS}	—	0.3 × V_{DD}	V
V_{DD} supply current Run[4] (0°C-70°C), f_{OP} = 8 MHz (all modules on including USB) Wait[5] (0°C-70°C), f_{OP} = 8 MHz (all modules on including USB) Stop (0°C-70°C) with RC on, LVI on and all other modules off Stop (0°C-70°C) with RC off, LVI on and all other modules off	I_{DD}	 — — — —	 — — — —	 18 16 350 280	 mA mA µA µA
Digital I/O ports Hi-Z leakage current All ports, $\overline{RST}$	I_{IL}	—	—	± 10	µA
Input current $\overline{IRQ}$	I_{IN}	—	—	± 1	µA
Capacitance Ports (as input or output)	C_{OUT} C_{IN}	— —	— —	12 8	pF
POR rise-time ramp rate[6]	R_{POR}	0.035	—	—	V/ms
POR assert voltage[7]	V_{POR_assert}	0.90	1.62	2.1	V
Monitor mode entry voltage (at $\overline{IRQ}$ pin)	V_{TST}	1.5 × V_{DD}	—	8	V

Table 19-4. DC Electrical Characteristics (Continued)

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Pullup resistors[8]					
PTA0–PTA7 configured as KBI0–KBI7	R_{PU1}	21	30	39	kΩ
$\overline{RST}$, $\overline{IRQ}$, PTD2, PTD3, PTD7	R_{PU2}	21	30	39	kΩ
PTE2–PTE3 with USB disabled	R_{PU3}	4	5	6	kΩ
PTE2/D+ with USB enabled (to REG33V)[9]	R_{PU4}(Idle)	900	—	1575	Ω
PTE3/D– with USB enabled (to REG33V)[10]	R_{PU4}(Tran)	1425	—	3090	Ω
PTB0–PTB1, PTB5 with internal pullup enabled	R_{PU5}	21	30	39	kΩ
Low-voltage inhibit for external VDD, trip falling voltage (kick-in)	V_{TRIPF1}	3.0	3.3	3.5	V
Low-voltage inhibit for external VDD, trip rising voltage (recovery)	V_{TRIPR1}	3.07	3.4	3.6	V

1. V_{DD} = 3.9 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. When V_{DD} drops below 3.9V, the VREF33 regulator output will not be guaranteed within 3.3V +/- 10%.
4. Run (operating) I_{DD} measured using external square wave clock source. 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}.
5. Wait I_{DD} measured using external square wave clock source. 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}.
6. 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.
7. The internal 2.5V regulator has embedded a LVI_POR circuitry when the regulator voltage drops below $V_{LVI\_POR\_assert}$ voltage it triggers the CPU reset. The reset is released when the regulator voltage returns above $V_{LVI\_POR\_release}$ voltage.
8. R_{PU1} and R_{PU2} are measured at V_{DD} = 5.0V
9. The resistor value is measured at V_{DD} = 3.9 to 5.5 Vdc, V_{SS} = 0 Vdc.
10. The resistor value is measured at V_{DD} = 3.9 to 5.5 Vdc, V_{SS} = 0 Vdc.

19.6 Control Timing

Table 19-5. Control Timing

Characteristic[1]	Symbol	Min	Max	Unit
Internal operating frequency[2]	f_{OP}	—	8	MHz
$\overline{RST}$ input pulse width low[3]	t_{IRL}	750	—	ns

1. V_{SS} = 0 Vdc; 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.

19.7 Internal RC Clock Timing

Table 19-6. Internal RC Clock Timing

Characteristic[1]	Symbol	Min	TYP	Max	Unit
Internal RC Clock frequency	f_{OP}	74	88	105	kHz

1. V_{SS} = 0 Vdc; timing shown with respect to 20% V_{DD} and 70% V_{DD}, unless otherwise noted.

19.8 Crystal Oscillator Characteristics

Table 19-7. Oscillator Characteristics

Characteristic	Symbol	Min	Typ	Max	Unit
Crystal frequency[1]	f_{XCLK}	1	—	4	MHz
External clock Reference frequency[1, 2]	f_{XCLK}	dc	—	4	MHz
Crystal load capacitance[3]	C_L	—	—	—	
Crystal fixed capacitance[3]	C_1	—	$2 \times C_L$	—	
Crystal tuning capacitance[3]	C_2	—	$2 \times C_L$	—	
Feedback bias resistor	R_B	—	10 MΩ	—	
Series resistor[3, 4]	R_S	—	—	—	

1. The USB module is designed to function at f_{XCLK} = 4MHz.
2. No more than 10% duty cycle deviation from 50%.
3. Consult crystal vendor data sheet.
4. Not required for high-frequency crystals.

19.9 USB DC Electrical Characteristic

The USB electrical performance is compliant to the USB specification 2.0.

Table 19-8. USB DC Electrical Characteristics

Characteristic[1]	Symbol	Conditions	Min	Typ	Max	Unit
Hi-Z state data line leakage	I_{LO}	$0V<V_{IN}<3.3V$	–10		+10	μA
Voltage input high (driven)	V_{IH}		2.0			V
Voltage input high (floating)	V_{IHZ}		2.7		3.6	V
Voltage input low	V_{IL}				0.8	V
Differential input sensitivity	V_{DI}	|(D+) – (D–)|	0.2			V
Differential common mode range	V_{CM}	Includes V_{DI} Range	0.8		2.5	V
Static output low	V_{OL}	R_L of 1.425 K to 3.6 V			0.3	V
Static output high	V_{OH}	R_L of 14.25 K to GND	2.8		3.6	V
Output signal crossover voltage	V_{CRS}		1.3	—	2.0	V
Regulator bypass capacitor	$C_{REGBYPASS}$			0.47		μF
Regulator bulk capacitor	$C_{REGBULK}$		4.7			μF

1. V_{DD} = 3.9 to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.

19.10 Timer Interface Module Characteristics

Table 19-9. Timer Interface Characteristics

Characteristic	Symbol	Min	Max	Unit
Input capture pulse width	t_{TIH}, t_{TIL}	1	—	t_{CYC}

19.11 FLASH Program/Erase Timing

Table 19-10. Flash Program/Erase Timing

Characteristic	Symbol	Min	Max	Unit
PROG/ERASE to NVSTR setup time	T_{nvs}	5	—	μs
NVSTR hold time	T_{nvh}	5	—	μs
NVSTR hold time (mass erase)	T_{nvhl}	100	—	μs
NVSTR to program setup time	T_{pgs}	10	—	μs
Program Time	T_{prog}	20	40	μs
Page Erase Time	T_{erase}	20	—	ms
Mass Erase Time	T_{me}	200	—	ms
Recovery time	T_{rcv}	1	—	μs
Accumulative program HV period	T_{hv}	—	8	ms

19.12 CGM Electrical Specifications

Table 19-11. CGM Electrical Specifications

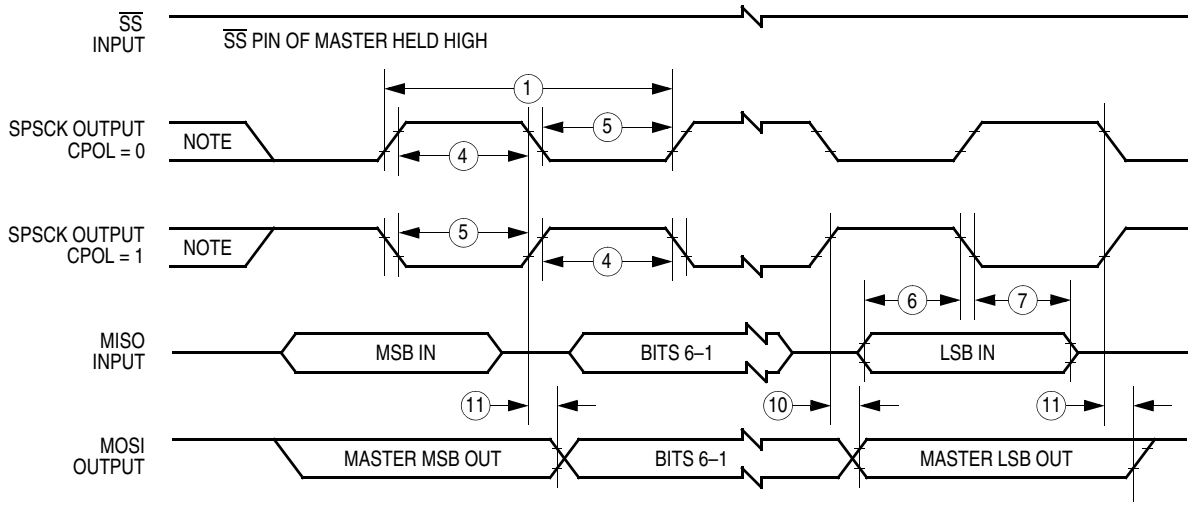
Characteristic	Symbol	Min	Typ	Max	Unit
Operating Voltage	V_{DD}	3.5	—	5.5	V
Reference frequency	f_{RDV}	—	4	—	MHz
VCO center-of-range frequency	f_{VRS}	—	48M	—	Hz
VCO multiply factor	N	1	—	6	
VCO prescale multiplier	2^P	0	—	1	
Reference divider factor	R	1	1	1	
VCO operating frequency	f_{VCLK}	24	—	48	MHz
Manual acquisition time	t_{LOCK}	—	—	5	ms
Automatic lock time	t_{LOCK}	—	—	5	ms

19.13 5.0V SPI Characteristics

Table 19-12. SPI Characteristics

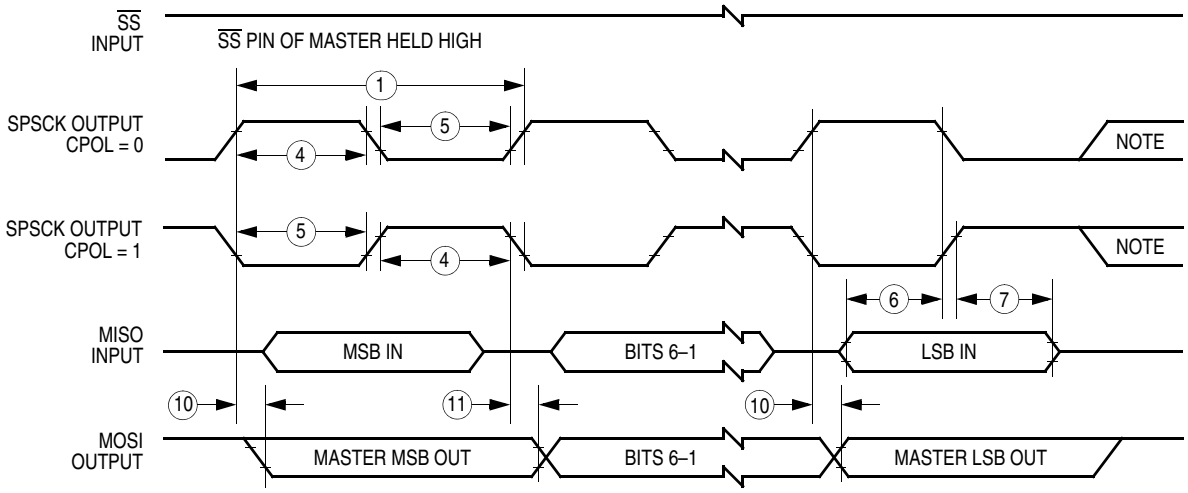
Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	 $f_{OP(M)}$ $f_{OP(S)}$	 $f_{OP}/128$ dc	 $f_{OP}/2$ f_{OP}	 MHz MHz
1	Cycle time Master Slave	 $t_{CYC(M)}$ $t_{CYC(S)}$	 2 1	 128 —	 t_{CYC} t_{CYC}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{CYC}
3	Enable lag time	$t_{Lag(S)}$	1	—	t_{CYC}
4	Clock (SPSCK) high time Master Slave	 $t_{SCKH(M)}$ $t_{SCKH(S)}$	 $t_{CYC} - 25$ $1/2\ t_{CYC} - 25$	 $64\ t_{CYC}$ —	 ns ns
5	Clock (SPSCK) low time Master Slave	 $t_{SCKL(M)}$ $t_{SCKL(S)}$	 $t_{CYC} - 25$ $1/2\ t_{CYC} - 25$	 $64\ t_{CYC}$ —	 ns ns
6	Data setup time (inputs) Master Slave	 $t_{SU(M)}$ $t_{SU(S)}$	 30 30	 — —	 ns ns
7	Data hold time (inputs) Master Slave	 $t_{H(M)}$ $t_{H(S)}$	 30 30	 — —	 ns ns
8	Access time, slave[3] CPHA = 0 CPHA = 1	 $t_{A(CP0)}$ $t_{A(CP1)}$	 0 0	 40 40	 ns ns
9	Disable time, slave[4]	$t_{DIS(S)}$	—	40	ns
10	Data valid time, after enable edge Master Slave[5]	 $t_{V(M)}$ $t_{V(S)}$	 — —	 50 50	 ns ns
11	Data hold time, outputs, after enable edge Master Slave	 $t_{HO(M)}$ $t_{HO(S)}$	 0 0	 — —	 ns ns

1. Numbers refer to dimensions in Figure 19-1 and Figure 19-2.
2. All timing is shown with respect to 20% V_{DD} and 70% V_{DD}, unless noted; 100 pF load on all SPI pins.
3. Time to data active from high-impedance state
4. Hold time to high-impedance state
5. With 100 pF on all SPI pins



Note: This first clock edge is generated internally, but is not seen at the SPSCK pin.

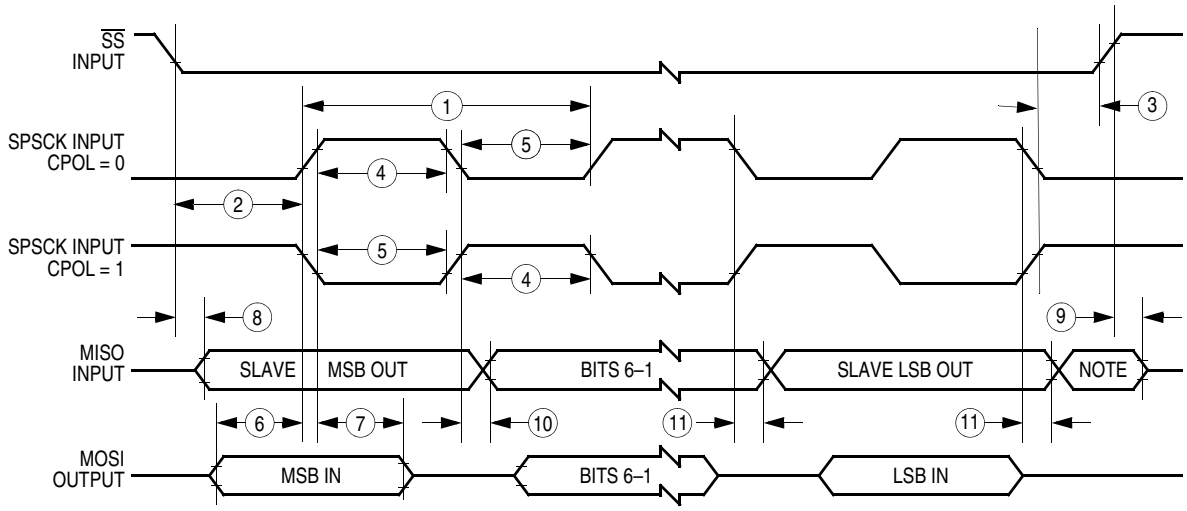
a) SPI Master Timing (CPHA = 0)



Note: This last clock edge is generated internally, but is not seen at the SPSCK pin.

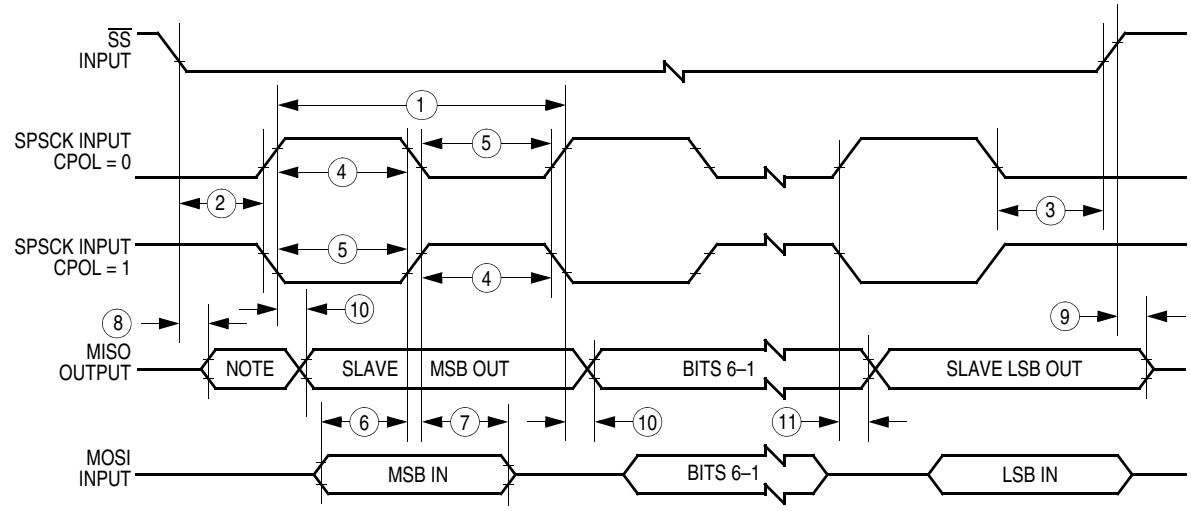
b) SPI Master Timing (CPHA = 1)

Figure 19-1. SPI Master Timing



Note: Not defined but normally MSB of character just received

a) SPI Slave Timing (CPHA = 0)



Note: Not defined but normally LSB of character previously transmitted

b) SPI Slave Timing (CPHA = 1)

Figure 19-2. SPI Slave Timing

Chapter 20
Ordering Information and Mechanical Specifications

20.1 Introduction

This section contains ordering information for the MC68HC908JW32. In addition, this section gives the package dimensions for the 48-pin quad flat non-leaded package.

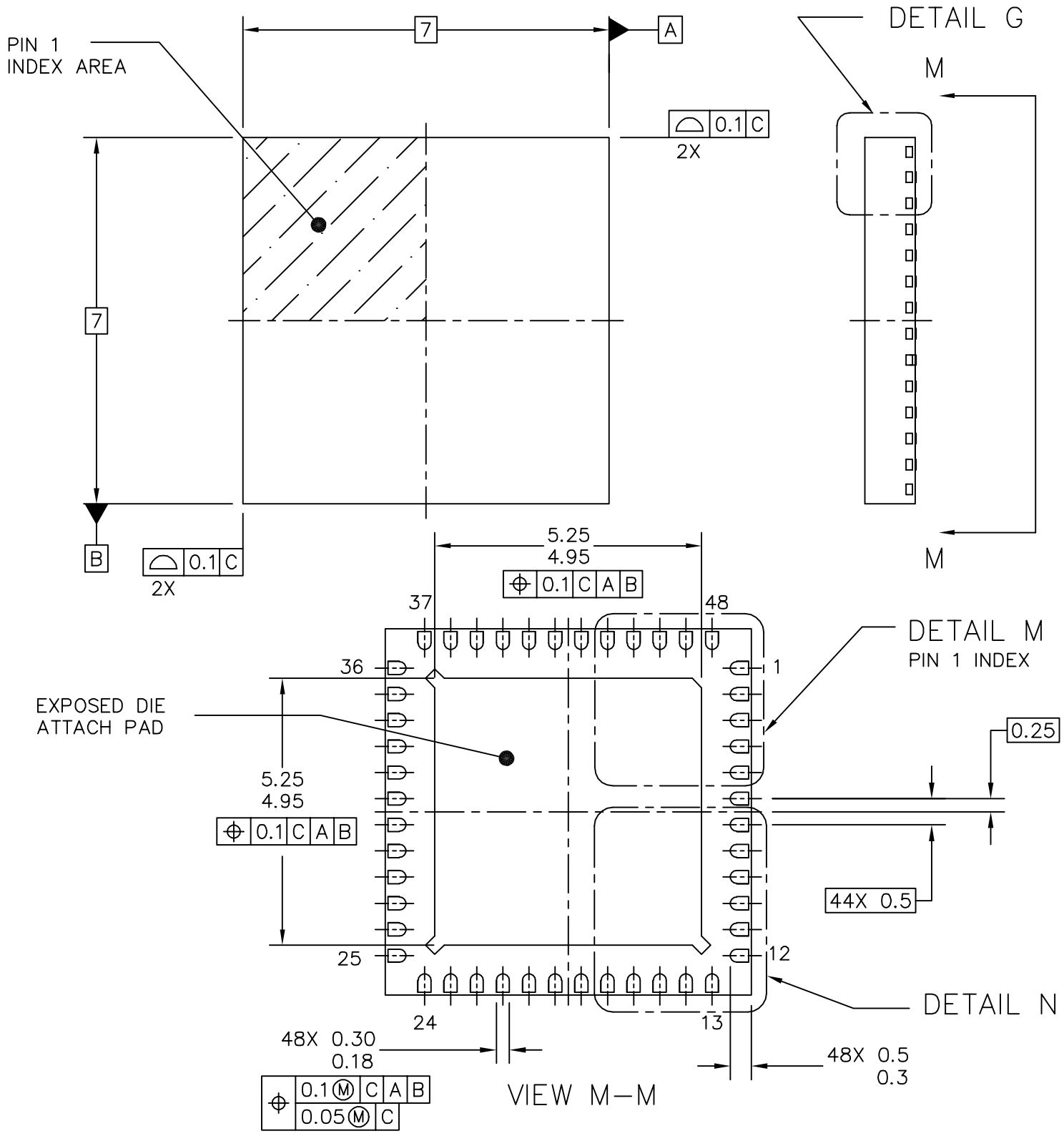
20.2 Ordering Information

Table 20-1. MC Order Numbers

MC Order Number	Package	Operating Temperature Range
MCHC908JW32FC	48-pin QFN	0 to +70 °C

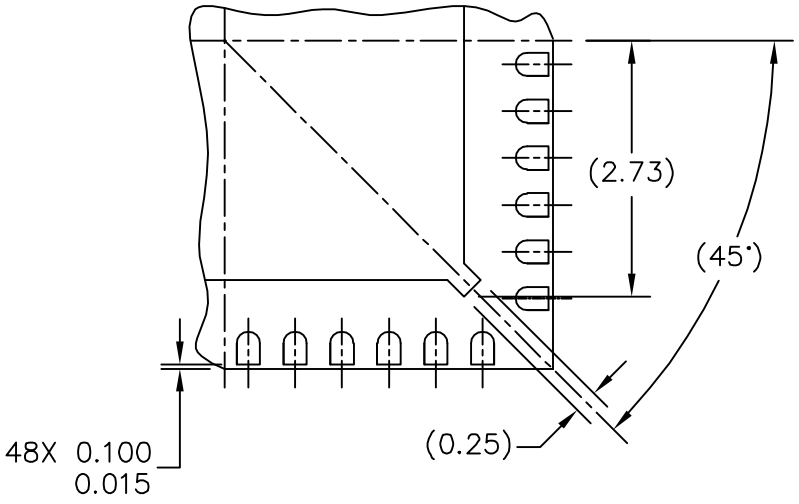
20.3 Package Dimensions

Refer to the following pages for detailed package dimensions.

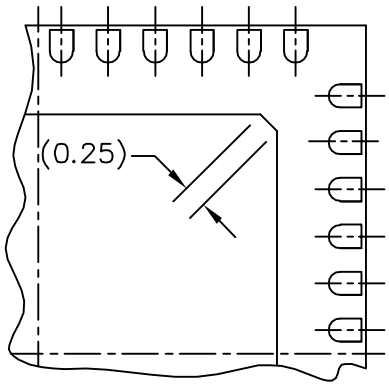


MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

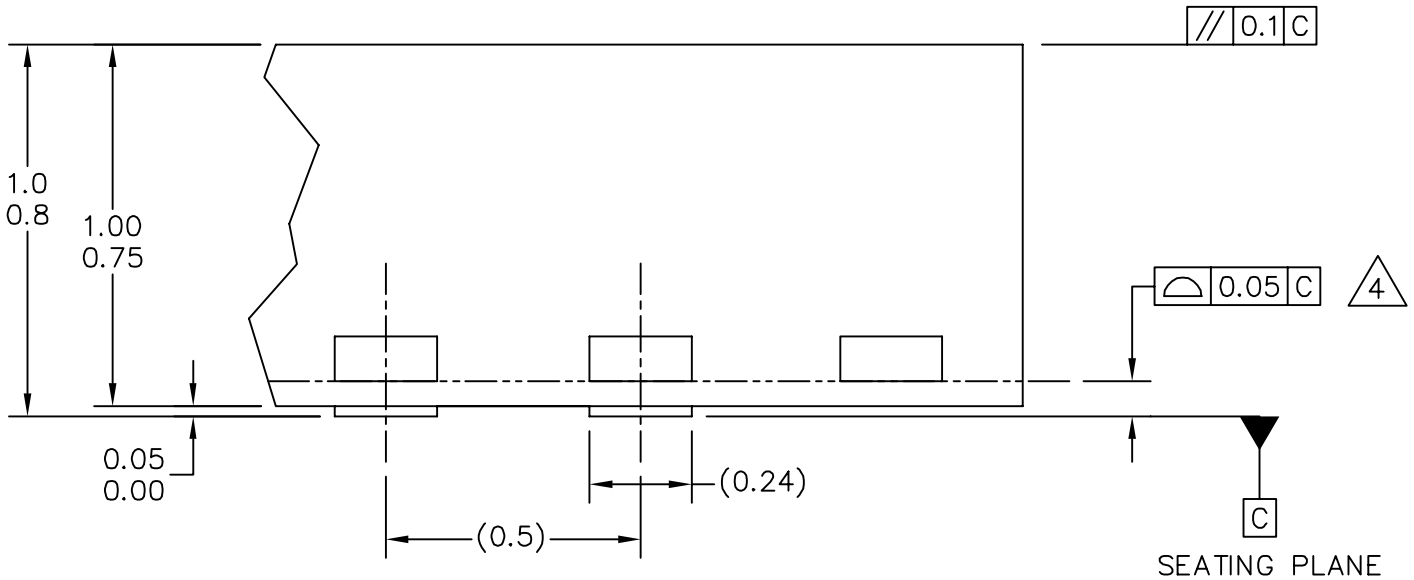
TITLE: THERMALLY ENHANCED QUAD FLAT NON–LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)	DOCUMENT NO: 98ARH99048A	REV: F
	CASE NUMBER: 1314–05	05 DEC 2005
	STANDARD: JEDEC–MO–220 VKKD–2	



DETAIL N
PREFERRED CORNER CONFIGURATION

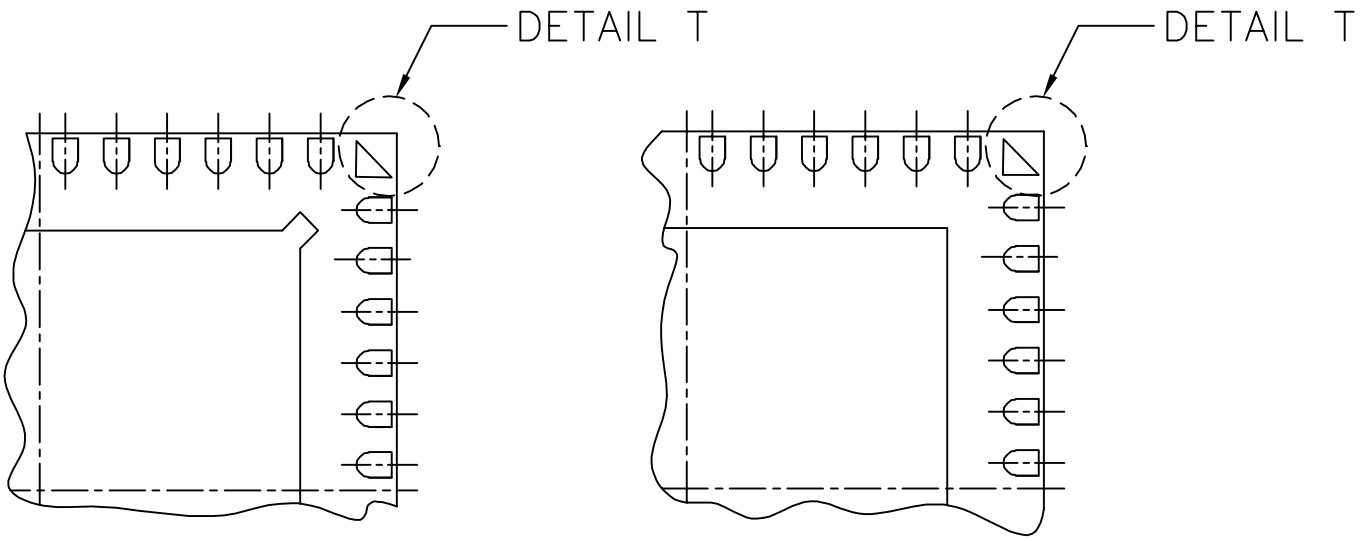


DETAIL M
PREFERED PIN 1 BACKSIDE IDENTIFIER



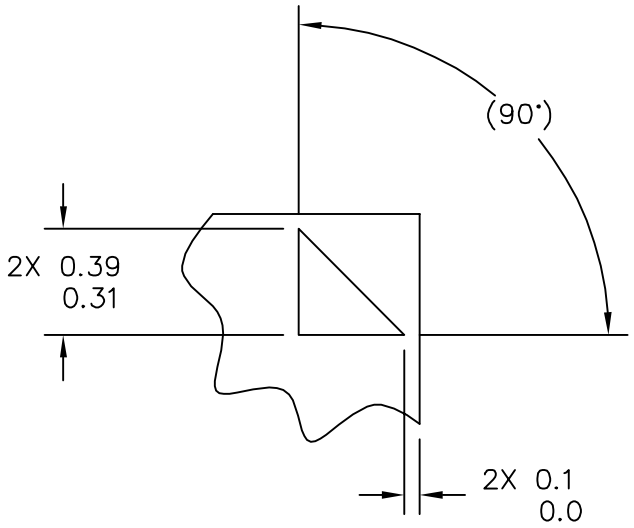
DETAIL G
VIEW ROTATED 90° CW

	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)	DOCUMENT NO: 98ARH99048A	REV: F
	CASE NUMBER: 1314-05	05 DEC 2005
	STANDARD: JEDEC-MO-220 VKKD-2	



DETAIL M
PIN 1 BACKSIDE IDENTIFIER OPTION

DETAIL M
PIN 1 BACKSIDE IDENTIFIER OPTION



DETAIL T

MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE

TITLE: THERMALLY ENHANCED QUAD FLAT NON-LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)	DOCUMENT NO: 98ARH99048A	REV: F
	CASE NUMBER: 1314-05	05 DEC 2005
	STANDARD: JEDEC-MO-220 VKKD-2	

NOTES:

1. DIMENSIONS ARE IN MILLIMETERS.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M–1994.
3. THE COMPLETE JEDEC DESIGNATOR FOR THIS PACKAGE IS: HF–PQFN.
4. COPLANARITY APPLIES TO LEADS, CORNER LEADS, AND DIE ATTACH PAD.
5. MIN METAL GAP SHOULD BE 0.2MM.

	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: THERMALLY ENHANCED QUAD FLAT NON–LEADED PACKAGE (QFN) 48 TERMINAL, 0.5 PITCH (7 X 7 X 1)	DOCUMENT NO: 98ARH99048A	REV: F
	CASE NUMBER: 1314–05	05 DEC 2005
	STANDARD: JEDEC–MO–220 VKKD–2	

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.

MC68HC908JW32
Rev. 5, 10/2006

freescale™
semiconductor