MC68HC908AP64A
MC68HC908AP32A
MC68HC908AP16A
MC68HC908AP8A

Data Sheet

M68HC08
Microcontrollers

MC68HC908AP64A
Rev. 3
10/2007

freescale.com

freescale™
semiconductor

MC68HC908AP64A
MC68HC908AP32A
MC68HC908AP16A
MC68HC908AP8A

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.

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc.
This product incorporates SuperFlash® technology licensed from SST.

© Freescale Semiconductor, Inc., 2005, 2007. All rights reserved.

Revision History

Date	Revision Level	Description	Page Number(s)
October 2007	3	Clarified or updated information in Chapter 7 System Integration Module (SIM). In Chapter 8 Monitor Mode (MON), corrected and updated monitor mode entry details. In Chapter 9 Timer Interface Module (TIM), updated functional details. In 15.7 I/O Registers, corrected COCO bit description. In 16.1 Introduction, added unused pin information. In Chapter 21 Break Module (BRK), updated functional details. In 22.5 5V DC Electrical Characteristics, updated stop I_{DD} values.	—
January 2007	2	15.7.2 ADC Clock Control Register — Changed “The ADC clock should be set to between 500 kHz and 2 MHz” to “The ADC clock should be set to between 500 kHz and 1 MHz”	250
Mar 2005	1	First general release.	—

List of Chapters

Chapter 1 General Description. 19
Chapter 2 Memory 29
Chapter 3 Configuration & Mask Option Registers (CONFIG & MOR) 49
Chapter 4 Central Processor Unit (CPU). 55
Chapter 5 Oscillator (OSC) 71
Chapter 6 Clock Generator Module (CGM) 79
Chapter 7 System Integration Module (SIM). 97
Chapter 8 Monitor Mode (MON) 115
Chapter 9 Timer Interface Module (TIM) 131
Chapter 10 Timebase Module (TBM) 147
Chapter 11 Serial Communications Interface Module (SCI) 151
Chapter 12 Infrared Serial Communications Interface Module (IRSCI) 177
Chapter 13 Serial Peripheral Interface Module (SPI) 207
Chapter 14 Multi-Master IIC Interface (MMIIC) 227
Chapter 15 Analog-to-Digital Converter (ADC). 245
Chapter 16 Input/Output (I/O) Ports. 257
Chapter 17 External Interrupt (IRQ). 269
Chapter 18 Keyboard Interrupt Module (KBI) 275
Chapter 19 Computer Operating Properly (COP). 281
Chapter 20 Low-Voltage Inhibit (LVI). 285
Chapter 21 Break Module (BRK) 289
Chapter 22 Electrical Specifications 295
Chapter 23 Mechanical Specifications 309
Chapter 24 Ordering Information. 313

Table of Contents

Chapter 1
General Description

1.1 Introduction . . . 19
1.2 Features . . . 19
1.3 MCU Block Diagram . . . 20
1.4 Pin Assignment . . . 22
1.5 Pin Functions . . . 25
1.6 Power Supply Bypassing (V_{DD}, V_{DDA}, V_{SS}, V_{SSA}) . . . 26
1.7 Regulator Power Supply Configuration (VREG) . . . 27

Chapter 2
Memory

2.1 Introduction . . . 29
2.2 Input/Output (I/O) Section . . . 29
2.3 Monitor ROM . . . 29
2.4 Random-Access Memory (RAM) . . . 41
2.5 FLASH Memory . . . 42
2.5.1 Functional Description . . . 42
2.5.2 FLASH Control Register . . . 43
2.5.3 FLASH Page Erase Operation . . . 43
2.5.4 FLASH Mass Erase Operation . . . 44
2.5.5 FLASH Program Operation . . . 44
2.5.6 FLASH Protection . . . 45
2.5.7 FLASH Block Protect Register . . . 47

Chapter 3
Configuration & Mask Option Registers (CONFIG & MOR)

3.1 Introduction . . . 49
3.2 Functional Description . . . 50
3.3 Configuration Register 1 (CONFIG1) . . . 50
3.4 Configuration Register 2 (CONFIG2) . . . 52
3.5 Mask Option Register (MOR) . . . 53

Chapter 4
Central Processor Unit (CPU)

4.1 Introduction . . . 55
4.2 Features . . . 55
4.3 CPU Registers . . . 56
4.3.1 Accumulator . . . 56

4.3.2 Index Register . . . 56
4.3.3 Stack Pointer . . . 57
4.3.4 Program Counter . . . 57
4.3.5 Condition Code Register . . . 58
4.4 Arithmetic/Logic Unit (ALU) . . . 59
4.5 Low-Power Modes . . . 59
4.5.1 Wait Mode . . . 59
4.5.2 Stop Mode . . . 59
4.6 CPU During Break Interrupts . . . 60
4.7 Instruction Set Summary . . . 60
4.8 Opcode Map . . . 60

Chapter 5
Oscillator (OSC)

5.1 Introduction . . . 71
5.2 Clock Selection . . . 71
5.2.1 CGM Reference Clock Selection . . . 72
5.2.2 TBM Reference Clock Selection . . . 73
5.3 Internal Oscillator . . . 74
5.4 RC Oscillator . . . 75
5.5 X-tal Oscillator . . . 75
5.6 I/O Signals . . . 76
5.6.1 Crystal Amplifier Input Pin (OSC1) . . . 76
5.6.2 Crystal Amplifier Output Pin (OSC2) . . . 76
5.6.3 Oscillator Enable Signal (SIMOSCEN) . . . 76
5.6.4 CGM Oscillator Clock (CGMXCLK) . . . 77
5.6.5 CGM Reference Clock (CGMRCLK) . . . 77
5.6.6 Oscillator Clock to Time Base Module (OSCCLK) . . . 77
5.7 Low Power Modes . . . 77
5.7.1 Wait Mode . . . 77
5.7.2 Stop Mode . . . 77
5.8 Oscillator During Break Mode . . . 77

Chapter 6
Clock Generator Module (CGM)

6.1 Introduction . . . 79
6.2 Features . . . 79
6.3 Functional Description . . . 79
6.3.1 Oscillator Module . . . 81
6.3.2 Phase-Locked Loop Circuit (PLL) . . . 81
6.3.3 PLL Circuits . . . 81
6.3.4 Acquisition and Tracking Modes . . . 82
6.3.5 Manual and Automatic PLL Bandwidth Modes . . . 83
6.3.6 Programming the PLL . . . 83
6.3.7 Special Programming Exceptions . . . 86
6.3.8 Base Clock Selector Circuit . . . 86

6.3.9 CGM External Connections . . . 86
6.4 I/O Signals . . . 87
6.4.1 External Filter Capacitor Pin (CGMXFC) . . . 87
6.4.2 PLL Analog Power Pin (V_{DDA}) . . . 87
6.4.3 PLL Analog Ground Pin (V_{SSA}) . . . 87
6.4.4 Oscillator Output Frequency Signal (CGMXCLK) . . . 88
6.4.5 CGM Reference Clock (CGMRCLK) . . . 88
6.4.6 CGM VCO Clock Output (CGMVCLK) . . . 88
6.4.7 CGM Base Clock Output (CGMOUT) . . . 88
6.4.8 CGM CPU Interrupt (CGMINT) . . . 88
6.5 CGM Registers . . . 88
6.5.1 PLL Control Register . . . 89
6.5.2 PLL Bandwidth Control Register . . . 91
6.5.3 PLL Multiplier Select Registers . . . 92
6.5.4 PLL VCO Range Select Register . . . 92
6.5.5 PLL Reference Divider Select Register . . . 93
6.6 Interrupts . . . 93
6.7 Special Modes . . . 94
6.7.1 Wait Mode . . . 94
6.7.2 Stop Mode . . . 94
6.7.3 CGM During Break Interrupts . . . 94
6.8 Acquisition/Lock Time Specifications . . . 95
6.8.1 Acquisition/Lock Time Definitions . . . 95
6.8.2 Parametric Influences on Reaction Time . . . 95
6.8.3 Choosing a Filter . . . 96

Chapter 7
System Integration Module (SIM)

7.1 Introduction . . . 97
7.2 SIM Bus Clock Control and Generation . . . 99
7.2.1 Bus Timing . . . 99
7.2.2 Clock Start-up from POR or LVI Reset . . . 100
7.2.3 Clocks in Stop Mode and Wait Mode . . . 100
7.3 Reset and System Initialization . . . 100
7.3.1 External Pin Reset . . . 100
7.3.2 Active Resets from Internal Sources . . . 101
7.3.2.1 Power-On Reset . . . 101
7.3.2.2 Computer Operating Properly (COP) Reset . . . 102
7.3.2.3 Illegal Opcode Reset . . . 102
7.3.2.4 Illegal Address Reset . . . 103
7.3.2.5 Low-Voltage Inhibit (LVI) Reset . . . 103
7.3.2.6 Monitor Mode Entry Module Reset . . . 103
7.4 SIM Counter . . . 103
7.4.1 SIM Counter During Power-On Reset . . . 103
7.4.2 SIM Counter During Stop Mode Recovery . . . 103
7.4.3 SIM Counter and Reset States . . . 104
7.5 Exception Control . . . 104

7.5.1 Interrupts 104
7.5.1.1 Hardware Interrupts 105
7.5.1.2 SWI Instruction 106
7.5.2 Interrupt Status Registers 106
7.5.2.1 Interrupt Status Register 1 107
7.5.2.2 Interrupt Status Register 2 107
7.5.2.3 Interrupt Status Register 3 107
7.5.3 Reset 109
7.5.4 Break Interrupts 109
7.5.5 Status Flag Protection in Break Mode 109
7.6 Low-Power Modes 109
7.6.1 Wait Mode 109
7.6.2 Stop Mode 110
7.7 SIM Registers 111
7.7.1 SIM Break Status Register 112
7.7.2 SIM Reset Status Register 113
7.7.3 SIM Break Flag Control Register 114

Chapter 8
Monitor Mode (MON)

8.1 Introduction 115
8.2 Features 115
8.3 Functional Description 115
8.3.1 Entering Monitor Mode 117
8.3.2 Data Format 119
8.3.3 Break Signal 120
8.3.4 Baud Rate 120
8.3.5 Commands 120
8.4 Security 124
8.5 ROM-Resident Routines 126
8.5.1 PRGRNGE 127
8.5.2 ERARNGE 128
8.5.3 LDRNGE 129
8.5.4 MON_PRGRNGE 130
8.5.5 MON_ERARNGE 130

Chapter 9
Timer Interface Module (TIM)

9.1 Introduction 131
9.2 Features 131
9.3 Pin Name Conventions 131
9.4 Functional Description 132
9.4.1 TIM Counter Prescaler 134
9.4.2 Input Capture 134
9.4.3 Output Compare 134
9.4.3.1 Unbuffered Output Compare 134
9.4.3.2 Buffered Output Compare 135

9.4.4 Pulse Width Modulation (PWM) . . . 135
9.4.4.1 Unbuffered PWM Signal Generation . . . 136
9.4.4.2 Buffered PWM Signal Generation . . . 137
9.4.4.3 PWM Initialization . . . 137
9.5 Interrupts . . . 138
9.6 Low-Power Modes . . . 138
9.6.1 Wait Mode . . . 138
9.6.2 Stop Mode . . . 138
9.7 TIM During Break Interrupts . . . 138
9.8 I/O Signals . . . 139
9.9 I/O Registers . . . 139
9.9.1 TIM Status and Control Register . . . 139
9.9.2 TIM Counter Registers . . . 141
9.9.3 TIM Counter Modulo Registers . . . 141
9.9.4 TIM Channel Status and Control Registers . . . 142
9.9.5 TIM Channel Registers . . . 145

Chapter 10
Timebase Module (TBM)

10.1 Introduction . . . 147
10.2 Features . . . 147
10.3 Functional Description . . . 147
10.4 Timebase Register Description . . . 148
10.5 Interrupts . . . 150
10.6 Low-Power Modes . . . 150
10.6.1 Wait Mode . . . 150
10.6.2 Stop Mode . . . 150

Chapter 11
Serial Communications Interface Module (SCI)

11.1 Introduction . . . 151
11.2 Features . . . 151
11.3 Pin Name Conventions . . . 152
11.4 Functional Description . . . 153
11.4.1 Data Format . . . 154
11.4.2 Transmitter . . . 154
11.4.2.1 Character Length . . . 155
11.4.2.2 Character Transmission . . . 155
11.4.2.3 Break Characters . . . 155
11.4.2.4 Idle Characters . . . 156
11.4.2.5 Inversion of Transmitted Output . . . 156
11.4.2.6 Transmitter Interrupts . . . 156
11.4.3 Receiver . . . 156
11.4.3.1 Character Length . . . 156
11.4.3.2 Character Reception . . . 156
11.4.3.3 Data Sampling . . . 158

11.4.3.4 Framing Errors . . . 160
11.4.3.5 Baud Rate Tolerance . . . 160
11.4.3.6 Receiver Wakeup . . . 161
11.4.3.7 Receiver Interrupts . . . 162
11.4.3.8 Error Interrupts . . . 162
11.5 Low-Power Modes . . . 163
11.5.1 Wait Mode . . . 163
11.5.2 Stop Mode . . . 163
11.6 SCI During Break Module Interrupts . . . 163
11.7 I/O Signals . . . 163
11.7.1 TxD (Transmit Data) . . . 163
11.7.2 RxD (Receive Data) . . . 164
11.8 I/O Registers . . . 164
11.8.1 SCI Control Register 1 . . . 165
11.8.2 SCI Control Register 2 . . . 167
11.8.3 SCI Control Register 3 . . . 169
11.8.4 SCI Status Register 1 . . . 170
11.8.5 SCI Status Register 2 . . . 173
11.8.6 SCI Data Register . . . 173
11.8.7 SCI Baud Rate Register . . . 174

Chapter 12
Infrared Serial Communications Interface Module (IRSCI)

12.1 Introduction . . . 177
12.2 Pin Name Conventions . . . 178
12.3 IRSCI Module Overview . . . 179
12.4 Infrared Functional Description . . . 179
12.4.1 Infrared Transmit Encoder . . . 180
12.4.2 Infrared Receive Decoder . . . 180
12.5 SCI Functional Description . . . 181
12.5.1 Data Format . . . 182
12.5.2 Transmitter . . . 182
12.5.2.1 Character Length . . . 182
12.5.2.2 Character Transmission . . . 183
12.5.2.3 Break Characters . . . 184
12.5.2.4 Idle Characters . . . 184
12.5.2.5 Transmitter Interrupts . . . 184
12.5.3 Receiver . . . 185
12.5.3.1 Character Length . . . 186
12.5.3.2 Character Reception . . . 187
12.5.3.3 Data Sampling . . . 187
12.5.3.4 Framing Errors . . . 189
12.5.3.5 Baud Rate Tolerance . . . 189
12.5.3.6 Receiver Wakeup . . . 191
12.5.3.7 Receiver Interrupts . . . 191
12.5.3.8 Error Interrupts . . . 191
12.6 Low-Power Modes . . . 192

12.6.1 Wait Mode . . . 192
12.6.2 Stop Mode . . . 192
12.7 SCI During Break Module Interrupts . . . 192
12.8 I/O Signals . . . 192
12.8.1 PTC6/SCTxD (Transmit Data) . . . 193
12.8.2 PTC7/SCRxD (Receive Data) . . . 193
12.9 I/O Registers . . . 193
12.9.1 IRSCI Control Register 1 . . . 194
12.9.2 IRSCI Control Register 2 . . . 196
12.9.3 IRSCI Control Register 3 . . . 197
12.9.4 IRSCI Status Register 1 . . . 199
12.9.5 IRSCI Status Register 2 . . . 201
12.9.6 IRSCI Data Register . . . 202
12.9.7 IRSCI Baud Rate Register . . . 202
12.9.8 IRSCI Infrared Control Register . . . 205

Chapter 13
Serial Peripheral Interface Module (SPI)

13.1 Introduction . . . 207
13.2 Features . . . 207
13.3 Pin Name Conventions and I/O Register Addresses . . . 207
13.4 Functional Description . . . 208
13.4.1 Master Mode . . . 208
13.4.2 Slave Mode . . . 210
13.5 Transmission Formats . . . 210
13.5.1 Clock Phase and Polarity Controls . . . 210
13.5.2 Transmission Format When CPHA = 0 . . . 211
13.5.3 Transmission Format When CPHA = 1 . . . 212
13.5.4 Transmission Initiation Latency . . . 212
13.6 Queuing Transmission Data . . . 213
13.7 Error Conditions . . . 214
13.7.1 Overflow Error . . . 215
13.7.2 Mode Fault Error . . . 216
13.8 Interrupts . . . 217
13.9 Resetting the SPI . . . 218
13.10 Low-Power Modes . . . 219
13.10.1 Wait Mode . . . 219
13.10.2 Stop Mode . . . 219
13.11 SPI During Break Interrupts . . . 219
13.12 I/O Signals . . . 220
13.12.1 MISO (Master In/Slave Out) . . . 220
13.12.2 MOSI (Master Out/Slave In) . . . 220
13.12.3 SPSCK (Serial Clock) . . . 220
13.12.4 $\overline{SS}$ (Slave Select) . . . 221
13.12.5 CGND (Clock Ground) . . . 222
13.13 I/O Registers . . . 222

13.13.1 SPI Control Register 222
13.13.2 SPI Status and Control Register 224
13.13.3 SPI Data Register 226

Chapter 14
Multi-Master IIC Interface (MMIIC)

14.1 Introduction 227
14.2 Features 227
14.3 I/O Pins 228
14.4 Multi-Master IIC System Configuration 228
14.5 Multi-Master IIC Bus Protocol 229
14.5.1 START Signal 229
14.5.2 Slave Address Transmission 229
14.5.3 Data Transfer 230
14.5.4 Repeated START Signal 230
14.5.5 STOP Signal 230
14.5.6 Arbitration Procedure 230
14.5.7 Clock Synchronization 230
14.5.8 Handshaking 231
14.5.9 Packet Error Code 231
14.6 MMIIC I/O Registers 231
14.6.1 MMIIC Address Register (MMADR) 232
14.6.2 MMIIC Control Register 1 (MMCR1) 233
14.6.3 MMIIC Control Register 2 (MMCR2) 234
14.6.4 MMIIC Status Register (MMSR) 235
14.6.5 MMIIC Data Transmit Register (MMDTR) 236
14.6.6 MMIIC Data Receive Register (MMDRR) 237
14.6.7 MMIIC CRC Data Register (MMCRCDR) 238
14.6.8 MMIIC Frequency Divider Register (MMFDR) 238
14.7 Program Algorithm 239
14.7.1 Data Sequence 240
14.8 SMBus Protocols with PEC and without PEC 241
14.8.1 Quick Command 241
14.8.2 Send Byte 241
14.8.3 Receive Byte 241
14.8.4 Write Byte/Word 242
14.8.5 Read Byte/Word 242
14.8.6 Process Call 243
14.8.7 Block Read/Write 243
14.9 SMBus Protocol Implementation 244

Chapter 15
Analog-to-Digital Converter (ADC)

15.1 Introduction 245
15.2 Features 245
15.3 Functional Description 246
15.3.1 ADC Port I/O Pins 246

15.3.2 Voltage Conversion . . . 246
15.3.3 Conversion Time . . . 247
15.3.4 Continuous Conversion . . . 248
15.3.5 Auto-Scan Mode . . . 248
15.3.6 Result Justification . . . 249
15.3.7 Data Register Interlocking . . . 249
15.3.8 Monotonicity . . . 249
15.4 Interrupts . . . 249
15.5 Low-Power Modes . . . 249
15.5.1 Wait Mode . . . 250
15.5.2 Stop Mode . . . 250
15.6 I/O Signals . . . 250
15.6.1 ADC Voltage In (V_{ADIN}) . . . 250
15.6.2 ADC Analog Power Pin (V_{DDA}) . . . 250
15.6.3 ADC Analog Ground Pin (V_{SSA}) . . . 250
15.6.4 ADC Voltage Reference High Pin (V_{REFH}) . . . 250
15.6.5 ADC Voltage Reference Low Pin (V_{REFL}) . . . 250
15.7 I/O Registers . . . 251
15.7.1 ADC Status and Control Register . . . 251
15.7.2 ADC Clock Control Register . . . 252
15.7.3 ADC Data Register 0 (ADRH0 and ADRL0) . . . 253
15.7.4 ADC Auto-Scan Mode Data Registers (ADRL1–ADRL3) . . . 255
15.7.5 ADC Auto-Scan Control Register (ADASCR) . . . 255

Chapter 16
Input/Output (I/O) Ports

16.1 Introduction . . . 257
16.2 Port A . . . 260
16.2.1 Port A Data Register (PTA) . . . 260
16.2.2 Data Direction Register (DDRA) . . . 260
16.2.3 Port-A LED Control Register (LEDA) . . . 262
16.3 Port B . . . 262
16.3.1 Port B Data Register (PTB) . . . 262
16.3.2 Data Direction Register B (DDRB) . . . 263
16.4 Port C . . . 264
16.4.1 Port C Data Register (PTC) . . . 264
16.4.2 Data Direction Register C (DDRC) . . . 265
16.5 Port D . . . 266
16.5.1 Port D Data Register (PTD) . . . 266
16.5.2 Data Direction Register D (DDRD) . . . 267

Chapter 17
External Interrupt (IRQ)

17.1 Introduction . . . 269
17.2 Features . . . 269
17.3 Functional Description . . . 269
17.4 $\overline{IRQ1}$ and $\overline{IRQ2}$ Pins . . . 271

17.5 IRQ Module During Break Interrupts 272
17.6 IRQ Registers 272
17.6.1 IRQ1 Status and Control Register 272
17.6.2 IRQ2 Status and Control Register 273

Chapter 18
Keyboard Interrupt Module (KBI)

18.1 Introduction 275
18.2 Features 275
18.3 I/O Pins 275
18.4 Functional Description 276
18.4.1 Keyboard Initialization 277
18.5 Keyboard Interrupt Registers 277
18.5.1 Keyboard Status and Control Register 277
18.5.2 Keyboard Interrupt Enable Register 278
18.6 Low-Power Modes 279
18.6.1 Wait Mode 279
18.6.2 Stop Mode 279
18.7 Keyboard Module During Break Interrupts 279

Chapter 19
Computer Operating Properly (COP)

19.1 Introduction 281
19.2 Functional Description 281
19.3 I/O Signals 282
19.3.1 ICLK 282
19.3.2 STOP Instruction 282
19.3.3 COPCTL Write 282
19.3.4 Power-On Reset 282
19.3.5 Internal Reset 282
19.3.6 Reset Vector Fetch 282
19.3.7 COPD (COP Disable) 282
19.3.8 COPRS (COP Rate Select) 283
19.4 COP Control Register 283
19.5 Interrupts 283
19.6 Monitor Mode 283
19.7 Low-Power Modes 283
19.7.1 Wait Mode 284
19.7.2 Stop Mode 284
19.8 COP Module During Break Mode 284

Chapter 20
Low-Voltage Inhibit (LVI)

20.1 Introduction 285
20.2 Features 285
20.3 Functional Description 285

20.3.1 Low V_{DD} Detector 286
20.3.2 Low V_{REG} Detector 286
20.3.3 Polled LVI Operation 286
20.3.4 Forced Reset Operation 287
20.3.5 Voltage Hysteresis Protection 287
20.4 LVI Status Register 287
20.5 LVI Interrupts 287
20.6 Low-Power Modes 288
20.6.1 Wait Mode 288
20.6.2 Stop Mode 288

Chapter 21
Break Module (BRK)

21.1 Introduction 289
21.2 Features 289
21.3 Functional Description 289
21.3.1 Flag Protection During Break Interrupts 290
21.3.2 CPU During Break Interrupts 290
21.3.3 TIMI and TIM2 During Break Interrupts 291
21.3.4 COP During Break Interrupts 291
21.4 Low-Power Modes 291
21.4.1 Wait Mode 291
21.5 Break Module Registers 291
21.5.1 Break Status and Control Register 291
21.5.2 Break Address Registers 292
21.5.3 SIM Break Status Register 292
21.5.4 SIM Break Flag Control Register 293

Chapter 22
Electrical Specifications

22.1 Introduction 295
22.2 Absolute Maximum Ratings 295
22.3 Functional Operating Range 296
22.4 Thermal Characteristics 296
22.5 5V DC Electrical Characteristics 297
22.6 5V Control Timing 298
22.7 5V Oscillator Characteristics 299
22.8 5V ADC Electrical Characteristics 300
22.9 MMIIC Electrical Characteristics 301
22.10 CGM Electrical Specification 303
22.11 5V SPI Characteristics 304
22.12 Memory Characteristics 307

Chapter 23
Mechanical Specifications

23.1 Introduction 309
23.2 48-Pin Low-Profile Quad Flat Pack (LQFP) 310
23.3 44-Pin Quad Flat Pack (QFP) 311
23.4 42-Pin Shrink Dual In-Line Package (SDIP) 312

Chapter 24
Ordering Information

24.1 Introduction 313
24.2 MC Order Numbers 313

Chapter 1
General Description

1.1 Introduction

The MC68HC908AP64A 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.

Table 1-1. Summary of Device Variations

Device	RAM Size (bytes)	FLASH Memory Size (bytes)
MC68HC908AP64A	2,048	62,368
MC68HC908AP32A	2,048	32,768
MC68HC908AP16A	1,024	16,384
MC68HC908AP8A	1,024	8,192

1.2 Features

Features of the MC68HC908AP64A include the following:

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- Maximum internal bus frequency:
  - 8-MHz at 5V operating voltage
- Clock input options:
  - RC-oscillator
  - 1- to 8-MHz crystal-oscillator with 32MHz internal PLL
- User program FLASH memory with security[1] feature
  - 62,368 bytes for MC68HC908AP64A
  - 32,768 bytes for MC68HC908AP32A
  - 16,384 bytes for MC68HC908AP16A
  - 8,192 bytes for MC68HC908AP8A
- On-chip RAM
  - 2,048 bytes for MC68HC908AP64A and MC68HC908AP32A
  - 1,024 bytes for MC68HC908AP16A and MC68HC908AP8A
- Two 16-bit, 2-channel timer interface modules (TIM1 and TIM2) with selectable input capture, output compare, and PWM capability on each channel

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

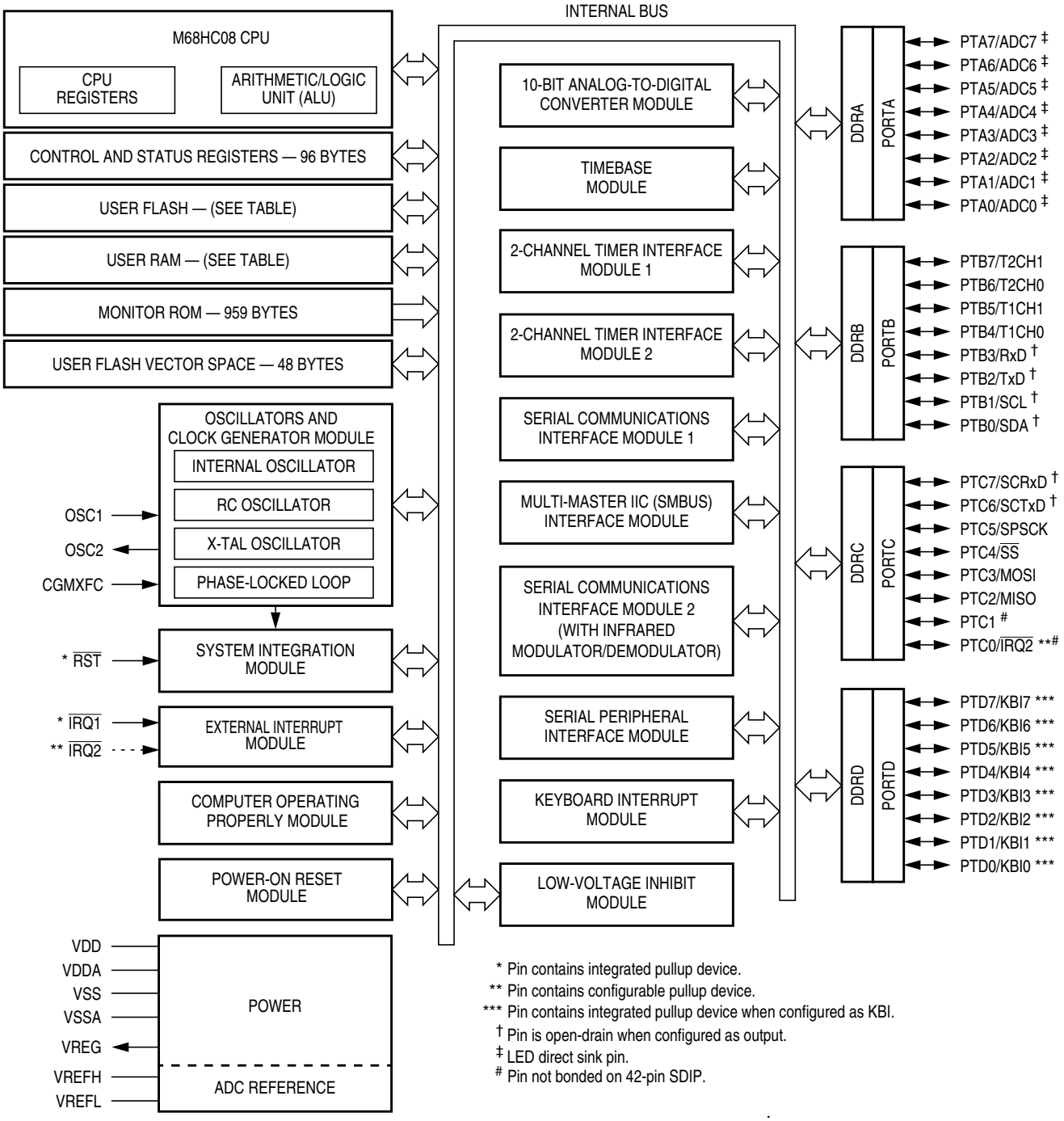
- Timebase module
- Serial communications interface module 1 (SCI)
- Serial communications interface module 2 (SCI) with infrared (IR) encoder/decoder
- Serial peripheral interface module (SPI)
- System management bus (SMBus), version 1.0/1.1 (multi-master IIC bus)
- 8-channel, 10-bit analog-to-digital converter (ADC)
- $\overline{\text{IRQ1}}$ external interrupt pin with integrated pullup
- $\overline{\text{IRQ2}}$ external interrupt pin with programmable pullup
- 8-bit keyboard wakeup port with integrated pullup
- 32 general-purpose input/output (I/O) pins:
  - 31 shared-function I/O pins
  - 8 LED drivers (sink)
  - 6 × 25mA open-drain I/O with pullup
- Low-power design (fully static with stop and wait modes)
- Master reset pin (with integrated pullup) and power-on reset
- System protection features
  - Optional computer operating properly (COP) reset, driven by internal RC oscillator
  - Low-voltage detection with optional reset or interrupt
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- 48-pin low quad flat pack (LQFP), 44-pin quad flat pack (QFP), and 42-pin shrink dual-in-line package (SDIP)
- Specific features of the MC68HC908AP64A in 42-pin SDIP are:
  - 30 general-purpose I/Os only
  - External interrupt on $\overline{\text{IRQ1}}$ only

Features of the CPU08 include the following:

- Enhanced HC05 programming model
- Extensive loop control functions
- 16 addressing modes (eight more than the HC05)
- 16-bit Index register and stack pointer
- Memory-to-memory data transfers
- Fast 8 × 8 multiply instruction
- Fast 16/8 divide instruction
- Binary-coded decimal (BCD) instructions
- Optimization for controller applications
- Efficient C language support

1.3 MCU Block Diagram

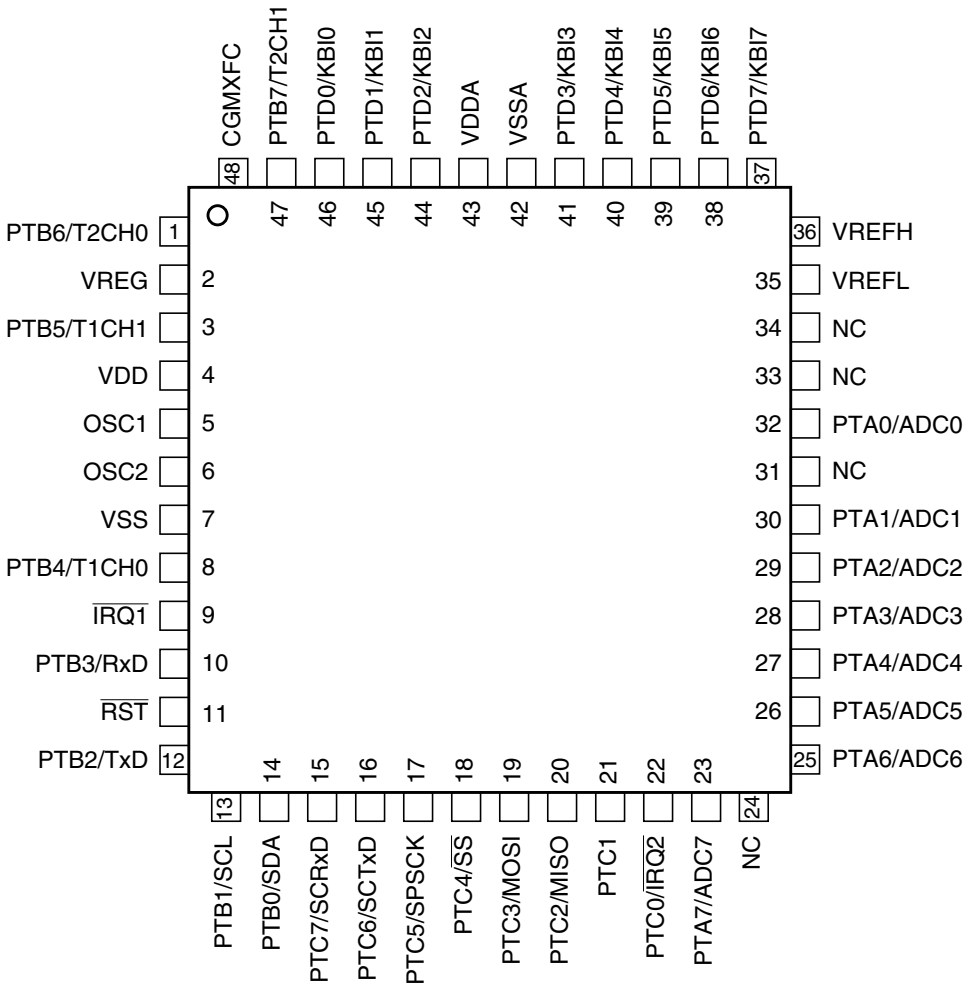
Figure 1-1 shows the structure of the MC68HC908AP64A.



DEVICE	USER RAM (bytes)	USER FLASH (bytes)
MC68HC908AP64A	2,048	62,368
MC68HC908AP32A	2,048	32,768
MC68HC908AP16A	1,024	16,384
MC68HC908AP8A	1,024	8,192

Figure 1-1. MC68HC908AP64A Block Diagram

1.4 Pin Assignment



NC: No connection

Figure 1-2. 48-Pin LQFP Pin Assignments

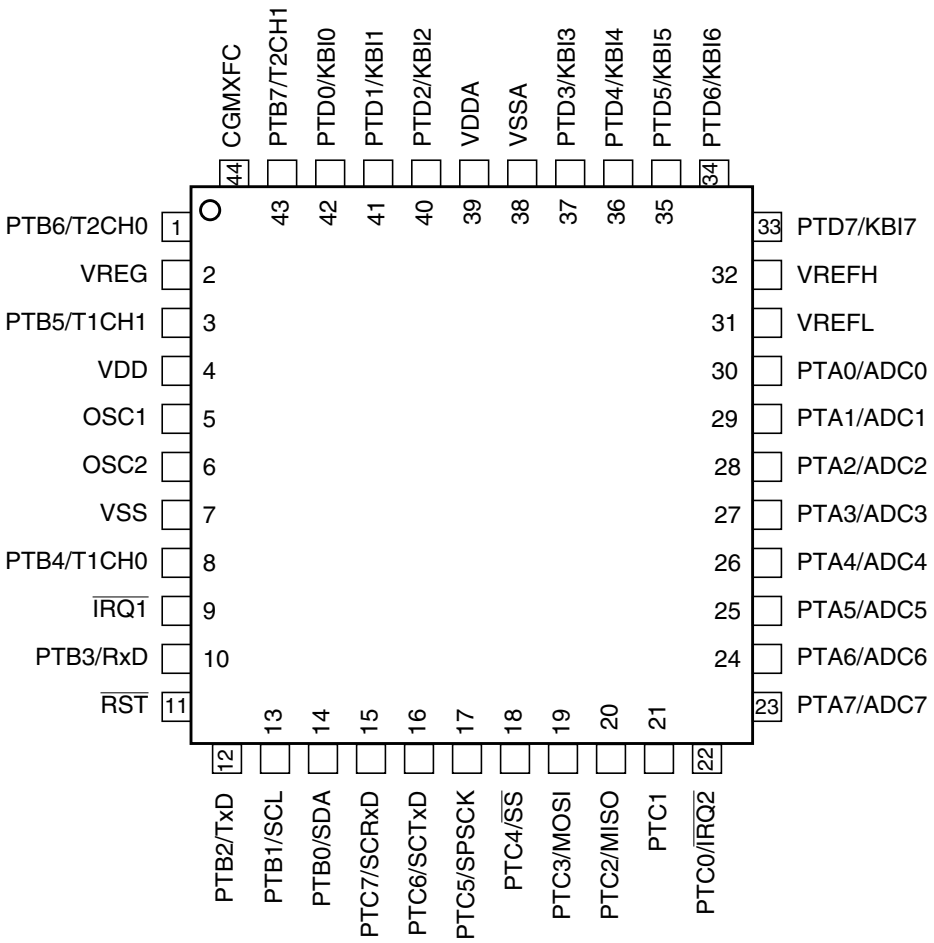
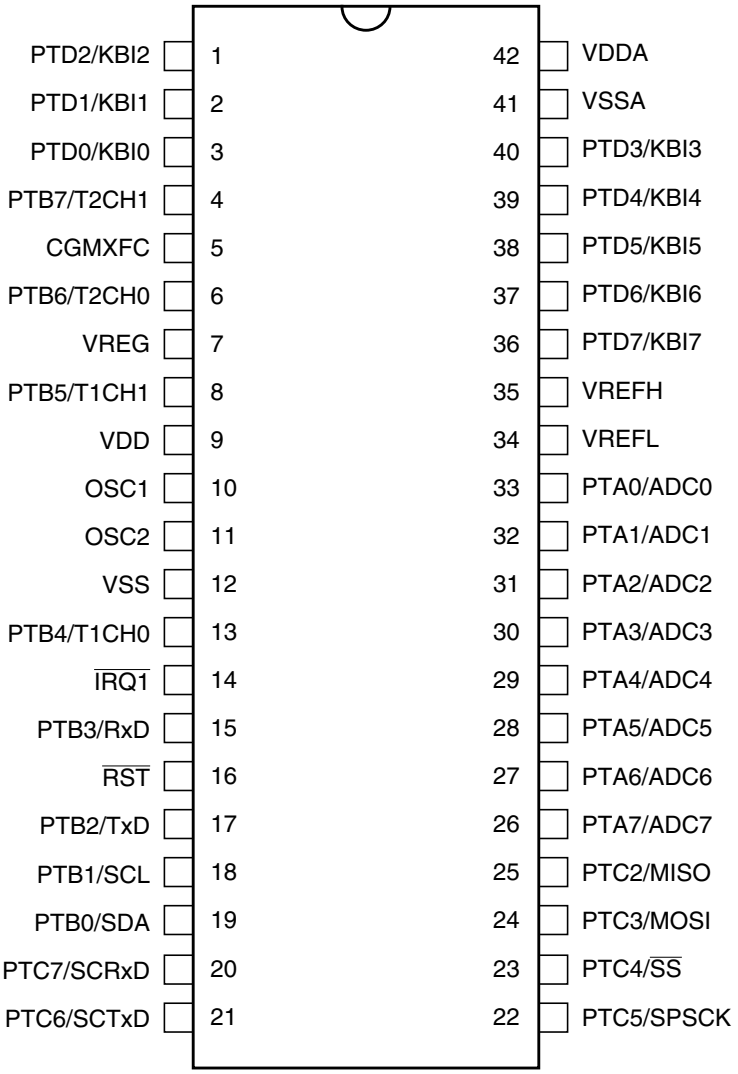


Figure 1-3. 44-Pin QFP Pin Assignments



Pins not available on 42-pin package	Internal connection
PTC0/IRQ2	Unconnected
PTC1	Unconnected

Figure 1-4. 42-Pin SDIP Pin Assignment

1.5 Pin Functions

Description of the pin functions are provided in Table 1-2.

Table 1-2. Pin Functions

PIN NAME	PIN DESCRIPTION	IN/OUT	VOLTAGE LEVEL
V_{DD}	Power supply.	In	4.5 to 5.5
V_{SS}	Power supply ground.	Out	0 V
V_{DDA}	Power supply for analog circuits.	In	V_{DD}
V_{SSA}	Power supply ground for analog circuits.	Out	V_{SS}
V_{REFH}	ADC input reference high.	In	V_{DDA}
V_{REFL}	ADC input reference low.	Out	V_{SSA}
V_{REG}	Internal (2.5 V) regulator output. Require external capacitors for decoupling.	Out	2.5 V(1)
$\overline{RST}$	Reset input, active low; with internal pullup and schmitt trigger input.	In	V_{DD}
$\overline{IRQ1}$	External IRQ1 pin; with internal pullup and schmitt trigger input.	In	V_{DD}
	Used for mode entry selection.	In	V_{DD} to V_{TST}
OSC1	Crystal or RC oscillator input.	In	V_{REG}
OSC2	Crystal OSC option: crystal oscillator output; inverted OSC1.	Out	V_{REG}
	RC OSC option: bus clock output.	Out	V_{REG}
	Internal OSC option: bus clock output.	Out	V_{REG}
CGMXFC	CGM external filter capacitor connection.	In/Out	Analog
PTA0/ADC0 : PTA7/ADC7	8-bit general purpose I/O port.	In/Out	V_{DD}
	Pins as ADC inputs, ADC0–ADC7.	In	V_{REFH}
	Each pin has high current sink for LED.	Out	V_{DD}

Table 1-2. Pin Functions

PIN NAME	PIN DESCRIPTION	IN/OUT	VOLTAGE LEVEL
PTB0/SDA PTB1/SCL PTB2/TxD PTB3/RxD PTB4/T1CH0 PTB5/T1CH1 PTB6/T2CH0 PTB7/T2CH1	8-bit general purpose I/O port; PTB0–PTB3 are open drain when configured as output. PTB4–PTB7 have schmitt trigger inputs.	In/Out	V_{DD}
	PTB0 as SDA of MMIIC.	In/Out	V_{DD}
	PTB1 as SCL of MMIIC.	In/Out	V_{DD}
	PTB2 as TxD of SCI; open drain output.	Out	V_{DD}
	PTB3 as RxD of SCI.	In	V_{DD}
	PTB4 as T1CH0 of TIM1.	In/Out	V_{DD}
	PTB5 as T1CH1 of TIM1.	In/Out	V_{DD}
	PTB6 as T2CH0 of TIM2.	In/Out	V_{DD}
	PTB7 as T2CH1 of TIM2.	In/Out	V_{DD}
PTC0/$\overline{IRQ2}$ PTC1 PTC2/MISO PTC3/MOSI PTC4/$\overline{SS}$ PTC5/SPSCK PTC6/SCTxD PTC7/SCRxD	8-bit general purpose I/O port; PTC6 and PTC7 are open drain when configured as output.	In/Out	V_{DD}
	PTC0 is shared with $\overline{IRQ2}$ and has schmitt trigger input.	In	V_{DD}
	PTC2 as MISO of SPI.	In	V_{DD}
	PTC3 as MOSI of SPI.	Out	V_{DD}
	PTC4 as $\overline{SS}$ of SPI.	In	V_{DD}
	PTC5 as SPSCK of SPI.	In/Out	V_{DD}
	PTC6 as SCTxD of IRSCI; open drain output.	Out	V_{DD}
	PTC7 as SCRxD of IRSCI.	In	V_{DD}
PTD0/KBI0 : PTD7/KBI7	8-bit general purpose I/O port with schmitt trigger inputs.	In/Out	V_{DD}
	Pins as keyboard interrupts (with pullup), KBI0–KBI7.	In	V_{DD}

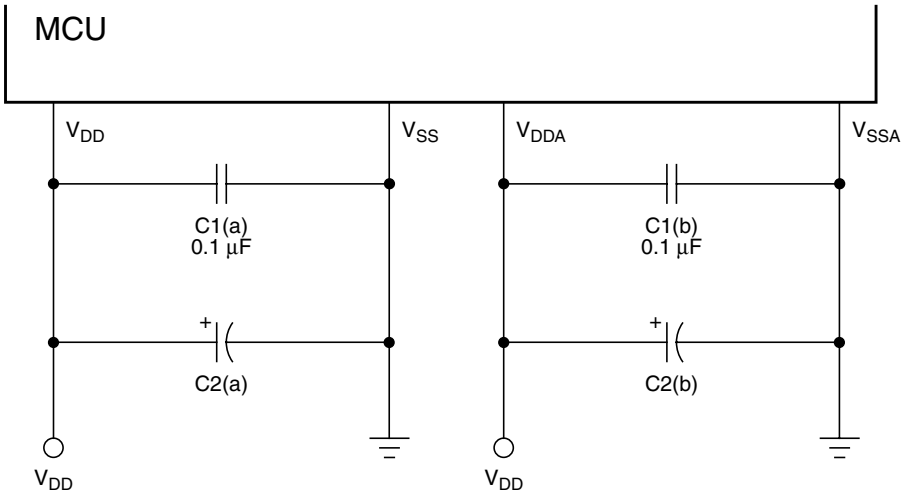
1. See Chapter 22 Electrical Specifications for V_{REG} tolerance.

1.6 Power Supply Bypassing (VDD, VDDA, VSS, VSSA)

V_{DD} and V_{SS} are the power supply and ground pins, the MCU operates from a single power supply together with an on chip voltage regulator.

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 bypass capacitors as close to the MCU power pins as possible. Use high-frequency-response ceramic capacitor for C_{BYPASS}, C_{BULK} are optional bulk current bypass capacitors for use in applications that require the port pins to source high current level.

V_{DDA} and V_{SSA} are the power supply and ground pins for the analog circuits of the MCU. These pins should be decoupled as per the digital power supply pins.



NOTE: Component values shown represent typical applications.

Figure 1-5. Power Supply Bypassing

1.7 Regulator Power Supply Configuration (VREG)

V_{REG} is the output from the on-chip regulator. All internal logics, except for the I/O pads, are powered by V_{REG} output. V_{REG} requires an external ceramic bypass capacitor of 100 nF as Figure 1-6 shows. Place the bypass capacitor as close to the V_{REG} pin as possible.

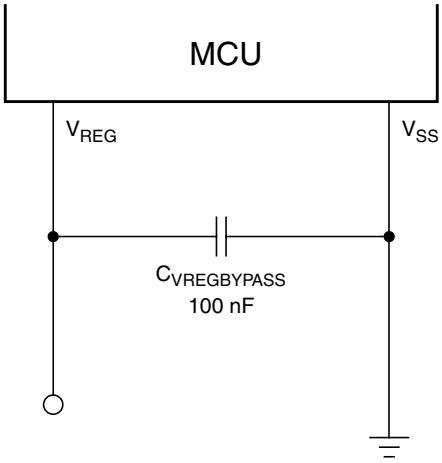


Figure 1-6. Regulator Power Supply Bypassing

Chapter 2
Memory

2.1 Introduction

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

- 62,368 bytes of user FLASH — MC68HC908AP64A
  32,768 bytes of user FLASH — MC68HC908AP32A
  16,384 bytes of user FLASH — MC68HC908AP16A
  8,192 bytes of user FLASH — MC68HC908AP8A
- 2,048 bytes of RAM — MC68HC908AP64A and MC68HC908AP32A
  1,024 bytes of RAM — MC68HC908AP16A and MC68HC908AP8A
- 48 bytes of user-defined vectors
- 959 bytes of monitor ROM

2.2 Input/Output (I/O) Section

Most of the control, status, and data registers are in the zero page area of $0000–$005F. Additional I/O registers have these addresses:

- $FE00; SIM break status register, SBSR
- $FE01; SIM reset status register, SRSR
- $FE02; Reserved
- $FE03; SIM break flag control register, SBFCR
- $FE04; interrupt status register 1, INT1
- $FE05; interrupt status register 2, INT2
- $FE06; interrupt status register 3, INT3
- $FE07; Reserved
- $FE08; FLASH control register, FLCR
- $FE09; 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
- $FE0F; LVI Status register, LVISR
- $FFCF; Mask option register, MOR (FLASH register)
- $FFFF; COP control register, COPCTL

2.3 Monitor ROM

The 959 bytes at addresses $FC00–$FDFF and $FE10–$FFCE are reserved ROM addresses that contain the instructions for the monitor functions. (See Chapter 8 Monitor Mode (MON).)

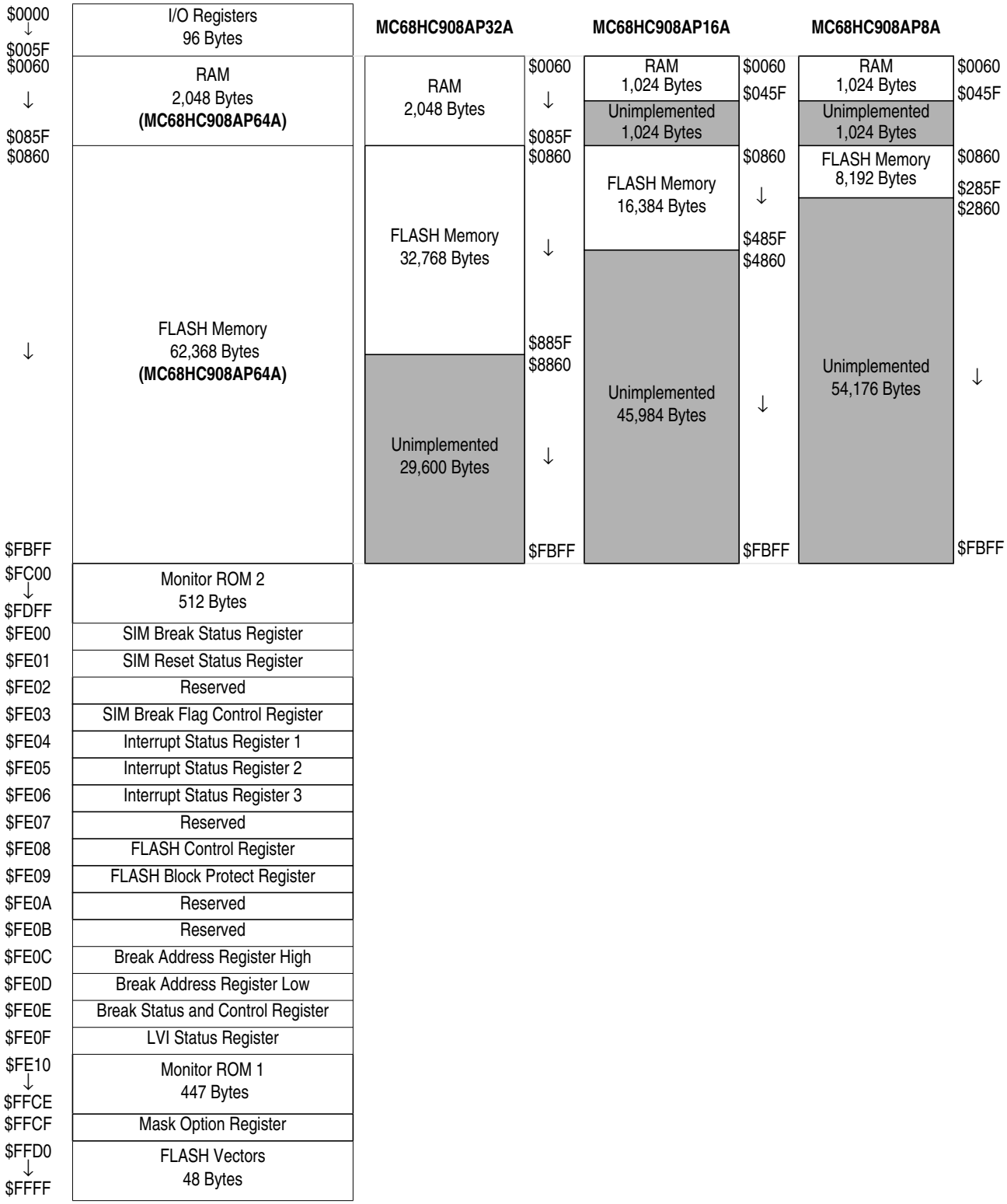


Figure 2-1. Memory Map

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC)	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0
$0007	Data Direction Register D (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$0008	Unimplemented	Read: Write:								
		Reset:								
$0009	Unimplemented	Read: Write:								
		Reset:								
$000A	Unimplemented	Read: Write:								
		Reset:								
$000B	Unimplemented	Read: Write:								
		Reset:								
$000C	Port-A LED Control Register (LEDA)	Read: Write:	LEDA7	LEDA6	LEDA5	LEDA4	LEDA3	LEDA2	LEDA1	LEDA0
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$000D	Unimplemented	Read: Write:								
		Reset:								
$000E	Unimplemented	Read: Write:								
		Reset:								
$000F	Unimplemented	Read: Write:								
		Reset:								
$0010	SPI Control Register (SPCR)	Read: Write:	SPRIE	R	SPMSTR	CPOL	CPHA	SPWOM	SPE	SPTIE
		Reset:	0	0	1	0	1	0	0	0
$0011	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
$0012	SPI Data Register (SPDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0013	SCI Control Register 1 (SCC1)	Read: Write:	LOOPS	ENSCI	TXINV	M	WAKE	ILTY	PEN	PTY
		Reset:	0	0	0	0	0	0	0	0
$0014	SCI Control Register 2 (SCC2)	Read: Write:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Reset:	0	0	0	0	0	0	0	0
$0015	SCI Control Register 3 (SCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$0016	SCI Status Register 1 (SCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0017	SCI Status Register 2 (SCS2)	Read:	0	0	0	0	0	0	BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0018	SCI Data Register (SCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0019	SCI Baud Rate Register (SCBR)	Read:	0	0	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

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

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	Keyboard Status and Control Register (KBSCR)	Read:	0	0	0	0	KEYF	0	IMASK	MODE
		Write:						ACK		
		Reset:	0	0	0	0	0	0	0	0
$001B	Keyboard Interrupt Enable Register (KBIER)	Read: Write:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Reset:	0	0	0	0	0	0	0	0
$001C	IRQ2 Status and Control Register (INTSCR2)	Read:	0	PUC0ENB	0	0	IRQ2F	0	IMASK2	MODE2
		Write:						ACK2		
		Reset:	0	0	0	0	0	0	0	0
$001D	Configuration Register 2 (CONFIG2)†	Read:	STOP_ ICLKDIS	STOP_ RCLKEN	STOP_ XCLKEN	OSCCLK1	OSCCLK0	0	0	SCIBDSRC
		Write:								
		Reset:	0	0	0	0	0	0	0	0

† One-time writable register after each reset.

$001E	IRQ1 Status and Control Register (INTSCR1)	Read:	0	0	0	0	IRQ1F	0	IMASK1	MODE1
		Write:						ACK1		
		Reset:	0	0	0	0	0	0	0	0
$001F	Configuration Register 1 (CONFIG1)†	Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVIREGD	SSREC	STOP	COPD
		Reset:	0	0	0	0	0	0	0	0

† One-time writable register after each reset.

$0020	Timer 1 Status and Control Register (T1SC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$0021	Timer 1 Counter Register High (T1CNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0022	Timer 1 Counter Register Low (T1CNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0023	Timer 1 Counter Modulo Register High (T1MODH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	1	1	1	1	1	1	1	1
$0024	Timer 1 Counter Modulo Register Low (T1MODL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$0025	Timer 1 Channel 0 Status and Control Register (T1SC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0026	Timer 1 Channel 0 Register High (T1CH0H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0027	Timer 1 Channel 0 Register Low (T1CH0L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0028	Timer 1 Channel 1 Status and Control Register (T1SC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0029	Timer 1 Channel 1 Register High (T1CH1H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$002A	Timer 1 Channel 1 Register Low (T1CH1L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$002B	Timer 2 Status and Control Register (T2SC)	Read:	TOF	TOIE	TSTOP	0	0	PS2	PS1	PS0
		Write:	0			TRST				
		Reset:	0	0	1	0	0	0	0	0
$002C	Timer 2 Counter Register High (T2CNTH)	Read:	Bit 15	14	13	12	11	10	9	Bit 8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002D	Timer 2 Counter Register Low (T2CNTL)	Read:	Bit 7	6	5	4	3	2	1	Bit 0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$002E	Timer 2 Counter Modulo Register High (T2MODH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	1	1	1	1	1	1	1	1
$002F	Timer 2 Counter Modulo Register Low (T2MODL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	1	1	1	1	1	1	1	1
$0030	Timer 2 Channel 0 Status and Control Register (T2SC0)	Read:	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	TOV0	CH0MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0031	Timer 2 Channel 0 Register High (T2CH0H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0032	Timer 2 Channel 0 Register Low (T2CH0L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0033	Timer 2 Channel 1 Status and Control Register (T2SC1)	Read:	CH1F	CH1IE	0	MS1A	ELS1B	ELS1A	TOV1	CH1MAX
		Write:	0							
		Reset:	0	0	0	0	0	0	0	0
$0034	Timer 2 Channel 1 Register High (T2CH1H)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	Indeterminate after reset							
$0035	Timer 2 Channel 1 Register Low (T2CH1L)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	Indeterminate after reset							
$0036	PLL Control Register (PCTL)	Read:	PLLIE	PLLF	PLLON	BCS	PRE1	PRE0	VPR1	VPR0
		Write:								
		Reset:	0	0	1	0	0	0	0	0
$0037	PLL Bandwidth Control Register (PBWC)	Read:	AUTO	LOCK	$\overline{ACQ}$	0	0	0	0	R
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0038	PLL Multiplier Select Register High (PMSH)	Read:	0	0	0	0	MUL11	MUL10	MUL9	MUL8
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0039	PLL Multiplier Select Register Low (PMSL)	Read: Write:	MUL7	MUL6	MUL5	MUL4	MUL3	MUL2	MUL1	MUL0
		Reset:	0	1	0	0	0	0	0	0
$003A	PLL VCO Range Select Register (PMRS)	Read: Write:	VRS7	VRS6	VRS5	VRS4	VRS3	VRS2	VRS1	VRS0
		Reset:	0	1	0	0	0	0	0	0
$003B	PLL Reference Divider Select Register (PMDS)	Read:	0	0	0	0	RDS3	RDS2	RDS1	RDS0
		Write:								
		Reset:	0	0	0	0	0	0	0	1
$003C	Unimplemented	Read: Write:								
		Reset:								
$003D	Unimplemented	Read: Write:								
		Reset:								
$003E	Unimplemented	Read: Write:								
		Reset:								
$003F	Unimplemented	Read: Write:								
		Reset:								

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0040	IRSCI Control Register 1 (IRSCC1)	Read:	LOOPS	ENSCI	0	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0041	IRSCI Control Register 2 (IRSCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0042	IRSCI Control Register 3 (IRSCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$0043	IRSCI Status Register 1 (IRSCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0044	IRSCI Status Register 2 (IRSCS2)	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0045	IRSCI Data Register (IRSCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0046	IRSCI Baud Rate Register (IRSCBR)	Read:	CKS	0	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0047	IRSCI Infrared Control Register (IRSCIRCR)	Read:	R	0	0	0	R	TNP1	TNP0	IREN
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0048	MMIIC Address Register (MMADR)	Read:	MMAD7	MMAD6	MMAD5	MMAD4	MMAD3	MMAD2	MMAD1	MMEXTAD
		Write:								
		Reset:	1	0	1	0	0	0	0	0
$0049	MMIIC Control Register 1 (MMCR1)	Read:	MMEN	MMIEN	0	0	MMTXAK	REPSEN	MMCRCBYTE	0
		Write:			MMCLRBB					
		Reset:	0	0	0	0	0	0	0	0
$004A	MMIIC Control Register 2 (MMCR2)	Read:	MMALIF	MMNAKIF	MMBB	MMAST	MMRW	0	0	MMCRCEF
		Write:	0	0						
		Reset:	0	0	0	0	0	0	0	Unaffected
$004B	MMIIC Status Register (MMSR)	Read:	MMRXIF	MMTXIF	MMATCH	MMSRW	MMRXAK	MMCRCBF	MMTXBE	MMRXBF
		Write:	0	0						
		Reset:	0	0	0	0	1	0	1	0
$004C	MMIIC Data Transmit Register (MMDTR)	Read:	MMTD7	MMTD6	MMTD5	MMTD4	MMTD3	MMTD2	MMTD1	MMTD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$004D	MMIIC Data Receive Register (MMDRR)	Read:	MMRD7	MMRD6	MMRD5	MMRD4	MMRD3	MMRD2	MMRD1	MMRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004E	MMIIC CRC Data Register (MMCRDR)	Read:	MMCRCD7	MMCRCD6	MMCRCD5	MMCRCD4	MMCRCD3	MMCRCD2	MMCRCD1	MMCRCD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004F	MMIIC Frequency Divider Register (MMFDR)	Read:	0	0	0	0	0	MMBR2	MMBR1	MMBR0
		Write:								
		Reset:	0	0	0	0	0	1	0	0
$0050	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$0051	Timebase Control Register (TBCR)	Read:	TBIF	TBR2	TBR1	TBR0	0	TBIE	TBON	R
		Write:					TACK			
		Reset:	0	0	0	0	0	0	0	0
$0052	Unimplemented	Read:								
		Write:								
		Reset:								
$0053	Unimplemented	Read:								
		Write:								
		Reset:								
$0054	Unimplemented	Read:								
		Write:								
		Reset:								
$0055	Unimplemented	Read:								
		Write:								
		Reset:								
$0056	Unimplemented	Read:								
		Write:								
		Reset:								
$0057	ADC Status and Control Register (ADSCR)	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$0058	ADC Clock Control Register (ADICLK)	Read:	ADIV2	ADIV1	ADIV0	ADICLK	MODE1	MODE0	0	0
		Write:								R
		Reset:	0	0	0	0	0	0	0	0
$0059	ADC Data Register High 0 (ADRH0)	Read:	ADx	ADx	ADx	ADx	ADx	ADx	ADx	ADx
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 7 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$005A	ADC Data Register Low 0 (ADRL0)	Read:	ADx	ADx	ADx	ADx	ADx	ADx	ADx	ADx
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005B	ADC Data Register Low 1 (ADRL1)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005C	ADC Data Register Low 2 (ADRL2)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005D	ADC Data Register Low 3 (ADRL3)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005E	ADC Auto-scan Control Register (ADASCR)	Read:								
		Write:						AUTO1	AUTO0	ASCAN
		Reset:	0	0	0	0	0	0	0	0
$005F	Unimplemented	Read:								
		Write:								
		Reset:								
$FE00	SIM Break Status Register (SBSR)	Read:	R	R	R	R	R	R	SBSW	R
		Write:							Note	
		Reset:							0	
$FE01	SIM Reset Status Register (SRSR)	Read:	POR	PIN	COP	ILOP	ILAD	MODRST	LVI	0
		Write:								
		Reset:	1	0	0	0	0	0	0	0
$FE02	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$FE03	SIM Break Flag Control Register (SBFCR)	Read:	BCFE	R	R	R	R	R	R	R
		Write:								
		Reset:	0							
$FE04	Interrupt Status Register 1 (INT1)	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE05	Interrupt Status Register 2 (INT2)	Read:	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

Note: Writing a logic 0 clears SBSW.

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

Figure 2-2. Control, Status, and Data Registers (Sheet 8 of 9)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE06	Interrupt Status Register 3 (INT3)	Read:	0	IF21	IF20	IF19	IF18	IF17	IF16	IF15
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$FE07	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$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:								
		Reset:								
$FE0B	Reserved	Read:	R	R	R	R	R	R	R	R
		Write:								
		Reset:								
$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)	Reset:	BRKE	BRKA	0	0	0	0	0	0
		Read:								
		Write:	0	0	0	0	0	0	0	0
$FE0F	LVI Status Register (LVISR)	Reset:	LVIOUT	0	0	0	0	0	0	0
		Read:								
		Write:	0	0	0	0	0	0	0	0
$FFCF	Mask Option Register (MOR)#	Read:	OSCSEL1	OSCSEL0	R	R	R	R	R	R
		Write:								
		Erased:	1	1	1	1	1	1	1	1
		Reset:	U	U	U	U	U	U	U	U
$FFFF	COP Control Register (COPCTL)	Read:	Low byte of reset vector							
		Write:	Writing clears COP counter (any value)							
		Reset:	Unaffected by reset							

MOR is a non-volatile FLASH register; write by programming.

U = Unaffected X = Indeterminate = Unimplemented R = Reserved

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

Table 2-1. Vector Addresses

Priority	INT Flag	Address	Vector
Lowest	—	$FFD0	Reserved
		$FFD1	Reserved
	IF21	$FFD2	TBM Vector (High)
		$FFD3	TBM Vector (Low)
	IF20	$FFD4	SCI2 (IRSCI) Transmit Vector (High)
		$FFD5	SCI2 (IRSCI) Transmit Vector (Low)
	IF19	$FFD6	SCI2 (IRSCI) Receive Vector (High)
		$FFD7	SCI2 (IRSCI) Receive Vector (Low)
	IF18	$FFD8	SCI2 (IRSCI) Error Vector (High)
		$FFD9	SCI2 (IRSCI) Error Vector (Low)
	IF17	$FFDA	SPI Transmit Vector (High)
		$FFDB	SPI Transmit Vector (Low)
	IF16	$FFDC	SPI Receive Vector (High)
		$FFDD	SPI Receive Vector (Low)
	IF15	$FFDE	ADC Conversion Complete Vector (High)
		$FFDF	ADC Conversion Complete Vector (Low)
	IF14	$FFE0	Keyboard Vector (High)
		$FFE1	Keyboard Vector (Low)
	IF13	$FFE2	SCI Transmit Vector (High)
		$FFE3	SCI Transmit Vector (Low)
	IF12	$FFE4	SCI Receive Vector (High)
		$FFE5	SCI Receive Vector (Low)
	IF11	$FFE6	SCI Error Vector (High)
		$FFE7	SCI Error Vector (Low)
	IF10	$FFE8	MMIIC Interrupt Vector (High)
		$FFE9	MMIIC Interrupt Vector (Low)
	IF9	$FFEA	TIM2 Overflow Vector (High)
		$FFEB	TIM2 Overflow Vector (Low)

Table 2-1. Vector Addresses (Continued)

Priority	INT Flag	Address	Vector
↓	IF8	$FFEC	TIM2 Channel 1 Vector (High)
		$FFED	TIM2 Channel 1 Vector (Low)
	IF7	$FFEE	TIM2 Channel 0 Vector (High)
		$FFEF	TIM2 Channel 0 Vector (Low)
	IF6	$FFF0	TIM1 Overflow Vector (High)
		$FFF1	TIM1 Overflow Vector (Low)
	IF5	$FFF2	TIM1 Channel 1 Vector (High)
		$FFF3	TIM1 Channel 1 Vector (Low)
	IF4	$FFF4	TIM1 Channel 0 Vector (High)
		$FFF5	TIM1 Channel 0 Vector (Low)
	IF3	$FFF6	PLL Vector (High)
		$FFF7	PLL Vector (Low)
	IF2	$FFF8	$\overline{IRQ2}$ Vector (High)
		$FFF9	$\overline{IRQ2}$ Vector (Low)
	IF1	$FFFA	$\overline{IRQ1}$ Vector (High)
		$FFFB	$\overline{IRQ1}$ 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 $085F (or $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, direct addressing mode instructions can access efficiently all page zero RAM locations. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

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

NOTE

For M6805 compatibility, the H register is not stacked.

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

NOTE

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

2.5 FLASH Memory

This sub-section describes the operation of the embedded FLASH memory. 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.

Device	FLASH Memory Size (Bytes)	Memory Address Range
MC68HC908AP64A	62,368	$0860–$FBFF
MC68HC908AP32A	32,768	$0860–$885F
MC68HC908AP16A	16,384	$0860–$485F
MC68HC908AP8A	8,192	$0860–$285F

2.5.1 Functional Description

The FLASH memory consists of an array of 62,368 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:

- $0860–$FBFF; user memory, 62,368 bytes
- $FFD0–$FFFF; user interrupt vectors, 48 bytes
- $FFCF; mask option register

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

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). (See ***NOTE*** below.)
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

Due to the relatively long mass erase time, user should take care in the code to prevent a COP reset from happening while the HVEN bit is set.

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, $XX80 or $XXC0. 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.

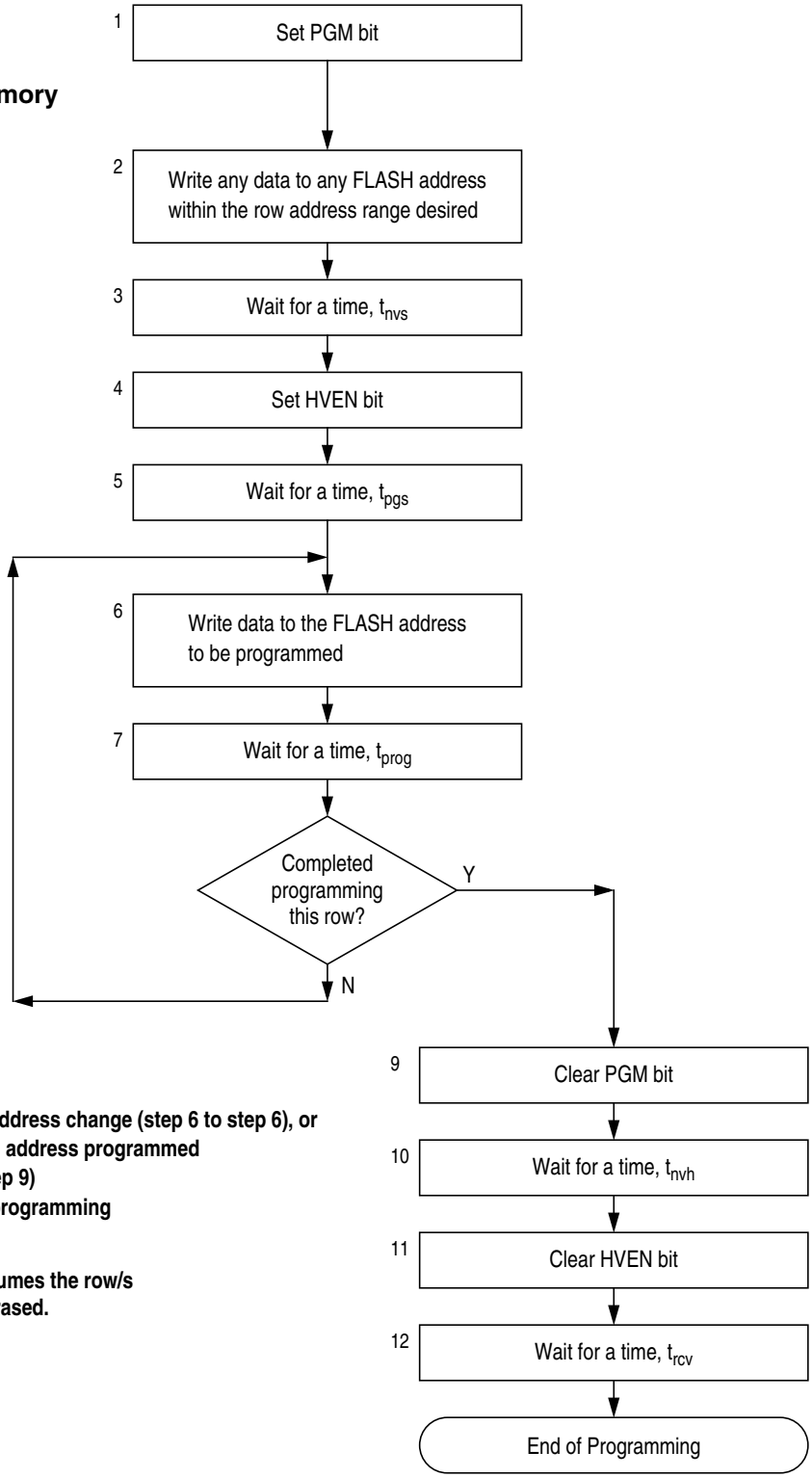
2.5.6 FLASH Protection

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 mask option register ($FFCF) and the 48 bytes of user interrupt vectors ($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.

16-bit memory address

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

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, X400, X0600, X800, XA00, XC00, or XE00 (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 to $09	The entire FLASH memory is protected.
$0A or $0B (**0000 101x**)	$0A00 to $FFFF
$0C or $0D (**0000 110x**)	$0C00 to $FFFF
and so on...	
$FA or $FB (**1111 1101x**)	$FA00 to $FFFF
$FC or $FD or $FE	$FFCF to $FFFF
$FF	The entire FLASH memory is **NOT** protected.(1)

1. Except for the mask option register ($FFCF) and the 48-byte user vectors ($FFD0–$FFFF). These FLASH locations are always protected.

Chapter 3
Configuration & Mask Option Registers (CONFIG & MOR)

3.1 Introduction

This section describes the configuration registers, CONFIG1 and CONFIG2; and the mask option register, MOR.

The configuration registers enable or disable these options:

- Computer operating properly module (COP)
- COP timeout period (262,128 or 8176 ICLK cycles)
- Low-voltage inhibit (LVI) on V_{DD}
- LVI on V_{REG}
- LVI module reset
- LVI module in stop mode
- STOP instruction
- Stop mode recovery time (32 ICLK or 4096 ICLK cycles)
- Oscillator (internal, RC, and crystal) during stop mode
- Serial communications interface clock source (CGMXCLK or f_{BUS})

The mask option register selects one of the following oscillator options:

- Internal oscillator
- RC oscillator
- Crystal oscillator

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001D	Configuration Register 2 (CONFIG2)†	Read:	STOP_ ICLKDIS	STOP_ RCLKEN	STOP_ XCLKEN	OSCCLK1	OSCCLK0	0	0	SCIBDSRC
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$001F	Configuration Register 1 (CONFIG1)†	Read: Write:	COPRS	LVISTOP	LVIRSTD	LVIPWRD	LVIREGD	SSREC	STOP	COPD
		Reset:	0	0	0	0	0	0	0	0
$FFCF	Mask-Option-Register (MOR)#	Read: Write:	OSCSEL1	OSCSEL0	R	R	R	R	R	R
		Erased:	1	1	1	1	1	1	1	1

† One-time writable register after each reset.
# MOR is a non-volatile FLASH register; write by programming.

= Unimplemented R = Reserved

Figure 3-1. CONFIG and MOR Registers Summary

3.2 Functional Description

The configuration registers and the mask option register are used in the initialization of various options. These two types of registers are configured differently:

- Configuration registers — Write-once registers after reset
- Mask option register — FLASH register (write by programming)

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

NOTE

The CONFIG registers are not in the FLASH memory but are special registers containing one-time writable latches after each reset. Upon a reset, the CONFIG registers default to predetermined settings as shown in Figure 3-2 and Figure 3-3.

The mask option register (MOR) is used for selecting one of the three clock options for the MCU. The MOR is a byte located in FLASH memory, and is written to by a FLASH programming routine.

3.3 Configuration Register 1 (CONFIG1)

Address: $001F

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

Figure 3-2. Configuration Register 1 (CONFIG1)

COPRS — COP Rate Select Bit

COPRS selects the COP time out period. Reset clears COPRS. (See Chapter 19 Computer Operating Properly (COP).)

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

LVISTOP — LVI Enable in Stop Mode Bit

When the LVIPWRD or LVIREGD bit is clear, setting the LVISTOP bit enables the LVI to operate during stop mode. Reset clears LVISTOP. (See Chapter 20 Low-Voltage Inhibit (LVI).)

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

NOTE

If LVISTOP=0, set LVIRSTD=1 before entering stop mode.

LVIRSTD — LVI Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module. (See Chapter 20 Low-Voltage Inhibit (LVI).)

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

LVIPWRD — V_{DD} LVI Circuit Disable Bit

LVIPWRD disables the V_{DD} LVI circuit. (See Chapter 20 Low-Voltage Inhibit (LVI).)

1 = V_{DD} LVI circuit disabled
0 = V_{DD} LVI circuit enabled

LVIREGD — V_{REG} LVI Circuit Disable Bit

LVIREGD disables the V_{REG} LVI circuit. (See Chapter 20 Low-Voltage Inhibit (LVI).)

1 = V_{REG} LVI circuit disabled
0 = V_{REG} LVI circuit enabled

NOTE

If LVIPWRD=1 and LVIREGD=1, set LVIRSTD=1 before entering stop mode.

SSREC — Short Stop Recovery Bit

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

1 = Stop mode recovery after 32 ICLK cycles
0 = Stop mode recovery after 4096 ICLK 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.

When the LVI is disabled in stop mode (LVISTOP=0), the system stabilization time for long stop recovery (4096 ICLK cycles) gives a delay longer than the LVI's turn-on time. 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 ICLK 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 Bit

STOP enables the STOP instruction.

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

COPD — COP Disable Bit

COPD disables the COP module. (See Chapter 19 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:	STOP_ ICLKDIS	STOP_ RCLKEN	STOP_ XCLKEN	OSCCLK1	OSCCLK0	0	0	SCIBD-SRC
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 3-3. Configuration Register 2 (CONFIG2)

STOP_ICLKDIS — Internal Oscillator Stop Mode Disable

STOP_ICLKDIS disables the internal oscillator during stop mode. Setting the STOP_ICLKDIS bit disables the oscillator during stop mode. (See Chapter 5 Oscillator (OSC).)
Reset clears this bit.

1 = Internal oscillator disabled during stop mode
0 = Internal oscillator enabled to operate during stop mode

STOP_RCLKEN — RC Oscillator Stop Mode Enable Bit

STOP_RCLKEN enables the RC oscillator to continue operating during stop mode. Setting the STOP_RCLKEN bit allows the oscillator to operate continuously even during stop mode. This is useful for driving the timebase module to allow it to generate periodic wake up while in stop mode. (See Chapter 5 Oscillator (OSC).)
Reset clears this bit.

1 = RC oscillator enabled to operate during stop mode
0 = RC oscillator disabled during stop mode

STOP_XCLKEN — X-tal Oscillator Stop Mode Enable Bit

STOP_XCLKEN enables the crystal (x-tal) oscillator to continue operating during stop mode. Setting the STOP_XCLKEN bit allows the x-tal oscillator to operate continuously even during stop mode. This is useful for driving the timebase module to allow it to generate periodic wake up while in stop mode. (See Chapter 5 Oscillator (OSC).) Reset clears this bit.

1 = X-tal oscillator enabled to operate during stop mode
0 = X-tal oscillator disabled during stop mode

OSCCLK1, OSCCLK0 — Oscillator Output Control Bits

OSCCLK1 and OSCCLK0 select which oscillator output to be driven out as OSCCLK to the timebase module (TBM). Reset clears these two bits.

OSCCLK1	OSCCLK0	Timebase Clock Source
0	0	Internal oscillator (ICLK)
0	1	RC oscillator (RCCLK)
1	0	X-tal oscillator (XTAL)
1	1	Not used

SCIBDSRC — SCI Baud Rate Clock Source

SCIBDSRC selects the clock source used for the standard SCI module (non-infrared SCI). The setting of this bit affects the frequency at which the SCI operates.

1 = Internal data bus clock, f_{BUS}, is used as clock source for SCI

0 = Oscillator clock, CGMXCLK, is used as clock source for SCI

3.5 Mask Option Register (MOR)

The mask option register (MOR) is used for selecting one of the three clock options for the MCU. The MOR is a byte located in FLASH memory, and is written to by a FLASH programming routine.

Address: $FFCF

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OSCSEL1	OSCSEL0	R	R	R	R	R	R
Reset:	Unaffected by reset							
Erased:	1	1	1	1	1	1	1	1

R = Reserved

Figure 3-4. Mask Option Register (MOR)

OSCSEL1, OSCSEL0 — Oscillator Selection Bits

OSCSEL1 and OSCSEL0 select which oscillator is used for the MCU CGMXCLK clock. The erase state of these two bits is logic 1. These bits are unaffected by reset. (See Table 3-1).

Bits 5–0 — Should be left as 1's

Table 3-1. CGMXCLK Clock Selection

OSCSEL1	OSCSEL0	CGMXCLK	OSC2 pin	Comments
0	0	—	—	Not used
0	1	ICLK	f_{BUS}	Internal oscillator generates the CGMXCLK.
1	0	RCCLK	f_{BUS}	RC oscillator generates the CGMXCLK. Internal oscillator is available after each POR or reset.
1	1	X-TAL	Inverting output of XTAL	X-tal oscillator generates the CGMXCLK. Internal oscillator is available after each POR or reset.

NOTE

The internal oscillator is a free running oscillator and is available after each POR or reset. It is turned-off in stop mode by setting the STOP_ICLKDIS bit in CONFIG2.

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* (Freescale document order number CPU08RM/AD) contains a description of the CPU instruction set, addressing modes, and architecture.

4.2 Features

- 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.

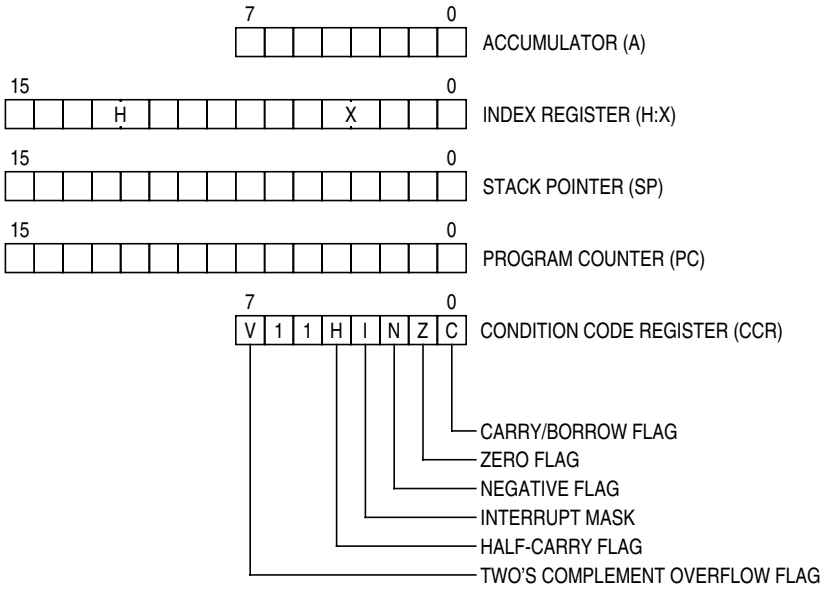


Figure 4-1. CPU Registers

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.

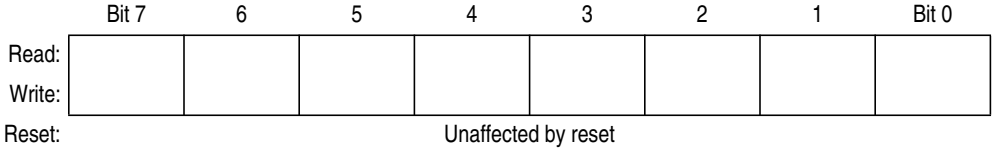


Figure 4-2. Accumulator (A)

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.

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

X = Indeterminate

Figure 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.

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

Figure 4-4. Stack Pointer (SP)

NOTE

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

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.

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

Figure 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 logic 1. The following paragraphs describe the functions of the condition code register.

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

X = Indeterminate

Figure 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* (Freescale 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.

4.8 Opcode Map

The opcode map is provided in Table 4-2.

Table 4-1. Instruction Set Summary

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

Table 4-1. Instruction Set Summary

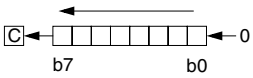
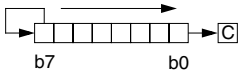
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
AND #*opr* AND *opr* AND *opr* AND *opr*,X AND *opr*,X AND ,X AND *opr*,SP AND *opr*,SP	Logical AND	A ← (A) & (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	A4 B4 C4 D4 E4 F4 9EE4 9ED4	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
ASL *opr* ASLA ASLX ASL *opr*,X ASL ,X ASL *opr*,SP	Arithmetic Shift Left (Same as LSL)		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff	4 1 1 4 3 5
ASR *opr* ASRA ASRX ASR *opr*,X ASR *opr*,X ASR *opr*,SP	Arithmetic Shift Right		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff	4 1 1 4 3 5
BCC *rel*	Branch if Carry Bit Clear	PC ← (PC) + 2 + rel ? (C) = 0	–	–	–	–	–	–	REL	24	rr	3
BCLR *n*, *opr*	Clear Bit n in M	Mn ← 0	–	–	–	–	–	–	DIR (b0) DIR (b1) DIR (b2) DIR (b3) DIR (b4) DIR (b5) DIR (b6) DIR (b7)	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	4 4 4 4 4 4 4 4
BCS *rel*	Branch if Carry Bit Set (Same as BLO)	PC ← (PC) + 2 + *rel* ? (C) = 1	–	–	–	–	–	–	REL	25	rr	3
BEQ *rel*	Branch if Equal	PC ← (PC) + 2 + *rel* ? (Z) = 1	–	–	–	–	–	–	REL	27	rr	3
BGE *opr*	Branch if Greater Than or Equal To (Signed Operands)	PC ← (PC) + 2 + *rel* ? (N ⊕ V) = 0	–	–	–	–	–	–	REL	90	rr	3
BGT *opr*	Branch if Greater Than (Signed Operands)	PC ← (PC) + 2 +*rel* ? (Z) | (N ⊕ V)=0	–	–	–	–	–	–	REL	92	rr	3
BHCC *rel*	Branch if Half Carry Bit Clear	PC ← (PC) + 2 + *rel* ? (H) = 0	–	–	–	–	–	–	REL	28	rr	3
BHCS *rel*	Branch if Half Carry Bit Set	PC ← (PC) + 2 + *rel* ? (H) = 1	–	–	–	–	–	–	REL	29	rr	3
BHI *rel*	Branch if Higher	PC ← (PC) + 2 + *rel* ? (C) | (Z) = 0	–	–	–	–	–	–	REL	22	rr	3
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

Table 4-1. Instruction Set Summary

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

Table 4-1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR V	H	I	N	Z	C	Address Mode	Opcode	Operand	Cycles
BSR *rel*	Branch to Subroutine	PC ← (PC) + 2; push (PCL) SP ← (SP) – 1; push (PCH) SP ← (SP) – 1 PC ← (PC) + *rel*	–	–	–	–	–	–	REL	AD	rr	4
CBEQ *opr,rel* CBEQA #*opr,rel* CBEQX #*opr,rel* CBEQ *opr*,X+,*rel* CBEQ X+,*rel* CBEQ *opr*,SP,*rel*	Compare and Branch if Equal	PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 3 + rel ? (X) – (M) = $00 PC ← (PC) + 3 + rel ? (A) – (M) = $00 PC ← (PC) + 2 + rel ? (A) – (M) = $00 PC ← (PC) + 4 + rel ? (A) – (M) = $00	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff rr	5 4 4 5 4 6
CLC	Clear Carry Bit	C ← 0	–	–	–	–	–	0	INH	98		1
CLI	Clear Interrupt Mask	I ← 0	–	–	0	–	–	–	INH	9A		2
CLR *opr* CLRA CLRX CLRH CLR *opr*,X CLR ,X CLR *opr*,SP	Clear	M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff	3 1 1 1 3 2 4
CMP #*opr* CMP *opr* CMP *opr* CMP *opr*,X CMP *opr*,X CMP ,X CMP *opr*,SP CMP *opr*,SP	Compare A with M	(A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A1 B1 C1 D1 E1 F1 9EE1 9ED1	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
COM *opr* COMA COMX COM *opr*,X COM ,X COM *opr*,SP	Complement (One's Complement)	M ← ($\overline{M}$) = $FF – (M) A ← ($\overline{A}$) = $FF – (M) X ← ($\overline{X}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\overline{M}$) = $FF – (M) M ← ($\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

Table 4-1. Instruction Set Summary

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

Table 4-1. Instruction Set Summary

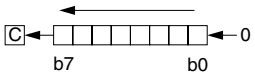
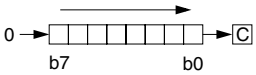
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
LDX #*opr* LDX *opr* LDX *opr* LDX *opr*,X LDX *opr*,X LDX ,X LDX *opr*,SP LDX *opr*,SP	Load X from M	X ← (M)	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP1 SP2	AE BE CE DE EE FE 9EEE 9EDE	ii dd hh ll ee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

Table 4-1. Instruction Set Summary

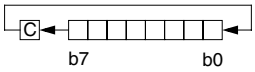
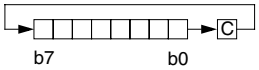
Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
PULH	Pull H from Stack	SP ← (SP + 1); Pull (H)	–	–	–	–	–	–	INH	8A		2
PULX	Pull X from Stack	SP ← (SP + 1); Pull (X)	–	–	–	–	–	–	INH	88		2
ROL *opr* ROLA ROLX ROL *opr*,X ROL ,X ROL *opr*,SP	Rotate Left through Carry		↕	–	–	↕	↕	↕	DIR INH INH 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 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

Table 4-1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				
STX *opr* STX *opr* STX *opr*,X STX *opr*,X STX ,X STX *opr*,SP STX *opr*,SP	Store X in M	M ← (X)	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP1 SP2	BF CF DF EF FF 9EEF 9EDF	dd hh ll ee ff ff ff ee ff	3 4 4 3 2 4 5
SUB #*opr* SUB *opr* SUB *opr* SUB *opr*,X SUB *opr*,X SUB ,X SUB *opr*,SP SUB *opr*,SP	Subtract	A ← (A) – (M)	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP1 SP2	A0 B0 C0 D0 E0 F0 9EE0 9ED0	ii dd hh ll ee ff ff ff ee ff	2 3 4 4 3 2 4 5
SWI	Software Interrupt	PC ← (PC) + 1; Push (PCL) SP ← (SP) – 1; Push (PCH) SP ← (SP) – 1; Push (X) SP ← (SP) – 1; Push (A) SP ← (SP) – 1; Push (CCR) SP ← (SP) – 1; I ← 1 PCH ← Interrupt Vector High Byte PCL ← Interrupt Vector Low Byte	–	–	1	–	–	–	INH	83		9
TAP	Transfer A to CCR	CCR ← (A)	↕	↕	↕	↕	↕	↕	INH	84		2
TAX	Transfer A to X	X ← (A)	–	–	–	–	–	–	INH	97		1
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 ← 0; Inhibit CPU clocking until interrupted	–	–	0	–	–	–	INH	8F		1

Table 4-1. Instruction Set Summary

Source Form	Operation	Description	Effect on CCR						Address Mode	Opcode	Operand	Cycles
			V	H	I	N	Z	C				

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

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	High Byte of Opcode in Hexadecimal
Low Byte of Opcode in Hexadecimal: 0	5 BRSET0 3 DIR — Cycles, Opcode Mnemonic, Number of Bytes / Addressing Mode

Chapter 5
Oscillator (OSC)

5.1 Introduction

The oscillator module consist of three types of oscillator circuits:

- Internal oscillator
- RC oscillator
- 1MHz to 8MHz crystal (x-tal) oscillator

The reference clock for the CGM and other MCU sub-systems is selected by programming the mask option register located at $FFCF.

The reference clock for the timebase module (TBM) is selected by the two bits, OSCCLK1 and OSCCLK0, in the CONFIG2 register.

The internal oscillator runs continuously after a POR or reset, and is always available. The RC and crystal oscillator cannot run concurrently; one is disabled while the other is selected; because the RC and x-tal circuits share the same OSC1 pin.

NOTE

The oscillator circuits are powered by the on-chip V_{REG} regulator, therefore, the output swing on OSC1 and OSC2 is from V_{SS} to V_{REG}.

Figure 5-1. shows the block diagram of the oscillator module.

5.2 Clock Selection

Reference clocks are selectable for the following sub-systems:

- CGMXCLK and CGMRCLK — Reference clock for clock generator module (CGM) and other MCU sub-systems other than TBM and COP. This is the main reference clock for the MCU.
- OSCCLK — Reference clock for timebase module (TBM).

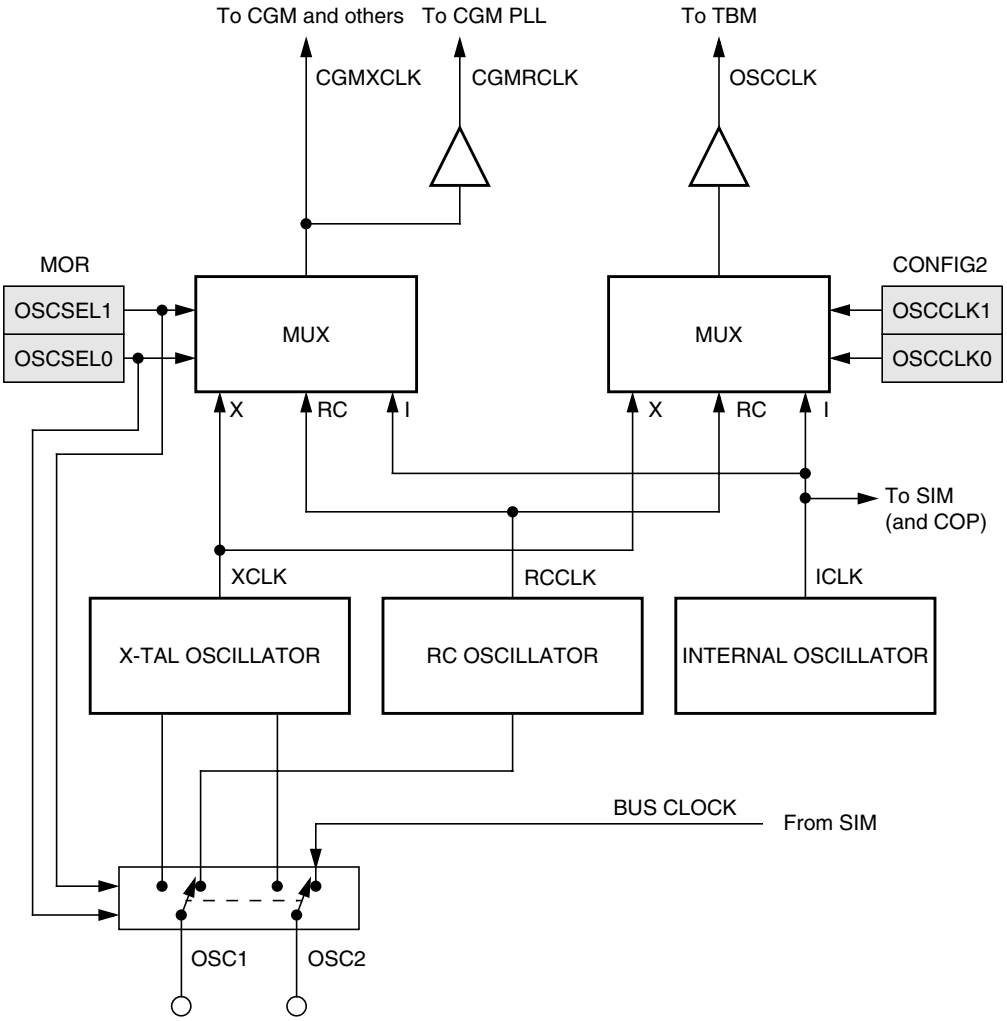


Figure 5-1. Oscillator Module Block Diagram

5.2.1 CGM Reference Clock Selection

The clock generator module (CGM) reference clock (CGMXCLK) is the reference clock input to the MCU. It is selected by programming two bits in a FLASH memory location; the mask option register (MOR), at $FFCF. See 3.5 Mask Option Register (MOR).

Address: $FFCF

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	OSCSEL1	OSCSEL0	R	R	R	R	R	R
Reset:	Unaffected by reset							
Erased:	1	1	1	1	1	1	1	1

R = Reserved

Figure 5-2. Mask Option Register (MOR)

Table 5-1. CGMXCLK Clock Selection

OSCSEL1	OSCSEL0	CGMXCLK	OSC2 Pin	Comments
0	0	—	—	Not used
0	1	ICLK	f_{BUS}	Internal oscillator generates the CGMXCLK.
1	0	RCCLK	f_{BUS}	RC oscillator generates the CGMXCLK. Internal oscillator is available after each POR or reset.
1	1	XCLK	Inverting output of X-TAL	X-tal oscillator generates the CGMXCLK. Internal oscillator is available after each POR or reset.

NOTE

The internal oscillator is a free running oscillator and is available after each POR or reset. It is turned-off in stop mode by setting the STOP_ICLKDIS bit in CONFIG2.

5.2.2 TBM Reference Clock Selection

The timebase module reference clock (OSCCLK) is selected by configuring two bits in the CONFIG2 register, at $001D. See Chapter 3 Configuration & Mask Option Registers (CONFIG & MOR).

Address: $001D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	STOP_ ICLKDIS	STOP_ RCLKEN	STOP_ XCLKEN	OSCCLK1	OSCCLK0	0	0	SCIBD-SRC
Write:								
Reset:	0	0	0	0	0	0	0	0

Figure 5-3. Configuration Register 2 (CONFIG2)

Table 5-2. Timebase Module Reference Clock Selection

OSCCLK1	OSCCLK0	Timebase Clock Source
0	0	Internal oscillator (ICLK)
0	1	RC oscillator (RCCLK)
1	0	X-tal oscillator (XCLK)
1	1	Not used

NOTE

The RCCLK or XCLK is only available if that clock is selected as the CGM reference clock, whereas the ICLK is always available.

5.3 Internal Oscillator

The internal oscillator clock (ICLK), with a frequency of f_{ICLK}, is a free running clock that requires no external components. It can be selected as the CGMXCLK for the CGM and MCU sub-systems; and the OSCCLK clock for the TBM. The ICLK is also the reference clock input to the computer operating properly (COP) module.

Due to the simplicity of the internal oscillator, it does not have the accuracy and stability of the RC oscillator or the x-tal oscillator. Therefore, the ICLK is not suitable where an accurate bus clock is required and it should not be used as the CGMRCLK to the CGM PLL.

The internal oscillator by default is always available and is free running after POR or reset. It can be turned-off in stop mode by setting the STOP_ICLKDIS bit before executing the STOP instruction.

Figure 5-4 shows the logical representation of components of the internal oscillator circuitry.

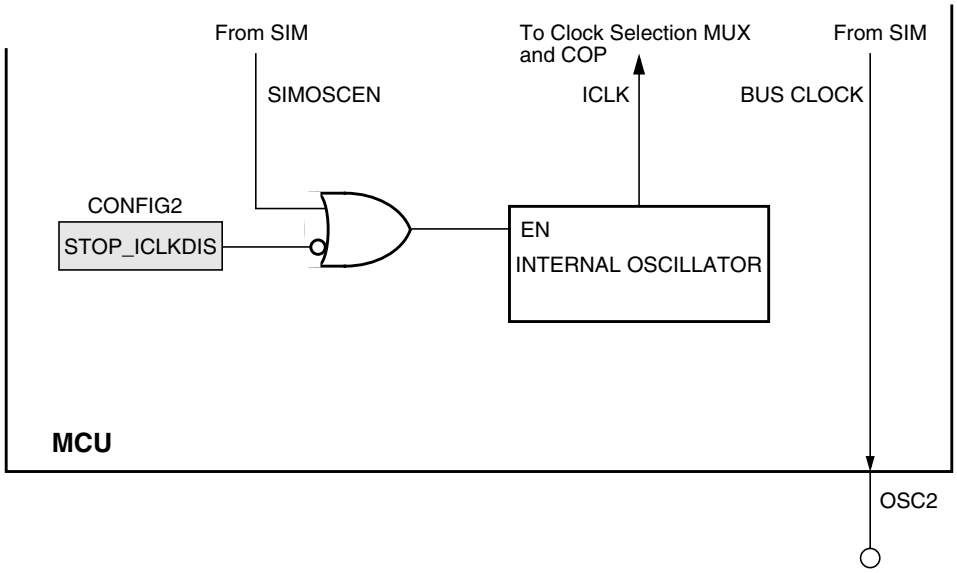


Figure 5-4. Internal Oscillator

5.4 RC Oscillator

The RC oscillator circuit is designed for use with an external resistor and a capacitor.

In its typical configuration, the RC oscillator requires two external components, one R and one C. Component values should have a tolerance of 1% or less, to obtain a clock source with less than 10% tolerance. The oscillator configuration uses two components:

- C_{EXT}
- R_{EXT}

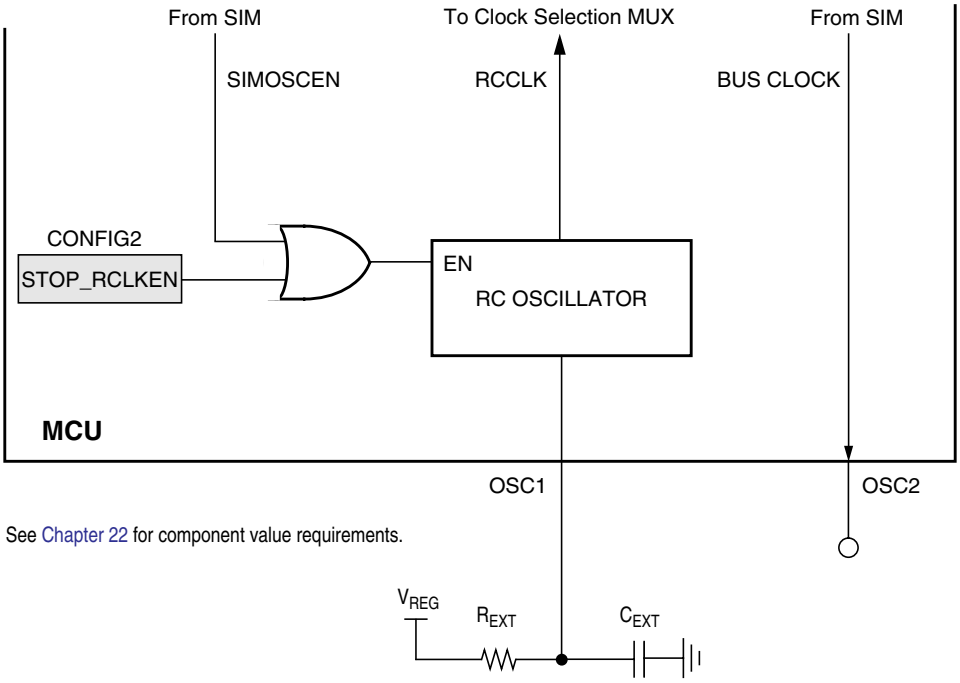


Figure 5-5. RC Oscillator

5.5 X-tal Oscillator

The crystal (x-tal) oscillator circuit is designed for use with an external 1–8 MHz crystal to provide an accurate clock source.

In its typical configuration, the x-tal oscillator is connected in a Pierce oscillator configuration, as shown in Figure 5-6. This figure shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

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

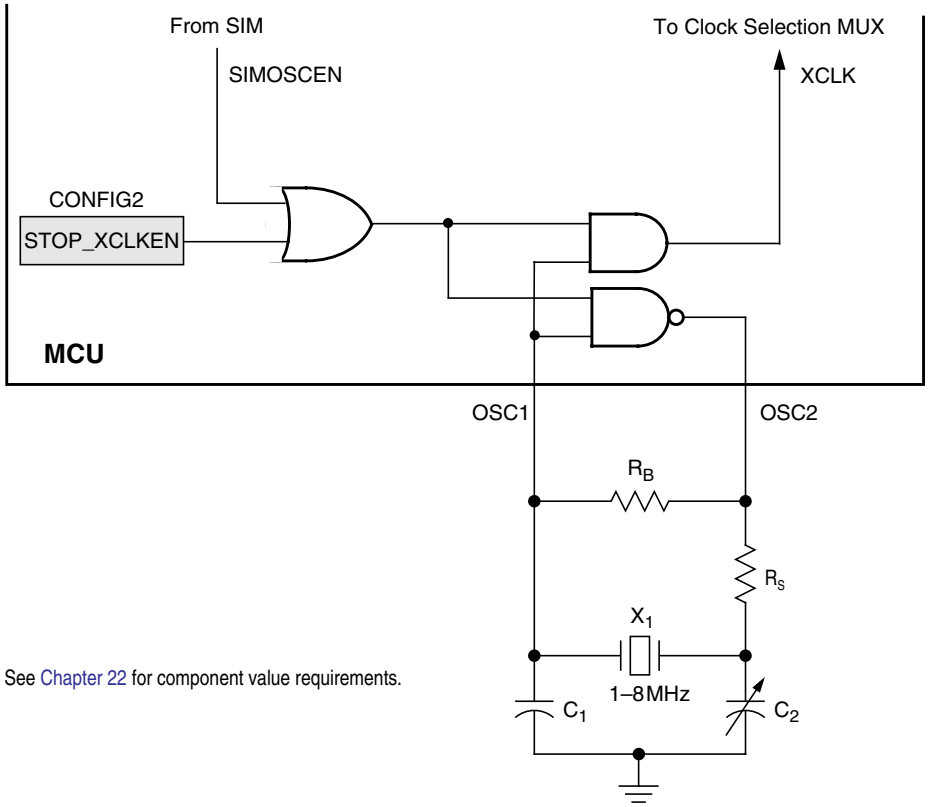


Figure 5-6. Crystal Oscillator

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

5.6 I/O Signals

The following paragraphs describe the oscillator I/O signals.

5.6.1 Crystal Amplifier Input Pin (OSC1)

OSC1 pin is an input to the crystal oscillator amplifier or the input to the RC oscillator circuit.

5.6.2 Crystal Amplifier Output Pin (OSC2)

When the x-tal oscillator is selected, OSC2 pin is the output of the crystal oscillator inverting amplifier.

When the RC oscillator or internal oscillator is selected, OSC2 pin is the output of the internal bus clock.

5.6.3 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal from the system integration module (SIM) enables/disables the x-tal oscillator, the RC-oscillator, or the internal oscillator circuit.

5.6.4 CGM Oscillator Clock (CGMXCLK)

The CGMXCLK clock is output from the x-tal oscillator, RC oscillator or the internal oscillator. This clock drives to CGM and other MCU sub-systems.

5.6.5 CGM Reference Clock (CGMRCLK)

This is buffered signal of CGMXCLK, it is used by the CGM as the phase-locked-loop (PLL) reference clock.

5.6.6 Oscillator Clock to Time Base Module (OSCCLK)

The OSCCLK is the reference clock that drives the timebase module. See Chapter 10 Timebase Module (TBM).

5.7 Low Power Modes

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

5.7.1 Wait Mode

The WAIT instruction has no effect on the oscillator module. CGMXCLK continues to drive to the clock generator module, and OSCCLK continues to drive the timebase module.

5.7.2 Stop Mode

The STOP instruction disables the x-tal or the RC oscillator circuit, and hence the CGMXCLK clock stops running. For continuous x-tal or RC oscillator operation in stop mode, set the STOP_XCLKEN (for x-tal) or STOP_RCLKEN (for RC) bit to logic 1 before entering stop mode.

The internal oscillator clock continues operation in stop mode. It can be disabled by setting the STOP_ICLKDIS bit to logic 1 before entering stop mode.

5.8 Oscillator During Break Mode

The oscillator continues to drive CGMXCLK when the device enters the break state.

Chapter 6
Clock Generator Module (CGM)

6.1 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, CGMPCLK, divided by two. CGMOUT is the clock from which the SIM derives the system clocks, including the bus clock, which is at a frequency of CGMOUT 2.

The PLL is a frequency generator designed for use with a crystal (1 to 8 MHz) to generate a base frequency and dividing to a maximum bus frequency of 8 MHz.

6.2 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

6.3 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, and the divided VCO clock, CGMPCLK.
- Base clock selector circuit — This software-controlled circuit selects either CGMXCLK divided by two or the divided VCO clock, CGMPCLK, divided by two as the base clock, CGMOUT. The SIM derives the system clocks from either CGMOUT or CGMXCLK.

Figure 6-1 shows the structure of the CGM.

Figure 6-2 is a summary of the CGM registers.

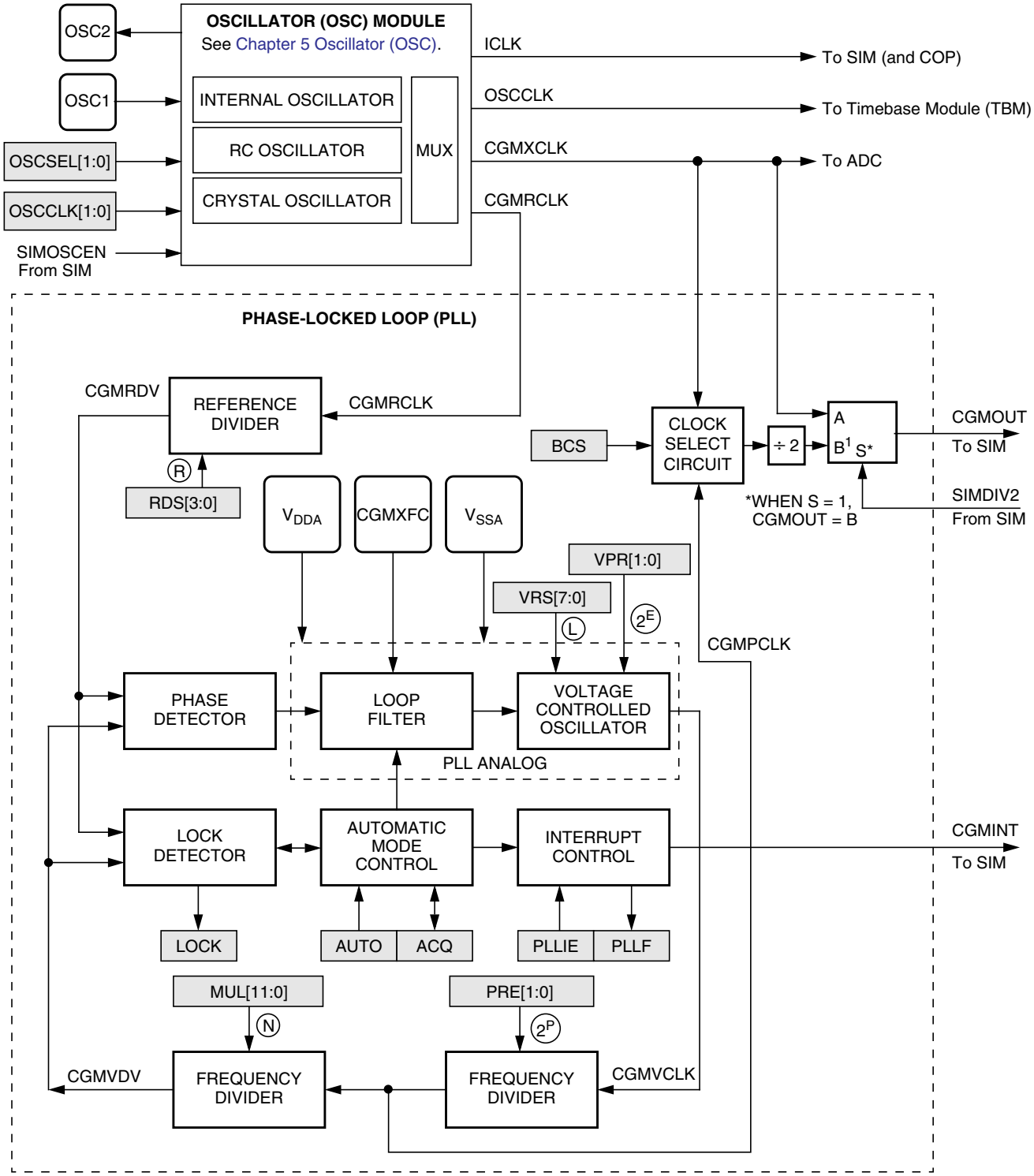


Figure 6-1. CGM Block Diagram

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

= Unimplemented R = Reserved

NOTES:
1. When AUTO = 0, PLLIE is forced clear and is read-only.
2. When AUTO = 0, PLLF and LOCK read as clear.
3. When AUTO = 1, $\overline{ACQ}$ 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 6-2. CGM I/O Register Summary

6.3.1 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 5 Oscillator (OSC) for detailed oscillator circuit description. See Chapter 10 Timebase Module (TBM) for detailed description on TBM.

6.3.2 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.

6.3.3 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 (1 MHz–8 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 6.3.6 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 6.3.4 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.

6.3.4 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 6.5.2 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 6.3.8 Base Clock Selector Circuit.) The PLL is automatically in tracking mode when not in acquisition mode or when the $\overline{ACQ}$ bit is set.

6.3.5 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 6.5.2 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 6.3.8 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 6.6 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 6.5.2 PLL Bandwidth Control Register.) is a read-only indicator of the mode of the filter. (See 6.3.4 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 6.8 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 6.8 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 6.5.1 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 6.8 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.

6.3.6 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:

$$f_{VCLK} = 2^P \times f_{CGMPCLK} = 2^P \times 4 \times f_{BUS}$$

   where P is the power of two multiplier, and can be 0, 1, 2, or 3
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 22 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}.

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

$$f_{BUS} = \frac{f_{VCLK}}{2^P \times 4}$$

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 $f_{NOM} = 125\,kHz$

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

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,

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

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 6-1 provides numeric examples (numbers are in hexadecimal notation):

Table 6-1. Numeric Examples

CGMVCLK	CGMPCLK	f_{BUS}	f_{RCLK}	R	N	P	E	L
32 MHz	32 MHz	8.0 MHz	2 MHz	1	10	0	2	40
16 MHz	16 MHz	4.0 MHz	2 MHz	1	8	0	1	40
32 MHz	32 MHz	8.0 MHz	4 MHz	1	8	0	2	40
16 MHz	16 MHz	4.0 MHz	4 MHz	1	4	0	1	40
32 MHz	32 MHz	8.0 MHz	8 MHz	1	4	0	2	40
16 MHz	16 MHz	4.0 MHz	8 MHz	1	2	0	1	40
29.4912 MHz	29.4912 MHz	7.3728 MHz	4.9152 MHz	1	6	0	2	3C
19.6608 MHz	19.6608 MHz	4.9152 MHz	4.9152 MHz	1	4	0	2	27

6.3.7 Special Programming Exceptions

The programming method described in 6.3.6 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 6.3.8 Base Clock Selector Circuit.)

6.3.8 Base Clock Selector Circuit

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

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.

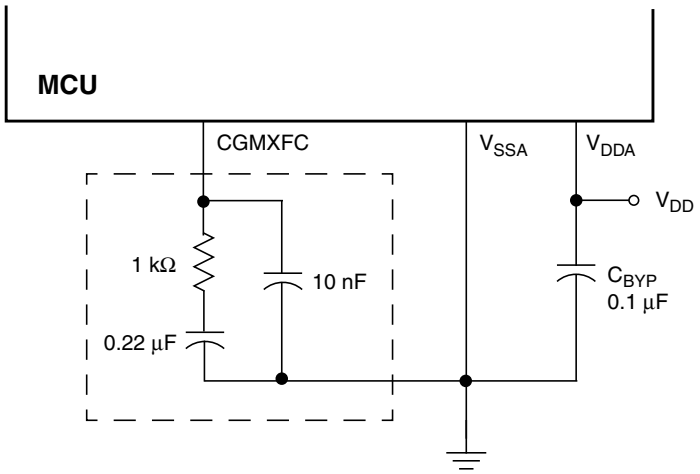
6.3.9 CGM External Connections

In its typical configuration, the CGM requires up to four external components.

Figure 6-3 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 6.8 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 6-3. CGM External Connections

6.4 I/O Signals

The following paragraphs describe the CGM I/O signals.

6.4.1 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 6-3.)

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.

6.4.2 PLL Analog Power Pin (V_{DDA})

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

NOTE

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

6.4.3 PLL Analog Ground Pin (V_{SSA})

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

NOTE

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

6.4.4 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.

6.4.5 CGM Reference Clock (CGMRCLK)

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

6.4.6 CGM VCO Clock Output (CGMVCLK)

CGMVCLK is the clock output from the VCO.

6.4.7 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 divided VCO clock, CGMPCLK, divided by two.

6.4.8 CGM CPU Interrupt (CGMINT)

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

6.5 CGM Registers

The following registers control and monitor operation of the CGM:

- PLL control register (PCTL)
  (See 6.5.1 PLL Control Register.)
- PLL bandwidth control register (PBWC)
  (See 6.5.2 PLL Bandwidth Control Register.)
- PLL multiplier select registers (PMSH and PMSL)
  (See 6.5.3 PLL Multiplier Select Registers.)
- PLL VCO range select register (PMRS)
  (See 6.5.4 PLL VCO Range Select Register.)
- PLL reference divider select register (PMDS)
  (See 6.5.5 PLL Reference Divider Select Register.)

6.5.1 PLL Control Register

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

Address: $0036

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

= Unimplemented

Figure 6-4. 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 6.3.8 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 divided VCO clock, CGMPCLK, 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 CGMPCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. (See 6.3.8 Base Clock Selector Circuit.) Reset clears the BCS bit.

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

NOTE

PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMPCLK requires two writes to the PLL control register. (See 6.3.8 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 6.3.3 PLL Circuits and 6.3.6 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 6-2. 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 6.3.3 PLL Circuits, 6.3.6 Programming the PLL, and 6.5.4 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 6-3. 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.

6.5.2 PLL Bandwidth Control Register

The PLL bandwidth control register (PBWC):

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

Address: $0037

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

= Unimplemented R = Reserved

Figure 6-5. 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

6.5.3 PLL Multiplier Select Registers

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

Address: $0038

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

= Unimplemented

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

Address: $0039

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

Figure 6-7. 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 6.3.3 PLL Circuits and 6.3.6 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).

6.5.4 PLL VCO Range Select Register

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

Address: $003A

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

Figure 6-8. 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 6.3.3 PLL Circuits, 6.3.6 Programming the PLL, and 6.5.1 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 6.3.7 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 6.3.8 Base Clock Selector Circuit and 6.3.7 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.

6.5.5 PLL Reference Divider Select Register

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

Address: $003B

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

= Unimplemented

Figure 6-9. 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 6.3.3 PLL Circuits and 6.3.6 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 6.3.7 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.

6.6 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, CGMPCLK, divided by two can be selected as the CGMOUT source by setting BCS in the PCTL. When the PLL exits lock, the

VCO clock frequency is corrupt, and appropriate precautions should be taken. If the application is not frequency sensitive, interrupts should be disabled to prevent PLL interrupt service routines from impeding software performance or from exceeding stack limitations.

NOTE

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

6.7 Special Modes

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

6.7.1 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.

6.7.2 Stop Mode

The STOP instruction disables the PLL analog circuits and no clock will be driven out of the VCO.

When entering stop mode with the VCO clock (CGMPCLK) selected, before executing the STOP instruction:

1. Set the oscillator stop mode enable bit (STOP_XCLKEN in CONFIG2) if continuos clock is required in stop mode.
2. Clear the BCS bit to select CGMXCLK as CGMOUT.

On exit from stop mode:

1. Set the PLLON bit if cleared before entering stop mode.
2. Wait for PLL to lock by checking the LOCK bit.
3. Set BCS bit to select CGMPCLK as CGMOUT.

6.7.3 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 7.7.3 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.

6.8 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.

6.8.1 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.

6.8.2 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 6.3.3 PLL Circuits, 6.3.6 Programming the PLL, and 6.5.5 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 6.8.3 Choosing a Filter.)

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

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

6.8.3 Choosing a Filter

As described in 6.8.2 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 6-10 is recommended when using a 4MHz reference clock (CGMRCLK). Figure 6-10 **(a)** is used for applications requiring better stability. Figure 6-10 **(b)** is used in low-cost applications where stability is not critical.

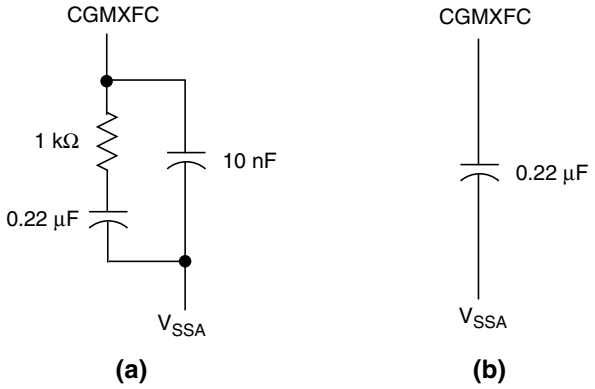


Figure 6-10. PLL Filter

Chapter 7
System Integration Module (SIM)

7.1 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 7-1. Figure 7-2 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

Table 7-1 shows the internal signal names used in this section.

Table 7-1. Signal Name Conventions

Signal Name	Description
ICLK	Internal oscillator clock
CGMXCLK	Selected oscillator clock from oscillator module
CGMVCLK, CGMPCLK	PLL output and the divided PLL output
CGMOUT	CGMPCLK-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

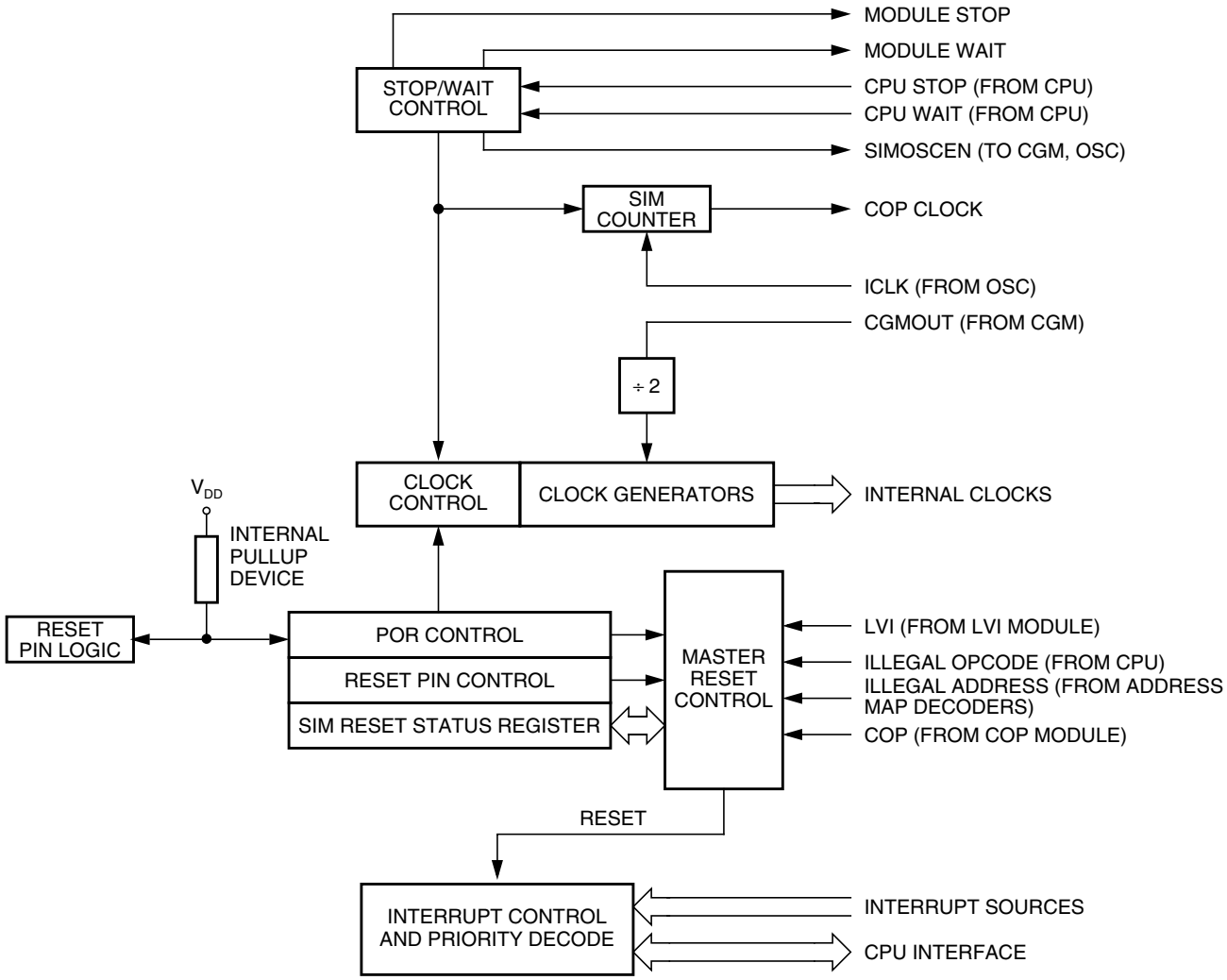


Figure 7-1. SIM Block Diagram

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$FE00	SIM Break Status Register (SBSR)	Read: Write:	R	R	R	R	R	R	SBSW NOTE	R
		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	MODRST	LVI	0
		Write:								
		POR:	1	0	0	0	0	0	0	0
$FE03	SIM Break Flag Control Register (SBFCR)	Read: Write:	BCFE	R	R	R	R	R	R	R
		Reset:	0							
$FE04	Interrupt Status Register 1 (INT1)	Read:	IF6	IF5	IF4	IF3	IF2	IF1	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

Figure 7-2. SIM I/O Register Summary

$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	IF21	IF20	IF19	IF18	IF17	IF16	IF15
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 7-2. SIM I/O Register Summary

7.2 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 7-3. This clock can come from either an external oscillator or from the on-chip PLL. (See Chapter 6 Clock Generator Module (CGM).)

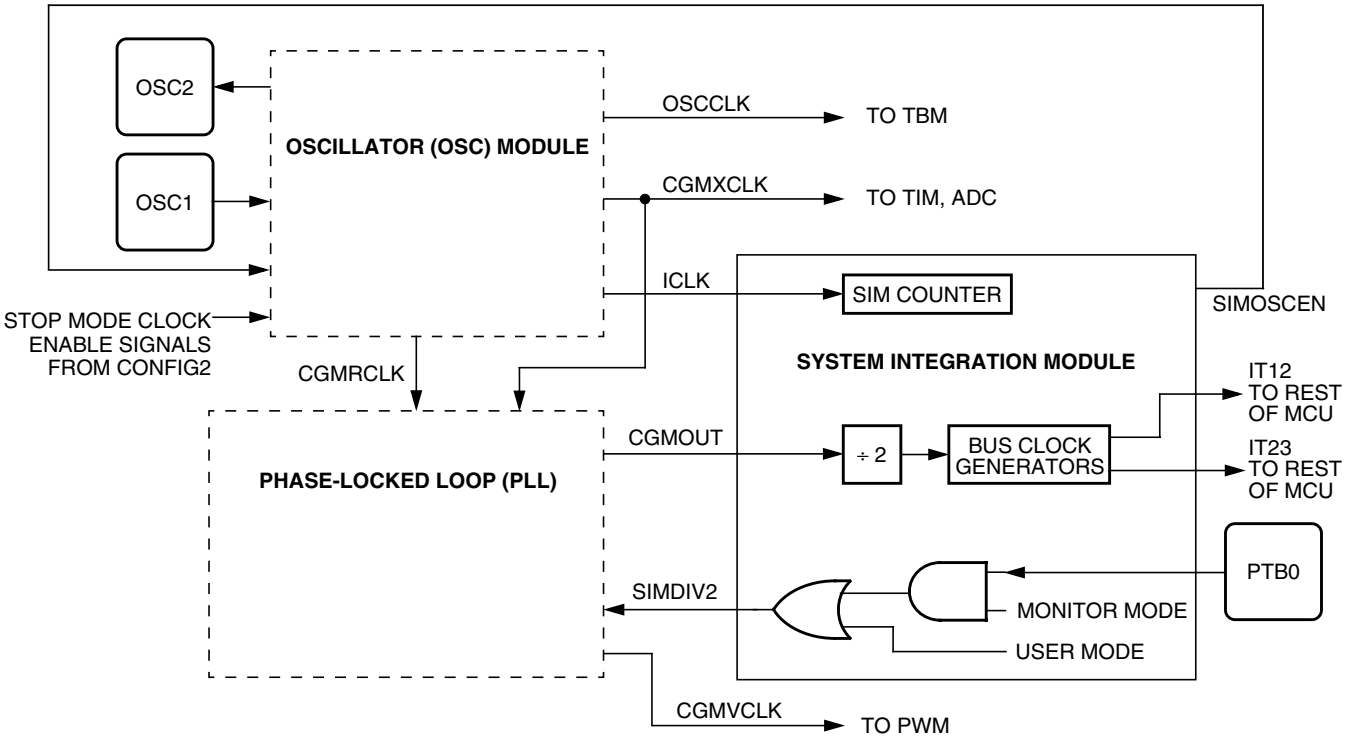


Figure 7-3. CGM Clock Signals

7.2.1 Bus Timing

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

7.2.2 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 ICLK cycle POR timeout has completed. The $\overline{RST}$ pin is driven low by the SIM during this entire period. The IBUS clocks start upon completion of the timeout.

7.2.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows ICLK 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 ICLK cycles. (See 7.6.2 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.

7.3 Reset and System Initialization

The MCU has these reset sources:

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

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

An internal reset clears the SIM counter (see 7.4 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See 7.7 SIM Registers.)

7.3.1 External Pin Reset

The $\overline{RST}$ pin circuit includes an internal pull-up device. Pulling the asynchronous $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for at least the minimum t_{RL} time and no other reset sources are present. See Table 7-2 for details. Figure 7-4 shows the relative timing.

Table 7-2. Reset Recovery

Reset Recovery Type	Actual Number of Cycles
POR/LVI	4163 (4096 + 64 + 3)
All others	67 (64 + 3)

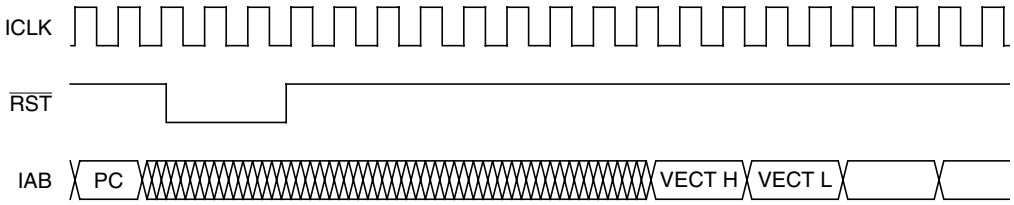


Figure 7-4. External Reset Timing

7.3.2 Active Resets from Internal Sources

All internal reset sources actively pull the $\overline{RST}$ pin low for 32 ICLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see Figure 7-5). An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR (see Figure 7-6).

NOTE

For LVI or POR resets, the SIM cycles through 4096 + 32 ICLK cycles during which the SIM forces the $\overline{RST}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{RST}$ shown in Figure 7-5.

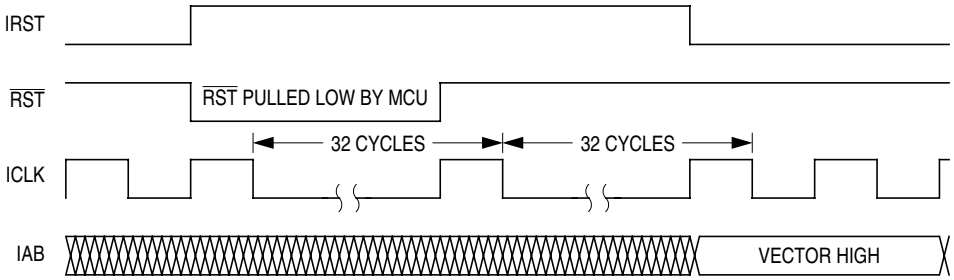


Figure 7-5. Internal Reset Timing

The COP reset is asynchronous to the bus clock.

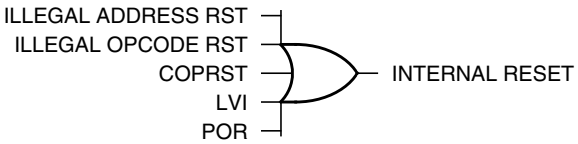


Figure 7-6. Sources of Internal Reset

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

7.3.2.1 Power-On Reset

When power is first applied to the MCU, the power-on reset module (POR) generates a pulse to indicate that power-on has occurred. The external reset pin ($\overline{RST}$) is held low while the SIM counter counts out 4096 + 32 ICLK cycles. Thirty-two ICLK 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 ICLK 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.

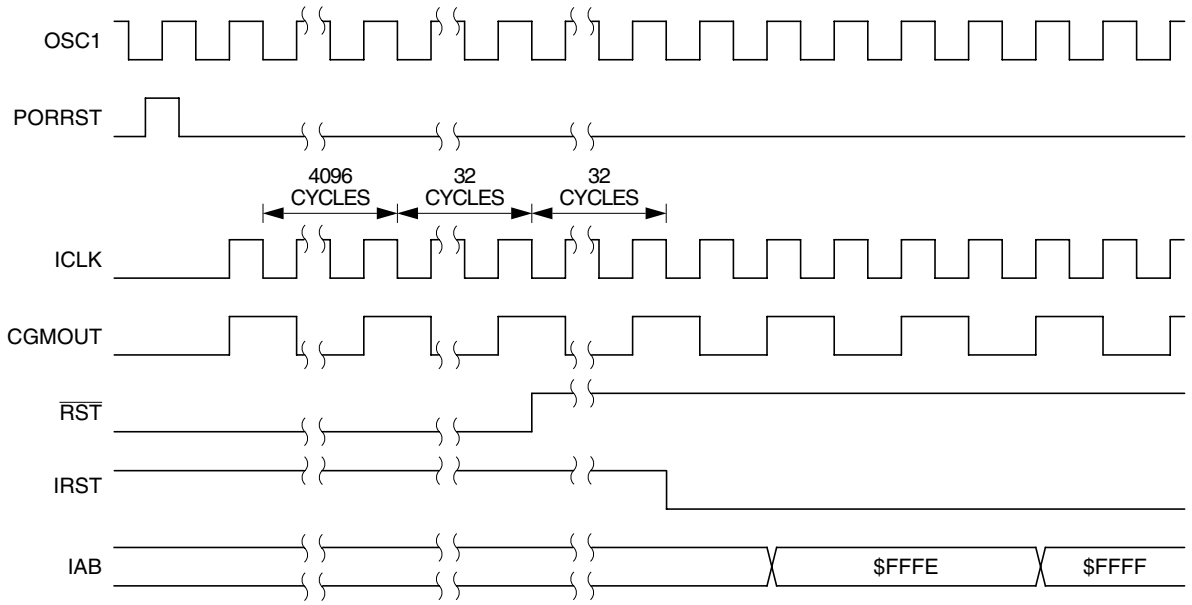


Figure 7-7. POR Recovery

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

To prevent a COP module timeout, write any value to location $FFFF. Writing to location $FFFF clears the COP counter and bits 12 through 5 of the SIM counter. The SIM counter output, which occurs at least every $2^{13} – 2^4$ ICLK 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 $\overline{RST}$ pin or the $\overline{IRQ1}$ 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 $\overline{RST}$ or the $\overline{IRQ1}$ pin. This prevents the COP from becoming disabled as a result of external noise. During a break state, V_{TST} on the $\overline{RST}$ pin disables the COP module.

7.3.2.3 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.

7.3.2.4 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.

7.3.2.5 Low-Voltage Inhibit (LVI) Reset

The low-voltage inhibit module (LVI) asserts its output to the SIM when the V_{DD} voltage falls to the LVI_{TRIPF} voltage. The LVI bit in the SIM reset status register (SRSR) is set, and the external reset pin ($\overline{RST}$) is held low while the SIM counter counts out 4096 + 32 ICLK cycles. Thirty-two ICLK 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.

7.3.2.6 Monitor Mode Entry Module Reset

The monitor mode entry module reset asserts its output to the SIM when monitor mode is entered in the condition where the reset vectors are blank ($FF). (See Chapter 8 Monitor Mode (MON).) When MODRST gets asserted, an internal reset occurs. The SIM actively pulls down the $\overline{RST}$ pin for all internal reset sources.

7.4 SIM Counter

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

7.4.1 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.

7.4.2 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.

7.4.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See 7.6.2 Stop Mode for details.) The SIM counter is free-running after all reset states. (See 7.3.2 Active Resets from Internal Sources for counter control and internal reset recovery sequences.)

7.5 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

7.5.1 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 7-8 shows interrupt entry timing, and Figure 7-9 shows interrupt recovery timing.

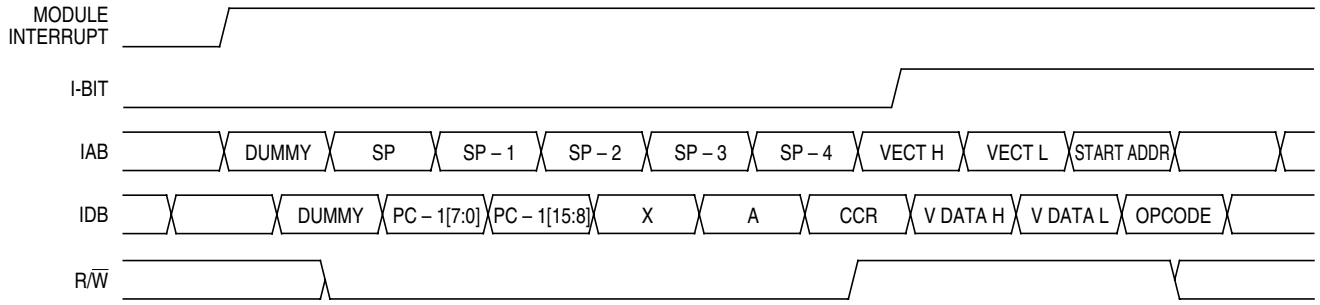


Figure 7-8. Interrupt Entry Timing

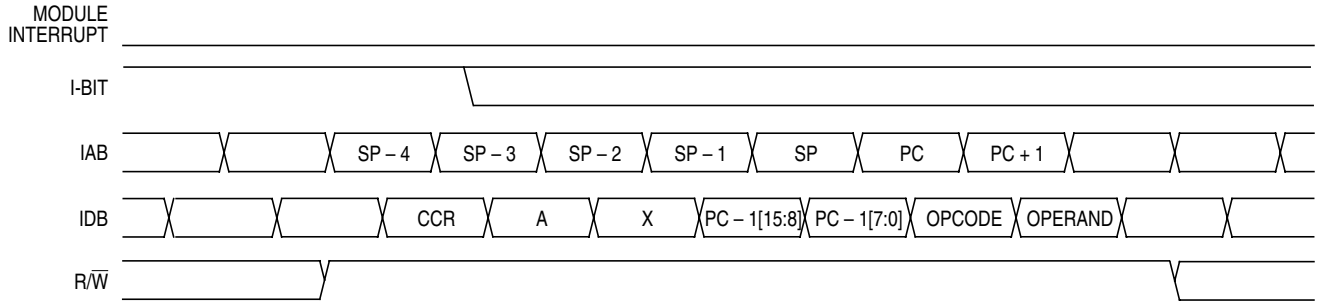


Figure 7-9. Interrupt Recovery Timing

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 7-10.)

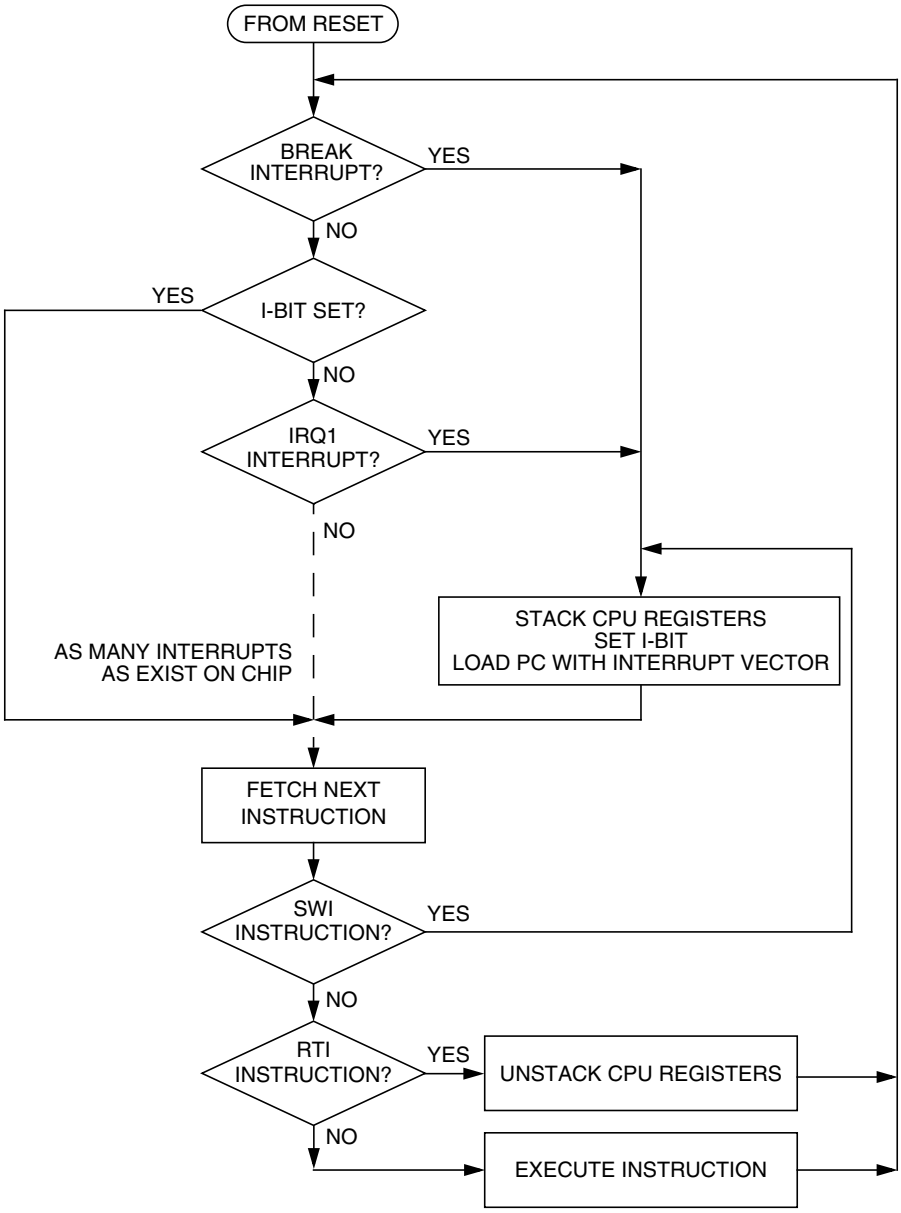


Figure 7-10. Interrupt Processing

7.5.1.1 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 7-11 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.

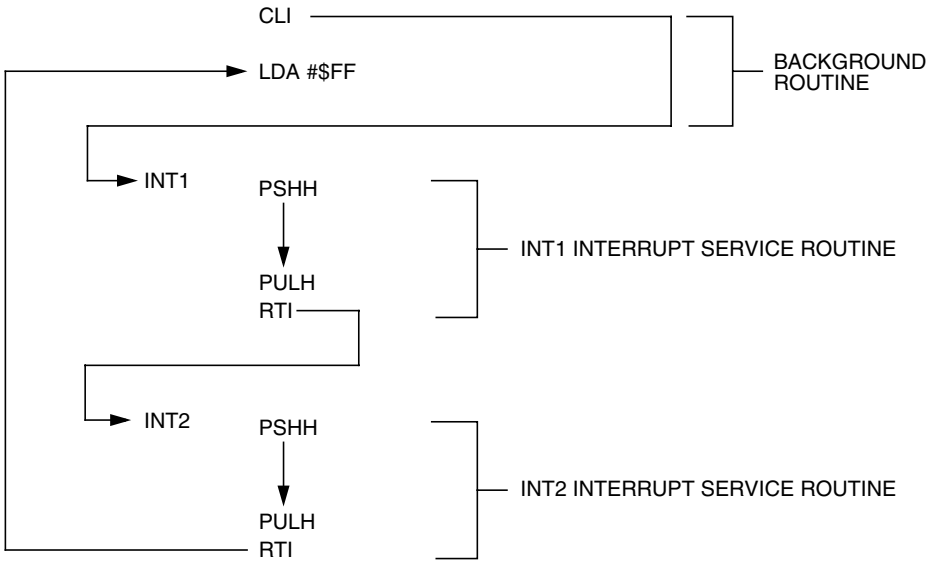


Figure 7-11. Interrupt Recognition Example

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.

7.5.1.2 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.

7.5.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. Table 7-3 summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

7.5.2.1 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 7-12. 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 7-3.

1 = Interrupt request present
0 = No interrupt request present

Bit 0 and Bit 1 — Always read 0

7.5.2.2 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 7-13. 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 7-3.

1 = Interrupt request present
0 = No interrupt request present

7.5.2.3 Interrupt Status Register 3

Address: $FE06

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	IF21	IF20	IF19	IF18	IF17	IF16	IF15
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 7-14. Interrupt Status Register 3 (INT3)

IF21–IF15 — Interrupt Flags 21–15

These flags indicate the presence of an interrupt request from the source shown in Table 7-3.

1 = Interrupt request present
0 = No interrupt request present

Table 7-3. Interrupt Sources

Priority	INT Flag	Vector Address	Interrupt Source
Lowest	—	$FFD0	Reserved
		$FFD1	
	IF21	$FFD2	Timebase
		$FFD3	
	IF20	$FFD4	Infrared SCI Transmit
		$FFD5	
	IF19	$FFD6	Infrared SCI Receive
		$FFD7	
	IF18	$FFD8	Infrared SCI Error
		$FFD9	
	IF17	$FFDA	SPI Transmit
		$FFDB	
	IF16	$FFDC	SPI Receive
		$FFDD	
	IF15	$FFDE	ADC Conversion Complete
		$FFDF	
	IF14	$FFE0	Keyboard
		$FFE1	
	IF13	$FFE2	SCI Transmit
		$FFE3	
	IF12	$FFE4	SCI Receive
		$FFE5	
	IF11	$FFE6	SCI Error
		$FFE7	
	IF10	$FFE8	MMIIC
		$FFE9	
	IF9	$FFEA	TIM2 Overflow
		$FFEB	
	IF8	$FFEC	TIM2 Channel 1
		$FFED	
	IF7	$FFEE	TIM2 Channel 0
		$FFEF	
	IF6	$FFF0	TIM1 Overflow
		$FFF1	
	IF5	$FFF2	TIM1 Channel 1
		$FFF3	
	IF4	$FFF4	TIM1 Channel 0
		$FFF5	
	IF3	$FFF6	PLL
		$FFF7	
	IF2	$FFF8	$\overline{\text{IRQ2}}$
		$FFF9	
	IF1	$FFFA	IRQ1
		$FFFB	
	—	$FFFC	SWI
		$FFFD	
	—	$FFFE	Reset
Highest		$FFFF	

7.5.3 Reset

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

7.5.4 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. (See Chapter 21 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.

7.5.5 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.

7.6 Low-Power Modes

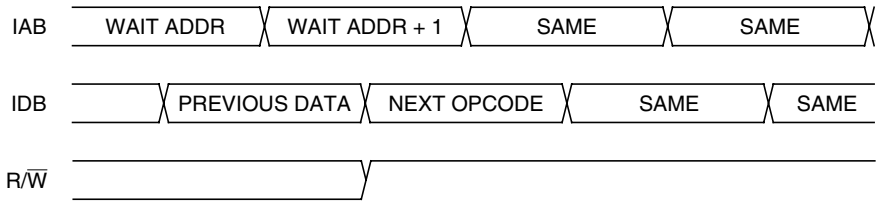
Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described in the following subsections. Both STOP and WAIT clear the interrupt mask (I) in the condition code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

7.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. Figure 7-15 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.

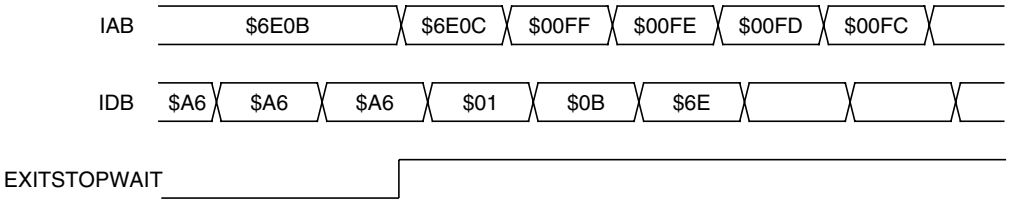
Wait mode also can be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the mask option register is logic 0, then the computer operating properly module (COP) is enabled and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



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

Figure 7-15. Wait Mode Entry Timing

Figure 7-16 and Figure 7-17 show the timing for WAIT recovery.



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

Figure 7-16. Wait Recovery from Interrupt or Break

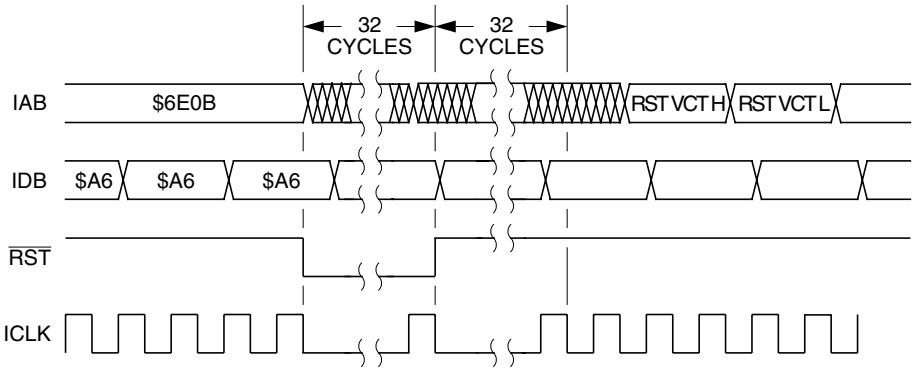


Figure 7-17. Wait Recovery from Internal Reset

7.6.2 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 ICLK cycles down to 32. This is ideal for applications using canned oscillators that do not require long start-up times from stop mode.

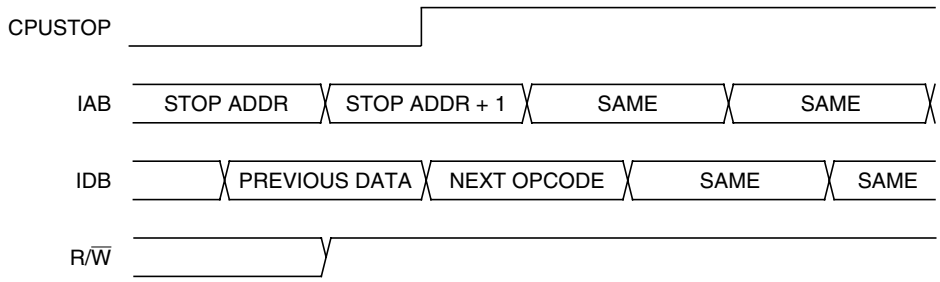
NOTE

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

The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. Figure 7-18 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 7-18. Stop Mode Entry Timing

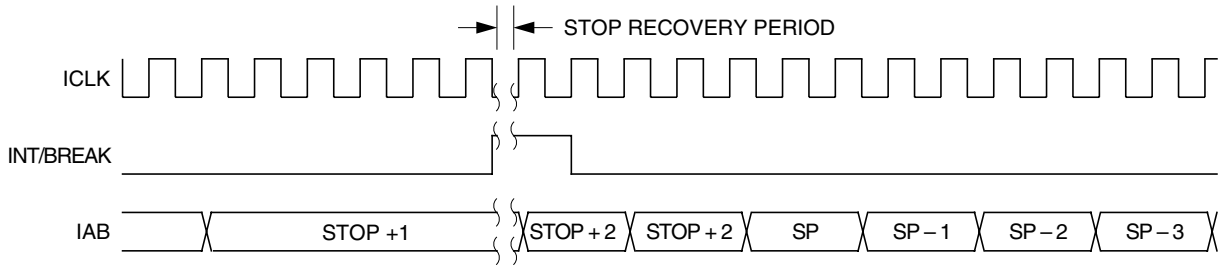


Figure 7-19. Stop Mode Recovery from Interrupt

7.7 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

7.7.1 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 7-20. SIM Break Status Register (SBSR)

SBSW — SIM Break Stop/Wait

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it.

1 = Wait mode was exited by break interrupt.

0 = Wait mode was not exited by break interrupt.

7.7.2 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.

The register is initialized on power up with the POR bit set and all other bits cleared. During a POR or any other internal reset, the $\overline{RST}$ pin is pulled low. After the pin is released, it will be sampled 32 CGMXCLK cycles later. If the pin is not above V_{IH} at this time, then the PIN bit may be set, in addition to whatever other bits are set.

Address: $FE01

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

= Unimplemented

Figure 7-21. 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

MODRST — Monitor Mode Entry Module Reset Bit

1 = Last reset caused by monitor mode entry when vector locations $FFFE and $FFFF are $FF after POR while $\overline{IRQ1} = V_{DD}$
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

7.7.3 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.

Address: $FE03

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

R = Reserved

Figure 7-22. 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 8
Monitor Mode (MON)

8.1 Introduction

The monitor module allows debugging and programming of the microcontroller unit (MCU) through a single-wire interface with a host computer. Monitor mode entry can be achieved without use of the higher test voltage, V_{TST}, as long as vector addresses $FFFE and $FFFF are blank, thus reducing the hardware requirements for in-circuit programming.

8.2 Features

Features of the monitor ROM include:

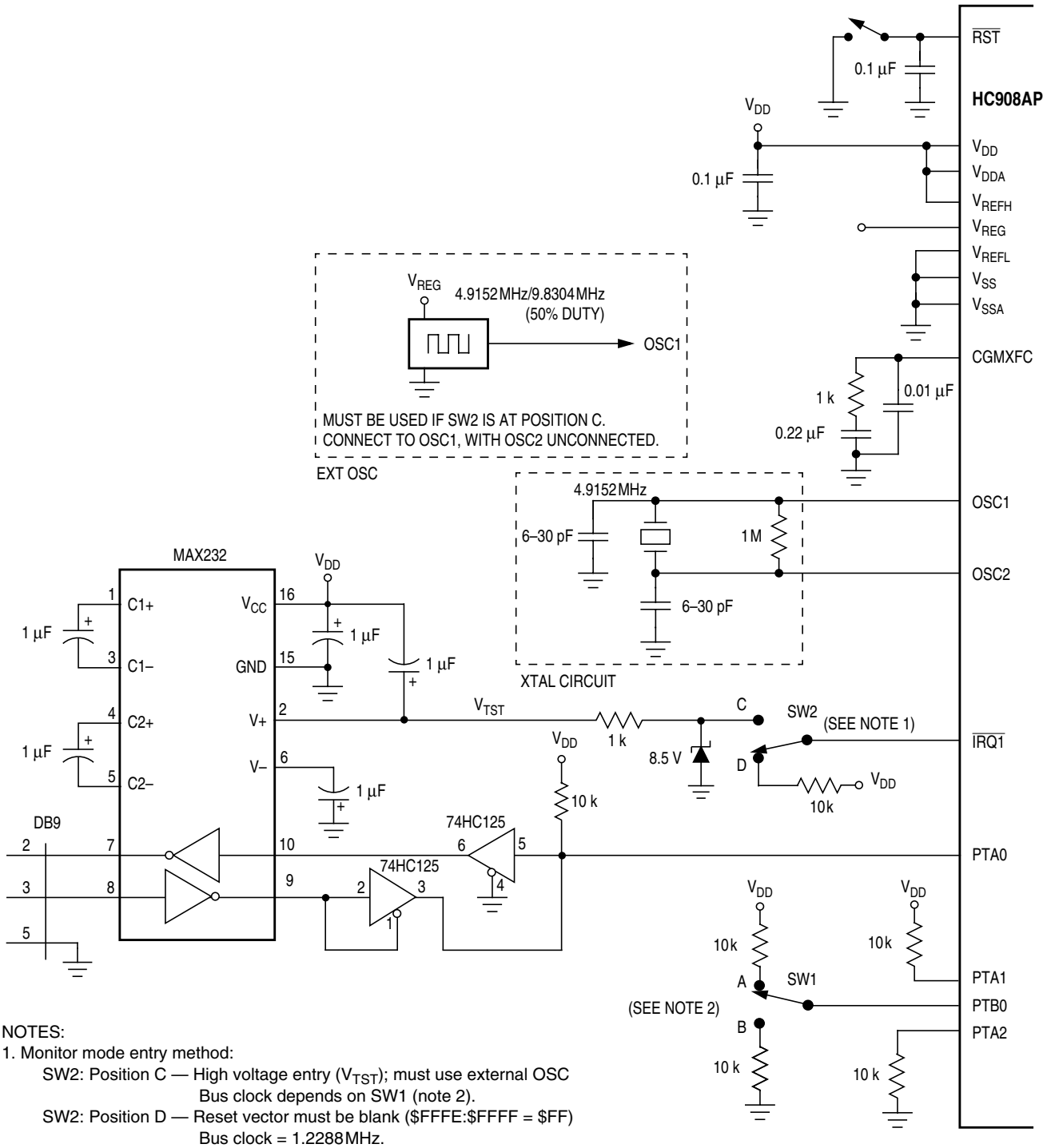
- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- Execution of code in RAM or FLASH
- FLASH memory security feature[1]
- 959 bytes monitor ROM code size
  ($FC00–$FDFF and $FE10–$FFCE)
- Monitor mode entry without high voltage, V_{TST}, if reset vector is blank ($FFFE and $FFFF contain $FF)
- Standard monitor mode entry if high voltage, V_{TST}, is applied to $\overline{IRQ1}$
- Resident routines for in-circuit programming

8.3 Functional Description

The monitor module receives and executes commands from a host computer. Figure 8-1 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.

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH 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 = 1.2288 MHz.
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
3. See Table 22-4 for V_{TST} voltage level requirements.

Figure 8-1. Monitor Mode Circuit

8.3.1 Entering Monitor Mode

Table 8-1 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. If $FFFE and $FFFF do not contain $FF (programmed state):
   – The external clock is 4.9152 MHz with PTB0 low or 9.8304 MHz with PTB0 high
   – $\overline{IRQ1} = V_{TST}$
2. If $FFFE and $FFFF both contain $FF (erased state):
   – The external clock is 9.8304 MHz
   – $\overline{IRQ1} = V_{DD}$ (this can be implemented through the internal $\overline{IRQ1}$ pullup)

If V_{TST} is applied to $\overline{IRQ1}$ and PTB0 is low upon monitor mode entry (above condition set 1), the bus frequency is a divide-by-two of the input clock. If PTB0 is high with V_{TST} applied to $\overline{IRQ1}$ upon monitor mode entry, the bus frequency will be a divide-by-four of the input clock. Holding the PTB0 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 $\overline{IRQ1}$. 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 $\overline{IRQ1}$ (above condition set 2), then all port A pin requirements and conditions, including the PTB0 frequency divisor selection, are not in effect. This is to reduce circuit requirements when performing in-circuit programming.

NOTE

If the reset vector is blank and monitor mode is entered, the chip will see an additional reset cycle after the initial POR reset. Once the part has been programmed, the traditional method of applying a voltage, V_{TST}, to $\overline{IRQ1}$ must be used to enter monitor mode.

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), the COP is always disabled regardless of the state of $\overline{IRQ1}$ or $\overline{RST}$.
- If monitor mode was entered with V_{TST} on $\overline{IRQ1}$ (condition set 1), then the COP is disabled as long as V_{TST} is applied to either $\overline{IRQ1}$ or $\overline{RST}$.

The second condition states that as long as V_{TST} is maintained on the $\overline{IRQ1}$ 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 $\overline{IRQ1}$), 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 $\overline{IRQ1}$ pin in the interest of freeing the $\overline{IRQ1}$ for normal functionality in monitor mode.

Figure 8-2 shows a simplified diagram of the monitor mode entry when the reset vector is blank and just V_{DD} voltage is applied to the $\overline{IRQ1}$ pin. An external oscillator of 9.8304 MHz is required for a baud rate of 9600, as the internal bus frequency is automatically set to the external frequency divided by four.

Enter monitor mode with pin configuration shown in Figure 8-1 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.

Once out of reset, the MCU waits for the host to send eight security bytes. (See 8.4 Security.) After the security bytes, the MCU sends a break signal (10 consecutive logic 0's) to the host, indicating that it is ready to receive a command.

Table 8-1. Monitor Mode Signal Requirements and Options

$\overline{IRQ1}$	$\overline{RST}$	Address $FFFE/ $FFFF	PTA2	PTA1	PTA0	PTB0	External Clock(1)	Bus Frequency	PLL	COP	Baud Rate	Comment
X	GND	X	X	X	X	X	X	0	X	Disabled	0	No operation until reset goes high
V_{TST}(2)	V_{DD} or V_{TST}	X	0	1	1	0	4.9152 MHz	2.4576 MHz	OFF	Disabled	9600	PTA1 and PTA2 voltages only required if $\overline{IRQ1}$ = V_{TST}; PTB0 determines frequency divider
V_{TST}(2)	V_{DD} or V_{TST}	X	0	1	1	1	9.8304 MHz	2.4576 MHz	OFF	Disabled	9600	PTA1 and PTA2 voltages only required if $\overline{IRQ1}$ = V_{TST}; PTB0 determines frequency divider
V_{DD}	V_{DD}	Blank "$FFFF"	X	X	1	X	4.9152 MHz	1.2288 MHz	OFF	Disabled	4800	External frequency always divided by 4
V_{DD}	V_{DD}	Blank "$FFFF"	X	X	1	X	9.8304 MHz	2.4576 MHz	OFF	Disabled	9600	External frequency always divided by 4
V_{DD} or GND	V_{TST}	Blank "$FFFF"	X	X	X	X	X	—	OFF	Enabled	—	Enters user mode — will encounter an illegal address reset
V_{DD} or GND	V_{DD} or V_{TST}	Not Blank	X	X	X	X	X	—	OFF	Enabled	—	Enters user mode

1. External clock is derived by a 4.9152 MHz crystal/off-chip oscillator or a 9.8304 MHz off-chip oscillator.
2. Monitor mode entry by $\overline{IRQ1}$ = V_{TST}, a 4.9152/9.8304 MHz off-chip oscillator must be used. The MCU internal crystal oscillator circuit is bypassed.

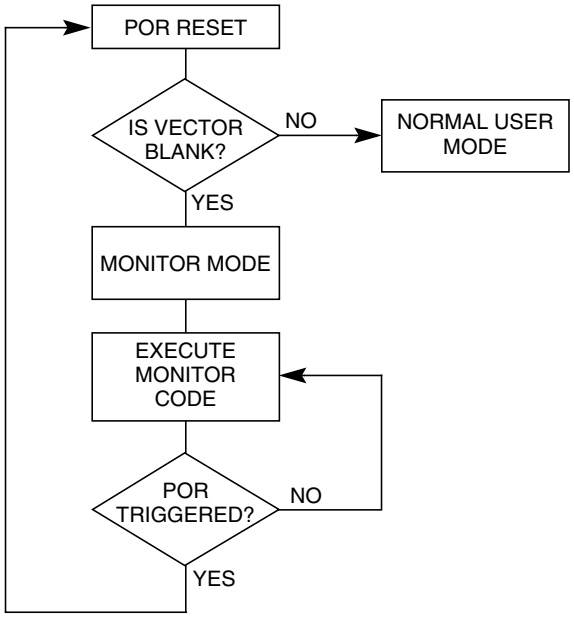


Figure 8-2. Low-Voltage Monitor Mode Entry Flowchart

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 8-2 summarizes the differences between user mode and monitor mode vectors.

Table 8-2. 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

8.3.2 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.

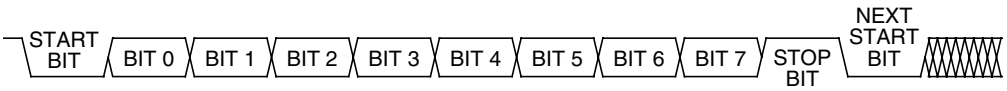


Figure 8-3. Monitor Data Format

8.3.3 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.

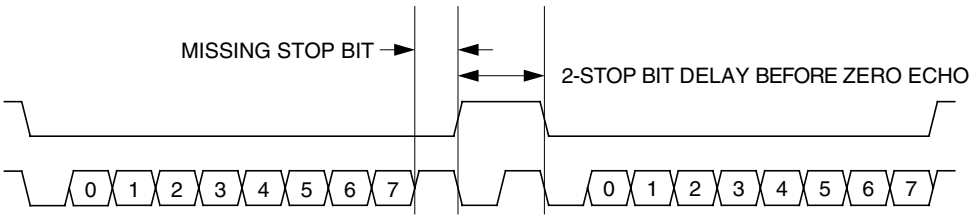


Figure 8-4. Break Transaction

8.3.4 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 8-3 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 8-3. 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

8.3.5 Commands

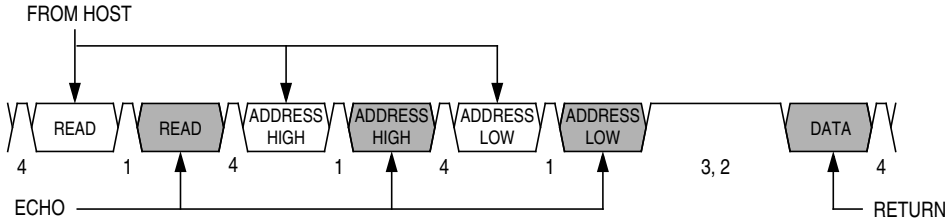
The monitor ROM firmware uses these commands:

- 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.

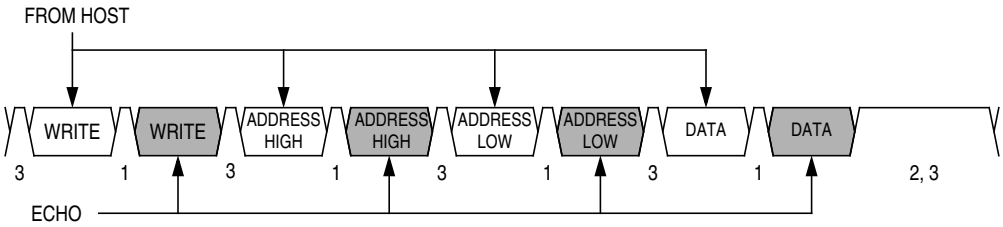
NOTE

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



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

Figure 8-5. Read Transaction



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

Figure 8-6. Write Transaction

A brief description of each monitor mode command is given in Table 8-4 through Table 8-9.

Table 8-4. 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

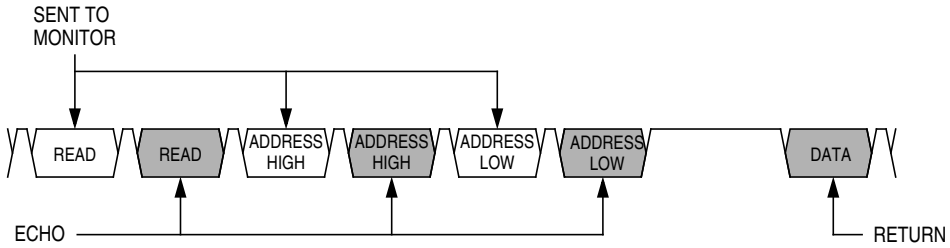


Table 8-5. WRITE (Write Memory) Command

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

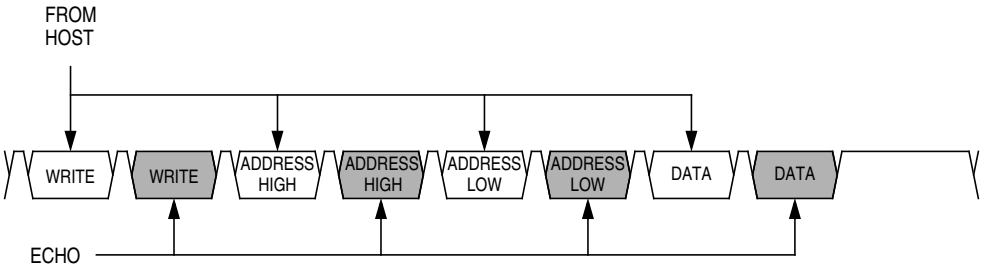


Table 8-6. IREAD (Indexed Read) Command

Description	Read next 2 bytes in memory from last address accessed
Operand	None
Data Returned	Returns contents of next two addresses
Opcode	$1A

Command Sequence

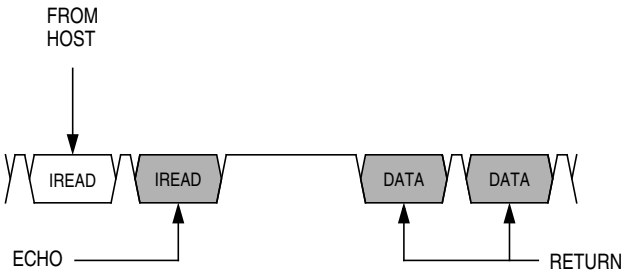
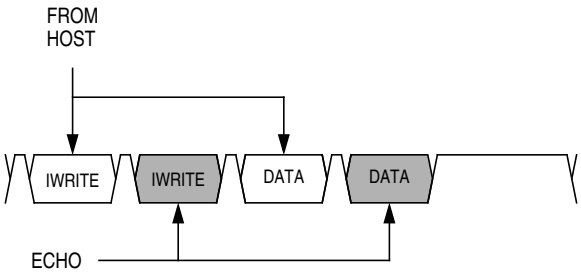


Table 8-7. IWRITE (Indexed Write) Command

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 8-8. 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	

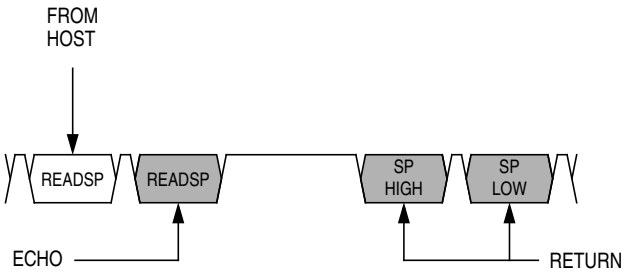
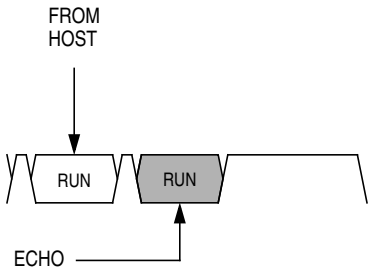


Table 8-9. RUN (Run User Program) Command

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.

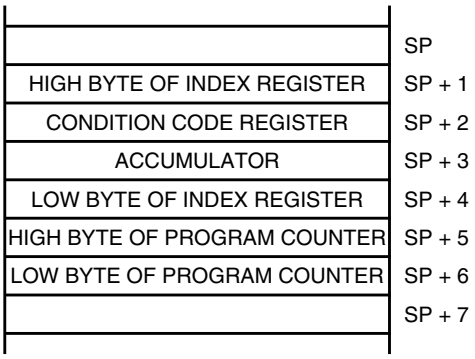


Figure 8-7. Stack Pointer at Monitor Mode Entry

8.4 Security

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

NOTE

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

During monitor mode entry, the MCU waits after the power-on reset for the host to send the eight security bytes on pin PTA0. If the received bytes match those at locations $FFF6–$FFFD, the host bypasses the security feature and can read all FLASH locations and execute code from FLASH. Security remains bypassed until a power-on reset occurs. If the reset was not a power-on reset, security remains bypassed and security code entry is not required. (See Figure 8-8.)

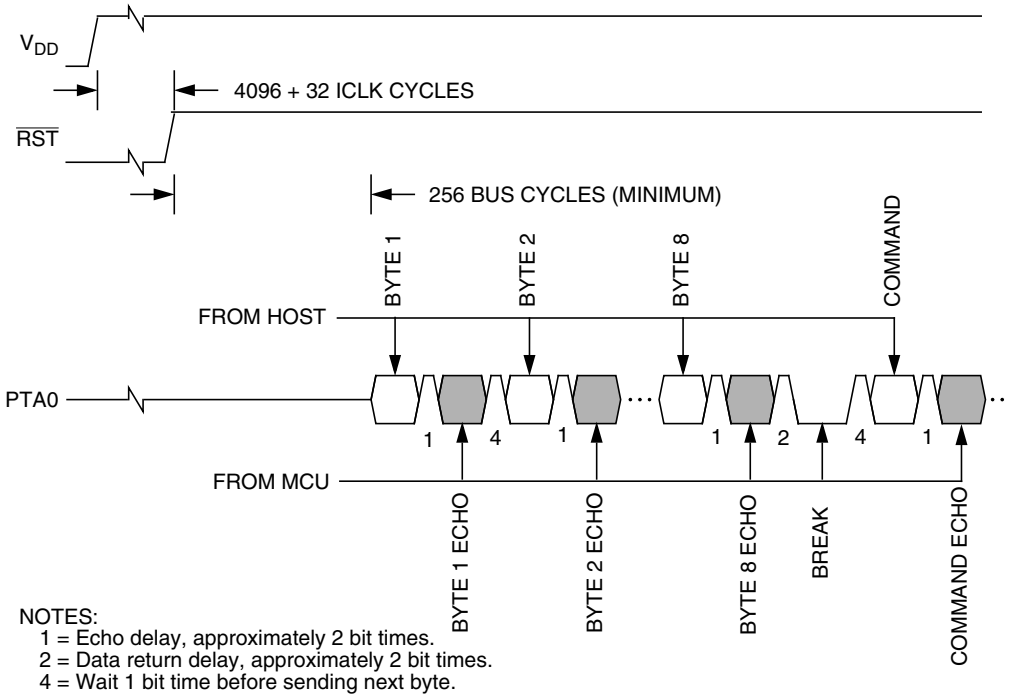


Figure 8-8. Monitor Mode Entry Timing

Upon power-on reset, if the received bytes of the security code do not match the data at locations $FFF6–$FFFD, the host fails to bypass the security feature. The MCU remains in monitor mode, but reading a FLASH location returns an invalid value and trying to execute code from FLASH causes an illegal address reset. After receiving the eight security bytes from the host, the MCU transmits a break character, signifying that it is ready to receive a command.

NOTE

The MCU does not transmit a break character until after the host sends the eight security 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 FLASH can be accessed.

If the security sequence fails, the device should be reset by a power-on reset and brought up in monitor mode to attempt another entry. After failing the security sequence, the FLASH module can also be mass erased by executing an erase routine that was downloaded into internal RAM. The mass erase operation clears the security code locations so that all eight security bytes become $FF (blank).

8.5 ROM-Resident Routines

Seven routines stored in the monitor ROM area (thus ROM-resident) are provided for FLASH memory manipulation. Five of the seven routines are intended to simplify FLASH program, erase, and load operations. The other two routines are intended to simplify the use of the FLASH memory as EEPROM. Table 8-10 shows a summary of the ROM-resident routines.

Table 8-10. Summary of ROM-Resident Routines

Routine Name	Routine Description	Call Address	Stack Used (bytes)
PRGRNGE	Program a range of locations	$FC34	15
ERARNGE	Erase a page or the entire array	$FCE4	9
LDRNGE	Loads data from a range of locations	$FC00	7
MON_PRGRNGE	Program a range of locations in monitor mode	$FF24	17
MON_ERARNGE	Erase a page or the entire array in monitor mode	$FF28	11

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 8-9.

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.

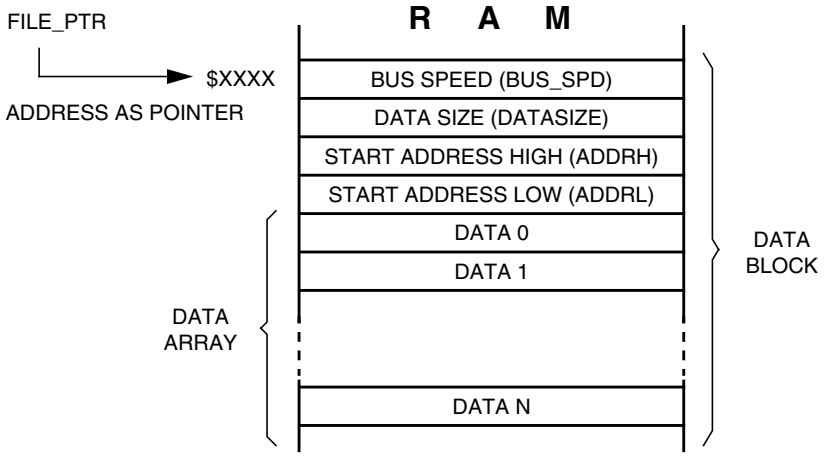


Figure 8-9. Data Block Format for ROM-Resident Routines

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. ERARNGE and MON_ERARNGE routines 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 and MON_PRGRNGE. For the read routine, LDRNGE, data is read from FLASH and stored in this array.

8.5.1 PRGRNGE

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

Table 8-11. PRGRNGE Routine

Routine Name	PRGRNGE
Routine Description	Program a range of locations
Calling Address	$FC34
Stack Used	15 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      $FC34
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
```

8.5.2 ERARNGE

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

Table 8-12. ERARNGE Routine

Routine Name	ERARNGE
Routine Description	Erase a page or the entire array
Calling Address	$FCE4
Stack Used	9 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 8.5.1 PRGRNGE).

```
ERARNGE         EQU      $FCE4
MAIN:
        BSR     INITIALISATION
        :
        :
        LDHX    #FILE_PTR
        JSR     ERARNGE
        :
```

8.5.3 LDRNGE

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

Table 8-13. LDRNGE Routine

Routine Name	LDRNGE
Routine Description	Loads data from a range of locations
Calling Address	$FC00
Stack Used	7 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 8.5.1 PRGRNGE).

```
LDRNGE          EQU      $FC00
MAIN:
        BSR     INITIALIZATION
        :
        :
        LDHX    #FILE_PTR
        JSR     LDRNGE
        :
```

8.5.4 MON_PRGRNGE

In monitor mode, MON_PRGRNGE is used to program a range of FLASH locations with data loaded into the data array.

Table 8-14. MON_PRGRNGE Routine

Routine Name	MON_PRGRNGE
Routine Description	Program a range of locations, in monitor mode
Calling Address	$FF24
Stack Used	17 bytes
Data Block Format	Bus speed Data size Starting address (high byte) Starting address (low byte) Data 1 : Data N

The MON_PRGRNGE routine is designed to be used in monitor mode. It performs the same function as the PRGRNGE routine (see 8.5.1 PRGRNGE), except that MON_PRGRNGE returns to the main program via an SWI instruction. After a MON_PRGRNGE call, the SWI instruction will return the control back to the monitor code.

8.5.5 MON_ERARNGE

In monitor mode, ERARNGE is used to erase a range of locations in FLASH.

Table 8-15. MON_ERARNGE Routine

Routine Name	MON_ERARNGE
Routine Description	Erase a page or the entire array, in monitor mode
Calling Address	$FF28
Stack Used	11 bytes
Data Block Format	Bus speed Data size Starting address (high byte) Starting address (low byte)

The MON_ERARNGE routine is designed to be used in monitor mode. It performs the same function as the ERARNGE routine (see 8.5.2 ERARNGE), except that MON_ERARNGE returns to the main program via an SWI instruction. After a MON_ERARNGE call, the SWI instruction will return the control back to the monitor code.

Chapter 9
Timer Interface Module (TIM)

9.1 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 9-1 is a block diagram of the TIM.

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

9.2 Features

Features of the TIM include:

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

9.3 Pin Name Conventions

The text that follows describes both timers, TIM1 and TIM2. The TIM input/output (I/O) pin names are T[1,2]CH0 (timer channel 0) and T[1,2]CH1 (timer channel 1), where “1” is used to indicate TIM1 and “2” is used to indicate TIM2. The two TIMs share four I/O pins with four I/O port pins. The external clock input for TIM2 is shared with the an ADC channel pin. The full names of the TIM I/O pins are listed in Table 9-1. The generic pin names appear in the text that follows.

Table 9-1. Pin Name Conventions

TIM Generic Pin Names:		T[1,2]CH0	T[1,2]CH1
Full TIM Pin Names:	TIM1	PTB4/T1CH0	PTB5/T1CH1
	TIM2	PTB6/T2CH0	PTB7/T2CH1

NOTE

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

9.4 Functional Description

Figure 9-1 shows the structure of the TIM. The central component of the TIM is the 16-bit TIM counter that can operate as a free-running counter or a modulo up-counter. The TIM counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH:TMODL, control the modulo value of the TIM counter. Software can read the TIM counter value at any time without affecting the counting sequence.

The two TIM channels (per timer) are programmable independently as input capture or output compare channels.

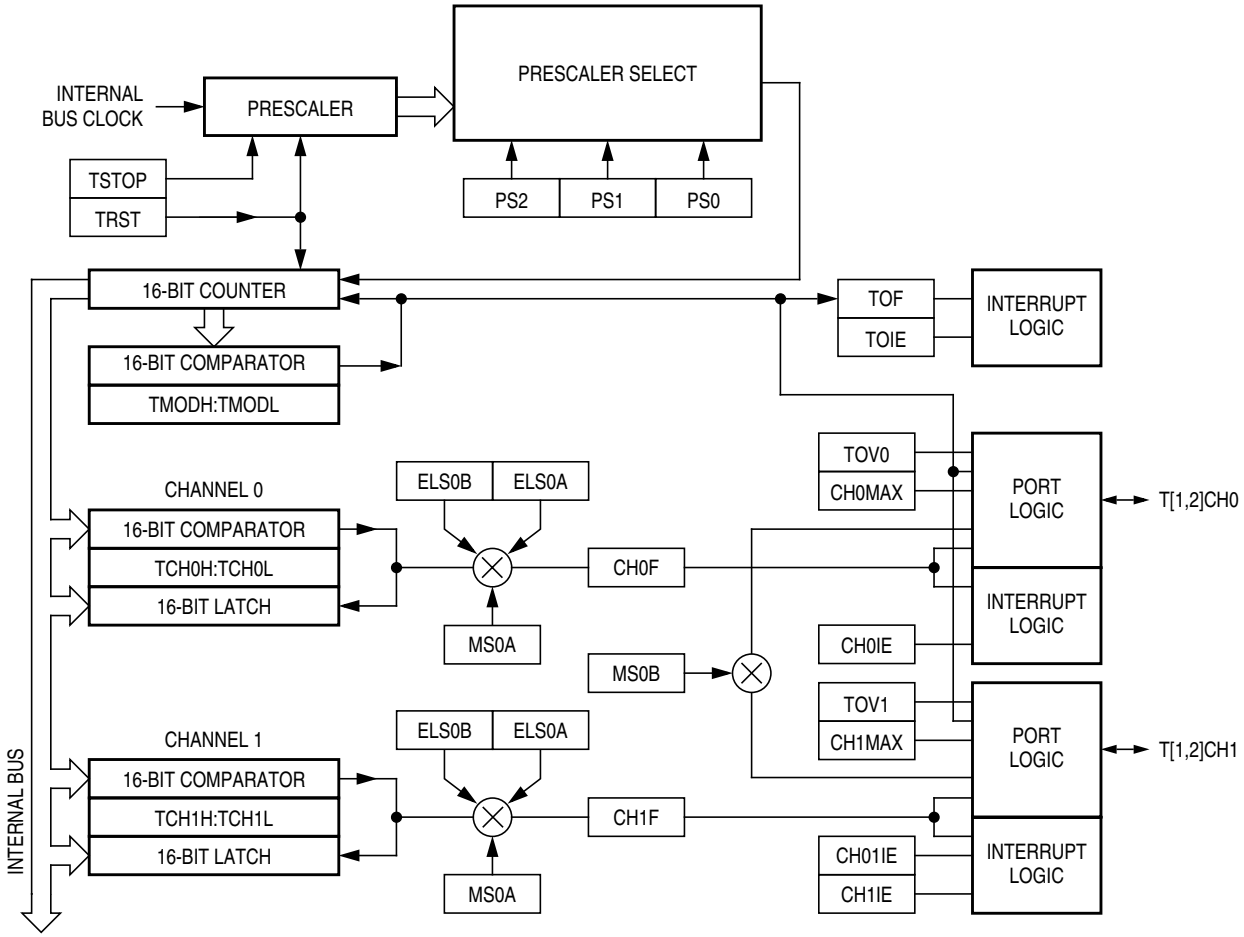


Figure 9-1. TIM Block Diagram

Figure 9-2 summarizes the timer registers.

NOTE

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

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

= Unimplemented

Figure 9-2. TIM I/O Register Summary (Sheet 1 of 2)

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

= Unimplemented

Figure 9-2. TIM I/O Register Summary (Sheet 2 of 2)

9.4.1 TIM Counter Prescaler

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

9.4.2 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.

9.4.3 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.

9.4.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 9.4.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

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

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

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

9.4.3.2 Buffered Output Compare

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

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

NOTE

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

9.4.4 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 9-3 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 polarity of the PWM pulse is 1. Program the TIM to set the pin if the polarity of the PWM pulse is 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 9.9.1 TIM Status and Control Register.

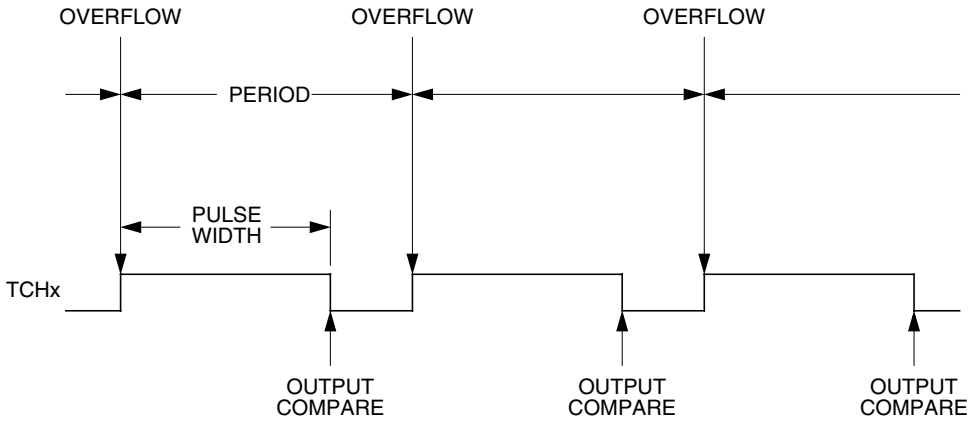


Figure 9-3. PWM Period and Pulse Width

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%.

9.4.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 9.4.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

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

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

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

NOTE

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

9.4.4.2 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.

9.4.4.3 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 9-3.)
   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 9-3.)

   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 9.9.4 TIM Channel Status and Control Registers.)

9.5 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.

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 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.

9.6.2 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.

9.7 TIM During Break Interrupts

A break interrupt stops the TIM counter and inhibits input captures.

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

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

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

9.8 I/O Signals

Port B shares four of its pins with the TIM. The four TIM channel I/O pins are T1CH0, T1CH1, T2CH0, and T2CH1 as described in 9.3 Pin Name Conventions.

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

9.9 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)

9.9.1 TIM Status and Control Register

The TIM status and control register (TSC):

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

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

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

= Unimplemented

Figure 9-4. 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. Also, when the TSTOP bit is set and the timer is configured for input capture operation, input captures are inhibited until the TSTOP bit is cleared.

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 9-2 shows. Reset clears the PS[2:0] bits.

Table 9-2. Prescaler Selection

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

9.9.2 TIM Counter Registers

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

NOTE

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

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

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

Figure 9-5. TIM Counter Registers High (TCNTH)

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

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

Figure 9-6. TIM Counter Registers Low (TCNTL)

9.9.3 TIM Counter Modulo Registers

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

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

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

Figure 9-7. TIM Counter Modulo Register High (TMODH)

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

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

Figure 9-8. TIM Counter Modulo Register Low (TMODL)

NOTE

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

9.9.4 TIM Channel Status and Control Registers

Each of the TIM channel status and control registers:

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

Address: T1SC0, $0025 and T2SC0, $0030

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

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

Address: T1SC1, $0028 and T2SC1, $0033

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

Figure 9-10. 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 9-3.

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 9-3. 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 9-3 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

Table 9-3. 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	00	Output compare or PWM	Software compare only
01	01		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

After initially enabling a TIM channel register for input capture operation, and selecting the edge sensitivity, clear CHxF to ignore any erroneous detection flags.

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 9-11 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.

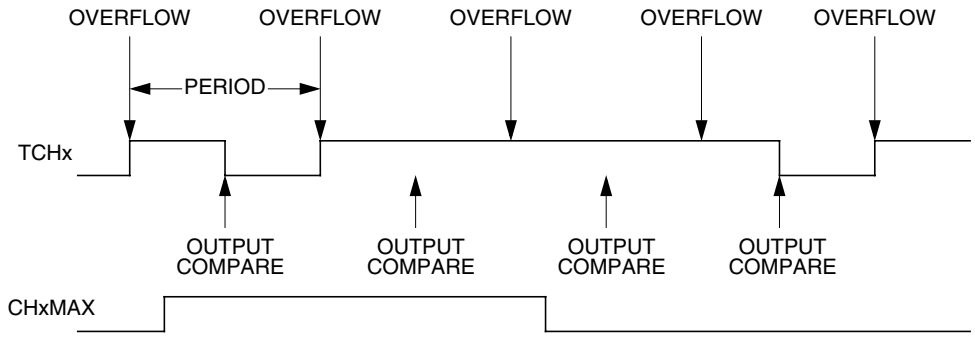


Figure 9-11. CHxMAX Latency

9.9.5 TIM Channel Registers

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

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

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

Address: T1CH0H, $0026 and T2CH0H, $0031

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

Figure 9-12. TIM Channel 0 Register High (TCH0H)

Address: T1CH0L, $0027 and T2CH0L $0032

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

Figure 9-13. TIM Channel 0 Register Low (TCH0L)

Address: T1CH1H, $0029 and T2CH1H, $0034

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

Figure 9-14. TIM Channel 1 Register High (TCH1H)

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

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

Figure 9-15. TIM Channel 1 Register Low (TCH1L)

Chapter 10
Timebase Module (TBM)

10.1 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.

10.2 Features

Features of the TBM module include:

- Eight user selectable periodic interrupts
- User selectable oscillator clock source enable during stop mode to allow periodic wake-up from stop

10.3 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 10-1, starts counting when the TBON bit is set. When the counter overflows at the tap selected by TBR[2:0], 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.

The reference clock OSCCLK is derived from the oscillator module, see 5.2.2 TBM Reference Clock Selection.

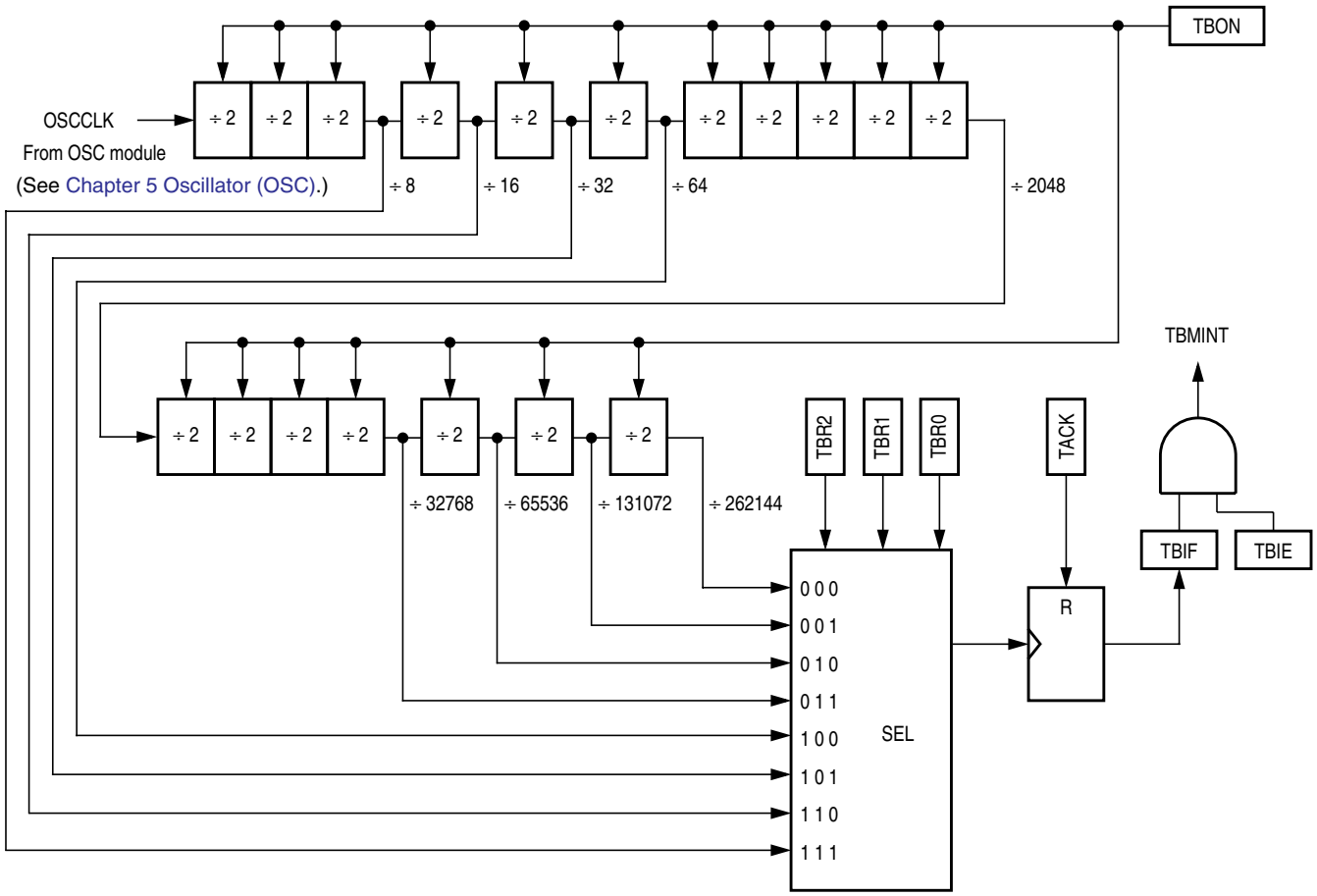


Figure 10-1. Timebase Block Diagram

10.4 Timebase Register Description

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

Address: $0051

	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 10-2. 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

TBR[2:0] — Timebase Rate Selection

These read/write bits are used to select the rate of timebase interrupts as shown in Table 10-1.

NOTE

Do not change TBR[2:0] bits while the timebase is enabled (TBON = 1).

Table 10-1. Timebase Rate Selection for OSCCLK = 4 MHz

TBR2	TBR1	TBR0	Divider	Timebase Interrupt Rate[1]	
				Hz	ms
0	0	0	262144	15.259	65.536
0	0	1	131072	30.518	32.768
0	1	0	65536	61.035	16.384
0	1	1	32768	122.07	8.192
1	0	0	64	62,500	0.016
1	0	1	32	125,000	0.008
1	1	0	16	250,000	0.004
1	1	1	8	500,000	0.002

1. If user selects one of the shaded timebase settings, ensure that the operating bus frequency is fast enough to service the periodic interrupts.

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 0’s

10.5 Interrupts

The timebase module can interrupt the CPU on a regular basis with a rate defined by TBR[2:0]. 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 2-1 . Vector Addresses.

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

10.6 Low-Power Modes

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

10.6.1 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.

10.6.2 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_ICLKDIS, STOP_RCLKEN, or STOP_XCLKEN) 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 11
Serial Communications Interface Module (SCI)

11.1 Introduction

The MC68HC908AP64A has two SCI modules:

- SCI1 is a standard SCI module, and
- SCI2 is an infrared SCI module.

This section describes SCI1, the serial communications interface (SCI) module, which allows high-speed asynchronous communications with peripheral devices and other MCUs.

NOTE

When the SCI is enabled, the TxD pin is an open-drain output and requires a pullup resistor to be connected for proper SCI operation.

11.2 Features

Features of the SCI module include the following:

- Full-duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- 32 programmable baud rates
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter CPU interrupt requests
- Programmable transmitter output polarity
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Interrupt-driven operation with eight interrupt flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection
- Configuration register bit, SCIBDSRC, to allow selection of baud rate clock source

11.3 Pin Name Conventions

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

- RxD (receive data)
- TxD (transmit data)

SCI I/O (input/output) lines are implemented by sharing parallel I/O port pins. The full name of an SCI input or output reflects the name of the shared port pin. Table 11-1 shows the full names and the generic names of the SCI I/O pins. The generic pin names appear in the text of this section.

Table 11-1. Pin Name Conventions

Generic Pin Names:	RxD	TxD
Full Pin Names:	PTB3/RxD	PTB2/TxD

NOTE

When the SCI is enabled, the TxD pin is an open-drain output and requires a pullup resistor to be connected for proper SCI operation.

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

= Unimplemented R = Reserved U = Unaffected

Figure 11-1. SCI I/O Register Summary

11.4 Functional Description

Figure 11-2 shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication among the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

The baud rate clock source for the SCI can be selected via the configuration bit, SCIBDSRC, of the CONFIG2 register ($001D).

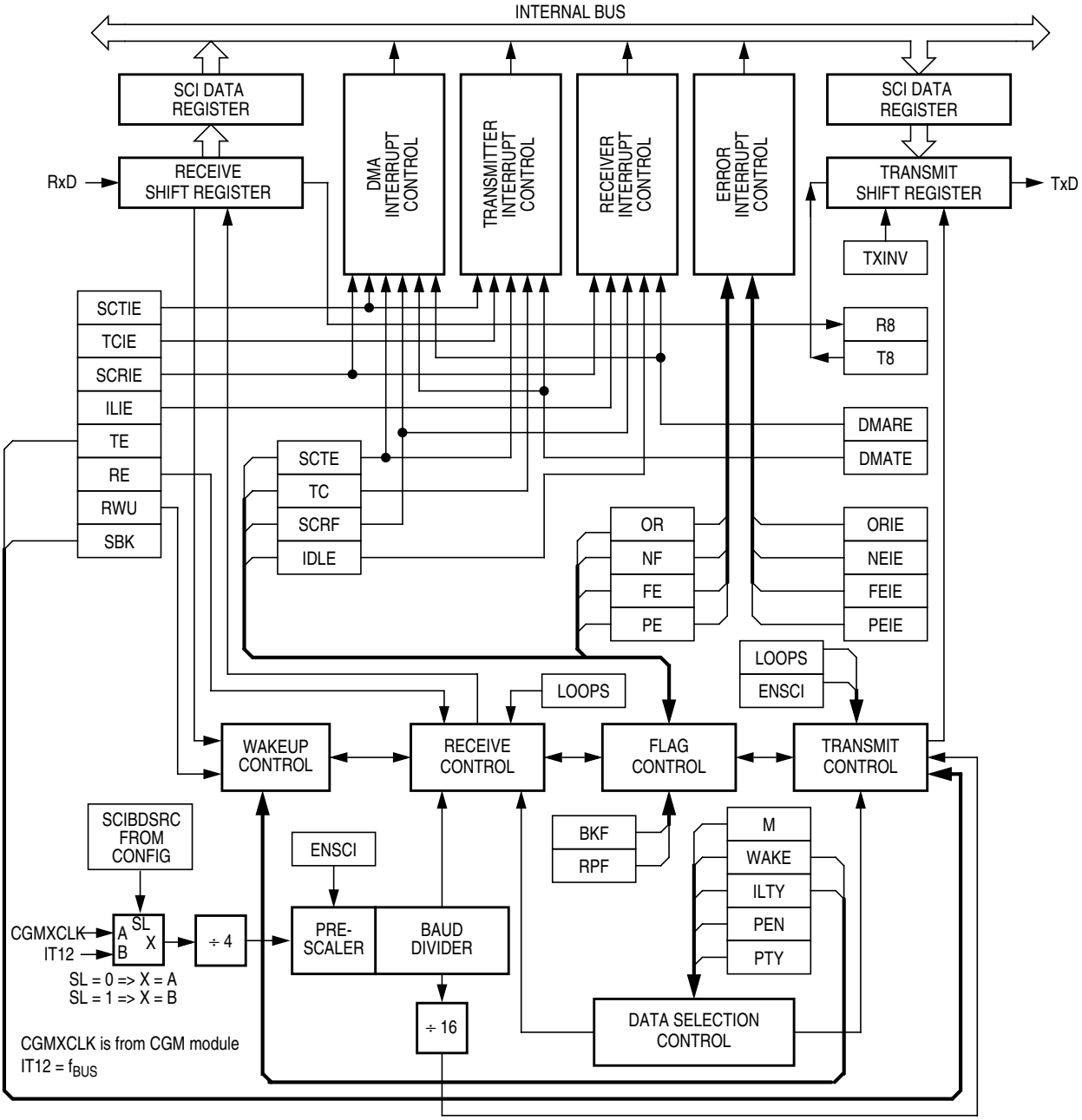


Figure 11-2. SCI Module Block Diagram

11.4.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 11-3.

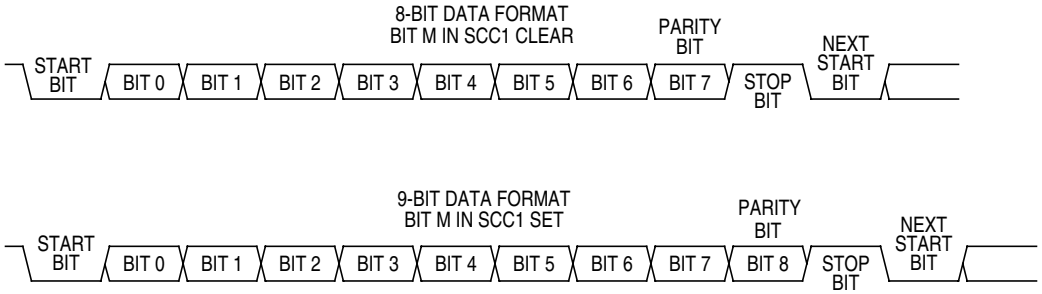


Figure 11-3. SCI Data Formats

11.4.2 Transmitter

Figure 11-4 shows the structure of the SCI transmitter.

The baud rate clock source for the SCI can be selected via the configuration bit, SCIBDSRC. Source selection values are shown in Figure 11-4.

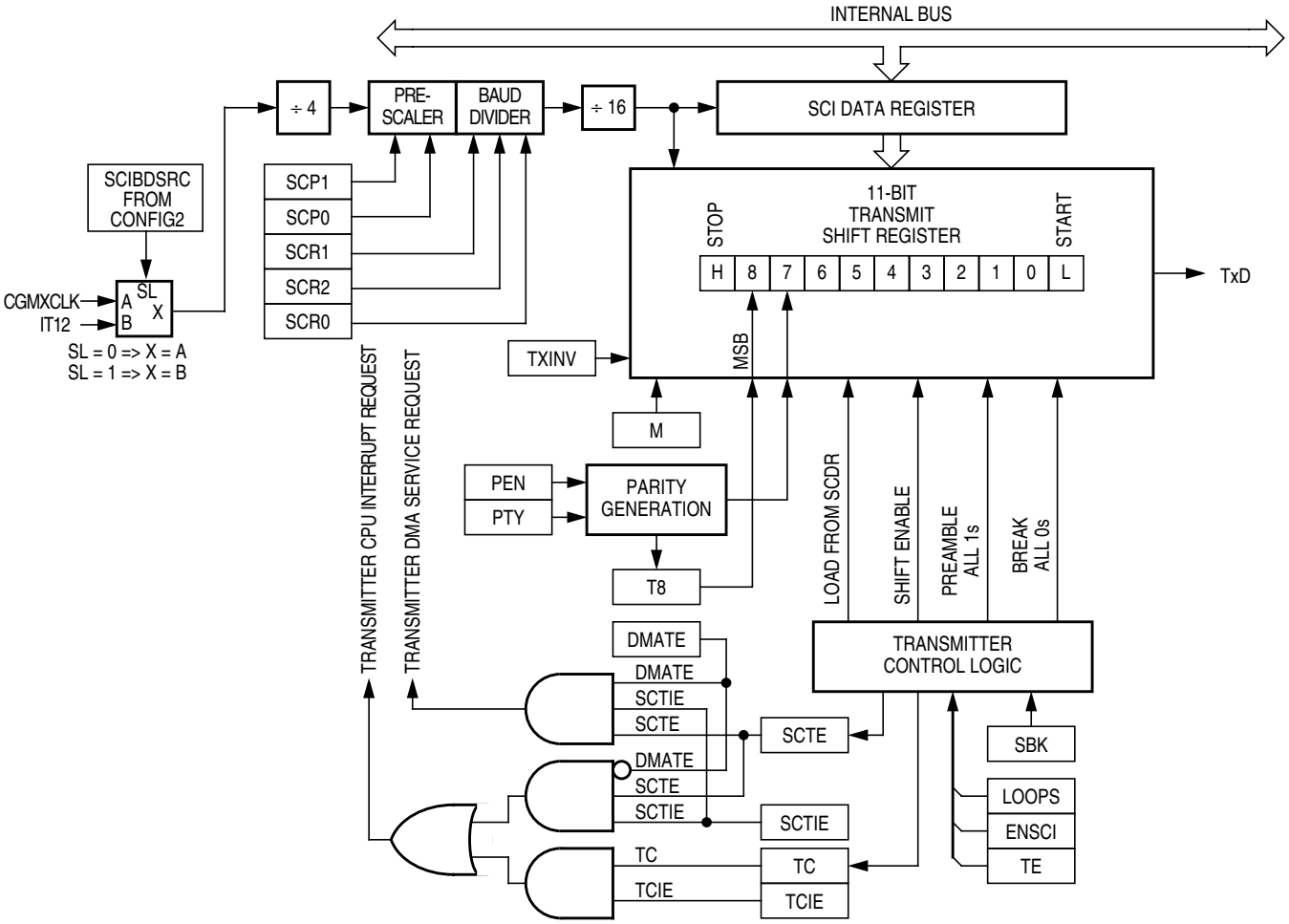


Figure 11-4. SCI Transmitter

11.4.2.1 Character Length

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When transmitting 9-bit data, bit T8 in SCI control register 3 (SCC3) is the ninth bit (bit 8).

11.4.2.2 Character Transmission

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

1. Enable the SCI by writing a logic 1 to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2).
3. Clear the SCI transmitter empty bit by first reading SCI status register 1 (SCS1) and then writing to the SCDR.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of logic 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

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

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

11.4.2.3 Break Characters

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

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

Receiving a break character has these effects on SCI registers:

- Sets the framing error bit (FE) in SCS1
- Sets the SCI receiver full bit (SCRF) in SCS1
- Clears the SCI data register (SCDR)
- Clears the R8 bit in SCC3
- Sets the break flag bit (BKF) in SCS2
- May set the overrun (OR), noise flag (NF), parity error (PE), or reception in progress flag (RPF) bits

11.4.2.4 Idle Characters

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

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

NOTE

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current character shifts out to the TxD pin. Setting TE after the stop bit appears on TxD causes data previously written to the SCDR to be lost.

Toggle the TE bit for a queued idle character when the SCTE bit becomes set and just before writing the next byte to the SCDR.

11.4.2.5 Inversion of Transmitted Output

The transmit inversion bit (TXINV) in SCI control register 1 (SCC1) reverses the polarity of transmitted data. All transmitted values, including idle, break, start, and stop bits, are inverted when TXINV is at logic 1. (See 11.8.1 SCI Control Register 1.)

11.4.2.6 Transmitter Interrupts

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

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

11.4.3 Receiver

Figure 11-5 shows the structure of the SCI receiver.

11.4.3.1 Character Length

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in SCI control register 1 (SCC1) determines character length. When receiving 9-bit data, bit R8 in SCI control register 2 (SCC2) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

11.4.3.2 Character Reception

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

After a complete character shifts into the receive shift register, the data portion of the character transfers to the SCDR. The SCI receiver full bit, SCRF, in SCI status register 1 (SCS1) becomes set, indicating that

the received byte can be read. If the SCI receive interrupt enable bit, SCRIE, in SCC2 is also set, the SCRF bit generates a receiver CPU interrupt request.

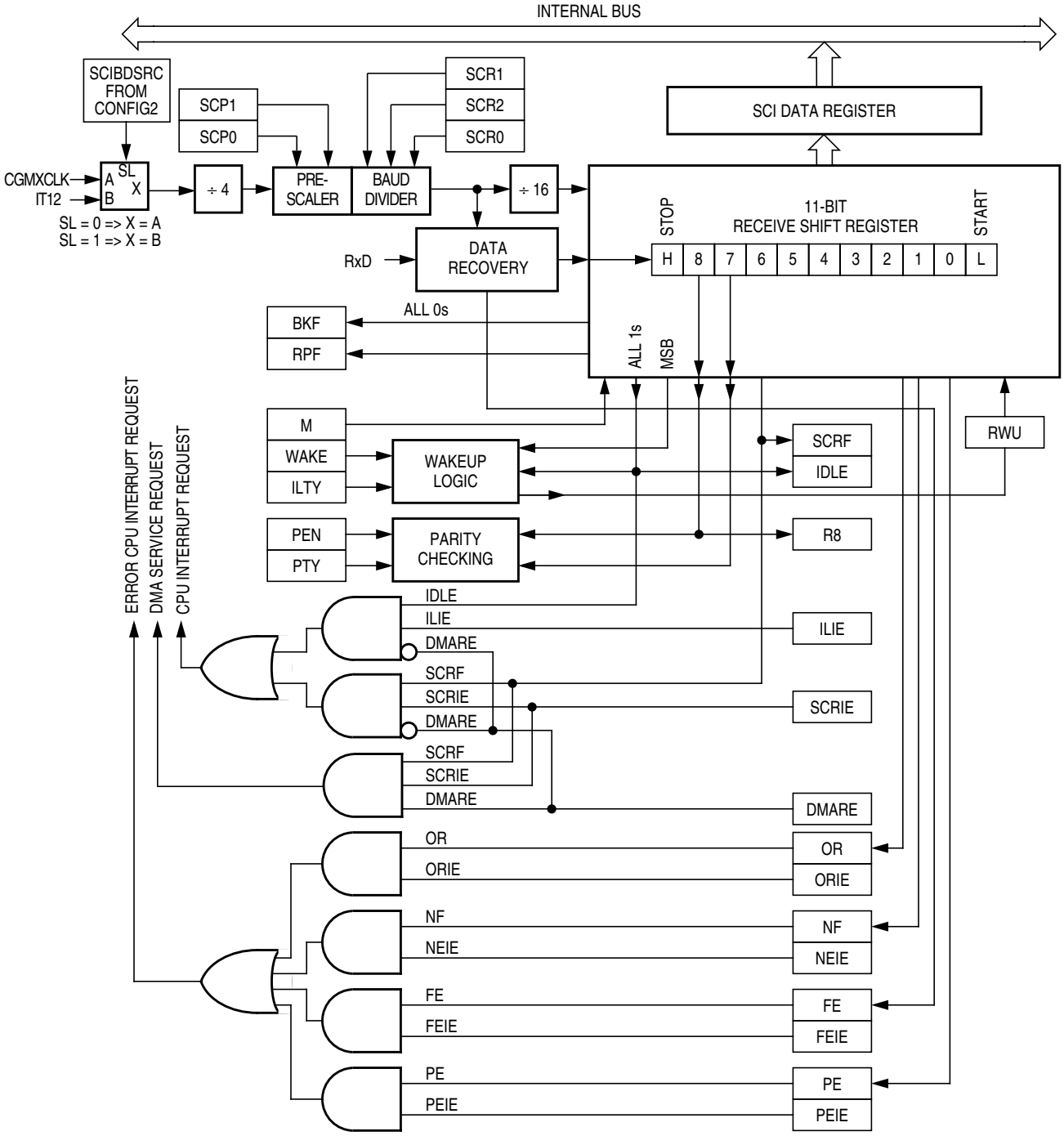


Figure 11-5. SCI Receiver Block Diagram

11.4.3.3 Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see Figure 11-6):

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

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

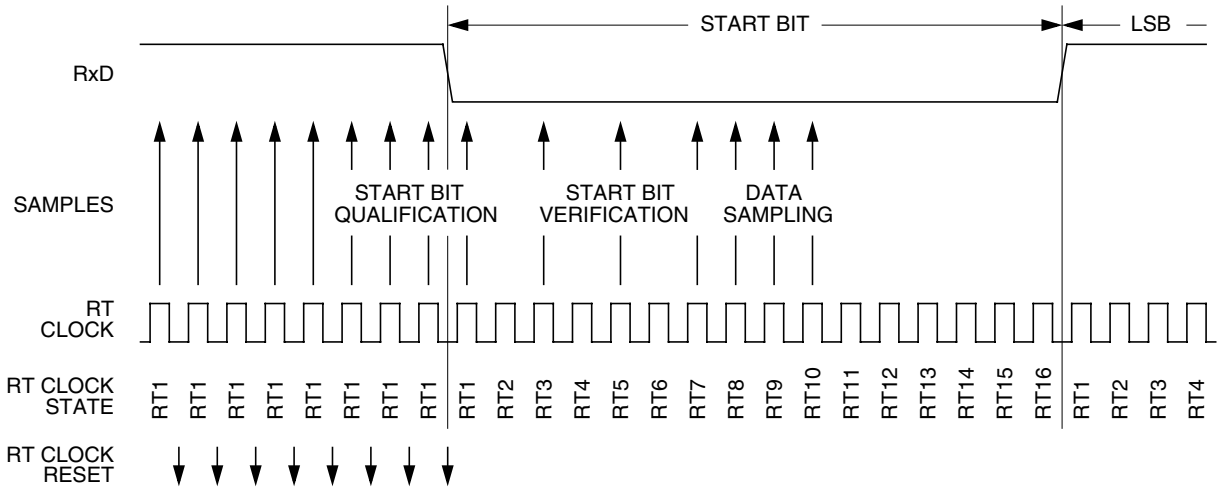


Figure 11-6. Receiver Data Sampling

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 11-2 summarizes the results of the start bit verification samples.

Table 11-2. Start Bit Verification

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

Start bit verification is not successful if any two of the three verification samples are logic 1s. If start bit verification is not successful, the RT clock is reset and a new search for a start bit begins.

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 11-3 summarizes the results of the data bit samples.

Table 11-3. Data Bit Recovery

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

NOTE

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

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 11-4 summarizes the results of the stop bit samples.

Table 11-4. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

11.4.3.4 Framing Errors

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

11.4.3.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming character, it resynchronizes the RT clock on any valid falling edge within the character. Resynchronization within characters corrects misalignments between transmitter bit times and receiver bit times.

Slow Data Tolerance

Figure 11-7 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

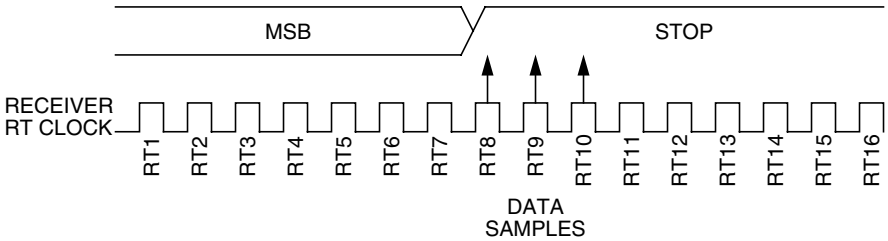


Figure 11-7. Slow Data

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 11-7, the receiver counts 154 RT cycles at the point when the count of the transmitting device is
9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit character with no errors is

$$\left|\frac{154 - 147}{154}\right| \times 100 = 4.54\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver
10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 11-7, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is

$$\left|\frac{170 - 163}{170}\right| \times 100 = 4.12\%$$

Fast Data Tolerance

Figure 11-8 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

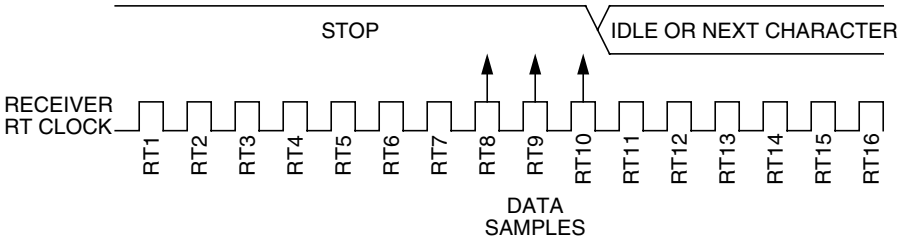


Figure 11-8. Fast Data

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 11-8, the receiver counts 154 RT cycles at the point when the count of the transmitting device is
10 bit times × 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.9\dot{0}\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver
10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 11-8, the receiver counts 170 RT cycles at the point when the count of the transmitting device is
11 bit times × 16 RT cycles = 176 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is

$$\left|\frac{170 - 176}{170}\right| \times 100 = 3.53\%$$

11.4.3.6 Receiver Wakeup

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in SCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

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

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

NOTE

With the WAKE bit clear, setting the RWU bit after the RxD pin has been idle may cause the receiver to wake up immediately.

11.4.3.7 Receiver Interrupts

The following sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in SCS1 indicates that the receive shift register has transferred a character to the SCDR. SCRF can generate a receiver CPU interrupt request. Setting the SCI receive interrupt enable bit, SCRIE, in SCC2 enables the SCRF bit to generate receiver CPU interrupts.
- Idle input (IDLE) — The IDLE bit in SCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in SCC2 enables the IDLE bit to generate CPU interrupt requests.

11.4.3.8 Error Interrupts

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

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the SCDR. The previous character remains in the SCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in SCC3 enables OR to generate SCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the SCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in SCC3 enables NF to generate SCI error CPU interrupt requests.
- Framing error (FE) — The FE bit in SCS1 is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in SCC3 enables FE to generate SCI error CPU interrupt requests.
- Parity error (PE) — The PE bit in SCS1 is set when the SCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, in SCC3 enables PE to generate SCI error CPU interrupt requests.

11.5 Low-Power Modes

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

11.5.1 Wait Mode

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

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

Refer to 7.6 Low-Power Modes for information on exiting wait mode.

11.5.2 Stop Mode

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

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

Refer to 7.6 Low-Power Modes for information on exiting stop mode.

11.6 SCI During Break Module 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.

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.

11.7 I/O Signals

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

- PTB2/TxD — Transmit data
- PTB3/RxD — Receive data

11.7.1 TxD (Transmit Data)

When the SCI is enabled (ENSCI=1), the PTB2/TxD pin becomes the serial data output, TxD, from the SCI transmitter regardless of the state of the DDRB2 bit in data direction register B (DDRB). The TxD pin is an open-drain output and requires a pullup resistor to be connected for proper SCI operation.

NOTE

The PTB2/TxD pin is an open-drain pin when configured as an output. Therefore, when configured as a general purpose output pin (PTB2), a pullup resistor must be connected to this pin.

11.7.2 RxD (Receive Data)

When the SCI is enabled (ENSCI=1), the PTB3/RxD pin becomes the serial data input, RxD, to the SCI receiver regardless of the state of the DDRB3 bit in data direction register B (DDRB).

NOTE

The PTB3/RxD pin is an open-drain pin when configured as an output. Therefore, when configured as a general purpose output pin (PTB3), a pullup resistor must be connected to this pin.

11.8 I/O Registers

These I/O registers control and monitor SCI operation:

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

11.8.1 SCI Control Register 1

SCI control register 1:

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

Address: $0013

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

Figure 11-9. SCI Control Register 1 (SCC1)

LOOPS — Loop Mode Select Bit

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

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI Bit

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

1 = SCI enabled
0 = SCI disabled

TXINV — Transmit Inversion Bit

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

NOTE

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

M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are eight or nine bits long. (See Table 11-5.) The ninth bit can serve as an extra stop bit, as a receiver wakeup signal, or as a parity bit. Reset clears the M bit.

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

WAKE — Wakeup Condition Bit

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

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

This read/write bit determines when the SCI starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. Reset clears the ILTY bit.

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

PEN — Parity Enable Bit

This read/write bit enables the SCI parity function. (See Table 11-5.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See Figure 11-3.) Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See Table 11-5.) Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

NOTE

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

Table 11-5. Character Format Selection

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

11.8.2 SCI Control Register 2

SCI control register 2:

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

Address: $0014

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

Figure 11-10. SCI Control Register 2 (SCC2)

SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests. Reset clears the SCTIE bit.

1 = SCTE enabled to generate CPU interrupt
0 = SCTE not enabled to generate CPU interrupt

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate SCI transmitter CPU interrupt requests. Reset clears the TCIE bit.

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

SCRIE — SCI Receive Interrupt Enable Bit

This read/write bit enables the SCRF bit to generate SCI receiver CPU interrupt requests. Reset clears the SCRIE bit.

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

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate SCI receiver CPU interrupt requests. Reset clears the ILIE bit.

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

TE — Transmitter Enable Bit

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

1 = Transmitter enabled
0 = Transmitter disabled

NOTE

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

RE — Receiver Enable Bit

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

1 = Receiver enabled
0 = Receiver disabled

NOTE

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

RWU — Receiver Wakeup Bit

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

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

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

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

NOTE

Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK before the preamble begins causes the SCI to send a break character instead of a preamble.

11.8.3 SCI Control Register 3

SCI control register 3:

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

Address: $0015

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

= Unimplemented U = Unaffected

Figure 11-11. SCI Control Register 3 (SCC3)

R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the SCDR receives the other 8 bits.

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

T8 — Transmitted Bit 8

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

DMARE — DMA Receive Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

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

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

DMATE — DMA Transfer Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

1 = SCTE DMA service requests enabled; SCTE CPU interrupt requests disabled

0 = SCTE DMA service requests disabled; SCTE CPU interrupt requests enabled

ORIE — Receiver Overrun Interrupt Enable Bit

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

1 = SCI error CPU interrupt requests from OR bit enabled

0 = SCI error CPU interrupt requests from OR bit disabled

NEIE — Receiver Noise Error Interrupt Enable Bit

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

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

FEIE — Receiver Framing Error Interrupt Enable Bit

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

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

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the parity error bit, PE. (See 11.8.4 SCI Status Register 1.) Reset clears PEIE.

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

11.8.4 SCI Status Register 1

SCI status register 1 (SCS1) contains flags to signal these conditions:

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

Address: $0016

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

= Unimplemented

Figure 11-12. SCI Status Register 1 (SCS1)

SCTE — SCI Transmitter Empty Bit

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

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

TC — Transmission Complete Bit

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an SCI transmitter CPU interrupt request if the TCIE bit in SCC2 is also set. TC is automatically cleared when data, preamble or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — SCI Receiver Full Bit

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

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

IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive logic 1s appear on the receiver input. IDLE generates an SCI receiver CPU interrupt request if the ILIE bit in SCC2 is also set. Clear the IDLE bit by reading SCS1 with IDLE set and then reading the SCDR. After the receiver is enabled, it must receive a valid character that sets the SCRF bit before an idle condition can set the IDLE bit. Also, after the IDLE bit has been cleared, a valid character must again set the SCRF bit before an idle condition can set the IDLE bit. Reset clears the IDLE bit.

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

OR — Receiver Overrun Bit

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

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

Software latency may allow an overrun to occur between reads of SCS1 and SCDR in the flag-clearing sequence. Figure 11-13 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

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

NF — Receiver Noise Flag Bit

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

1 = Noise detected
0 = No noise detected

FE — Receiver Framing Error Bit

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

1 = Framing error detected
0 = No framing error detected

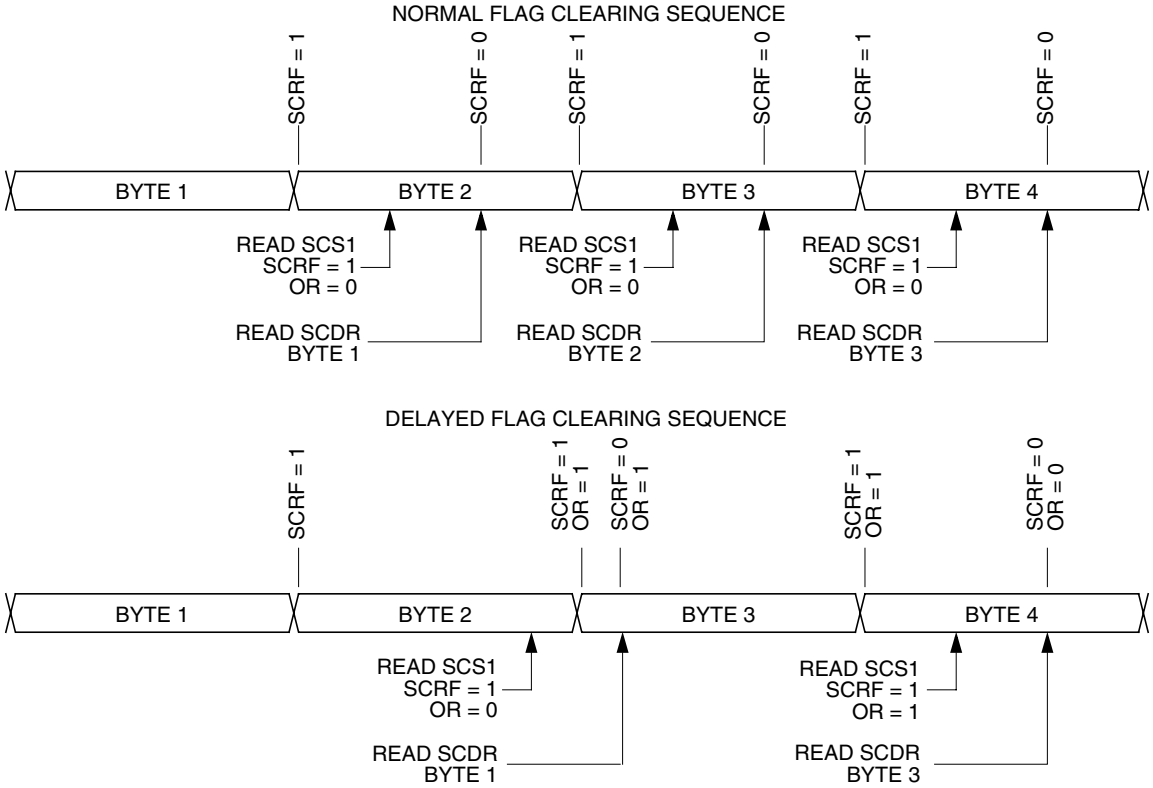


Figure 11-13. Flag Clearing Sequence

PE — Receiver Parity Error Bit

This clearable, read-only bit is set when the SCI detects a parity error in incoming data. PE generates an SCI error CPU interrupt request if the PEIE bit in SCC3 is also set. Clear the PE bit by reading SCS1 with PE set and then reading the SCDR. Reset clears the PE bit.

1 = Parity error detected
0 = No parity error detected

11.8.5 SCI Status Register 2

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

- Break character detected
- Incoming data

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

= Unimplemented

Figure 11-14. SCI Status Register 2 (SCS2)

BKF — Break Flag Bit

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

1 = Break character detected
0 = No break character detected

RPF — Reception in Progress Flag Bit

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

1 = Reception in progress
0 = No reception in progress

11.8.6 SCI Data Register

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

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

Figure 11-15. SCI Data Register (SCDR)

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

Reading the SCDR accesses the read-only received data bits, R7–R0. Writing to the SCDR writes the data to be transmitted, T7–T0. Reset has no effect on the SCDR.

NOTE

Do not use read/modify/write instructions on the SCI data register.

11.8.7 SCI Baud Rate Register

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

Address:	$0019	6	5	4	3	2	1	Bit 0
Read:	0	0	SCP1	SCP0	R	SCR2	SCR1	SCR0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 11-16. SCI Baud Rate Register (SCBR)

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in Table 11-6. Reset clears SCP1 and SCP0.

Table 11-6. SCI Baud Rate Prescaling

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

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in Table 11-7. Reset clears SCR2–SCR0.

Table 11-7. SCI Baud Rate Selection

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

Use this formula to calculate the SCI baud rate:

$$\text{baud rate} = \frac{\text{SCI clock source}}{64 \times \text{PD} \times \text{BD}}$$

where:

SCI clock source = f_{BUS} or CGMXCLK
(selected by SCIBDSRC bit in CONFIG2 register)
PD = prescaler divisor
BD = baud rate divisor

Table 11-8 shows the SCI baud rates that can be generated with a 4.9152-MHz bus clock when f_{BUS} is selected as SCI clock source.

Table 11-8. SCI Baud Rate Selection Examples

SCP1 and SCP0	Prescaler Divisor (PD)	SCR2, SCR1, and SCR0	Baud Rate Divisor (BD)	Baud Rate (f_{BUS} = 4.9152 MHz)
00	1	000	1	76,800
00	1	001	2	38,400
00	1	010	4	19,200
00	1	011	8	9600
00	1	100	16	4800
00	1	101	32	2400
00	1	110	64	1200
00	1	111	128	600
01	3	000	1	25,600
01	3	001	2	12,800
01	3	010	4	6400
01	3	011	8	3200
01	3	100	16	1600
01	3	101	32	800
01	3	110	64	400
01	3	111	128	200
10	4	000	1	19,200
10	4	001	2	9600
10	4	010	4	4800
10	4	011	8	2400
10	4	100	16	1200
10	4	101	32	600
10	4	110	64	300
10	4	111	128	150
11	13	000	1	5908
11	13	001	2	2954
11	13	010	4	1477
11	13	011	8	739
11	13	100	16	369
11	13	101	32	185
11	13	110	64	92
11	13	111	128	46

Chapter 12
Infrared Serial Communications Interface Module (IRSCI)

12.1 Introduction

The MC68HC908AP64A has two SCI modules:

- SCI1 is a standard SCI module, and
- SCI2 is an infrared SCI module.

This section describes SCI2, the infrared serial communications interface (IRSCI) module which allows high-speed asynchronous communications with peripheral devices and other MCUs. This IRSCI consists of an SCI module for conventional SCI functions and a software programmable infrared encoder/decoder sub-module for encoding/decoding the serial data for connection to infrared LEDs in remote control applications.

NOTE

When the IRSCI is enabled, the SCTxD pin is an open-drain output and requires a pullup resistor to be connected for proper SCI operation.

Features of the SCI module include the following:

- Full duplex operation
- Standard mark/space non-return-to-zero (NRZ) format
- Programmable 8-bit or 9-bit character length
- Separately enabled transmitter and receiver
- Separate receiver and transmitter CPU interrupt requests
- Two receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
- Interrupt-driven operation with eight interrupt flags:
  - Transmitter empty
  - Transmission complete
  - Receiver full
  - Idle receiver input
  - Receiver overrun
  - Noise error
  - Framing error
  - Parity error
- Receiver framing error detection
- Hardware parity checking
- 1/16 bit-time noise detection

Features of the infrared (IR) sub-module include the following:

- IR sub-module enable/disable for infrared SCI or conventional SCI on SCTxD and SCRxD pins
- Software selectable infrared modulation/demodulation (3/16, 1/16 or 1/32 width pulses)

12.2 Pin Name Conventions

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

- RxD (receive data)
- TxD (transmit data)

IRSCI I/O (input/output) lines are implemented by sharing parallel I/O port pins. The full name of an IRSCI input or output reflects the name of the shared port pin. Table 12-1 shows the full names and the generic names of the IRSCI I/O pins. The generic pin names appear in the text of this section.

Table 12-1. Pin Name Conventions

Generic Pin Names:	RxD	TxD
Full Pin Names:	PTC7/SCRxD	PTC6/SCTxD

NOTE

When the IRSCI is enabled, the SCTxD pin is an open-drain output and requires a pullup resistor to be connected for proper SCI operation.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0040	IRSCI Control Register 1 (IRSCC1)	Read:	LOOPS	ENSCI	0	M	WAKE	ILTY	PEN	PTY
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0041	IRSCI Control Register 2 (IRSCC2)	Read:	SCTIE	TCIE	SCRIE	ILIE	TE	RE	RWU	SBK
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0042	IRSCI Control Register 3 (IRSCC3)	Read:	R8	T8	DMARE	DMATE	ORIE	NEIE	FEIE	PEIE
		Write:								
		Reset:	U	U	0	0	0	0	0	0
$0043	IRSCI Status Register 1 (IRSCS1)	Read:	SCTE	TC	SCRF	IDLE	OR	NF	FE	PE
		Write:								
		Reset:	1	1	0	0	0	0	0	0
$0044	IRSCI Status Register 2 (IRSCS2)	Read:							BKF	RPF
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0045	IRSCI Data Register (IRSCDR)	Read:	R7	R6	R5	R4	R3	R2	R1	R0
		Write:	T7	T6	T5	T4	T3	T2	T1	T0
		Reset:	Unaffected by reset							
$0046	IRSCI Baud Rate Register (IRSCBR)	Read:	CKS	0	SCP1	SCP0	R	SCR2	SCR1	SCR0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$0047	IRSCI Infrared Control Register (IRSCIRCR)	Read:	R	0	0	0	R	TNP1	TNP0	IREN
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved U = Unaffected

Figure 12-1. IRSCI I/O Registers Summary

12.3 IRSCI Module Overview

The IRSCI consists of a serial communications interface (SCI) and a infrared interface sub-module as shown in Figure 12-2.

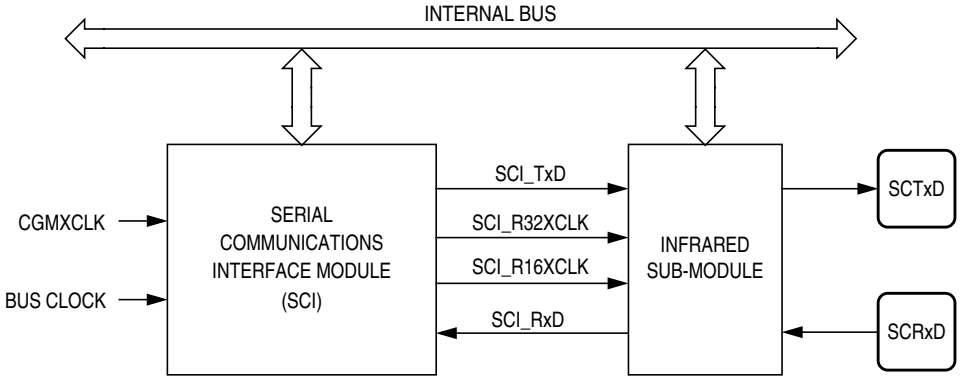


Figure 12-2. IRSCI Block Diagram

The SCI module provides serial data transmission and reception, with a programmable baud rate clock based on the bus clock or the CGMXCLK.

The infrared sub-module receives two clock sources from the SCI module: SCI_R16XCLK and SCI_R32XCLK. Both reference clocks are used to generate the narrow pulses during data transmission. The SCI_R16XCLK and SCI_R32XCLK are internal clocks with frequencies that are 16 and 32 times the baud rate respectively. Both SCI_R16XCLK and SCI_R32XCLK clocks are used for transmitting data. The SCI_R16XCLK clock is used only for receiving data.

NOTE

For proper SCI function (transmit or receive), the bus clock MUST be programmed to at least 32 times that of the selected baud rate.
When the infrared sub-module is disabled, signals on the TxD and RxD pins pass through unchanged to the SCI module.

12.4 Infrared Functional Description

Figure 12-3 shows the structure of the infrared sub-module.

The infrared sub-module provides the capability of transmitting narrow pulses to an infrared LED and receiving narrow pulses and transforming them to serial bits, which are sent to the SCI module. The infrared sub-module receives two clocks from the SCI. One of these two clocks is selected as the base clock to generate the 3/16, 1/16, or 1/32 bit width narrow pulses during transmission.

The sub-module consists of two main blocks: the transmit encoder and the receive decoder. When transmitting data, the SCI data stream is encoded by the infrared sub-module. For every "0" bit, a narrow "low" pulse is transmitted; no pulse is transmitted for "1" bits. When receiving data, the infrared pulses should be detected using an infrared photo diode for conversion to CMOS voltage levels before connecting to the RxD pin for the infrared decoder. The SCI data stream is reconstructed by stretching the "0" pulses.

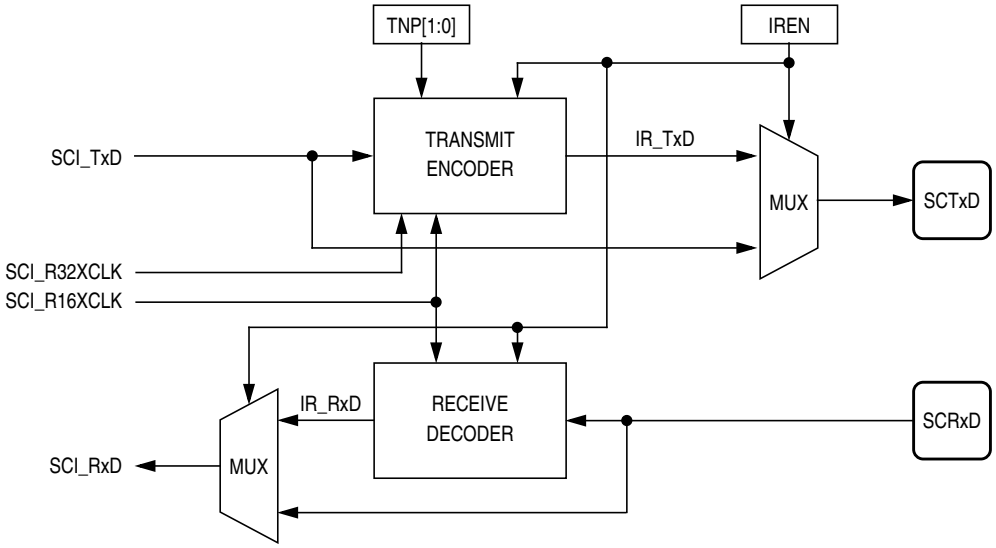


Figure 12-3. Infrared Sub-Module Diagram

12.4.1 Infrared Transmit Encoder

The infrared transmit encoder converts the "0" bits in the serial data stream from the SCI module to narrow "low" pulses, to the TxD pin. The narrow pulse is sent with a duration of 1/32, 1/16, or 3/16 of a data bit width**.** When two consecutive zeros are sent, the two consecutive narrow pulses will be separated by a time equal to a data bit width.

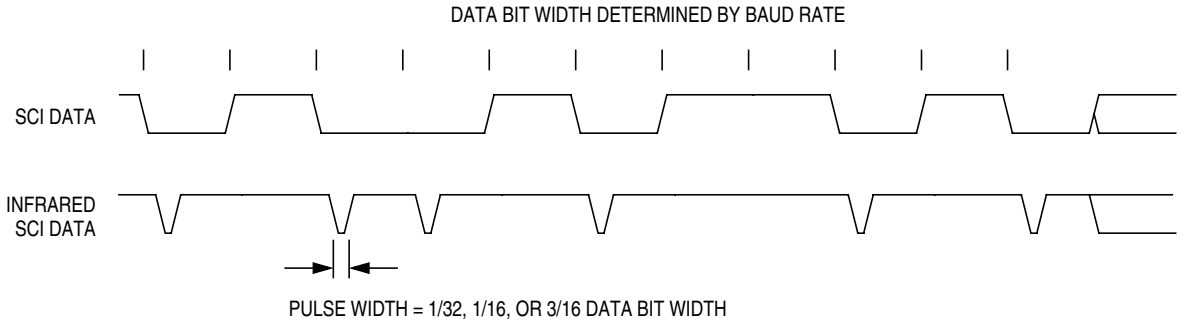


Figure 12-4. Infrared SCI Data Example

12.4.2 Infrared Receive Decoder

The infrared receive decoder converts low narrow pulses from the RxD pin to standard SCI data bits. The reference clock, SCI_R16XCLK, clocks a four bit internal counter which counts from 0 to 15. An incoming pulse starts the internal counter and a "0" is sent out to the IR_RxD output. Subsequent incoming pulses are ignored when the counter count is between 0 and 7; IR_RxD remains "0". Once the counter passes 7, an incoming pulse will reset the counter; IR_RxD remains "0". When the counter reaches 15, the IR_RxD output returns to "1", the counter stops and waits for further pulses. A pulse is interpreted as jitter if it arrives shortly after the counter reaches 15; IR_RxD remains "1".

12.5 SCI Functional Description

Figure 12-5 shows the structure of the SCI.

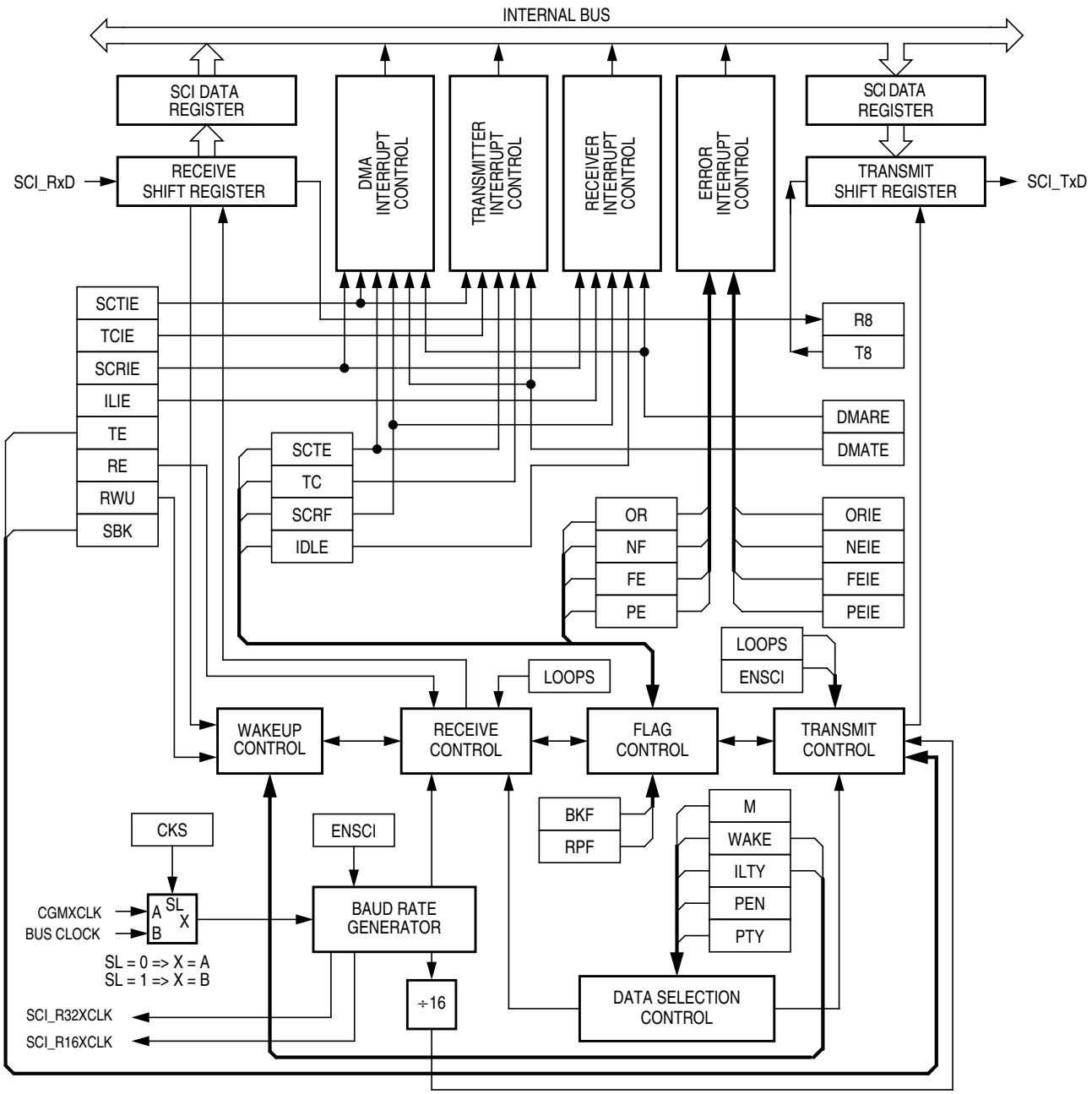


Figure 12-5. SCI Module Block Diagram

The SCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

NOTE

For SCI operations, the IR sub-module is transparent to the SCI module. Data at going out of the SCI transmitter and data going into the SCI receiver is always in SCI format. It makes no difference to the SCI module whether the IR sub-module is enabled or disabled.

NOTE

This SCI module is a standard HC08 SCI module with the following modifications:

- *A control bit, CKS, is added to the SCI baud rate control register to select between two input clocks for baud rate clock generation*
- *The TXINV bit is removed from the SCI control register 1*

12.5.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 12-6.

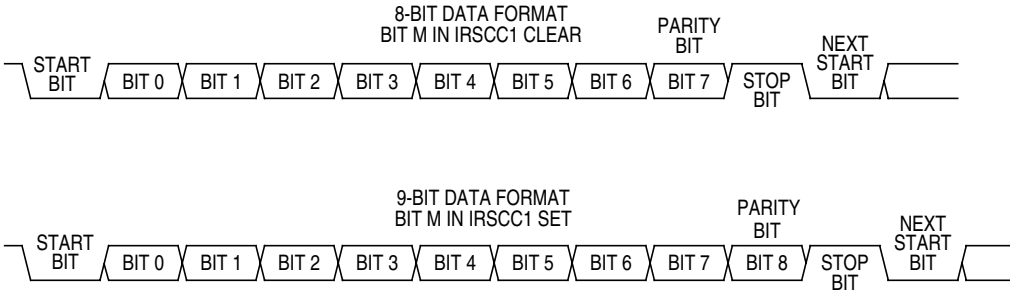


Figure 12-6. SCI Data Formats

12.5.2 Transmitter

Figure 12-7 shows the structure of the SCI transmitter.

The baud rate clock source for the SCI can be selected by the CKS bit, in the SCI baud rate register (see 12.9.7 IRSCI Baud Rate Register).

12.5.2.1 Character Length

The transmitter can accommodate either 8-bit or 9-bit data. The state of the M bit in IRSCI control register 1 (IRSCC1) determines character length. When transmitting 9-bit data, bit T8 in IRSCI control register 3 (IRSCC3) is the ninth bit (bit 8).

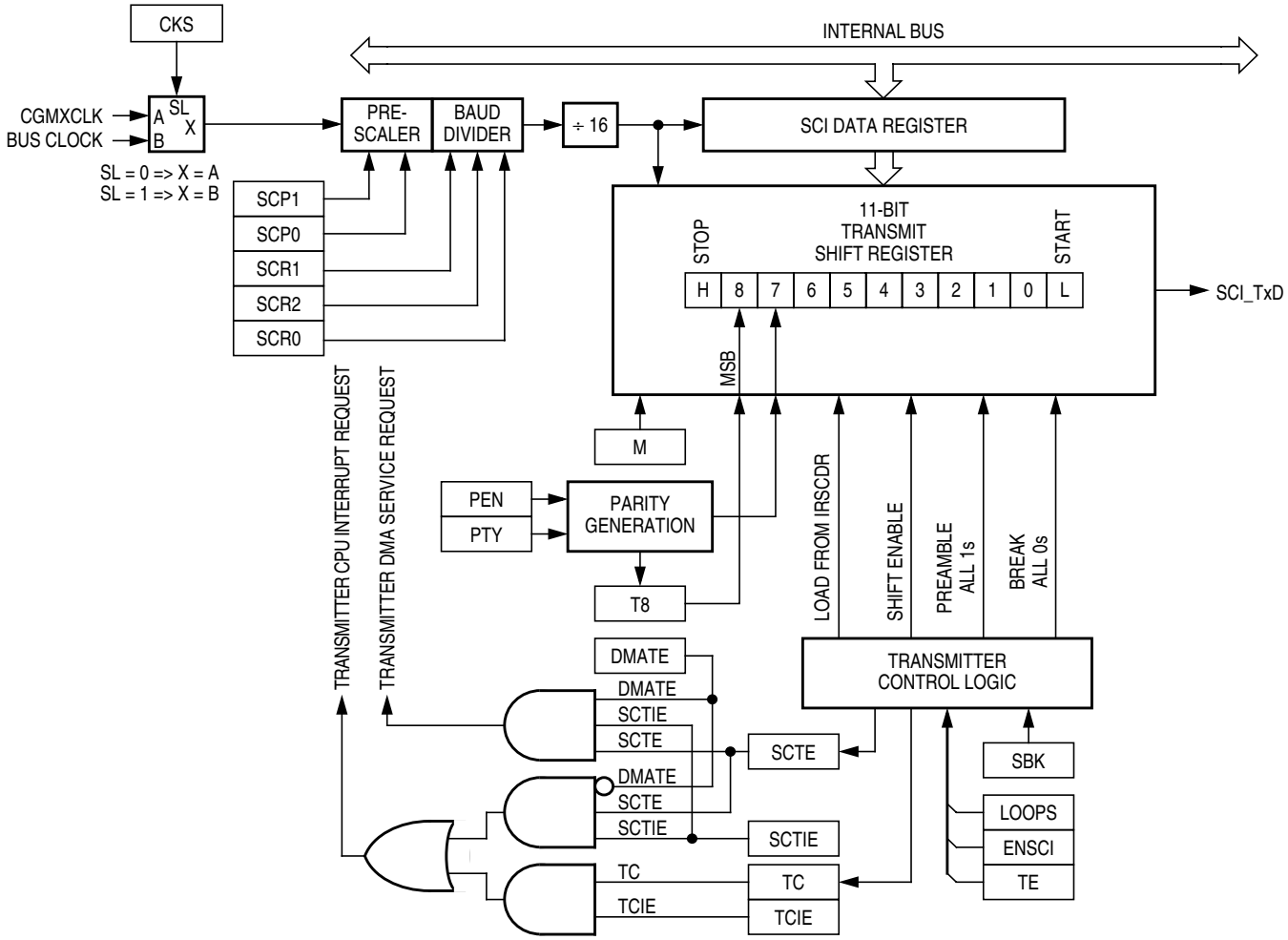


Figure 12-7. SCI Transmitter

12.5.2.2 Character Transmission

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

1. Enable the SCI by writing a logic 1 to the enable SCI bit (ENSCI) in IRSCI control register 1 (IRSCC1).
2. Enable the transmitter by writing a logic 1 to the transmitter enable bit (TE) in IRSCI control register 2 (IRSCC2).
3. Clear the SCI transmitter empty bit by first reading IRSCI status register 1 (IRSCS1) and then writing to the IRSCDR.
4. Repeat step 3 for each subsequent transmission.

At the start of a transmission, transmitter control logic automatically loads the transmit shift register with a preamble of logic 1s. After the preamble shifts out, control logic transfers the IRSCDR data into the transmit shift register. A logic 0 start bit automatically goes into the least significant bit position of the transmit shift register. A logic 1 stop bit goes into the most significant bit position.

The SCI transmitter empty bit, SCTE, in IRSCS1 becomes set when the IRSCDR transfers a byte to the transmit shift register. The SCTE bit indicates that the IRSCDR can accept new data from the internal data bus. If the SCI transmit interrupt enable bit, SCTIE, in IRSCC2 is also set, the SCTE bit generates a transmitter interrupt request.

When the transmit shift register is not transmitting a character, the TxD pin goes to the idle condition, logic 1. If at any time software clears the ENSCI bit in IRSCI control register 1 (IRSCC1), the transmitter and receiver relinquish control of the port pins.

12.5.2.3 Break Characters

Writing a logic 1 to the send break bit, SBK, in IRSCC2 loads the transmit shift register with a break character. A break character contains all logic 0s and has no start, stop, or parity bit. Break character length depends on the M bit in IRSCC1. As long as SBK is at logic 1, transmitter logic continuously loads break characters into the transmit shift register. After software clears the SBK bit, the shift register finishes transmitting the last break character and then transmits at least one logic 1. The automatic logic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

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

Receiving a break character has the following effects on SCI registers:

- Sets the framing error bit (FE) in IRSCS1
- Sets the SCI receiver full bit (SCRF) in IRSCS1
- Clears the SCI data register (IRSCDR)
- Clears the R8 bit in IRSCC3
- Sets the break flag bit (BKF) in IRSCS2
- May set the overrun (OR), noise flag (NF), parity error (PE), or reception in progress flag (RPF) bits

12.5.2.4 Idle Characters

An idle character contains all logic 1s and has no start, stop, or parity bit. Idle character length depends on the M bit in IRSCC1. The preamble is a synchronizing idle character that begins every transmission.

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

NOTE

When queueing an idle character, return the TE bit to logic 1 before the stop bit of the current character shifts out to the TxD pin. Setting TE after the stop bit appears on TxD causes data previously written to the IRSCDR to be lost.

Toggle the TE bit for a queued idle character when the SCTE bit becomes set and just before writing the next byte to the IRSCDR.

12.5.2.5 Transmitter Interrupts

The following conditions can generate CPU interrupt requests from the SCI transmitter:

- SCI transmitter empty (SCTE) — The SCTE bit in IRSCS1 indicates that the IRSCDR has transferred a character to the transmit shift register. SCTE can generate a transmitter CPU interrupt request. Setting the SCI transmit interrupt enable bit, SCTIE, in IRSCC2 enables the SCTE bit to generate transmitter CPU interrupt requests.
- Transmission complete (TC) — The TC bit in IRSCS1 indicates that the transmit shift register and the IRSCDR are empty and that no break or idle character has been generated. The transmission complete interrupt enable bit, TCIE, in IRSCC2 enables the TC bit to generate transmitter CPU interrupt requests.

12.5.3 Receiver

Figure 12-8 shows the structure of the SCI receiver.

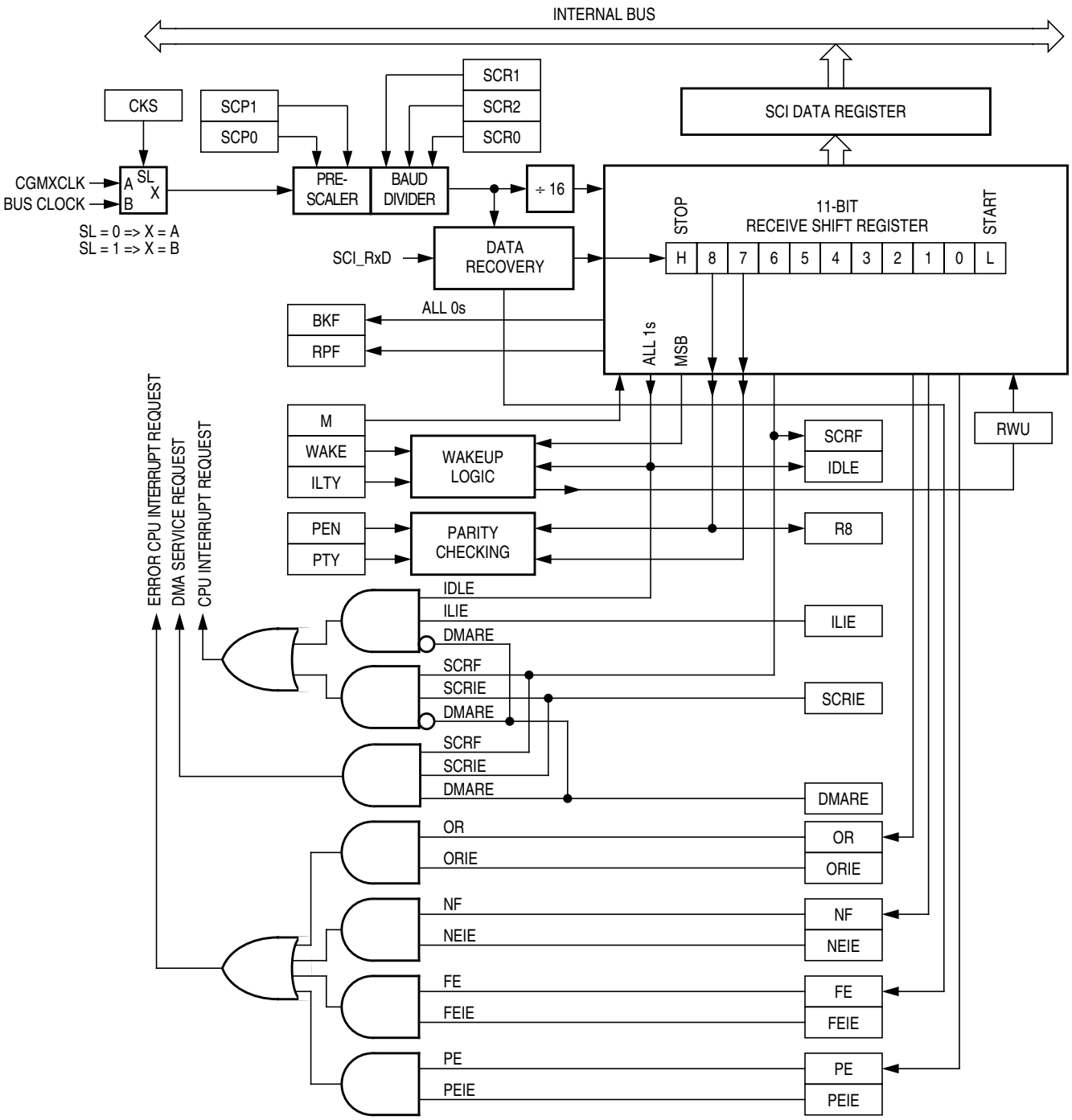


Figure 12-8. SCI Receiver Block Diagram

12.5.3.1 Character Length

The receiver can accommodate either 8-bit or 9-bit data. The state of the M bit in IRSCI control register 1 (IRSCC1) determines character length. When receiving 9-bit data, bit R8 in IRSCI control register 2 (IRSCC2) is the ninth bit (bit 8). When receiving 8-bit data, bit R8 is a copy of the eighth bit (bit 7).

12.5.3.2 Character Reception

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

After a complete character shifts into the receive shift register, the data portion of the character transfers to the IRSCDR. The SCI receiver full bit, SCRF, in IRSCI status register 1 (IRSCS1) becomes set, indicating that the received byte can be read. If the SCI receive interrupt enable bit, SCRIE, in IRSCC2 is also set, the SCRF bit generates a receiver CPU interrupt request.

12.5.3.3 Data Sampling

The receiver samples the RxD pin at the RT clock rate. The RT clock is an internal signal with a frequency 16 times the baud rate. To adjust for baud rate mismatch, the RT clock is resynchronized at the following times (see Figure 12-9):

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

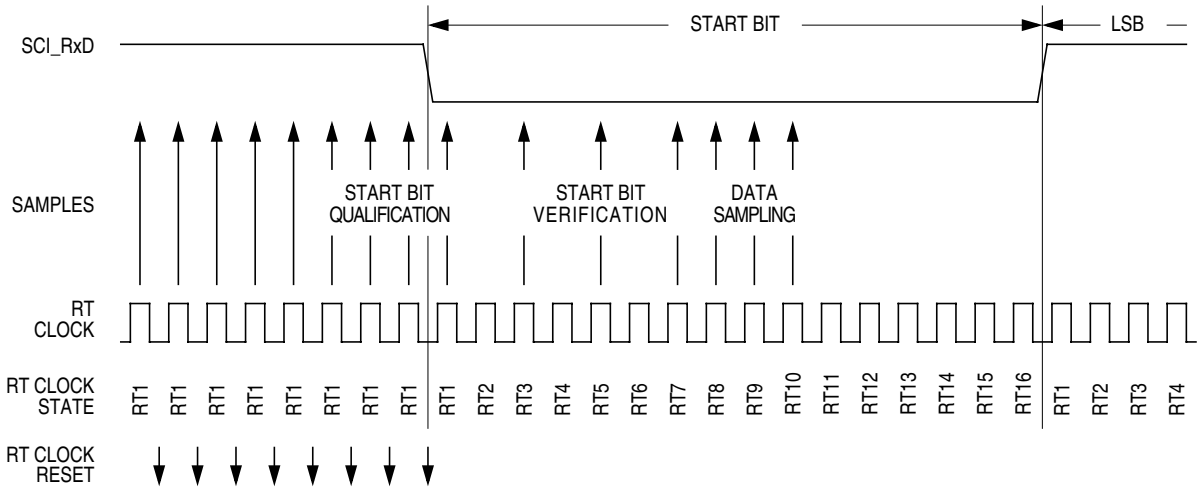


Figure 12-9. Receiver Data Sampling

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

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 12-2 summarizes the results of the start bit verification samples.

Table 12-2. Start Bit Verification

RT3, RT5, and RT7 Samples	Start Bit Verification	Noise Flag
000	Yes	0
001	Yes	1
010	Yes	1

Table 12-2. Start Bit Verification

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

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

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 12-3 summarizes the results of the data bit samples.

Table 12-3. Data Bit Recovery

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

NOTE

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

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 12-4 summarizes the results of the stop bit samples.

Table 12-4. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
000	1	0
001	1	1

Table 12-4. Stop Bit Recovery

RT8, RT9, and RT10 Samples	Framing Error Flag	Noise Flag
010	1	1
011	0	1
100	1	1
101	0	1
110	0	1
111	0	0

12.5.3.4 Framing Errors

If the data recovery logic does not detect a logic 1 where the stop bit should be in an incoming character, it sets the framing error bit, FE, in IRSCS1. The FE flag is set at the same time that the SCRF bit is set. A break character that has no stop bit also sets the FE bit.

12.5.3.5 Baud Rate Tolerance

A transmitting device may be operating at a baud rate below or above the receiver baud rate. Accumulated bit time misalignment can cause one of the three stop bit data samples to fall outside the actual stop bit. Then a noise error occurs. If more than one of the samples is outside the stop bit, a framing error occurs. In most applications, the baud rate tolerance is much more than the degree of misalignment that is likely to occur.

As the receiver samples an incoming character, it resynchronizes the RT clock on any valid falling edge within the character. Resynchronization within characters corrects misalignments between transmitter bit times and receiver bit times.

Slow Data Tolerance

Figure 12-10 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

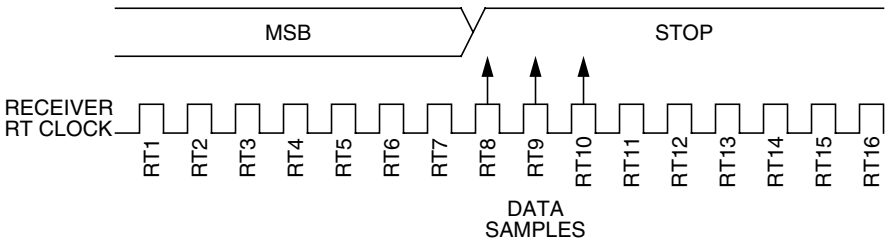


Figure 12-10. Slow Data

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 12-10, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit character with no errors is

$$\left|\frac{154 - 147}{154}\right| \times 100 = 4.54\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver
10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 12-10, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles + 3 RT cycles = 163 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is

$$\left|\frac{170 - 163}{170}\right| \times 100 = 4.12\%$$

Fast Data Tolerance

Figure 12-11 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

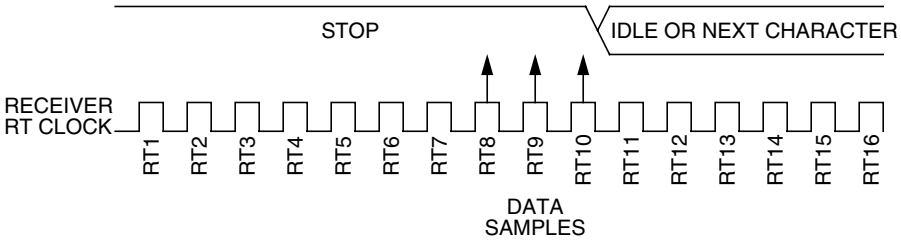


Figure 12-11. Fast Data

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.

With the misaligned character shown in Figure 12-11, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.9\dot{0}\%$$

For a 9-bit character, data sampling of the stop bit takes the receiver
10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.

With the misaligned character shown in Figure 12-11, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

The maximum percent difference between the receiver count and the transmitter count of a fast 9-bit character with no errors is

$$\left|\frac{170 - 176}{170}\right| \times 100 = 3.53\%$$

12.5.3.6 Receiver Wakeup

So that the MCU can ignore transmissions intended only for other receivers in multiple-receiver systems, the receiver can be put into a standby state. Setting the receiver wakeup bit, RWU, in IRSCC2 puts the receiver into a standby state during which receiver interrupts are disabled.

Depending on the state of the WAKE bit in IRSCC1, either of two conditions on the RxD pin can bring the receiver out of the standby state:

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

NOTE

Clearing the WAKE bit after the RxD pin has been idle may cause the receiver to wake up immediately.

12.5.3.7 Receiver Interrupts

The following sources can generate CPU interrupt requests from the SCI receiver:

- SCI receiver full (SCRF) — The SCRF bit in IRSCS1 indicates that the receive shift register has transferred a character to the IRSCDR. SCRF can generate a receiver interrupt request. Setting the SCI receive interrupt enable bit, SCRIE, in IRSCC2 enables the SCRF bit to generate receiver CPU interrupts.
- Idle input (IDLE) — The IDLE bit in IRSCS1 indicates that 10 or 11 consecutive logic 1s shifted in from the RxD pin. The idle line interrupt enable bit, ILIE, in IRSCC2 enables the IDLE bit to generate CPU interrupt requests.

12.5.3.8 Error Interrupts

The following receiver error flags in IRSCS1 can generate CPU interrupt requests:

- Receiver overrun (OR) — The OR bit indicates that the receive shift register shifted in a new character before the previous character was read from the IRSCDR. The previous character remains in the IRSCDR, and the new character is lost. The overrun interrupt enable bit, ORIE, in IRSCC3 enables OR to generate SCI error CPU interrupt requests.
- Noise flag (NF) — The NF bit is set when the SCI detects noise on incoming data or break characters, including start, data, and stop bits. The noise error interrupt enable bit, NEIE, in IRSCC3 enables NF to generate SCI error CPU interrupt requests.

- Framing error (FE) — The FE bit in IRSCS1 is set when a logic 0 occurs where the receiver expects a stop bit. The framing error interrupt enable bit, FEIE, in IRSCC3 enables FE to generate SCI error CPU interrupt requests.
- Parity error (PE) — The PE bit in IRSCS1 is set when the SCI detects a parity error in incoming data. The parity error interrupt enable bit, PEIE, in IRSCC3 enables PE to generate SCI error CPU interrupt requests.

12.6 Low-Power Modes

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

12.6.1 Wait Mode

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

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

Refer to 7.6 Low-Power Modes for information on exiting wait mode.

12.6.2 Stop Mode

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

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

Refer to 7.6 Low-Power Modes for information on exiting stop mode.

12.7 SCI During Break Module Interrupts

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

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

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

12.8 I/O Signals

The two IRSCI I/O pins are:

- PTC6/SCTxD — Transmit data
- PTC7/SCRxD — Receive data

12.8.1 PTC6/SCTxD (Transmit Data)

The PTC6/SCTxD pin is the serial data (standard or infrared) output from the SCI transmitter. The IRSCI shares the PTC6/SCTxD pin with port C. When the IRSCI is enabled, the PTC6/SCTxD pin is an output regardless of the state of the DDRC6 bit in data direction register C (DDRC).

NOTE

The PTC6/SCTxD pin is an open-drain pin when configured as an output. Therefore, when configured as SCTxD or a general purpose output pin (PTC6), a pullup resistor must be connected to this pin.

12.8.2 PTC7/SCRxD (Receive Data)

The PTC7/SCRxD pin is the serial data input to the IRSCI receiver. The IRSCI shares the PTC7/SCRxD pin with port C. When the IRSCI is enabled, the PTC7/SCRxD pin is an input regardless of the state of the DDRC7 bit in data direction register C (DDRC).

NOTE

The PTC7/SCRxD pin is an open-drain pin when configured as an output. Therefore, when configured as a general purpose output pin (PTC7), a pullup resistor must be connected to this pin.

Table 12-5 shows a summary of I/O pin functions when the SCI is enabled.

Table 12-5. SCI Pin Functions (Standard and Infrared)

IRSCC1 [ENSCI]	IRSCIRCR [IREN]	IRSCC2 [TE]	IRSCC2 [RE]	TxD Pin	RxD Pin
1	0	0	0	Hi-Z(1)	Input ignored (terminate externally)
1	0	0	1	Hi-Z(1)	Input sampled, pin should idle high
1	0	1	0	Output SCI (idle high)	Input ignored (terminate externally)
1	0	1	1	Output SCI (idle high)	Input sampled, pin should idle high
1	1	0	0	Hi-Z(1)	Input ignored (terminate externally)
1	1	0	1	Hi-Z(1)	Input sampled, pin should idle high
1	1	1	0	Output IR SCI (idle high)	Input ignored (terminate externally)
1	1	1	1	Output IR SCI (idle high)	Input sampled, pin should idle high
0	X	X	X	Pins under port control (standard I/O port)	

1. After completion of transmission in progress.

12.9 I/O Registers

The following I/O registers control and monitor SCI operation:

- IRSCI control register 1 (IRSCC1)
- IRSCI control register 2 (IRSCC2)
- IRSCI control register 3 (IRSCC3)

- IRSCI status register 1 (IRSCS1)
- IRSCI status register 2 (IRSCS2)
- IRSCI data register (IRSCDR)
- IRSCI baud rate register (IRSCBR)
- IRSCI infrared control register (IRSCIRCR)

12.9.1 IRSCI Control Register 1

SCI control register 1:

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

Address: $0040

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

Figure 12-12. IRSCI Control Register 1 (IRSCC1)

LOOPS — Loop Mode Select Bit

This read/write bit enables loop mode operation for the SCI only. In loop mode the RxD pin is disconnected from the SCI, and the transmitter output goes into the receiver input. Both the transmitter and the receiver must be enabled to use loop mode. The infrared encoder/decoder is not in the loop. Reset clears the LOOPS bit.

1 = Loop mode enabled
0 = Normal operation enabled

ENSCI — Enable SCI Bit

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

1 = SCI enabled
0 = SCI disabled

M — Mode (Character Length) Bit

This read/write bit determines whether SCI characters are eight or nine bits long. (See Table 12-6.) The ninth bit can serve as an extra stop bit, as a receiver wakeup signal, or as a parity bit. Reset clears the M bit.

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

WAKE — Wakeup Condition Bit

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

1 = Address mark wakeup
0 = Idle line wakeup

ILTY — Idle Line Type Bit

This read/write bit determines when the SCI starts counting logic 1s as idle character bits. The counting begins either after the start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit may cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. Reset clears the ILTY bit.

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

PEN — Parity Enable Bit

This read/write bit enables the SCI parity function. (See Table 12-6.) When enabled, the parity function inserts a parity bit in the most significant bit position. (See Figure 12-6.) Reset clears the PEN bit.

1 = Parity function enabled
0 = Parity function disabled

PTY — Parity Bit

This read/write bit determines whether the SCI generates and checks for odd parity or even parity. (See Table 12-6.) Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

NOTE

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

Table 12-6. Character Format Selection

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

12.9.2 IRSCI Control Register 2

IRSCI control register 2:

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

Address: $0041

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

Figure 12-13. IRSCI Control Register 2 (IRSCC2)

SCTIE — SCI Transmit Interrupt Enable Bit

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests. Reset clears the SCTIE bit.

1 = SCTE enabled to generate CPU interrupt
0 = SCTE not enabled to generate CPU interrupt

TCIE — Transmission Complete Interrupt Enable Bit

This read/write bit enables the TC bit to generate SCI transmitter CPU interrupt requests. Reset clears the TCIE bit.

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

SCRIE — SCI Receive Interrupt Enable Bit

This read/write bit enables the SCRF bit to generate SCI receiver CPU interrupt requests. Reset clears the SCRIE bit.

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

ILIE — Idle Line Interrupt Enable Bit

This read/write bit enables the IDLE bit to generate SCI receiver CPU interrupt requests. Reset clears the ILIE bit.

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

TE — Transmitter Enable Bit

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

1 = Transmitter enabled
0 = Transmitter disabled

NOTE

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

RE — Receiver Enable Bit

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

1 = Receiver enabled
0 = Receiver disabled

NOTE

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

RWU — Receiver Wakeup Bit

This read/write bit puts the receiver in a standby state during which receiver interrupts are disabled. The WAKE bit in IRSCC1 determines whether an idle input or an address mark brings the receiver out of the standby state and clears the RWU bit. Reset clears the RWU bit.

1 = Standby state
0 = Normal operation

SBK — Send Break Bit

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

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

NOTE

Do not toggle the SBK bit immediately after setting the SCTE bit. Toggling SBK before the preamble begins causes the SCI to send a break character instead of a preamble.

12.9.3 IRSCI Control Register 3

IRSCI control register 3:

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

Address: $0042

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

= Unimplemented U = Unaffected

Figure 12-14. IRSCI Control Register 3 (IRSCC3)

R8 — Received Bit 8

When the SCI is receiving 9-bit characters, R8 is the read-only ninth bit (bit 8) of the received character. R8 is received at the same time that the IRSCDR receives the other 8 bits.
When the SCI is receiving 8-bit characters, R8 is a copy of the eighth bit (bit 7). Reset has no effect on the R8 bit.

T8 — Transmitted Bit 8

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

DMARE — DMA Receive Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

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

DMATE — DMA Transfer Enable Bit

CAUTION

The DMA module is not included on this MCU. Writing a logic 1 to DMARE or DMATE may adversely affect MCU performance.

1 = SCTE DMA service requests enabled; SCTE CPU interrupt requests disabled
0 = SCTE DMA service requests disabled; SCTE CPU interrupt requests enabled

ORIE — Receiver Overrun Interrupt Enable Bit

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

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

NEIE — Receiver Noise Error Interrupt Enable Bit

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

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

FEIE — Receiver Framing Error Interrupt Enable Bit

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

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

PEIE — Receiver Parity Error Interrupt Enable Bit

This read/write bit enables SCI error CPU interrupt requests generated by the parity error bit, PE. (See 12.9.4 IRSCI Status Register 1.) Reset clears PEIE.

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

12.9.4 IRSCI Status Register 1

SCI status register 1 contains flags to signal these conditions:

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

Address: $0043

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

= Unimplemented

Figure 12-15. IRSCI Status Register 1 (IRSCS1)

SCTE — SCI Transmitter Empty Bit

This clearable, read-only bit is set when the IRSCDR transfers a character to the transmit shift register. SCTE can generate an SCI transmitter CPU interrupt request. When the SCTIE bit in IRSCC2 is set, SCTE generates an SCI transmitter CPU interrupt request. In normal operation, clear the SCTE bit by reading IRSCS1 with SCTE set and then writing to IRSCDR. Reset sets the SCTE bit.

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

TC — Transmission Complete Bit

This read-only bit is set when the SCTE bit is set, and no data, preamble, or break character is being transmitted. TC generates an SCI transmitter CPU interrupt request if the TCIE bit in IRSCC2 is also set. TC is automatically cleared when data, preamble or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

SCRF — SCI Receiver Full Bit

This clearable, read-only bit is set when the data in the receive shift register transfers to the SCI data register. SCRF can generate an SCI receiver CPU interrupt request. When the SCRIE bit in IRSCC2 is set, SCRF generates a CPU interrupt request. In normal operation, clear the SCRF bit by reading IRSCS1 with SCRF set and then reading the IRSCDR. Reset clears SCRF.

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

IDLE — Receiver Idle Bit

This clearable, read-only bit is set when 10 or 11 consecutive logic 1s appear on the receiver input. IDLE generates an SCI receiver CPU interrupt request if the ILIE bit in IRSCC2 is also set. Clear the IDLE bit by reading IRSCS1 with IDLE set and then reading the IRSCDR. After the receiver is enabled,

it must receive a valid character that sets the SCRF bit before an idle condition can set the IDLE bit. Also, after the IDLE bit has been cleared, a valid character must again set the SCRF bit before an idle condition can set the IDLE bit. Reset clears the IDLE bit.

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

OR — Receiver Overrun Bit

This clearable, read-only bit is set when software fails to read the IRSCDR before the receive shift register receives the next character. The OR bit generates an SCI error CPU interrupt request if the ORIE bit in IRSCC3 is also set. The data in the shift register is lost, but the data already in the IRSCDR is not affected. Clear the OR bit by reading IRSCS1 with OR set and then reading the IRSCDR. Reset clears the OR bit.

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

Software latency may allow an overrun to occur between reads of IRSCS1 and IRSCDR in the flag-clearing sequence. Figure 12-16 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of IRSCDR does not clear the OR bit because OR was not set when IRSCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the IRSCDR instead of byte 2.

In applications that are subject to software latency or in which it is important to know which byte is lost due to an overrun, the flag-clearing routine can check the OR bit in a second read of IRSCS1 after reading the data register.

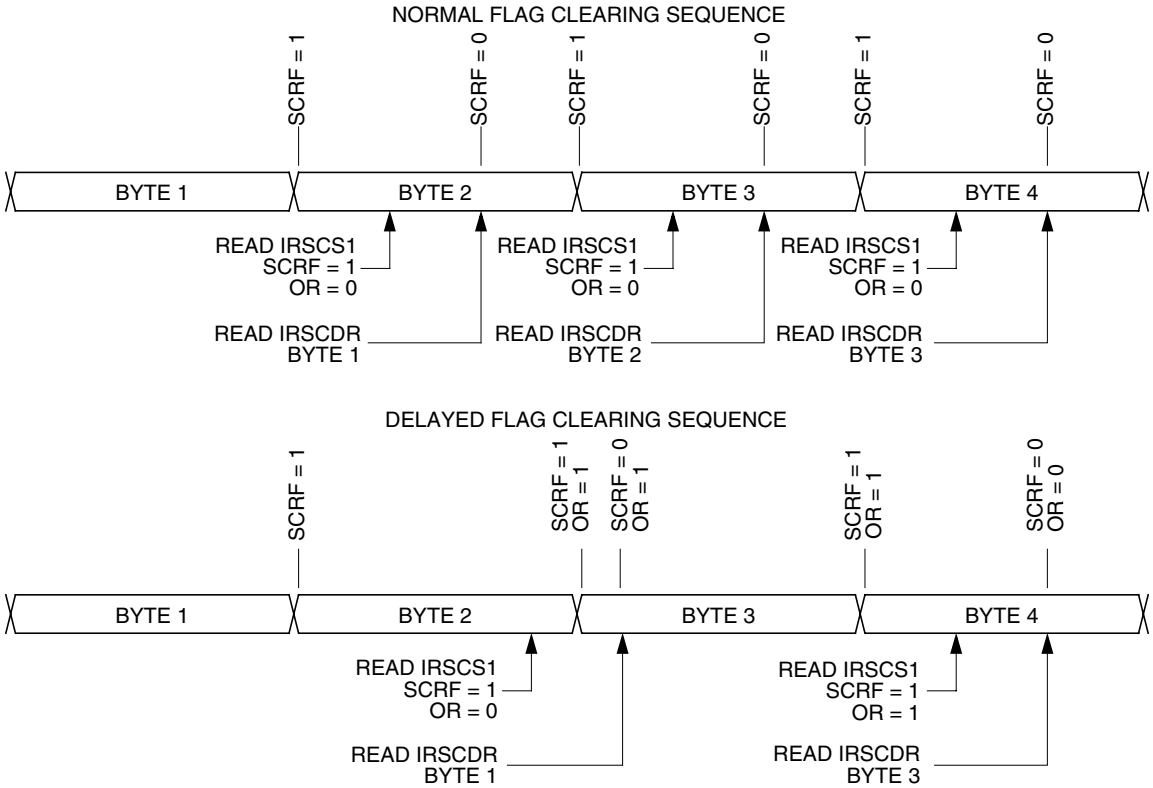


Figure 12-16. Flag Clearing Sequence

NF — Receiver Noise Flag Bit

This clearable, read-only bit is set when the SCI detects noise on the RxD pin. NF generates an SCI error CPU interrupt request if the NEIE bit in IRSCC3 is also set. Clear the NF bit by reading IRSCS1 and then reading the IRSCDR. Reset clears the NF bit.

1 = Noise detected
0 = No noise detected

FE — Receiver Framing Error Bit

This clearable, read-only bit is set when a logic 0 is accepted as the stop bit. FE generates an SCI error CPU interrupt request if the FEIE bit in IRSCC3 also is set. Clear the FE bit by reading IRSCS1 with FE set and then reading the IRSCDR. Reset clears the FE bit.

1 = Framing error detected
0 = No framing error detected

PE — Receiver Parity Error Bit

This clearable, read-only bit is set when the SCI detects a parity error in incoming data. PE generates an SCI error CPU interrupt request if the PEIE bit in IRSCC3 is also set. Clear the PE bit by reading IRSCS1 with PE set and then reading the IRSCDR. Reset clears the PE bit.

1 = Parity error detected
0 = No parity error detected

12.9.5 IRSCI Status Register 2

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

- Break character detected
- Incoming data

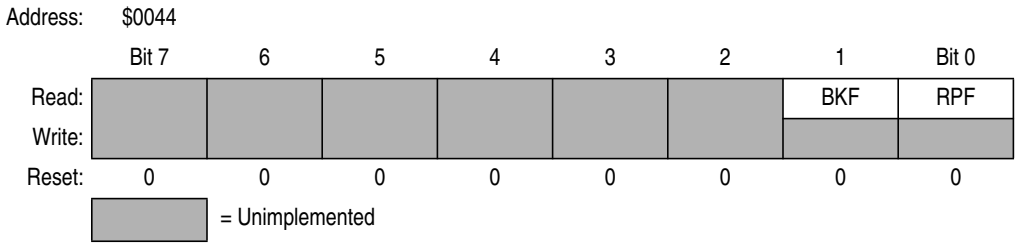


Figure 12-17. IRSCI Status Register 2 (IRSCS2)

BKF — Break Flag Bit

This clearable, read-only bit is set when the SCI detects a break character on the RxD pin. In IRSCS1, the FE and SCRF bits are also set. In 9-bit character transmissions, the R8 bit in IRSCC3 is cleared. BKF does not generate a CPU interrupt request. Clear BKF by reading IRSCS2 with BKF set and then reading the IRSCDR. Once cleared, BKF can become set again only after logic 1s again appear on the RxD pin followed by another break character. Reset clears the BKF bit.

1 = Break character detected
0 = No break character detected

RPF — Reception in Progress Flag Bit

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

1 = Reception in progress
0 = No reception in progress

12.9.6 IRSCI Data Register

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

Address: $0045

	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 12-18. IRSCI Data Register (IRSCDR)

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

Reading the IRSCDR accesses the read-only received data bits, R7–R0. Writing to the IRSCDR writes the data to be transmitted, T7–T0. Reset has no effect on the IRSCDR.

NOTE

Do not use read/modify/write instructions on the IRSCI data register.

12.9.7 IRSCI Baud Rate Register

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

Address: $0046

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

= Unimplemented R = Reserved

Figure 12-19. IRSCI Baud Rate Register (IRSCBR)

CKS — Baud Clock Input Select

This read/write bit selects the source clock for the baud rate generator. Reset clears the CKS bit, selecting CGMXCLK.

1 = Bus clock drives the baud rate generator
0 = CGMXCLK drives the baud rate generator

SCP1 and SCP0 — SCI Baud Rate Prescaler Bits

These read/write bits select the baud rate prescaler divisor as shown in Table 12-7. Reset clears SCP1 and SCP0.

Table 12-7. SCI Baud Rate Prescaling

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

SCR2–SCR0 — SCI Baud Rate Select Bits

These read/write bits select the SCI baud rate divisor as shown in Table 12-8. Reset clears SCR2–SCR0.

Table 12-8. IRSCI Baud Rate Selection

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

Use this formula to calculate the SCI baud rate:

$$\text{baud rate} = \frac{\text{SCI clock source}}{16 \times \text{PD} \times \text{BD}}$$

where:

SCI clock source = f_{BUS} or CGMXCLK
(selected by CKS bit)
PD = prescaler divisor
BD = baud rate divisor

Table 12-9 shows the SCI baud rates that can be generated with a 4.9152-MHz bus clock when f_{BUS} is selected as SCI clock source.

Table 12-9. IRSCI Baud Rate Selection Examples

SCP1 and SCP0	Prescaler Divisor (PD)	SCR2, SCR1, and SCR0	Baud Rate Divisor (BD)	Baud Rate (f_{BUS} = 4.9152 MHz)
00	1	000	1	—
00	1	001	2	—
00	1	010	4	76800
00	1	011	8	38400
00	1	100	16	19200
00	1	101	32	9600
00	1	110	64	4800
00	1	111	128	2400
01	3	000	1	—
01	3	001	2	51200
01	3	010	4	25600
01	3	011	8	12800
01	3	100	16	6400
01	3	101	32	3200
01	3	110	64	1600
01	3	111	128	800
10	4	000	1	76800
10	4	001	2	38400
10	4	010	4	19200
10	4	011	8	9600
10	4	100	16	4800
10	4	101	32	2400
10	4	110	64	1200
10	4	111	128	600
11	13	000	1	23632
11	13	001	2	11816
11	13	010	4	5908
11	13	011	8	2954
11	13	100	16	1477
11	13	101	32	739
11	13	110	64	369
11	13	111	128	185

12.9.8 IRSCI Infrared Control Register

The infrared control register contains the control bits for the infrared sub-module.

- Enables the infrared sub-module
- Selects the infrared transmitter narrow pulse width

Address: $0047

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R	0	0	0	R	TNP1	TNP0	IREN
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 12-20. IRSCI Infrared Control Register (IRSCIRCR)

TNP1 and TNP0 — Transmitter Narrow Pulse Bits

These read/write bits select the infrared transmitter narrow pulse width as shown in Table 12-10. Reset clears TNP1 and TNP0.

Table 12-10. Infrared Narrow Pulse Selection

TNP1 and TNP0	Prescaler Divisor (PD)
00	SCI transmits a 3/16 narrow pulse
01	SCI transmits a 1/16 narrow pulse
10	SCI transmits a 1/32 narrow pulse
11	

IREN — Infrared Enable Bit

This read/write bit enables the infrared sub-module for encoding and decoding the SCI data stream. When this bit is clear, the infrared sub-module is disabled. Reset clears the IREN bit.

1 = infrared sub-module enabled
0 = infrared sub-module disabled

Chapter 13
Serial Peripheral Interface Module (SPI)

13.1 Introduction

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

13.2 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

13.3 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 13-1. The generic pin names appear in the text that follows.

Table 13-1. Pin Name Conventions

SPI Generic Pin Names:		MISO	MOSI	$\overline{SS}$	SPSCK	CGND
Full SPI Pin Names:	SPI	PTC2/MISO	PTC3/MOSI	PTC4/$\overline{SS}$	PTC5/SPSCK	V_{SS}

Figure 13-1 summarizes the SPI I/O registers.

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

= Unimplemented R = Reserved

Figure 13-1. SPI I/O Register Summary

13.4 Functional Description

Figure 13-2 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.

13.4.1 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 13.13.1 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 13-3.)

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See 13.13.2 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.

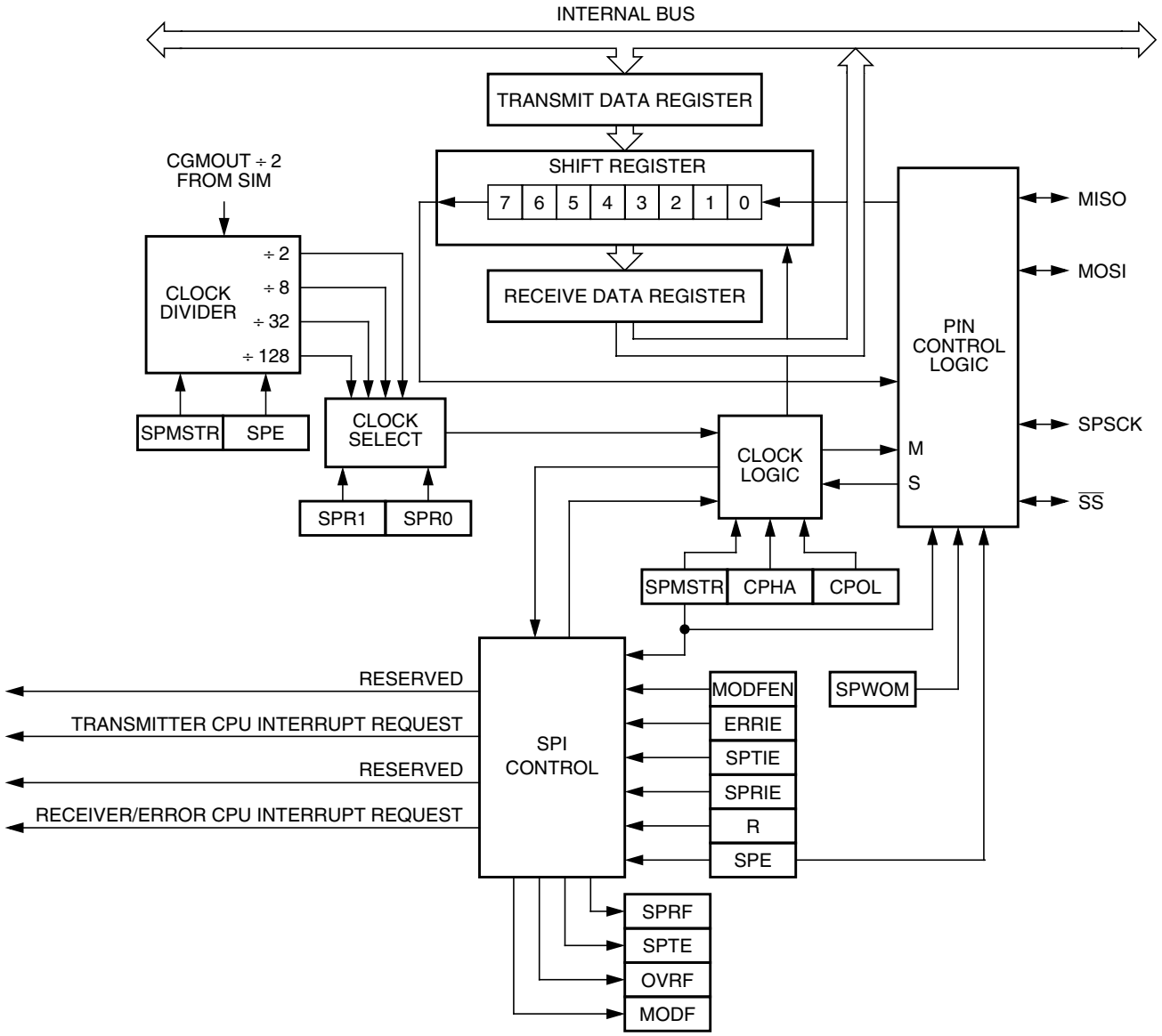


Figure 13-2. SPI Module Block Diagram

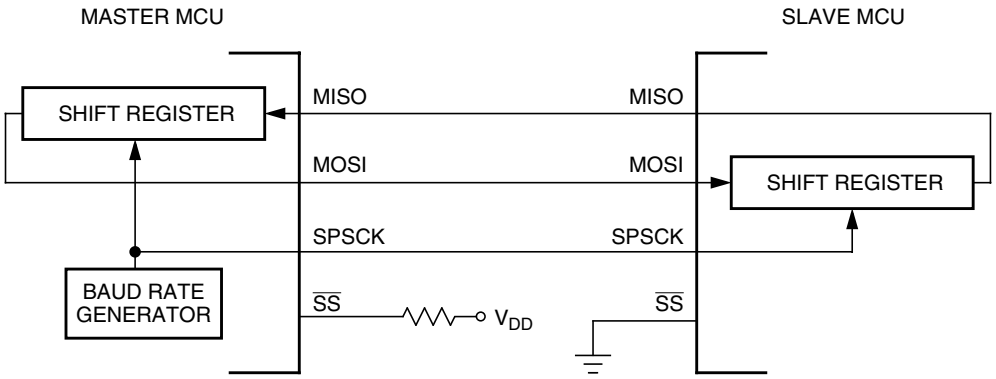


Figure 13-3. Full-Duplex Master-Slave Connections

13.4.2 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 13.7.2 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 13.5 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.

13.5 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.

13.5.1 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).

13.5.2 Transmission Format When CPHA = 0

Figure 13-4 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 13.7.2 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 13-5.

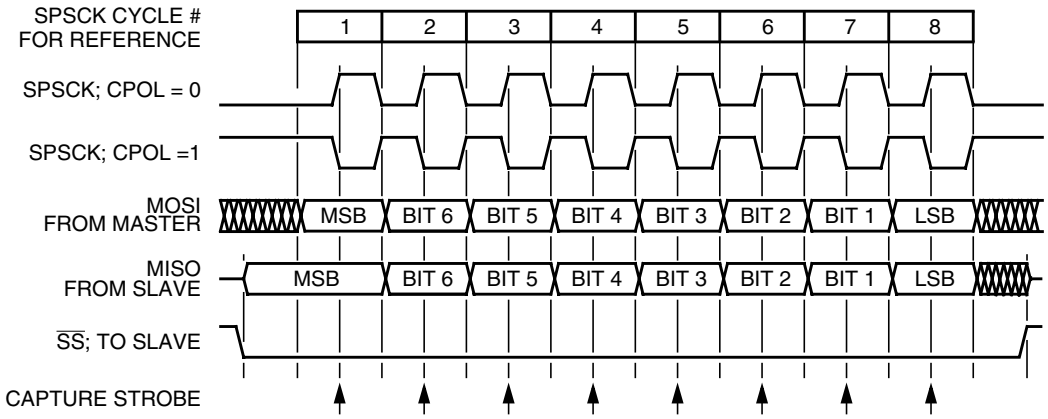


Figure 13-4. Transmission Format (CPHA = 0)

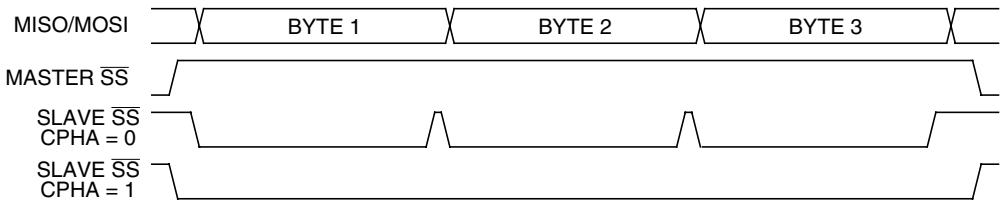


Figure 13-5. CPHA/$\overline{SS}$ Timing

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.

13.5.3 Transmission Format When CPHA = 1

Figure 13-6 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 13.7.2 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.

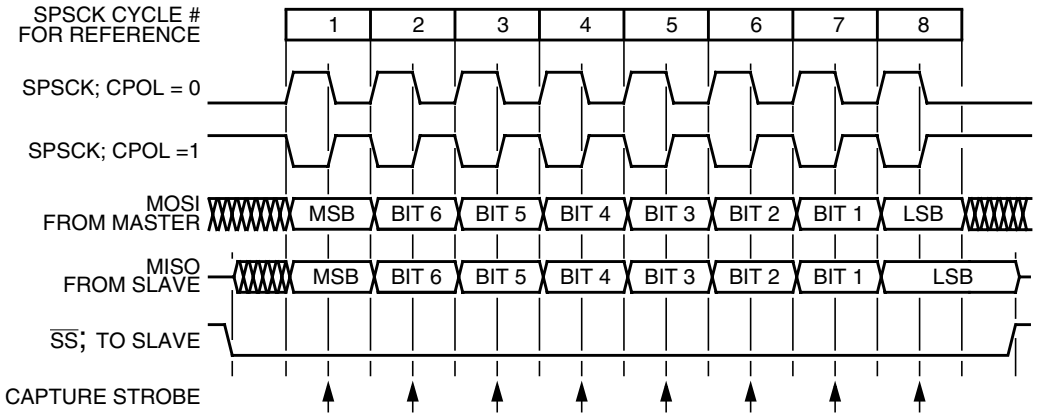


Figure 13-6. Transmission Format (CPHA = 1)

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.

13.5.4 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 13-7.) 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 13-7. 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.

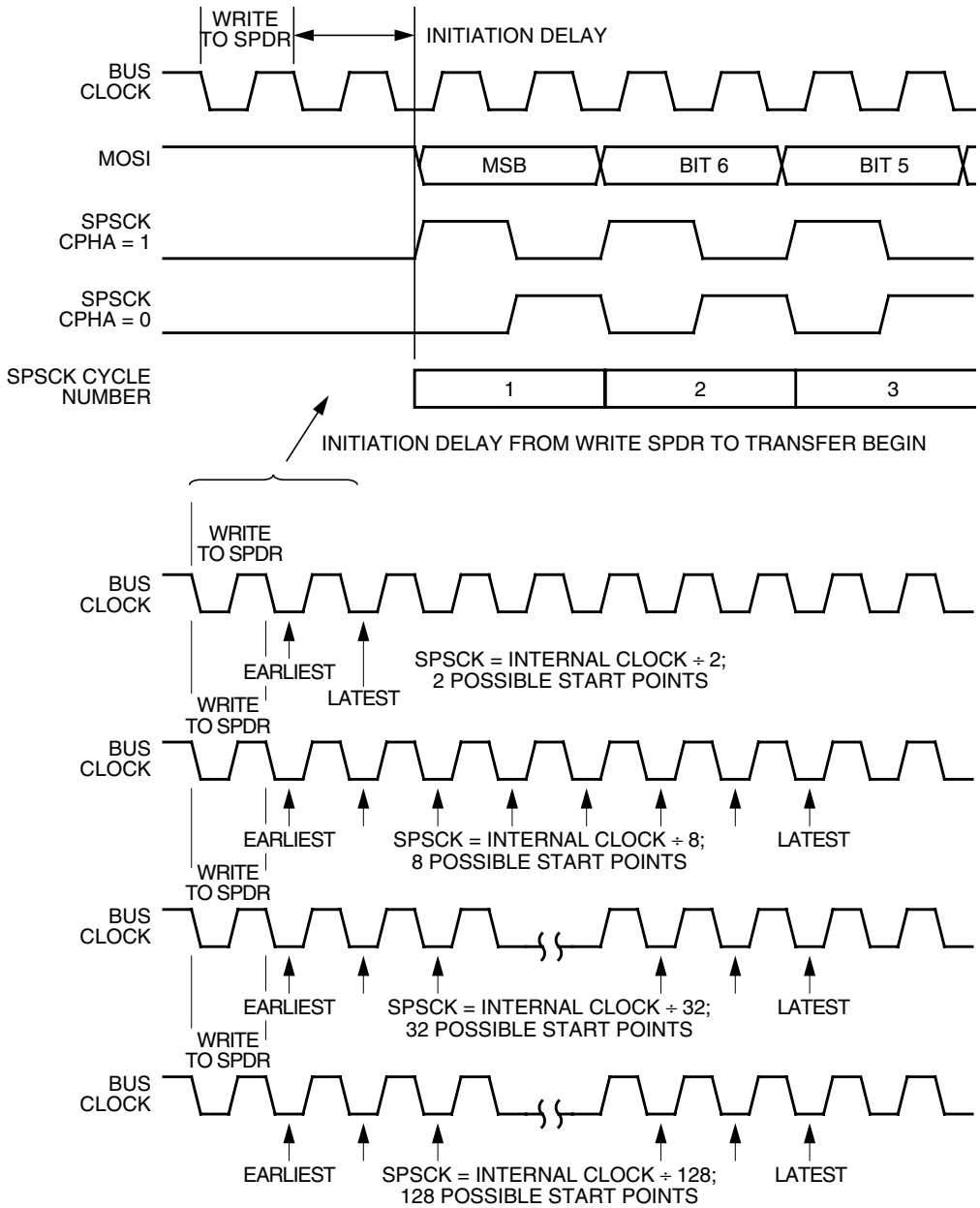


Figure 13-7. Transmission Start Delay (Master)

13.6 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 13-8 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA: CPOL = 1:0).

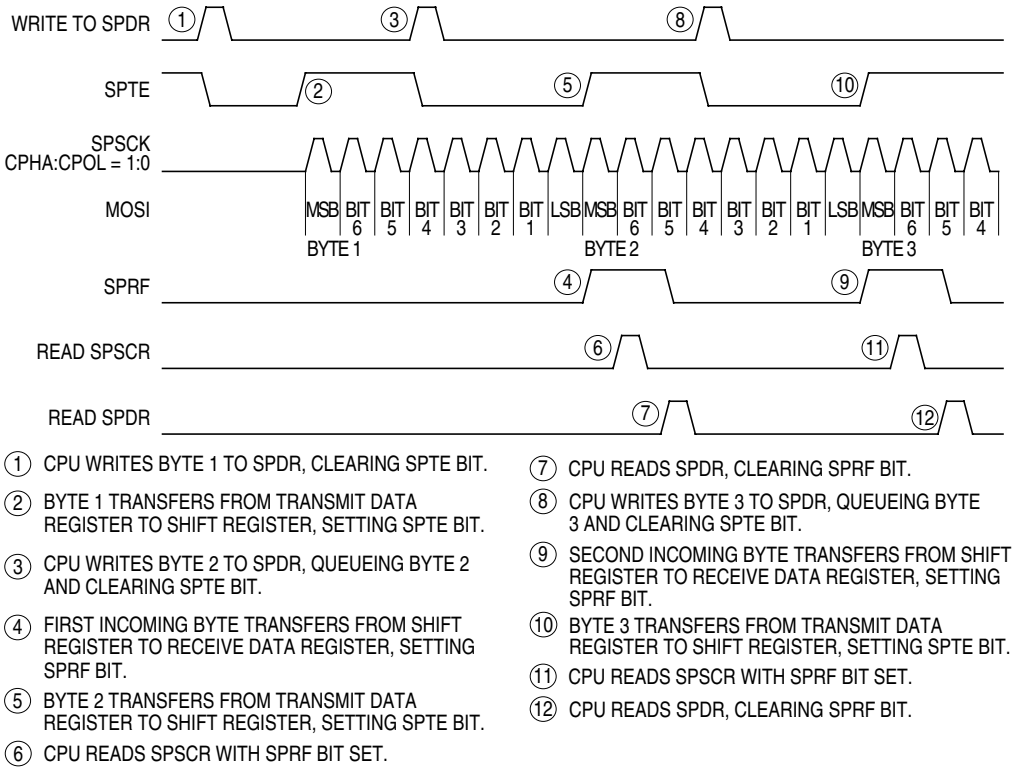


Figure 13-8. SPRF/SPTE CPU Interrupt Timing

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.

13.7 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.

13.7.1 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 13-4 and Figure 13-6.) 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 13-11.) 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 13-9 shows how it is possible to miss an overflow. The first part of Figure 13-9 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.

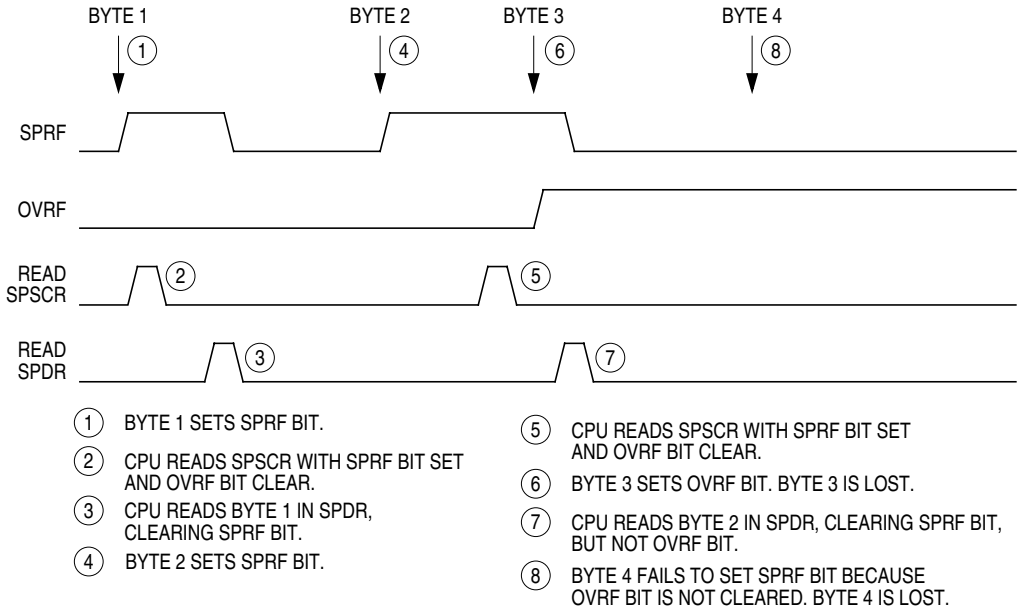


Figure 13-9. Missed Read of Overflow Condition

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 13-10 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit.

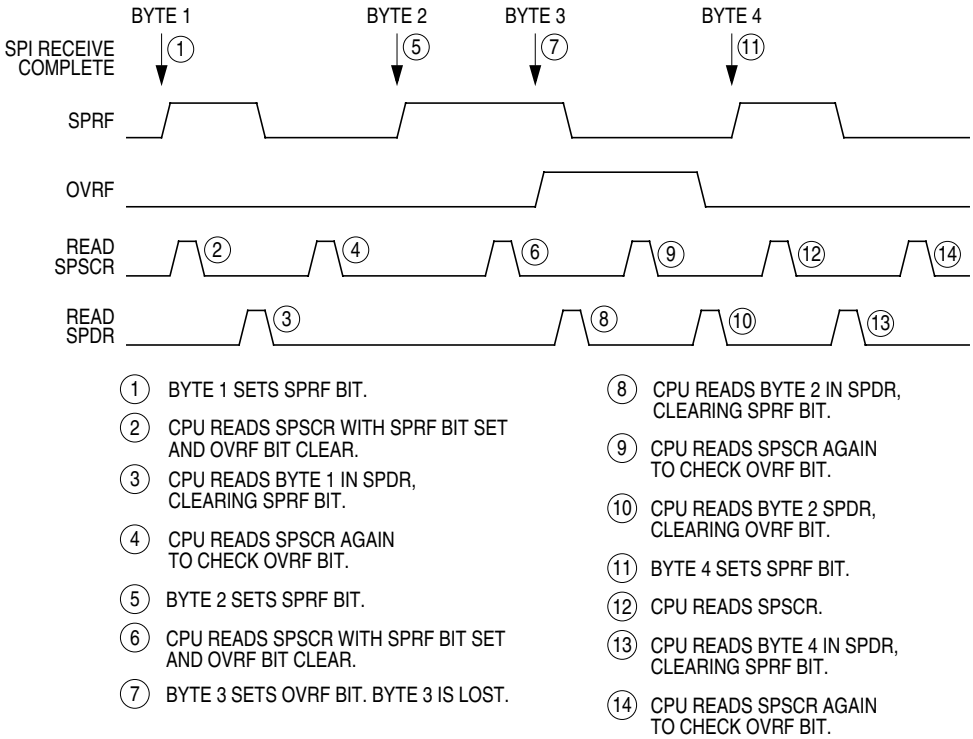


Figure 13-10. Clearing SPRF When OVRF Interrupt Is Not Enabled

13.7.2 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 13-11.) 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 13.5 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.

13.8 Interrupts

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

Table 13-2. 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 13-11.)

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.

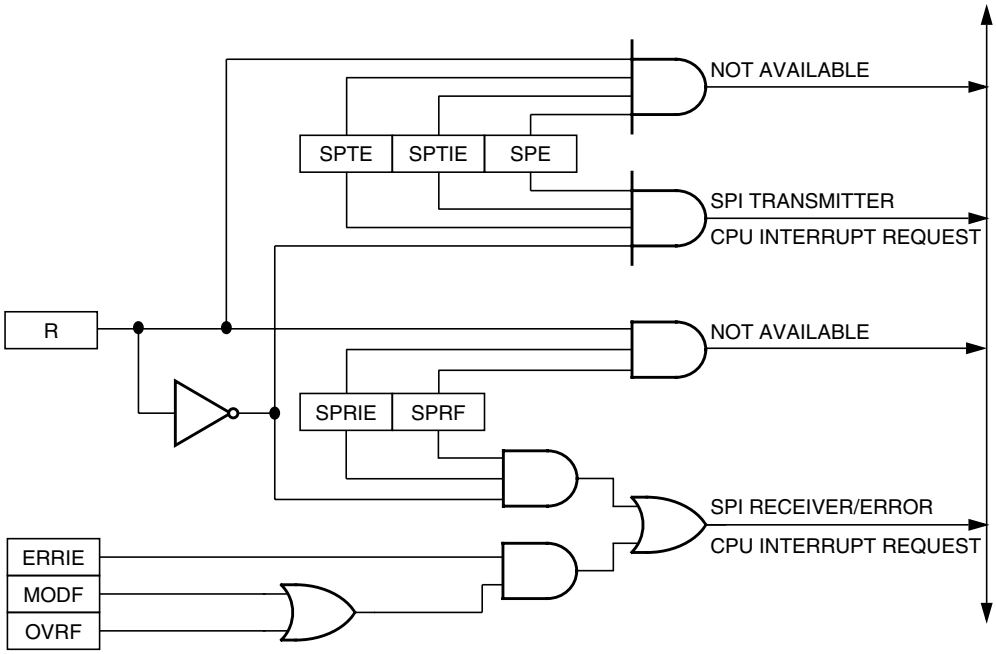


Figure 13-11. SPI Interrupt Request Generation

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.

13.9 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.

13.10 Low-Power Modes

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

13.10.1 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 13.8 Interrupts.)

13.10.2 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.

13.11 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 7 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.

13.12 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}.

13.12.1 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.

13.12.2 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.

13.12.3 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.

13.12.4 $\overline{SS}$ (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 13.5 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 13-12.

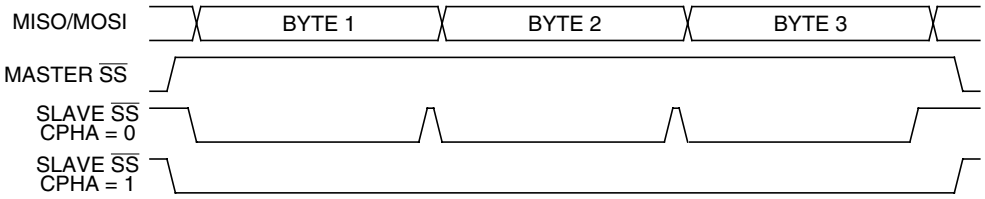


Figure 13-12. CPHA/$\overline{SS}$ Timing

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 13.13.2 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 13.7.2 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 13-3.)

Table 13-3. 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

13.12.5 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 13-1.

13.13 I/O Registers

Three registers control and monitor SPI operation:

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

13.13.1 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: $0010

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

= Unimplemented R = Reserved

Figure 13-13. 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 13-4 and Figure 13-6.) 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 13-4 and Figure 13-6.) 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 13-12.) 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 13.9 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

13.13.2 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 $0011

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

= Unimplemented

Figure 13-14. 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 13.12.4 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 13.7.2 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 13-4. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

Table 13-4. 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

13.13.3 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 13-2.)

Address: $0012

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

Figure 13-15. 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 14
Multi-Master IIC Interface (MMIIC)

14.1 Introduction

The multi-master IIC (MMIIC) interface is a two wire, bidirectional serial bus which provides a simple, efficient way for data exchange between devices. The interface is designed for internal serial communication between the MCU and other IIC devices. It has hardware generated START and STOP signals; and byte by byte interrupt driven software algorithm.

This bus is suitable for applications which need frequent communications over a short distance between a number of devices. It also provides a flexibility that allows additional devices to be connected to the bus. The maximum data rate is 100k-bps, and the maximum communication distance and number of devices that can be connected is limited by a maximum bus capacitance of 400pF.

This MMIIC interface is also SMBus (System Management Bus) version 1.0 and 1.1 compatible, with hardware cyclic redundancy code (CRC) generation, making it suitable for smart battery applications.

14.2 Features

Features of the MMIC module include:

- Full SMBus version 1.0/1.1 compliance
- Multi-master IIC bus standard
- Software programmable for one of eight different serial clock frequencies
- Software controllable acknowledge bit generation
- Interrupt driven byte by byte data transfer
- Calling address identification interrupt
- Arbitration loss detection and no-ACK awareness in master mode and automatic mode switching from master to slave
- Auto detection of R/W bit and switching of transmit or receive mode accordingly
- Detection of START, repeated START, and STOP signals
- Auto generation of START and STOP condition in master mode
- Repeated start generation
- Master clock generator with eight selectable baud rates
- Automatic recognition of the received acknowledge bit
- Busy detection
- Software enabled 8-bit CRC generation/decoding

14.3 I/O Pins

The MMIIC module uses two I/O pins, shared with standard port I/O pins. The full name of the MMIIC I/O pins are listed in Table 14-1. The generic pin name appear in the text that follows.

The SDA and SDL pins are open-drain. When configured as general purpose output pins (PTB0 and PTB1), pullup resistors must be connected to these pins.

Table 14-1. Pin Name Conventions

MMIIC Generic Pin Names:	Full MCU Pin Names:	Pin Selected for MMIIC Function By:
SDA	PTB0/SDA	MMEN bit in MMCR1 ($0049)
SCL	PTB1/SCL	

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0048	MMIIC Address Register (MMADR)	Read: Write:	MMAD7	MMAD6	MMAD5	MMAD4	MMAD3	MMAD2	MMAD1	MMEXTAD
		Reset:	1	0	1	0	0	0	0	0
$0049	MMIIC Control Register 1 (MMCR1)	Read:	MMEN	MMIEN	0	0	MMTXAK	REPSEN	MMCRCBYTE	0
		Write:			MMCLRBB					
		Reset:	0	0	0	0	0	0	0	0
$004A	MMIIC Control Register 2 (MMCR2)	Read:	MMALIF	MMNAKIF	MMBB	MMAST	MMRW	0	0	MMCRCEF
		Write:	0	0						
		Reset:	0	0	0	0	0	0	0	Unaffected
$004B	MMIIC Status Register (MMSR)	Read:	MMRXIF	MMTXIF	MMATCH	MMSRW	MMRXAK	MMCRCBF	MMTXBE	MMRXBF
		Write:	0	0						
		Reset:	0	0	0	0	1	0	1	0
$004C	MMIIC Data Transmit Register (MMDTR)	Read: Write:	MMTD7	MMTD6	MMTD5	MMTD4	MMTD3	MMTD2	MMTD1	MMTD0
		Reset:	0	0	0	0	0	0	0	0
$004D	MMIIC Data Receive Register (MDDRR)	Read:	MMRD7	MMRD6	MMRD5	MMRD4	MMRD3	MMRD2	MMRD1	MMRD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004E	MMIIC CRC Data Register (MMCRDR)	Read:	MMCRCD7	MMCRCD6	MMCRCD5	MMCRCD4	MMCRCD3	MMCRCD2	MMCRCD1	MMCRCD0
		Write:								
		Reset:	0	0	0	0	0	0	0	0
$004F	MMIIC Frequency Divider Register (MMFDR)	Read:	0	0	0	0	0	MMBR2	MMBR1	MMBR0
		Write:								
		Reset:	0	0	0	0	0	1	0	0

= Unimplemented

Figure 14-1. MMIIC I/O Register Summary

14.4 Multi-Master IIC System Configuration

The multi-master IIC system uses a serial data line SDA and a serial clock line SCL for data transfer. All devices connected to it must have open collector (drain) outputs and the logical-AND function is performed on both lines by two pull-up resistors.

14.5 Multi-Master IIC Bus Protocol

Normally a standard communication is composed of four parts:

1. START signal,
2. slave address transmission,
3. data transfer, and
4. STOP signal.

These are described briefly in the following sections and illustrated in Figure 14-2.

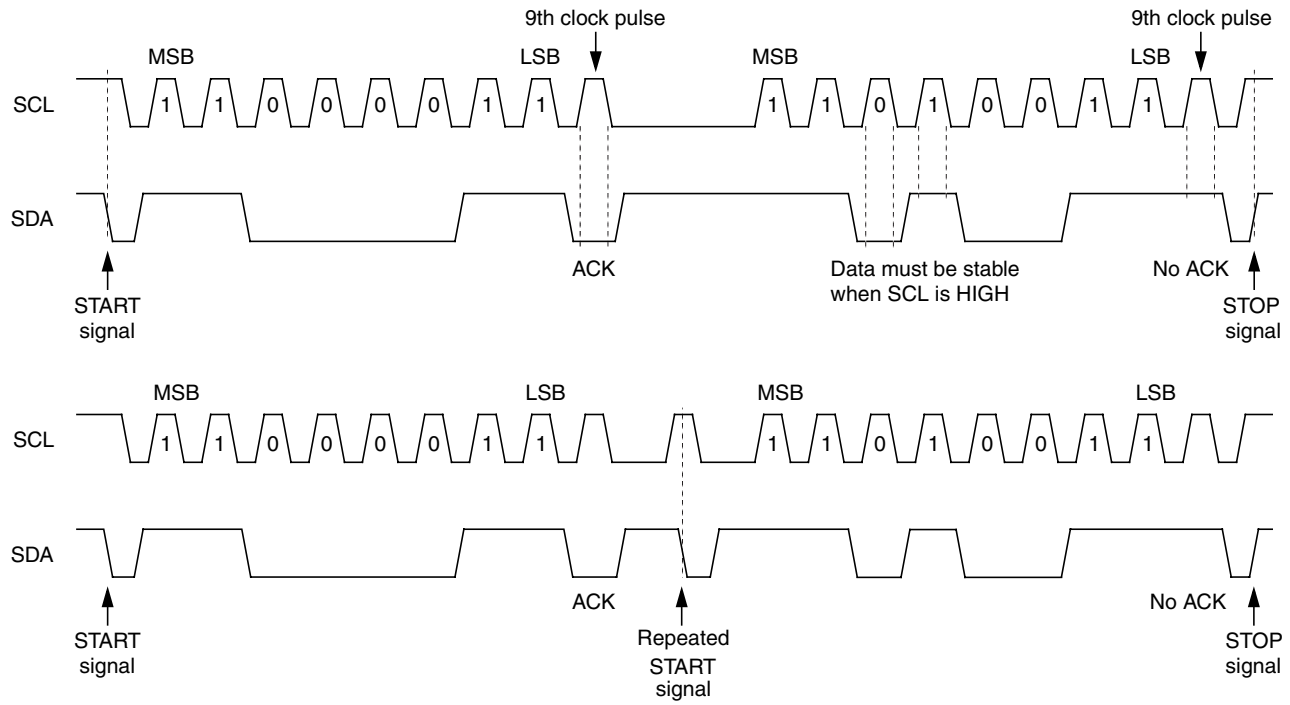


Figure 14-2. Multi-Master IIC Bus Transmission Signal Diagram

14.5.1 START Signal

When the bus is free, (i.e. no master device is engaging the bus — both SCL and SDA lines are at logic high) a master may initiate communication by sending a START signal. As shown in Figure 14-2, a START signal is defined as a high to low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and wakes up all slaves.

14.5.2 Slave Address Transmission

The first byte transferred immediately after the START signal is the slave address transmitted by the master. This is a 7-bit calling address followed by a R/W-bit. The R/W-bit dictates to the slave the desired direction of the data transfer. A logic 0 indicates that the master wishes to transmit data to the slave; a logic 1 indicates that the master wishes to receive data from the slave.

Only the slave with a matched address will respond by sending back an acknowledge bit by pulling SDA low on the 9th clock cycle.
(See Figure 14-2.)

14.5.3 Data Transfer

Once a successful slave addressing is achieved, the data transfer can proceed byte by byte in the direction specified by the R/W-bit sent by the calling master.

Each data byte is 8 bits. Data can be changed only when SCL is low and must be held stable when SCL is high as shown in Figure 14-2. The MSB is transmitted first and each byte has to be followed by an acknowledge bit. This is signalled by the receiving device by pulling the SDA low on the 9th clock cycle. Therefore, one complete data byte transfer requires 9 clock cycles.

If the slave receiver does not acknowledge the master, the SDA line should be left high by the slave. The master can then generate a STOP signal to abort the data transfer or a START signal (repeated START) to commence a new transfer.

If the master receiver does not acknowledge the slave transmitter after a byte has been transmitted, it means an “end of data” to the slave. The slave should release the SDA line for the master to generate a STOP or START signal.

14.5.4 Repeated START Signal

As shown in Figure 14-2, a repeated START signal is used to generate START signal without first generating a STOP to terminate the communication. This is used by the master to communicate with another slave or with the same slave in a different mode (transmit/receive mode) without releasing the bus.

14.5.5 STOP Signal

The master can terminate the communication by generating a STOP signal to free the bus. However, the master may generate a START signal followed by a calling command without first generating a STOP signal. This is called repeat START. A STOP signal is defined as a low to high transition of SDA while SCL is at logic high (see Figure 14-2).

14.5.6 Arbitration Procedure

The interface circuit is a multi-master system which allows more than one master to be connected. If two or more masters try to control the bus at the same time, a clock synchronization procedure determines the bus clock. The clock low period is equal to the longest clock low period and the clock high period is equal to the shortest one among the masters. A data arbitration procedure determines the priority. A master will lose arbitration if it transmits a logic 1 while another transmits a logic 0. The losing master will immediately switch over to slave receive mode and stops its data and clock outputs. The transition from master to slave will not generate a STOP condition. Meanwhile a software bit will be set by hardware to indicates loss of arbitration.

14.5.7 Clock Synchronization

Since wired-AND logic is performed on SCL line, a high to low transition on the SCL line will affect the devices connected to the bus. The devices start counting their low period once a device’s clock has gone low, it will hold the SCL line low until the clock high state is reached. However, the change of low to high

in this device clock may not change the state of the SCL line if another device clock is still in its low period. Therefore the synchronized clock SCL will be held low by the device which last releases SCL to logic high. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized SCL line will be released and go high, and all devices will start counting their high periods. The first device to complete its high period will again pull the SCL line low. Figure 14-3 illustrates the clock synchronization waveforms.

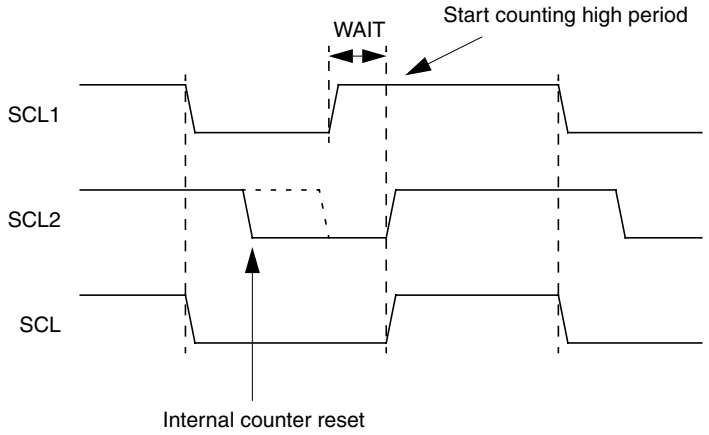


Figure 14-3. Clock Synchronization

14.5.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. A slave device may hold the SCL low after completion of one byte data transfer and will halt the bus clock, forcing the master clock into a wait state until the slave releases the SCL line.

14.5.9 Packet Error Code

The packet error code (PEC) for the MMIIC interface is in the form a cyclic redundancy code (CRC). The PEC is generated by hardware for every transmitted and received byte of data. The transmission of the generated PEC is controlled by user software.

The CRC data register, MMCRCDR, contains the generated PEC byte, with three other bits in the MMIIC control registers and status register monitoring and controlling the PEC byte.

14.6 MMIIC I/O Registers

These I/O registers control and monitor MMIIC operation:

- MMIIC address register (MMADR) — $0048
- MMIIC control register 1 (MMCR1) — $0049
- MMIIC control register 2 (MMCR2) — $004A
- MMIIC status register (MMSR) — $004B
- MMIIC data transmit register (MMDTR) — $004C
- MMIIC data receive register (MMDRR) — $004D
- MMIIC CRC data register (MMCRCDR) — $004E
- MMIIC frequency divide register (MMFDR) — $004F

14.6.1 MMIIC Address Register (MMADR)

Address: $0048

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	MMAD7	MMAD6	MMAD5	MMAD4	MMAD3	MMAD2	MMAD1	MMEXTAD
Reset:	1	0	1	0	0	0	0	0

Figure 14-4. MMIIC Address Register (MMADR)

MMAD[7:1] — Multi-Master Address

These seven bits represent the MMIIC interface's own specific slave address when in slave mode, and the calling address when in master mode. Software must update MMAD[7:1] as the calling address while entering master mode and restore its own slave address after master mode is relinquished. This register is cleared as $A0 upon reset.

MMEXTAD — Multi-Master Expanded Address

This bit is set to expand the address of the MMIIC in slave mode. When set, the MMIIC will acknowledge the following addresses from a calling master: $MMAD[7:1], 0000000, and 0001100. Reset clears this bit.

1 = MMIIC responds to the following calling addresses:
$MMAD[7:1], 0000000, and 0001100.

0 = MMIIC responds to address $MMAD[7:1]

For example, when MMADR is configured as:

MMAD7	MMAD6	MMAD5	MMAD4	MMAD3	MMAD2	MMAD1	MMEXTAD
1	1	0	1	0	1	0	1

The MMIIC module will respond to the calling address:

Bit 7	6	5	4	3	2	Bit 1
1	1	0	1	0	1	0

or the general calling address:

0	0	0	0	0	0	0

or the calling address:

Bit 7	6	5	4	3	2	Bit 1
0	0	0	1	1	0	0

Note that bit-0 of the 8-bit calling address is the MMRW bit from the calling master.

14.6.2 MMIIC Control Register 1 (MMCR1)

Address: $0049

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MMEN	MMIEN	0	0	MMTXAK	REPSEN	MMCRCBYTE	0
Write:			MMCLRBB					
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-5. MMIIC Control Register 1 (MMCR1)

MMEN — MMIIC Enable

This bit is set to enable the Multi-master IIC module. When MMEN = 0, module is disabled and all flags will restore to its power-on default states. Reset clears this bit.

1 = MMIIC module enabled
0 = MMIIC module disabled

MMIEN — MMIIC Interrupt Enable

When this bit is set, the MMTXIF, MMRXIF, MMALIF, and MMNAKIF flags are enabled to generate an interrupt request to the CPU. When MMIEN is cleared, the these flags are prevented from generating an interrupt request. Reset clears this bit.

1 = MMTXIF, MMRXIF, MMALIF, and/or MMNAKIF bit set will generate interrupt request to CPU
0 = MMTXIF, MMRXIF, MMALIF, and/or MMNAKIF bit set will not generate interrupt request to CPU

MMCLRBB — MMIIC Clear Busy Flag

Writing a logic 1 to this write-only bit clears the MMBB flag. MMCLRBB always reads as a logic 0. Reset clears this bit.

1 = Clear MMBB flag
0 = No affect on MMBB flag

MMTXAK — MMIIC Transmit Acknowledge Enable

This bit is set to disable the MMIIC from sending out an acknowledge signal to the bus at the 9th clock bit after receiving 8 data bits. When MMTXAK is cleared, an acknowledge signal will be sent at the 9th clock bit. Reset clears this bit.

1 = MMIIC does not send acknowledge signals at 9th clock bit
0 = MMIIC sends acknowledge signal at 9th clock bit

REPSEN — Repeated Start Enable

This bit is set to enable repeated START signal to be generated when in master mode transfer (MMAST = 1). The REPSEN bit is cleared by hardware after the completion of repeated START signal or when the MMAST bit is cleared. Reset clears this bit.

1 = Repeated START signal will be generated if MMAST bit is set
0 = No repeated START signal will be generated

MMCRCBYTE — MMIIC CRC Byte

In receive mode, this bit is set by software to indicate that the next receiving byte will be the packet error checking (PEC) data.

In master receive mode, after completion of CRC generation on the received PEC data, an acknowledge signal is sent if MMTXAK = 0; no acknowledge is sent If MMTXAK = 1.

In slave receive mode, no acknowledge signal is sent if a CRC error is detected on the received PEC data. If no CRC error is detected, an acknowledge signal is sent if MMTXAK = 0; no acknowledge is sent If MMTXAK = 1.

Under normal operation, the user software should clear MMTXAK bit before setting MMCRCBYTE bit to ensure that an acknowledge signal is sent when no CRC error is detected.
The MMCRCBYTE bit should not be set in transmit mode. This bit is cleared by the next START signal. Reset also clears this bit.

1 = Next receiving byte is the packet error checking (PEC) data
0 = Next receiving byte is not PEC data

14.6.3 MMIIC Control Register 2 (MMCR2)

Address: $004A

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MMALIF	MMNAKIF	MMBB	MMAST	MMRW	0	0	MMCRCEF
Write:	0	0						
Reset:	0	0	0	0	0	0	0	Unaffected

= Unimplemented

Figure 14-6. MMIIC Control Register 2 (MMCR2)

MMALIF — Arbitration Loss Interrupt Flag

This flag is set when software attempt to set MMAST but the MMBB has been set by detecting the start condition on the lines or when the MMIIC is transmitting a "1" to SDA line but detected a "0" from SDA line in master mode — an arbitration loss. This bit generates an interrupt request to the CPU if the MMIEN bit in MMCR1 is set. This bit is cleared by writing "0" to it or by reset.

1 = Lost arbitration in master mode
0 = No arbitration lost

MMNAKIF — No AcKnowledge Interrupt Flag (Master Mode)

This flag is only set in master mode (MMAST = 1) when there is no acknowledge bit detected after one data byte or calling address is transferred. This flag also clears MMAST. MMNAKIF generates an interrupt request to CPU if the MMIEN bit in MMCR1 is set. This bit is cleared by writing "0" to it or by reset.

1 = No acknowledge bit detected
0 = Acknowledge bit detected

MMBB — MMIIC Bus Busy Flag

This flag is set after a start condition is detected (bus busy), and is cleared when a stop condition (bus idle) is detected or the MMIIC is disabled. Reset clears this bit.

1 = Start condition detected
0 = Stop condition detected or MMIIC is disabled

MMAST — MMIIC Master Control

This bit is set to initiate a master mode transfer. In master mode, the module generates a start condition to the SDA and SCL lines, followed by sending the calling address stored in MMADR. When the MMAST bit is cleared by MMNAKIF set (no acknowledge) or by software, the module generates the stop condition to the lines after the current byte is transmitted.
If an arbitration loss occurs (MMALIF = 1), the module reverts to slave mode by clearing MMAST, and releasing SDA and SCL lines immediately.
This bit is cleared by writing "0" to it or by reset.

1 = Master mode operation
0 = Slave mode operation

MMRW — MMIIC Master Read/Write

This bit is transmitted out as bit 0 of the calling address when the module sets the MMAST bit to enter master mode. The MMRW bit determines the transfer direction of the data bytes that follows. When it is "1", the module is in master receive mode. When it is "0", the module is in master transmit mode. Reset clears this bit.

1 = Master mode receive
0 = Master mode transmit

MMCRCEF — MMIIC CRC Error Flag

This flag is set when a CRC error is detected, and cleared when no CRC error is detected. The MMCRCEF is only meaningful after receiving a PEC data. This flag is unaffected by reset.

1 = CRC error detected on PEC byte
0 = No CRC error detected on PEC byte

14.6.4 MMIIC Status Register (MMSR)

Address: $004B

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MMRXIF	MMTXIF	MMATCH	MMSRW	MMRXAK	MMCRCBF	MMTXBE	MMRXBF
Write:	0	0						
Reset:	0	0	0	0	1	0	1	0

= Unimplemented

Figure 14-7. MMIIC Status Register (MMSR)

MMRXIF — MMIIC Receive Interrupt Flag

This flag is set after the data receive register (MMDRR) is loaded with a new received data. Once the MMDRR is loaded with received data, no more received data can be loaded to the MMDRR register until the CPU reads the data from the MMDRR to clear MMRXBF flag. MMRXIF generates an interrupt request to CPU if the MMIEN bit in MMCR is also set. This bit is cleared by writing "0" to it or by reset; or when the MMEN = 0.

1 = New data in data receive register (MMDRR)
0 = No data received

MMTXIF — MMIIC Transmit Interrupt Flag

This flag is set when data in the data transmit register (MMDTR) is downloaded to the output circuit, and that new data can be written to the MMDTR. MMTXIF generates an interrupt request to CPU if the MMIEN bit in MMCR is also set. This bit is cleared by writing "0" to it or when the MMEN = 0.

1 = Data transfer completed
0 = Data transfer in progress

MMATCH — MMIIC Address Match Flag

This flag is set when the received data in the data receive register (MMDRR) is a calling address which matches with the address or its extended addresses (MMEXTAD = 1) specified in the address register (MMADR). The MMATCH flag is set at the 9th clock of the calling address and will be cleared on the 9th clock of the next receiving data. Note: slave transmits do not clear MMATCH.

1 = Received address matches MMADR
0 = Received address does not match

MMSRW — MMIIC Slave Read/Write Select

This bit indicates the data direction when the module is in slave mode. It is updated after the calling address is received from a master device. MMSRW = 1 when the calling master is reading data from the module (slave transmit mode). MMSRW = 0 when the master is writing data to the module (receive mode).

1 = Slave mode transmit
0 = Slave mode receive

MMRXAK — MMIIC Receive Acknowledge

When this bit is cleared, it indicates an acknowledge signal has been received after the completion of eight data bits transmission on the bus. When MMRXAK is set, it indicates no acknowledge signal has been detected at the 9th clock; the module will release the SDA line for the master to generate STOP or repeated START condition. Reset sets this bit.

1 = No acknowledge signal received at 9th clock
0 = Acknowledge signal received at 9th clock

MMCRCBF — CRC Data Buffer Full Flag

This flag is set when the CRC data register (MMCRCDR) is loaded with a CRC byte for the current received or transmitted data.

In transmit mode, after a byte of data has been sent (MMTXIF = 1), the MMCRCBF will be set when the CRC byte has been generated and ready in the MMCRCDR. The content of the MMCRCDR should be copied to the MMDTR for transmission.

In receive mode, the MMCRCBF is set when the CRC byte has been generated and ready in MMCRCDR, for the current byte of received data.

The MMCRCBF bit is cleared when the CRC data register is read. Reset also clears this bit.

1 = Data ready in CRC data register (MMCRCDR)
0 = Data not ready in CRC data register (MMCRCDR)

MMTXBE — MMIIC Transmit Buffer Empty

This flag indicates the status of the data transmit register (MMDTR). When the CPU writes the data to the MMDTR, the MMTXBE flag will be cleared. MMTXBE is set when MMDTR is emptied by a transfer of its data to the output circuit. Reset sets this bit.

1 = Data transmit register empty
0 = Data transmit register full

MMRXBF — MMIIC Receive Buffer Full

This flag indicates the status of the data receive register (MMDRR). When the CPU reads the data from the MMDRR, the MMRXBF flag will be cleared. MMRXBF is set when MMDRR is full by a transfer of data from the input circuit to the MMDRR. Reset clears this bit.

1 = Data receive register full
0 = Data receive register empty

14.6.5 MMIIC Data Transmit Register (MMDTR)

Address: $004C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	MMTD7	MMTD6	MMTD5	MMTD4	MMTD3	MMTD2	MMTD1	MMTD0
Reset:	0	0	0	0	0	0	0	0

Figure 14-8. MMIIC Data Transmit Register (MMDTR)

When the MMIIC module is enabled, MMEN = 1, data written into this register depends on whether module is in master or slave mode.

In slave mode, the data in MMDTR will be transferred to the output circuit when:

- the module detects a matched calling address (MMATCH = 1), with the calling master requesting data (MMSRW = 1); or
- the previous data in the output circuit has be transmitted and the receiving master returns an acknowledge bit, indicated by a received acknowledge bit (MMRXAK = 0).

If the calling master does not return an acknowledge bit (MMRXAK = 1), the module will release the SDA line for master to generate a STOP or repeated START condition. The data in the MMDTR will not be transferred to the output circuit until the next calling from a master. The transmit buffer empty flag remains cleared (MMTXBE = 0).

In master mode, the data in MMDTR will be transferred to the output circuit when:

- the module receives an acknowledge bit (MMRXAK = 0), after setting master transmit mode (MMRW = 0), and the calling address has been transmitted; or
- the previous data in the output circuit has be transmitted and the receiving slave returns an acknowledge bit, indicated by a received acknowledge bit (MMRXAK = 0).

If the slave does not return an acknowledge bit (MMRXAK = 1), the master will generate a STOP or repeated START condition. The data in the MMDTR will not be transferred to the output circuit. The transmit buffer empty flag remains cleared (MMTXBE = 0).

The sequence of events for slave transmit and master transmit are illustrated in Figure 14-12.

14.6.6 MMIIC Data Receive Register (MMDRR)

Address: $004D

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MMRD7	MMRD6	MMRD5	MMRD4	MMRD3	MMRD2	MMRD1	MMRD0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-9. MMIIC Data Receive Register (MMDRR)

When the MMIIC module is enabled, MMEN = 1, data in this read-only register depends on whether module is in master or slave mode.

In slave mode, the data in MMDRR is:

- the calling address from the master when the address match flag is set (MMATCH = 1); or
- the last data received when MMATCH = 0.

In master mode, the data in the MMDRR is:

- the last data received.

When the MMDRR is read by the CPU, the receive buffer full flag is cleared (MMRXBF = 0), and the next received data is loaded to the MMDRR. Each time when new data is loaded to the MMDRR, the MMRXIF interrupt flag is set, indicating that new data is available in MMDRR.

The sequence of events for slave receive and master receive are illustrated in Figure 14-12.

14.6.7 MMIIC CRC Data Register (MMCRCDR)

Address: $004E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	MMCRCD7	MMCRCD6	MMCRCD5	MMCRCD4	MMCRCD3	MMCRCD2	MMCRCD1	MMCRCD0
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 14-10. MMIIC CRC Data Register (MMCRCDR)

When the MMIIC module is enabled, MMEN = 1, and the CRC buffer full flag is set (MMCRCBF = 1), data in this read-only register contains the generated CRC byte for the last byte of received or transmitted data.

A CRC byte is generated for each received and transmitted data byte and loaded to the CRC data register. The MMCRCBF bit will be set to indicate the CRC byte is ready in the CRC data register.

Reading the CRC data register clears the MMCRCBF bit. If the CRC data register is not read, the MMCRCBF bit will be cleared by hardware before the next CRC byte is loaded.

14.6.8 MMIIC Frequency Divider Register (MMFDR)

Address: $004F

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	0	MMBR2	MMBR1	MMBR0
Write:								
Reset:	0	0	0	0	0	1	0	0

= Unimplemented

Figure 14-11. MMIIC Frequency Divider Register (MMFDR)

The three bits in the frequency divider register (MMFDR) selects the divider to divide the bus clock to the desired baud rate for the MMIIC data transfer.

Table 14-2 shows the divider values for MMBR[2:0].

Table 14-2. MMIIC Baud Rate Selection

MMBR2	MMBR1	MMBR0	Divider	MMIIC Baud Rates for Bus Clocks:			
				8MHz	4MHz	2MHz	1MHz
0	0	0	20	400kHz	200kHz	100kHz	50kHz
0	0	1	40	200kHz	100kHz	50kHz	25kHz
0	1	0	80	100kHz	50kHz	25kHz	12.5kHz
0	1	1	160	50kHz	25kHz	12.5kHz	6.25kHz
1	0	0	320	25kHz	12.5kHz	6.25kHz	3.125kHz
1	0	1	640	12.5kHz	6.25kHz	3.125kHz	1.5625kHz
1	1	0	1280	6.25kHz	3.125kHz	1.5625kHz	0.78125kHz
1	1	1	2560	3.125kHz	1.5625kHz	0.78125kHz	0.3906kHz

NOTE

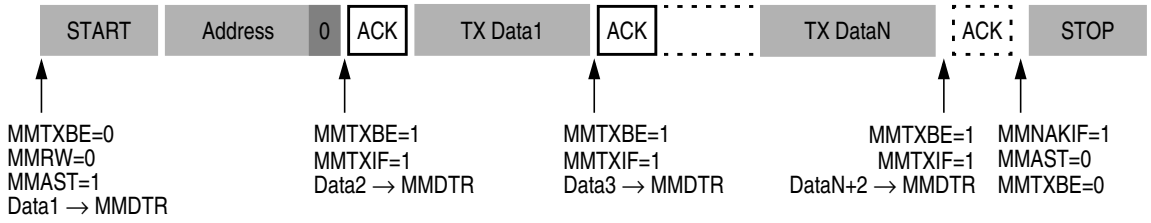
The frequency of the MMIIC baud rate is only guaranteed for 100kHz to 10kHz. The divider is available for the flexibility on bus frequency selection.

14.7 Program Algorithm

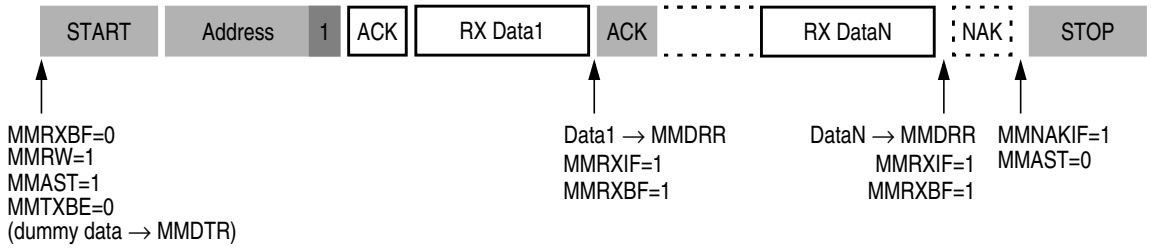
When the MMIIC module detects an arbitration loss in master mode, it releases both SDA and SCL lines immediately. But if there are no further STOP conditions detected, the module will hang up. Therefore, it is recommended to have time-out software to recover from this condition. The software can start the time-out counter by looking at the MMBB (bus busy) flag and reset the counter on the completion of one byte transmission. If a time-out has occurred, software can clear the MMEN bit (disable MMIIC module) to release the bus, and hence clear the MMBB flag. This is the only way to clear the MMBB flag by software if the module hangs up due to a no STOP condition received. The MMIIC can resume operation again by setting the MMEN bit.

14.7.1 Data Sequence

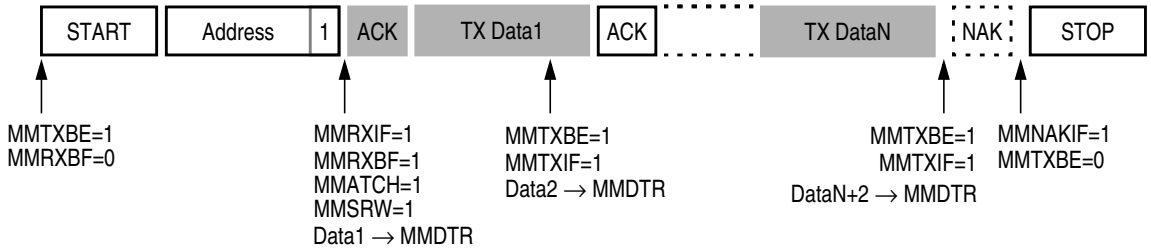
(a) Master Transmit Mode



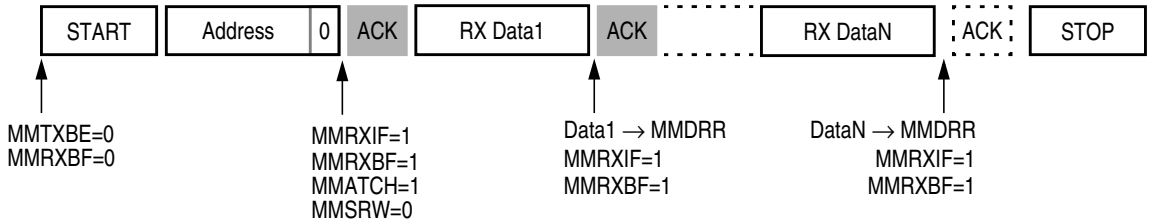
(b) Master Receive Mode



(c) Slave Transmit Mode



(d) Slave Receive Mode



Shaded data packets indicate transmissions by the MCU

Figure 14-12. Data Transfer Sequences for Master/Slave Transmit/Receive Modes

14.8 SMBus Protocols with PEC and without PEC

Following is a description of the various MMIIC bus protocols with and without a packet error code (PEC).

14.8.1 Quick Command

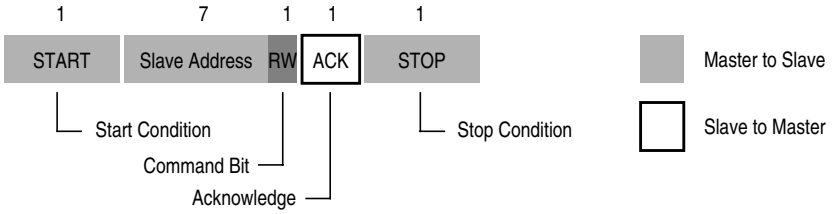


Figure 14-13. Quick Command

14.8.2 Send Byte

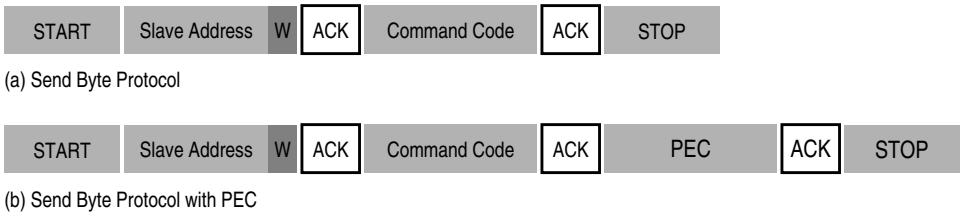


Figure 14-14. Send Byte

14.8.3 Receive Byte

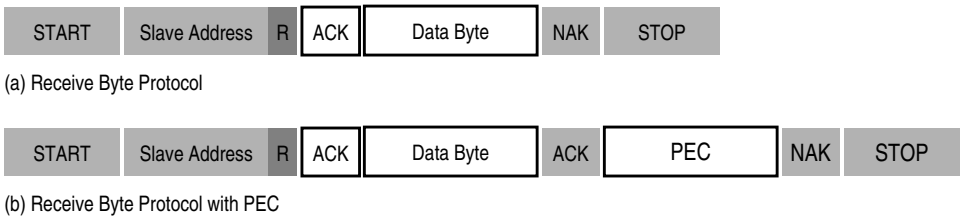


Figure 14-15. Receive Byte

14.8.4 Write Byte/Word

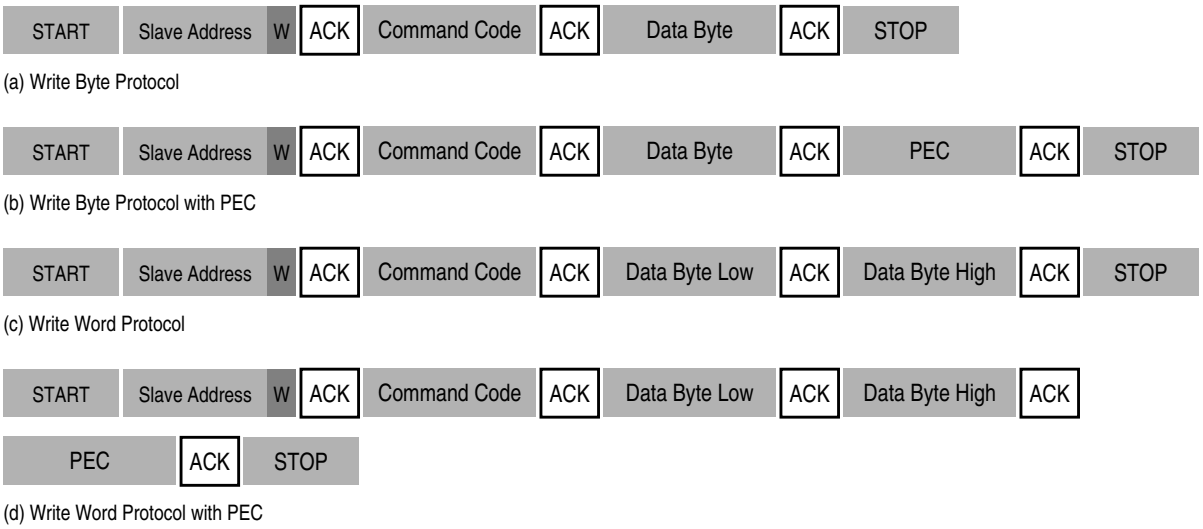


Figure 14-16. Write Byte/Word

14.8.5 Read Byte/Word

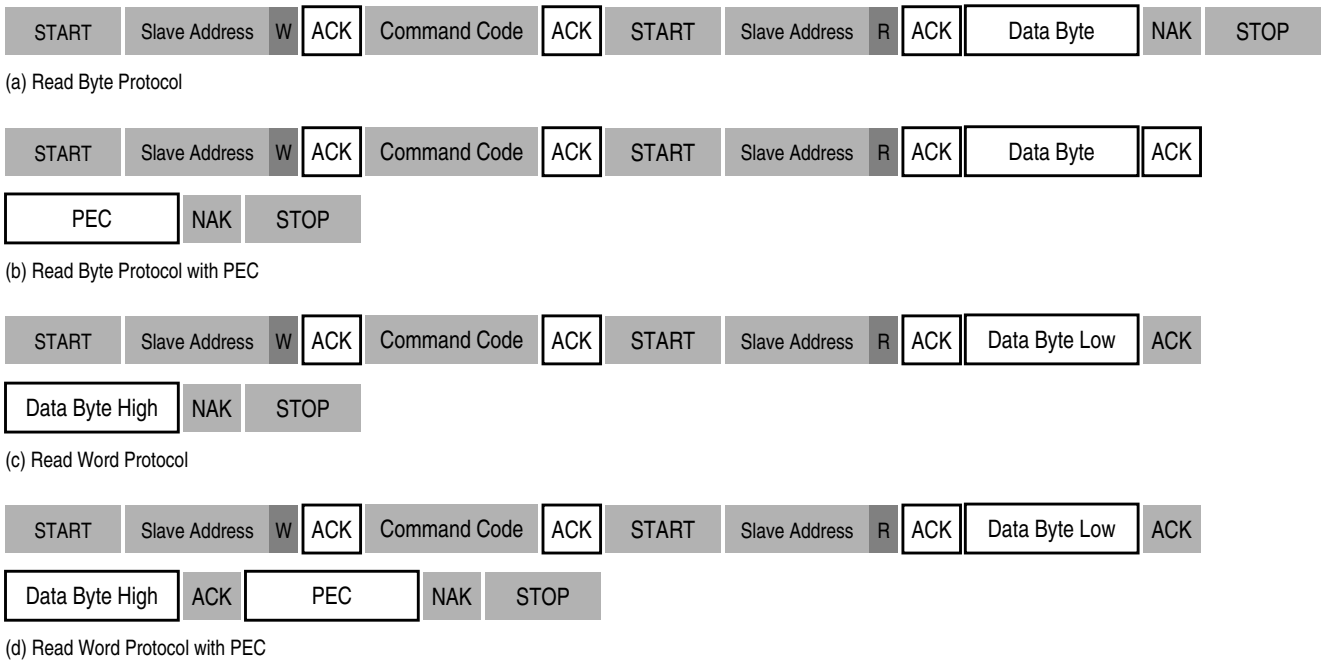


Figure 14-17. Read Byte/Word

14.8.6 Process Call

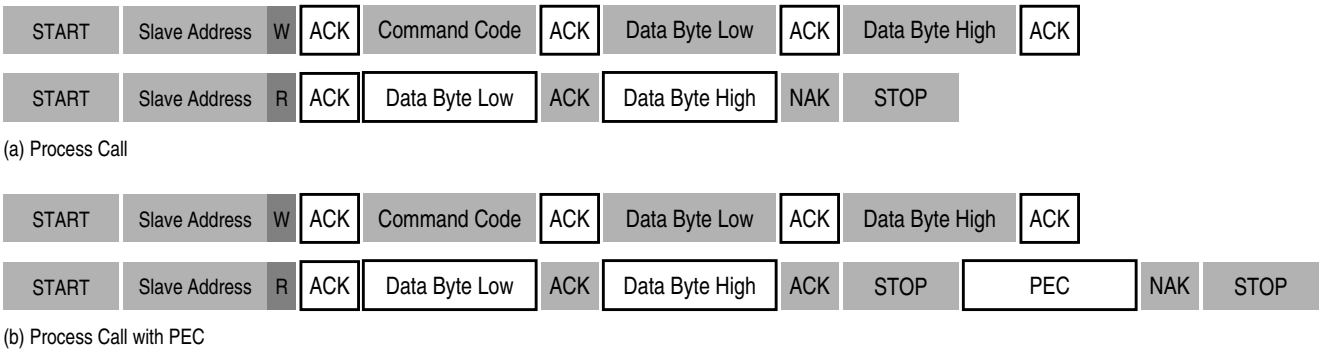


Figure 14-18. Process Call

14.8.7 Block Read/Write

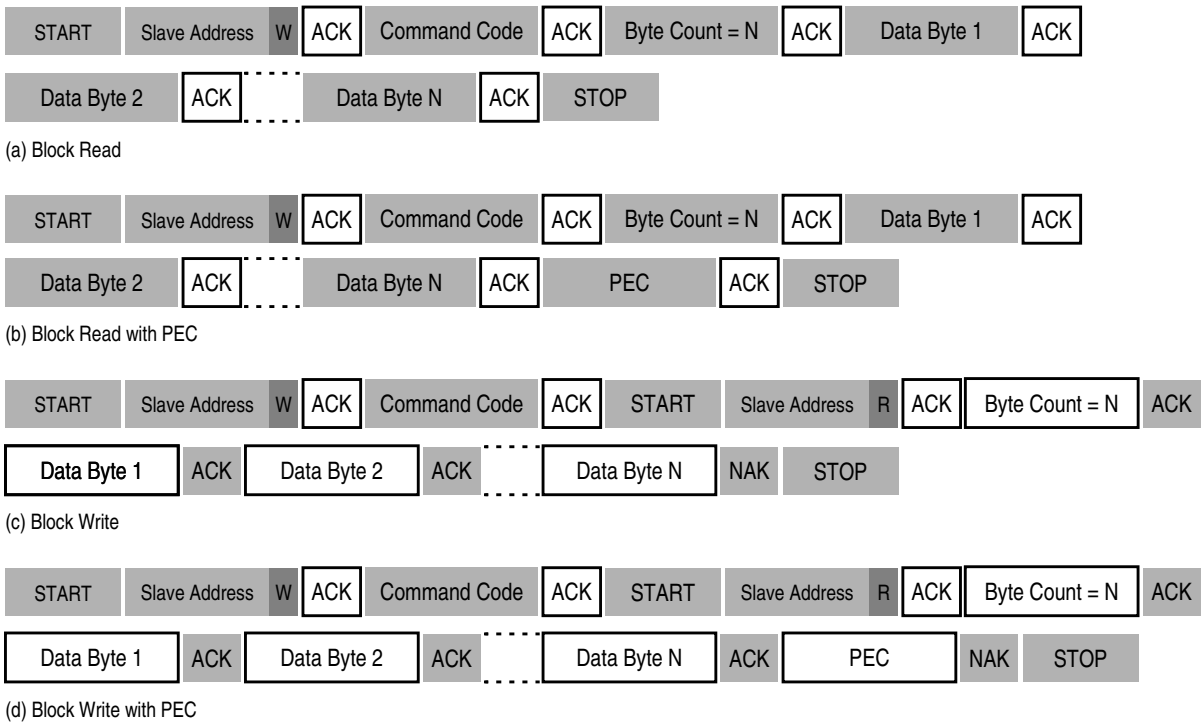


Figure 14-19. Block Read/Write

14.9 SMBus Protocol Implementation

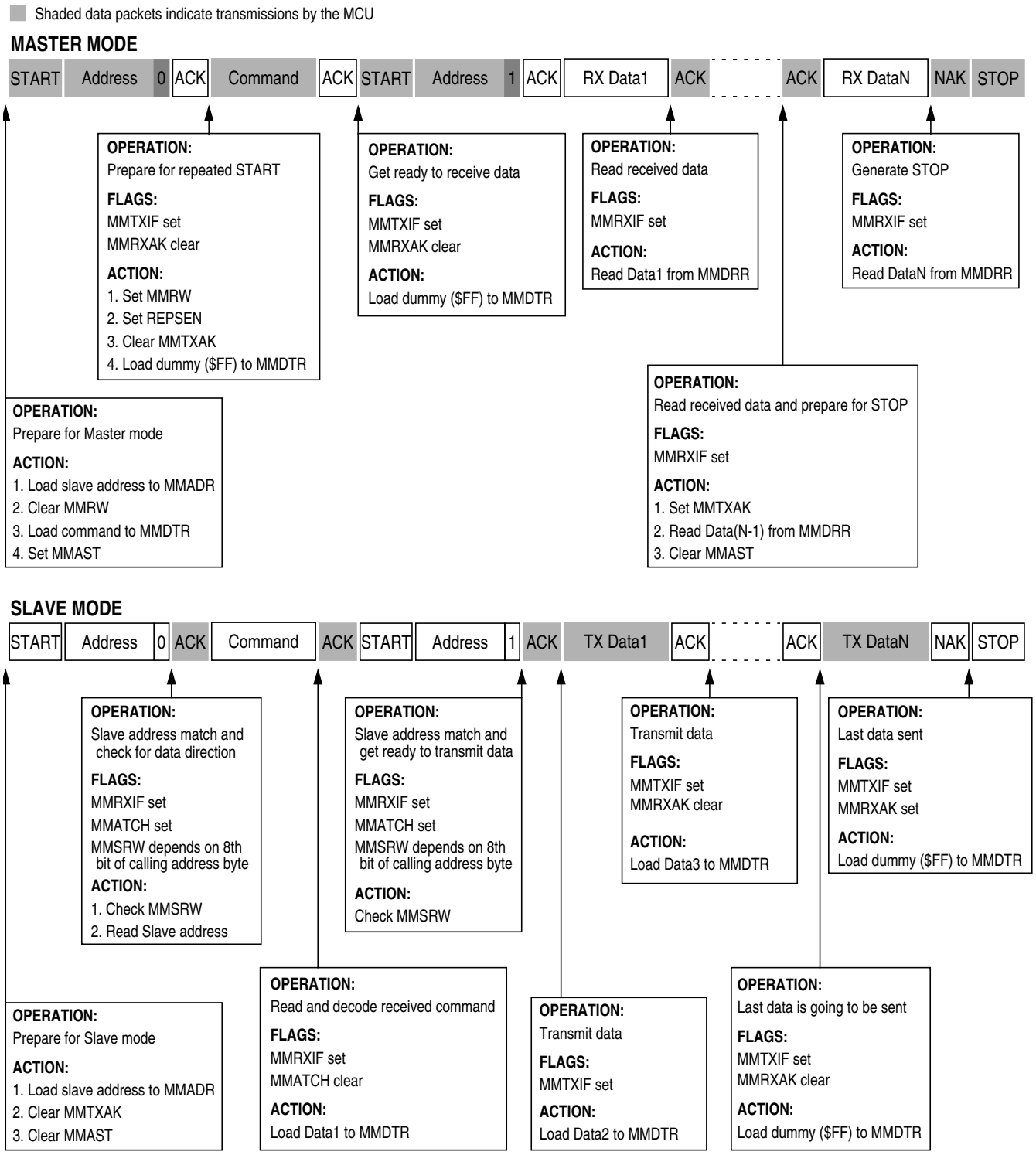


Figure 14-20. SMBus Protocol Implementation

Chapter 15
Analog-to-Digital Converter (ADC)

15.1 Introduction

This section describes the analog-to-digital converter (ADC). The ADC is a 8-channel 10-bit linear successive approximation ADC.

15.2 Features

Features of the ADC module include:

- Eight channels with multiplexed input
- High impedance buffered input
- Linear successive approximation with monotonicity
- 10-bit resolution
- Single or continuous conversion
- Auto-scan conversion on four channels
- Conversion complete flag or conversion complete interrupt
- Selectable ADC clock
- Conversion result justification
  - 8-bit truncated mode
  - Right justified mode
  - Left justified mode
  - Left justified sign mode

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0057	ADC Status and Control Register (ADSCR)	Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
		Write:								
		Reset:	0	0	0	1	1	1	1	1
$0058	ADC Clock Control Register (ADICLK)	Read:	ADIV2	ADIV1	ADIV0	ADICLK	MODE1	MODE0	0	0
		Write:								R
		Reset:	0	0	0	0	0	1	0	0
$0059	ADC Data Register High 0 (ADRH0)	Read:	ADx	ADx	ADx	ADx	ADx	ADx	ADx	ADx
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005A	ADC Data Register Low 0 (ADRL0)	Read:	ADx	ADx	ADx	ADx	ADx	ADx	ADx	ADx
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005B	ADC Data Register Low 1 (ADRL1)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

Figure 15-1. ADC I/O Register Summary

$005C	ADC Data Register Low 2 (ADRL3)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005D	ADC Data Register Low 3 (ADRL3)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005E	ADC Auto-scan Control Register (ADASCR)	Read:	0	0	0	0	0	AUTO1	AUTO0	ASCAN
		Write:								
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 15-1. ADC I/O Register Summary

15.3 Functional Description

The ADC provides eight pins for sampling external sources at pins PTA0/ADC0–PTA7/ADC7. An analog multiplexer allows the single ADC converter to select one of eight ADC channels as ADC voltage in (V_{ADIN}). V_{ADIN} is converted by the successive approximation register-based analog-to-digital converter. When the conversion is completed, ADC places the result in the ADC data register, high and low byte (ADRH0 and ADRL0), and sets a flag or generates an interrupt.

An additional three ADC data registers (ADRL1–ADRL3) are available to store the individual converted data for ADC channels ADC1–ADC3 when the auto-scan mode is enabled. Data from channel ADC0 is stored in ADRL0 in the auto-scan mode.

Figure 15-2 shows the structure of the ADC module.

15.3.1 ADC Port I/O Pins

PTA0–PTA7 are general-purpose I/O pins that are shared with the ADC channels. The channel select bits, ADCH[4:0], define which ADC channel/port pin will be used as the input signal. The ADC overrides the port I/O logic by forcing that pin as input to the ADC. The remaining ADC channels/port pins are controlled by the port I/O logic and can be used as general-purpose I/O. Writes to the port data register or data direction register will not have any affect on the port pin that is selected by the ADC. Read of a port pin which is in use by the ADC will return the pin condition if the corresponding DDR bit is at logic 0. If the DDR bit is at logic 1, the value in the port data latch is read.

15.3.2 Voltage Conversion

When the input voltage to the ADC equals V_{REFH}, the ADC converts the signal to $3FF (full scale). If the input voltage equals V_{REFL}, the ADC converts it to $000. Input voltages between V_{REFH} and V_{REFL} are a straight-line linear conversion. All other input voltages will result in $3FF if greater than V_{REFH} and $000 if less than V_{REFL}.

NOTE

Input voltage should not exceed the analog supply voltages.

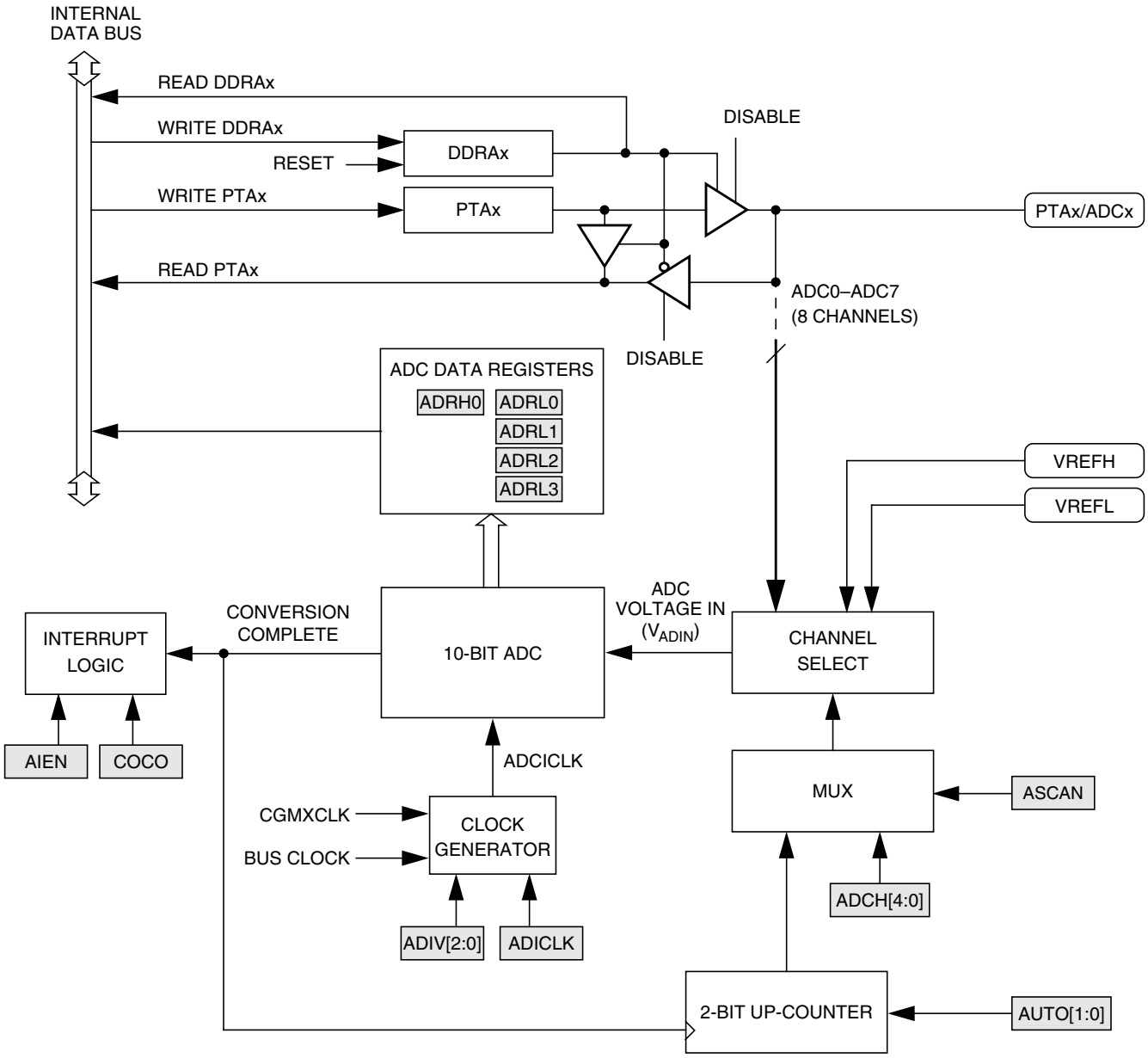


Figure 15-2. ADC Block Diagram

15.3.3 Conversion Time

Conversion starts after a write to the ADSCR. One conversion will take between 16 and 17 ADC clock cycles, therefore:

$$\text{Conversion time} = \frac{\text{16 to17 ADC cycles}}{\text{ADC frequency}}$$

Number of bus cycles = conversion time × bus frequency

The ADC conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is either the bus clock or CGMXCLK and is selectable by the ADICLK bit located in the ADC clock register. The divide ratio is selected by the ADIV[2:0] bits.

For example, if a 4MHz CGMXCLK is selected as the ADC input clock source, with a divide-by-four prescale, and the bus speed is set at 2MHz:

$$\text{Conversion time} = \frac{16 \text{ to}17 \text{ ADC cycles}}{4\text{MHz} \div 4} = 16 \text{ to } 17\ \mu\text{s}$$

$$\text{Number of bus cycles} = 16\ \mu\text{s} \times 2\text{MHz} = 32 \text{ to } 34 \text{ cycles}$$

NOTE

The ADC frequency must be between f_{ADIC} minimum and f_{ADIC} maximum to meet A/D specifications. (See 22.5 5V DC Electrical Characteristics.).

Since an ADC cycle may be comprised of several bus cycles (four in the previous example) and the start of a conversion is initiated by a bus cycle write to the ADSCR, from zero to four additional bus cycles may occur before the start of the initial ADC cycle. This results in a fractional ADC cycle and is represented as the 17th cycle.

15.3.4 Continuous Conversion

In the continuous conversion mode, the ADC continuously converts the selected channel, filling the ADC data register with new data after each conversion. Data from the previous conversion will be overwritten whether that data has been read or not. Conversions will continue until the ADCO bit is cleared. The COCO bit is set after each conversion and can be cleared by writing to the ADC status and control register or reading of the ADRL0 data register.

15.3.5 Auto-Scan Mode

In auto-scan mode, the ADC input channel is selected by the value of the 2-bit up-counter, instead of the channel select bits, ADCH[4:0]. The value of the counter also defines the data register ADRLx to be used to store the conversion result. When ASCAN bit is set, a write to ADC status and control register (ADSCR) will reset the auto-scan up-counter and ADC conversion will start on the channel 0 up to the channel number defined by the integer value of AUTO[1:0]. After a channel conversion is completed, data is stored in ADRLx and the COCO-bit will be set. The counter value will be incremented by 1 and a new conversion will start. This process will continue until the counter value reaches the value of AUTO[1:0]. When this happens, it indicates that the current channel is the last channel to be converted. Upon the completion on the last channel, the counter value will not be incremented and no further conversion will be performed. To start another auto-scan cycle, a write to ADSCR must be performed.

NOTE

The system only provides 8-bit data storage in auto-scan code, user must clear MODE[1:0] bits to select 8-bit truncation mode before entering auto-scan mode.

It is recommended that user should disable the auto-scan function before switching channel and also before entering STOP mode.

15.3.6 Result Justification

The conversion result may be formatted in four different ways.

- Left justified
- Right justified
- Left justified sign data mode
- 8-bit truncation

All four of these modes are controlled using MODE0 and MODE1 bits located in the ADC clock control register (ADICLK).

Left justification will place the eight most significant bits (MSB) in the corresponding ADC data register high (ADRH). This may be useful if the result is to be treated as an 8-bit result where the least significant two bits, located in the ADC data register low (ADRL) can be ignored. However, you must read ADRL after ADRH or else the interlocking will prevent all new conversions from being stored.

Right justification will place only the two MSBs in the corresponding ADC data register high (ADRH) and the eight LSB bits in ADC data register low (ADRL). This mode of operation typically is used when a 10-bit unsigned result is desired.

Left justified sign data mode is similar to left justified mode with one exception. The MSB of the 10-bit result, AD9 located in ADRH is complemented. This mode of operation is useful when a result, represented as a signed magnitude from mid-scale, is needed.

Finally, 8-bit truncation mode will place the eight MSBs in ADC data register low (ADRL). The two LSBs are dropped. This mode of operation is used when compatibility with 8-bit ADC designs are required. No interlocking between ADRH and ADRL is present.

15.3.7 Data Register Interlocking

Reading ADRH in any 10-bit mode latches the contents of ADRL until ADRL is read. Until ADRL is read all subsequent ADC results will be lost. This register interlocking can also be reset by a write to the ADC status and control register, or ADC clock control register. A power-on reset or reset will also clear the interlocking. Note that an external conversion request will not reset the lock.

15.3.8 Monotonicity

The conversion process is monotonic and has no missing codes.

15.4 Interrupts

When the AIEN bit is set, the ADC module is capable of generating a CPU interrupt after each ADC conversion or after an auto-scan conversion cycle. A CPU interrupt is generated if the COCO bit is at logic 0. The COCO bit is not used as a conversion complete flag when interrupts are enabled. The interrupt vector is defined in Table 2-1 . Vector Addresses.

15.5 Low-Power Modes

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

15.5.1 Wait Mode

The ADC continues normal operation in wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting the ADCH[4:0] bits to logic 1's before executing the WAIT instruction.

15.5.2 Stop Mode

The ADC module is inactive after the execution of a STOP instruction. Any pending conversion is aborted. ADC conversions resume when the MCU exits stop mode. Allow one conversion cycle to stabilize the analog circuitry before attempting a new ADC conversion after exiting stop mode.

15.6 I/O Signals

The ADC module has eight channels shared with port A I/O pins.

15.6.1 ADC Voltage In (V_{ADIN})

V_{ADIN} is the input voltage signal from one of the eight ADC channels to the ADC module.

15.6.2 ADC Analog Power Pin (V_{DDA})

The ADC analog portion uses V_{DDA} as its power pin. Connect the V_{DDA} pin to the same voltage potential as V_{DD}. External filtering may be necessary to ensure clean V_{DDA} for good results.

NOTE

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

15.6.3 ADC Analog Ground Pin (V_{SSA})

The ADC analog portion uses V_{SSA} as its ground pin. Connect the V_{SSA} pin to the same voltage potential as V_{SS}.

15.6.4 ADC Voltage Reference High Pin (V_{REFH})

V_{REFH} is the power supply for setting the reference voltage V_{REFH}. Connect the V_{REFH} pin to the same voltage potential as V_{DDA}. There will be a finite current associated with V_{REFH} (see Chapter 22 Electrical Specifications).

NOTE

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

15.6.5 ADC Voltage Reference Low Pin (V_{REFL})

V_{REFL} is the lower reference supply for the ADC. Connect the V_{REFL} pin to the same voltage potential as V_{SSA}. There will be a finite current associated with V_{REFL} (see Chapter 22 Electrical Specifications).

15.7 I/O Registers

These I/O registers control and monitor ADC operation:

- ADC status and control register (ADSCR) — $0057
- ADC clock control register (ADICLK) — $0058
- ADC data register high:low 0 (ADRH0:ADRL0) — $0059:$005A
- ADC data register low 1–3 (ADRL1–ADRL3) — $005B–$005D
- ADC auto-scan control register (ADASCR) — $005E

15.7.1 ADC Status and Control Register

Function of the ADC status and control register is described here.

Address:	$0057							
Read:	COCO	AIEN	ADCO	ADCH4	ADCH3	ADCH2	ADCH1	ADCH0
Write:								
Reset:	0	0	0	1	1	1	1	1

Figure 15-3. ADC Status and Control Register (ADSCR)

COCO — Conversions Complete Bit

In non-interrupt mode (AIEN = 0), COCO is a read-only bit that is set at the end of each conversion. COCO will stay set until cleared by a read of the ADC data register. Reset clears this bit.

In interrupt mode (AIEN = 1), COCO is a read-only bit that is not set at the end of a conversion. It always reads as a 0.

1 = Conversion completed (AIEN = 0)
0 = Conversion not completed (AIEN = 0) or CPU interrupt enabled (AIEN = 1)

NOTE

The write function of the COCO bit is reserved. When writing to the ADSCR register, always have a 0 in the COCO bit position.

AIEN — ADC Interrupt Enable Bit

When this bit is set, an interrupt is generated at the end of an ADC conversion. The interrupt signal is cleared when the data register, ADR0, is read or the ADSCR is written. Reset clears the AIEN bit.

1 = ADC interrupt enabled
0 = ADC interrupt disabled

ADCO — ADC Continuous Conversion Bit

When set, the ADC will convert samples continuously and update the ADC data register at the end of each conversion. Only one conversion is allowed when this bit is cleared. Reset clears the ADCO bit.

1 = Continuous ADC conversion
0 = One ADC conversion

This bit should not be set when auto-scan mode is enabled; i.e. when ASCAN=1.

ADCH[4:0] — ADC Channel Select Bits

ADCH[4:0] form a 5-bit field which is used to select one of the ADC channels when not in auto-scan mode. The five channel select bits are detailed in Table 15-1.

NOTE

Care should be taken when using a port pin as both an analog and a digital input simultaneously to prevent switching noise from corrupting the analog

signal. Recovery from the disabled state requires one conversion cycle to stabilize.

Table 15-1. MUX Channel Select

ADCH4	ADCH3	ADCH2	ADCH1	ADCH0	ADC Channel	Input Select
0	0	0	0	0	ADC0	PTA0
0	0	0	0	1	ADC1	PTA1
0	0	0	1	0	ADC2	PTA2
0	0	0	1	1	ADC3	PTA3
0	0	1	0	0	ADC4	PTA4
0	0	1	0	1	ADC5	PTA5
0	0	1	1	0	ADC6	PTA6
0	0	1	1	1	ADC7	PTA7
0 ↓ 1	1 ↓ 1	0 ↓ 1	0 ↓ 0	0 ↓ 0	ADC8 ↓ ADC28	Reserved
1	1	1	0	1	ADC29	V_{REFH} (see Note 2)
1	1	1	1	0	ADC30	V_{REFL} (see Note 2)
1	1	1	1	1	ADC powered-off	—

NOTES:
1. If any unused channels are selected, the resulting ADC conversion will be unknown.
2. The voltage levels supplied from internal reference nodes as specified in the table are used to verify the operation of the ADC converter both in production test and for user applications.

15.7.2 ADC Clock Control Register

The ADC clock control register (ADICLK) selects the clock frequency for the ADC.

Address:	$0058							
Read:	ADIV2	ADIV1	ADIV0	ADICLK	MODE1	MODE0	0	0
Write:								R
Reset:	0	0	0	0	0	1	0	0

= Unimplemented R = Reserved

Figure 15-4. ADC Clock Control Register (ADICLK)

ADIV[2:0] — ADC Clock Prescaler Bits

ADIV2, ADIV1, and ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock.

Table 15-2 shows the available clock configurations. The ADC clock should be set to between 500 kHz and 1 MHz.

Table 15-2. ADC Clock Divide Ratio

ADIV2	ADIV1	ADIV0	ADC Clock Rate
0	0	0	ADC input clock ÷ 1
0	0	1	ADC input clock ÷ 2
0	1	0	ADC input clock ÷ 4
0	1	1	ADC input clock ÷ 8
1	X	X	ADC input clock ÷ 16

X = don't care

ADICLK — ADC Input Clock Select Bit

ADICLK selects either bus clock or CGMXCLK as the input clock source to generate the internal ADC clock. Reset selects CGMXCLK as the ADC clock source.

If the external clock (CGMXCLK) is equal to or greater than 1 MHz, CGMXCLK can be used as the clock source for the ADC. If CGMXCLK is less than 1 MHz, use the PLL-generated bus clock as the clock source. As long as the internal ADC clock is at f_{ADIC}, correct operation can be guaranteed.

1 = Internal bus clock
0 = External clock, CGMXCLK

$$f_{ADIC} = \frac{\text{CGMXCLK or bus frequency}}{\text{ADIV[2:0]}}$$

MODE1 and MODE0 — Modes of Result Justification

MODE1 and MODE0 selects between four modes of operation. The manner in which the ADC conversion results will be placed in the ADC data registers is controlled by these modes of operation. Reset returns right-justified mode.

Table 15-3. ADC Mode Select

MODE1	MODE0	Justification Mode
0	0	8-bit truncated mode
0	1	Right justified mode
1	0	Left justified mode
1	1	Left justified sign data mode

15.7.3 ADC Data Register 0 (ADRH0 and ADRL0)

The ADC data register 0 consist of a pair of 8-bit registers: high byte (ADRH0), and low byte (ADRL0). This pair form a 16-bit register to store the 10-bit ADC result for the selected ADC result justification mode.

In 8-bit truncated mode, the ADRL0 holds the eight most significant bits (MSBs) of the 10-bit result. The ADRL0 is updated each time an ADC conversion completes. In 8-bit truncated mode, ADRL0 contains no interlocking with ADRH0. (See Figure 15-5 . ADRH0 and ADRL0 in 8-Bit Truncated Mode.)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0059	ADC Data Register High 0 (ADRH0)	Read:	0	0	0	0	0	0	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005A	ADC Data Register Low 0 (ADRL0)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

Figure 15-5. ADRH0 and ADRL0 in 8-Bit Truncated Mode

In right justified mode the ADRH0 holds the two MSBs, and the ADRL0 holds the eight least significant bits (LSBs), of the 10-bit result. ADRH0 and ADRL0 are updated each time a single channel ADC conversion completes. Reading ADRH0 latches the contents of ADRL0. Until ADRL0 is read all subsequent ADC results will be lost. (See Figure 15-6 . ADRH0 and ADRL0 in Right Justified Mode.)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0059	ADC Data Register High 0 (ADRH0)	Read:	0	0	0	0	0	0	AD9	AD8
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005A	ADC Data Register Low 0 (ADRL0)	Read:	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

Figure 15-6. ADRH0 and ADRL0 in Right Justified Mode

In left justified mode the ADRH0 holds the eight most significant bits (MSBs), and the ADRL0 holds the two least significant bits (LSBs), of the 10-bit result. The ADRH0 and ADRL0 are updated each time a single channel ADC conversion completes. Reading ADRH0 latches the contents of ADRL0. Until ADRL0 is read all subsequent ADC results will be lost. (See Figure 15-7 . ADRH0 and ADRL0 in Left Justified Mode.)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0059	ADC Data Register High 0 (ADRH0)	Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005A	ADC Data Register Low 0 (ADRL0)	Read:	AD1	AD0	0	0	0	0	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

Figure 15-7. ADRH0 and ADRL0 in Left Justified Mode

In left justified sign mode the ADRH0 holds the eight MSBs with the MSB complemented, and the ADRL0 holds the two least significant bits (LSBs), of the 10-bit result. The ADRH0 and ADRL0 are updated each time a single channel ADC conversion completes. Reading ADRH0 latches the contents of ADRL0. Until ADRL0 is read all subsequent ADC results will be lost. (See Figure 15-8 ADRH0 and ADRL0 in Left Justified Sign Data Mode.)

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0059	ADC Data Register High 0 (ADRH0)	Read:	$\overline{AD9}$	AD8	AD7	AD6	AD5	AD4	AD3	AD2
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0
$005A	ADC Data Register Low 0 (ADRL0)	Read:	AD1	AD0	0	0	0	0	0	0
		Write:	R	R	R	R	R	R	R	R
		Reset:	0	0	0	0	0	0	0	0

Figure 15-8 ADRH0 and ADRL0 in Left Justified Sign Data Mode

15.7.4 ADC Auto-Scan Mode Data Registers (ADRL1–ADRL3)

The ADC data registers 1 to 3 (ADRL1–ADRL3), are 8-bit registers for conversion results in 8-bit truncated mode, for channels ADC1 to ADC3, when the ADC is operating in auto-scan mode (MODE[1:0] = 00).

Address: ADRL1, $005B; ADRL2, $005C; and ADRL3, $005D

Read:	AD9	AD8	AD7	AD6	AD5	AD4	AD3	AD2
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 15-9. ADC Data Register Low 1 to 3 (ADRL1–ADRL3)

15.7.5 ADC Auto-Scan Control Register (ADASCR)

The ADC auto-scan control register (ADASCR) enables and controls the ADC auto-scan function.

Address: $005E

Read:	0	0	0	0	0	AUTO1	AUTO0	ASCAN
Write:								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented R = Reserved

Figure 15-10. ADC Scan Control Register (ADASCR)

AUTO[1:0] — Auto-Scan Mode Channel Select Bits

AUTO1 and AUTO0 form a 2-bit field which is used to define the number of auto-scan channels used when in auto-scan mode. Reset clears these bits.

Table 15-4. Auto-scan Mode Channel Select

AUTO1	AUTO0	Auto-Scan Channels
0	0	ADC0 only
0	1	ADC0 to ADC1
1	0	ADC0 to ADC2
1	1	ADC0 to ADC3

ASCAN — Auto-scan Mode Enable Bit

This bit enable/disable the auto-scan mode. Reset clears this bit.

1 = Auto-scan mode is enabled

0 = Auto-scan mode is disabled

Auto-scan mode should not be enabled when ADC continuous conversion is enabled; i.e. when ADCO=1.

Chapter 16
Input/Output (I/O) Ports

16.1 Introduction

Thirty-two (32) bidirectional input-output (I/O) pins form four parallel ports. All I/O pins are programmable as inputs or outputs.

Input pins and I/O port pins that are not used in the application must be terminated. This prevents excess current caused by floating inputs, and enhances immunity during noise or transient events. Termination methods include:

1. Configuring unused pins as outputs and driving high or low;
2. Configuring unused pins as inputs and enabling internal pull-ups;
3. Configuring unused pins as inputs and using external pull-up or pull-down resistors.

Never connect unused pins directly to V_{DD} or V_{SS}.

Since some general-purpose I/O pins are not available on all packages, these pins must be terminated as well. Either method 1 or 2 above are appropriate.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0000	Port A Data Register (PTA)	Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
		Reset:	Unaffected by reset							
$0001	Port B Data Register (PTB)	Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
		Reset:	Unaffected by reset							
$0002	Port C Data Register (PTC)	Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
		Reset:	Unaffected by reset							
$0003	Port D Data Register (PTD)	Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
		Reset:	Unaffected by reset							
$0004	Data Direction Register A (DDRA)	Read: Write:	DDRA7	DDRA6	DDRA5	DDRA4	DDRA3	DDRA2	DDRA1	DDRA0
		Reset:	0	0	0	0	0	0	0	0
$0005	Data Direction Register B (DDRB)	Read: Write:	DDRB7	DDRB6	DDRB5	DDRB4	DDRB3	DDRB2	DDRB1	DDRB0
		Reset:	0	0	0	0	0	0	0	0
$0006	Data Direction Register C (DDRC)	Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
		Reset:	0	0	0	0	0	0	0	0

Figure 16-1. I/O Port Register Summary

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$0007	Data Direction Register D (DDRD)	Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
		Reset:	0	0	0	0	0	0	0	0
$000C	Port-A LED Control Register (LEDA)	Read: Write:	LEDA7	LEDA6	LEDA5	LEDA4	LEDA3	LEDA2	LEDA1	LEDA0
		Reset:	0	0	0	0	0	0	0	0

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

Table 16-1. Port Control Register Bits Summary

Port	Bit	DDR	Module Control			Pin
			Module	Register	Control Bit	
A	0	DDRA0	ADC	ADSCR ($0057)	ADCH[4:0]	PTA0/ADC0
	1	DDRA1				PTA1/ADC1
	2	DDRA2				PTA2/ADC2
	3	DDRA3				PTA3/ADC3
	4	DDRA4				PTA4/ADC4
	5	DDRA5				PTA5/ADC5
	6	DDRA6				PTA6/ADC6
	7	DDRA7				PTA7/ADC7
B	0	DDRB0	MBUS	MMCR1 ($0049)	MMEN	PTB0/SDA(1)
	1	DDRB1				PTB1/SCL(1)
	2	DDRB2	SCI	SCC1 ($0013)	ENSCI	PTB2/TxD(1)
	3	DDRB3				PTB3/RxD(1)
	4	DDRB4	TIM1	T1SC0 ($0025)	ELS0B:ELS0A	PTB4/T1CH0(2)
	5	DDRB5		T1SC1 ($0028)	ELS1B:ELS1A	PTB5/T1CH1(2)
	6	DDRB6	TIM2	T2SC0 ($0030)	ELS0B:ELS0A	PTB6/T2CH0(2)
	7	DDRB7		T2SC1 ($0033)	ELS1B:ELS1A	PTB7/T2CH1(2)
C	0	DDRC0	IRQ2	INTSCR2 ($001C)	IMASK2	PTC0/$\overline{\text{IRQ2}}$(2)
	1	DDRC1	—	—	—	PTC1
	2	DDRC2	SPI	SPCR ($0010)	SPE	PTC2/MISO
	3	DDRC3				PTC3/MOSI
	4	DDRC4				PTC4/$\overline{\text{SS}}$
	5	DDRC5				PTC5/SPSCK
	6	DDRC6	IRSCI	IRSCC1 ($0040)	ENSCI	PTC6/SCTxD(1)
	7	DDRC7				PTC7/SCRxD(1)
D	0	DDRD0	KBI	KBIER ($001B)	KBIE0	PTD0/KBI0(2)
	1	DDRD1			KBIE1	PTD1/KBI1(2)
	2	DDRD2			KBIE2	PTD2/KBI2(2)
	3	DDRD3			KBIE3	PTD3/KBI3(2)
	4	DDRD4			KBIE4	PTD4/KBI4(2)
	5	DDRD5			KBIE5	PTD5/KBI5(2)
	6	DDRD6			KBIE6	PTD6/KBI6(2)
	7	DDRD7			KBIE7	PTD7/KBI7(2)

1. Pin is open-drain when configured as output. Pullup resistor must be connected when configured as output.
2. Pin has schmitt trigger when configured as input.

16.2 Port A

Port A is an 8-bit special-function port that shares all of its pins with the analog-to-digital converter (ADC) module. Port A pins also have LED direct drive capability.

16.2.1 Port A Data Register (PTA)

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

Address: $0000

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTA7	PTA6	PTA5	PTA4	PTA3	PTA2	PTA1	PTA0
Reset:	Unaffected by reset							
Alternative Function:	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
Additional Function:	LED drive	LED drive	LED drive	LED drive	LED drive	LED drive	LED drive	LED drive

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

PTA[7:0] — Port A Data Bits

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

ADC7–ADC0 — ADC Channels 7 to 0

ADC7–ADC0 are pins used for the input channels to the analog-to-digital converter module. The channel select bits, ADCH[4:0], in the ADC status and control register define which port pin will be used as an ADC input and overrides any control from the port I/O logic.

NOTE

Care must be taken when reading port A while applying analog voltages to ADC7–ADC0 pins. If the appropriate ADC channel is not enabled, excessive current drain may occur if analog voltages are applied to the PTAx/ADCx pin, while PTA is read as a digital input. Those ports not selected as analog input channels are considered digital I/O ports.

LED drive — Direct LED drive pins

PTA7–PTA0 pins can be configured for direct LED drive. See 16.2.3 Port-A LED Control Register (LEDA).

16.2.2 Data Direction Register (DDRA)

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

Address: $0004

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

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

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

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

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

NOTE

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

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

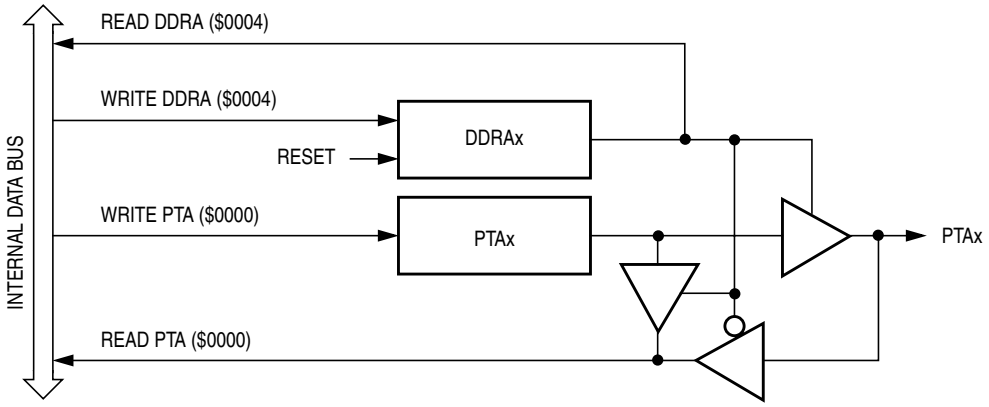


Figure 16-4. Port A I/O Circuit

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

Table 16-2 summarizes the operation of the port A pins.

Table 16-2. 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.

16.2.3 Port-A LED Control Register (LEDA)

The port-A LED control register (LEDA) controls the direct LED drive capability on PTA7–PTA0 pins. Each bit is individually configurable and requires that the data direction register, DDRA, bit be configured as an output.

Address: $000C

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	LEDA7	LEDA6	LEDA5	LEDA4	LEDA3	LEDA2	LEDA1	LEDA0
Reset:	0	0	0	0	0	0	0	0

Figure 16-5. Port A LED Control Register (LEDA)

LEDA[7:0] — Port A 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 A pin is configured for direct LED drive, with 15mA current sinking capability

0 = Corresponding port A pin is configured for standard drive

16.3 Port B

Port B is an 8-bit special-function port that shares two of its pins with the multi-master IIC (MMIIC) module, two of its pins with SCI module, and four of its pins with two timer interface (TIM1 and TIM2) modules.

NOTE

PTB3–PTB0 are open-drain pins when configured as outputs regardless whether the pins are used as general purpose I/O pins, MMIIC pins, or SCI pins. Therefore, when configured as general purpose output pins, MMIIC pins, or SCI pins (the TxD pin), pullup resistors must be connected to these pins.

16.3.1 Port B Data Register (PTB)

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

Address: $0001

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTB7	PTB6	PTB5	PTB4	PTB3	PTB2	PTB1	PTB0
Reset:	Unaffected by reset							
Alternative Function:	T2CH1	T2CH0	T1CH1	T1CH0	RxD	TxD	SCL	SDA

Figure 16-6. Port B Data Register (PTB)

PTB[7:0] — Port B Data Bits

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

SDA and SCL — Multi-Master IIC Data and Clock

The SDA and SCL pins are multi-master IIC data and clock pins. Setting the MMEN bit in the MMIIC control register 1 (MMCR1) configures the PTB0/SDA and PTB1/SCL pins for MMIIC function and overrides any control from the port I/O logic.

TxD and RxD — SCI Transmit and Receive Data

The TxD and RxD pins are SCI transmit and receive data pins. Setting the ENSCI bit in the SCI control register 1 (SCC1) configures the PTB2/TxD and PTB3/RxD pins for SCI function and overrides any control from the port I/O logic.

T1CH0 and T1CH1 — Timer 1 Channel I/O

The T1CH0 and T1CH1 pins are the TIM1 input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTB4/T1CH0–PTB5/T1CH1 pins are timer channel I/O pins or general-purpose I/O pins.

T2CH0 and T2CH1 — Timer 2 Channel I/O

The T2CH0 and T2CH1 pins are the TIM2 input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTB6/T2CH0–PTB7/T2CH1 pins are timer channel I/O pins or general-purpose I/O pins.

16.3.2 Data Direction Register B (DDRB)

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

Address: $0005

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

Figure 16-7. Data Direction Register B (DDRB)

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

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

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

NOTE

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

Figure 16-8 shows the port B I/O logic.

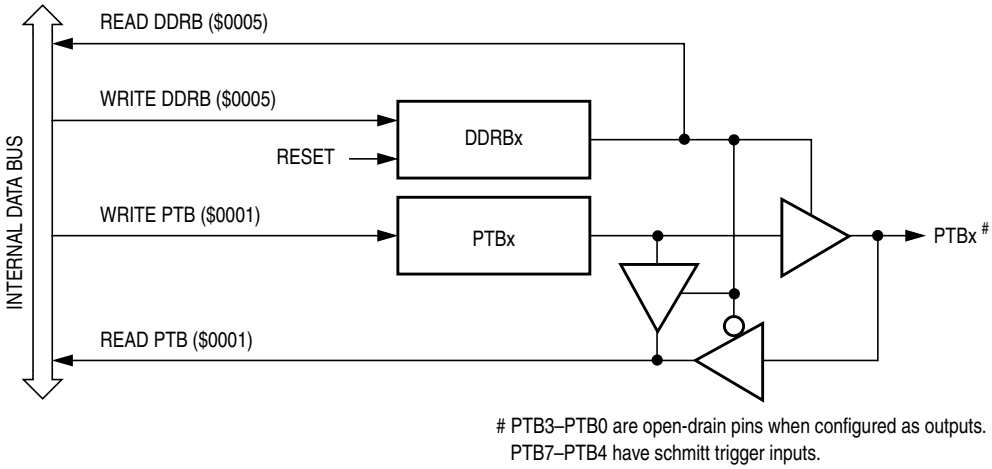


Figure 16-8. Port B I/O Circuit

When DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 16-3 summarizes the operation of the port B pins.

Table 16-3. Port B Pin Functions

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

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

16.4 Port C

Port C is an 8-bit special-function port that shares one of its pins with the $\overline{\text{IRQ2}}$, four of its pins with the SPI module, and two of its pins with the IRSCI module.

16.4.1 Port C Data Register (PTC)

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

Address: $0002

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTC7	PTC6	PTC5	PTC4	PTC3	PTC2	PTC1	PTC0
Reset:	Unaffected by reset							
Alternative Function:	SCRxD	SCTxD	SPSCK	SS	MOSI	MISO		IRQ2

Figure 16-9. Port C Data Register (PTC)

PTC[7:0] — Port C Data Bits

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

$\overline{\text{IRQ2}}$ — IRQ2 input pin

The PTC0/$\overline{\text{IRQ2}}$ pin is always available as input pin to the IRQ2 module. Care must be taken to available unwanted interrupts when this pin is used as general purpose I/O. PTC0/$\overline{\text{IRQ2}}$ pin has an internal pullup, and can be disabled by setting the PUC0ENB bit in the IRQ2 status and control register (INTSCR2).

MISO, MOSI, $\overline{\text{SS}}$, and SPSCK — SPI Data I/O, Select, and Clock Pins

These pins are the SPI data in/out, select, and clock pins. Setting the SPE bit in the SPI control register (SPCR) configures PTC2/MISO, PTC3/MOSI, PTC4/$\overline{\text{SS}}$, and PTC5/SPSCK pins for SPI function and overrides any control from the port I/O logic.

SCTxD and SCRxD — IrSCI Transmit and Receive Data

The SCTxD and SCRxD pins are IRSCI transmit and receive data pins. Setting the ENSCI bit in the IRSCI control register 1 (IRSCC1) configures the PTC6/SCTxD and PTC7/SCRxD pins for IRSCI function and overrides any control from the port I/O logic.

16.4.2 Data Direction Register C (DDRC)

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

Address: $0006

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRC7	DDRC6	DDRC5	DDRC4	DDRC3	DDRC2	DDRC1	DDRC0
Reset:	0	0	0	0	0	0	0	0

Figure 16-10. Data Direction Register C (DDRC)

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

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

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

NOTE

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

Figure 16-11 shows the port C I/O logic.

NOTE

For those devices packaged in a 42-pin shrink dual in-line package, PTC0 and PTC1 are not connected. DDRC0 and DDRC1 should be set to a 1 to configure PTC0 and PTC1 as outputs.

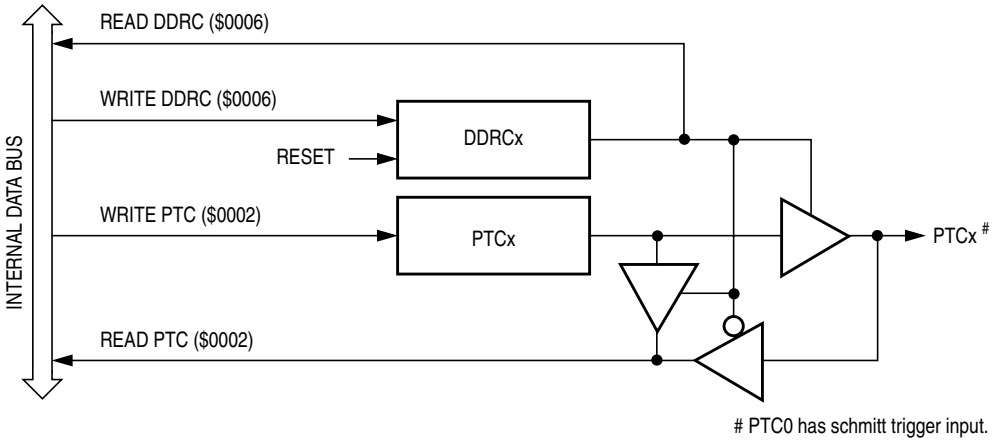


Figure 16-11. Port C I/O Circuit

When DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When DDRCx is a logic 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 16-4 summarizes the operation of the port C pins.

Table 16-4. Port C Pin Functions

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

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

16.5 Port D

Port D is an 8-bit special function port that shares all of its pins with the keyboard interrupt module.

16.5.1 Port D Data Register (PTD)

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

Address: $0003

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	PTD7	PTD6	PTD5	PTD4	PTD3	PTD2	PTD1	PTD0
Reset:	Unaffected by reset							
Alternative Function:	KBI7	KBI6	KBI5	KBI4	KBI3	KBI2	KBI1	KBI0

Figure 16-12. Port D Data Register (PTD)

PTD[7:0] — Port D Data Bits

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

KBI7–KBI0 — Keyboard Interrupt Inputs

The keyboard interrupt enable bits, KBIE[7:0], in the keyboard interrupt enable register (KBIER), enable the port D pins as external interrupt pins. See Chapter 18 Keyboard Interrupt Module (KBI).

16.5.2 Data Direction Register D (DDRD)

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

Address: $0007

	Bit 7	6	5	4	3	2	1	Bit 0
Read: Write:	DDRD7	DDRD6	DDRD5	DDRD4	DDRD3	DDRD2	DDRD1	DDRD0
Reset:	0	0	0	0	0	0	0	0

Figure 16-13. Data Direction Register D (DDRD)

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

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

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

NOTE

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

Figure 16-14 shows the port D I/O logic.

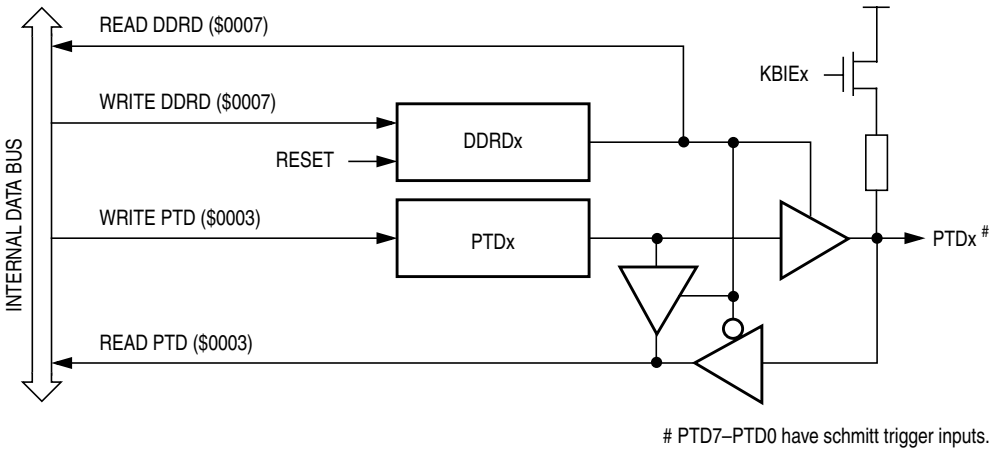


Figure 16-14. Port D I/O Circuit

When bit DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit.

Table 16-5 summarizes the operation of the port D pins.

Table 16-5. Port D Pin Functions

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

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

Chapter 17
External Interrupt (IRQ)

17.1 Introduction

The external interrupt (IRQ) module provides two maskable interrupt inputs: $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$.

17.2 Features

Features of the IRQ module include:

- A dedicated external interrupt pin, $\overline{\text{IRQ1}}$
- An external interrupt pin shared with a port pin, PTC0/$\overline{\text{IRQ2}}$
- Separate IRQ interrupt control bits for $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$
- Programmable edge-only or edge and level interrupt sensitivity
- Automatic interrupt acknowledge
- Internal pullup resistor, with disable option on $\overline{\text{IRQ2}}$

NOTE

References to either IRQ1 or IRQ2 may be made in the following text by omitting the IRQ number. For example, IRQF may refer generically to IRQ1F and IRQ2F, and IMASK may refer to IMASK1 and IMASK2.

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001C	IRQ2 Status and Control Register (INTSCR2)	Read:	0	PUC0ENB	0	0	IRQ2F	0	IMASK2	MODE2
		Write:						ACK2		
		Reset:	0	0	0	0	0	0	0	0
$001E	IRQ1 Status and Control Register (INTSCR1)	Read:	0	0	0	0	IRQ1F	0	IMASK1	MODE1
		Write:						ACK1		
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-1. External Interrupt I/O Register Summary

17.3 Functional Description

A logic 0 applied to the external interrupt pin can latch a CPU interrupt request. Figure 17-2 and Figure 17-3 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 latch that caused the vector fetch.

- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (INTSCR). 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 falling-edge and low-level-triggered. The MODE bit in the INTSCR controls the triggering sensitivity of the $\overline{IRQ}$ pin.

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

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

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

The vector fetch or software clear 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 MODE1 control bit, thereby clearing the interrupt even if the pin stays low.

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

NOTE

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

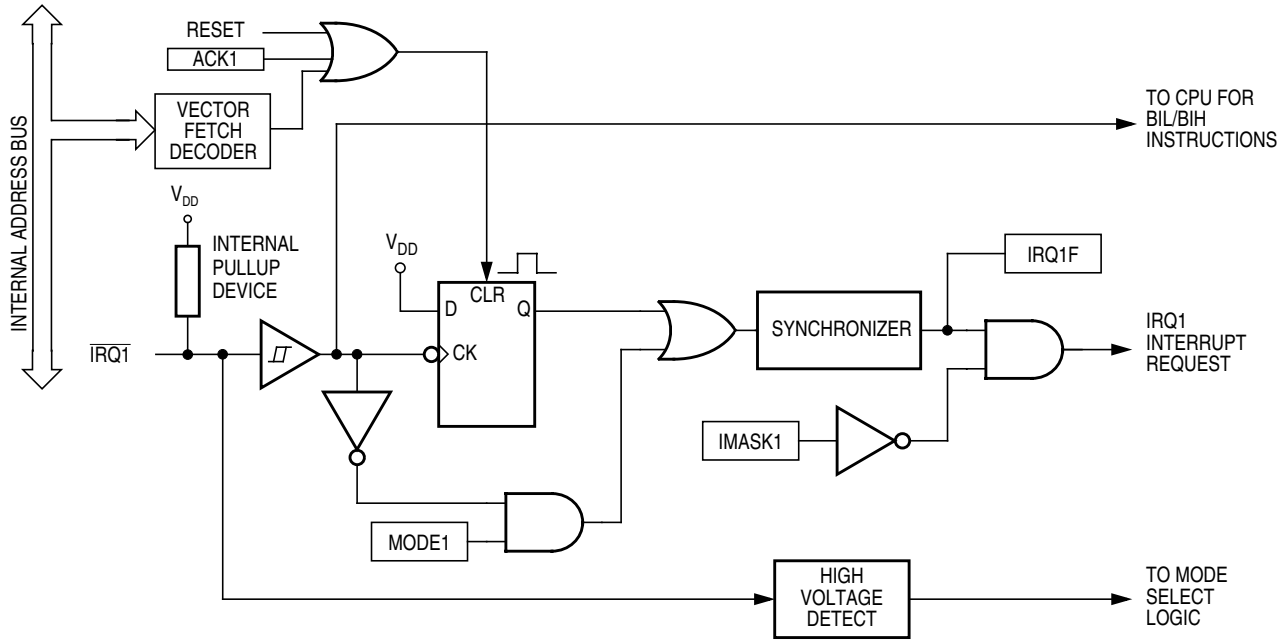


Figure 17-2. IRQ1 Block Diagram

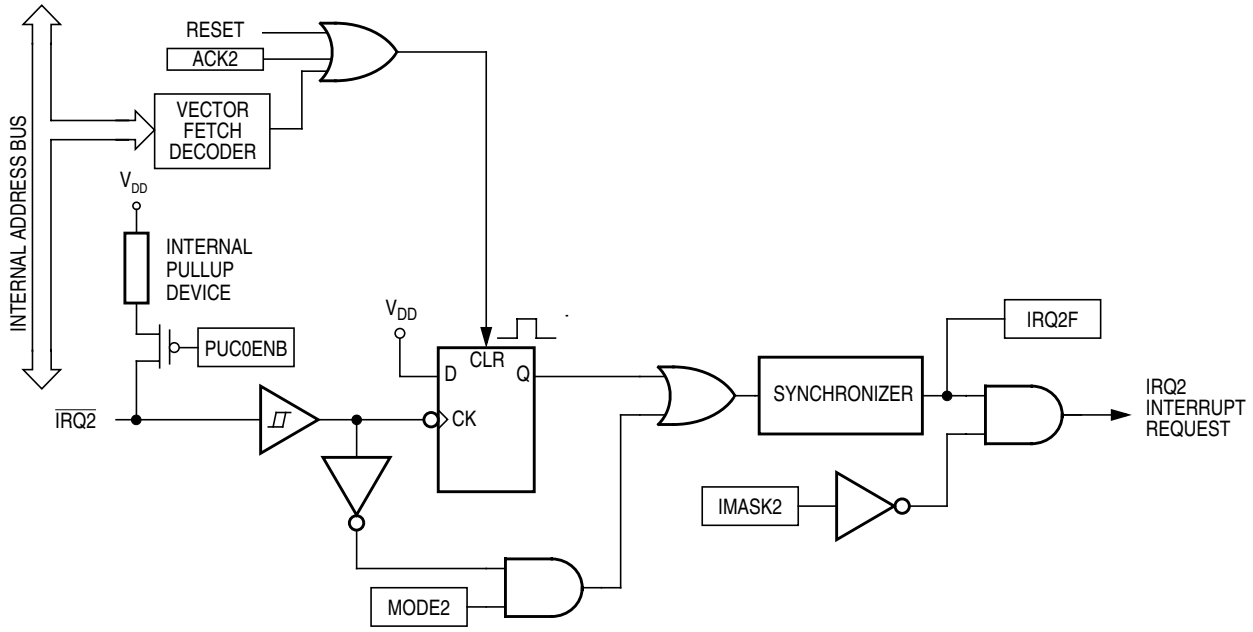


Figure 17-3. IRQ2 Block Diagram

17.4 $\overline{\text{IRQ1}}$ and $\overline{\text{IRQ2}}$ Pins

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 (INTSCR). The ACK bit is useful in applications that poll the $\overline{\text{IRQ}}$ pin and require software to clear the IRQ latch. Writing to the ACK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the $\overline{\text{IRQ}}$ pin. A falling edge that occurs after writing to the ACK bit another interrupt request. If the IRQ mask bit, IMASK, is clear, the CPU loads the program counter with the vector address at location defined in Table 2-1 . Vector Addresses.
- Return of the $\overline{\text{IRQ}}$ pin to logic 1 — As long as the $\overline{\text{IRQ}}$ pin is at logic 0, IRQ remains active.

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

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

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

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

NOTE

The BIH and BIL instructions do not read the logic level on the $\overline{\text{IRQ2}}$ pin.

NOTE

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

The $\overline{\text{IRQ1}}$ pin has a permanent internal pullup device connected, while the $\overline{\text{IRQ2}}$ pin has an optional pullup device that can be enabled or disabled by the PUC0ENB bit in the INTSCR2 register.

17.5 IRQ Module During Break Interrupts

The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latch during the break state. (See Chapter 21 Break Module (BRK).)

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

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

17.6 IRQ Registers

Each IRQ is controlled and monitored by an status and control register.

- IRQ1 Status and Control Register — $001E
- IRQ2 Status and Control Register — $001C

17.6.1 IRQ1 Status and Control Register

The IRQ1 status and control register (INTSCR1) controls and monitors operation of IRQ1. The INTSCR1 has the following functions:

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

Address: $001E

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	0	0	0	IRQ1F	0	IMASK1	MODE1
Write:						ACK1		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-4. IRQ1 Status and Control Register (INTSCR1)

IRQ1F — IRQ1 Flag Bit

This read-only status bit is high when the IRQ1 interrupt is pending.

1 = $\overline{\text{IRQ1}}$ interrupt pending
0 = $\overline{\text{IRQ1}}$ interrupt not pending

ACK1 — IRQ1 Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the IRQ1 latch. ACK1 always reads as logic 0. Reset clears ACK1.

IMASK1 — IRQ1 Interrupt Mask Bit

Writing a logic 1 to this read/write bit disables IRQ1 interrupt requests. Reset clears IMASK1.

1 = $\overline{\text{IRQ1}}$ interrupt requests disabled
0 = $\overline{\text{IRQ1}}$ interrupt requests enabled

MODE1 — IRQ1 Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ1}}$ pin. Reset clears MODE1.

1 = $\overline{\text{IRQ1}}$ interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ1}}$ interrupt requests on falling edges only

17.6.2 IRQ2 Status and Control Register

The IRQ2 status and control register (INTSCR2) controls and monitors operation of IRQ2. The INTSCR2 has the following functions:

- Enables/disables the internal pullup device on $\overline{\text{IRQ2}}$ pin
- Shows the state of the IRQ2 flag
- Clears the IRQ2 latch
- Masks IRQ2 interrupt request
- Controls triggering sensitivity of the $\overline{\text{IRQ2}}$ interrupt pin

Address: $001C

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	0	PUC0ENB	0	0	IRQ2F	0	IMASK2	MODE2
Write:						ACK2		
Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 17-5. IRQ2 Status and Control Register (INTSCR2)

PUC0ENB — $\overline{\text{IRQ2}}$ Pin Pullup Enable Bit.

Setting this bit to logic 1 disables the pullup on PTC0/$\overline{\text{IRQ2}}$ pin.
Reset clears this bit.

1 = $\overline{\text{IRQ2}}$ pin internal pullup is disabled
0 = $\overline{\text{IRQ2}}$ pin internal pullup is enabled

IRQ2F — IRQ2 Flag Bit

This read-only status bit is high when the IRQ2 interrupt is pending.

1 = $\overline{\text{IRQ2}}$ interrupt pending
0 = $\overline{\text{IRQ2}}$ interrupt not pending

ACK2 — IRQ2 Interrupt Request Acknowledge Bit

Writing a logic 1 to this write-only bit clears the IRQ2 latch. ACK2 always reads as logic 0. Reset clears ACK2.

IMASK2 — IRQ2 Interrupt Mask Bit

Writing a logic 1 to this read/write bit disables IRQ2 interrupt requests. Reset clears IMASK2.

1 = IRQ2 interrupt requests disabled
0 = IRQ2 interrupt requests enabled

MODE2 — IRQ2 Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ2}}$ pin. Reset clears MODE2.

1 = $\overline{\text{IRQ2}}$ interrupt requests on falling edges and low levels
0 = $\overline{\text{IRQ2}}$ interrupt requests on falling edges only

Chapter 18
Keyboard Interrupt Module (KBI)

18.1 Introduction

The keyboard interrupt module (KBI) provides eight independently maskable external interrupts which are accessible via PTD0–PTD7. When a port pin is enabled for keyboard interrupt function, an internal 30kΩ pullup device is also enabled on the pin.

18.2 Features

Features of the keyboard interrupt module include the following:

- Eight keyboard interrupt pins with pullup devices
- Separate keyboard interrupt enable bits and one keyboard interrupt mask
- Programmable edge-only or edge- and level- interrupt sensitivity
- Exit from low-lower modes

Addr.	Register Name		Bit 7	6	5	4	3	2	1	Bit 0
$001A	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
$001B	Keyboard Interrupt Enable Register (KBIER)	Read: Write:	KBIE7	KBIE6	KBIE5	KBIE4	KBIE3	KBIE2	KBIE1	KBIE0
		Reset:	0	0	0	0	0	0	0	0

= Unimplemented

Figure 18-1. KBI I/O Register Summary

18.3 I/O Pins

The eight keyboard interrupt pins are shared with standard port I/O pins. The full name of the KBI pins are listed in Table 18-1. The generic pin name appear in the text that follows.

Table 18-1. Pin Name Conventions

KBI Generic Pin Name	Full MCU Pin Name	Pin Selected for KBI Function by KBIEx Bit in KBIER
KBI0–KBI7	PTD0/KBI0–PTD7/KBI7	KBIE0–KBIE7

18.4 Functional Description

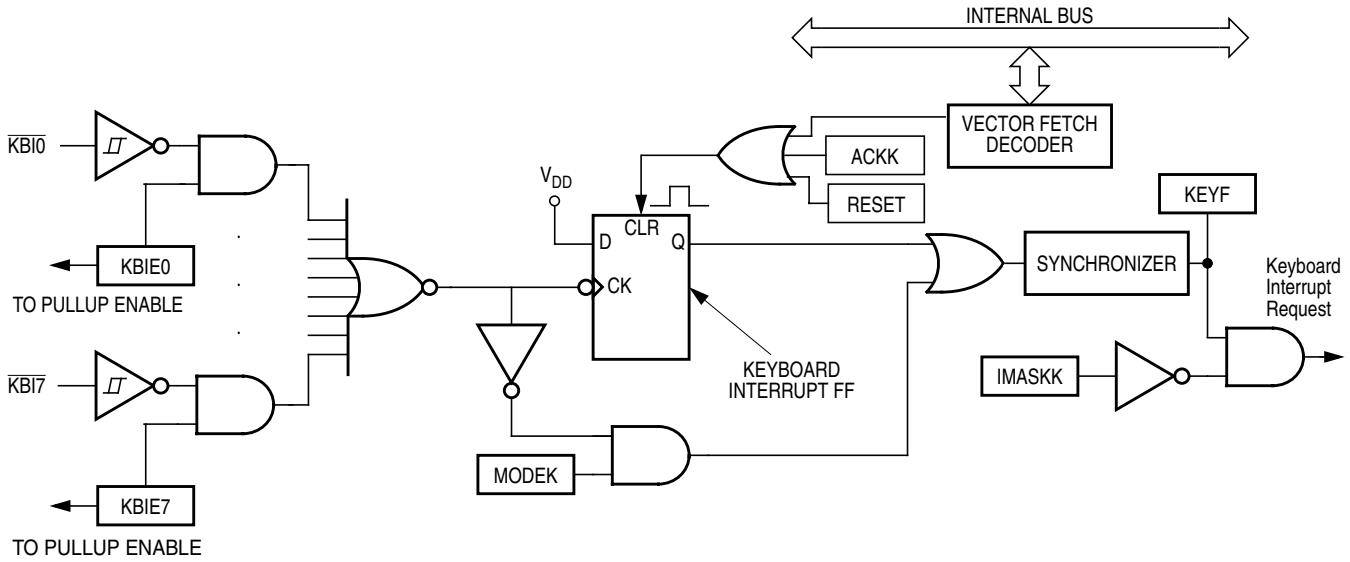


Figure 18-2. Keyboard Interrupt Block Diagram

Writing to the KBIE7–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port D pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin in port D also enables its internal pull-up device. A logic 0 applied to an enabled keyboard interrupt pin latches a keyboard interrupt request.

A keyboard interrupt is latched when one or more keyboard pins goes low after all were high. The MODEK bit in the keyboard status and control register controls the triggering mode of the keyboard interrupt.

- If the keyboard interrupt is edge-sensitive only, a falling edge on a keyboard pin does not latch an interrupt request if another keyboard pin is already low. To prevent losing an interrupt request on one pin because another pin is still low, software can disable the latter pin while it is low.
- If the keyboard interrupt is falling edge- and low level-sensitive, an interrupt request is present as long as any keyboard pin is low.

If the MODEK bit is set, the keyboard interrupt pins are both falling edge- and low level-sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register KBSCR. The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the vector address at locations $FFE0 and $FFE1.
- Return of all enabled keyboard interrupt pins to logic 1 — As long as any enabled keyboard interrupt pin is at logic 0, the keyboard interrupt remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to logic 1 may occur in any order.

If the MODEK bit is clear, the keyboard interrupt pin is falling-edge-sensitive only. With MODEK clear, a vector fetch or software clear immediately clears the keyboard interrupt request.

Reset clears the keyboard interrupt request and the MODEK bit, clearing the interrupt request even if a keyboard interrupt pin stays at logic 0.

The keyboard flag bit (KEYF) in the keyboard status and control register can be used to see if a pending interrupt exists. The KEYF bit is not affected by the keyboard interrupt mask bit (IMASKK) which makes it useful in applications where polling is preferred.

To determine the logic level on a keyboard interrupt pin, 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.

18.4.1 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pull-up to reach a logic 1. Therefore a false interrupt can occur as soon as the pin is enabled.

To prevent a false interrupt on keyboard initialization:

1. Mask keyboard interrupts by setting the IMASKK bit in the keyboard status and control register.
2. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.
3. Write to the ACKK bit in the keyboard status and control register to clear any false interrupts.
4. Clear the IMASKK bit.

An interrupt signal on an edge-triggered pin can be acknowledged immediately after enabling the pin. An interrupt signal on an edge- and level-triggered interrupt pin must be acknowledged after a delay that depends on the external load.

Another way to avoid a false interrupt:

1. Configure the keyboard pins as outputs by setting the appropriate DDR bits in data direction register.
2. Write logic 1s to the appropriate data register bits.
3. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

18.5 Keyboard Interrupt Registers

Two registers control the operation of the keyboard interrupt module:

- Keyboard Status and Control Register — $001A
- Keyboard Interrupt Enable Register — $001B

18.5.1 Keyboard Status and Control Register

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

Address: $001A

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

= Unimplemented

Figure 18-3. 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

18.5.2 Keyboard Interrupt Enable Register

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

Address: $001B

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

Figure 18-4. 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 = KBIx pin enabled as keyboard interrupt pin
0 = KBIx pin not enabled as keyboard interrupt pin

18.6 Low-Power Modes

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

18.6.1 Wait Mode

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

18.6.2 Stop Mode

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

18.7 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 SIM break flag control register (BFCR) enables software to clear status bits during the break state.

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

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

Chapter 19
Computer Operating Properly (COP)

19.1 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 configuration register 1 (CONFIG1).

19.2 Functional Description

Figure 19-1 shows the structure of the COP module.

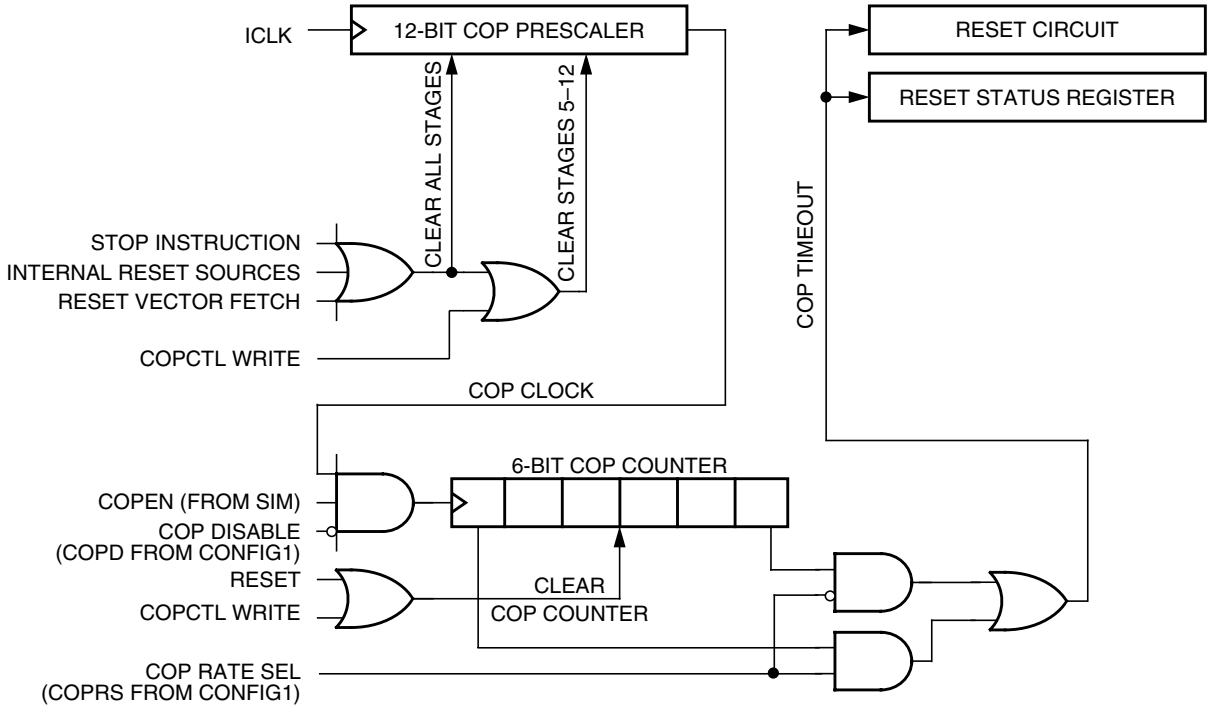


Figure 19-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by the 12-bit SIM counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after 262,128 or 8176 ICLK cycles, depending on the state of the COP rate select bit, COPRS, in the CONFIG1 register. With a 8176 ICLK cycle overflow option, a 88-kHz ICLK gives a COP timeout period of ~93ms. 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 $\overline{RST}$ pin low for 32 ICLK cycles and sets the COP bit in the SIM reset status register (SRSR).

In monitor mode, the COP is disabled if the $\overline{RST}$ pin or the $\overline{IRQ1}$ is held at V_{TST}. During the break state, V_{TST} on the $\overline{RST}$ 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.

19.3 I/O Signals

The following paragraphs describe the signals shown in Figure 19-1.

19.3.1 ICLK

ICLK is the internal oscillator output signal. See Chapter 22 Electrical Specifications for ICLK frequency specification.

19.3.2 STOP Instruction

The STOP instruction clears the COP prescaler.

19.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see 19.4 COP Control Register) clears the COP counter and clears bits 12 through 5 of the prescaler. Reading the COP control register returns the low byte of the reset vector.

19.3.4 Power-On Reset

The power-on reset (POR) circuit clears the COP prescaler 4096 ICLK cycles after power-up.

19.3.5 Internal Reset

An internal reset clears the COP prescaler and the COP counter.

19.3.6 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.

19.3.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the CONFIG1 register. (See Figure 19-2 . Configuration Register 1 (CONFIG1).)

19.3.8 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the CONFIG1 register.

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

Figure 19-2. Configuration Register 1 (CONFIG1)

COPRS — COP Rate Select Bit

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

1 = COP time out period = $2^{13} – 2^4$ ICLK cycles
0 = COP time out period = $2^{18} – 2^4$ ICLK cycles

COPD — COP Disable Bit

COPD disables the COP module.

1 = COP module disabled
0 = COP module enabled

19.4 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 19-3. COP Control Register (COPCTL)

19.5 Interrupts

The COP does not generate CPU interrupt requests.

19.6 Monitor Mode

When monitor mode is entered with V_{TST} on the $\overline{IRQ1}$ pin, the COP is disabled as long as V_{TST} remains on the $\overline{IRQ1}$ pin or the $\overline{RST}$ pin. When monitor mode is entered by having blank reset vectors and not having V_{TST} on the $\overline{IRQ1}$ pin, the COP is automatically disabled until a POR occurs.

19.7 Low-Power Modes

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

19.7.1 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.

19.7.2 Stop Mode

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

To prevent inadvertently turning off the COP with a STOP instruction, a configuration option is available that disables the STOP instruction. When the STOP bit in the configuration register has the STOP instruction is disabled, execution of a STOP instruction results in an illegal opcode reset.

19.8 COP Module During Break Mode

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

Chapter 20
Low-Voltage Inhibit (LVI)

20.1 Introduction

This section describes the low-voltage inhibit (LVI) module. The LVI module monitors the voltage on the V_{DD} pin and V_{REG} pin, and can force a reset when V_{DD} voltage falls below V_{TRIPF1}, or V_{REG} voltage falls below V_{TRIPF2}.

NOTE

The V_{REG} pin is the output of the internal voltage regulator and is guaranteed to meet operating specification as long as V_{DD} is within the MCU operating voltage.

The LVI feature is intended to provide the safe shutdown of the microcontroller and thus protection of related circuitry prior to any application V_{DD} voltage collapsing completely to an unsafe level. It is not intended that users operate the microcontroller at lower than the specified operating voltage, V_{DD}.

20.2 Features

Features of the LVI module include:

- Independent voltage monitoring circuits for V_{DD} and V_{REG}
- Independent disable for V_{DD} and V_{REG} LVI circuits
- 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 20-1. LVI I/O Register Summary

20.3 Functional Description

Figure 20-2 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 and the V_{REG} 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, and LVIREGD are in the CONFIG1 register. See Chapter 3 Configuration & Mask Option Registers (CONFIG & MOR) for details of the LVI configuration bits. Once

an LVI reset occurs, the MCU remains in reset until V_{DD} rises above V_{TRIPR1} and V_{REG} rises above V_{TRIPR2}, 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.

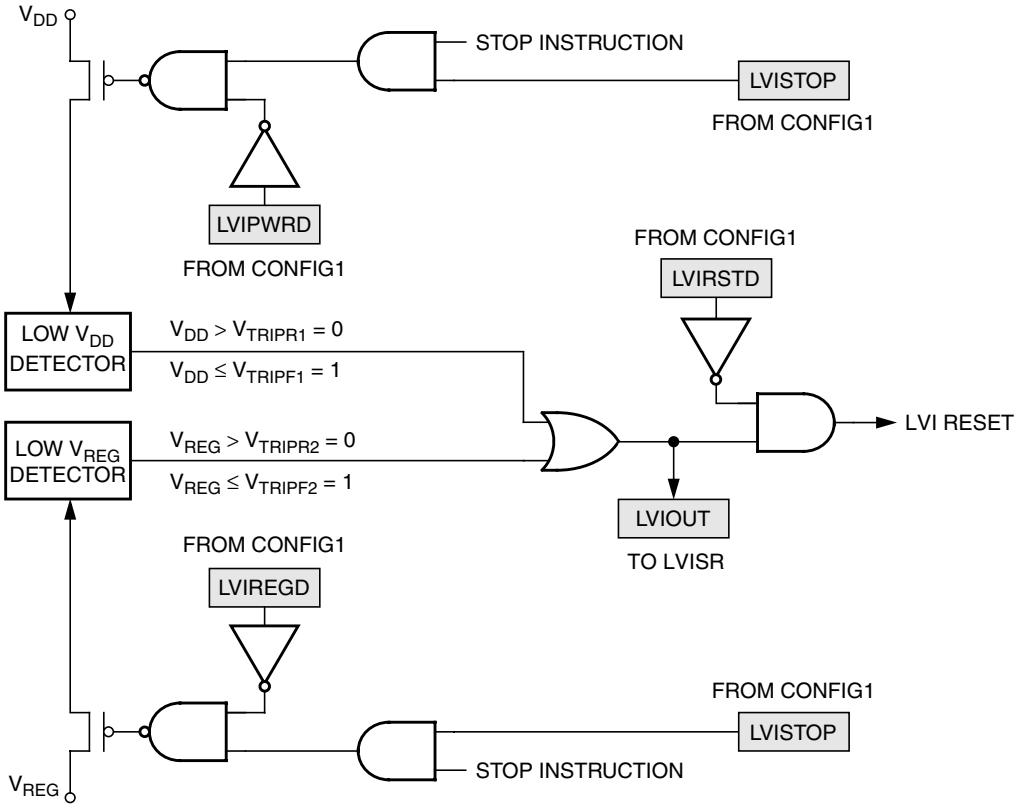


Figure 20-2. LVI Module Block Diagram

20.3.1 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.

20.3.2 Low V_{REG} Detector

The low V_{REG} detector circuit monitors the V_{REG} voltage and forces a LVI reset when the V_{REG} voltage falls below the trip voltage, V_{TRIPF2}. The V_{REG} LVI circuit can be disabled by the setting the LVIREGD bit in CONFIG1 register.

20.3.3 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.

20.3.4 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.

20.3.5 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}.

20.4 LVI Status Register

The LVI status register (LVISR) indicates if the V_{DD} voltage was detected below V_{TRIPF1} or V_{REG} voltage was detected below V_{TRIPF2}.

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 20-3. 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 20-1. LVIOUT Bit Indication

V_{DD}, V_{REG}	LVIOUT
$V_{DD} > V_{TRIPR1}$ and $V_{REG} > V_{TRIPR2}$	0
$V_{DD} < V_{TRIPF1}$ or $V_{DD} < V_{TRIPF2}$	1
$V_{TRIPF1} < V_{DD} < V_{TRIPR1}$ or $V_{TRIPF2} < V_{REG} < V_{TRIPR2}$	Previous value

20.5 LVI Interrupts

The LVI module does not generate interrupt requests.

20.6 Low-Power Modes

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

20.6.1 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.

20.6.2 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 21
Break Module (BRK)

21.1 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.

21.2 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: Write:	R	R	R	R	R	R	SBSW Note	R
		Reset:							0	
$FE03	SIM Break Flag Control Register (SBFCR)	Read: Write:	BCFE	R	R	R	R	R	R	R
		Reset:	0							
$FE0C	Break Address Register High (BRKH)	Read: Write:	Bit 15	14	13	12	11	10	9	Bit 8
		Reset:	0	0	0	0	0	0	0	0
$FE0D	Break Address Register Low (BRKL)	Read: Write:	Bit 7	6	5	4	3	2	1	Bit 0
		Reset:	0	0	0	0	0	0	0	0
$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 21-1. Break Module I/O Register Summary

21.3 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 SIM. The SIM then causes the CPU to load the instruction register with a software interrupt instruction (SWI) after completion of the current CPU instruction. The program counter vectors to $FFFC and $FFFD ($FEFC and $FEFD in monitor mode).

The following events can cause a break interrupt to occur:

- A CPU-generated address (the address in the program counter) matches the contents of the break address registers.
- Software writes a logic 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 is generated. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. Figure 21-2 shows the structure of the break module.

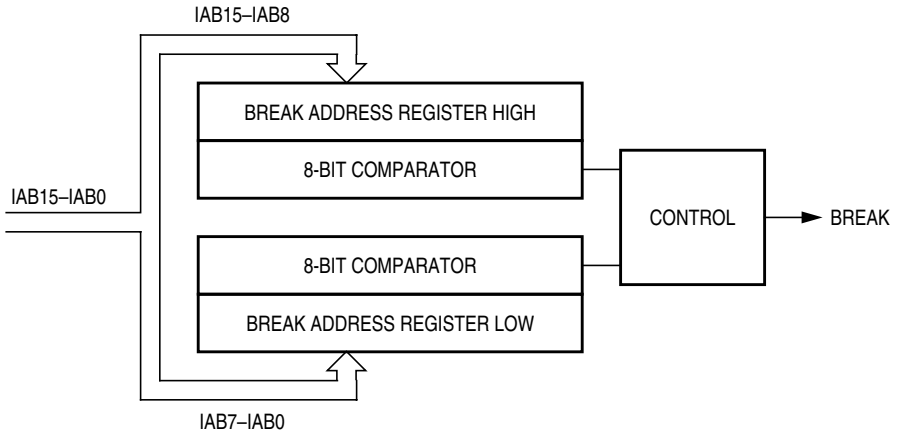


Figure 21-2. Break Module Block Diagram

21.3.1 Flag Protection During Break Interrupts

The system integration module (SIM) controls whether or not module status bits can be cleared during the break state. The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state.

21.3.2 CPU During Break Interrupts

When the internal address bus matches the value written in the break address registers or when software writes a 1 to the BRKA bit in the break status and control register, the CPU starts a break interrupt by:

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

The break interrupt timing is:

- When a break address is placed at the address of the instruction opcode, the instruction is not executed until after completion of the break interrupt routine.
- When a break address is placed at an address of an instruction operand, the instruction is executed before the break interrupt.
- When software writes a 1 to the BRKA bit, the break interrupt occurs just before the next instruction is executed.

By updating a break address and clearing the BRKA bit in a break interrupt routine, a break interrupt can be generated continuously.

CAUTION

A break address should be placed at the address of the instruction opcode. When software does not change the break address and clears the BRKA bit in the first break interrupt routine, the next break interrupt will not be generated after exiting the interrupt routine even when the internal address bus matches the value written in the break address registers.

21.3.3 TIMI and TIM2 During Break Interrupts

A break interrupt stops the timer counters.

21.3.4 COP During Break Interrupts

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

21.4 Low-Power Modes

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

21.4.1 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 7 System Integration Module (SIM)) Clear the BW bit by writing logic 0 to it.

21.5 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)

21.5.1 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 21-3. 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

21.5.2 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 21-4. 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 21-5. Break Address Register Low (BRKL)

21.5.3 SIM Break Status Register

The SIM break status register (SBSR) contains a flag to indicate that a break caused an exit from wait mode. This register is used only in emulation mode.

Address: $FE00

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

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

Figure 21-6. SIM Break Status Register (SBSR)

SBSW — Break Wait Bit

SBSW can be read within the break interrupt routine. The user can modify the return address on the stack by subtracting 1 from it. The following code is an example.

1 = Wait mode was exited by break interrupt
0 = Wait mode was not exited by break interrupt

21.5.4 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.

Address: $FE03

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

R = Reserved

Figure 21-7. 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 22
Electrical Specifications

22.1 Introduction

This section contains electrical and timing specifications.

22.2 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 5 V DC Electrical Characteristics for guaranteed operating conditions.

Table 22-1. Absolute Maximum Ratings

Characteristic[1]	Symbol	Value	Unit
Supply voltage	V_{DD}	−0.3 to +6.0	V
Input voltage All pins (except $\overline{IRQ1}$) $\overline{IRQ1}$ pin	V_{IN}	V_{SS}−0.3 to V_{DD} +0.3 V_{SS}−0.3 to 8.5	V 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 $V_{SS} \leq (V_{IN}$ or $V_{OUT}) \leq V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either V_{SS} or V_{DD}.)

22.3 Functional Operating Range

Table 22-2. Operating Range

Characteristic	Symbol	Value	Unit
Operating temperature range	T_A	–40 to +85	°C
Operating voltage range	V_{DD}	4.5 to 5.5 (1)	V

1. Functionality of the device has been characterized down to the LVI trip voltage, but user should ensure that the device supply voltage is within the specified range of 4.5 to 5.5V after power-up in order for the LVI circuit to configure properly.

22.4 Thermal Characteristics

Table 22-3. Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 42-Pin SDIP 44-Pin QFP 48-Pin LQFP	θ_{JA}	 60 95 80	 °C/W °C/W °C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273\ °C)$	W
Constant(2)	K	$P_D \times (T_A + 273\ °C) + P_D^2 \times \theta_{JA}$	W/°C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C
Maximum junction temperature	T_{JM}	100	°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.

22.5 5V DC Electrical Characteristics

Table 22-4. DC Electrical Characteristics (5V)

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Output high voltage (I_{LOAD} = –12mA) PTA[0:7], PTB[4:7], PTC[0:5], PTD[0:7] (I_{LOAD} = –15mA) PTA[0:7], PTB[4:7], PTC[0:5], PTD[0:7]	 V_{OH} V_{OH}	 V_{DD}–0.8 V_{DD}–1.0	 — —	 — —	 V V
Output low voltage (I_{LOAD} = 6mA) PTA[0:7], PTB[4:7], PTC[0:5], PTD[0:7] (I_{LOAD} = 12mA) PTB[0:3], PTC[6:7] (I_{LOAD} = 15mA) PTA[0:7], PTB[4:7], PTC[0:5], PTD[0:7] (I_{LOAD} = 15mA) PTB[0:3], PTC[6:7] (I_{LOAD} = 15mA) as TxD, RxD, SCTxD, SCRxD (I_{LOAD} = see Table 22-8) as SDA, SCL	 V_{OL} V_{OL} V_{OL} V_{OL} V_{OLSCI} V_{OLIIC}	 — — — — — —	 — — — — — —	 0.4 0.4 0.8 0.6 0.4 0.4	 V V V V V V
LED sink current (V_{OL} = 3V) PTA[0:7]	I_{OL}	9	15	25	mA
Input high voltage PTA[0:7], PTB[0:7], PTC[0:7], PTD[0:7], $\overline{RST}$, $\overline{IRQ1}$ OSC1	V_{IH}	$0.7 \times V_{DD}$ $0.7 \times V_{REG}$	— —	V_{DD} V_{REG}	V V
Input low voltage PTA[0:7], PTB[0:7], PTC[0:7], PTD[0:7], $\overline{RST}$, $\overline{IRQ1}$ OSC1	V_{IL}	V_{SS} V_{SS}	— —	$0.3 \times V_{DD}$ $0.3 \times V_{REG}$	V V
V_{DD} supply current, f_{OP} = 8 MHz Run[3] Wait[4] Stop with OSC, TBM, and LVI modules on[5] with OSC and TBM modules on[5] all modules off[6]	 I_{DD}	 — — — — —	 10 2.5 1.8 1 20	 20 10 2.5 1.5 125	 mA mA mA mA µA
Digital I/O ports Hi-Z leakage current	I_{IL}	—	—	± 10	µA
Input current	I_{IN}	—	—	± 1	µA
Capacitance Ports (as input or output)	C_{OUT} C_{IN}	— —	— —	12 8	pF pF
POR rearm voltage[7]	V_{POR}	0	—	100	mV
POR rise time ramp rate[8]	R_{POR}	0.035	—	—	V/ms
Monitor mode entry voltage	V_{TST}	$1.4 \times V_{DD}$		8.5	V
Pullup resistors[9] PTD[0:7] RST, $\overline{IRQ1}$, $\overline{IRQ2}$	 R_{PU1} R_{PU2}	 21 21	 27 27	 39 39	 kΩ kΩ
Low-voltage inhibit, trip falling voltage1	V_{TRIPF1}	2.90	3.10	3.30	V
Low-voltage inhibit, trip rising voltage1	V_{TRIPR1}	3.25	3.45	3.65	V

Table 22-4. DC Electrical Characteristics (5V)

Characteristic[1]	Symbol	Min	Typ[2]	Max	Unit
Low-voltage inhibit, trip voltage2	V_{TRIPF2}	2.25	2.45	2.65	V
V_{REG}[10]	V_{REG}	2.25	2.50	2.75	V

1. V_{DD} = 4.5 to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. Typical values reflect average measurements at midpoint of voltage range, 25 °C only.
3. Run (operating) I_{DD} measured using external 32MHz clock to OSC1; all inputs 0.2 V from rail; no dc loads; less than 100pF on all outputs; C_L = 20 pF on OSC2; all ports configured as inputs; OSC2 capacitance linearly affects run I_{DD}; measured with all modules enabled.
4. Wait I_{DD} measured using external 32MHz to OSC1; all inputs 0.2 V from rail; no dc loads; less than 100 pF on all outputs. C_L = 20 pF on OSC2; all ports configured as inputs; OSC2 capacitance linearly affects wait I_{DD}.
5. STOP I_{DD} measured using external 8MHz clock to OSC1; no port pins sourcing current.
6. STOP I_{DD} measured with OSC1 grounded; no port pins sourcing current.
7. Maximum is highest voltage that POR is guaranteed. The rearm voltage is triggered by V_{REG}.
8. If minimum V_{DD} is not reached before the internal POR reset is released, $\overline{RST}$ must be driven low externally until minimum V_{DD} is reached.
9. R_{PU1} and R_{PU2} are measured at V_{DD} = 5.0V
10. Measured from $V_{DD} = V_{TRIPF1}$ (Min) to 5.5 V.

22.6 5V Control Timing

Table 22-5. Control Timing (5V)

Characteristic[1]	Symbol	Min	Max	Unit
Internal operating frequency[2]	f_{OP}	—	8	MHz
$\overline{RST}$ input pulse width low[3]	t_{RL}	100	—	ns

1. V_{DD} = 4.5 to 5.5 Vdc, 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.

22.7 5V Oscillator Characteristics

Table 22-6. Oscillator Specifications (5V)

Characteristic[1]	Symbol	Min	Typ	Max	Unit
Internal oscillator clock frequency	f_{ICLK}	64k	88k[2]	104k	Hz
External reference clock to OSC1[3]	f_{OSC}	dc		32M	Hz
Crystal reference frequency[4]	$f_{XTALCLK}$	1M	—	8M	Hz
Crystal load capacitance[5]	C_L	—	—	—	
Crystal fixed capacitance[5]	C_1	—	$2 \times C_L$	—	
Crystal tuning capacitance[5]	C_2	—	$2 \times C_L$	—	
Feedback bias resistor	R_B	—	1M	—	Ω
Series resistor[5]	R_S	—	0	—	Ω
External RC clock frequency	f_{RCCLK}			7.6M	Hz
RC oscillator external R	R_{EXT}	See Figure 22-1			Ω
RC oscillator external C	C_{EXT}	—	10	—	pF

1. The oscillator circuit operates at V_{REG}.
2. Typical value reflect average measurements at midpoint of voltage range, 25 °C only.
3. No more than 10% duty cycle deviation from 50%. The max. frequency is limited by an EMC filter.
4. Fundamental mode crystals only.
5. Consult crystal vendor data sheet.

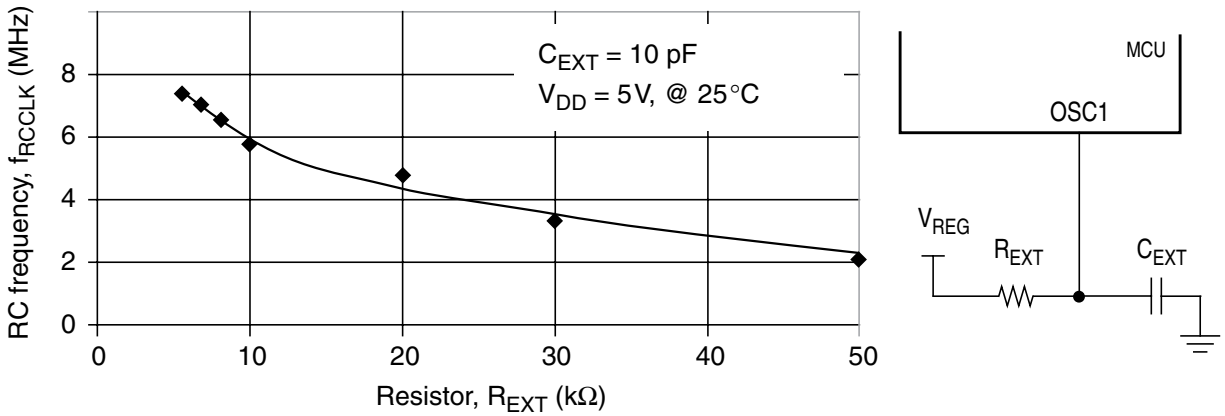


Figure 22-1. RC vs. Frequency

22.8 5V ADC Electrical Characteristics

Table 22-7. ADC Electrical Characteristics (5V)

Characteristic[1]	Symbol	Min	Max	Unit	Notes
Supply voltage	V_{DDA}	4.5	5.5	V	V_{DDA} is an dedicated pin and should be tied to V_{DD} on the PCB with proper decoupling.
Input range	V_{ADIN}	0	V_{DDA}	V	$V_{ADIN} \leq V_{DDA}$
Resolution	B_{AD}	10	10	bits	
Absolute accuracy	A_{AD}	—	± 1.5	LSB	Includes quantization. ±0.5 LSB = ±1 ADC step.
ADC internal clock	f_{ADIC}	500k	1.048M	Hz	$t_{ADIC} = 1/f_{ADIC}$
Conversion range	R_{AD}	V_{REFL}	V_{REFH}	V	
ADC voltage reference high	V_{REFH}	—	$V_{DDA} + 0.1$	V	
ADC voltage reference low	V_{REFL}	$V_{SSA} - 0.1$	—	V	
Conversion time	t_{ADC}	16	17	t_{ADIC} cycles	
Sample time	t_{ADS}	5	—	t_{ADIC} cycles	
Monotonicity	M_{AD}	Guaranteed			
Zero input reading	Z_{ADI}	000	001	HEX	$V_{ADIN} = V_{REFL}$
Full-scale reading	F_{ADI}	3FD	3FF	HEX	$V_{ADIN} = V_{REFH}$
Input capacitance	C_{ADI}	—	20	pF	Not tested.
Input impedance	R_{ADI}	20M	—	Ω	
V_{REFH}/V_{REFL}	I_{VREF}	—	1.6	mA	Not tested.

1. V_{DD} = 4.5 to 5.5 Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.

22.9 MMIIC Electrical Characteristics

Table 22-8. MMIIC DC Electrical Characteristics

Characteristic[1]	Symbol	Min	Typ	Max	Unit	Comments
Input low	V_{IL}	–0.5	—	0.8	V	Data, clock input low.
Input high	V_{IH}	2.1	—	5.5	V	Data, clock input high.
Output low	V_{OL}	—	—	0.4	V	Data, clock output low; @$I_{PULLUP,MAX}$
Input leakage	I_{LEAK}	—	—	± 5	µA	Input leakage current
Pullup current	I_{PULLUP}	100	—	350	µA	Current through pull-up resistor or current source. See note.[2]

1. V_{DD} = 4.5 to 5.5Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H, unless otherwise noted.
2. The I_{PULLUP} (max) specification is determined primarily by the need to accommodate a maximum of 1.1kΩ equivalent series resistor of removable SMBus devices, such as the smart battery, while maintaining the V_{OL} (max) of the bus.

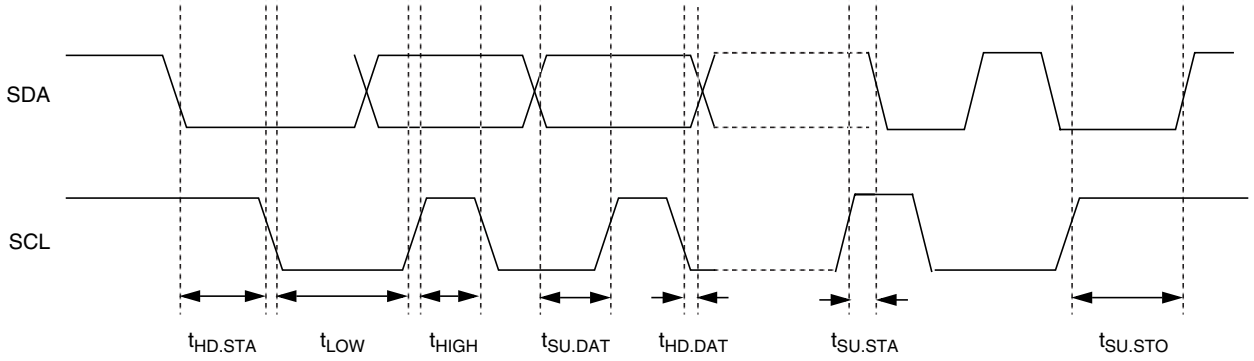


Figure 22-2. MMIIC Signal Timings

See Table 22-9 for MMIIC timing parameters.

Table 22-9. MMIIC Interface Input/Output Signal Timing

Characteristic	Symbol	Min	Typ	Max	Unit	Comments
Operating frequency	f_{SMB}	10	—	100	kHz	MMIIC operating frequency
Bus free time	t_{BUF}	4.7	—	—	µs	Bus free time between STOP and START condition
Repeated start hold time.	$t_{HD.STA}$	4.0	—	—	µs	Hold time after (repeated) START condition. After this period, the first clock is generated.
Repeated start setup time.	$t_{SU.STA}$	4.7	—	—	µs	Repeated START condition setup time.
Stop setup time	$t_{SU.STO}$	4.0	—	—	µs	Stop condition setup time.
Hold time	$t_{HD.DAT}$	300	—	—	ns	Data hold time.
Setup time	$t_{SU.DAT}$	250	—	—	ns	Data setup time.
Clock low time-out	$t_{TIMEOUT}$	25	—	35	ms	Clock low time-out.(1)
Clock low	t_{LOW}	4.7	—	—	µs	Clock low period
Clock high	t_{HIGH}	4.0	—	—	µs	Clock high period.(2)
Slave clock low extend time	$t_{LOW.SEXT}$	—	—	25	ms	Cumulative clock low extend time (slave device)(3)
Master clock low extend time	$t_{LOW.MEXT}$	—	—	10	ms	Cumulative clock low extend time (master device) (4)
Fall time	t_F	—	—	300	ns	Clock/Data Fall Time(5)
Rise time	t_R	—	—	1000	ns	Clock/Data Rise Time(5)

1. Devices participating in a transfer will timeout when any clock low exceeds the value of $T_{TIMEOUT}$ min. of 25ms. Devices that have detected a timeout condition must reset the communication no later than $T_{TIMEOUT}$ max of 35ms. The maximum value specified must be adhered to by both a master and a slave as it incorporates the cumulative limit for both a master (10 ms) and a slave (25 ms).
   Software should turn-off the MMIIC module to release the SDA and SCL lines.
2. $T_{HIGH\ MAX}$ provides a simple guaranteed method for devices to detect the idle conditions.
3. $T_{LOW.SEXT}$ is the cumulative time a slave device is allowed to extend the clock cycles in one message from the initial start to the stop. If a slave device exceeds this time, it is expected to release both its clock and data lines and reset itself.
4. $T_{LOW.MEXT}$ is the cumulative time a master device is allowed to extend its clock cycles within each byte of a message as defined from start-to-ack, ack-to-ack, or ack-to-stop.
5. Rise and fall time is defined as follows: $T_R = (V_{ILMAX} – 0.15)$ to $(V_{IHMIN} + 0.15)$, $T_F = 0.9 \times V_{DD}$ to $(V_{ILMAX} – 0.15)$.

22.10 CGM Electrical Specification

Table 22-10. CGM Electrical Specifications

Characteristic	Symbol	Min	Typ	Max	Unit
Reference frequency	f_{RDV}	1	—	8	MHz
Range nominal multiplies	f_{NOM}	—	125	—	kHz
VCO center-of-range frequency	f_{VRS}	125k	—	40M	Hz
VCO range linear range multiplier	L	1	—	255	
VCO power-of-two-range multiplier	2^E	1	—	4	
VCO multiply factor	N	1	—	4095	
VCO prescale multiplier	2^P	1	—	8	
Reference divider factor	R	1	1	15	
VCO operating frequency	f_{VCLK}	125k	—	40M	Hz
Manual acquisition time	t_{LOCK}	—	—	50	ms
Automatic lock time	t_{LOCK}	—	—	50	ms
PLL jitter[(1)]	f_J	0	—	$f_{RCLK} \times 0.025\% \times 2^P N/4$	Hz

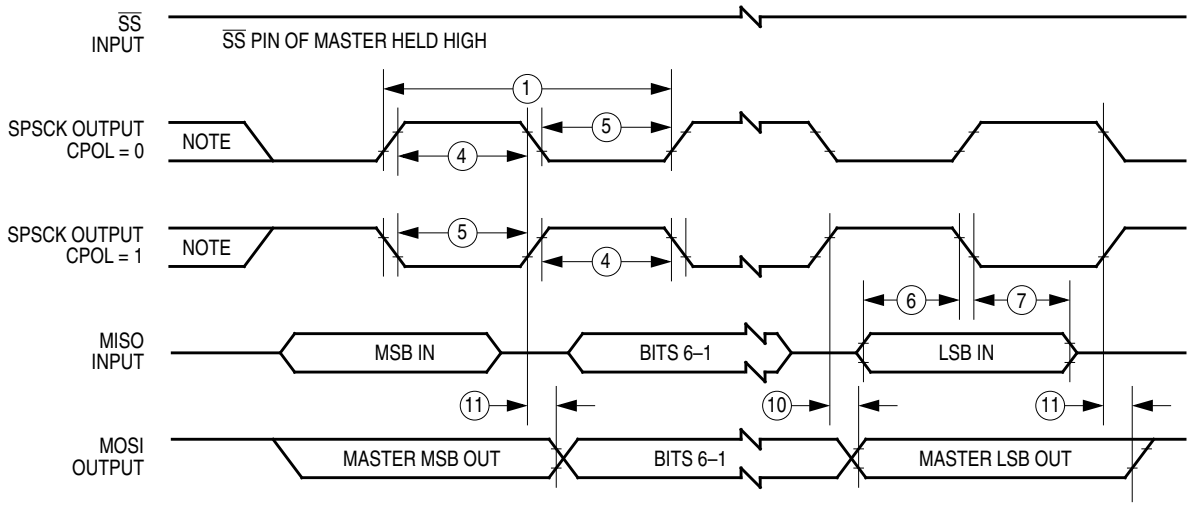
1. Deviation of average bus frequency over 2ms. N = VCO multiplier.

22.11 5V SPI Characteristics

Table 22-11. SPI Characteristics (5V)

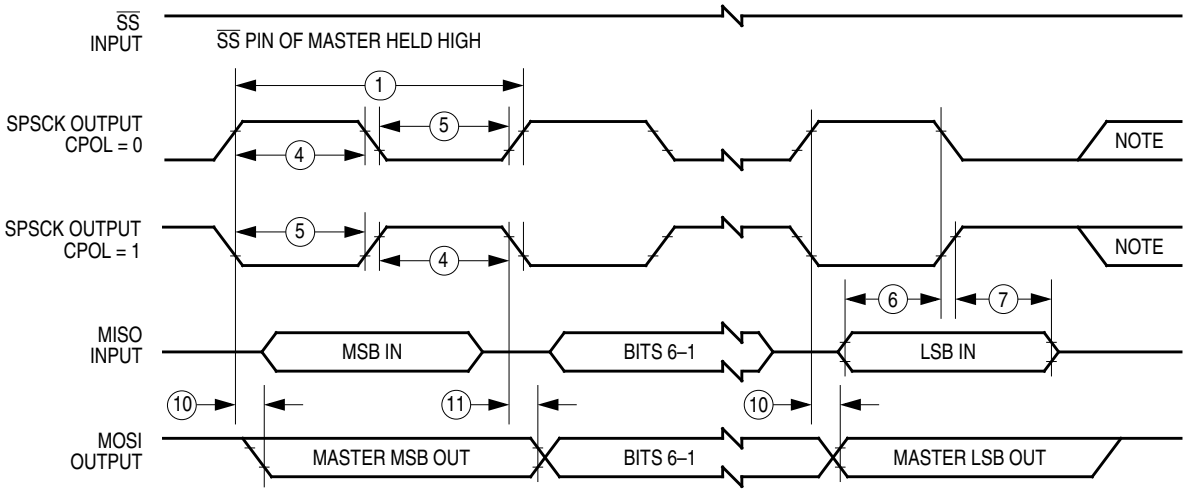
Diagram Number[1]	Characteristic[2]	Symbol	Min	Max	Unit
	Operating frequency Master Slave	$f_{OP(M)}$ $f_{OP(S)}$	$f_{OP}/128$ dc	$f_{OP}/2$ f_{OP}	MHz MHz
1	Cycle time Master Slave	$t_{CYC(M)}$ $t_{CYC(S)}$	2 1	128 —	t_{CYC} t_{CYC}
2	Enable lead time	$t_{Lead(S)}$	1	—	t_{CYC}
3	Enable lag time	$t_{Lag(S)}$	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 22-3 and Figure 22-4.
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.

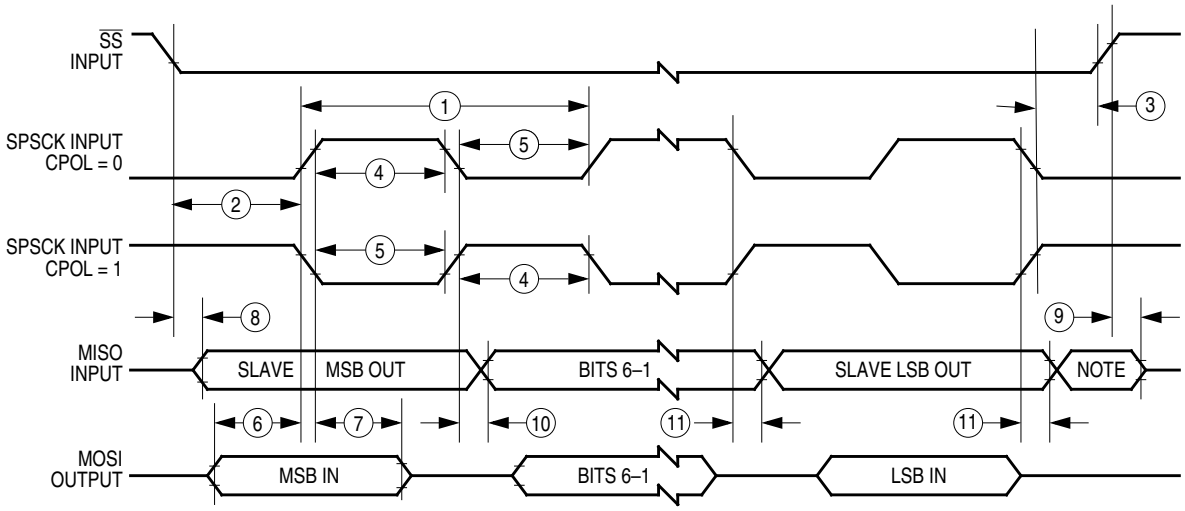
a) SPI Master Timing (CPHA = 0)



Note: This last clock edge is generated internally, but is not seen at the SPSCK pin.

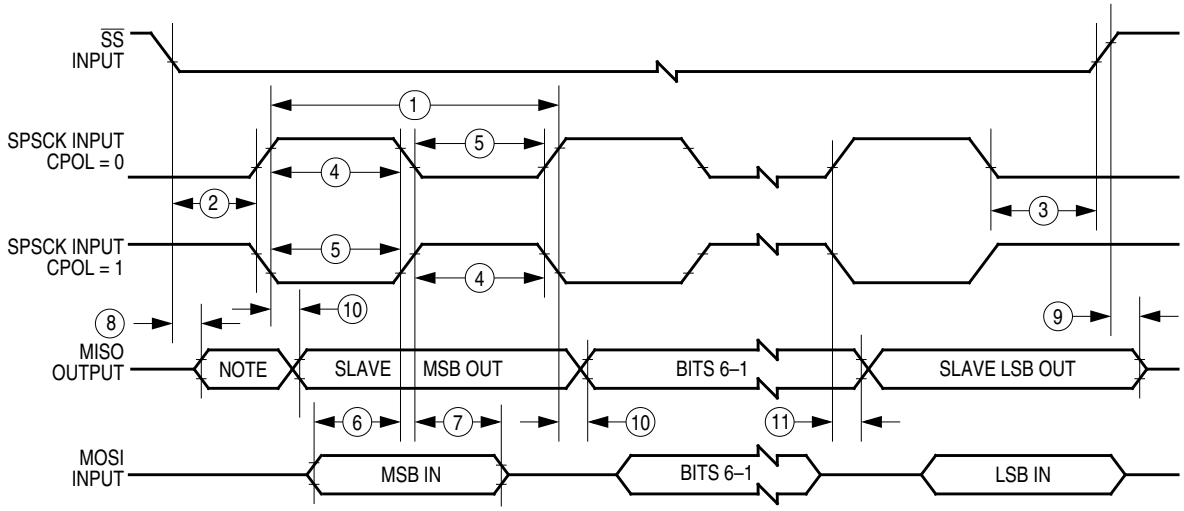
b) SPI Master Timing (CPHA = 1)

Figure 22-3. 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 22-4. SPI Slave Timing

22.12 Memory Characteristics

Table 22-12. Memory Characteristics

Characteristic	Symbol	Min.	Max.	Unit
Data retention voltage	V_{RDR}	1.3	—	V
Number of rows per page		8		Rows
Number of bytes per page		512		Bytes
Read bus clock frequency	f_{read}(1)	32k	8M	Hz
Page erase time	t_{erase}(2)	20	—	ms
Mass erase time	t_{me}(3)	200	—	ms
PGM/ERASE to HVEN setup time	t_{nvs}	5	—	µs
High-voltage hold time	t_{nvh}	5	—	µs
High-voltage hold time (mass erase)	t_{nvh1}	100	—	µs
Program hold time	t_{pgs}	10	—	µs
Program time	t_{prog}	20	40	µs
Address/data setup time	t_{ads}	20	—	ns
Address/data hold time	t_{adh}	—	30	ns
Recovery time	t_{rcv}(4)	1	—	µs
Cumulative HV period	t_{hv}(5)	—	8	ms
Row erase endurance(6)	—	10k	—	Cycles
Row program endurance(7)	—	10k	—	Cycles
Data retention time(8)	—	10	—	Years

1. f_{read} is defined as the frequency range for which the FLASH memory can be read.
2. If the page erase time is longer than t_{erase} (Min.), there is no erase-disturb, but it reduces the endurance of the FLASH memory.
3. If the mass erase time is longer than t_{me} (Min.), there is no erase-disturb, but it reduces the endurance of the FLASH memory.
4. It is defined as the time it needs before the FLASH can be read after turning off the high voltage charge pump, by clearing HVEN to logic 0.
5. t_{hv} is the cumulative high voltage programming time to the same row before next erase, and the same address can not be programmed twice before next erase.
6. The minimum row endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase/program cycles.
7. The minimum row endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase/program cycle.
8. The FLASH is guaranteed to retain data over the entire operating temperature range for at least the minimum time specified.

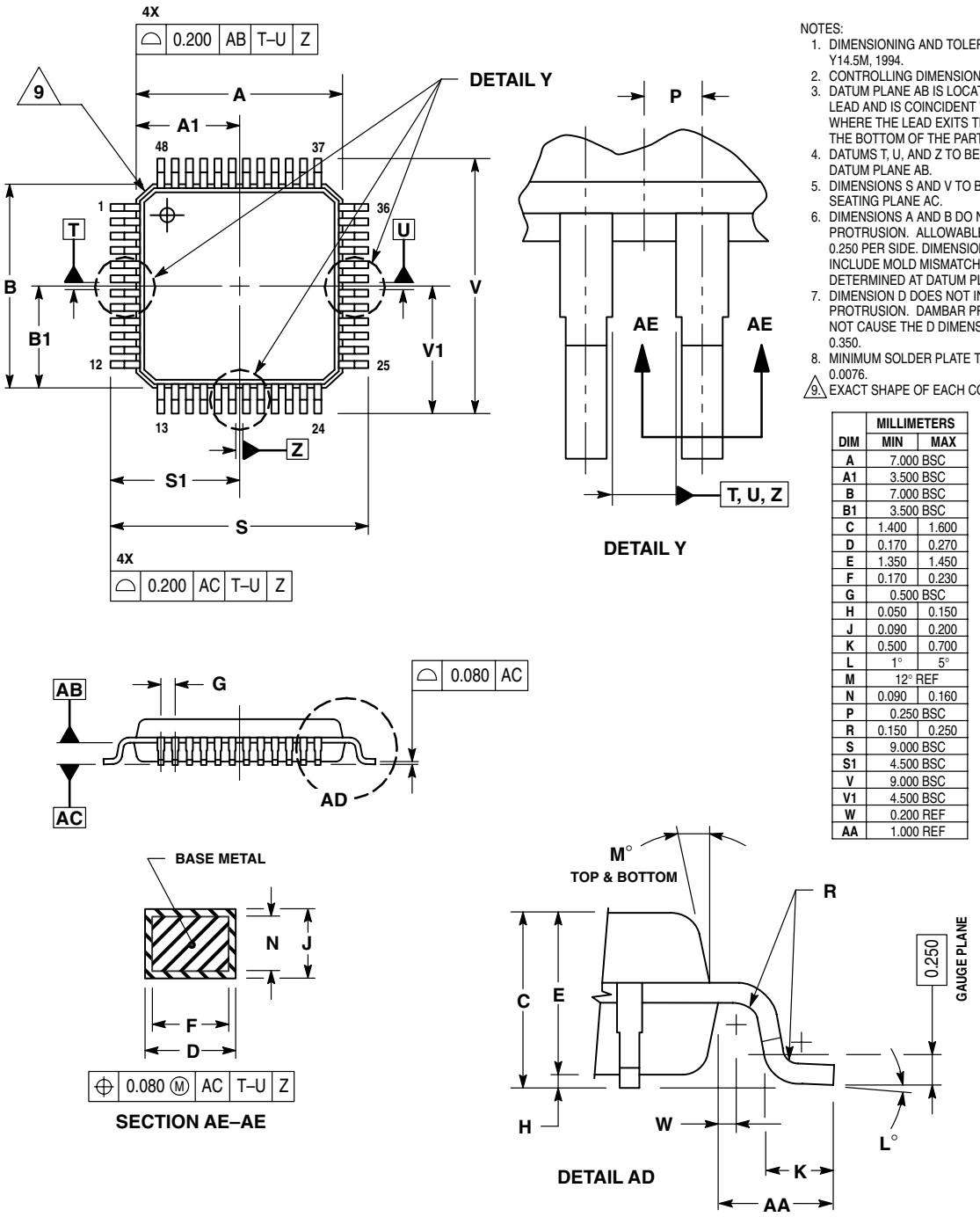
Chapter 23
Mechanical Specifications

23.1 Introduction

This section gives the dimensions for:

- 48-pin plastic low-profile quad flat pack (case #932)
- 44-pin plastic quad flat pack (case #824A)
- 42-pin shrink dual in-line package (case #858)

23.2 48-Pin Low-Profile Quad Flat Pack (LQFP)

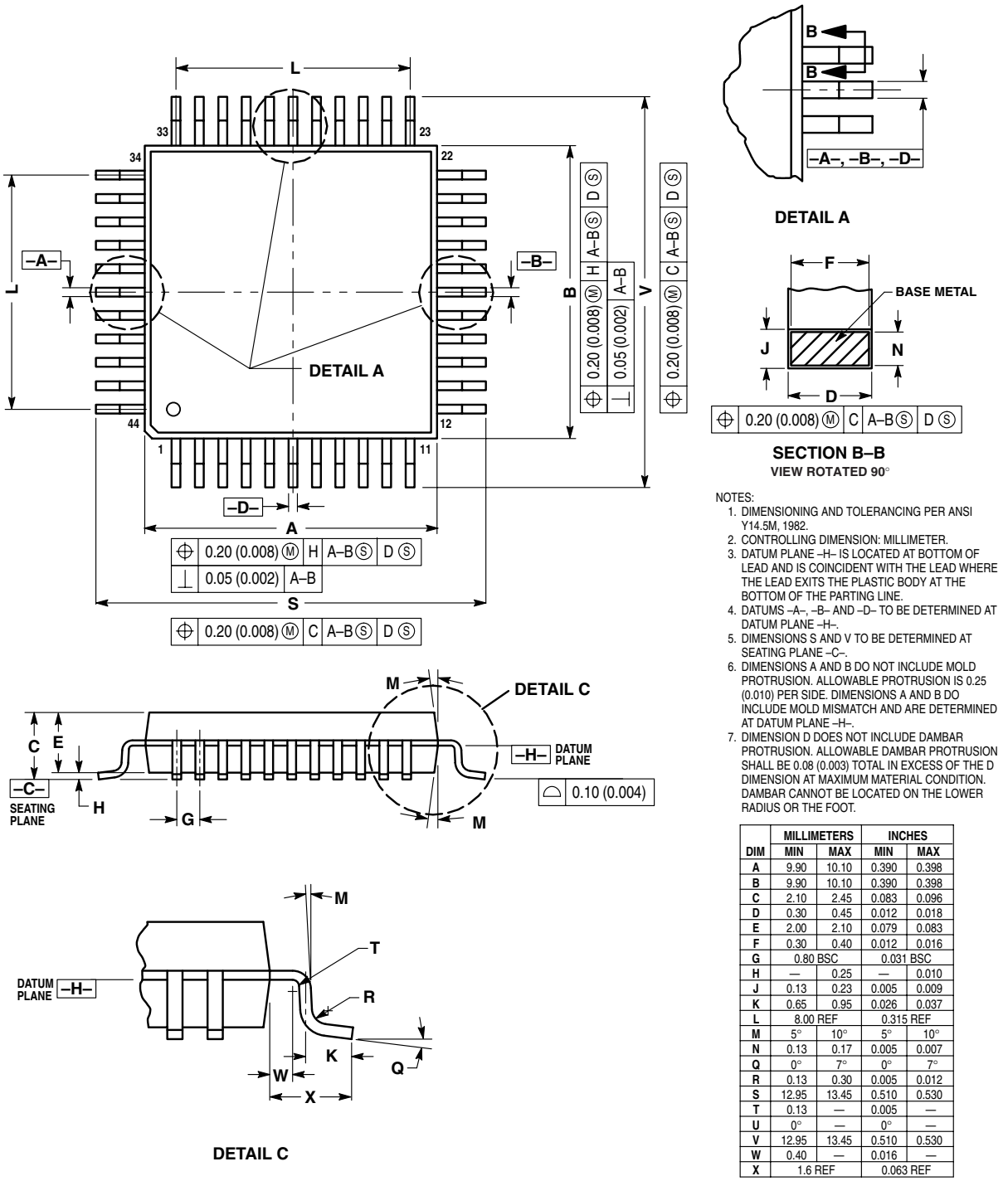


NOTES:
1. DIMENSIONING AND TOLERANCING PER ASME Y14.5M, 1994.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE AB IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS T, U, AND Z TO BE DETERMINED AT DATUM PLANE AB.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE AC.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.250 PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE AB.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.350.
8. MINIMUM SOLDER PLATE THICKNESS SHALL BE 0.0076.
9. EXACT SHAPE OF EACH CORNER IS OPTIONAL.

	MILLIMETERS	
DIM	MIN	MAX
A	7.000 BSC	
A1	3.500 BSC	
B	7.000 BSC	
B1	3.500 BSC	
C	1.400	1.600
D	0.170	0.270
E	1.350	1.450
F	0.170	0.230
G	0.500 BSC	
H	0.050	0.150
J	0.090	0.200
K	0.500	0.700
L	1°	5°
M	12° REF	
N	0.090	0.160
P	0.250 BSC	
R	0.150	0.250
S	9.000 BSC	
S1	4.500 BSC	
V	9.000 BSC	
V1	4.500 BSC	
W	0.200 REF	
AA	1.000 REF	

Figure 23-1. 48-Pin LQFP (Case #932)

23.3 44-Pin Quad Flat Pack (QFP)

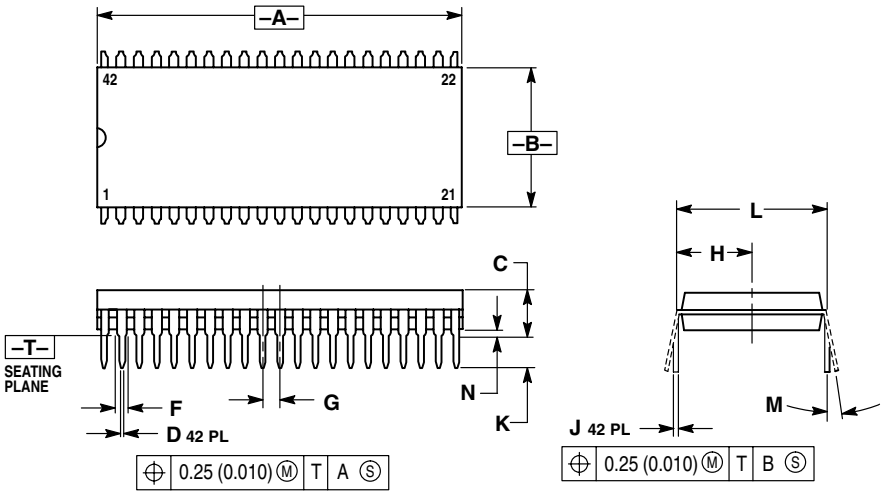


NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: MILLIMETER.
3. DATUM PLANE –H– IS LOCATED AT BOTTOM OF LEAD AND IS COINCIDENT WITH THE LEAD WHERE THE LEAD EXITS THE PLASTIC BODY AT THE BOTTOM OF THE PARTING LINE.
4. DATUMS –A–, –B– AND –D– TO BE DETERMINED AT DATUM PLANE –H–.
5. DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE –C–.
6. DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE –H–.
7. DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 (0.003) TOTAL IN EXCESS OF THE D DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT.

	MILLIMETERS		INCHES	
DIM	MIN	MAX	MIN	MAX
A	9.90	10.10	0.390	0.398
B	9.90	10.10	0.390	0.398
C	2.10	2.45	0.083	0.096
D	0.30	0.45	0.012	0.018
E	2.00	2.10	0.079	0.083
F	0.30	0.40	0.012	0.016
G	0.80 BSC		0.031 BSC	
H	—	0.25	—	0.010
J	0.13	0.23	0.005	0.009
K	0.65	0.95	0.026	0.037
L	8.00 REF		0.315 REF	
M	5°	10°	5°	10°
N	0.13	0.17	0.005	0.007
Q	0°	7°	0°	7°
R	0.13	0.30	0.005	0.012
S	12.95	13.45	0.510	0.530
T	0.13	—	0.005	—
U	0°	—	0°	—
V	12.95	13.45	0.510	0.530
W	0.40	—	0.016	—
X	1.6 REF		0.063 REF	

Figure 23-2. 44-Pin QFP (Case #824A)

23.4 42-Pin Shrink Dual In-Line Package (SDIP)



NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. DIMENSION L TO CENTER OF LEAD WHEN FORMED PARALLEL.
4. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH. MAXIMUM MOLD FLASH 0.25 (0.010).

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	1.435	1.465	36.45	37.21
B	0.540	0.560	13.72	14.22
C	0.155	0.200	3.94	5.08
D	0.014	0.022	0.36	0.56
F	0.032	0.046	0.81	1.17
G	0.070 BSC		1.778 BSC	
H	0.300 BSC		7.62 BSC	
J	0.008	0.015	0.20	0.38
K	0.115	0.135	2.92	3.43
L	0.600 BSC		15.24 BSC	
M	0°	15°	0°	15°
N	0.020	0.040	0.51	1.02

Figure 23-3. 42-Pin SDIP (Case #858)

Chapter 24
Ordering Information

24.1 Introduction

This section contains device ordering numbers.

24.2 MC Order Numbers

Table 24-1. MC Order Numbers

MC Order Number	RAM Size (bytes)	FLASH Size (bytes)	Package	Operating Temperature Range
MC908AP64ACB	2,048	62,368	42-pin SDIP	−40 to +85 °C
MC908AP64ACFB	2,048	62,368	44-pin QFP	−40 to +85 °C
MC908AP64ACFA	2,048	62,368	48-pin LQFP	−40 to +85 °C
MC908AP32ACB	2,048	32,768	42-pin SDIP	−40 to +85 °C
MC908AP32ACFB	2,048	32,768	44-pin QFP	−40 to +85 °C
MC908AP32ACFA	2,048	32,768	48-pin LQFP	−40 to +85 °C
MC908AP16ACB	1,024	16,384	42-pin SDIP	−40 to +85 °C
MC908AP16ACFB	1,024	16,384	44-pin QFP	−40 to +85 °C
MC908AP16ACFA	1,024	16,384	48-pin LQFP	−40 to +85 °C
MC908AP8ACB	1,024	8,192	42-pin SDIP	−40 to +85 °C
MC908AP8ACFB	1,024	8,192	44-pin QFP	−40 to +85 °C
MC908AP8ACFA	1,024	8,192	48-pin LQFP	−40 to +85 °C

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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

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

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2007. All rights reserved.

MC68HC908AP64A
Rev. 3, 10/2007