# MC68HC908AZ60A
# MC68HC908AS60A
# MC68HC908AZ60E

Data Sheet

***M68HC08***
***Microcontrollers***

MC68HC908AZ60A
Rev. 6
05/2006

freescale.com

freescale™
semiconductor

# MC68HC908AZ60A
# MC68HC908AS60A

**Data Sheet**

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

http://freescale.com

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

© Freescale Semiconductor, Inc., 2006. All rights reserved.

# List of Chapters


Chapter 1 General Description . . . . . 21
Chapter 2 Memory Map . . . . . 35
Chapter 3 Random-Access Memory (RAM) . . . . . 49
Chapter 4 FLASH-1 Memory . . . . . 51
Chapter 5 FLASH-2 Memory . . . . . 61
Chapter 6 EEPROM-1 Memory . . . . . 71
Chapter 7 EEPROM-2 Memory . . . . . 85
Chapter 8 Central Processor Unit (CPU) . . . . . 99
Chapter 9 System Integration Module (SIM) . . . . . 111
Chapter 10 Clock Generator Module (CGM) . . . . . 127
Chapter 11 Configuration Register (CONFIG-1) . . . . . 145
Chapter 12 Configuration Register (CONFIG-2) . . . . . 147
Chapter 13 Break Module (BRK) . . . . . 149
Chapter 14 Monitor ROM (MON) . . . . . 153
Chapter 15 Computer Operating Properly (COP) . . . . . 163
Chapter 16 Low-Voltage Inhibit (LVI) . . . . . 167
Chapter 17 External Interrupt Module (IRQ) . . . . . 171
Chapter 18 Serial Communications Interface (SCI) . . . . . 177
Chapter 19 Serial Peripheral Interface (SPI) . . . . . 205
Chapter 20 Timer Interface Module B (TIMB) . . . . . 227
Chapter 21 Programmable Interrupt Timer (PIT) . . . . . 243
Chapter 22 Input/Output Ports . . . . . 249
Chapter 23 MSCAN Controller (MSCAN08) . . . . . 267

Chapter 24 Keyboard Module (KBI) . . . . . 301
Chapter 25 Timer Interface Module A (TIMA) . . . . . 307
Chapter 26 Analog-to-Digital Converter (ADC) . . . . . 327
Chapter 27 Byte Data Link Controller (BDLC) . . . . . 335
Chapter 28 Electrical Specifications . . . . . 365
Appendix A MC68HC908AS60 and MC68HC908AZ60 . . . . . 381
Appendix B MC68HC908AZ60E . . . . . 385
Revision History . . . . . 401
Glossary . . . . . 405

# Table of Contents

## Chapter 1
## General Description


1.1 Introduction . . . 21
1.2 Features . . . 21
1.3 MCU Block Diagram . . . 22
1.4 Pin Assignments . . . 25
1.4.1 Power Supply Pins ($V_{DD}$ and $V_{SS}$) . . . 28
1.4.2 Oscillator Pins (OSC1 and OSC2) . . . 28
1.4.3 External Reset Pin ($\overline{RST}$) . . . 28
1.4.4 External Interrupt Pin ($\overline{IRQ}$) . . . 28
1.4.5 Analog Power Supply Pin ($V_{DDA}$) . . . 29
1.4.6 Analog Ground Pin ($V_{SSA}$) . . . 29
1.4.7 External Filter Capacitor Pin (CGMXFC) . . . 29
1.4.8 ADC Analog Power Supply Pin ($V_{DDAREF}$) . . . 29
1.4.9 ADC Analog Ground Pin ($A_{VSS}/V_{REFL}$) . . . 29
1.4.10 ADC Reference High Voltage Pin ($V_{REFH}$) . . . 29
1.4.11 Port A Input/Output (I/O) Pins (PTA7–PTA0) . . . 29
1.4.12 Port B I/O Pins (PTB7/ATD7–PTB0/ATD0) . . . 29
1.4.13 Port C I/O Pins (PTC5–PTC0) . . . 29
1.4.14 Port D I/O Pins (PTD7–PTD0/ATD8) . . . 29
1.4.15 Port E I/O Pins (PTE7/SPSCK–PTE0/TxD) . . . 30
1.4.16 Port F I/O Pins (PTF6–PTF0/TACH2) . . . 30
1.4.17 Port G I/O Pins (PTG2/KBD2–PTG0/KBD0) . . . 30
1.4.18 Port H I/O Pins (PTH1/KBD4–PTH0/KBD3) . . . 30
1.4.19 CAN Transmit Pin (CANTx) . . . 30
1.4.20 CAN Receive Pin (CANRx) . . . 30
1.4.21 BDLC Transmit Pin (BDTxD) . . . 30
1.4.22 BDLC Receive Pin (BDRxD) . . . 30
1.5 Ordering Information . . . 34


## Chapter 2
## Memory Map


2.1 Introduction . . . 35
2.2 I/O Section . . . 39
2.3 Additional Status and Control Registers . . . 44
2.4 Vector Addresses and Priority . . . 46


## Chapter 3
## Random-Access Memory (RAM)


3.1 Introduction . . . 49
3.2 Functional Description . . . 49

## Chapter 4
## FLASH-1 Memory


4.1 Introduction . . . 51
4.2 Functional Description . . . 51
4.3 FLASH-1 Control and Block Protect Registers . . . 52
4.3.1 FLASH-1 Control Register . . . 52
4.3.2 FLASH-1 Block Protect Register . . . 53
4.4 FLASH-1 Block Protection . . . 54
4.5 FLASH-1 Mass Erase Operation . . . 55
4.6 FLASH-1 Page Erase Operation . . . 56
4.7 FLASH-1 Program Operation . . . 56
4.8 Low-Power Modes . . . 59
4.8.1 WAIT Mode . . . 59
4.8.2 STOP Mode . . . 59


## Chapter 5
## FLASH-2 Memory


5.1 Introduction . . . 61
5.2 Functional Description . . . 61
5.3 FLASH-2 Control and Block Protect Registers . . . 62
5.3.1 FLASH-2 Control Register . . . 62
5.3.2 FLASH-2 Block Protect Register . . . 62
5.4 FLASH-2 Block Protection . . . 64
5.5 FLASH-2 Mass Erase Operation . . . 65
5.6 FLASH-2 Page Erase Operation . . . 66
5.7 FLASH-2 Program Operation . . . 66
5.8 Low-Power Modes . . . 69
5.8.1 WAIT Mode . . . 69
5.8.2 STOP Mode . . . 69


## Chapter 6
## EEPROM-1 Memory


6.1 Introduction . . . 71
6.2 Features . . . 71
6.3 EEPROM-1 Register Summary . . . 71
6.4 Functional Description . . . 73
6.4.1 EEPROM-1 Configuration . . . 73
6.4.2 EEPROM-1 Timebase Requirements . . . 73
6.4.3 EEPROM-1 Program/Erase Protection . . . 74
6.4.4 EEPROM-1 Block Protection . . . 74
6.4.5 EEPROM-1 Programming and Erasing . . . 75
6.4.5.1 Program/Erase Using AUTO Bit . . . 75
6.4.5.2 EEPROM-1 Programming . . . 76
6.4.5.3 EEPROM-1 Erasing . . . 77
6.5 EEPROM-1 Register Descriptions . . . 78
6.5.1 EEPROM-1 Control Register . . . 78
6.5.2 EEPROM-1 Array Configuration Register . . . 79

6.5.3 EEPROM-1 Nonvolatile Register . . . 81
6.5.4 EEPROM-1 Timebase Divider Register . . . 81
6.5.5 EEPROM-1 Timebase Divider Nonvolatile Register . . . 82
6.6 Low-Power Modes . . . 83
6.6.1 Wait Mode . . . 83
6.6.2 Stop Mode . . . 83

## Chapter 7
## EEPROM-2 Memory

7.1 Introduction . . . 85
7.2 Features . . . 85
7.3 EEPROM-2 Register Summary . . . 85
7.4 Functional Description . . . 87
7.4.1 EEPROM-2 Configuration . . . 87
7.4.2 EEPROM-2 Timebase Requirements . . . 87
7.4.3 EEPROM-2 Program/Erase Protection . . . 88
7.4.4 EEPROM-2 Block Protection . . . 88
7.4.5 EEPROM-2 Programming and Erasing . . . 89
7.4.5.1 Program/Erase Using AUTO Bit . . . 89
7.4.5.2 EEPROM-2 Programming . . . 90
7.4.5.3 EEPROM-2 Erasing . . . 91
7.5 EEPROM-2 Register Descriptions . . . 92
7.5.1 EEPROM-2 Control Register . . . 92
7.5.2 EEPROM-2 Array Configuration Register . . . 93
7.5.3 EEPROM-2 Nonvolatile Register . . . 95
7.5.4 EEPROM-2 Timebase Divider Register . . . 95
7.5.5 EEPROM-2 Timebase Divider Nonvolatile Register . . . 96
7.6 Low-Power Modes . . . 97
7.6.1 Wait Mode . . . 97
7.6.2 Stop Mode . . . 97

## Chapter 8
## Central Processor Unit (CPU)

8.1 Introduction . . . 99
8.2 Features . . . 99
8.3 CPU Registers . . . 99
8.3.1 Accumulator . . . 100
8.3.2 Index Register . . . 100
8.3.3 Stack Pointer . . . 101
8.3.4 Program Counter . . . 101
8.3.5 Condition Code Register . . . 102
8.4 Arithmetic/Logic Unit (ALU) . . . 103
8.5 Low-Power Modes . . . 103
8.5.1 Wait Mode . . . 103
8.5.2 Stop Mode . . . 103
8.6 CPU During Break Interrupts . . . 103
8.7 Instruction Set Summary . . . 104
8.8 Opcode Map . . . 109

## Chapter 9
## System Integration Module (SIM)


9.1 Introduction . . . 111
9.2 SIM Bus Clock Control and Generation . . . 113
9.2.1 Bus Timing . . . 113
9.2.2 Clock Startup from POR or LVI Reset . . . 114
9.2.3 Clocks in Stop Mode and Wait Mode . . . 114
9.3 Reset and System Initialization . . . 114
9.3.1 External Pin Reset . . . 114
9.3.2 Active Resets from Internal Sources . . . 115
9.3.2.1 Power-On Reset . . . 116
9.3.2.2 Computer Operating Properly (COP) Reset . . . 116
9.3.2.3 Illegal Opcode Reset . . . 116
9.3.2.4 Illegal Address Reset . . . 117
9.3.2.5 Low-Voltage Inhibit (LVI) Reset . . . 117
9.4 SIM Counter . . . 117
9.4.1 SIM Counter During Power-On Reset . . . 117
9.4.2 SIM Counter During Stop Mode Recovery . . . 117
9.4.3 SIM Counter and Reset States . . . 118
9.5 Program Exception Control . . . 118
9.5.1 Interrupts . . . 118
9.5.1.1 Hardware Interrupts . . . 120
9.5.1.2 SWI Instruction . . . 121
9.5.2 Reset . . . 121
9.5.3 Break Interrupts . . . 121
9.5.4 Status Flag Protection in Break Mode . . . 121
9.6 Low-Power Modes . . . 121
9.6.1 Wait Mode . . . 122
9.6.2 Stop Mode . . . 123
9.7 SIM Registers . . . 124
9.7.1 SIM Break Status Register . . . 124
9.7.2 SIM Reset Status Register . . . 124
9.7.3 SIM Break Flag Control Register . . . 125


## Chapter 10
## Clock Generator Module (CGM)


10.1 Introduction . . . 127
10.2 Features . . . 127
10.3 Functional Description . . . 127
10.3.1 Crystal Oscillator Circuit . . . 129
10.3.2 Phase-Locked Loop Circuit (PLL) . . . 129
10.3.2.1 Circuits . . . 130
10.3.2.2 Acquisition and Tracking Modes . . . 130
10.3.2.3 Manual and Automatic PLL Bandwidth Modes . . . 131
10.3.2.4 Programming the PLL . . . 132
10.3.2.5 Special Programming Exceptions . . . 133
10.3.3 Base Clock Selector Circuit . . . 133
10.3.4 CGM External Connections . . . 134

10.4 I/O Signals . . . 135
10.4.1 Crystal Amplifier Input Pin (OSC1) . . . 135
10.4.2 Crystal Amplifier Output Pin (OSC2) . . . 135
10.4.3 External Filter Capacitor Pin (CGMXFC) . . . 135
10.4.4 Analog Power Pin ($V_{DDA}$) . . . 135
10.4.5 Oscillator Enable Signal (SIMOSCEN) . . . 135
10.4.6 Crystal Output Frequency Signal (CGMXCLK) . . . 135
10.4.7 CGM Base Clock Output (CGMOUT) . . . 135
10.4.8 CGM CPU Interrupt (CGMINT) . . . 135
10.5 CGM Registers . . . 136
10.5.1 PLL Control Register . . . 136
10.5.2 PLL Bandwidth Control Register . . . 137
10.5.3 PLL Programming Register . . . 138
10.6 Interrupts . . . 139
10.7 Low-Power Modes . . . 140
10.7.1 Wait Mode . . . 140
10.7.2 Stop Mode . . . 140
10.8 CGM During Break Interrupts . . . 140
10.9 Acquisition/Lock Time Specifications . . . 140
10.9.1 Acquisition/Lock Time Definitions . . . 140
10.9.2 Parametric Influences on Reaction Time . . . 141
10.9.3 Choosing a Filter Capacitor . . . 142
10.9.4 Reaction Time Calculation . . . 142

## Chapter 11
## Configuration Register (CONFIG-1)

11.1 Introduction . . . 145
11.2 Functional Description . . . 145

## Chapter 12
## Configuration Register (CONFIG-2)

12.1 Introduction . . . 147
12.2 Functional Description . . . 147

## Chapter 13
## Break Module (BRK)

13.1 Introduction . . . 149
13.2 Features . . . 149
13.3 Functional Description . . . 149
13.3.1 Flag Protection During Break Interrupts . . . 151
13.3.2 CPU During Break Interrupts . . . 151
13.3.3 TIM During Break Interrupts . . . 151
13.3.4 COP During Break Interrupts . . . 151
13.4 Low-Power Modes . . . 151
13.4.1 Wait Mode . . . 151
13.4.2 Stop Mode . . . 151
13.5 Break Module Registers . . . 151
13.5.1 Break Status and Control Register . . . 152
13.5.2 Break Address Registers . . . 152

## Chapter 14
## Monitor ROM (MON)

14.1 Introduction . . . 153
14.2 Features . . . 153
14.3 Functional Description . . . 153
14.3.1 Entering Monitor Mode . . . 155
14.3.2 Data Format . . . 156
14.3.3 Echoing . . . 156
14.3.4 Break Signal . . . 156
14.3.5 Commands . . . 157
14.3.6 MC68HC908AS60A Baud Rate . . . 159
14.3.7 MC68HC908AZ60A Baud Rate . . . 160
14.3.8 Security . . . 160

## Chapter 15
## Computer Operating Properly (COP)

15.1 Introduction . . . 163
15.2 Functional Description . . . 163
15.3 I/O Signals . . . 163
15.3.1 CGMXCLK . . . 164
15.3.2 STOP Instruction . . . 164
15.3.3 COPCTL Write . . . 164
15.3.4 Power-On Reset . . . 164
15.3.5 Internal Reset . . . 164
15.3.6 Reset Vector Fetch . . . 165
15.3.7 COPD . . . 165
15.3.8 COPL . . . 165
15.4 COP Control Register . . . 165
15.5 Interrupts . . . 165
15.6 Monitor Mode . . . 165
15.7 Low-Power Modes . . . 165
15.7.1 Wait Mode . . . 165
15.7.2 Stop Mode . . . 166
15.8 COP Module During Break Interrupts . . . 166

## Chapter 16
## Low-Voltage Inhibit (LVI)

16.1 Introduction . . . 167
16.2 Features . . . 167
16.3 Functional Description . . . 167
16.3.1 Polled LVI Operation . . . 168
16.3.2 Forced Reset Operation . . . 168
16.3.3 False Reset Protection . . . 168
16.4 LVI Status Register . . . 169
16.5 LVI Interrupts . . . 169
16.6 Low-Power Modes . . . 169
16.6.1 Wait Mode . . . 169
16.6.2 Stop Mode . . . 170

## Chapter 17
## External Interrupt Module (IRQ)


17.1 Introduction . . . 171
17.2 Features . . . 171
17.3 Functional Description . . . 171
17.4 $\overline{\text{IRQ}}$ Pin . . . 174
17.5 IRQ Module During Break Interrupts . . . 174
17.6 IRQ Status and Control Register . . . 175


## Chapter 18
## Serial Communications Interface (SCI)


18.1 Introduction . . . 177
18.2 Features . . . 177
18.3 Pin Name Conventions . . . 177
18.4 Functional Description . . . 178
18.4.1 Data Format . . . 180
18.4.2 Transmitter . . . 180
18.4.2.1 Character Length . . . 180
18.4.2.2 Character Transmission . . . 180
18.4.2.3 Break Characters . . . 182
18.4.2.4 Idle Characters . . . 183
18.4.2.5 Inversion of Transmitted Output . . . 183
18.4.2.6 Transmitter Interrupts . . . 183
18.4.3 Receiver . . . 183
18.4.3.1 Character Length . . . 186
18.4.3.2 Character Reception . . . 186
18.4.3.3 Data Sampling . . . 186
18.4.3.4 Framing Errors . . . 188
18.4.3.5 Baud Rate Tolerance . . . 188
18.4.3.6 Receiver Wakeup . . . 190
18.4.3.7 Receiver Interrupts . . . 190
18.4.3.8 Error Interrupts . . . 190
18.5 Low-Power Modes . . . 191
18.5.1 Wait Mode . . . 191
18.5.2 Stop Mode . . . 191
18.6 SCI During Break Module Interrupts . . . 191
18.7 I/O Signals . . . 192
18.7.1 PTE0/SCTxD (Transmit Data) . . . 192
18.7.2 PTE1/SCRxD (Receive Data) . . . 192
18.8 I/O Registers . . . 192
18.8.1 SCI Control Register 1 . . . 192
18.8.2 SCI Control Register 2 . . . 194
18.8.3 SCI Control Register 3 . . . 196
18.8.4 SCI Status Register 1 . . . 197
18.8.5 SCI Status Register 2 . . . 200
18.8.6 SCI Data Register . . . 201
18.8.7 SCI Baud Rate Register . . . 201

## Chapter 19
## Serial Peripheral Interface (SPI)

19.1 Introduction . . . 205
19.2 Features . . . 205
19.3 Pin Name and Register Name Conventions . . . 205
19.4 Functional Description . . . 206
19.4.1 Master Mode . . . 208
19.4.2 Slave Mode . . . 208
19.5 Transmission Formats . . . 209
19.5.1 Clock Phase and Polarity Controls . . . 209
19.5.2 Transmission Format When CPHA = 0 . . . 210
19.5.3 Transmission Format When CPHA = 1 . . . 211
19.5.4 Transmission Initiation Latency . . . 211
19.6 Error Conditions . . . 213
19.6.1 Overflow Error . . . 213
19.6.2 Mode Fault Error . . . 214
19.7 Interrupts . . . 216
19.8 Queuing Transmission Data . . . 217
19.9 Resetting the SPI . . . 218
19.10 Low-Power Modes . . . 218
19.10.1 Wait Mode . . . 218
19.10.2 Stop Mode . . . 218
19.11 SPI During Break Interrupts . . . 218
19.12 I/O Signals . . . 219
19.12.1 MISO (Master In/Slave Out) . . . 219
19.12.2 MOSI (Master Out/Slave In) . . . 219
19.12.3 SPSCK (Serial Clock) . . . 219
19.12.4 $\overline{SS}$ (Slave Select) . . . 220
19.12.5 $V_{SS}$ (Clock Ground) . . . 221
19.13 I/O Registers . . . 221
19.13.1 SPI Control Register . . . 221
19.13.2 SPI Status and Control Register . . . 222
19.13.3 SPI Data Register . . . 225

## Chapter 20
## Timer Interface Module B (TIMB)

20.1 Introduction . . . 227
20.2 Features . . . 227
20.3 Functional Description . . . 229
20.3.1 TIMB Counter Prescaler . . . 229
20.3.2 Input Capture . . . 229
20.3.3 Output Compare . . . 230
20.3.3.1 Unbuffered Output Compare . . . 230
20.3.3.2 Buffered Output Compare . . . 230
20.3.4 Pulse Width Modulation (PWM) . . . 231
20.3.4.1 Unbuffered PWM Signal Generation . . . 231
20.3.4.2 Buffered PWM Signal Generation . . . 232
20.3.4.3 PWM Initialization . . . 233

20.4 Interrupts . . . . . 234
20.5 Low-Power Modes . . . . . 234
20.5.1 Wait Mode . . . . . 234
20.5.2 Stop Mode . . . . . 234
20.6 TIMB During Break Interrupts . . . . . 234
20.7 I/O Signals . . . . . 235
20.7.1 TIMB Clock Pin (PTD4/ATD12/TBCLK) . . . . . 235
20.7.2 TIMB Channel I/O Pins (PTF5/TBCH1–PTF4/TBCH0) . . . . . 235
20.8 I/O Registers . . . . . 235
20.8.1 TIMB Status and Control Register . . . . . 235
20.8.2 TIMB Counter Registers . . . . . 237
20.8.3 TIMB Counter Modulo Registers . . . . . 238
20.8.4 TIMB Channel Status and Control Registers . . . . . 238
20.8.5 TIMB Channel Registers . . . . . 241

## Chapter 21
## Programmable Interrupt Timer (PIT)

21.1 Introduction . . . . . 243
21.2 Features . . . . . 243
21.3 Functional Description . . . . . 243
21.4 PIT Counter Prescaler . . . . . 244
21.5 Low-Power Modes . . . . . 244
21.5.1 Wait Mode . . . . . 244
21.5.2 Stop Mode . . . . . 245
21.6 PIT During Break Interrupts . . . . . 245
21.7 I/O Registers . . . . . 245
21.7.1 PIT Status and Control Register . . . . . 245
21.7.2 PIT Counter Registers . . . . . 247
21.7.3 PIT Counter Modulo Registers . . . . . 248

## Chapter 22
## Input/Output Ports

22.1 Introduction . . . . . 249
22.2 Port A . . . . . 250
22.2.1 Port A Data Register . . . . . 250
22.2.2 Data Direction Register A . . . . . 250
22.3 Port B . . . . . 252
22.3.1 Port B Data Register . . . . . 252
22.3.2 Data Direction Register B . . . . . 253
22.4 Port C . . . . . 254
22.4.1 Port C Data Register . . . . . 254
22.4.2 Data Direction Register C . . . . . 254
22.5 Port D . . . . . 256
22.5.1 Port D Data Register . . . . . 256
22.5.2 Data Direction Register D . . . . . 257
22.6 Port E . . . . . 258
22.6.1 Port E Data Register . . . . . 258
22.6.2 Data Direction Register E . . . . . 259

22.7 Port F . . . . . 260
22.7.1 Port F Data Register . . . . . 261
22.7.2 Data Direction Register F . . . . . 261
22.8 Port G . . . . . 263
22.8.1 Port G Data Register . . . . . 263
22.8.2 Data Direction Register G . . . . . 263
22.9 Port H . . . . . 265
22.9.1 Port H Data Register . . . . . 265
22.9.2 Data Direction Register H . . . . . 265

## Chapter 23
## MSCAN Controller (MSCAN08)

23.1 Introduction . . . . . 267
23.2 Features . . . . . 267
23.3 External Pins . . . . . 268
23.4 Message Storage . . . . . 268
23.4.1 Background . . . . . 268
23.4.2 Receive Structures . . . . . 269
23.4.3 Transmit Structures . . . . . 270
23.5 Identifier Acceptance Filter . . . . . 271
23.6 Interrupts . . . . . 274
23.6.1 Interrupt Acknowledge . . . . . 274
23.6.2 Interrupt Vectors . . . . . 275
23.7 Protocol Violation Protection . . . . . 275
23.8 Low Power Modes . . . . . 275
23.8.1 MSCAN08 Sleep Mode . . . . . 276
23.8.2 MSCAN08 Soft Reset Mode . . . . . 277
23.8.3 MSCAN08 Power Down Mode . . . . . 277
23.8.4 CPU Wait Mode . . . . . 278
23.8.5 Programmable Wakeup Function . . . . . 278
23.9 Timer Link . . . . . 278
23.10 Clock System . . . . . 278
23.11 Memory Map . . . . . 281
23.12 Programmer's Model of Message Storage . . . . . 282
23.12.1 Message Buffer Outline . . . . . 282
23.12.2 Identifier Registers . . . . . 282
23.12.3 Data Length Register (DLR) . . . . . 284
23.12.4 Data Segment Registers (DSRn) . . . . . 285
23.12.5 Transmit Buffer Priority Registers . . . . . 285
23.13 Programmer's Model of Control Registers . . . . . 286
23.13.1 MSCAN08 Module Control Register 0 . . . . . 287
23.13.2 MSCAN08 Module Control Register 1 . . . . . 288
23.13.3 MSCAN08 Bus Timing Register 0 . . . . . 289
23.13.4 MSCAN08 Bus Timing Register 1 . . . . . 290
23.13.5 MSCAN08 Receiver Flag Register (CRFLG) . . . . . 291
23.13.6 MSCAN08 Receiver Interrupt Enable Register . . . . . 293
23.13.7 MSCAN08 Transmitter Flag Register . . . . . 294

23.13.8 MSCAN08 Transmitter Control Register . . . . . 295
23.13.9 MSCAN08 Identifier Acceptance Control Register. . . . . 296
23.13.10 MSCAN08 Receive Error Counter . . . . . 297
23.13.11 MSCAN08 Transmit Error Counter . . . . . 297
23.13.12 MSCAN08 Identifier Acceptance Registers . . . . . 298
23.13.13 MSCAN08 Identifier Mask Registers (CIDMR0-3) . . . . . 299

## Chapter 24
## Keyboard Module (KBI)

24.1 Introduction . . . . . 301
24.2 Features. . . . . 301
24.3 Functional Description . . . . . 301
24.4 Keyboard Initialization . . . . . 303
24.5 Low-Power Modes . . . . . 304
24.5.1 Wait Mode . . . . . 304
24.5.2 Stop Mode . . . . . 304
24.6 Keyboard Module During Break Interrupts. . . . . 304
24.7 I/O Registers . . . . . 304
24.7.1 Keyboard Status and Control Register. . . . . 305
24.7.2 Keyboard Interrupt Enable Register. . . . . 306

## Chapter 25
## Timer Interface Module A (TIMA)

25.1 Introduction . . . . . 307
25.2 Features. . . . . 307
25.3 Functional Description . . . . . 309
25.3.1 TIMA Counter Prescaler. . . . . 310
25.3.2 Input Capture . . . . . 310
25.3.3 Output Compare. . . . . 310
25.3.3.1 Unbuffered Output Compare . . . . . 311
25.3.3.2 Buffered Output Compare . . . . . 311
25.3.4 Pulse Width Modulation (PWM) . . . . . 312
25.3.4.1 Unbuffered PWM Signal Generation . . . . . 313
25.3.4.2 Buffered PWM Signal Generation . . . . . 313
25.3.4.3 PWM Initialization . . . . . 314
25.4 Interrupts . . . . . 315
25.5 Low-Power Modes . . . . . 315
25.5.1 Wait Mode . . . . . 315
25.5.2 Stop Mode . . . . . 315
25.6 TIMA During Break Interrupts . . . . . 316
25.7 I/O Signals . . . . . 316
25.7.1 TIMA Clock Pin (PTD6/ATD14/TACLK) . . . . . 316
25.7.2 TIMA Channel I/O Pins (PTF3–PTF0/TACH2 and PTE3/TACH1–PTE2/TACH0) . . . . . 316
25.8 I/O Registers . . . . . 317
25.8.1 TIMA Status and Control Register . . . . . 317
25.8.2 TIMA Counter Registers. . . . . 318
25.8.3 TIMA Counter Modulo Registers . . . . . 319
25.8.4 TIMA Channel Status and Control Registers . . . . . 320
25.8.5 TIMA Channel Registers . . . . . 323

## Chapter 26
## Analog-to-Digital Converter (ADC)


26.1 Introduction . . . 327
26.2 Features . . . 327
26.3 Functional Description . . . 327
26.3.1 ADC Port I/O Pins . . . 328
26.3.2 Voltage Conversion . . . 329
26.3.3 Conversion Time . . . 329
26.3.4 Continuous Conversion . . . 329
26.3.5 Accuracy and Precision . . . 329
26.4 Interrupts . . . 329
26.5 Low-Power Modes . . . 330
26.5.1 Wait Mode . . . 330
26.5.2 Stop Mode . . . 330
26.6 I/O Signals . . . 330
26.6.1 ADC Analog Power Pin ($V_{DDAREF}$)/ADC Voltage Reference Pin ($V_{REFH}$) . . . 330
26.6.2 ADC Analog Ground Pin ($V_{SSA}$)/ADC Voltage Reference Low Pin ($V_{REFL}$) . . . 330
26.6.3 ADC Voltage In (ADCVIN) . . . 330
26.7 I/O Registers . . . 330
26.7.1 ADC Status and Control Register . . . 331
26.7.2 ADC Data Register . . . 332
26.7.3 ADC Input Clock Register . . . 333


## Chapter 27
## Byte Data Link Controller (BDLC)


27.1 Introduction . . . 335
27.2 Features . . . 335
27.3 Functional Description . . . 335
27.3.1 BDLC Operating Modes . . . 337
27.3.1.1 Power Off Mode . . . 337
27.3.1.2 Reset Mode . . . 337
27.3.1.3 Run Mode . . . 338
27.3.1.4 BDLC Wait Mode . . . 338
27.3.1.5 BDLC Stop Mode . . . 338
27.3.1.6 Digital Loopback Mode . . . 338
27.3.1.7 Analog Loopback Mode . . . 338
27.4 BDLC MUX Interface . . . 339
27.4.1 Rx Digital Filter . . . 339
27.4.1.1 Operation . . . 339
27.4.1.2 Performance . . . 340
27.4.2 J1850 Frame Format . . . 340
27.4.3 J1850 VPW Symbols . . . 343
27.4.4 J1850 VPW Valid/Invalid Bits and Symbols . . . 345
27.4.5 Message Arbitration . . . 348
27.5 BDLC Protocol Handler . . . 349
27.5.1 Protocol Architecture . . . 350
27.5.2 Rx and Tx Shift Registers . . . 350
27.5.3 Rx and Tx Shadow Registers . . . 350

27.5.4 Digital Loopback Multiplexer . . . 351
27.5.5 State Machine . . . 351
27.5.5.1 4X Mode . . . 351
27.5.5.2 Receiving a Message in Block Mode . . . 351
27.5.5.3 Transmitting a Message in Block Mode . . . 351
27.5.5.4 J1850 Bus Errors . . . 351
27.5.5.5 Summary . . . 353
27.6 BDLC CPU Interface . . . 353
27.6.1 BDLC Analog and Roundtrip Delay Register . . . 354
27.6.2 BDLC Control Register 1 . . . 355
27.6.3 BDLC Control Register 2 . . . 356
27.6.4 BDLC State Vector Register . . . 361
27.6.5 BDLC Data Register . . . 362
27.7 Low-Power Modes . . . 363
27.7.1 Wait Mode . . . 363
27.7.2 Stop Mode . . . 363

## Chapter 28
## Electrical Specifications

28.1 Electrical Specifications . . . 365
28.1.1 Maximum Ratings . . . 365
28.1.2 Functional Operating Range . . . 366
28.1.3 Thermal Characteristics . . . 366
28.1.4 5.0 Volt DC Electrical Characteristics . . . 367
28.1.5 Control Timing . . . 368
28.1.6 ADC Characteristics . . . 368
28.1.7 5.0 Vdc ± 0.5 V Serial Peripheral Interface (SPI) Timing . . . 369
28.1.8 CGM Operating Conditions . . . 372
28.1.9 CGM Component Information . . . 372
28.1.10 CGM Acquisition/Lock Time Information . . . 373
28.1.11 Timer Module Characteristics . . . 374
28.1.12 RAM Memory Characteristics . . . 374
28.1.13 EEPROM Memory Characteristics . . . 374
28.1.14 FLASH Memory Characteristics . . . 375
28.1.15 BDLC Transmitter VPW Symbol Timings . . . 376
28.1.16 BDLC Receiver VPW Symbol Timings . . . 376
28.1.17 BDLC Transmitter DC Electrical Characteristics . . . 377
28.1.18 BDLC Receiver DC Electrical Characteristics . . . 377
28.2 Mechanical Specifications . . . 378
28.2.1 51-Pin Plastic Leaded Chip Carrier (PLCC) . . . 378
28.2.2 64-Pin Quad Flat Pack (QFP) . . . 379

## Appendix A
## MC68HC908AS60 and MC68HC908AZ60

A.1 Changes from the MC68HC908AS60 and MC68HC908AZ60 (non-A suffix devices) . . . 381
A.1.1 Specification . . . 381
A.1.2 FLASH . . . 381
A.1.2.1 FLASH Architecture . . . 381
A.1.2.2 FLASH Control Registers . . . 381
A.1.2.3 FLASH Programming Procedure . . . 381
A.1.2.4 FLASH Programming Time . . . 381
A.1.2.5 FLASH Block Protection . . . 382
A.1.2.6 FLASH Endurance . . . 382
A.1.3 EEPROM . . . 382
A.1.3.1 EEPROM Architecture . . . 382
A.1.3.2 EEPROM Clock Source and Prescaler . . . 382
A.1.3.3 EEPROM AUTO Programming & Erasing . . . 382
A.1.4 CONFIG-2 . . . 383
A.1.5 Keyboard Interrupt . . . 383
A.1.6 Current Consumption . . . 383
A.1.7 Illegal Address Reset . . . 383
A.1.8 Monitor Mode Entry and COP Disable Voltage . . . 383
A.1.9 Low-Voltage Inhibit (LVI) . . . 383

## Appendix B
## MC68HC908AZ60E

B.1 Introduction . . . 385
B.2 Detailed Memory Map Changes (MC68HC908AS60A references have been removed) . . . 387
B.3 I/O Section . . . 390
B.4 Additional Status and Control Registers . . . 394
B.5 Vector Addresses and Priority . . . 396
B.6 Ordering Information . . . 397
B.6.1 MC Order Numbers . . . 397
B.7 Configuration Register (CONFIG-3) . . . 398
B.8 SCI . . . 398
B.9 MSCAN . . . 398
B.10 ADC . . . 399

## Revision History

Revision History . . . 401

## Glossary

Glossary . . . 405

# Chapter 1
# General Description

## 1.1 Introduction

The MC68HC908AS60A, MC68HC908AZ60A, and MC68HC908AZ60E are members 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.

These parts are designed to emulate the MC68HC08ASxx and MC68HC08AZxx automotive families and may offer extra features which are not available on those devices. It is the user's responsibility to ensure compatibility between the features used on the MC68HC908AS60A, MC68HC908AZ60A, and MC68HC908AZ60E and those which are available on the device which will ultimately be used in the application.

For detailed information regarding the MC68HC908AZ60E refer to Appendix B MC68HC908AZ60E.

## 1.2 Features

Features of the MC68HC908AS60A and MC68HC908AZ60A include:

- High-Performance M68HC08 Architecture
- Fully Upward-Compatible Object Code with M6805, M146805, and M68HC05 Families
- 8.4 MHz Internal Bus Frequency
- 60 Kbytes of FLASH Electrically Erasable Read-Only Memory (FLASH)
- FLASH Data Security
- 1 Kbyte of On-Chip Electrically Erasable Programmable Read-Only Memory with Security Option (EEPROM)
- 2 Kbyte of On-Chip RAM
- Clock Generator Module (CGM)
- Serial Peripheral Interface Module (SPI)
- Serial Communications Interface Module (SCI)
- 8-Bit, 15-Channel Analog-to-Digital Converter (ADC-15)
- 16-Bit, 6-Channel Timer Interface Module (TIMA-6)
- Programmable Interrupt Timer (PIT)
- System Protection Features
  - Computer Operating Properly (COP) with Optional Reset
  - Low-Voltage Detection with Optional Reset
  - Illegal Opcode Detection with Optional Reset
  - Illegal Address Detection with Optional Reset
- Low-Power Design (Fully Static with Stop and Wait Modes)

- Master Reset Pin and Power-On Reset
- 16-Bit, 2-Channel Timer Interface Module (TIMB) (AZ only)
- 5-Bit Keyboard Interrupt Module (64-Pin QFP only)
- MSCAN Controller Implements CAN 2.0b Protocol as Defined in BOSCH Specification September 1991 (AZ only)
- SAE J1850 Byte Data Link Controller Digital Module (AS only)

Features of the CPU08 include:

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

## 1.3 MCU Block Diagram

Figure 1-1 shows the structure of the MC68HC908AZ60A.

Figure 1-2 shows the structure of the MC68HC908AS60A.

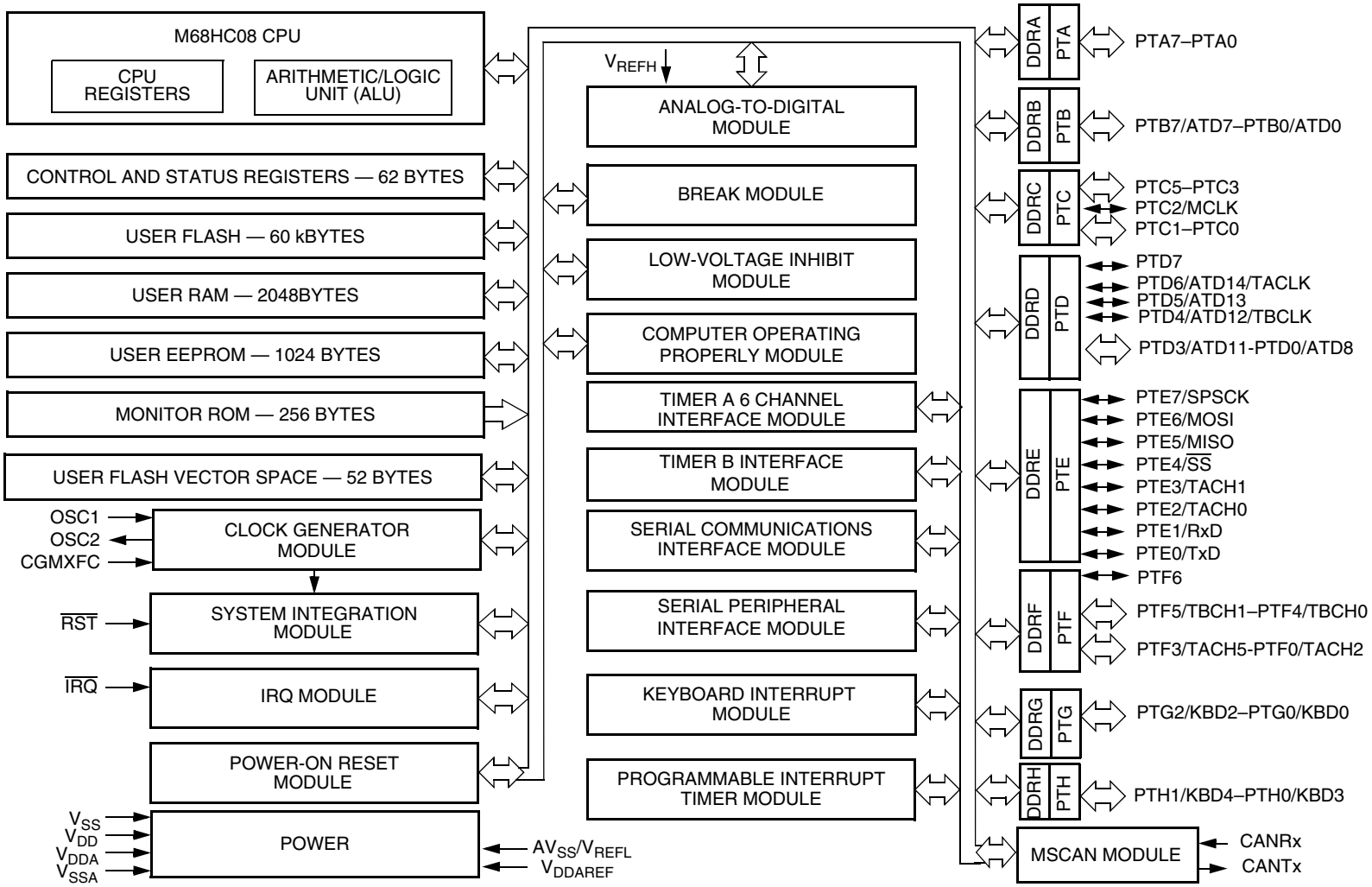



**Figure 1-1. MCU Block Diagram for the MC68HC908AZ60A (64-Pin QFP)**

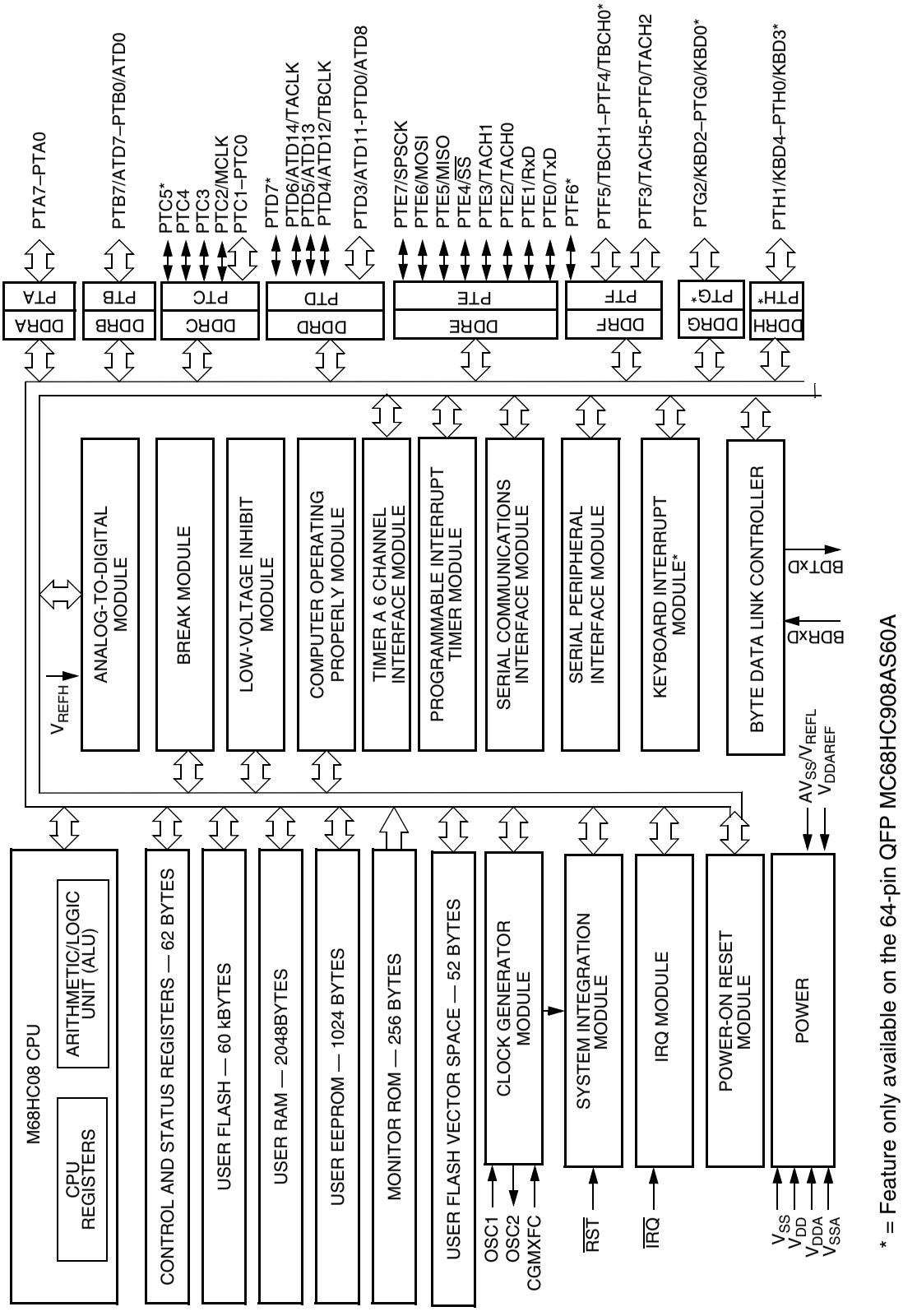



**Figure 1-2. MCU Block Diagram for the MC68HC908AS60A (64-Pin QFP and 52-Pin PLCC)**

* = Feature only available on the 64-pin QFP MC68HC908AS60A

## 1.4 Pin Assignments

Figure 1-3 shows the MC68HC908AZ60A pin assignments.

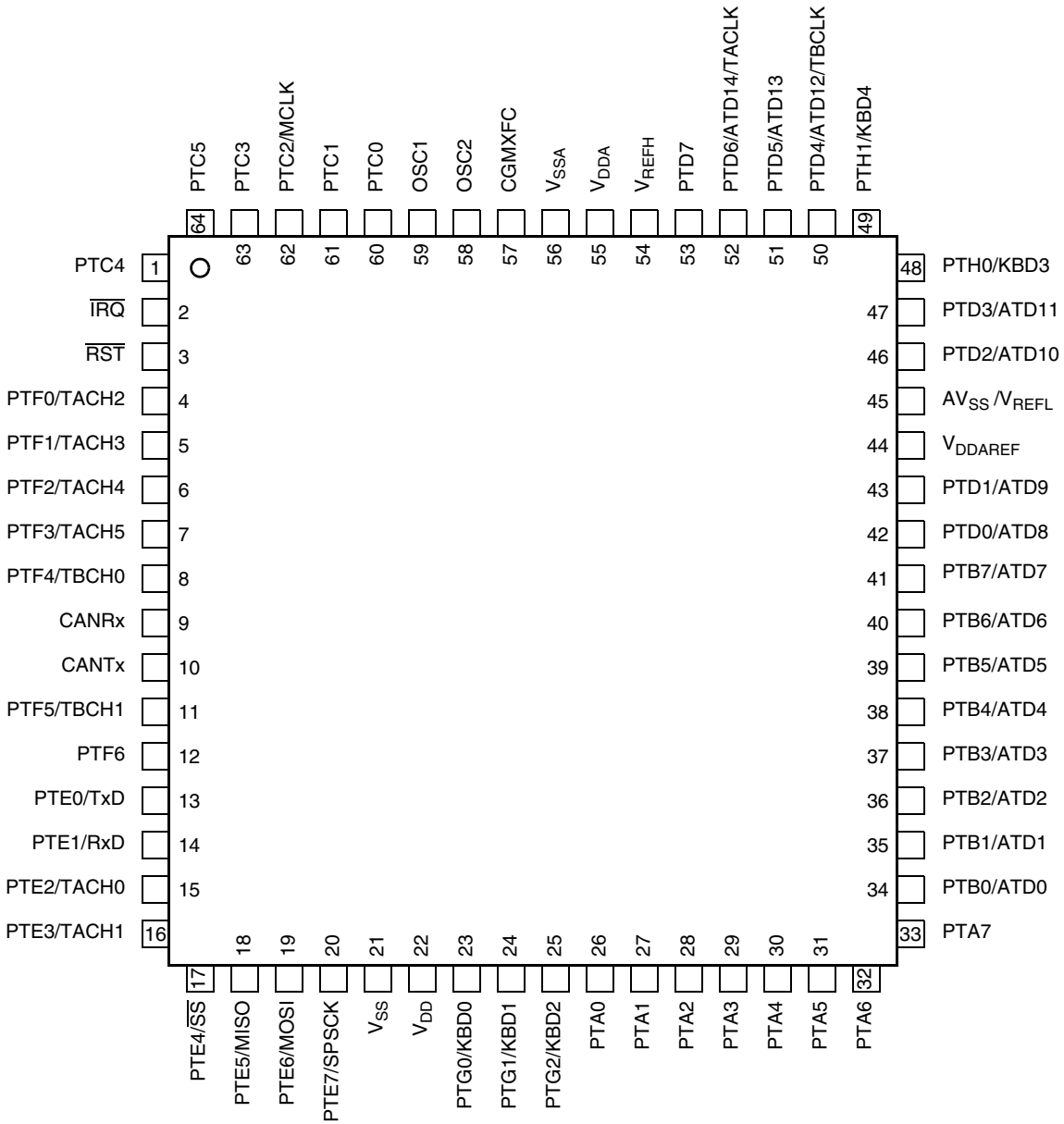



**Figure 1-3. MC68HC908AZ60A (64-Pin QFP)**

Figure 1-4 shows the MC68HC908AS60A 64-pin QFP pin assignments.

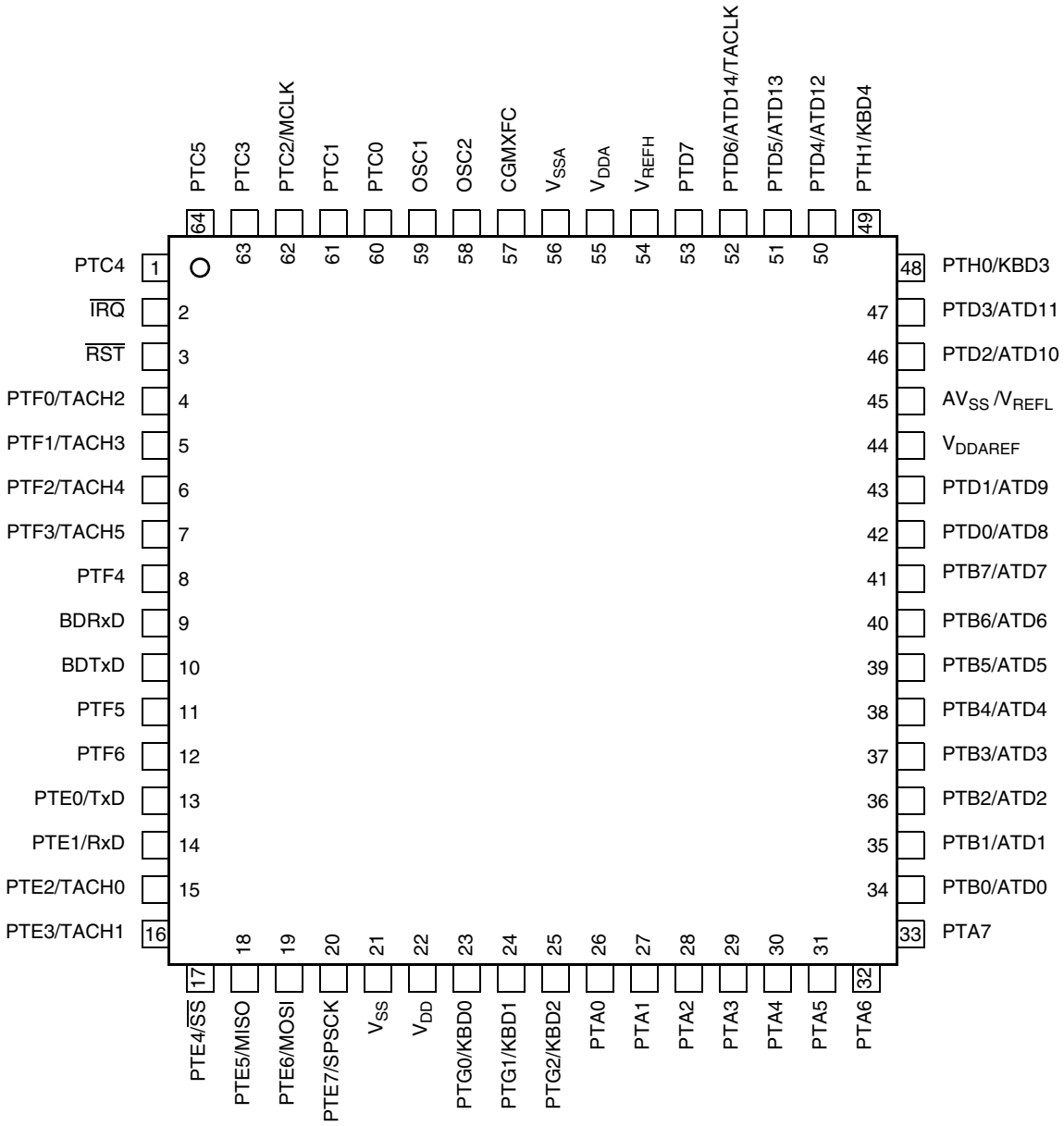



**Figure 1-4. MC68HC908AS60A (64-Pin QFP)**

Figure 1-5 shows MC68HC908AS60A 52-pin PLCC pin assignments.

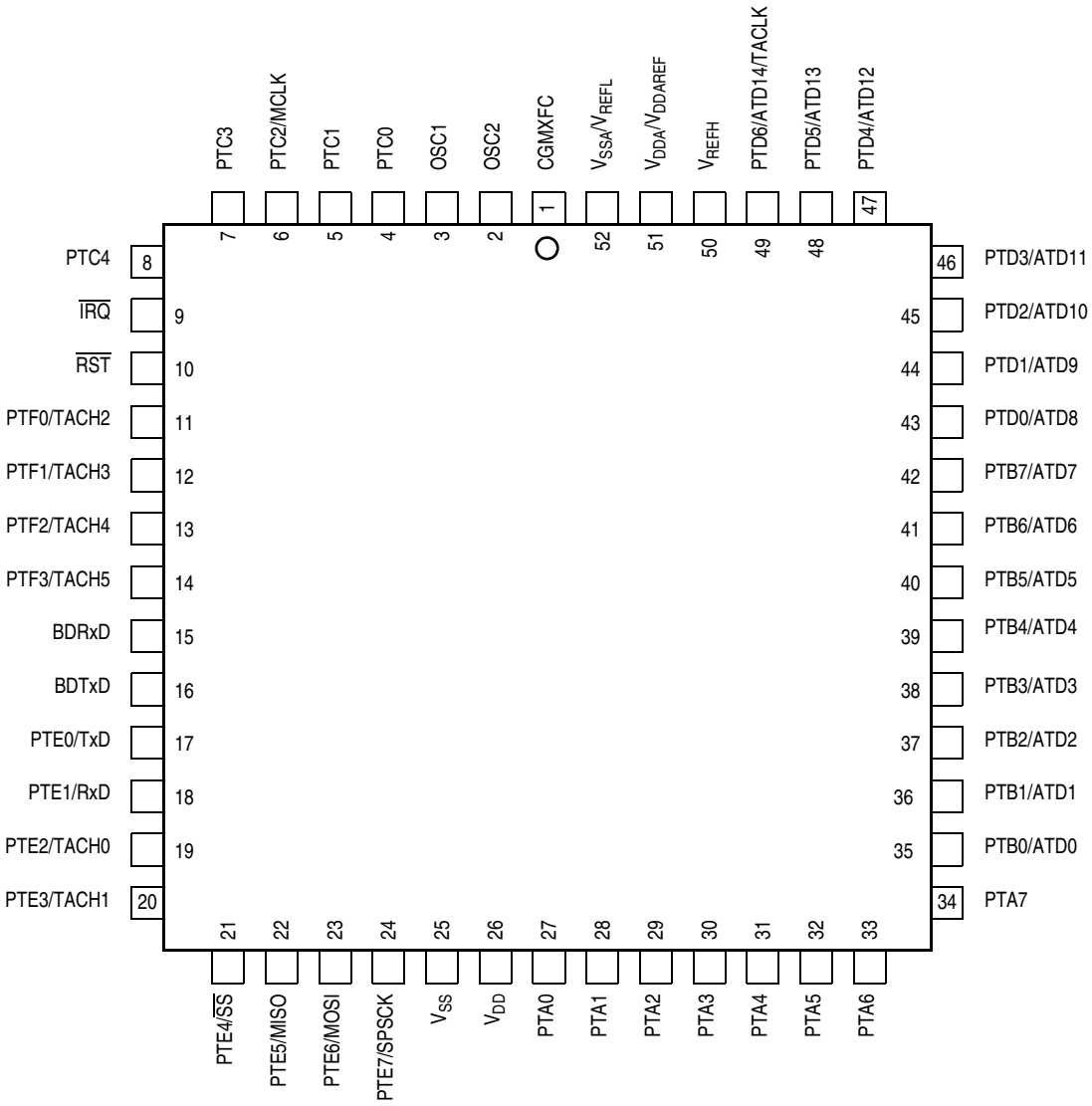



**Figure 1-5. MC68HC908AS60A (52-Pin PLCC)**

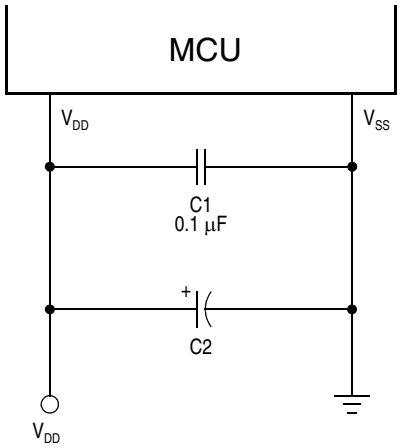
***NOTE***

*The following pin descriptions are just a quick reference. For a more detailed representation, see Chapter 22 Input/Output Ports.*

### 1.4.1 Power Supply Pins ($V_{DD}$ and $V_{SS}$)

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

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as shown in Figure 1-6. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.




NOTE: Component values shown represent typical applications.

**Figure 1-6. Power Supply Bypassing**

$V_{SS}$ is also the ground for the port output buffers and the ground return for the serial clock in the Serial Peripheral Interface module (SPI). See Chapter 19 Serial Peripheral Interface (SPI).

***NOTE***

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

### 1.4.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. See Chapter 10 Clock Generator Module (CGM).

### 1.4.3 External Reset Pin ($\overline{RST}$)

A 0 on the $\overline{RST}$ pin forces the MCU to a known startup state. $\overline{RST}$ is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. See Chapter 9 System Integration Module (SIM) for more information.

### 1.4.4 External Interrupt Pin ($\overline{IRQ}$)

$\overline{IRQ}$ is an asynchronous external interrupt pin. See Chapter 17 External Interrupt Module (IRQ).

### 1.4.5 Analog Power Supply Pin ($V_{DDA}$)

$V_{DDA}$ is the power supply pin for the analog portion of the Clock Generator Module (CGM). See Chapter 10 Clock Generator Module (CGM).

### 1.4.6 Analog Ground Pin ($V_{SSA}$)

$V_{SSA}$ is the ground connection for the analog portion of the Clock Generator Module (CGM). See Chapter 10 Clock Generator Module (CGM).

### 1.4.7 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the Clock Generator Module (CGM). See Chapter 10 Clock Generator Module (CGM).

### 1.4.8 ADC Analog Power Supply Pin ($V_{DDAREF}$)

$V_{DDAREF}$ is the power supply pin for the analog portion of the Analog-to-Digital Converter (ADC). See Chapter 26 Analog-to-Digital Converter (ADC).

### 1.4.9 ADC Analog Ground Pin ($AV_{SS}/V_{REFL}$)

The $AV_{SS}/V_{REFL}$ pin provides both the analog ground connection and the reference low voltage for the Analog-to-Digital Converter (ADC). See Chapter 26 Analog-to-Digital Converter (ADC).

### 1.4.10 ADC Reference High Voltage Pin ($V_{REFH}$)

$V_{REFH}$ provides the reference high voltage for the Analog-to-Digital Converter (ADC). See Chapter 26 Analog-to-Digital Converter (ADC).

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

PTA7–PTA0 are general-purpose bidirectional I/O port pins. See Chapter 22 Input/Output Ports.

### 1.4.12 Port B I/O Pins (PTB7/ATD7–PTB0/ATD0)

Port B is an 8-bit special function port that shares all eight pins with the Analog-to-Digital Converter (ADC). See Chapter 26 Analog-to-Digital Converter (ADC) and Chapter 22 Input/Output Ports.

### 1.4.13 Port C I/O Pins (PTC5–PTC0)

PTC5–PTC3 and PTC1–PTC0 are general-purpose bidirectional I/O port pins. PTC2/MCLK is a special function port that shares its pin with the system clock which has a frequency equivalent to the system clock. See Chapter 22 Input/Output Ports.

### 1.4.14 Port D I/O Pins (PTD7–PTD0/ATD8)

Port D is an 8-bit special-function port that shares seven of its pins with the Analog-to-Digital Converter module (ADC-15), one of its pins with the Timer Interface Module A (TIMA), and one more of its pins with the Timer Interface Module B (TIMB). See Chapter 25 Timer Interface Module A (TIMA), Chapter 20 Timer Interface Module B (TIMB), Chapter 26 Analog-to-Digital Converter (ADC) and Chapter 22 Input/Output Ports.

### 1.4.15 Port E I/O Pins (PTE7/SPSCK–PTE0/TxD)

Port E is an 8-bit special function port that shares two of its pins with the Timer Interface Module A (TIMA), four of its pins with the Serial Peripheral Interface module (SPI), and two of its pins with the Serial Communication Interface module (SCI). See Chapter 18 Serial Communications Interface (SCI), Chapter 19 Serial Peripheral Interface (SPI), Chapter 25 Timer Interface Module A (TIMA), and Chapter 22 Input/Output Ports.

### 1.4.16 Port F I/O Pins (PTF6–PTF0/TACH2)

Port F is a 7-bit special function port that shares its pins with the Timer Interface Module B (TIMB). Six of its pins are shared with the Timer Interface Module A (TIMA-6). See Chapter 25 Timer Interface Module A (TIMA), Chapter 20 Timer Interface Module B (TIMB), and Chapter 22 Input/Output Ports.

### 1.4.17 Port G I/O Pins (PTG2/KBD2–PTG0/KBD0)

Port G is a 3-bit special function port that shares all of its pins with the Keyboard Module (KBD). See Chapter 24 Keyboard Module (KBI) and Chapter 22 Input/Output Ports.

### 1.4.18 Port H I/O Pins (PTH1/KBD4–PTH0/KBD3)

Port H is a 2-bit special-function port that shares all of its pins with the Keyboard Module (KBD). See Chapter 24 Keyboard Module (KBI) and Chapter 22 Input/Output Ports.

### 1.4.19 CAN Transmit Pin (CANTx)

This pin is the digital output from the CAN module (CANTx). See Chapter 23 MSCAN Controller (MSCAN08).

### 1.4.20 CAN Receive Pin (CANRx)

This pin is the digital input to the CAN module (CANRx). See Chapter 23 MSCAN Controller (MSCAN08).

### 1.4.21 BDLC Transmit Pin (BDTxD)

This pin is the digital output from the BDLC module (BDTxD). See Chapter 27 Byte Data Link Controller (BDLC).

### 1.4.22 BDLC Receive Pin (BDRxD)

This pin is the digital input to the CAN module (BDRxD). See Chapter 27 Byte Data Link Controller (BDLC).

**Table 1-1. External Pins Summary**

| Pin Name | Function | Driver Type | Hysteresis[1] | Reset State |
|---|---|---|---|---|
| PTA7–PTA0 | General-Purpose I/O | Dual State | No | Input Hi-Z |
| PTB7/ATD7–PTB0/ATD0 | General-Purpose I/O<br>ADC Channel | Dual State | No | Input Hi-Z |
| PTC5–PTC0 | General-Purpose I/O | Dual State | No | Input Hi-Z |
| PTD7 | General Purpose I/O | Dual State | No | Input Hi-Z |
| PTD6/ATD14/TACLK<br>ADC Channel | General-Purpose I/O<br>ADC Channel/<br>Timer External Input Clock | Dual State | No | Input Hi-Z |
| PTD5/ATD13 ADC Channel | General-Purpose I/O<br>ADC Channel | Dual State | No | Input Hi-Z |
| PTD4/ATD12/TBCLK<br>ADC Channel | General-Purpose I/O<br>ADC Channel/<br>Timer External Input Clock | Dual State | No | Input Hi-Z |
| PTD3/ATD11–PTD0/ATD8<br>ADC Channels | General-Purpose I/O<br>ADC Channel | Dual State | No | Input Hi-Z |
| PTE7/SPSCK | General-Purpose I/O<br>SPI Clock | Dual State<br>Open Drain | Yes | Input Hi-Z |
| PTE6/MOSI | General-Purpose I/O<br>SPI Data Path | Dual State<br>Open Drain | Yes | Input Hi-Z |
| PTE5/MISO | General-Purpose I/O<br>SPI Data Path | Dual State<br>Open Drain | Yes | Input Hi-Z |
| PTE4/$\overline{SS}$ | General-Purpose I/O<br>SPI Slave Select | Dual State | Yes | Input Hi-Z |
| PTE3/TACH1 | General-Purpose I/O<br>Timer A Channel 1 | Dual State | Yes | Input Hi-Z |
| PTE2/TACH0 | General-Purpose I/O<br>Timer A Channel 0 | Dual State | Yes | Input Hi-Z |
| PTE1/RxD | General-Purpose I/O<br>SCI Receive Data | Dual State | Yes | Input Hi-Z |
| PTE0/TxD | General-Purpose I/O<br>SCI Transmit Data | Dual State | No | Input Hi-Z |
| PTF6 | General-Purpose I/O | Dual State | No | Input Hi-Z |
| PTF5/TBCH1–PTF4/TBCH0 | General-Purpose I/O/<br>Timer B Channel | Dual State | Yes | Input Hi-Z |
| PTF3/TACH5 | General-Purpose I/O<br>Timer A Channel 5 | Dual State | Yes | Input Hi-Z |
| PTF2/TACH4 | General-Purpose I/O<br>Timer A Channel 4 | Dual State | Yes | Input Hi-Z |
| PTF1/TACH3 | General-Purpose I/O<br>Timer A Channel 3 | Dual State | Yes | Input Hi-Z |

— Continued on next page

**Table 1-1. External Pins Summary (Continued)**

| Pin Name | Function | Driver Type | Hysteresis[1] | Reset State |
|---|---|---|---|---|
| PTF0/TACH2 | General-Purpose I/O<br>Timer A Channel 2 | Dual State | Yes | Input Hi-Z |
| PTG2/KBD2–PTG0/KBD0 | General-Purpose I/O/<br>Keyboard Wakeup Pin | Dual State | Yes | Input Hi-Z |
| PTH1/KBD4 –PTH0/KBD3 | General-Purpose I/O/<br>Keyboard Wakeup Pin | Dual State | Yes | Input Hi-Z |
| $V_{DD}$ | Chip Power Supply | N/A | N/A | N/A |
| $V_{SS}$ | Chip Ground | N/A | N/A | N/A |
| $V_{DDA}$ | CGM Analog Power Supply | | | |
| $V_{SSA}$ | CGM Analog Ground | | | |
| $V_{DDAREF}$ | ADC Power Supply | N/A | N/A | N/A |
| $A_{VSS}/V_{REFL}$ | ADC Ground/<br>ADC Reference Low Voltage | N/A | N/A | N/A |
| $V_{REFH}$ | A/D Reference High Voltage | N/A | N/A | N/A |
| OSC1 | External Clock In | N/A | No | Input Hi-Z |
| OSC2 | External Clock Out | N/A | N/A | Output |
| CGMXFC | PLL Loop Filter Cap | N/A | N/A | N/A |
| $\overline{IRQ}$ | External Interrupt Request | N/A | N/A | Input Hi-Z |
| $\overline{RST}$ | Reset | N/A | N/A | Output Low |
| CANRx | CAN Serial Input | N/A | Yes | Input Hi-Z |
| CANTx | CAN Serial Output | Output | No | Output |
| BDRxD | BDLC Serial Input | N/A | Yes | Input Hi-Z |
| BDTxD | BDLC Serial Output | Output | No | Output |

1. Hysteresis is not 100% tested but is typically a minimum of 300 mV.

**Table 1-2. Clock Signal Naming Conventions**

| Clock Signal Name | Description |
|---|---|
| CGMXCLK | Buffered version of OSC1 from Clock Generation Module (CGM) |
| CGMOUT | PLL-based or OSC1-based clock output from Clock Generator Module (CGM) |
| Bus Clock | CGMOUT divided by two |
| SPSCK | SPI serial clock |
| TACLK | External clock input for TIMA |
| TBCLK | External clock input for TIMB |

**Table 1-3. Clock Source Summary**

| Module | Clock Source |
|---|---|
| ADC | CGMXCLK or Bus Clock |
| CAN | CGMXCLK or CGMOUT |
| COP | CGMXCLK |
| CPU | Bus Clock |
| FLASH | Bus Clock |
| EEPROM | CGMXCLK or Bus Clock |
| RAM | Bus Clock |
| SPI | Bus Clock/SPSCK |
| SCI | CGMXCLK |
| TIMA | Bus Clock or PTD6/ATD14/TACLK |
| TIMB | Bus Clock or PTD4/TBCLK |
| PIT | Bus Clock |
| SIM | CGMOUT and CGMXCLK |
| IRQ | Bus Clock |
| BRK | Bus Clock |
| LVI | Bus Clock and CGMXCLK |
| CGM | OSC1 and OSC2 |

## 1.5 Ordering Information

This subsection contains instructions for ordering the MC68HC908AZ60A / MC68HC908AS60A.

**Table 1-4. MC Order Numbers**

| MC Order Number | Operating Temperature Range |
|---|---|
| MC68HC908AS60ACFU (64-Pin QFP) | –40°C to + 85°C |
| MC68HC908AS60AVFU (64-Pin QFP) | –40°C to + 105°C |
| MC68HC908AS60AMFU (64-Pin QFP) | –40°C to + 125°C |
| MC68HC908AS60ACFN (52-Pin PLCC) | –40°C to + 85°C |
| MC68HC908AS60AVFN (52-Pin PLCC) | –40°C to + 105°C |
| MC68HC908AS60AMFN (52-Pin PLCC) | –40°C to + 125°C |
| MC68HC908AZ60ACFU (64-Pin QFP) | –40°C to + 85°C |
| MC68HC908AZ60AVFU (64-Pin QFP) | –40°C to + 105°C |
| MC68HC908AZ60AMFU (64-Pin QFP) | –40°C to + 125°C |

# Chapter 2
# Memory Map

## 2.1 Introduction

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

- 60K Bytes of FLASH EEPROM
- 2048 Bytes of RAM
- 1024 Bytes of EEPROM with Protect Option
- 52 Bytes of User-Defined Vectors
- 256 Bytes of Monitor ROM

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

- **Reserved** — Accessing a reserved location can have unpredictable effects on MCU operation.
- **Unused** — These locations are reserved in the memory map for future use**,** accessing an unused location can have unpredictable effects on MCU operation.
- **Unimplemented** — Accessing an unimplemented location can cause an illegal address reset (within the constraints as outlined in the Chapter 9 System Integration Module (SIM)).

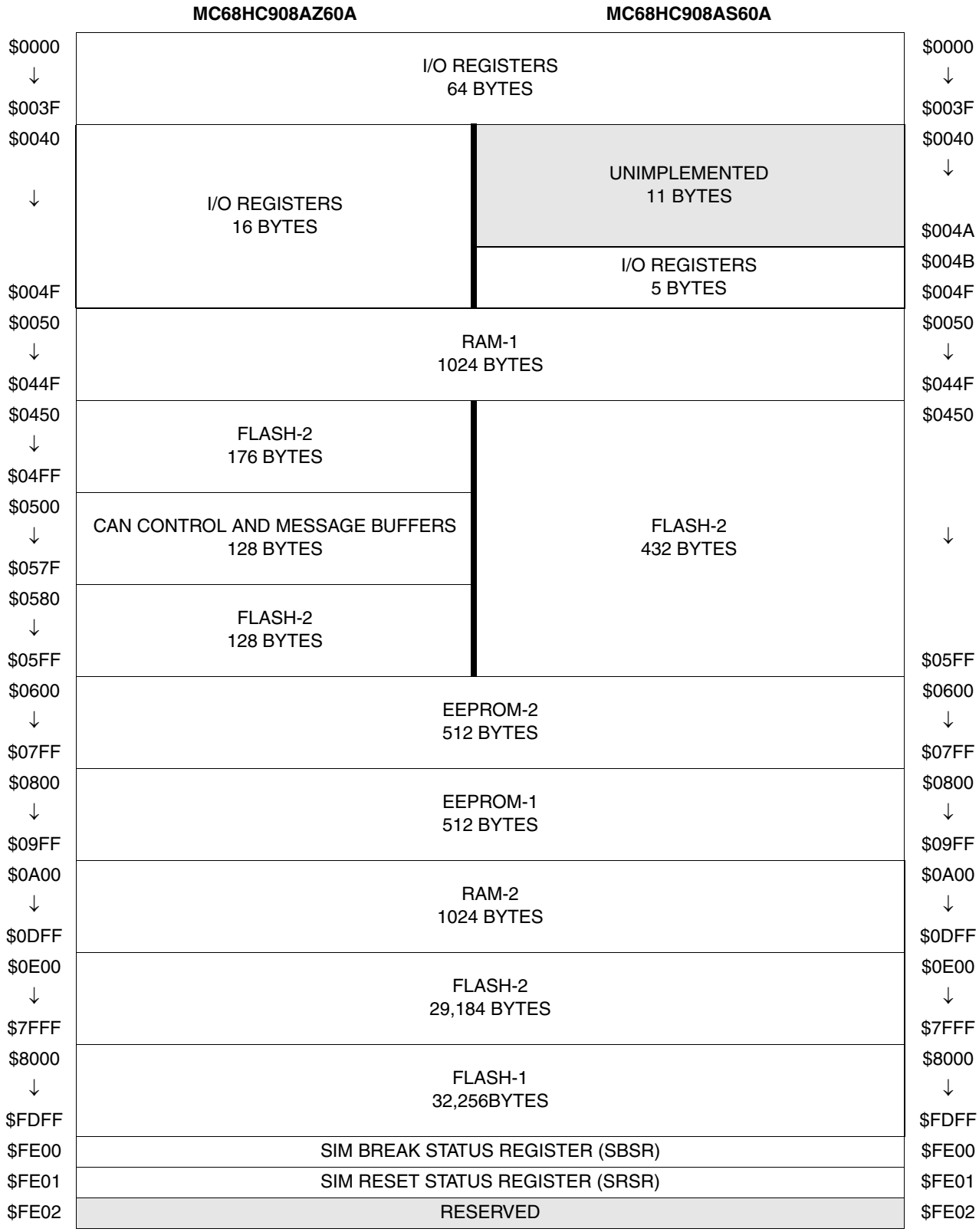



**Figure 2-1. Memory Map (Sheet 1 of 3)**

| | MC68HC908AZ60A / MC68HC908AS60A | |
|---|---|---|
| $FE03 | SIM BREAK FLAG CONTROL REGISTER (SBFCR) | $FE03 |
| $FE04 | RESERVED | $FE04 |
| $FE05 | RESERVED | $FE05 |
| $FE06 | RESERVED | $FE06 |
| $FE07 | RESERVED | $FE07 |
| $FE08 | FLASH-2 CONTROL REGISTER (FL2CR) | $FE08 |
| $FE09 | CONFIGURATION WRITE-ONCE REGISER (CONFIG-2) | $FE09 |
| $FE0A | RESERVED | $FE0A |
| $FE0B | RESERVED | $FE0B |
| $FE0C | BREAK ADDRESS REGISTER HIGH (BRKH) | $FE0C |
| $FE0D | BREAK ADDRESS REGISTER LOW (BRKL) | $FE0D |
| $FE0E | BREAK STATUS AND CONTROL REGISTER (BSCR) | $FE0E |
| $FE0F | LVI STATUS REGISTER (LVISR) | $FE0F |
| $FE10 | EEPROM-1EEDIVH NONVOLATILE REGISTER(EE1DIVHNVR) | $FE10 |
| $FE11 | EEPROM-1EEDIVL NONVOLATILE REGISTER(EE1DIVLNVR) | $FE11 |
| $FE12 | RESERVED | $FE12 |
| $FE13 | RESERVED | $FE13 |
| $FE14 | RESERVED | $FE14 |
| $FE15 | RESERVED | $FE15 |
| $FE16 | RESERVED | $FE16 |
| $FE17 | RESERVED | $FE17 |
| $FE18 | RESERVED | $FE18 |
| $FE19 | RESERVED | $FE19 |
| $FE1A | EEPROM-1 EE DIVIDER HIGH REGISTER(EE1DIVH) | $FE1A |
| $FE1B | EEPROM-1 EE DIVIDER LOW REGISTER(EE1DIVL) | $FE1B |
| $FE1C | EEPROM-1 EEPROM NONVOLATILE REGISTER (EE1NVR) | $FE1C |
| $FE1D | EEPROM-1 EEPROM CONTROL REGISTER (EE1CR) | $FE1D |
| $FE1E | RESERVED | $FE1E |
| $FE1F | EEPROM-1 EEPROM ARRAY CONFIGURATION REGISTER (EE1ACR) | $FE1F |
| $FE20 ↓ $FF1F | MONITOR ROM 256BYTES | $FE20 ↓ $FF1F |
| $FF20 ↓ $FF6F | UNIMPLEMENTED 80 BYTES | $FF20 ↓ $FF6F |
| $FF70 | EEPROM-2 EEDIVH NONVOLATILE REGISTER (EE2DIVHNVR) | $FF70 |
| $FF71 | EEPROM-2 EEDIVL NONVOLATILE REGISTER (EE2DIVLNVR) | $FF71 |
| $FF72 | RESERVED | $FF72 |
| $FF73 | RESERVED | $FF73 |
| $FF74 | RESERVED | $FF74 |

**Figure 2-1. Memory Map (Sheet 2 of 3)**

| | MC68HC908AZ60A / MC68HC908AS60A | |
|---|---|---|
| $FF75 | RESERVED | $FF75 |
| $FF76 | RESERVED | $FF76 |
| $FF77 | RESERVED | $FF77 |
| $FF78 | RESERVED | $FF78 |
| $FF79 | RESERVED | $FF79 |
| $FF7A | EEPROM-2 EE DIVIDER HIGH REGISTER (EE2DIVH) | $FF7A |
| $FF7B | EEPROM-2 EE DIVIDER LOW REGISTER (EE2DIVL) | $FF7B |
| $FF7C | EEPROM-2 EEPROM NONVOLATILE REGISTER (EE2NVR) | $FF7C |
| $FF7D | EEPROM-2 EEPROM CONTROL REGISTER (EE2CR) | $FF7D |
| $FF7E | RESERVED | $FF7E |
| $FF7F | EEPROM-2 EEPROM ARRAY CONFIGURATION REGISTER (EE2ACR) | $FF7F |
| $FF80 | FLASH-1 BLOCK PROTECT REGISTER (FL1BPR) | $FF80 |
| $FF81 | FLASH-2 BLOCK PROTECT REGISTER (FL2BPR) | $FF81 |
| $FF82 ↓ $FF87 | RESERVED 6 BYTES | $FF82 ↓ $FF87 |
| $FF88 | FLASH-1 CONTROL REGISTER (FL1CR) | $FF88 |
| $FF89 | RESERVED | $FF89 |
| $FF8A | RESERVED | $FF8A |
| $FF8B ↓ $FFCB | RESERVED 65 BYTES | $FF8B ↓ $FFCB |
| $FFCC ↓ $FFFF | VECTORS 52 BYTES See Table 2-1 | $FFCC ↓ $FFFF |

1. Registers appearing in italics are for Freescale test purpose only and only appear in the Memory Map for reference.
2. While some differences between MC68HC908AS60A and MC68HC908AZ60A are highlighted, some registers remain available on both parts. Refer to individual modules for details whether these registers are active or inactive.

**Figure 2-1. Memory Map (Sheet 3 of 3)**

## 2.2 I/O Section

Addresses $0000–$004F, shown in Figure 2-2, contain the I/O Data, Status, and Control Registers.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0000 | Port A Data Register (PTA) | Read: | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| | | Write: | | | | | | | | |
| $0001 | Port B Data Register (PTB) | Read: | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| | | Write: | | | | | | | | |
| $0002 | Port C Data Register (PTC) | Read: | 0 | 0 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| | | Write: | R | R | | | | | | |
| $0003 | Port D Data Register (PTD) | Read: | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| | | Write: | | | | | | | | |
| $0004 | Data Direction Register A (DDRA) | Read: | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | Write: | | | | | | | | |
| $0005 | Data Direction Register B (DDRB) | Read: | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | Write: | | | | | | | | |
| $0006 | Data Direction Register C (DDRC) | Read: | MCLKEN | 0 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | Write: | | R | | | | | | |
| $0007 | Data Direction Register D (DDRD) | Read: | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDR2 | DDRD1 | DDRD0 |
| | | Write: | | | | | | | | |
| $0008 | Port E Data Register (PTE) | Read: | PTE7 | PTE6 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 |
| | | Write: | | | | | | | | |
| $0009 | Port F Data Register (PTF) | Read: | 0 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | | Write: | R | | | | | | | |
| $000A | Port G Data Register (PTG) | Read: | 0 | 0 | 0 | 0 | 0 | PTG2 | PTG1 | PTG0 |
| | | Write: | R | R | R | R | R | | | |
| $000B | Port H Data Register (PTH) | Read: | 0 | 0 | 0 | 0 | 0 | 0 | PTH1 | PTH0 |
| | | Write: | R | R | R | R | R | R | | |
| $000C | Data Direction Register E (DDRE) | Read: | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| | | Write: | | | | | | | | |
| $000D | Data Direction Register F (DDRF) | Read: | 0 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | | Write: | R | | | | | | | |
| $000E | Data Direction Register G (DDRG) | Read: | 0 | 0 | 0 | 0 | 0 | DDRG2 | DDRG1 | DDRG0 |
| | | Write: | R | R | R | R | R | | | |
| $000F | Data Direction Register H (DDRH) | Read: | 0 | 0 | 0 | 0 | 0 | 0 | DDRH1 | DDRH0 |
| | | Write: | R | R | R | R | R | R | | |

= Unimplemented  R = Reserved

**Figure 2-2. I/O Data, Status and Control Registers (Sheet 1 of 5)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0010 | SPI Control Register (SPCR) | Read: | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| | | Write: | | | | | | | | |
| $0011 | SPI Status and Control Register (SPSCR) | Read: | SPRF | ERRIE | OVRF | MODF | SPTE | MODFEN | SPR1 | SPR0 |
| | | Write: | | | | | | | | |
| $0012 | SPI Data Register (SPDR) | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| $0013 | SCI Control Register 1 (SCC1) | Read: | LOOPS | ENSCI | TXINV | M | WAKE | ILTY | PEN | PTY |
| | | Write: | | | | | | | | |
| $0014 | SCI Control Register 2 (SCC2) | Read: | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK |
| | | Write: | | | | | | | | |
| $0015 | SCI Control Register 3 (SCC3) | Read: | R8 | T8 | R | R | ORIE | NEIE | FEIE | PEIE |
| | | Write: | | | | | | | | |
| $0016 | SCI Status Register 1 (SCS1) | Read: | SCTE | TC | SCRF | IDLE | OR | NF | FE | PE |
| | | Write: | | | | | | | | |
| $0017 | SCI Status Register 2 (SCS2) | Read: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| | | Write: | | | | | | | | |
| $0018 | SCI Data Register (SCDR) | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| $0019 | SCI Baud Rate Register (SCBR) | Read: | 0 | 0 | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| | | Write: | | | | | | | | |
| $001A | IRQ Status and Control Register (ISCR) | Read: | 0 | 0 | 0 | 0 | IRQF | 0 | IMASK | MODE |
| | | Write: | R | R | R | R | | ACK | | |
| $001B | Keyboard Status and Control Register (KBSCR) | Read: | 0 | 0 | 0 | 0 | KEYF | 0 | IMASKK | MODEK |
| | | Write: | | | | | | ACKK | | |
| $001C | PLL Control Register (PCTL) | Read: | PLLIE | PLLF | PLLON | BCS | 1 | 1 | 1 | 1 |
| | | Write: | | | | | | | | |
| $001D | PLL Bandwidth Control Register (PBWC) | Read: | AUTO | LOCK | $\overline{\text{ACQ}}$ | XLD | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| $001E | PLL Programming Register (PPG) | Read: | MUL7 | MUL6 | MUL5 | MUL4 | VRS7 | VRS6 | VRS5 | VRS4 |
| | | Write: | | | | | | | | |
| $001F | Configuration Write-Once Register (CONFIG-1) | Read: | LVISTOP | R | LVIRST | LVIPWR | SSREC | COPL | STOP | COPD |
| | | Write: | | | | | | | | |
| $0020 | Timer A Status and Control Register (TASC) | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | R | | | |

= Unimplemented  R = Reserved

**Figure 2-2. I/O Data, Status and Control Registers (Sheet 2 of 5)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0021 | Keyboard Interrupt Enable Register (KBIER) | Read: | 0 | 0 | 0 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| | | Write: | | | | | | | | |
| $0022 | Timer A Counter Register High (TACNTH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0023 | Timer A Counter Register Low (TACNTL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0024 | Timer A Modulo Register High (TAMODH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0025 | Timer A Modulo Register Low (TAMODL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0026 | Timer A Channel 0 Status and Control Register (TASC0) | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| $0027 | Timer A Channel 0 Register High (TACH0H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0028 | Timer A Channel 0 Register Low (TACH0L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0029 | Timer A Channel 1 Status and Control Register (TASC1) | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | R | | | | | |
| $002A | Timer A Channel 1 Register High (TACH1H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $002B | Timer A Channel 1 Register Low (TACH1L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $002C | Timer A Channel 2 Status and Control Register (TASC2) | Read: | CH2F | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| | | Write: | 0 | | | | | | | |
| $002D | Timer A Channel 2 Register High (TACH2H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $002E | Timer A Channel 2 Register Low (TACH2L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $002F | Timer A Channel 3 Status and Control Register (TASC3) | Read: | CH3F | CH3IE | 0 | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| | | Write: | 0 | | R | | | | | |
| $0030 | Timer A Channel 3 Register High (TACH3H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0031 | Timer A Channel 3 Register Low (TACH3L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure 2-2. I/O Data, Status and Control Registers (Sheet 3 of 5)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0032 | Timer A Channel 4 Status and Control Register (TASC4) | Read: | CH4F | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| | | Write: | 0 | | | | | | | |
| $0033 | Timer A Channel 4 Register High (TACH4H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0034 | Timer A Channel 4 Register Low (TACH4L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0035 | Timer A Channel 5 Status and Control Register (TASC5) | Read: | CH5F | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| | | Write: | 0 | | R | | | | | |
| $0036 | Timer A Channel 5 Register High (TACH5H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0037 | Timer A Channel 5 Register Low (TACH5L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0038 | Analog-to-Digital Status and Control Register (ADSCR) | Read: | COCO | AIEN | ADCO | ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 |
| | | Write: | R | | | | | | | |
| $0039 | Analog-to-Digital Data Register (ADR) | Read: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| | | Write: | | | | | | | | |
| $003A | Analog-to-Digital Input Clock Register (ADICLK) | Read: | ADIV2 | ADIV1 | ADIV0 | ADICLK | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| $003B | BDLC Analog and Roundtrip Delay Register (BARD) | Read: | ATE | RXPOL | 0 | 0 | BO3 | BO2 | BO1 | BO0 |
| | | Write: | | | R | R | | | | |
| $003C | BDLC Control Register 1 (BCR1) | Read: | IMSG | CLKS | R1 | R0 | 0 | 0 | IE | WCM |
| | | Write: | | | | | R | R | | |
| $003D | BDLC Control Register 2 (BCR2) | Read: | ALOOP | DLOOP | RX4XE | NBFS | TEOD | TSIFR | TMIFR1 | TMIFR0 |
| | | Write: | | | | | | | | |
| $003E | BDLC State Vector Register (BSVR) | Read: | 0 | 0 | I3 | I2 | I1 | I0 | 0 | 0 |
| | | Write: | R | R | R | R | R | R | R | R |
| $003F | BDLC Data Register (BDR) | Read: | BD7 | BD6 | BD5 | BD4 | BD3 | BD2 | BD1 | BD0 |
| | | Write: | | | | | | | | |
| $0040 | Timer B Status and Control Register (TBSCR) | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | R | | | |
| $0041 | Timer B Counter Register High (TBCNTH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0042 | Timer B Counter Register Low (TBCNTL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure 2-2. I/O Data, Status and Control Registers (Sheet 4 of 5)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0043 | Timer B Modulo Register High (TBMODH) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0044 | Timer B Modulo Register Low (TBMODL) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0045 | Timer B CH0 Status and Control Register (TBSC0) | Read:<br>Write: | CH0F<br>0 | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| $0046 | Timer B CH0 Register High (TBCH0H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0047 | Timer B CH0 Register Low (TBCH0L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0048 | Timer B CH1 Status and Control Register (TBSC1) | Read:<br>Write: | CH1F<br>0 | CH1IE | 0<br>R | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| $0049 | Timer B CH1 Register High (TBCH1H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $004A | Timer B CH1 Register Low (TBCH1L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $004B | PIT Status and Control Register (PSC) | Read:<br>Write: | POF<br>0 | POIE | PSTOP | 0<br>PRST | 0 | PPS2 | PPS1 | PPS0 |
| $004C | PIT Counter Register High (PCNTH) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $004D | PIT Counter Register Low (PCNTL) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $004E | PIT Modulo Register High (PMODH) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $004F | PIT Modulo Register Low (PMODL) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

= Unimplemented  R = Reserved

**Figure 2-2. I/O Data, Status and Control Registers (Sheet 5 of 5)**

All registers are shown for both MC68HC908AS60A and MC68HC908AZ60A. Refer to individual module chapters to determine if the module is available and the register active or not.

## 2.3 Additional Status and Control Registers

Selected addresses in the range $FE00 to $FFCB contain additional Status and Control registers as shown in Figure 2-3. A noted exception is the COP Control Register (COPCTL) at address $FFFF.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE00 | SIM Break Status Register (SBSR) | Read: | R | R | R | R | R | R | BW | R |
| | | Write: | | | | | | | 0 | |
| $FE01 | SIM Reset Status Register (SRSR) | Read: | POR | PIN | COP | ILOP | ILAD | 0 | LVI | 0 |
| | | Write: | | | | | | | | |
| $FE03 | SIM Break Flag Control Register (SBFCR) | Read: | BCFE | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| $FE08 | FLASH-2 Control Register (FL2CR) | Read: | 0 | 0 | 0 | 0 | HVEN | VERF | ERASE | PGM |
| | | Write: | | | | | | | | |
| $FE09 | Configuration Write-Once Register (CONFIG-2) | Read: | EEDIVCLK | R | R | MSCAND | AT60A | R | R | AZxx |
| | | Write: | | | | | R | | | |
| $FE0C | Break Address Register High (BRKH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $FE0D | Break Address Register Low (BRKL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $FE0E | Break Status and Control Register (BRKSCR) | Read: | BRKE | BRKA | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| $FE0F | LVI Status Register (LVISR) | Read: | LVIOUT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| $FE10 | EE1DIV Hi Nonvolatile Register (EE1DIVHNVR) | Read: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| $FE11 | EE1DIV Lo Nonvolatile Register (EE1DIVLNVR) | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| $FE1A | EE1DIV Divider High Register (EE1DIVH) | Read: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| $FE1B | EE1DIV Divider Low Register (EE1DIVL) | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| $FE1C | EEPROM-1 Nonvolatile Register (EE1NVR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| $FE1D | EEPROM-1 Control Register (EE1CR) | Read: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure 2-3. Additional Status and Control Registers (Sheet 1 of 2)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE1F | EEPROM-1 Array Configuration Register (EE1ACR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| $FF70 | EE2DIV Hi Nonvolatile Register (EE2DIVHNVR) | Read: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| $FF71 | EE2DIV Lo Nonvolatile Register (EE2DIVLNVR) | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| $FF7A | EE2DIV Divider High Register (EE2DIVH) | Read: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| $FF7B | EE2DIV Divider Low Register (EE2DIVL) | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| $FE7C | EEPROM-2 Nonvolatile Register (EE2NVR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| $FE7D | EEPROM-2 Control Register (EE2CR) | Read: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| | | Write: | | | | | | | | |
| $FE7F | EEPROM-2 Array Configuration Register (EE2ACR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| $FF80 | FLASH-1 Block Protect Register (FL1BPR) | Read: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| | | Write: | | | | | | | | |
| $FF81 | FLASH-2 Block Protect Register (FL2BPR) | Read: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| | | Write: | | | | | | | | |
| $FF88 | FLASH-1 Control Register (FL1CR) | Read: | 0 | 0 | 0 | 0 | HVEN | VERF | ERASE | PGM |
| | | Write: | | | | | | | | |
| $FFFF | COP Control Register (COPCTL) | Read: | LOW BYTE OF RESET VECTOR | | | | | | | |
| | | Write: | WRITING TO $FFFF CLEARS COP COUNTER | | | | | | | |

= Unimplemented  R = Reserved

**Figure 2-3. Additional Status and Control Registers (Sheet 2 of 2)**

## 2.4 Vector Addresses and Priority

Addresses in the range $FFCC to $FFFF contain the user-specified vector locations. The vector addresses are shown in Table 2-1. Please note that certain vector addresses differ between the MC68HC908AS60A and the MC68HC908AZ60A as shown in the table. It is recommended that all vector addresses are defined.

**Table 2-1. Vector Addresses**

| | | Vector | |
|---|---|---|---|
| | **Address** | **MC68HC908AZ60A** | **MC68HC908AS60A** |
| Lowest Priority | $FFCC | TIMA Channel 5 Vector (High) | Reserved |
| | $FFCD | TIMA Channel 5 Vector (Low) | Reserved |
| | $FFCE | TIMA Channel 4 Vector (High) | Reserved |
| | $FFCF | TIMA Channel 4 Vector (Low) | Reserved |
| | $FFD0 | ADC Vector (High) | Reserved |
| | $FFD1 | ADC Vector (Low) | Reserved |
| | $FFD2 | Keyboard Vector (High) | |
| | $FFD3 | Keyboard Vector (Low) | |
| | $FFD4 | SCI Transmit Vector (High) | Reserved |
| | $FFD5 | SCI Transmit Vector (Low) | Reserved |
| | $FFD6 | SCI Receive Vector (High) | Reserved |
| | $FFD7 | SCI Receive Vector (Low) | Reserved |
| | $FFD8 | SCI Error Vector (High) | Reserved |
| | $FFD9 | SCI Error Vector (Low) | Reserved |
| | $FFDA | CAN Transmit Vector (High) | PIT Vector (High) |
| | $FFDB | CAN Transmit Vector (Low) | PIT Vector (Low) |
| | $FFDC | CAN Receive Vector (High) | BDLC Vector (High) |
| | $FFDD | CAN Receive Vector (Low) | BDLC Vector (Low) |
| | $FFDE | CAN Error Vector (High) | ADC Vector (High) |
| | $FFDF | CAN Error Vector (Low) | ADC Vector (Low) |
| | $FFE0 | CAN Wakeup Vector (High) | SCI Transmit Vector (High) |
| | $FFE1 | CAN Wakeup Vector (Low) | SCI Transmit Vector (Low) |
| | $FFE2 | SPI Transmit Vector (High) | SCI Receive Vector (High) |
| | $FFE3 | SPI Transmit Vector (Low) | SCI Receive Vector (Low) |
| | $FFE4 | SPI Receive Vector (High) | SCI Error Vector (High) |
| | $FFE5 | SPI Receive Vector (Low) | SCI Error Vector (Low) |
| | $FFE6 | TIMB Overflow Vector (High) | SPI Transmit Vector (High) |
| | $FFE7 | TIMB Overflow Vector (Low) | SPI Transmit Vector (Low) |
| | $FFE8 | TIMB CH1 Vector (High) | SPI Receive Vector (High) |
| | $FFE9 | TIMB CH1 Vector (Low) | SPI Receive Vector (Low) |
| | $FFEA | TIMB CH0 Vector (High) | TIMA Overflow Vector (High) |
| | $FFEB | TIMB CH0 Vector (Low) | TIMA Overflow Vector (Low) |
| | $FFEC | TIMA Overflow Vector (High) | TIMA Channel 5 Vector (High) |

**Table 2-1. Vector Addresses (Continued)**

| | | Vector | |
|---|---|---|---|
| | **Address** | **MC68HC908AZ60A** | **MC68HC908AS60A** |
| | $FFED | TIMA Overflow Vector (Low) | TIMA Channel 5 Vector (Low) |
| | $FFEE | TIMA CH3 Vector (High) | TIMA Channel 4 Vector (High) |
| | $FFEF | TIMA CH3 Vector (Low) | TIMA Channel 4 Vector (Low) |
| | $FFF0 | TIMA CH2 Vector (High) | TIMA Channel 3 Vector (High) |
| | $FFF1 | TIMA CH2 Vector (Low) | TIMA Channel 3 Vector (Low) |
| | $FFF2 | TIMA CH1 Vector (High) | TIMA Channel 2 Vector (High) |
| | $FFF3 | TIMA CH1 Vector (Low) | TIMA Channel 2 Vector (Low) |
| | $FFF4 | TIMA CH0 Vector (High) | TIMA Channel 1 Vector (High) |
| | $FFF5 | TIMA CH0 Vector (Low) | TIMA Channel 1 Vector (Low) |
| | $FFF6 | PIT Vector (High) | TIMA Channel 0 Vector (High) |
| | $FFF7 | PIT Vector (Low) | TIMA Channel 0 Vector (Low) |
| | $FFF8 | PLL Vector (High) | |
| | $FFF9 | PLL Vector (Low) | |
| | $FFFA | IRQ1 Vector (High) | |
| | $FFFB | IRQ1 Vector (Low) | |
| | $FFFC | SWI Vector (High) | |
| | $FFFD | SWI Vector (Low) | |
| | $FFFE | Reset Vector (High) | |
| Highest Priority | $FFFF | Reset Vector (Low) | |

# Chapter 3
# Random-Access Memory (RAM)

## 3.1 Introduction

This chapter describes the 2048 bytes of random-access memory (RAM).

## 3.2 Functional Description

Addresses $0050 through $044F and $0A00 through $0DFF are RAM locations. The location of the stack RAM is programmable with the reset stack pointer instruction (RSP). The 16-bit stack pointer allows the stack RAM 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 176 bytes of RAM. Because the location of the stack RAM is programmable, all page zero RAM locations can be used for input/output (I/O) control and user data or code. When the stack pointer is moved from its reset location at $00FF, direct addressing mode instructions can access all page zero RAM locations efficiently. Page zero RAM, therefore, provides ideal locations for frequently accessed global variables.

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

***NOTE***

*For M68HC05, M6805, and M146805 compatibility, the H register is not stacked.*

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

***NOTE***

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

# Chapter 4
# FLASH-1 Memory

## 4.1 Introduction

This chapter describes the operation of the embedded FLASH-1 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.

## 4.2 Functional Description

The FLASH-1 memory is an array of 32,256 bytes with two bytes of block protection (one byte for protecting areas within FLASH-1 array and one byte for protecting areas within FLASH-2 array) and an additional 40 bytes of user vectors on the MC68HC908AS60A and 52 bytes of user vectors on the MC68HC908AZ60A. An erased bit reads as a logic 1 and a programmed bit reads as a logic 0.

Memory in the FLASH-1 array is organized into rows within pages. There are two rows of memory per page with 64 bytes per row. The minimum erase block size is a single page,128 bytes. Programming is performed on a per-row basis, 64 bytes at a time. Program and erase operations are facilitated through control bits in the FLASH-1 Control Register (FL1CR). Details for these operations appear later in this chapter.

The FLASH-1 memory map consists of:

- $8000–$FDFF: User Memory (32,256 bytes)
- $FF80: FLASH-1 Block Protect Register (FL1BPR)
- $FF81: FLASH-2 Block Protect Register (FL2BPR)
- $FF88: FLASH-1 Control Register (FL1CR)
- $FFCC–$FFFF: these locations are reserved for user-defined interrupt and reset vectors. (Please see 2.4 Vector Addresses and Priority for details)

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.

## 4.3 FLASH-1 Control and Block Protect Registers

The FLASH-1 array has two registers that control its operation, the FLASH-1 Control Register (FL1CR) and the FLASH-1 Block Protect Register (FL1BPR).

### 4.3.1 FLASH-1 Control Register

The FLASH-1 Control Register (FL1CR) controls FLASH-1 program and erase operations.

Address: $FF88

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

= Unimplemented

**Figure 4-1. FLASH-1 Control Register (FL1CR)**

**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**

Setting this read/write bit configures the FLASH-1 array for mass or page erase operation.

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 set at the same time.

1 = Erase operation selected
0 = Erase operation unselected

**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 unselected

### 4.3.2 FLASH-1 Block Protect Register

The FLASH-1 Block Protect Register (FL1BPR) is implemented as a byte within the FLASH-1 memory and therefore can only be written during a FLASH programming sequence. The value in this register determines the starting location of the protected range within the FLASH-1 memory.

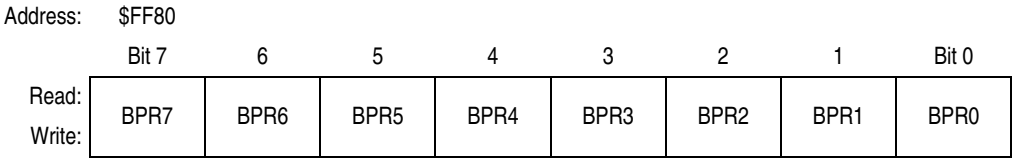



**Figure 4-2. FLASH-1 Block Protect Register (FL1BPR)**

**FL1BPR[7:0] — Block Protect Register Bit 7 to Bit 0**

These eight bits represent bits [14:7] of a 16-bit memory address. Bit 15 is logic 1 and bits [6:0] are logic 0s.

The resultant 16-bit address is used for specifying the start address of the FLASH-1 memory for block protection. FLASH-1 is protected from this start address to the end of FLASH-1 memory at $FFFF. With this mechanism, the protect start address can be $XX00 and $XX80 (128 byte page boundaries) within the FLASH-1 array.

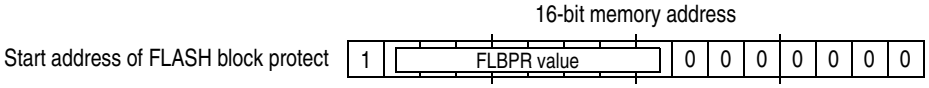



**Figure 4-3. FLASH-1 Block Protect Start Address**

**FLASH-1 Protected Ranges**

| FL1BPR[7:0] | Protected Range |
|---|---|
| $FF | No Protection |
| $FE | $FF00 – $FFFF |
| $FD | $FE80 – $FFFF |
| ↓ | ↓ |
| $0B | $8580 – $FFFF |
| $0A | $8500 – $FFFF |
| $09 | $8480 – $FFFF |
| $08 | $8400 – $FFFF |
| ↓ | ↓ |
| $04 | $8200 – $FFFF |
| $03 | $8180 – $FFFF |
| $02 | $8100 – $FFFF |
| $01 | $8080 – $FFFF |
| $00 | $8000 – $FFFF |

Decreasing the value in FL1BPR by one increases the protected range by one page (128 bytes). However, programming the block protect register with $FE protects a range twice that size, 256 bytes, in the corresponding array. $FE means that locations $FF00–$FFFF are protected in FLASH-1.

The FLASH memory does not exist at some locations. The block protection range configuration is unaffected if FLASH memory does not exist in that range. Refer to the memory map and make sure that the desired locations are protected.

## 4.4 FLASH-1 Block 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 for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by using the FLASH-1 Block Protection Register (FL1BPR). FL1BPR determines the range of the FLASH-1 memory which is to be protected. The range of the protected area starts from a location defined by FL1BPR and ends at the bottom of the FLASH-1 memory ($FFFF). When the memory is protected, the HVEN bit can not be set in either ERASE or PROGRAM operations.

***NOTE***

*In performing a program or erase operation, the FLASH-1 Block Protect Register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.*

When the FLASH-1 Block Protect Register is programmed with all 0's, the entire memory is protected from being programmed and erased. When all the bits are erased (all 1's), the entire memory is accessible for program and erase.

When bits within FL1BPR are programmed (logic 0), they lock a block of memory address ranges as shown in 4.3.2 FLASH-1 Block Protect Register. If FL1BPR is programmed with any value other than $FF, the protected block of FLASH memory can not be erased or programmed.

***NOTE***

*The vector locations and the FLASH Block Protect Registers are located in the same page. FL1BPR and FL2BPR are not protected with special hardware or software; therefore, if this page is not protected by FL1BPR and the vector locations are erased by either a page or a mass erase operation, both FL1BPR and FL2BPR will also get erased.*

## 4.5 FLASH-1 Mass Erase Operation

Use this step-by-step procedure to erase the entire FLASH-1 memory to read as logic 1:

1. Set both the ERASE bit and the MASS bit in the FLASH-1 Control Register (FL1CR).
2. Read the FLASH-1 Block Protect Register (FL1BPR).
3. Write to any FLASH-1 address within the FLASH-1 array with any data.

***NOTE***

*If the address written to in Step 3 is within address space protected by the FLASH-1 Block Protect Register (FL1BPR), no erase will occur.*

4. Wait for a time, $t_{NVS}$.
5. Set the HVEN bit.
6. Wait for a time, $t_{MERASE}$.
7. Clear the ERASE bit.
8. Wait for a time, $t_{NVHL}$.
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, after which the memory can be accessed in normal read mode.

***NOTE***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Care must be taken however to ensure that these operations do not access any address within the FLASH array memory space such as the COP Control Register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

## 4.6 FLASH-1 Page Erase Operation

Use this step-by-step procedure to erase a page (128 bytes) of FLASH-1 memory to read as logic 1:

1. Set the ERASE bit and clear the MASS bit in the FLASH-1 Control Register (FL1CR).
2. Read the FLASH-1 Block Protect Register (FL1BPR).
3. Write any data to any FLASH-1 address within the address range of the page (128 byte block) to be erased.
4. Wait for time, $t_{NVS}$.
5. Set the HVEN bit.
6. Wait for time, $t_{ERASE}$.
7. Clear the ERASE bit.
8. Wait for time, $t_{NVH}$.
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, after which the memory can be accessed in normal read mode.

***NOTE***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Care must be taken however to ensure that these operations do not access any address within the FLASH array memory space such as the COP Control Register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

## 4.7 FLASH-1 Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 64 consecutive bytes with address ranges as follows:

- $XX00 to $XX3F
- $XX40 to $XX7F
- $XX80 to $XXBF
- $XXC0 to $XXFF

During the programming cycle, make sure that all addresses being written to fit within one of the ranges specified above. Attempts to program addresses in different row ranges in one programming cycle will fail.

Use this step-by-step procedure to program a row of FLASH-1 memory.

***NOTE***

*In order to avoid program disturbs, the row must be erased before any byte on that row is programmed.*

1. Set the PGM bit in the FLASH-1 Control Register (FL1CR). This configures the memory for program operation and enables the latching of address and data programming.
2. Read the FLASH-1 Block Protect Register (FL1BPR).

3. Write to any FLASH-1 address within the row address range desired with any data.
4. Wait for time, $t_{NVS}$.
5. Set the HVEN bit.
6. Wait for time, $t_{PGS}$.
7. Write data byte to the FLASH-1 address to be programmed.
8. Wait for time, $t_{PROG}$.
9. Repeat step 7 and 8 until all the bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for time, $t_{NVH}$.
12. Clear the HVEN bit.
13. Wait for a time, $t_{RCV}$, after which the memory can be accessed in normal read mode.

The FLASH Programming Algorithm Flowchart is shown in Figure 4-4.

***NOTE***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Care must be taken however to ensure that these operations do not access any address within the FLASH array memory space such as the COP Control Register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

***D.*** *Do not exceed $t_{PROG}$ maximum or $t_{HV}$ maximum. $t_{HV}$ is defined as the cumulative high voltage programming time to the same row before next erase. $t_{HV}$ must satisfy this condition: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} X\ 64) \leq t_{HV}$ max. Please also see 28.1.14 FLASH Memory Characteristics.*

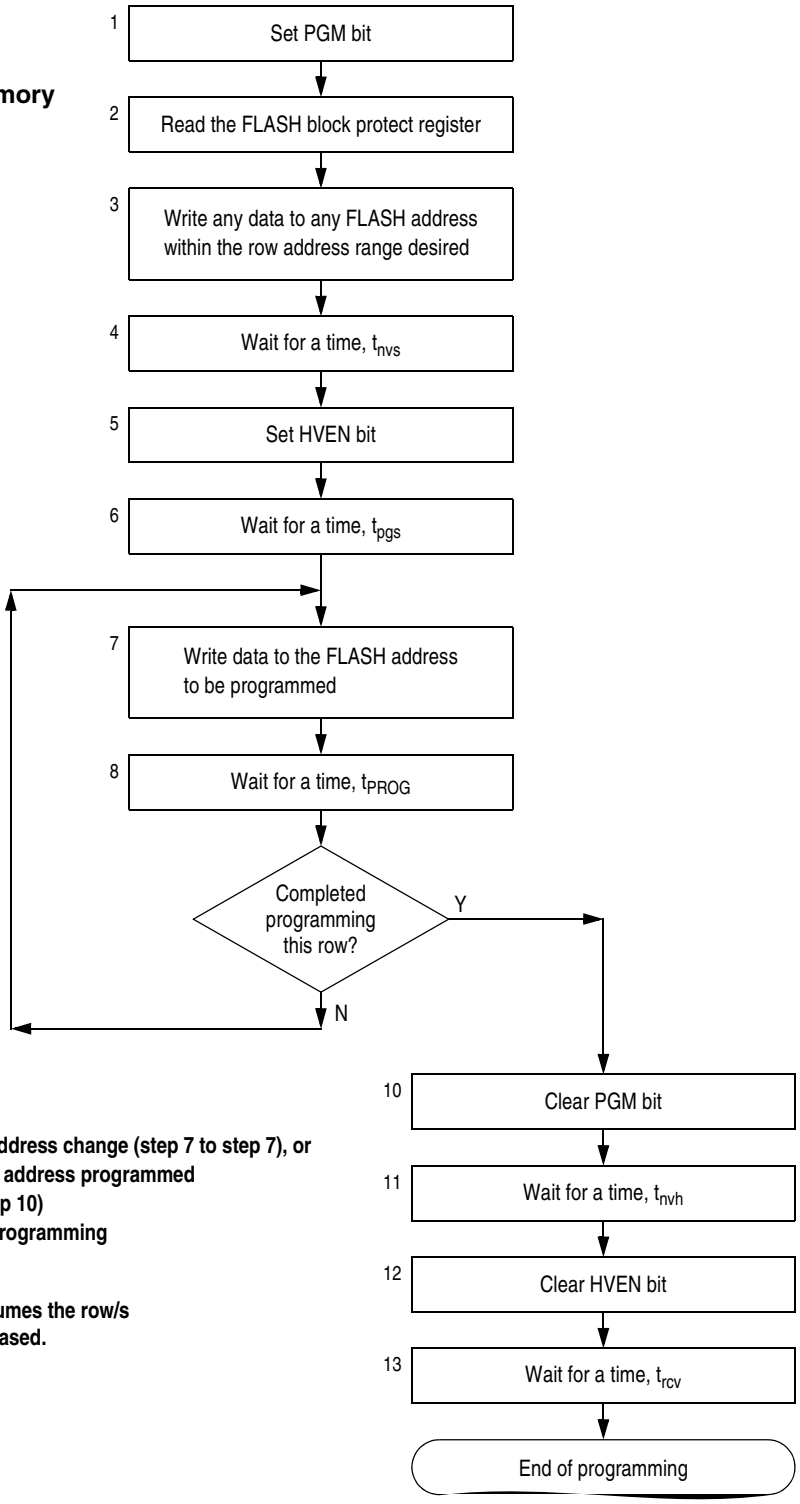
***E.*** *The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing the PGM bit (step 7 to step 10) must not exceed the maximum programming time, $t_{PROG}$ max.*

***F.*** *Be cautious when programming the FLASH-1 array to ensure that non-FLASH locations are not used as the address that is written to when selecting either the desired row address range in step 3 of the algorithm or the byte to be programmed in step 7 of the algorithm. This applies particularly to:*

*$FFD2-$FFD3 and $FFDA-$FFFF: Vector area on MC68HC908AS60A (40 bytes)*

*$FFCC-$FFFF: Vector area on MC68HC908AZ60A (52 bytes)*

**Algorithm for programming a row (64 bytes) of FLASH memory**




**NOTE:**

**The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing PGM bit (step 7 to step 10) 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 4-4. FLASH Programming Algorithm Flowchart**

## 4.8 Low-Power Modes

The WAIT and STOP instructions will place the MCU in low power consumption standby modes.

### 4.8.1 WAIT Mode

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. Wait mode will suspend any FLASH program/erase operations and leave the memory in a Standby Mode.

### 4.8.2 STOP Mode

Putting the MCU into stop mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. Stop mode will suspend any FLASH program/erase operations and leave the memory in a Standby Mode.

***NOTE***

*Standby Mode is the power saving mode of the FLASH module, in which all internal control signals to the FLASH are inactive and the current consumption of the FLASH is minimum.*

# Chapter 5
# FLASH-2 Memory

## 5.1 Introduction

This chapter describes the operation of the embedded FLASH-2 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.

## 5.2 Functional Description

The FLASH-2 memory is a non-continuos array consisting of a total of 29,616 bytes on the MC68HC908AS60A and 29,488 bytes on the MC68HC908AZ60A. An erased bit reads as a logic 1 and a programmed bit reads as a logic 0.

Memory in the FLASH-2 array is organized into rows within pages. There are two rows of memory per page with 64 bytes per row. The minimum erase block size is a single page,128 bytes. Programming is performed on a per-row basis, 64 bytes at a time. Program and erase operations are facilitated through control bits in the FLASH-2 Control Register (FL2CR). Details for these operations appear later in this chapter.

The FLASH-2 memory map consists of:

- $0450–$05FF: User Memory on MC68HC908AS60A (432 bytes)
- $0450–$04FF: User Memory on MC68HC908AZ60A (176 bytes)
- $0580–$05FF: User Memory on MC68HC908AZ60A (128 bytes)
- $0E00–$7FFF: User Memory (29,616 bytes)
- $FF81: FLASH-2 Block Protect Register (FL2BPR)

***NOTE***

*FL2BPR physically resides within FLASH-1 memory addressing space*

- $FE08: FLASH-2 Control Register (FL2CR)

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.

## 5.3 FLASH-2 Control and Block Protect Registers

The FLASH-2 array has two registers that control its operation, the FLASH-2 Control Register (FL2CR) and the FLASH-2 Block Protect Register (FL2BPR).

### 5.3.1 FLASH-2 Control Register

The FLASH-2 Control Register (FL2CR) controls FLASH-2 program and erase operations.

Address: $FE08

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

= Unimplemented

**Figure 5-1. FLASH-2 Control Register (FL2CR)**

**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**

Setting this read/write bit configures the FLASH-2 array for mass or page erase operation.

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 set at the same time.

1 = Erase operation selected
0 = Erase operation unselected

**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 unselected

### 5.3.2 FLASH-2 Block Protect Register

The FLASH-2 Block Protect Register (FL2BPR) is implemented as a byte within the FLASH-1 memory and therefore can only be written during a FLASH programming sequence. The value in this register determines the starting location of the protected range within the FLASH-2 memory.

Address: $FF81

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

**Figure 5-2. FLASH-2 Block Protect Register (FL2BPR)**

***NOTE***

*The FLASH-2 Block Protect Register (FL2BPR) controls the block protection for the FLASH-2 array. However, FL2BPR is implemented within the FLASH-1 memory array and therefore, the FLASH-1 Control Register (FL1CR) must be used to program/erase FL2BPR.*

### FL2BPR[7:0] — Block Protect Register Bit 7 to Bit 0

These eight bits represent bits [14:7] of a 16-bit memory address. Bit 15 is logic 1 and bits [6:0] are logic 0s.

The resultant 16-bit address is used for specifying the start address of the FLASH-2 memory for block protection. FLASH-2 is protected from this start address to the end of FLASH-2 memory at $7FFF. With this mechanism, the protect start address can be $XX00 and $XX80 (128 byte page boundaries) within the FLASH-2 array.

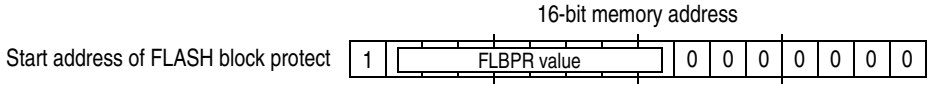



**Figure 5-3. FLASH-2 Block Protect Start Address**

### FLASH-2 Protected Ranges:

| FL2BPR[7:0] | Protected Range |
|---|---|
| $FF | No Protection |
| $FE | $7F00 – $7FFF |
| $FD | $7E80 – $7FFF |
| ↓ | ↓ |
| $0B | $0580 – $7FFF |
| $0A | $0500 – $7FFF |
| $09 | $0480 – $7FFF |
| $08 | $0450 – $7FFF |
| ↓ | ↓ |
| $04 | $0450 – $7FFF |
| $03 | $0450 – $7FFF |
| $02 | $0450 – $7FFF |
| $01 | $0450 – $7FFF |
| $00 | $0450 – $7FFF |

Decreasing the value in FL2BPR by one increases the protected range by one page (128 bytes). However, programming the block protect register with $FE protects a range twice that size, 256 bytes, in the corresponding array. $FE means that locations $7F00–$7FFF are protected in FLASH-2.

The FLASH memory does not exist at some locations. The block protection range configuration is unaffected if FLASH memory does not exist in that range. Refer to the memory map and make sure that the desired locations are protected.

## 5.4 FLASH-2 Block 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 for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by using the FLASH-2 Block Protection Register (FL2BPR). FL2BPR determines the range of the FLASH-2 memory which is to be protected. The range of the protected area starts from a location defined by FL2BPR and ends at the bottom of the FLASH-2 memory ($7FFF). When the memory is protected, the HVEN bit can not be set in either ERASE or PROGRAM operations.

***NOTE***

*In performing a program or erase operation, the FLASH-2 Block Protect Register must be read after setting the PGM or ERASE bit and before asserting the HVEN bit.*

When the FLASH-2 Block Protect Register is programmed with all 0's, the entire memory is protected from being programmed and erased. When all the bits are erased (all 1's), the entire memory is accessible for program and erase.

When bits within FL2BPR are programmed (logic 0), they lock a block of memory address ranges as shown in 5.3.2 FLASH-2 Block Protect Register. If FL2BPR is programmed with any value other than $FF, the protected block of FLASH memory can not be erased or programmed.

***NOTE***

*The vector locations and the FLASH Block Protect Registers are located in the same page. FL1BPR and FL2BPR are not protected with special hardware or software; therefore, if this page is not protected by FL1BPR and the vector locations are erased by either a page or a mass erase operation, both FL1BPR and FL2BPR will also get erased.*

## 5.5 FLASH-2 Mass Erase Operation

Use this step-by-step procedure to erase the entire FLASH-2 memory to read as logic 1:

1. Set both the ERASE bit and the MASS bit in the FLASH-2 Control Register (FL2CR).
2. Read the FLASH-2 Block Protect Register (FL2BPR).
3. Write to any FLASH-2 address within the FLASH-2 array with any data.

***NOTE***

*If the address written to in Step 3 is within address space protected by the FLASH-2 Block Protect Register (FL2BPR), no erase will occur.*

4. Wait for a time, $t_{NVS}$.
5. Set the HVEN bit.
6. Wait for a time, $t_{MERASE}$.
7. Clear the ERASE bit.
8. Wait for a time, $t_{NVHL}$.
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, after which the memory can be accessed in normal read mode.

***NOTE***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Care must be taken however to ensure that these operations do not access any address within the FLASH array memory space such as the COP Control Register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

## 5.6 FLASH-2 Page Erase Operation

Use this step-by-step procedure to erase a page (128 bytes) of FLASH-2 memory to read as logic 1:

1. Set the ERASE bit and clear the MASS bit in the FLASH-2 Control Register (FL2CR).
2. Read the FLASH-2 Block Protect Register (FL2BPR).
3. Write any data to any FLASH-2 address within the address range of the page (128 byte block) to be erased.
4. Wait for time, $t_{NVS}$.
5. Set the HVEN bit.
6. Wait for time, $t_{ERASE}$.
7. Clear the ERASE bit.
8. Wait for time, $t_{NVH}$.
9. Clear the HVEN bit.
10. Wait for a time, $t_{RCV}$, after which the memory can be accessed in normal read mode.

***NOTE***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Care must be taken however to ensure that these operations do not access any address within the FLASH array memory space such as the COP Control Register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

## 5.7 FLASH-2 Program Operation

Programming of the FLASH memory is done on a row basis. A row consists of 64 consecutive bytes with address ranges as follows:

- $XX00 to $XX3F
- $XX40 to $XX7F
- $XX80 to $XXBF
- $XXC0 to $XXFF

During the programming cycle, make sure that all addresses being written to fit within one of the ranges specified above. Attempts to program addresses in different row ranges in one programming cycle will fail.

- Use this step-by-step procedure to program a row of FLASH-2 memory.

***NOTE***

*In order to avoid program disturbs, the row must be erased before any byte on that row is programmed.*

1. Set the PGM bit in the FLASH-2 Control Register (FL2CR). This configures the memory for program operation and enables the latching of address and data programming.
2. Read the FLASH-2 Block Protect Register (FL2BPR).
3. Write to any FLASH-2 address within the row address range desired with any data.

4. Wait for time, $t_{NVS}$.
5. Set the HVEN bit.
6. Wait for time, $t_{PGS}$.
7. Write data byte to the FLASH-2 address to be programmed.
8. Wait for time, $t_{PROG}$.
9. Repeat step 7 and 8 until all the bytes within the row are programmed.
10. Clear the PGM bit.
11. Wait for time, $t_{NVH}$.
12. Clear the HVEN bit.
13. Wait for a time, $t_{RCV}$, after which the memory can be accessed in normal read mode.

The FLASH Programming Algorithm Flowchart is shown in Figure 5-4.

***NOTE***

***A.*** *Programming and erasing of FLASH locations can not be performed by code being executed from the same FLASH array.*

***B.*** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Care must be taken however to ensure that these operations do not access any address within the FLASH array memory space such as the COP Control Register (COPCTL) at $FFFF.*

***C.*** *It is highly recommended that interrupts be disabled during program/erase operations.*

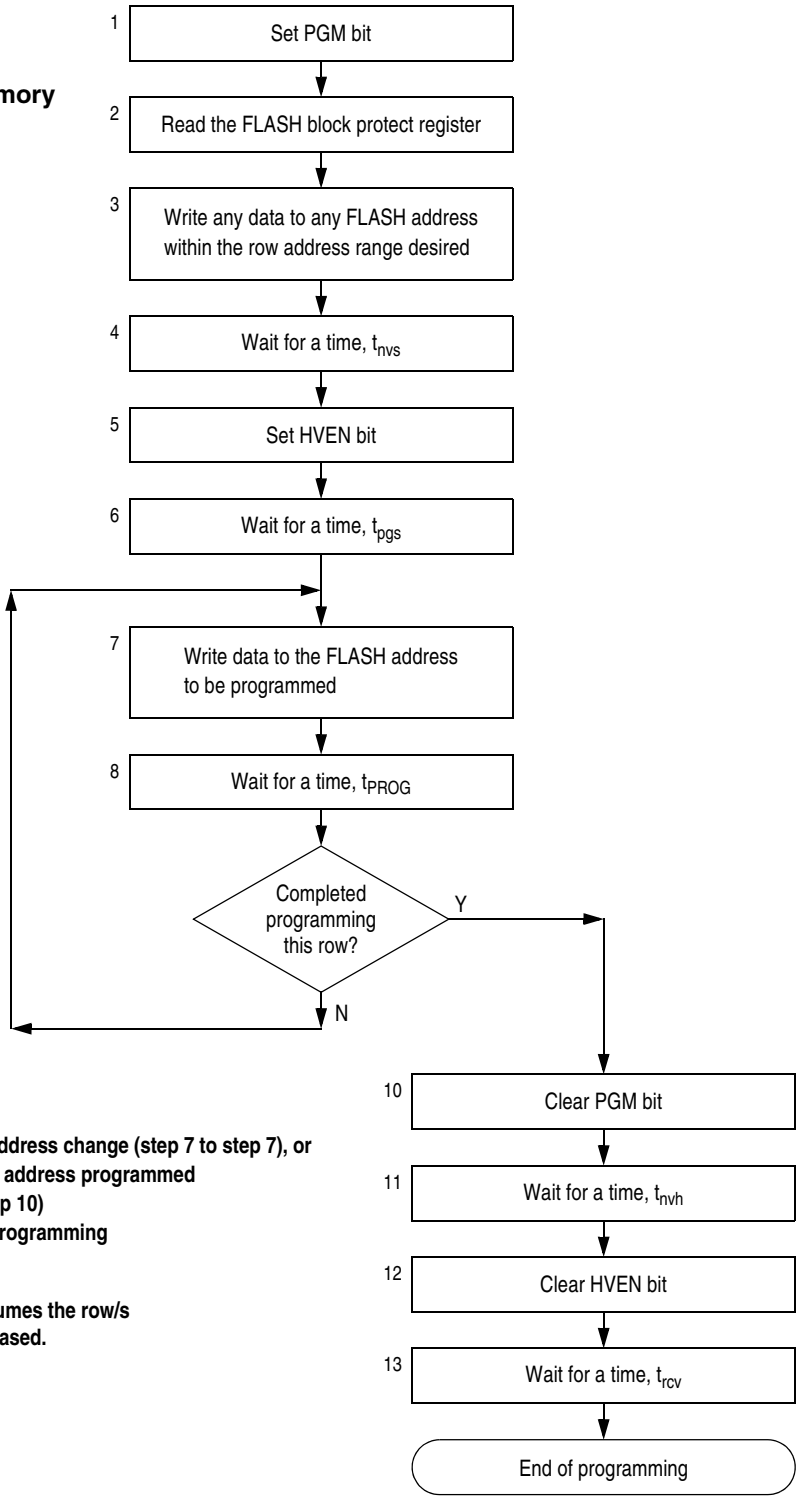
***D.*** *Do not exceed $t_{PROG}$ maximum or $t_{HV}$ maximum. $t_{HV}$ is defined as the cumulative high voltage programming time to the same row before next erase. $t_{HV}$ must satisfy this condition: $t_{NVS} + t_{NVH} + t_{PGS} + (t_{PROG} X\ 64) \leq t_{HV}$ max. Please also see 28.1.14 FLASH Memory Characteristics.*

***E.*** *The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing the PGM bit (step 7 to step 10) must not exceed the maximum programming time, $t_{PROG}$ max.*

***F.*** *Be cautious when programming the FLASH-2 array to ensure that non-FLASH locations are not used as the address that is written to when selecting either the desired row address range in step 3 of the algorithm or the byte to be programmed in step 7 of the algorithm. This applies particularly to:*

*$0450-$047F: First row of FLASH-2 (48 bytes)*

**Algorithm for programming a row (64 bytes) of FLASH memory**




**NOTE:**

**The time between each FLASH address change (step 7 to step 7), or the time between the last FLASH address programmed to clearing PGM bit (step 7 to step 10) 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 5-4. FLASH Programming Algorithm Flowchart**

## 5.8 Low-Power Modes

The WAIT and STOP instructions will place the MCU in low power consumption standby modes.

### 5.8.1 WAIT Mode

Putting the MCU into wait mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. Wait mode will suspend any FLASH program/erase operations and leave the memory in a Standby Mode.

### 5.8.2 STOP Mode

Putting the MCU into stop mode while the FLASH is in read mode does not affect the operation of the FLASH memory directly; however, no memory activity will take place since the CPU is inactive.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. Stop mode will suspend any FLASH program/erase operations and leave the memory in a Standby Mode.

***NOTE***

*Standby Mode is the power saving mode of the FLASH module, in which all internal control signals to the FLASH are inactive and the current consumption of the FLASH is minimum.*

# Chapter 6
# EEPROM-1 Memory

## 6.1 Introduction

This chapter describes the 512 bytes of electrically erasable programmable read-only memory (EEPROM) residing at address range $0800 to $09FF. There are 1024 bytes of EEPROM available on the MC68HC908AS60A and MC68HC908AZ60A which are physically located in two 512 byte arrays. For information relating to the array covering address range $0600 to $07FF please see Chapter 7 EEPROM-2 Memory.

## 6.2 Features

Features of the EEPROM-1 include the following:

- 512 bytes Nonvolatile Memory
- Byte, Block, or Bulk Erasable
- Nonvolatile EEPROM Configuration and Block Protection Options
- On-chip Charge Pump for Programming/Erasing
- Security Option
- AUTO Bit Driven Programming/Erasing Time Feature

## 6.3 EEPROM-1 Register Summary

The EEPROM-1 Register Summary is shown in Figure 6-1.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE10 | EE1DIV Nonvolatile Register High (EE1DIVHNVR)(1) | Read: Write: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Reset: | Unaffected by reset; $FF when blank | | | | | | | |
| $FE11 | EE1DIV Nonvolatile Register Low (EE1DIVLNVR)(1) | Read: Write: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Reset: | Unaffected by reset; $FF when blank | | | | | | | |
| $FE1A | EE1 Divider Register High (EE1DIVH) | Read: Write: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Reset: | Contents of EE1DIVHNVR ($FE10), Bits [6:3] = 0 | | | | | | | |
| $FE1B | EE1 Divider Register Low (EE1DIVL) | Read: Write: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Reset: | Contents of EE1DIVLNVR ($FE11) | | | | | | | |
| $FE1C | EEPROM-1 Nonvolatile Register (EE1NVR)(1) | Read: Write: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Reset: | Unaffected by reset; $FF when blank; factory programmed $F0 | | | | | | | |
| $FE1D | EEPROM-1 Control Register (EE1CR) | Read: Write: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FE1F | EEPROM-1 Array Configuration Register (EE1ACR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| | | Reset: | Contents of EE1NVR ($FE1C) | | | | | | | |

1. Nonvolatile EEPROM register; write by programming.

= Unimplemented  R = Reserved  UNUSED = Unused

**Figure 6-1. EEPROM-1 Register Summary**

## 6.4 Functional Description

The 512 bytes of EEPROM-1 are located at $0800-$09FF and can be programmed or erased without an additional external high voltage supply. The program and erase operations are enabled through the use of an internal charge pump. For each byte of EEPROM, the write/erase endurance is 10,000 cycles.

### 6.4.1 EEPROM-1 Configuration

The 8-bit EEPROM-1 Nonvolatile Register (EE1NVR) and the 16-bit EEPROM-1 Timebase Divider Nonvolatile Register (EE1DIVNVR) contain the default settings for the following EEPROM configurations:

- EEPROM-1 Timebase Reference
- EEPROM-1 Security Option
- EEPROM-1 Block Protection

EE1NVR and EE1DIVNVR are nonvolatile EEPROM registers. They are programmed and erased in the same way as EEPROM bytes. The contents of these registers are loaded into their respective volatile registers during a MCU reset. The values in these read/write volatile registers define the EEPROM-1 configurations.

For EE1NVR, the corresponding volatile register is the EEPROM-1 Array Configuration Register (EE1ACR). For the EE1DIVNCR (two 8-bit registers: EE1DIVHNVR and EE1DIVLNVR), the corresponding volatile register is the EEPROM-1 Divider Register (EE1DIV: EE1DIVH and EE1 DIVL).

### 6.4.2 EEPROM-1 Timebase Requirements

A 35μs timebase is required by the EEPROM-1 control circuit for program and erase of EEPROM content. This timebase is derived from dividing the CGMXCLK or bus clock (selected by EEDIVCLK bit in CONFIG-2 Register) using a timebase divider circuit controlled by the 16-bit EEPROM-1 Timebase Divider EE1DIV Register (EE1DIVH and EE1DIVL).

As the CGMXCLK or bus clock is user selected, the EEPROM-1 Timebase Divider Register must be configured with the appropriate value to obtain the 35 μs. The timebase divider value is calculated by using the following formula:

$$EE1DIV = INT[\text{Reference Frequency(Hz)} \times 35 \times 10^{-6} + 0.5]$$

This value is written to the EEPROM-1 Timebase Divider Register (EE1DIVH and EE1DIVL) or programmed into the EEPROM-1 Timebase Divider Nonvolatile Register prior to any EEPROM program or erase operations (6.4.1 EEPROM-1 Configuration and 6.4.2 EEPROM-1 Timebase Requirements).

### 6.4.3 EEPROM-1 Program/Erase Protection

The EEPROM has a special feature that designates the 16 bytes of addresses from $08F0 to $08FF to be permanently secured. This program/erase protect option is enabled by programming the EEPRTCT bit in the EEPROM-1 Nonvolatile Register (EE1NVR) to a logic zero.

Once the EEPRTCT bit is programmed to 0 for the first time:

- Programming and erasing of secured locations $08F0 to $08FF is permanently disabled.
- Secured locations $08F0 to $08FF can be read as normal.
- Programming and erasing of EE1NVR is permanently disabled.
- Bulk and Block Erase operations are disabled for the unprotected locations $0800-$08EF, $0900-$09FF.
- Single byte program and erase operations are still available for locations $0800-$08EF and $0900-$09FF for all bytes that are not protected by the EEPROM-1 Block Protect EEBPx bits (see 6.4.4 EEPROM-1 Block Protection and 6.5.2 EEPROM-1 Array Configuration Register)

***NOTE***

*Once armed, the protect option is permanently enabled. As a consequence, all functions in the EE1NVR will remain in the state they were in immediately before the security was enabled.*

### 6.4.4 EEPROM-1 Block Protection

The 512 bytes of EEPROM-1 are divided into four 128-byte blocks. Each of these blocks can be protected from erase/program operations by setting the EEBPx bit in the EE1NVR. Table 6-1 shows the address ranges for the blocks.

**Table 6-1. EEPROM-1 Array Address Blocks**

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

These bits are effective after a reset or a upon read of the EE1NVR register. The block protect configuration can be modified by erasing/programming the corresponding bits in the EE1NVR register and then reading the EE1NVR register. Please see 6.5.2 EEPROM-1 Array Configuration Register for more information.

***NOTE***

*Once EEDIVSECD in the EE1DIVHNVR is programmed to 0 and after a system reset, the EE1DIV security feature is permanently enabled because the EEDIVSECD bit in the EE1DIVH is always loaded with 0 thereafter. Once this security feature is armed, erase and program mode are disabled for EE1DIVHNVR and EE1DIVLNVR. Modifications to the EE1DIVH and EE1DIVL registers are also disabled. Therefore, be cautious on programming a value into the EE1DIVHNVR.*

### 6.4.5 EEPROM-1 Programming and Erasing

The unprogrammed or erase state of an EEPROM bit is a logic 1. The factory default for all bytes within the EEPROM-1 array is $FF.

The programming operation changes an EEPROM bit from logic 1 to logic 0 (programming cannot change a bit from logic 0 to a logic 1). In a single programming operation, the minimum EEPROM programming size is one bit; the maximum is eight bits (one byte).

The erase operation changes an EEPROM bit from logic 0 to logic 1. In a single erase operation, the minimum EEPROM erase size is one byte; the maximum is the entire EEPROM-1 array.

The EEPROM can be programmed such that one or multiple bits are programmed (written to a logic 0) at a time. However, the user may never program the same bit location more than once before erasing the entire byte. In other words, the user is not allowed to program a logic 0 to a bit that is already programmed (bit state is already logic 0).

For some applications it might be advantageous to track more than 10K events with a single byte of EEPROM by programming one bit at a time. For that purpose, a special selective bit programming technique is available. An example of this technique is illustrated in Table 6-2.

**Table 6-2. Example Selective Bit Programming Description**

| Description | Program Data in Binary | Result in Binary |
|---|---|---|
| Original state of byte (erased) | n/a | 1111:1111 |
| First event is recorded by programming bit position 0 | 1111:1110 | 1111:1110 |
| Second event is recorded by programming bit position 1 | 1111:1101 | 1111:1100 |
| Third event is recorded by programming bit position 2 | 1111:1011 | 1111:1000 |
| Fourth event is recorded by programming bit position 3 | 1111:0111 | 1111:0000 |
| Events five through eight are recorded in a similar fashion | | |

Note that none of the bit locations are actually programmed more than once although the byte was programmed eight times.

When this technique is utilized, a program/erase cycle is defined as multiple program sequences (up to eight) to a unique location followed by a single erase operation.

#### *6.4.5.1 Program/Erase Using AUTO Bit*

An additional feature available for EEPROM-1 program and erase operations is the AUTO mode. When enabled, AUTO mode will activate an internal timer that will automatically terminate the program/erase cycle and clear the EEPGM bit. Please see 6.4.5.2 EEPROM-1 Programming, 6.4.5.3 EEPROM-1 Erasing, and 6.5.1 EEPROM-1 Control Register for more information.

### 6.4.5.2 EEPROM-1 Programming

The unprogrammed or erase state of an EEPROM bit is a logic 1. Programming changes the state to a logic 0. Only EEPROM bytes in the non-protected blocks and the EE1NVR register can be programmed.

Use the following procedure to program a byte of EEPROM:

1. Clear EERAS1 and EERAS0 and set EELAT in the EE1CR.(A)

**NOTE**

*If using the AUTO mode, also set the AUTO bit during Step 1.*

2. Write the desired data to the desired EEPROM address.(B)
3. Set the EEPGM bit.(C) Go to Step 7 if AUTO is set.
4. Wait for time, $t_{EEPGM}$, to program the byte.
5. Clear EEPGM bit.
6. Wait for time, $t_{EEFPV}$, for the programming voltage to fall. Go to Step 8.
7. Poll the EEPGM bit until it is cleared by the internal timer.(D)
8. Clear EELAT bits.(E)

**NOTE**

***A.*** *EERAS1 and EERAS0 must be cleared for programming. Setting the EELAT bit configures the address and data buses to latch data for programming the array. Only data with a valid EEPROM-1 address will be latched. If EELAT is set, other writes to the EE1CR will be allowed after a valid EEPROM-1 write.*

***B.*** *If more than one valid EEPROM write occurs, the last address and data will be latched overriding the previous address and data. Once data is written to the desired address, do not read EEPROM-1 locations other than the written location. (Reading an EEPROM location returns the latched data and causes the read address to be latched).*

***C.*** *The EEPGM bit cannot be set if the EELAT bit is cleared or a non-valid EEPROM address is latched. This is to ensure proper programming sequence. Once EEPGM is set, do not read any EEPROM-1 locations; otherwise, the current program cycle will be unsuccessful. When EEPGM is set, the on-board programming sequence will be activated.*

***D.*** *The delay time for the EEPGM bit to be cleared in AUTO mode is less than $t_{EEPGM}$. However, on other MCUs, this delay time may be different. For forward compatibility, software should not make any dependency on this delay time.*

***E.*** *Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM-1 array.*

#### 6.4.5.3 EEPROM-1 Erasing

The programmed state of an EEPROM bit is logic 0. Erasing changes the state to a logic 1. Only EEPROM-1 bytes in the non-protected blocks and the EE1NVR register can be erased.

Use the following procedure to erase a byte, block or the entire EEPROM-1 array:

1. Configure EERAS1 and EERAS0 for byte, block or bulk erase; set EELAT in EE1CR.(A)

**NOTE**

*If using the AUTO mode, also set the AUTO bit in Step 1.*

2. Byte erase: write any data to the desired address.(B)
   Block erase: write any data to an address within the desired block.(B)
   Bulk erase: write any data to an address within the array.(B)
3. Set the EEPGM bit.(C) Go to Step 7 if AUTO is set.
4. Wait for a time: $t_{EEBYTE}$ for byte erase; $t_{EEBLOCK}$ for block erase; $t_{EEBULK.}$ for bulk erase.
5. Clear EEPGM bit.
6. Wait for a time, $t_{EEFPV}$, for the erasing voltage to fall. Go to Step 8.
7. Poll the EEPGM bit until it is cleared by the internal timer.(D)
8. Clear EELAT bits.(E)

**NOTE**

***A.*** *Setting the EELAT bit configures the address and data buses to latch data for erasing the array. Only valid EEPROM-1 addresses will be latched. If EELAT is set, other writes to the EE1CR will be allowed after a valid EEPROM-1 write.*

***B.*** *If more than one valid EEPROM write occurs, the last address and data will be latched overriding the previous address and data. Once data is written to the desired address, do not read EEPROM-1 locations other than the written location. (Reading an EEPROM location returns the latched data and causes the read address to be latched).*

***C.*** *The EEPGM bit cannot be set if the EELAT bit is cleared or a non-valid EEPROM address is latched. This is to ensure proper programming sequence. Once EEPGM is set, do not read any EEPROM-1 locations; otherwise, the current program cycle will be unsuccessful. When EEPGM is set, the on-board programming sequence will be activated.*

***D.*** *The delay time for the EEPGM bit to be cleared in AUTO mode is less than $t_{EEBYTE}$/$t_{EEBLOCK}$/$t_{EEBULK}$. However, on other MCUs, this delay time may be different. For forward compatibility, software should not make any dependency on this delay time.*

***E.*** *Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM-1 array.*

## 6.5 EEPROM-1 Register Descriptions

Four I/O registers and three nonvolatile registers control program, erase and options of the EEPROM-1 array.

### 6.5.1 EEPROM-1 Control Register

This read/write register controls programming/erasing of the array.

Address: $FE1D

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 6-2. EEPROM-1 Control Register (EE1CR)**

**Bit 7— Unused bit**

This read/write bit is software programmable but has no functionality.

**EEOFF — EEPROM-1 power down**

This read/write bit disables the EEPROM-1 module for lower power consumption. Any attempts to access the array will give unpredictable results. Reset clears this bit.

1 = Disable EEPROM-1 array
0 = Enable EEPROM-1 array

**EERAS1 and EERAS0 — Erase/Program Mode Select Bits**

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

**Table 6-3. EEPROM-1 Program/Erase Mode Select**

| EEBPx | EERAS1 | EERAS0 | MODE |
|---|---|---|---|
| 0 | 0 | 0 | Byte Program |
| 0 | 0 | 1 | Byte Erase |
| 0 | 1 | 0 | Block Erase |
| 0 | 1 | 1 | Bulk Erase |
| 1 | X | X | No Erase/Program |

X = don’t care

**EELAT — EEPROM-1 Latch Control**

This read/write bit latches the address and data buses for programming the EEPROM-1 array. EELAT cannot be cleared if EEPGM is still set. Reset clears this bit.

1 = Buses configured for EEPROM-1 programming or erase operation
0 = Buses configured for normal operation

**AUTO — Automatic Termination of Program/Erase Cycle**

When AUTO is set, EEPGM is cleared automatically after the program/erase cycle is terminated by the internal timer.

(See note D for 6.4.5.2 EEPROM-1 Programming, 6.4.5.3 EEPROM-1 Erasing, and 28.1.13 EEPROM Memory Characteristics)

1 = Automatic clear of EEPGM is enabled
0 = Automatic clear of EEPGM is disabled

**EEPGM — EEPROM-1 Program/Erase Enable**

This read/write bit enables the internal charge pump and applies the programming/erasing voltage to the EEPROM-1 array if the EELAT bit is set and a write to a valid EEPROM-1 location has occurred. Reset clears the EEPGM bit.

1 = EEPROM-1 programming/erasing power switched on
0 = EEPROM-1 programming/erasing power switched off

***NOTE***

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

## 6.5.2 EEPROM-1 Array Configuration Register

The EEPROM-1 array configuration register configures EEPROM-1 security and EEPROM-1 block protection.

This read-only register is loaded with the contents of the EEPROM-1 nonvolatile register (EE1NVR) after a reset.

Address: $FE1F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| Write: | | | | | | | | |
| Reset: | Contents of EE1NVR ($FE1C) | | | | | | | |

= Unimplemented

**Figure 6-3. EEPROM-1 Array Configuration Register (EE1ACR)**

**Bit 7:5 — Unused Bits**

These read/write bits are software programmable but have no functionality.

**EEPRTCT — EEPROM-1 Protection Bit**

The EEPRTCT bit is used to enable the security feature in the EEPROM (see EEPROM-1 Program/Erase Protection).

1 = EEPROM-1 security disabled
0 = EEPROM-1 security enabled

This feature is a write-once feature. Once the protection is enabled it may not be disabled.

### EEBP[3:0] — EEPROM-1 Block Protection Bits

These bits prevent blocks of EEPROM-1 array from being programmed or erased.

1 = EEPROM-1 array block is protected

0 = EEPROM-1 array block is unprotected

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

**Table 6-4. EEPROM-1 Block Protect and Security Summary**

| Address Range | EEBPx | EEPRTCT = 1 | EEPRTCT = 0 |
|---|---|---|---|
| $0800 - $087F | EEBP0 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP0 = 1 | Protected | Protected |
| $0880 - $08EF | EEBP1 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP1 = 1 | Protected | Protected |
| $08F0 - $08FF | EEBP1 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Secured<br>(No Programming or Erasing) |
| | EEBP1 = 1 | Protected | |
| $0900 - $097F | EEBP2 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP2 = 1 | Protected | Protected |
| $0980 - $09FF | EEBP3 = 0 | Byte Programming Available<br>Bulk, Block and Byte Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP3 = 1 | Protected | Protected |

### 6.5.3 EEPROM-1 Nonvolatile Register

The contents of this register is loaded into the EEPROM-1 array configuration register (EE1ACR) after a reset.

This register is erased and programmed in the same way as an EEPROM byte. (See 6.5.1 EEPROM-1 Control Register for individual bit descriptions).

Address: $FE1C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| Write: | | | | | | | | |
| Reset: | PV | | | | | | | |

PV = Programmed value or 1 in the erased state.

**Figure 6-4. EEPROM-1 Nonvolatile Register (EE1NVR)**

***NOTE***

*The EE1NVR will leave the factory programmed with $F0 such that the full array is available and unprotected.*

### 6.5.4 EEPROM-1 Timebase Divider Register

The 16-bit EEPROM-1 timebase divider register consists of two 8-bit registers: EE1DIVH and EE1DIVL. The 11-bit value in this register is used to configure the timebase divider circuit to obtain the 35 μs timebase for EEPROM-1 control.

These two read/write registers are respectively loaded with the contents of the EEPROM-1 timebase divider nonvolatile registers (EE1DIVHNVR and EE1DIVLNVR) after a reset.

Address: $FE1A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| Write: | | | | | | | | |
| Reset: | Contents of EE1DIVHNVR ($FE10), Bits [6:3] = 0 | | | | | | | |

= Unimplemented

**Figure 6-5. EE1DIV Divider High Register (EE1DIVH)**

Address: $FE1B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| Write: | | | | | | | | |
| Reset: | Contents of EE1DIVLNVR ($FE11) | | | | | | | |

**Figure 6-6. EE1DIV Divider Low Register (EE1DIVL)**

**EEDIVSECD — EEPROM-1 Divider Security Disable**

This bit enables/disables the security feature of the EE1DIV registers. When EE1DIV security feature is enabled, the state of the registers EE1DIVH and EE1DIVL are locked (including EEDIVSECD bit). The EE1DIVHNVR and EE1DIVLNVR nonvolatile memory registers are also protected from being erased/programmed.

1 = EE1DIV security feature disabled
0 = EE1DIV security feature enabled

**EEDIV[10:0] — EEPROM-1 timebase prescaler**

These prescaler bits store the value of EE1DIV which is used as the divisor to derive a timebase of 35μs from the selected reference clock source (CGMXCLK or bus block in the CONFIG-2 register) for the EEPROM-1 related internal timer and circuits. EEDIV[10:0] bits are readable at any time. They are writable when EELAT = 0 and EEDIVSECD = 1.

The EE1DIV value is calculated by the following formula:

$$EE1DIV = INT[Reference\ Frequency(Hz) \times 35 \times 10^{-6} + 0.5]$$

Where the result inside the bracket is rounded down to the nearest integer value

For example, if the reference frequency is 4.9152MHz, the EE1DIV value is 172

***NOTE***

*Programming/erasing the EEPROM with an improper EE1DIV value may result in data lost and reduce endurance of the EEPROM device.*

### 6.5.5 EEPROM-1 Timebase Divider Nonvolatile Register

The 16-bit EEPROM-1 timebase divider nonvolatile register consists of two 8-bit registers: EE1DIVHNVR and EE1DIVLNVR. The contents of these two registers are respectively loaded into the EEPROM-1 timebase divider registers, EE1DIVH and EE1DIVL, after a reset.

These two registers are erased and programmed in the same way as an EEPROM-1 byte.

Address: $FE10

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| Reset: | Unaffected by reset; $FF when blank | | | | | | | |

R = Reserved

**Figure 6-7. EEPROM-1 Divider Nonvolatile Register High (EE1DIVHNVR))**

Address: $FE11

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| Reset: | Unaffected by reset; $FF when blank | | | | | | | |

**Figure 6-8. EEPROM-1 Divider Nonvolatile Register Low (EE1DIVLNVR)**

These two registers are protected from erase and program operations if the EEDIVSECD is set to logic 1 in the EE1DIVH (see EEPROM-1 Timebase Divider Register) or programmed to a logic 1 in the EE1DIVHNVR.

***NOTE***

*Once EEDIVSECD in the EE1DIVHNVR is programmed to 0 and after a system reset, the EE1DIV security feature is permanently enabled because the EEDIVSECD bit in the EE1DIVH is always loaded with 0 thereafter. Once this security feature is armed, erase and program mode are disabled for EE1DIVHNVR and EE1DIVLNVR. Modifications to the EE1DIVH and EE1DIVL registers are also disabled. Therefore, care should be taken before programming a value into the EE1DIVHNVR.*

## 6.6 Low-Power Modes

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

### 6.6.1 Wait Mode

The WAIT instruction does not affect the EEPROM. It is possible to start the program or erase sequence on the EEPROM and put the MCU in wait mode.

### 6.6.2 Stop Mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while a programming or erasing sequence is in progress.

If stop mode is entered while EELAT and EEPGM are set, the programming sequence will be stopped and the programming voltage to the EEPROM array removed. The programming sequence will be restarted after leaving stop mode; access to the EEPROM is only possible after the programming sequence has completed.

If stop mode is entered while EELAT and EEPGM is cleared, the programming sequence will be terminated abruptly.

In either case, the data integrity of the EEPROM is not guaranteed.

# Chapter 7
# EEPROM-2 Memory

## 7.1 Introduction

This chapter describes the 512 bytes of electrically erasable programmable read-only memory (EEPROM) residing at address range $0600 to $07FF. There are 1024 bytes of EEPROM available on the MC68HC908AS60A and MC68HC908AZ60A which are physically located in two 512 byte arrays. For information relating to the array covering address range $0800 to $09FF please see Chapter 6 EEPROM-1 Memory.

## 7.2 Features

Features of the EEPROM-2 include the following:

- 512 bytes Nonvolatile Memory
- Byte, Block, or Bulk Erasable
- Nonvolatile EEPROM Configuration and Block Protection Options
- On-chip Charge Pump for Programming/Erasing
- Security Option
- AUTO Bit Driven Programming/Erasing Time Feature

## 7.3 EEPROM-2 Register Summary

The EEPROM-2 Register Summary is shown in Figure 7-1.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FF70 | EE2DIV Nonvolatile Register High (EE2DIVHNVR)* | Read: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset; $FF when blank | | | | | | | |
| $FF71 | EE2DIV Nonvolatile Register Low (EE2DIVLNVR)* | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset; $FF when blank | | | | | | | |
| $FF7A | EE2 Divider Register High (EE2DIVH) | Read: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| | | Reset: | Contents of EE2DIVHNVR ($FF70); Bits[6:3] = 0 | | | | | | | |
| $FF7B | EE2 Divider Register Low (EE2DIVL) | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| | | Reset: | Contents of EE2DIVLNVR ($FF71) | | | | | | | |
| $FF7C | EEPROM-2 Nonvolatile Register (EE2NVR)* | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| | | Reset: | Unaffected by reset; $FF when blank; factory programmed $F0 | | | | | | | |
| $FF7D | EEPROM-2 Control Register (EE2CR) | Read: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| | | Write: | | | | | | | | |
| | | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $FF7F | EEPROM-2 Array Configuration Register (EE2ACR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| | | Reset: | Contents of EE2NVR ($FF7C) | | | | | | | |

* Nonvolatile EEPROM register; write by programming.

= Unimplemented  R = Reserved  UNUSED = Unused

**Figure 7-1. EEPROM-2 Register Summary**

## 7.4 Functional Description

The 512 bytes of EEPROM-2 are located at $0600-$07FF and can be programmed or erased without an additional external high voltage supply. The program and erase operations are enabled through the use of an internal charge pump. For each byte of EEPROM, the write/erase endurance is 10,000 cycles.

### 7.4.1 EEPROM-2 Configuration

The 8-bit EEPROM-2 Nonvolatile Register (EE2NVR) and the 16-bit EEPROM-2 Timebase Divider Nonvolatile Register (EE2DIVNVR) contain the default settings for the following EEPROM configurations:

- EEPROM-2 Timebase Reference
- EEPROM-2 Security Option
- EEPROM-2 Block Protection

EE2NVR and EE2DIVNVR are nonvolatile EEPROM registers. They are programmed and erased in the same way as EEPROM bytes. The contents of these registers are loaded into their respective volatile registers during a MCU reset. The values in these read/write volatile registers define the EEPROM-2 configurations.

For EE2NVR, the corresponding volatile register is the EEPROM-2 Array Configuration Register (EE2ACR). For the EE2DIVNCR (two 8-bit registers: EE2DIVHNVR and EE2DIVLNVR), the corresponding volatile register is the EEPROM-2 Divider Register (EE2DIV: EE2DIVH and EE2 DIVL).

### 7.4.2 EEPROM-2 Timebase Requirements

A 35μs timebase is required by the EEPROM-2 control circuit for program and erase of EEPROM content. This timebase is derived from dividing the CGMXCLK or bus clock (selected by EEDIVCLK bit in CONFIG-2 Register) using a timebase divider circuit controlled by the 16-bit EEPROM-2 Timebase Divider EE2DIV Register (EE2DIVH and EE2DIVL).

As the CGMXCLK or bus clock is user selected, the EEPROM-2 Timebase Divider Register must be configured with the appropriate value to obtain the 35 μs. The timebase divider value is calculated by using the following formula:

$$EE2DIV = INT[\text{Reference Frequency(Hz)} \times 35 \times 10^{-6} + 0.5]$$

This value is written to the EEPROM-2 Timebase Divider Register (EE2DIVH and EE2DIVL) or programmed into the EEPROM-2 Timebase Divider Nonvolatile Register prior to any EEPROM program or erase operations (7.4.1 EEPROM-2 Configuration and 7.4.2 EEPROM-2 Timebase Requirements).

### 7.4.3 EEPROM-2 Program/Erase Protection

The EEPROM has a special feature that designates the 16 bytes of addresses from $06F0 to $06FF to be permanently secured. This program/erase protect option is enabled by programming the EEPRTCT bit in the EEPROM-2 Nonvolatile Register (EE2NVR) to a logic zero.

Once the EEPRTCT bit is programmed to 0 for the first time:

- Programming and erasing of secured locations $06F0 to $06FF is permanently disabled.
- Secured locations $06F0 to $06FF can be read as normal.
- Programming and erasing of EE2NVR is permanently disabled.
- Bulk and Block Erase operations are disabled for the unprotected locations $0600-$06EF, $0700-$07FF.
- Single byte program and erase operations are still available for locations $0600-$06EF and $0700-$07FF for all bytes that are not protected by the EEPROM-2 Block Protect EEBPx bits (see 7.4.4 EEPROM-2 Block Protection and 7.5.2 EEPROM-2 Array Configuration Register)

***NOTE***

*Once armed, the protect option is permanently enabled. As a consequence, all functions in the EE2NVR will remain in the state they were in immediately before the security was enabled.*

### 7.4.4 EEPROM-2 Block Protection

The 512 bytes of EEPROM-2 are divided into four 128-byte blocks. Each of these blocks can be protected from erase/program operations by setting the EEBPx bit in the EE2NVR. Table 7-1 shows the address ranges for the blocks.

**Table 7-1. EEPROM-2 Array Address Blocks**

| Block Number (EEBPx) | Address Range |
|---|---|
| EEBP0 | $0600–$067F |
| EEBP1 | $0680–$06FF |
| EEBP2 | $0700–$077F |
| EEBP3 | $0780–$07FF |

These bits are effective after a reset or a upon read of the EE2NVR register. The block protect configuration can be modified by erasing/programming the corresponding bits in the EE2NVR register and then reading the EE2NVR register. Please see 7.5.2 EEPROM-2 Array Configuration Register for more information.

***NOTE***

*Once EEDIVSECD in the EE2DIVHNVR is programmed to 0 and after a system reset, the EE2DIV security feature is permanently enabled because the EEDIVSECD bit in the EE2DIVH is always loaded with 0 thereafter. Once this security feature is armed, erase and program mode are disabled for EE2DIVHNVR and EE2DIVLNVR. Modifications to the EE2DIVH and EE2DIVL registers are also disabled. Therefore, be cautious on programming a value into the EE2DIVHNVR.*

### 7.4.5 EEPROM-2 Programming and Erasing

The unprogrammed or erase state of an EEPROM bit is a logic 1. The factory default for all bytes within the EEPROM-2 array is $FF.

The programming operation changes an EEPROM bit from logic 1 to logic 0 (programming cannot change a bit from logic 0 to a logic 1). In a single programming operation, the minimum EEPROM programming size is one bit; the maximum is eight bits (one byte).

The erase operation changes an EEPROM bit from logic 0 to logic 1. In a single erase operation, the minimum EEPROM erase size is one byte; the maximum is the entire EEPROM-2 array.

The EEPROM can be programmed such that one or multiple bits are programmed (written to a logic 0) at a time. However, the user may never program the same bit location more than once before erasing the entire byte. In other words, the user is not allowed to program a logic 0 to a bit that is already programmed (bit state is already logic 0).

For some applications it might be advantageous to track more than 10K events with a single byte of EEPROM by programming one bit at a time. For that purpose, a special selective bit programming technique is available. An example of this technique is illustrated in Table 7-2.

**Table 7-2. Example Selective Bit Programming Description**

| Description | Program Data in Binary | Result in Binary |
|---|---|---|
| Original state of byte (erased) | n/a | 1111:1111 |
| First event is recorded by programming bit position 0 | 1111:1110 | 1111:1110 |
| Second event is recorded by programming bit position 1 | 1111:1101 | 1111:1100 |
| Third event is recorded by programming bit position 2 | 1111:1011 | 1111:1000 |
| Fourth event is recorded by programming bit position 3 | 1111:0111 | 1111:0000 |
| Events five through eight are recorded in a similar fashion | | |

***NOTE***

*None of the bit locations are actually programmed more than once although the byte was programmed eight times.*

When this technique is utilized, a program/erase cycle is defined as multiple program sequences (up to eight) to a unique location followed by a single erase operation.

#### *7.4.5.1 Program/Erase Using AUTO Bit*

An additional feature available for EEPROM-2 program and erase operations is the AUTO mode. When enabled, AUTO mode will activate an internal timer that will automatically terminate the program/erase cycle and clear the EEPGM bit. Please see 7.4.5.2 EEPROM-2 Programming, 7.4.5.3 EEPROM-2 Erasing, and 7.5.1 EEPROM-2 Control Register for more information.

### 7.4.5.2 EEPROM-2 Programming

The unprogrammed or erase state of an EEPROM bit is a logic 1. Programming changes the state to a logic 0. Only EEPROM bytes in the non-protected blocks and the EE2NVR register can be programmed.

Use the following procedure to program a byte of EEPROM:

1. Clear EERAS1 and EERAS0 and set EELAT in the EE2CR.(A)

**NOTE**

*If using the AUTO mode, also set the AUTO bit during Step 1.*

2. Write the desired data to the desired EEPROM address.(B)
3. Set the EEPGM bit.(C) Go to Step 7 if AUTO is set.
4. Wait for time, $t_{EEPGM}$, to program the byte.
5. Clear EEPGM bit.
6. Wait for time, $t_{EEFPV}$, for the programming voltage to fall. Go to Step 8.
7. Poll the EEPGM bit until it is cleared by the internal timer.(D)
8. Clear EELAT bits.(E)

**NOTE**

***A.*** *EERAS1 and EERAS0 must be cleared for programming. Setting the EELAT bit configures the address and data buses to latch data for programming the array. Only data with a valid EEPROM-2 address will be latched. If EELAT is set, other writes to the EE2CR will be allowed after a valid EEPROM-2 write.*

***B.*** *If more than one valid EEPROM write occurs, the last address and data will be latched overriding the previous address and data. Once data is written to the desired address, do not read EEPROM-2 locations other than the written location. (Reading an EEPROM location returns the latched data and causes the read address to be latched).*

***C.*** *The EEPGM bit cannot be set if the EELAT bit is cleared or a non-valid EEPROM address is latched. This is to ensure proper programming sequence. Once EEPGM is set, do not read any EEPROM-2 locations; otherwise, the current program cycle will be unsuccessful. When EEPGM is set, the on-board programming sequence will be activated.*

***D.*** *The delay time for the EEPGM bit to be cleared in AUTO mode is less than $t_{EEPGM}$. However, on other MCUs, this delay time may be different. For forward compatibility, software should not make any dependency on this delay time.*

***E.*** *Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM-2 array.*

### 7.4.5.3 EEPROM-2 Erasing

The programmed state of an EEPROM bit is logic 0. Erasing changes the state to a logic 1. Only EEPROM-2 bytes in the non-protected blocks and the EE2NVR register can be erased.

Use the following procedure to erase a byte, block or the entire EEPROM-2 array:

1. Configure EERAS1 and EERAS0 for byte, block or bulk erase; set EELAT in EE2CR.(A)

**NOTE**

*If using the AUTO mode, also set the AUTO bit in Step 1.*

2. Byte erase: write any data to the desired address.(B)
   Block erase: write any data to an address within the desired block.(B)
   Bulk erase: write any data to an address within the array.(B)
3. Set the EEPGM bit.(C) Go to Step 7 if AUTO is set.
4. Wait for a time: $t_{EEBYTE}$ for byte erase; $t_{EEBLOCK}$ for block erase; $t_{EEBULK.}$ for bulk erase.
5. Clear EEPGM bit.
6. Wait for a time, $t_{EEFPV}$, for the erasing voltage to fall. Go to Step 8.
7. Poll the EEPGM bit until it is cleared by the internal timer.(D)
8. Clear EELAT bits.(E)

**NOTE**

***A.*** *Setting the EELAT bit configures the address and data buses to latch data for erasing the array. Only valid EEPROM-2 addresses will be latched. If EELAT is set, other writes to the EE2CR will be allowed after a valid EEPROM-2 write.*

***B.*** *If more than one valid EEPROM write occurs, the last address and data will be latched overriding the previous address and data. Once data is written to the desired address, do not read EEPROM-2 locations other than the written location. (Reading an EEPROM location returns the latched data and causes the read address to be latched).*

***C.*** *The EEPGM bit cannot be set if the EELAT bit is cleared or a non-valid EEPROM address is latched. This is to ensure proper programming sequence. Once EEPGM is set, do not read any EEPROM-2 locations; otherwise, the current program cycle will be unsuccessful. When EEPGM is set, the on-board programming sequence will be activated.*

***D.*** *The delay time for the EEPGM bit to be cleared in AUTO mode is less than $t_{EEBYTE}$/$t_{EEBLOCK}$/$t_{EEBULK}$. However, on other MCUs, this delay time may be different. For forward compatibility, software should not make any dependency on this delay time.*

***E.*** *Any attempt to clear both EEPGM and EELAT bits with a single instruction will only clear EEPGM. This is to allow time for removal of high voltage from the EEPROM-2 array.*

## 7.5 EEPROM-2 Register Descriptions

Four I/O registers and three nonvolatile registers control program, erase and options of the EEPROM-2 array.

### 7.5.1 EEPROM-2 Control Register

This read/write register controls programming/erasing of the array.

Address: $FF7D

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 7-2. EEPROM-2 Control Register (EE2CR)**

**Bit 7— Unused bit**

This read/write bit is software programmable but has no functionality.

**EEOFF — EEPROM-2 power down**

This read/write bit disables the EEPROM-2 module for lower power consumption. Any attempts to access the array will give unpredictable results. Reset clears this bit.

1 = Disable EEPROM-2 array
0 = Enable EEPROM-2 array

**EERAS1 and EERAS0 — Erase/Program Mode Select Bits**

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

**Table 7-3. EEPROM-2 Program/Erase Mode Select**

| EEBPx | EERAS1 | EERAS0 | MODE |
|---|---|---|---|
| 0 | 0 | 0 | Byte Program |
| 0 | 0 | 1 | Byte Erase |
| 0 | 1 | 0 | Block Erase |
| 0 | 1 | 1 | Bulk Erase |
| 1 | X | X | No Erase/Program |

X = don’t care

**EELAT — EEPROM-2 Latch Control**

This read/write bit latches the address and data buses for programming the EEPROM-2 array. EELAT cannot be cleared if EEPGM is still set. Reset clears this bit.

1 = Buses configured for EEPROM-2 programming or erase operation
0 = Buses configured for normal operation

**AUTO — Automatic Termination of Program/Erase Cycle**

When AUTO is set, EEPGM is cleared automatically after the program/erase cycle is terminated by the internal timer.

(See note D for 7.4.5.2 EEPROM-2 Programming, 7.4.5.3 EEPROM-2 Erasing, and 28.1.13 EEPROM Memory Characteristics)

1 = Automatic clear of EEPGM is enabled
0 = Automatic clear of EEPGM is disabled

**EEPGM — EEPROM-2 Program/Erase Enable**

This read/write bit enables the internal charge pump and applies the programming/erasing voltage to the EEPROM-2 array if the EELAT bit is set and a write to a valid EEPROM-2 location has occurred. Reset clears the EEPGM bit.

1 = EEPROM-2 programming/erasing power switched on
0 = EEPROM-2 programming/erasing power switched off

***NOTE***

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

## 7.5.2 EEPROM-2 Array Configuration Register

The EEPROM-2 array configuration register configures EEPROM-2 security and EEPROM-2 block protection.

This read-only register is loaded with the contents of the EEPROM-2 nonvolatile register (EE2NVR) after a reset.

Address: $FF7F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| Write: | | | | | | | | |
| Reset: | Contents of EE2NVR ($FF7C) | | | | | | | |

= Unimplemented

**Figure 7-3. EEPROM-2 Array Configuration Register (EE2ACR)**

**Bit 7:5 — Unused Bits**

These read/write bits are software programmable but have no functionality.

**EEPRTCT — EEPROM-2 Protection Bit**

The EEPRTCT bit is used to enable the security feature in the EEPROM (see EEPROM-2 Program/Erase Protection).

1 = EEPROM-2 security disabled
0 = EEPROM-2 security enabled

This feature is a write-once feature. Once the protection is enabled it may not be disabled.

### EEBP[3:0] — EEPROM-2 Block Protection Bits

These bits prevent blocks of EEPROM-2 array from being programmed or erased.

1 = EEPROM-2 array block is protected

0 = EEPROM-2 array block is unprotected

| Block Number (EEBPx) | Address Range |
|---|---|
| EEBP0 | $0600–$067F |
| EEBP1 | $0680–$06FF |
| EEBP2 | $0700–$077F |
| EEBP3 | $0780–$07FF |

**Table 7-4. EEPROM-2 Block Protect and Security Summary**

| Address Range | EEBPx | EEPRTCT = 1 | EEPRTCT = 0 |
|---|---|---|---|
| $0600 - $067F | EEBP0 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP0 = 1 | Protected | Protected |
| $0680 - $06EF | EEBP1 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP1 = 1 | Protected | Protected |
| $06F0 - $06FF | EEBP1 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Secured<br>(No Programming or Erasing) |
| | EEBP1 = 1 | Protected | |
| $0700 - $077F | EEBP2 = 0 | Byte Programming Available<br>Bulk, Block and Byte Erasing Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP2 = 1 | Protected | Protected |
| $0780 - $07FF | EEBP3 = 0 | Byte Programming Available<br>Bulk, Block and Byte Available | Byte Programming Available<br>Only Byte Erasing Available |
| | EEBP3 = 1 | Protected | Protected |

### 7.5.3 EEPROM-2 Nonvolatile Register

The contents of this register is loaded into the EEPROM-2 array configuration register (EE2ACR) after a reset.

This register is erased and programmed in the same way as an EEPROM byte. (See 7.5.1 EEPROM-2 Control Register for individual bit descriptions).

Address: $FF7C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| Write: | | | | | | | | |
| Reset: | PV | | | | | | | |

PV = Programmed value or 1 in the erased state.

**Figure 7-4. EEPROM-2 Nonvolatile Register (EE2NVR)**

***NOTE***

*The EE2NVR will leave the factory programmed with $F0 such that the full array is available and unprotected.*

### 7.5.4 EEPROM-2 Timebase Divider Register

The 16-bit EEPROM-2 timebase divider register consists of two 8-bit registers: EE2DIVH and EE2DIVL. The 11-bit value in this register is used to configure the timebase divider circuit to obtain the 35 µs timebase for EEPROM-2 control.

These two read/write registers are respectively loaded with the contents of the EEPROM-2 timebase divider nonvolatile registers (EE2DIVHNVR and EE2DIVLNVR) after a reset.

Address: $FF7A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| Write: | | | | | | | | |
| Reset: | Contents of EE2DIVHNVR ($FF70); Bits[6:3] = 0 | | | | | | | |

= Unimplemented

**Figure 7-5. EE2DIV Divider High Register (EE2DIVH)**

Address: $FF7B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| Write: | | | | | | | | |
| Reset: | Contents of EE2DIVLNVR ($FF71) | | | | | | | |

**Figure 7-6. EE2DIV Divider Low Register (EE2DIVL)**

**EEDIVSECD — EEPROM-2 Divider Security Disable**

This bit enables/disables the security feature of the EE2DIV registers. When EE2DIV security feature is enabled, the state of the registers EE2DIVH and EE2DIVL are locked (including EEDIVSECD bit). The EE2DIVHNVR and EE2DIVLNVR nonvolatile memory registers are also protected from being erased/programmed.

1 = EE2DIV security feature disabled
0 = EE2DIV security feature enabled

**EEDIV[10:0] — EEPROM-2 timebase prescaler**

These prescaler bits store the value of EE2DIV which is used as the divisor to derive a timebase of 35μs from the selected reference clock source (CGMXCLK or bus block in the CONFIG-2 register) for the EEPROM-2 related internal timer and circuits. EEDIV[10:0] bits are readable at any time. They are writable when EELAT = 0 and EEDIVSECD = 1.

The EE2DIV value is calculated by the following formula:

$$EE2DIV = INT[Reference\ Frequency(Hz) \times 35 \times 10^{-6} + 0.5]$$

Where the result inside the bracket is rounded down to the nearest integer value

For example, if the reference frequency is 4.9152MHz, the EE2DIV value is 172

***NOTE***

*Programming/erasing the EEPROM with an improper EE2DIV value may result in data lost and reduce endurance of the EEPROM device.*

## 7.5.5 EEPROM-2 Timebase Divider Nonvolatile Register

The 16-bit EEPROM-2 timebase divider nonvolatile register consists of two 8-bit registers: EE2DIVHNVR and EE2DIVLNVR. The contents of these two registers are respectively loaded into the EEPROM-2 timebase divider registers, EE2DIVH and EE2DIVL, after a reset.

These two registers are erased and programmed in the same way as an EEPROM-2 byte.

Address: $FF70

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| Reset: | Unaffected by reset; $FF when blank | | | | | | | |

R = Reserved

**Figure 7-7. EEPROM-2 Divider Nonvolatile Register High (EE2DIVHNVR))**

Address: $FF71

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| Reset: | Unaffected by reset; $FF when blank | | | | | | | |

**Figure 7-8. EEPROM-2 Divider Nonvolatile Register Low (EE2DIVLNVR)**

These two registers are protected from erase and program operations if the EEDIVSECD is set to logic 1 in EE2DIVH or programmed to a logic 1 in EE2DIVHNVR.

***NOTE***

*Once EEDIVSECD in the EE2DIVHNVR is programmed to 0 and after a system reset, the EE2DIV security feature is permanently enabled because the EEDIVSECD bit in the EE2DIVH is always loaded with 0 thereafter. Once this security feature is armed, erase and program mode are disabled for EE2DIVHNVR and EE2DIVLNVR. Modifications to the EE2DIVH and EE2DIVL registers are also disabled. Therefore, care should be taken before programming a value into the EE2DIVHNVR.*

## 7.6 Low-Power Modes

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

### 7.6.1 Wait Mode

The WAIT instruction does not affect the EEPROM. It is possible to start the program or erase sequence on the EEPROM and put the MCU in wait mode.

### 7.6.2 Stop Mode

The STOP instruction reduces the EEPROM power consumption to a minimum. The STOP instruction should not be executed while a programming or erasing sequence is in progress.

If stop mode is entered while EELAT and EEPGM are set, the programming sequence will be stopped and the programming voltage to the EEPROM array removed. The programming sequence will be restarted after leaving stop mode; access to the EEPROM is only possible after the programming sequence has completed.

If stop mode is entered while EELAT and EEPGM is cleared, the programming sequence will be terminated abruptly.

In either case, the data integrity of the EEPROM is not guaranteed.

# Chapter 8
# Central Processor Unit (CPU)

## 8.1 Introduction

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

## 8.2 Features

Features of the CPU include:

- Object code fully upward-compatible with M68HC05 Family
- 16-bit stack pointer with stack manipulation instructions
- 16-bit index register with x-register manipulation instructions
- 8-MHz CPU internal bus frequency
- 64-Kbyte program/data memory space
- 16 addressing modes
- Memory-to-memory data moves without using accumulator
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- Enhanced binary-coded decimal (BCD) data handling
- Modular architecture with expandable internal bus definition for extension of addressing range beyond 64 Kbytes
- Low-power stop and wait modes

## 8.3 CPU Registers

Figure 8-1 shows the five CPU registers. CPU registers are not part of the memory map.

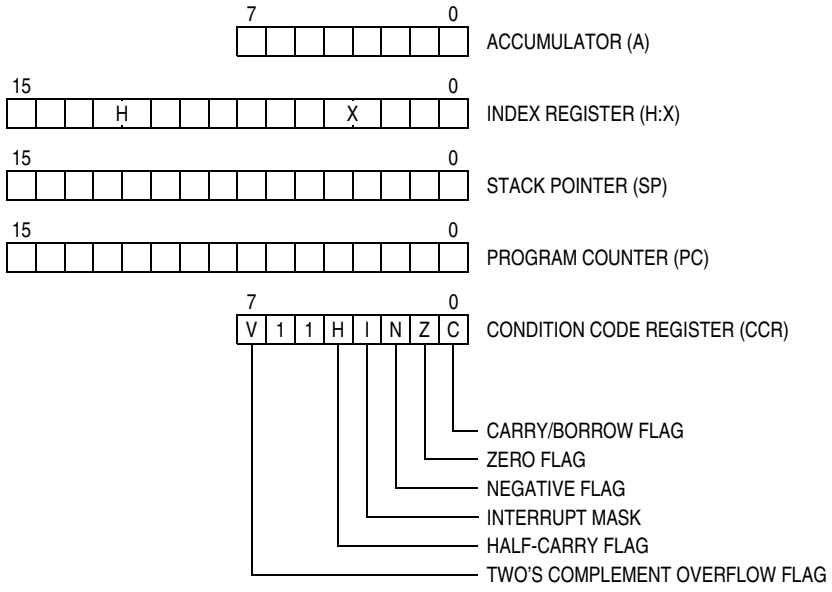



**Figure 8-1. CPU Registers**

### 8.3.1 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.

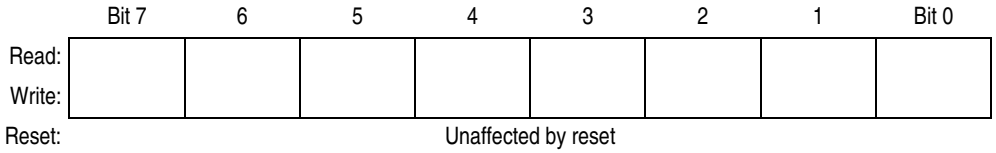



**Figure 8-2. Accumulator (A)**

### 8.3.2 Index Register

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

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

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

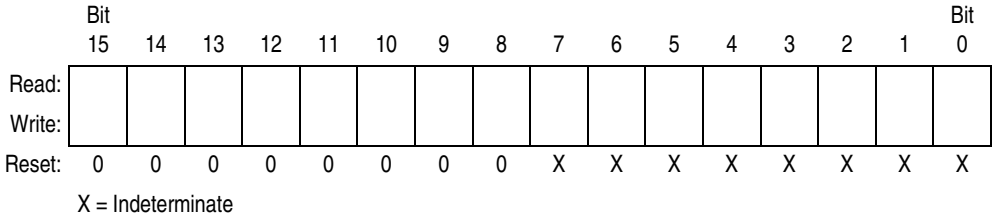



**Figure 8-3. Index Register (H:X)**

### 8.3.3 Stack Pointer

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

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

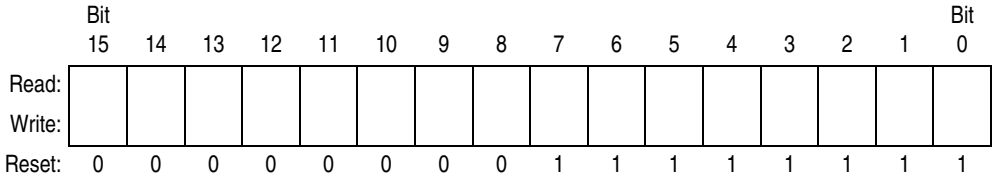



**Figure 8-4. Stack Pointer (SP)**

***NOTE***

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

### 8.3.4 Program Counter

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

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

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

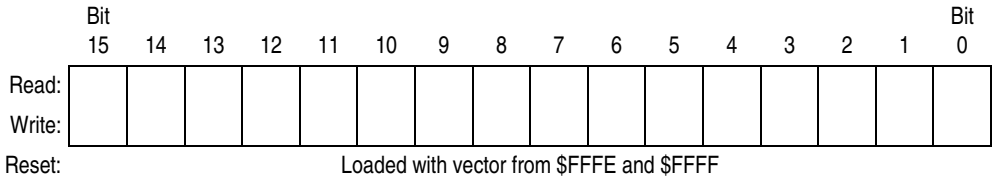



**Figure 8-5. Program Counter (PC)**

### 8.3.5 Condition Code Register

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

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

X = Indeterminate

**Figure 8-6. 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

## 8.4 Arithmetic/Logic Unit (ALU)

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

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

## 8.5 Low-Power Modes

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

### 8.5.1 Wait Mode

The 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

### 8.5.2 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.

## 8.6 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

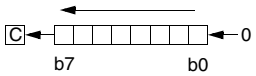
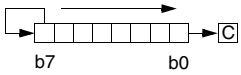
The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

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.

## 8.7 Instruction Set Summary

Table 8-1 provides a summary of the M68HC08 instruction set.

**Table 8-1. Instruction Set Summary (Sheet 1 of 6)**

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

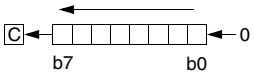
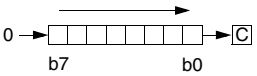
## Table 8-1. Instruction Set Summary (Sheet 2 of 6)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| BHS *rel* | Branch if Higher or Same (Same as BCC) | PC ← (PC) + 2 + *rel* ? (C) = 0 | – | – | – | – | – | – | REL | 24 | rr | 3 |
| BIH *rel* | Branch if $\overline{\text{IRQ}}$ Pin High | PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 1 | – | – | – | – | – | – | REL | 2F | rr | 3 |
| BIL *rel* | Branch if $\overline{\text{IRQ}}$ Pin Low | PC ← (PC) + 2 + *rel* ? $\overline{\text{IRQ}}$ = 0 | – | – | – | – | – | – | REL | 2E | rr | 3 |
| BIT #*opr*<br>BIT *opr*<br>BIT *opr*<br>BIT *opr*,X<br>BIT *opr*,X<br>BIT ,X<br>BIT *opr*,SP<br>BIT *opr*,SP | Bit Test | (A) & (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A5<br>B5<br>C5<br>D5<br>E5<br>F5<br>9EE5<br>9ED5 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>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)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 01<br>03<br>05<br>07<br>09<br>0B<br>0D<br>0F | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>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)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 00<br>02<br>04<br>06<br>08<br>0A<br>0C<br>0E | dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr<br>dd rr | 5<br>5<br>5<br>5<br>5<br>5<br>5<br>5 |
| BSET *n*,*opr* | Set Bit *n* in M | Mn ← 1 | – | – | – | – | – | – | DIR (b0)<br>DIR (b1)<br>DIR (b2)<br>DIR (b3)<br>DIR (b4)<br>DIR (b5)<br>DIR (b6)<br>DIR (b7) | 10<br>12<br>14<br>16<br>18<br>1A<br>1C<br>1E | dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd<br>dd | 4<br>4<br>4<br>4<br>4<br>4<br>4<br>4 |
| BSR *rel* | Branch to Subroutine | PC ← (PC) + 2; push (PCL)<br>SP ← (SP) – 1; push (PCH)<br>SP ← (SP) – 1<br>PC ← (PC) + *rel* | – | – | – | – | – | – | REL | AD | rr | 4 |
| CBEQ *opr*,*rel*<br>CBEQA #*opr*,*rel*<br>CBEQX #*opr*,*rel*<br>CBEQ *opr*,X+,*rel*<br>CBEQ X+,*rel*<br>CBEQ *opr*,SP,*rel* | Compare and Branch if Equal | PC ← (PC) + 3 + rel ? (A) – (M) = $00<br>PC ← (PC) + 3 + rel ? (A) – (M) = $00<br>PC ← (PC) + 3 + rel ? (X) – (M) = $00<br>PC ← (PC) + 3 + rel ? (A) – (M) = $00<br>PC ← (PC) + 2 + rel ? (A) – (M) = $00<br>PC ← (PC) + 4 + rel ? (A) – (M) = $00 | – | – | – | – | – | – | DIR<br>IMM<br>IMM<br>IX1+<br>IX+<br>SP1 | 31<br>41<br>51<br>61<br>71<br>9E61 | dd rr<br>ii rr<br>ii rr<br>ff rr<br>rr<br>ff rr | 5<br>4<br>4<br>5<br>4<br>6 |
| CLC | Clear Carry Bit | C ← 0 | – | – | – | – | – | 0 | INH | 98 | | 1 |
| CLI | Clear Interrupt Mask | I ← 0 | – | – | 0 | – | – | – | INH | 9A | | 2 |

## Table 8-1. Instruction Set Summary (Sheet 3 of 6)

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

## Table 8-1. Instruction Set Summary (Sheet 4 of 6)

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| JMP *opr*<br>JMP *opr*<br>JMP *opr*,X<br>JMP *opr*,X<br>JMP ,X | Jump | PC ← Jump Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BC<br>CC<br>DC<br>EC<br>FC | dd<br>hh ll<br>ee ff<br>ff | 2<br>3<br>4<br>3<br>2 |
| JSR *opr*<br>JSR *opr*<br>JSR *opr*,X<br>JSR *opr*,X<br>JSR ,X | Jump to Subroutine | PC ← (PC) + *n* (*n* = 1, 2, or 3)<br>Push (PCL); SP ← (SP) – 1<br>Push (PCH); SP ← (SP) – 1<br>PC ← Unconditional Address | – | – | – | – | – | – | DIR<br>EXT<br>IX2<br>IX1<br>IX | BD<br>CD<br>DD<br>ED<br>FD | dd<br>hh ll<br>ee ff<br>ff | 4<br>5<br>6<br>5<br>4 |
| LDA #*opr*<br>LDA *opr*<br>LDA *opr*<br>LDA *opr*,X<br>LDA *opr*,X<br>LDA ,X<br>LDA *opr*,SP<br>LDA *opr*,SP | Load A from M | A ← (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A6<br>B6<br>C6<br>D6<br>E6<br>F6<br>9EE6<br>9ED6 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| LDHX #*opr*<br>LDHX *opr* | Load H:X from M | H:X ← (M:M + 1) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR | 45<br>55 | ii jj<br>dd | 3<br>4 |
| LDX #*opr*<br>LDX *opr*<br>LDX *opr*<br>LDX *opr*,X<br>LDX *opr*,X<br>LDX ,X<br>LDX *opr*,SP<br>LDX *opr*,SP | Load X from M | X ← (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | AE<br>BE<br>CE<br>DE<br>EE<br>FE<br>9EEE<br>9EDE | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| LSL *opr*<br>LSLA<br>LSLX<br>LSL *opr*,X<br>LSL ,X<br>LSL *opr*,SP | Logical Shift Left<br>(Same as ASL) |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 38<br>48<br>58<br>68<br>78<br>9E68 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| LSR *opr*<br>LSRA<br>LSR*X*<br>LSR *opr*,X<br>LSR ,X<br>LSR *opr*,SP | Logical Shift Right |  | ↕ | – | – | 0 | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 34<br>44<br>54<br>64<br>74<br>9E64 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| MOV *opr*,*opr*<br>MOV *opr*,X+<br>MOV #*opr*,*opr*<br>MOV X+,*opr* | Move | $(M)_{Destination} \leftarrow (M)_{Source}$<br><br>H:X ← (H:X) + 1 (IX+D, DIX+) | 0 | – | – | ↕ | ↕ | – | DD<br>DIX+<br>IMD<br>IX+D | 4E<br>5E<br>6E<br>7E | dd dd<br>dd<br>ii dd<br>dd | 5<br>4<br>4<br>4 |
| MUL | Unsigned multiply | X:A ← (X) × (A) | – | 0 | – | – | – | 0 | INH | 42 | | 5 |
| NEG *opr*<br>NEGA<br>NEGX<br>NEG *opr*,X<br>NEG ,X<br>NEG *opr*,SP | Negate (Two's Complement) | M ← –(M) = $00 – (M)<br>A ← –(A) = $00 – (A)<br>X ← –(X) = $00 – (X)<br>M ← –(M) = $00 – (M)<br>M ← –(M) = $00 – (M) | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 30<br>40<br>50<br>60<br>70<br>9E60 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| NOP | No Operation | None | – | – | – | – | – | – | INH | 9D | | 1 |
| NSA | Nibble Swap A | A ← (A[3:0]:A[7:4]) | – | – | – | – | – | – | INH | 62 | | 3 |
| ORA #*opr*<br>ORA *opr*<br>ORA *opr*<br>ORA *opr*,X<br>ORA *opr*,X<br>ORA ,X<br>ORA *opr*,SP<br>ORA *opr*,SP | Inclusive OR A and M | A ← (A) \| (M) | 0 | – | – | ↕ | ↕ | – | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | AA<br>BA<br>CA<br>DA<br>EA<br>FA<br>9EEA<br>9EDA | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| PSHA | Push A onto Stack | Push (A); SP ← (SP) – 1 | – | – | – | – | – | – | INH | 87 | | 2 |
| PSHH | Push H onto Stack | Push (H); SP ← (SP) – 1 | – | – | – | – | – | – | INH | 8B | | 2 |
| PSHX | Push X onto Stack | Push (X); SP ← (SP) – 1 | – | – | – | – | – | – | INH | 89 | | 2 |

## Table 8-1. Instruction Set Summary (Sheet 5 of 6)

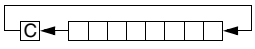
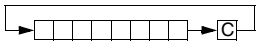
| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| PULA | Pull A from Stack | SP ← (SP + 1); Pull (A) | – | – | – | – | – | – | INH | 86 | | 2 |
| PULH | Pull H from Stack | SP ← (SP + 1); Pull (H) | – | – | – | – | – | – | INH | 8A | | 2 |
| PULX | Pull X from Stack | SP ← (SP + 1); Pull (X) | – | – | – | – | – | – | INH | 88 | | 2 |
| ROL *opr*<br>ROLA<br>ROLX<br>ROL *opr*,X<br>ROL ,X<br>ROL *opr*,SP | Rotate Left through Carry |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 39<br>49<br>59<br>69<br>79<br>9E69 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| ROR *opr*<br>RORA<br>RORX<br>ROR *opr*,X<br>ROR ,X<br>ROR *opr*,SP | Rotate Right through Carry |  | ↕ | – | – | ↕ | ↕ | ↕ | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 36<br>46<br>56<br>66<br>76<br>9E66 | dd<br><br><br>ff<br><br>ff | 4<br>1<br>1<br>4<br>3<br>5 |
| RSP | Reset Stack Pointer | SP ← $FF | – | – | – | – | – | – | INH | 9C | | 1 |
| RTI | Return from Interrupt | SP ← (SP) + 1; Pull (CCR)<br>SP ← (SP) + 1; Pull (A)<br>SP ← (SP) + 1; Pull (X)<br>SP ← (SP) + 1; Pull (PCH)<br>SP ← (SP) + 1; Pull (PCL) | ↕ | ↕ | ↕ | ↕ | ↕ | ↕ | INH | 80 | | 7 |
| RTS | Return from Subroutine | SP ← SP + 1; Pull (PCH)<br>SP ← SP + 1; Pull (PCL) | – | – | – | – | – | – | INH | 81 | | 4 |
| SBC #*opr*<br>SBC *opr*<br>SBC *opr*<br>SBC *opr*,X<br>SBC *opr*,X<br>SBC ,X<br>SBC *opr*,SP<br>SBC *opr*,SP | Subtract with Carry | A ← (A) – (M) – (C) | ↕ | – | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A2<br>B2<br>C2<br>D2<br>E2<br>F2<br>9EE2<br>9ED2 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SEC | Set Carry Bit | C ← 1 | – | – | – | – | – | 1 | INH | 99 | | 1 |
| SEI | Set Interrupt Mask | I ← 1 | – | – | 1 | – | – | – | INH | 9B | | 2 |
| STA *opr*<br>STA *opr*<br>STA *opr*,X<br>STA *opr*,X<br>STA ,X<br>STA *opr*,SP<br>STA *opr*,SP | Store A in M | M ← (A) | 0 | – | – | ↕ | ↕ | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | B7<br>C7<br>D7<br>E7<br>F7<br>9EE7<br>9ED7 | dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 3<br>4<br>4<br>3<br>2<br>4<br>5 |
| STHX *opr* | Store H:X in M | (M:M + 1) ← (H:X) | 0 | – | – | ↕ | ↕ | – | DIR | 35 | dd | 4 |
| STOP | Enable Interrupts, Stop Processing, Refer to MCU Documentation | I ← 0; Stop Processing | – | – | 0 | – | – | – | INH | 8E | | 1 |
| STX *opr*<br>STX *opr*<br>STX *opr*,X<br>STX *opr*,X<br>STX ,X<br>STX *opr*,SP<br>STX *opr*,SP | Store X in M | M ← (X) | 0 | – | – | ↕ | ↕ | – | DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | BF<br>CF<br>DF<br>EF<br>FF<br>9EEF<br>9EDF | dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SUB #*opr*<br>SUB *opr*<br>SUB *opr*<br>SUB *opr*,X<br>SUB *opr*,X<br>SUB ,X<br>SUB *opr*,SP<br>SUB *opr*,SP | Subtract | A ← (A) – (M) | ↕ | – | – | ↕ | ↕ | ↕ | IMM<br>DIR<br>EXT<br>IX2<br>IX1<br>IX<br>SP1<br>SP2 | A0<br>B0<br>C0<br>D0<br>E0<br>F0<br>9EE0<br>9ED0 | ii<br>dd<br>hh ll<br>ee ff<br>ff<br><br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |

**Table 8-1. Instruction Set Summary (Sheet 6 of 6)**

| Source Form | Operation | Description | Effect on CCR | | | | | | Address Mode | Opcode | Operand | Cycles |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | V | H | I | N | Z | C | | | | |
| SWI | Software Interrupt | PC ← (PC) + 1; Push (PCL)<br>SP ← (SP) – 1; Push (PCH)<br>SP ← (SP) – 1; Push (X)<br>SP ← (SP) – 1; Push (A)<br>SP ← (SP) – 1; Push (CCR)<br>SP ← (SP) – 1; I ← 1<br>PCH ← Interrupt Vector High Byte<br>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*<br>TSTA<br>TSTX<br>TST *opr*,X<br>TST ,X<br>TST *opr*,SP | Test for Negative or Zero | (A) – $00 or (X) – $00 or (M) – $00 | 0 | – | – | ↕ | ↕ | – | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1 | 3D<br>4D<br>5D<br>6D<br>7D<br>9E6D | dd<br><br><br>ff<br><br>ff | 3<br>1<br>1<br>3<br>2<br>4 |
| TSX | Transfer SP to H:X | H:X ← (SP) + 1 | – | – | – | – | – | – | INH | 95 | | 2 |
| TXA | Transfer X to A | A ← (X) | – | – | – | – | – | – | INH | 9F | | 1 |
| TXS | Transfer H:X to SP | (SP) ← (H:X) – 1 | – | – | – | – | – | – | INH | 94 | | 2 |
| WAIT | Enable Interrupts; Wait for Interrupt | I bit ← 0; Inhibit CPU clocking until interrupted | – | – | 0 | – | – | – | INH | 8F | | 1 |

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

## 8.8 Opcode Map

See Table 8-2.

## Table 8-2. 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 |
|---|---|
| 0 | 5 BRSET0 3 DIR |

Low Byte of Opcode in Hexadecimal

High Byte of Opcode in Hexadecimal
Cycles
Opcode Mnemonic
Number of Bytes / Addressing Mode

# Chapter 9
# System Integration Module (SIM)

## 9.1 Introduction

This chapter describes the system integration module (SIM), which supports up to 32 external and/or internal interrupts. Together with the central processor unit (CPU), the SIM controls all MCU activities. A block diagram of the SIM is shown in Figure 9-1. Figure 9-2 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 computer operating properly (COP) timeout
- Interrupt control:
  - Acknowledge timing
  - Arbitration control timing
  - Vector address generation
- CPU enable/disable timing

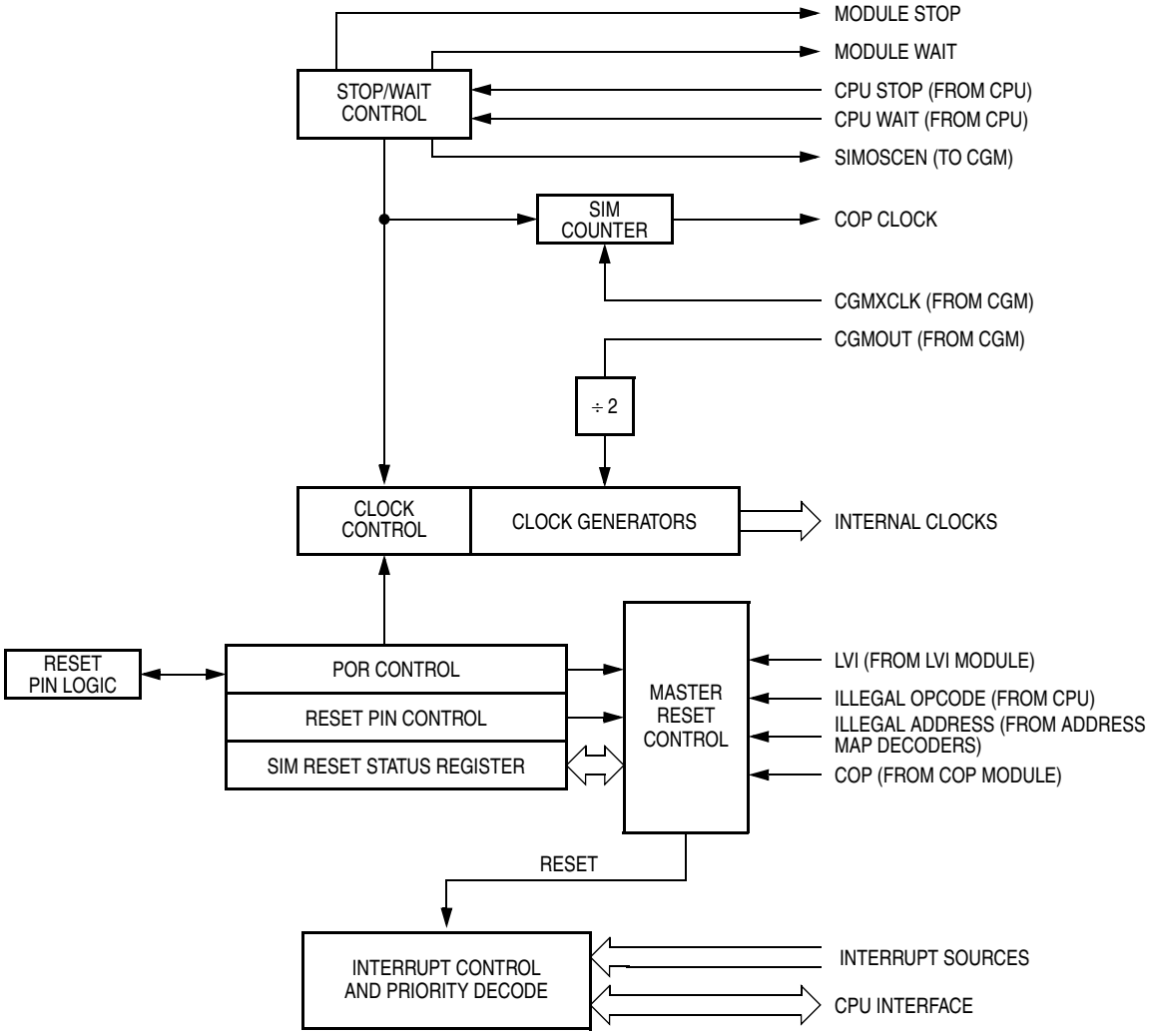



**Figure 9-1. SIM Block Diagram**

| Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| SIM Break Status Register (SBSR) | R | R | R | R | R | R | BW | R |
| SIM Reset Status Register (SRSR) | POR | PIN | COP | ILOP | ILAD | 0 | LVI | 0 |
| SIM Break Flag Control Register (SBFCR) | BCFE | R | R | R | R | R | R | R |

R = Reserved

**Figure 9-2. SIM I/O Register Summary**

**Table 9-1. I/O Register Address Summary**

| Register | SBSR | SRSR | SBFCR |
|---|---|---|---|
| Address | $FE00 | $FE01 | $FE03 |

Table 9-2 shows the internal signal names used in this chapter.

**Table 9-2. Signal Name Conventions**

| Signal Name | Description |
|---|---|
| CGMXCLK | Buffered Version of OSC1 from Clock Generator Module (CGM) |
| CGMVCLK | PLL Output |
| CGMOUT | PLL-Based or OSC1-Based Clock Output from CGM Module (Bus Clock = CGMOUT Divided by Two) |
| IAB | Internal Address Bus |
| IDB | Internal Data Bus |
| PORRST | Signal from the Power-On Reset Module to the SIM |
| IRST | Internal Reset Signal |
| R/$\overline{W}$ | Read/Write Signal |

## 9.2 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 9-3. This clock can come from either an external oscillator or from the on-chip PLL. (See Chapter 10 Clock Generator Module (CGM)).

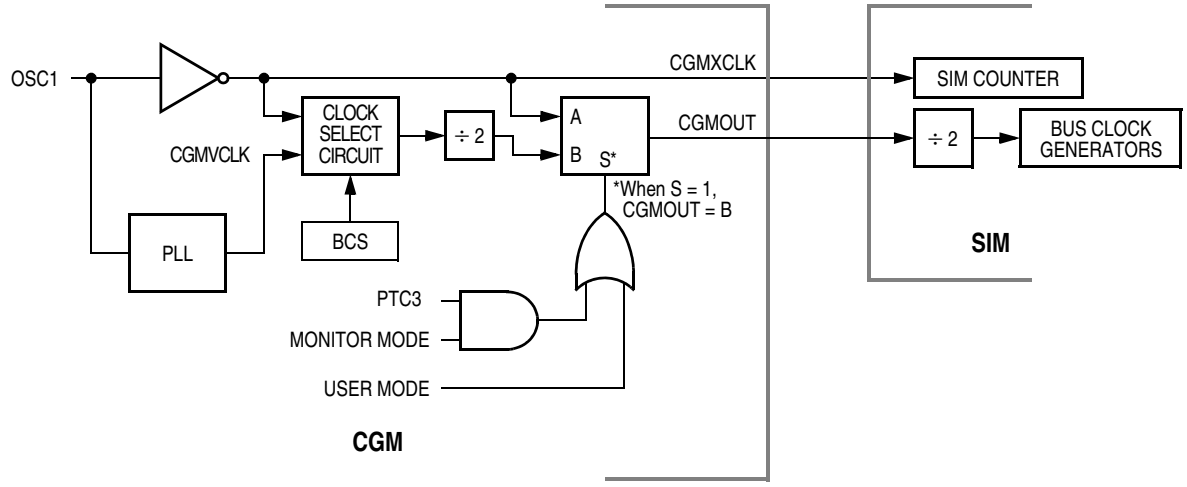



**Figure 9-3. CGM Clock Signals**

### 9.2.1 Bus Timing

In user mode, the internal bus frequency is either the crystal oscillator output (CGMXCLK) divided by four or the PLL output (CGMVCLK) divided by four. (See Chapter 10 Clock Generator Module (CGM)).

### 9.2.2 Clock Startup 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 4096 CGMXCLK cycles. The $\overline{RST}$ pin is driven low by the SIM during this entire period. The bus clocks start upon completion of the timeout.

### 9.2.3 Clocks in Stop Mode and Wait Mode

Upon exit from stop mode by an interrupt, break, or reset, the SIM allows CGMXCLK to clock the SIM counter. The CPU and peripheral clocks do not become active until after the stop delay timeout. This timeout is selectable as 4096 or 32 CGMXCLK cycles. See 9.6.2 Stop Mode.

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.

## 9.3 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 9.4 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR) (see 9.7 SIM Registers).

### 9.3.1 External Pin Reset

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 $t_{RL}$. Figure 9-4 shows the relative timing.

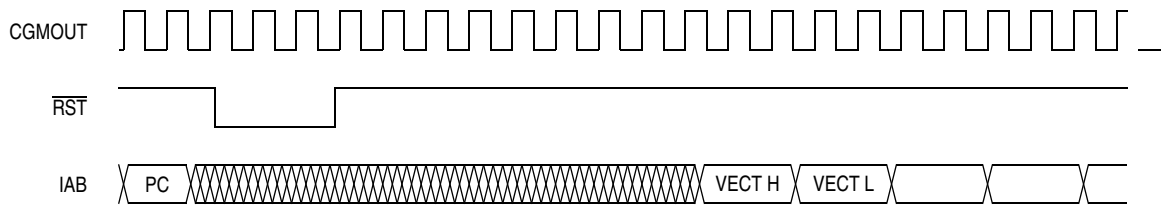



**Figure 9-4. External Reset Timing**

### 9.3.2 Active Resets from Internal Sources

All internal reset sources actively pull the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see Figure 9-5). An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR (see Figure 9-6). Note that for LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the $\overline{\text{RST}}$ pin low. The internal reset signal then follows the sequence from the falling edge of $\overline{\text{RST}}$ shown in Figure 9-5.

The COP reset is asynchronous to the bus clock.

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

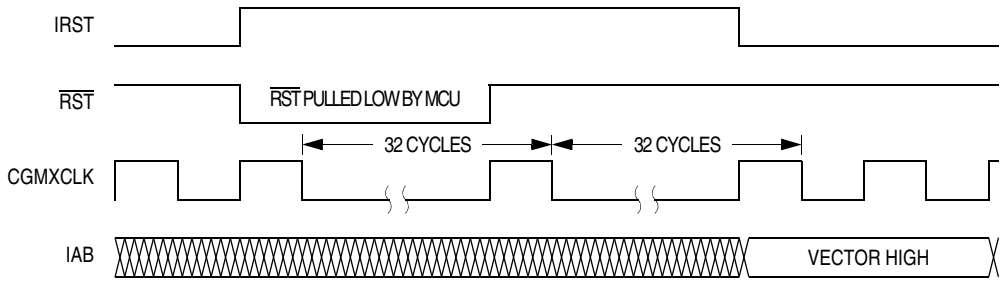



**Figure 9-5. Internal Reset Timing**

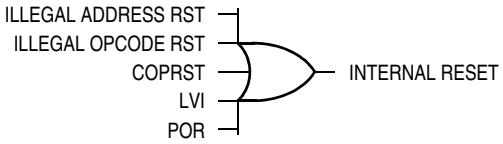



**Figure 9-6. Sources of Internal Reset**

**Table 9-3. PIN Bit Set Timing**

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

### 9.3.2.1 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 CGMXCLK cycles. Another sixty-four CGMXCLK cycles later, the CPU and memories are released from reset to allow the reset vector sequence to occur.

At power-on, the following events occur:

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

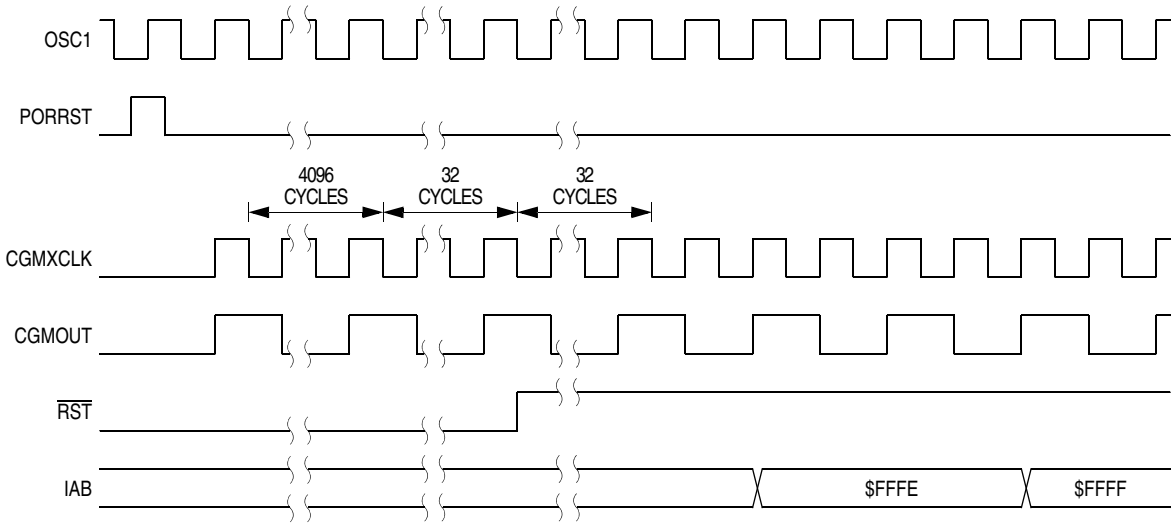



**Figure 9-7. POR Recovery**

### 9.3.2.2 Computer Operating Properly (COP) Reset

The overflow of the COP counter causes an internal reset and sets the COP bit in the SIM reset status register (SRSR) if the COPD bit in the CONFIG-1 register is at logic zero. See Chapter 15 Computer Operating Properly (COP).

### 9.3.2.3 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 CONFIG-1 register is logic zero, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset.

#### 9.3.2.4 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.

**WARNING**

***Extra care should be exercised if code in this part has been migrated from older HC08 devices since the illegal address reset specification may be different. Also, extra care should be exercised when using this emulation part for development of code to be run in ROM AZ, AB or AS family parts with a smaller memory size since some legal addresses will become illegal addresses on the smaller ROM memory map device and may as a result generate unwanted resets.***

#### 9.3.2.5 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 $V_{LVII}$ voltage. The LVI bit in the SIM reset status register (SRSR) is set and a chip reset is asserted if the LVIPWRD and LVIRSTD bits in the CONFIG-1 register are at logic zero. The $\overline{RST}$ pin will be held low until the SIM counts 4096 CGMXCLK cycles after $V_{DD}$ rises above $V_{LVIR}$. Another sixty-four CGMXCLK cycles later, the CPU is released from reset to allow the reset vector sequence to occur. See Chapter 16 Low-Voltage Inhibit (LVI).

## 9.4 SIM Counter

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

### 9.4.1 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.

### 9.4.2 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 or reset, the SIM senses the state of the short stop recovery bit, SSREC, in the CONFIG-1 register. If the SSREC bit is a logic one, 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.

### 9.4.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. See 9.6.2 Stop Mode for details. The SIM counter is free-running after all reset states. See 9.3.2 Active Resets from Internal Sources for counter control and internal reset recovery sequences.

## 9.5 Program 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

### 9.5.1 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 9-8 shows interrupt entry timing. Figure 9-10 shows interrupt recovery timing.

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

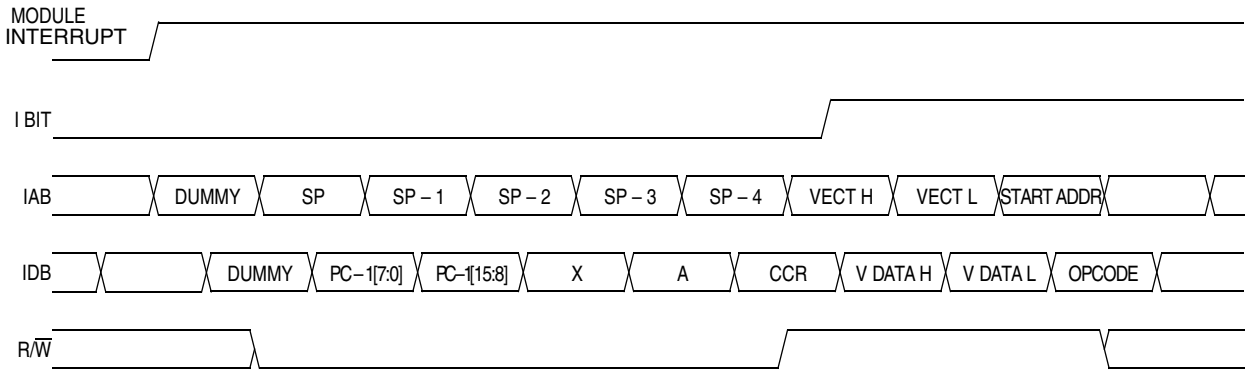



**Figure 9-8. Interrupt Entry**

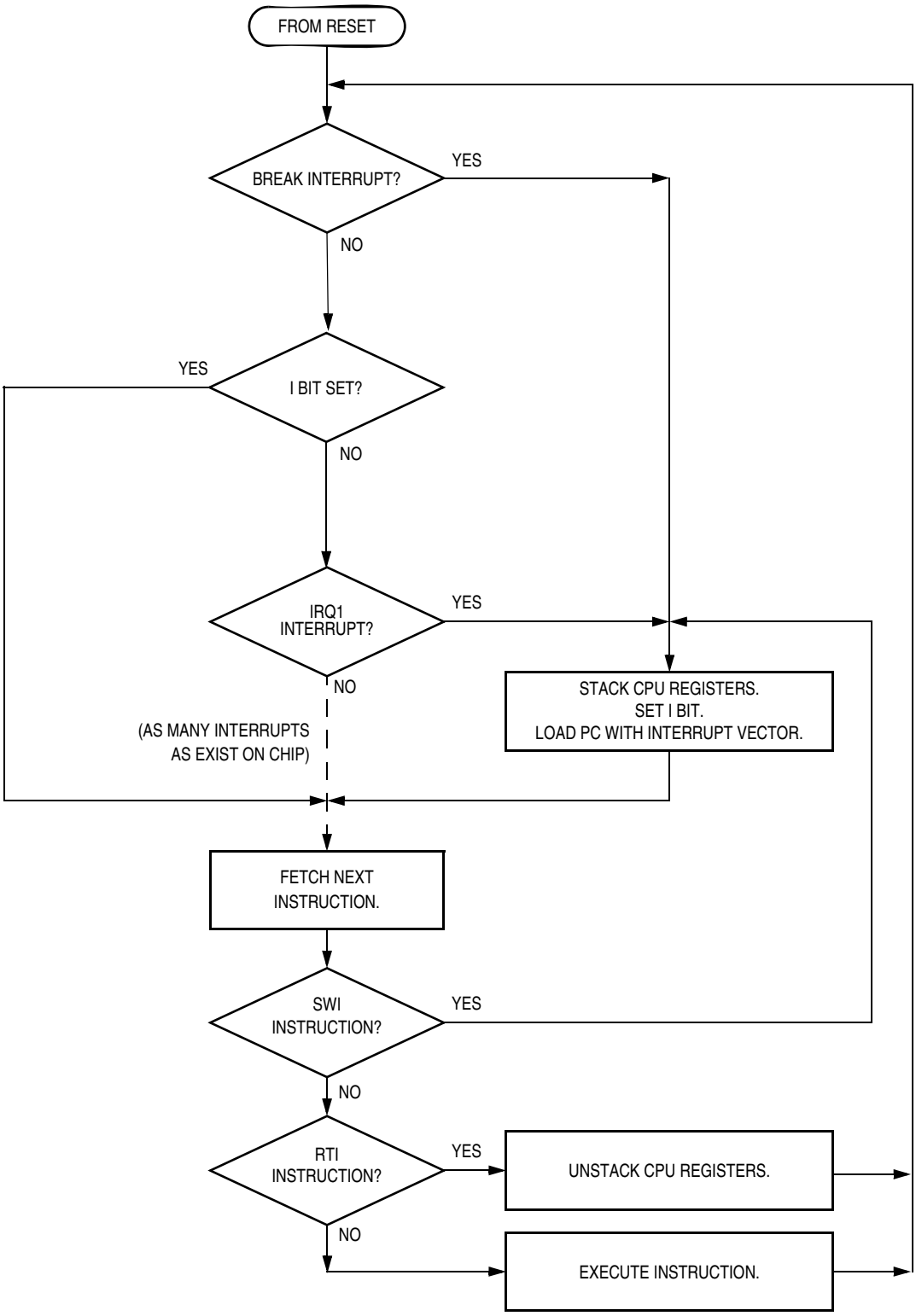



**Figure 9-9. Interrupt Processing**

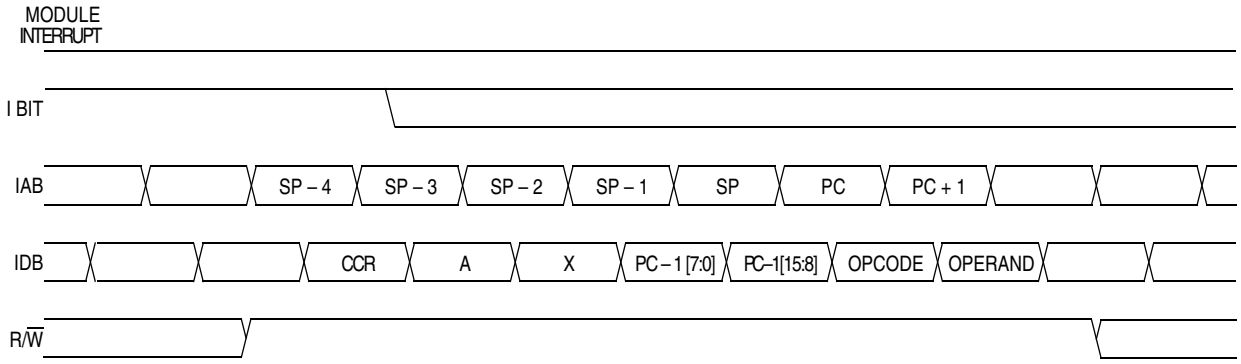



**Figure 9-10. Interrupt Recovery**

### *9.5.1.1 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 9-11 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.

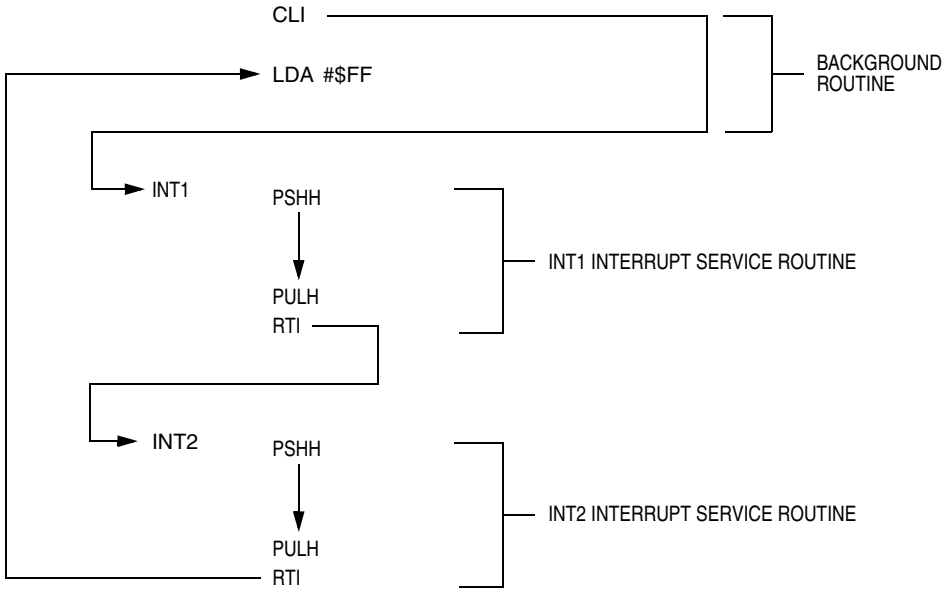



**Figure 9-11. Interrupt Recognition Example**

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

#### *9.5.1.2 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.*

### 9.5.2 Reset

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

### 9.5.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See Chapter 13 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.

### 9.5.4 Status Flag Protection in Break Mode

The SIM controls whether status flags contained in other modules can be cleared during break mode. The user can select 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 two-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.

## 9.6 Low-Power Modes

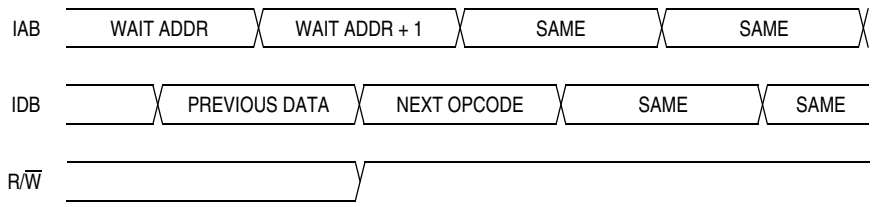
Executing the WAIT or STOP instruction puts the MCU in a low power- consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described below. Both STOP and WAIT clear the interrupt mask (I) in the condition code register, allowing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

### 9.6.1 Wait Mode

In wait mode, the CPU clocks are inactive while one set of peripheral clocks continue to run. Figure 9-12 shows the timing for wait mode entry.

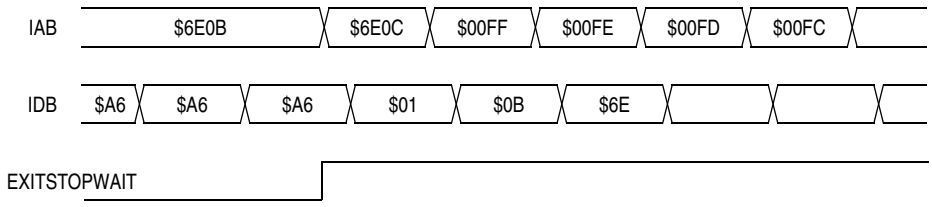
A module that is active during wait mode can wake up the CPU with an interrupt if the interrupt is enabled. Stacking for the interrupt begins one cycle after the WAIT instruction during which the interrupt occurred. Refer to the wait mode subsection of each module to see if the module is active or inactive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

Wait mode can also be exited by a reset or break. A break interrupt during wait mode sets the SIM break wait bit, BW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the configuration 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 9-12. Wait Mode Entry Timing**




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

**Figure 9-13. Wait Recovery from Interrupt or Break**

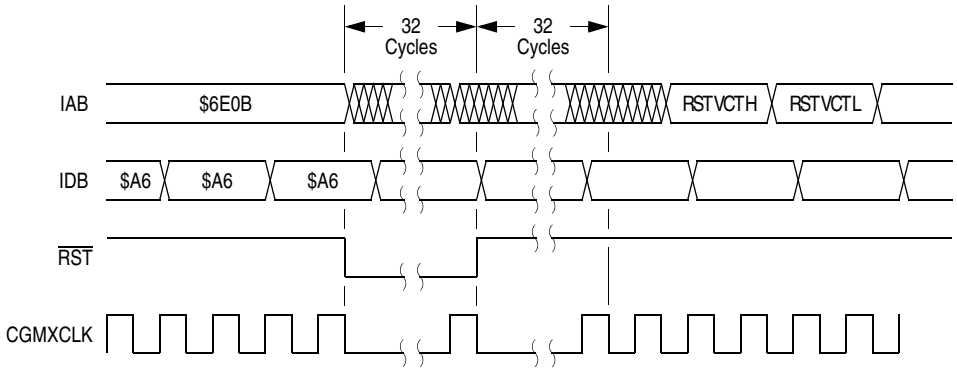



**Figure 9-14. Wait Recovery from Internal Reset**

### 9.6.2 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 also causes an exit from stop mode.

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

***NOTE***

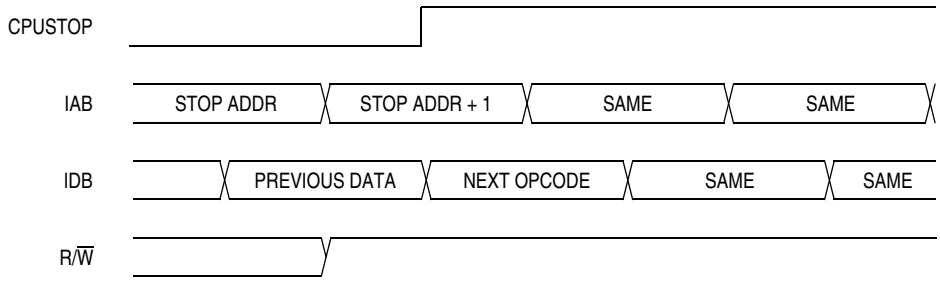
*External crystal applications should use the full stop recovery time by clearing the SSREC bit.*

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

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 9-15 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 9-15. Stop Mode Entry Timing**

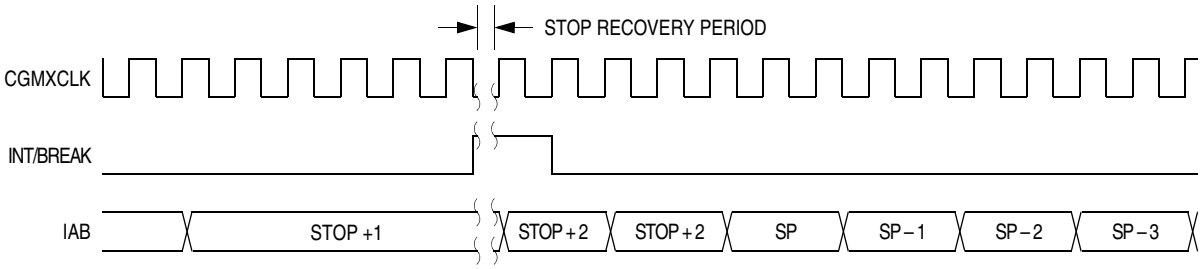



**Figure 9-16. Stop Mode Recovery from Interrupt or Break**

## 9.7 SIM Registers

The SIM has three memory mapped registers.

### 9.7.1 SIM Break Status Register

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

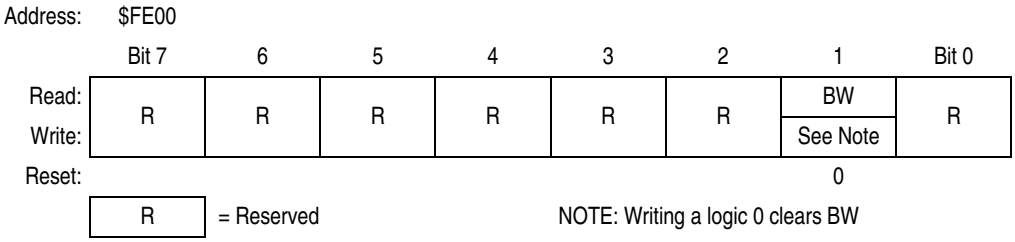



**Figure 9-17. SIM Break Status Register (SBSR)**

**BW — SIM Break Wait**

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

1 = Wait mode was exited by break interrupt
0 = Wait mode was not exited by break interrupt

### 9.7.2 SIM Reset Status Register

The SRSR register contains flags that show the source of the last reset. The status register will automatically clear after reading it. A power-on reset sets the POR bit and clears all other bits in the register. All other reset sources set the individual flag bits but do not clear the register. More than one reset source can be flagged at any time depending on the conditions at the time of the internal or external reset. For example, the POR and LVI bits can both be set if the power supply has a slow rise time.

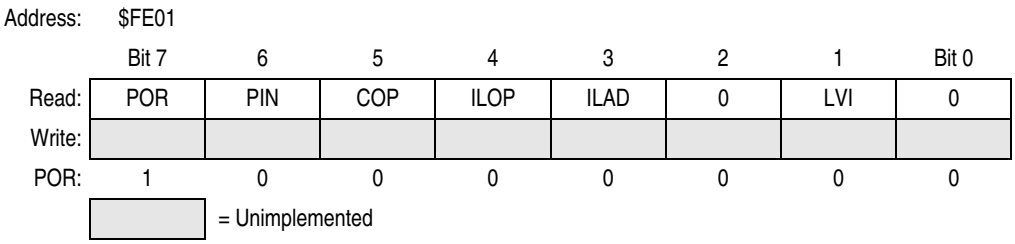



**Figure 9-18. SIM Reset Status Register (SRSR)**

**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

**LVI — Low-Voltage Inhibit Reset Bit**

1 = Last reset was caused by the LVI circuit
0 = POR or read of SRSR

### 9.7.3 SIM Break Flag Control Register

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

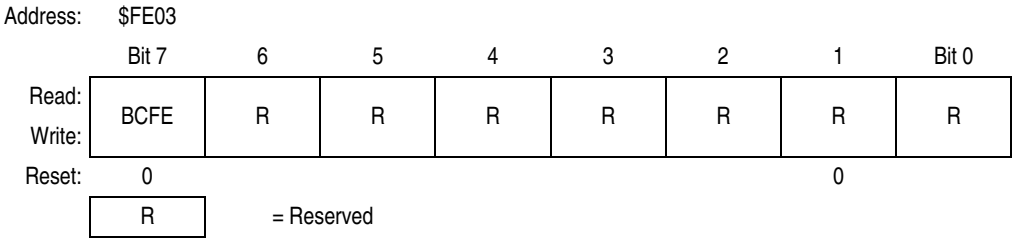



**Figure 9-19. SIM Break Flag Control Register (SBFCR)**

**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 10
# Clock Generator Module (CGM)

## 10.1 Introduction

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

## 10.2 Features

Features of the CGM include:

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

## 10.3 Functional Description

The CGM consists of three major submodules:

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

Figure 10-1 shows the structure of the CGM.

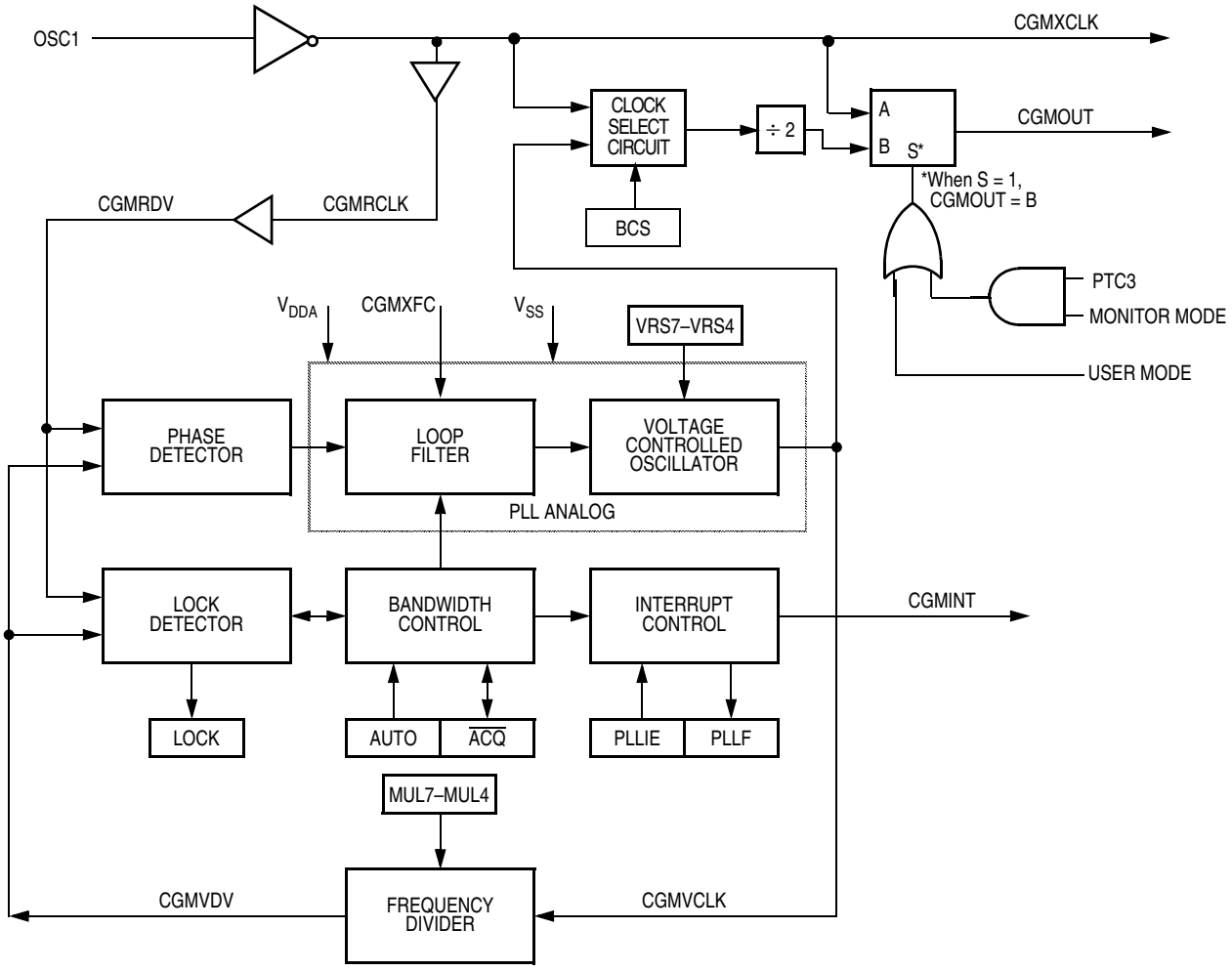



**Figure 10-1. CGM Block Diagram**

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

= Unimplemented

**Figure 10-2. I/O Register Summary**

**Table 10-1. I/O Register Address Summary**

| **Register** | PCTL | PBWC | PPG |
|---|---|---|---|
| **Address** | $001C | $001D | $001E |

### 10.3.1 Crystal Oscillator Circuit

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

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

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

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

### 10.3.2 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.

### 10.3.2.1 Circuits

The PLL consists of these circuits:

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

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

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

The VCO's output clock, CGMVCLK, running at a frequency $f_{CGMVCLK}$, is fed back through a programmable modulo divider. The modulo divider reduces the VCO clock by a factor, N. The divider's output is the VCO feedback clock, CGMVDV, running at a frequency $f_{CGMVDV} = f_{CGMVCLK}/N$. 10.3.2.4 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, as described in 10.3.2.2 Acquisition and Tracking Modes. The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

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

### 10.3.2.2 Acquisition and Tracking Modes

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

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL startup or when the PLL has suffered a severe noise hit and the VCO frequency is far off the desired frequency. When in acquisition mode, the $\overline{ACQ}$ bit is clear in the PLL bandwidth control register. See 10.5.2 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 10.3.3 Base Clock Selector Circuit. The PLL is automatically in tracking mode when it's not in acquisition mode or when the $\overline{ACQ}$ bit is set.

#### 10.3.2.3 Manual and Automatic PLL Bandwidth Modes

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

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. See 10.5.2 PLL Bandwidth Control Register. If PLL CPU interrupt requests are enabled, the software can wait for a PLL CPU interrupt request and then check the LOCK bit. If CPU interrupts are disabled, software can poll the LOCK bit continuously (during PLL startup, usually) or at periodic intervals. In either case, when the LOCK bit is set, the VCO clock is safe to use as the source for the base clock. See 10.3.3 Base Clock Selector Circuit. If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. See 10.6 Interrupts.

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

- The $\overline{ACQ}$ bit (See 10.5.2 PLL Bandwidth Control Register.) is a read-only indicator of the mode of the filter. See 10.3.2.2 Acquisition and Tracking Modes.
- The $\overline{ACQ}$ bit is set when the VCO frequency is within a certain tolerance, $\Delta_{trk}$, and is cleared when the VCO frequency is out of a certain tolerance, $\Delta_{unt}$. See Chapter 28 Electrical Specifications.
- 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, $\Delta_{Lock}$, and is cleared when the VCO frequency is out of a certain tolerance, $\Delta_{unl}$. See Chapter 28 Electrical Specifications.
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. See 10.5.1 PLL Control Register.

The PLL also can operate in manual mode (AUTO = 0). Manual mode is used by systems that do not require an indicator of the lock condition for proper operation. Such systems typically operate well below $f_{busmax}$ and require fast startup. 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 Chapter 28 Electrical 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.

### 10.3.2.4 Programming the PLL

Use this 9-step procedure to program the PLL. Table 10-2 lists the variables used and their meaning (Please also reference Figure 10-1).

**Table 10-2. Variable Definitions**

| Variable | Definition |
|---|---|
| $f_{BUSDES}$ | Desired Bus Clock Frequency |
| $f_{VCLKDES}$ | Desired VCO Clock Frequency |
| $f_{CGMRCLK}$ | Chosen Reference Crystal Frequency |
| $f_{CGMVCLK}$ | Calculated VCO Clock Frequency |
| $f_{BUS}$ | Calculated Bus Clock Frequency |
| $f_{NOM}$ | Nominal VCO Center Frequency |
| $f_{CGMVRS}$ | Shifted VCO Center Frequency |

1. Choose the desired bus frequency, $f_{BUSDES}$.
   Example: $f_{BUSDES}$ = 8 MHz
2. Calculate the desired VCO frequency, $f_{VCLKDES}$.
   Example: $f_{VCLKDES} = 4 \times f_{BUSDES}$
   $f_{VCLKDES} = 4 \times 8$ MHz = 32 MHz
3. Using a reference frequency, $f_{RCLK}$, equal to the crystal frequency, calculate the VCO frequency multiplier, N. Round the result to the nearest integer.

$$N = \frac{f_{VCLKDES}}{f_{CGMRCLK}}$$

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

4. Calculate the VCO frequency, $f_{CGMVCLK}$.

$$f_{CGMVCLK} = N \times f_{CGMRCLK}$$

   Example: $f_{CGMVCLK} = 8 \times 4$ MHz = 32 MHz
5. Calculate the bus frequency, $f_{BUS}$, and compare $f_{BUS}$ with $f_{BUSDES}$.

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

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

6. If the calculated $f_{bus}$ is not within the tolerance limits of your application, select another $f_{BUSDES}$ or another $f_{RCLK}$.

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

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

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

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

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

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

   ***NOTE***

   *For proper operation,.*

$$|f_{CGMVRS} - f_{CGMVCLK}| \leq \frac{f_{NOM}}{2}$$

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

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

#### *10.3.2.5 Special Programming Exceptions*

The programming method described in 10.3.2.4 Programming the PLL, does not account for two possible exceptions. A value of 0 for N or L is meaningless when used in the equations given. To account for these exceptions:

- A 0 value for N is interpreted 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 10.3.3 Base Clock Selector Circuit.

### 10.3.3 Base Clock Selector Circuit

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

The BCS bit in the PLL control register (PCTL) selects which clock drives CGMOUT. The VCO clock cannot be selected as the base clock source if the PLL is not turned on. The PLL cannot be turned off if the VCO clock is selected. The PLL cannot be turned on or off simultaneously with the selection or deselection of the VCO clock. The VCO clock also cannot be selected as the base clock source if the

factor L is programmed to a 0. This value would set up a condition inconsistent with the operation of the PLL, so that the PLL would be disabled and the crystal clock would be forced as the source of the base clock.

### 10.3.4 CGM External Connections

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

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in Figure 10-3. Figure 10-3 shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

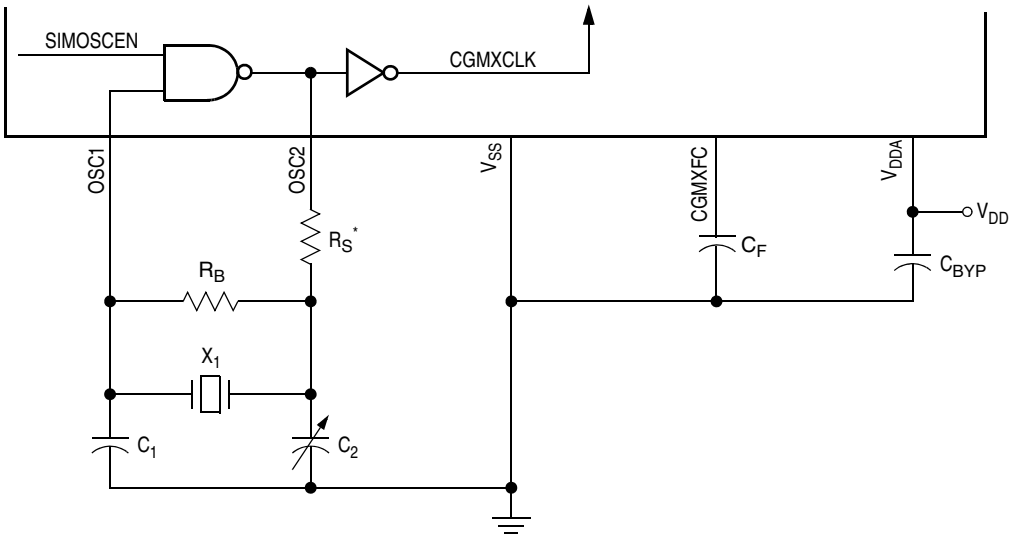
- Crystal, $X_1$
- Fixed capacitor, $C_1$
- Tuning capacitor, $C_2$ (can also be a fixed capacitor)
- Feedback resistor, $R_B$
- Series resistor, $R_S$ (optional)

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

Figure 10-3 also shows the external components for the PLL:

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

Routing should be done with great care to minimize signal cross talk and noise. (See 10.9 Acquisition/Lock Time Specifications for routing information and more information on the filter capacitor's value and its effects on PLL performance).




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

**Figure 10-3. CGM External Connections**

## 10.4 I/O Signals

The following paragraphs describe the CGM input/output (I/O) signals.

### 10.4.1 Crystal Amplifier Input Pin (OSC1)

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

### 10.4.2 Crystal Amplifier Output Pin (OSC2)

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

### 10.4.3 External Filter Capacitor Pin (CGMXFC)

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

***NOTE***

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

### 10.4.4 Analog Power Pin ($V_{DDA}$)

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

***NOTE***

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

### 10.4.5 Oscillator Enable Signal (SIMOSCEN)

The SIMOSCEN signal enables the oscillator and PLL.

### 10.4.6 Crystal Output Frequency Signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal ($f_{CGMXCLK}$) and comes directly from the crystal oscillator circuit. Figure 10-3 shows only the logical relation of CGMXCLK to OSC1 and OSC2 and may not represent the actual circuitry. The duty cycle of CGMXCLK is unknown and may depend on the crystal and other external factors. Also, the frequency and amplitude of CGMXCLK can be unstable at startup.

### 10.4.7 CGM Base Clock Output (CGMOUT)

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

### 10.4.8 CGM CPU Interrupt (CGMINT)

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

## 10.5 CGM Registers

Three registers control and monitor operation of the CGM:

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

### 10.5.1 PLL Control Register

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

Address: $001C

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

= Unimplemented

**Figure 10-4. PLL Control Register (PCTL)**

**PLLIE — PLL Interrupt Enable Bit**

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

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

**PLLF — PLL Flag Bit**

This read-only bit is set whenever the LOCK bit toggles. PLLF generates a CPU interrupt request if the PLLIE bit also is set. PLLF always reads as logic 0 when the AUTO bit in the PLL bandwidth control register (PBWC) is clear. Clear the PLLF bit by reading the PLL control register. Reset clears the PLLF bit.

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

***NOTE***

*Do not inadvertently clear the PLLF bit. Be aware that 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 10.3.3 Base Clock Selector Circuit. Reset sets this bit so that the loop can stabilize as the MCU is powering up.

1 = PLL on
0 = PLL off

**BCS — Base Clock Select Bit**

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGM output, CGMOUT. CGMOUT frequency is one-half the frequency of the selected clock. BCS cannot be set while the PLLON bit is clear. After toggling BCS,

it may take up to three CGMXCLK and three CGMVCLK cycles to complete the transition from one source clock to the other. During the transition, CGMOUT is held in stasis. See 10.3.3 Base Clock Selector Circuit. Reset and the STOP instruction clear the BCS bit.

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

***NOTE***

*PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. See 10.3.3 Base Clock Selector Circuit.*

**PCTL3–PCTL0 — Unimplemented**

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

## 10.5.2 PLL Bandwidth Control Register

The PLL bandwidth control register:

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

Address: $001D

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

= Unimplemented

**Figure 10-5. PLL Bandwidth Control Register (PBWC)**

**AUTO — Automatic Bandwidth Control Bit**

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

1 = Automatic bandwidth control
0 = Manual bandwidth control

**LOCK — Lock Indicator Bit**

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic 0 and has no meaning. Reset clears the LOCK bit.

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

**$\overline{ACQ}$ — 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

**XLD — Crystal Loss Detect Bit**

When the VCO output, CGMVCLK, is driving CGMOUT, this read/write bit can indicate whether the crystal reference frequency is active or not.

1 = Crystal reference not active
0 = Crystal reference active

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

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

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

**Bits 3–0 — Reserved for Test**

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

### 10.5.3 PLL Programming Register

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

Address: $001E

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

**Figure 10-6. PLL Programming Register (PPG)**

**MUL7–MUL4 — Multiplier Select Bits**

These read/write bits control the modulo feedback divider that selects the VCO frequency multiplier, N. (See 10.3.2.1 Circuits and 10.3.2.4 Programming the PLL). A value of $0 in the multiplier select bits configures the modulo feedback divider the same as a value of $1. Reset initializes these bits to $6 to give a default multiply value of 6.

**Table 10-3. VCO Frequency Multiplier (N) Selection**

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

***NOTE***

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

**VRS7–VRS4 — VCO Range Select Bits**

These read/write bits control the hardware center-of-range linear multiplier L, which controls the hardware center-of-range frequency, $f_{VRS}$. (See 10.3.2.1 Circuits, 10.3.2.4 Programming the PLL, and 10.5.1 PLL Control Register.) VRS7–VRS4 cannot be written when the PLLON bit in the PLL control register (PCTL) is set. See 10.3.2.5 Special Programming Exceptions. A value of $0 in the VCO range select bits disables the PLL and clears the BCS bit in the PCTL. (See 10.3.3 Base Clock Selector Circuit and 10.3.2.5 Special Programming Exceptions for more information.) Reset initializes the bits to $6 to give a default range multiply value of 6.

***NOTE***

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

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

## 10.6 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 interrupt requests from the PLL. PLLF, the interrupt flag in the PCTL, becomes set whether CPU interrupt requests are enabled or not. When the AUTO bit is clear, CPU interrupt requests from the PLL are disabled and PLLF reads as logic 0.

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

***NOTE***

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

## 10.7 Low-Power Modes

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

### 10.7.1 Wait Mode

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

### 10.7.2 Stop Mode

The STOP instruction disables the CGM and holds low all CGM outputs (CGMXCLK, CGMOUT, and CGMINT).

If CGMOUT is being driven by CGMVCLK and a STOP instruction is executed; the PLL will clear the BCS bit in the PLL control register, causing CGMOUT to be driven by CGMXCLK. When the MCU recovers from STOP, the crystal clock divided by two drives CGMOUT and BCS remains clear.

## 10.8 CGM During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See Chapter 13 Break Module (BRK).

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.

## 10.9 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.

### 10.9.1 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% acquisition time tolerance. If a command instructs the system to change from 0 Hz to

1 MHz, the acquisition time is the time taken for the frequency to reach 1 MHz ±50 kHz. Fifty kHz = 5% of the 1-MHz step input. If the system is operating at 1 MHz and suffers a –100 kHz noise hit, the acquisition time is the time taken to return from 900 kHz to 1 MHz ±5 kHz. Five kHz = 5% of the 100-kHz step input.

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

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

- Acquisition time, $t_{acq}$, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the tracking mode entry tolerance, $\Delta_{trk}$. Acquisition time is based on an initial frequency error, $(f_{des} - f_{orig})/f_{des}$, of not more than ±100%. In automatic bandwidth control mode (see 10.3.2.3 Manual and Automatic PLL Bandwidth Modes), acquisition time expires when the $\overline{ACQ}$ bit becomes set in the PLL bandwidth control register (PBWC).
- Lock time, $t_{Lock}$, is the time the PLL takes to reduce the error between the actual output frequency and the desired output frequency to less than the lock mode entry tolerance, $\Delta_{Lock}$. Lock time is based on an initial frequency error, $(f_{des} - f_{orig})/f_{des}$, of not more than ±100%. In automatic bandwidth control mode, lock time expires when the LOCK bit becomes set in the PLL bandwidth control register (PBWC). (See 10.3.2.3 Manual and Automatic PLL Bandwidth Modes).

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

### 10.9.2 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, $f_{CGMRDV}$ (please reference Figure 10-1). This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these corrections. This parameter is also under user control via the choice of crystal frequency $f_{CGMXCLK}$.

Another critical parameter is the external filter capacitor. The PLL modifies the voltage on the VCO by adding or subtracting charge from this capacitor. Therefore, the rate at which the voltage changes for a given frequency error (thus a change in charge) is proportional to the capacitor size. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. See 10.9.3 Choosing a Filter Capacitor.

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.

### 10.9.3 Choosing a Filter Capacitor

As described in 10.9.2 Parametric Influences on Reaction Time, the external filter capacitor, $C_F$, is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage. The value of the capacitor must, therefore, be chosen with supply potential and reference frequency in mind. For proper operation, the external filter capacitor must be chosen according to this equation:

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

For acceptable values of $C_{fact}$, (see Chapter 28 Electrical Specifications). For the value of $V_{DDA}$, choose the voltage potential at which the MCU is operating. If the power supply is variable, choose a value near the middle of the range of possible supply values.

This equation does not always yield a commonly available capacitor size, so round to the nearest available size. If the value is between two different sizes, choose the higher value for better stability. Choosing the lower size may seem attractive for acquisition time improvement, but the PLL may become unstable. Also, always choose a capacitor with a tight tolerance (±20% or better) and low dissipation.

### 10.9.4 Reaction Time Calculation

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

- Correct selection of filter capacitor, $C_F$ (see 10.9.3 Choosing a Filter Capacitor).
- Room temperature operation
- Negligible external leakage on CGMXFC
- Negligible noise

The K factor in the equations is derived from internal PLL parameters. $K_{acq}$ is the K factor when the PLL is configured in acquisition mode, and $K_{trk}$ is the K factor when the PLL is configured in tracking mode. (See 10.3.2.2 Acquisition and Tracking Modes).

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

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

$$t_{Lock} = t_{ACQ} + t_{AL}$$

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

In automatic bandwidth control mode, the acquisition and lock times are quantized into units based on the reference frequency. (See 10.3.2.3 Manual and Automatic PLL Bandwidth Modes). A certain number of clock cycles, $n_{ACQ}$, is required to ascertain that the PLL is within the tracking mode entry tolerance, $\Delta_{TRK}$, before exiting acquisition mode. A certain number of clock cycles, $n_{TRK}$, is required to ascertain that the PLL is within the lock mode entry tolerance, $\Delta_{Lock}$. Therefore, the acquisition time, $t_{ACQ}$, is an integer multiple of $n_{ACQ}/f_{CGMRDV}$, and the acquisition to lock time, $t_{AL}$, is an integer multiple of $n_{TRK}/f_{CGMRDV}$. Also, since the average frequency over the entire measurement period must be within the specified tolerance, the total time usually is longer than $t_{Lock}$ as calculated above.

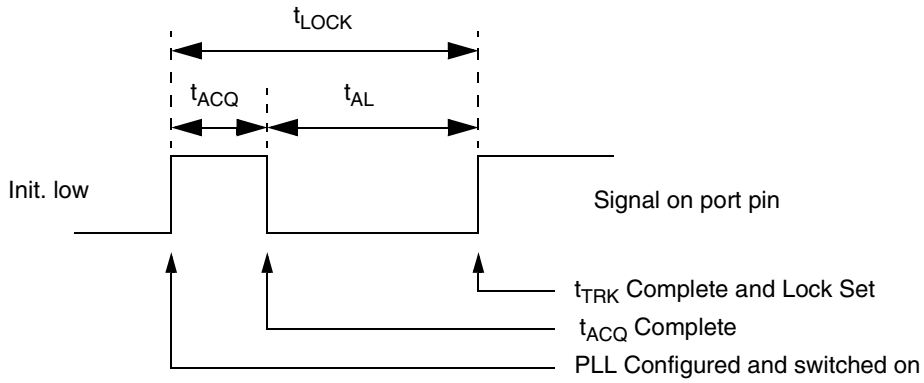
In manual mode, it is usually necessary to wait considerably longer than $t_{Lock}$ before selecting the PLL clock (see 10.3.3 Base Clock Selector Circuit), because the factors described in 10.9.2 Parametric Influences on Reaction Time, may slow the lock time considerably.

When defining a limit in software for the maximum lock time, the value must allow for variation due to all of the factors mentioned in this chapter, especially due to the $C_F$ capacitor and application specific influences.

The calculated lock time is only an indication and it is the customer's responsibility to allow enough of a guard band for their application. Prior to finalizing any software and while determining the maximum lock time, take into account all device to device differences. Typically, applications set the maximum lock time as an order of magnitude higher than the measured value. This is considered sufficient for all such device to device variation.

Freescale recommends measuring the lock time of the application system by utilizing dedicated software, running in FLASH, EEPROM or RAM. This should toggle a port pin when the PLL is first configured and switched on, then again when it goes from acquisition to lock mode and finally again when the PLL lock bit is set. The resultant waveform can be captured on an oscilloscope and used to determine the typical lock time for the microcontroller and the associated external application circuit.

For example,




***NOTE***

*The filter capacitor should be fully discharged prior to making any measurements.*

# Chapter 11
# Configuration Register (CONFIG-1)

## 11.1 Introduction

This chapter describes the configuration register (CONFIG-1), which contains bits that configure these options:

- Resets caused by the LVI module
- Power to the LVI module
- LVI enabled during stop mode
- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- Computer operating properly module (COP)
- Stop instruction enable/disable.

## 11.2 Functional Description

The configuration register is a write-once register. Out of reset, the configuration register will read the default value. Once the register is written, further writes will have no effect until a reset occurs.

***NOTE***

*If the LVI module and the LVI reset signal are enabled, a reset occurs when $V_{DD}$ falls to a voltage, $LVI_{TRIPF}$, and remains at or below that level for at least nine consecutive CPU cycles. Once an LVI reset occurs, the MCU remains in reset until $V_{DD}$ rises to a voltage, $LVI_{TRIPR}$.*

Address: $001F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | LVISTOP | R | LVIRST | LVIPWR | SSREC | COPL | STOP | COPD |
| Reset: | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 11-1. Configuration Register (CONFIG-1)**

**LVISTOP — LVI Stop Mode Enable Bit**

LVISTOP enables the LVI module in stop mode. (See Chapter 16 Low-Voltage Inhibit (LVI)).

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

***NOTE***

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

**LVIRST — LVI Reset Enable Bit**

LVIRST enables the reset signal from the LVI module. (See Chapter 16 Low-Voltage Inhibit (LVI)).

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

**LVIPWR — LVI Power Enable Bit**

LVIPWR enables the LVI module. (See Chapter 16 Low-Voltage Inhibit (LVI)).

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

**SSREC — Short Stop Recovery Bit**

SSREC enables the CPU to exit stop mode with a delay of 32 CGMXCLK cycles instead of a 4096-CGMXCLK cycle delay. (See 9.6.2 Stop Mode).

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

***NOTE***

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

**COPL — COP Long Timeout**

COPL enables the shorter COP timeout period. (See Chapter 15 Computer Operating Properly (COP)).

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

**STOP — STOP Instruction Enable Bit**

STOP enables the STOP instruction.

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

**COPD — COP Disable Bit**

COPD disables the COP module. (See Chapter 15 Computer Operating Properly (COP)).

1 = COP module disabled
0 = COP module enabled

***WARNING***

***Extra care should be exercised when using this emulation part for development of code to be run in ROM AZ, AB or AS parts that the options selected by setting the CONFIG-1 register match exactly the options selected on any ROM code request submitted. The enable/disable logic is not necessarily identical in all parts of the AS and AZ families. If in doubt, check with your local field applications representative.***

# Chapter 12
# Configuration Register (CONFIG-2)

## 12.1 Introduction

This chapter describes the configuration register (CONFIG-2). This register contains bits that configure these options:

- Configures the device to either the MC68HC08AZxx emulator or the MC68HC08ASxx emulator
- Disables the CAN module

## 12.2 Functional Description

The configuration register is a write-once register. Out of reset, the configuration register will read the default. Once the register is written, further writes will have no effect until a reset occurs.

Address: $FE09

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | EEDIV CLK | R | R | MSCAND | AT60A | R | R | AZxx |
| Write: | | | | | R | | | |
| Reset: | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |

R = Reserved

**Figure 12-1. Configuration Register (CONFIG-2)**

**AT60A — Device Indicator**

This read-only bit is used to distinguish an MC68HC908AS60A and MC68HC908AZ60A from older non-'A' suffix versions.

1 = 'A' version
0 = Non-'A' version

**EEDIVCLK — EEPROM Timebase Divider Clock Select Bit**

This bit selects the reference clock source for the EEPROM-1 and EEPROM-2 timebase divider modules.

1 = EExDIV clock input is driven by internal bus clock
0 = EExDIV clock input is driven by CGMXCLK

**MSCAND — MSCAN Disable Bit**

MSCAND disables the MSCAN module. (See Chapter 23 MSCAN Controller (MSCAN08)).

1 = MSCAN module disabled
0 = MSCAN Module enabled

### AZxx — AZxx Emulator Enable Bit

AZxx enables the MC68HC08AZxx emulator configuration. This bit will be 0 out of reset.

1 = MC68HC08AZxx emulator enabled
0 = MC68HC08ASxx emulator enabled

***NOTE***

*AZxx bit is reset by a POWER-ON-RESET only.*

# Chapter 13
# Break Module (BRK)

## 13.1 Introduction

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

## 13.2 Features

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

## 13.3 Functional Description

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

The following events can cause a break interrupt to occur:

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

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. Figure 13-1 shows the structure of the break module.

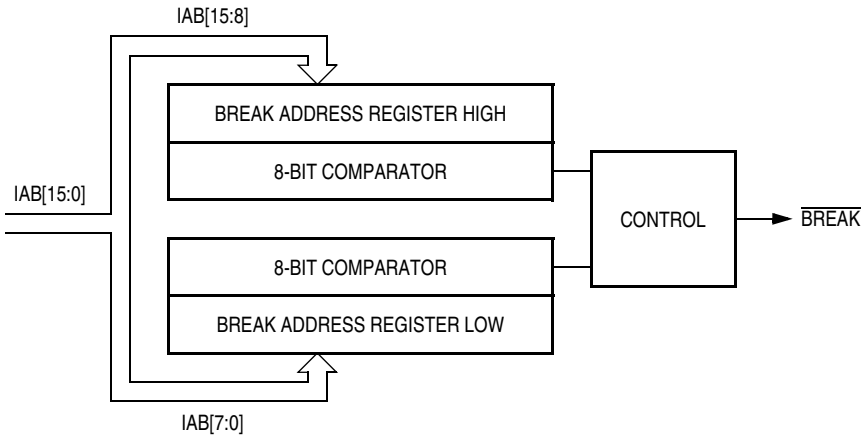



**Figure 13-1. Break Module Block Diagram**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| Break Address Register High (BRKH) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Break Address Register Low (BRKL) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Break Status and Control Register (BSCR) | Read: | BRKE | BRKA | 0 | 0 | 0 | 0 | 0 | 0 |
| | Write: | | | | | | | | |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 13-2. I/O Register Summary**

**Table 13-1. I/O Register Address Summary**

| **Register** | BRKH | BRKL | BSCR |
|---|---|---|---|
| **Address** | $FE0C | $FE0D | $FE0E |

### 13.3.1 Flag Protection During Break Interrupts

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

### 13.3.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with $FFFC:$FFFD ($FEFC:$FEFD in monitor mode)

The break interrupt begins after completion of the CPU instruction in progress. If the break address register match occurs on the last cycle of a CPU instruction, the break interrupt begins immediately.

### 13.3.3 TIM During Break Interrupts

A break interrupt stops the timer counter.

### 13.3.4 COP During Break Interrupts

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

## 13.4 Low-Power Modes

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

### 13.4.1 Wait Mode

If enabled, the break module is active in wait mode. The SIM break wait bit (BW) in the SIM break status register indicates whether wait was exited by a break interrupt. If so, the user can modify the return address on the stack by subtracting one from it. (See 9.7.1 SIM Break Status Register).

### 13.4.2 Stop Mode

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

## 13.5 Break Module Registers

These registers control and monitor operation of the break module:

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

### 13.5.1 Break Status and Control Register

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

| Address: | $FE0E | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | BRKE | BRKA | 0 | 0 | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 13-3. Break Status and Control Register (BSCR)**

**BRKE — Break Enable Bit**

This read/write bit enables breaks on break address register matches. Clear BRKE by writing a 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 1 to BRKA generates a break interrupt. Clear BRKA by writing a 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

### 13.5.2 Break Address Registers

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

| Register: | BRKH | BRKL | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Address: | $FE0C | $FE0D | | | | | | |
| | 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 |
| Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-4. Break Address Registers (BRKH and BRKL)**

# Chapter 14
# Monitor ROM (MON)

## 14.1 Introduction

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

## 14.2 Features

Features of the monitor ROM include:

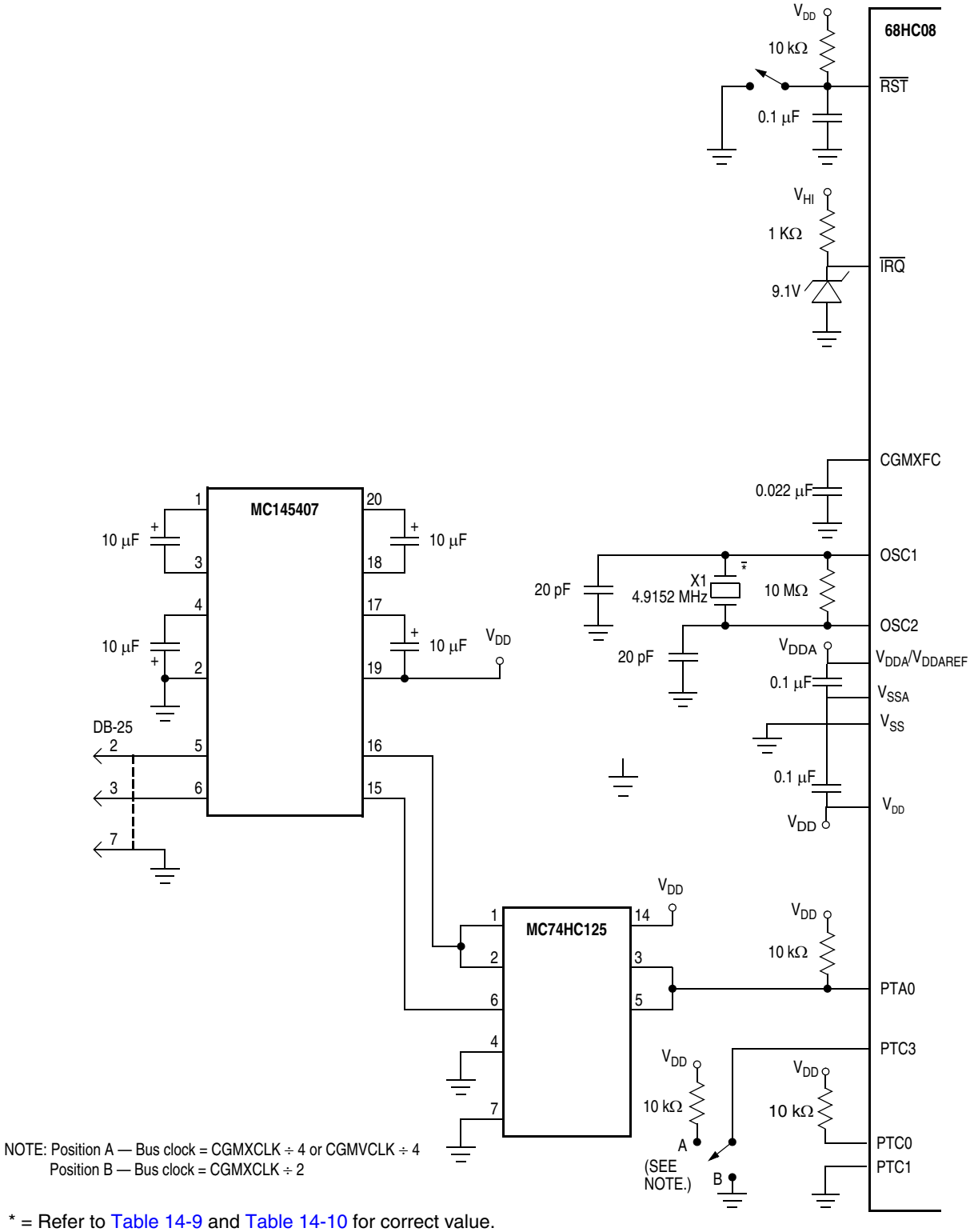
- Normal User-Mode Pin Functionality
- One Pin Dedicated to Serial Communication between Monitor ROM and Host Computer
- Standard Mark/Space Non-Return-to-Zero (NRZ) Communication with Host Computer
- Up to 28.8 kBaud Communication with Host Computer
- Execution of Code in RAM or FLASH
- FLASH Security
- FLASH Programming

## 14.3 Functional Description

Monitor ROM receives and executes commands from a host computer. Figure 14-1 shows a sample circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

While simple monitor commands can access any memory address, the MC68HC908AS60A and MC68HC908AZ60A have a FLASH security feature to prevent external viewing of the contents of FLASH. Proper procedures must be followed to verify FLASH content. Access to the FLASH is denied to unauthorized users of customer specified software (see 14.3.8 Security).

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




* = Refer to Table 14-9 and Table 14-10 for correct value.

**Figure 14-1. Monitor Mode Circuit**

### 14.3.1 Entering Monitor Mode

Table 14-1 shows the pin conditions for entering monitor mode.

**Table 14-1. Mode Selection**

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

1. For $V_{HI}$, 28.1.4 5.0 Volt DC Electrical Characteristics, and 28.1.1 Maximum Ratings.

Enter monitor mode by either

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

Once out of reset, the MCU waits for the host to send eight security bytes (see 14.3.8 Security). After the security bytes, the MCU sends a break signal (10 consecutive logic 0s) to the host computer, indicating that it is ready to receive a command.

Monitor mode uses alternate vectors for reset, SWI, and break interrupt. The alternate vectors are in the $FE page instead of the $FF page and allow code execution from the internal monitor firmware instead of user code. The COP module is disabled in monitor mode as long as $V_{HI}$ (see 28.1.4 5.0 Volt DC Electrical Characteristics), is applied to either the $\overline{IRQ}$ pin or the RESET pin. (See Chapter 9 System Integration Module (SIM) for more information on modes of operation).

***NOTE***

*Holding the PTC3 pin low when entering monitor mode causes a bypass of a divide-by-two stage at the oscillator. The CGMOUT frequency is equal to the CGMXCLK frequency, and the OSC1 input directly generates internal bus clocks. In this case, the OSC1 signal must have a 50% duty cycle at maximum bus frequency.*

Table 14-2 is a summary of the differences between user mode and monitor mode.

**Table 14-2. Mode Differences**

| Modes | Functions | | | | | | |
|---|---|---|---|---|---|---|---|
| | COP | Reset Vector High | Reset Vector Low | Break Vector High | Break Vector Low | SWI Vector High | SWI Vector Low |
| User | Enabled | $FFFE | $FFFF | $FFFC | $FFFD | $FFFC | $FFFD |
| Monitor | Disabled(1) | $FEFE | $FEFF | $FEFC | $FEFD | $FEFC | $FEFD |

1. If the high voltage ($V_{HI}$) is removed from the $\overline{IRQ}$ and/or RESET pin while in monitor mode, the SIM asserts its COP enable output. The COP is enabled or disabled by the COPD bit in the configuration register. (see 28.1.4 5.0 Volt DC Electrical Characteristics).

### 14.3.2 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. (See Figure 14-2 and Figure 14-3.)

The data transmit and receive rate can be anywhere up to 28.8 kBaud. Transmit and receive baud rates must be identical.

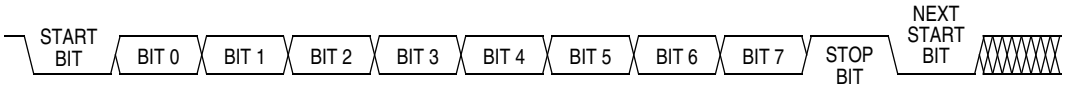



**Figure 14-2. Monitor Data Format**

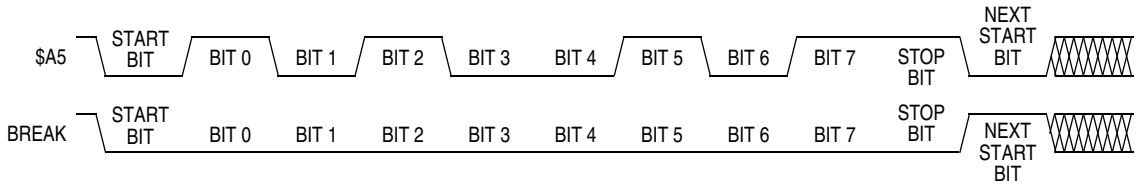



**Figure 14-3. Sample Monitor Waveforms**

### 14.3.3 Echoing

As shown in Figure 14-4, the monitor ROM immediately echoes each received byte back to the PTA0 pin for error checking.

Any result of a command appears after the echo of the last byte of the command.

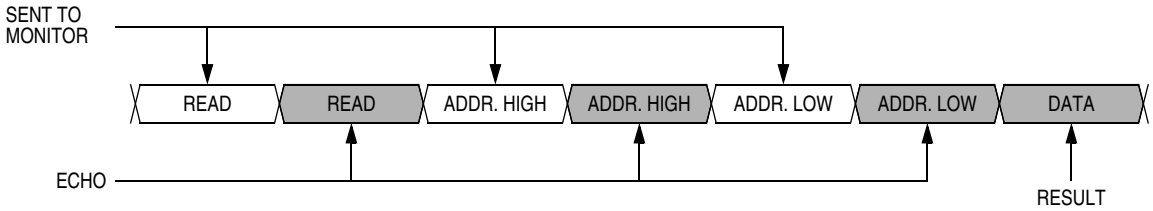



**Figure 14-4. Read Transaction**

### 14.3.4 Break Signal

A start bit followed by nine low bits is a break signal. (See Figure 14-5). When the monitor receives a break signal, it drives the PTA0 pin high for the duration of two bits before echoing the break signal.

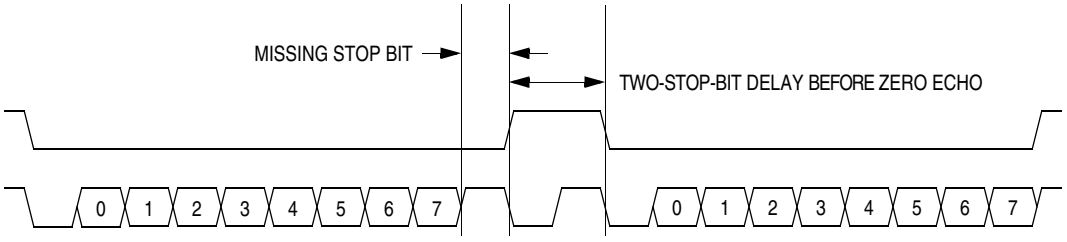



**Figure 14-5. Break Transaction**

### 14.3.5 Commands

The monitor ROM uses these commands:

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

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

**Table 14-3. READ (Read Memory) Command**

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

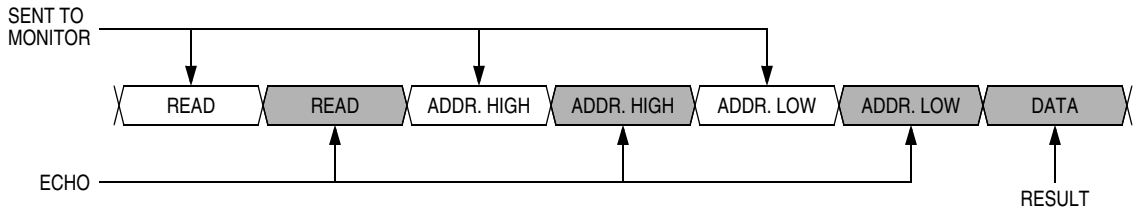
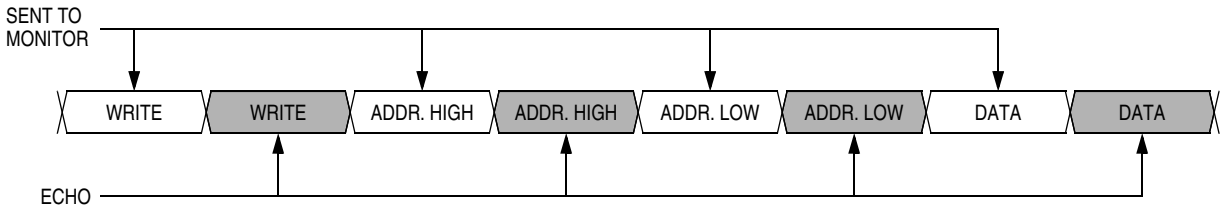


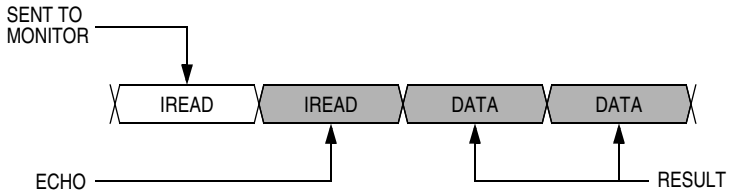

**Table 14-4. WRITE (Write Memory) Command**

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



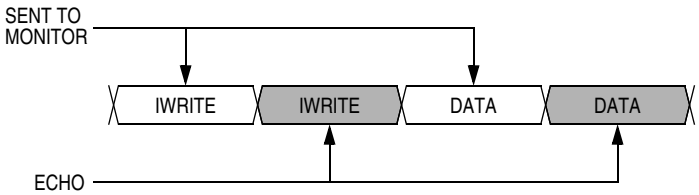
### Table 14-5. IREAD (Indexed Read) Command

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




### Table 14-6. IWRITE (Indexed Write) Command

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




### Table 14-7. READSP (Read Stack Pointer) Command

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

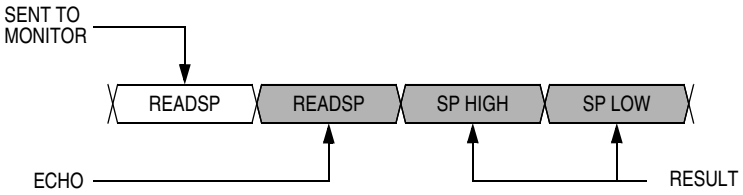
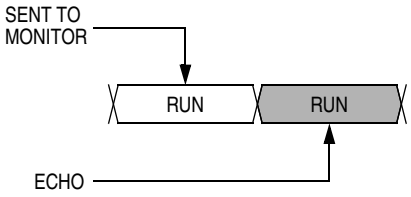


**Table 14-8. RUN (Run User Program) Command**

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



### 14.3.6 MC68HC908AS60A Baud Rate

With a 4.9152-MHz crystal and the PTC3 pin at logic 1 during reset, data is transferred between the monitor and host at 4800 baud. If the PTC3 pin is at logic 0 during reset, the monitor baud rate is 9600. When the CGM output, CGMOUT, is driven by the PLL, the baud rate is determined by the MUL[7:4] bits in the PLL programming register (PPG). (See Chapter 10 Clock Generator Module (CGM)).

**Table 14-9. MC68HC908AS60A Monitor Baud Rate Selection**

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

### 14.3.7 MC68HC908AZ60A Baud Rate

The MC68HC908AZ60A features a monitor mode which is optimised to operate with either a 4.9152 MHz crystal clock source (or multiples of 4.9152 MHz) or a 4 MHz crystal (or multiples of 4 MHz). This supports designs which use the MSCAN module, which is generally clocked from a 4 MHz, 8 MHz or 16 MHZ internal reference clock. The table below outlines the available baud rates for a range of crystals and how they can match to a PC baud rate.

**Table 14-10 MC68HC908AZ60A Monitor Baud Rate Selection**

| | Baud rate | | Closest PC baud PC | | Error % | |
|---|---|---|---|---|---|---|
| Clock freq | PTC3=0 | PTC3=1 | PTC3=0 | PTC3=1 | PTC3=0 | PTC3=1 |
| 32kHz | 57.97 | 28.98 | 57.6 | 28.8 | 0.64 | 0.63 |
| 1MHz | 1811.59 | 905.80 | 1800 | 900 | 0.64 | 0.64 |
| 2MHz | 3623.19 | 1811.59 | 3600 | 1800 | 0.64 | 0.64 |
| 4MHz | 7246.37 | 3623.19 | 7200 | 3600 | 0.64 | 0.64 |
| 4.194MHz | 7597.83 | 3798.91 | 7680 | 3840 | 1.08 | 1.08 |
| 4.9152MHz | 8904.35 | 4452.17 | 8861 | 4430 | 0.49 | 0.50 |
| 8MHz | 14492.72 | 7246.37 | 14400 | 7200 | 0.64 | 0.64 |
| 16MHz | 28985.51 | 14492.75 | 28800 | 14400 | 0.64 | 0.64 |

***WARNING***

***Care should be taken when setting the baud rate since incorrect baud rate setting can result in communications failure.***

### 14.3.8 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. If FLASH is unprogrammed, the eight security byte values to be sent are $FF, the unprogrammed state of FLASH.*

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

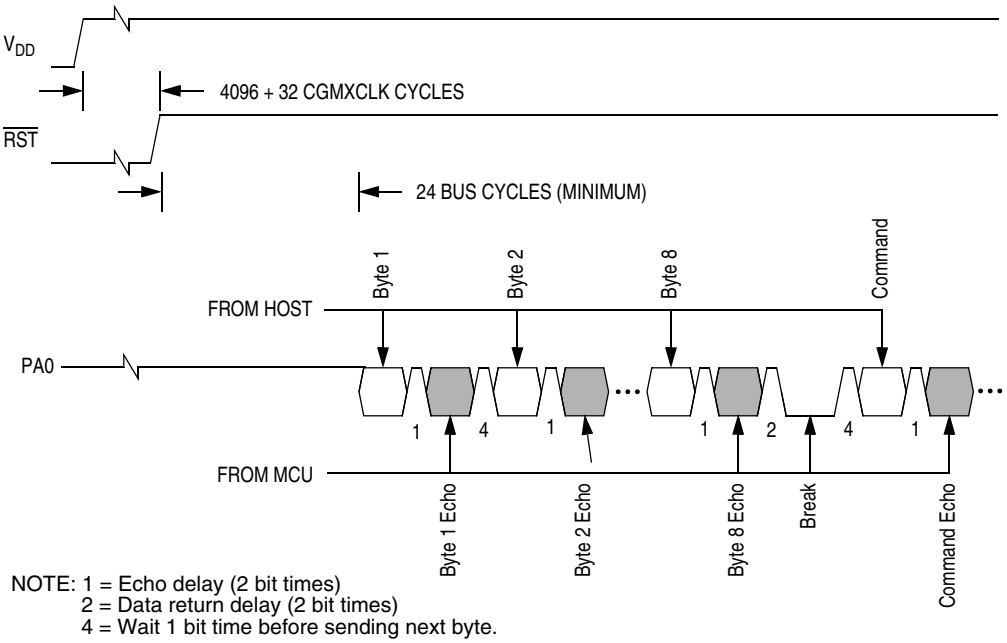



**Figure 14-6. Monitor Mode Entry Timing**

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. After the host bypasses security, any reset other than a power-on reset requires the host to send another eight bytes. If the reset was not a power-on reset, the security remains bypassed regardless of the data that the host sends.

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

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

***NOTE***

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

# Chapter 15
# Computer Operating Properly (COP)

## 15.1 Introduction

The COP module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by periodically clearing the COP counter.

## 15.2 Functional Description

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler. If not cleared by software, the COP counter overflows and generates an asynchronous reset after $2^{13} - 2^4$ or $2^{18} - 2^4$ CGMXCLK cycles, depending on the state of the COP long timeout bit, COPL, in the CONFIG-1. When COPL = 0, a 4.9152-MHz crystal gives a COP timeout period of 53.3 ms. Writing any value to location $FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 4–12 of the SIM counter.

***NOTE***

*Service the COP immediately after reset and before entering or after exiting stop mode to guarantee the maximum time before the first COP counter overflow.*

A COP reset pulls the $\overline{\text{RST}}$ pin low for 32 CGMXCLK cycles and sets the COP bit in the reset status register (RSR).

In monitor mode, the COP is disabled if the $\overline{\text{RST}}$ pin or the $\overline{\text{IRQ}}$ pin is held at $V_{Hi}$. During the break state, $V_{Hi}$ on the $\overline{\text{RST}}$ pin disables the COP.

***NOTE***

*Place COP clearing instructions in the main program and not in an interrupt subroutine. Such an interrupt subroutine could keep the COP from generating a reset even while the main program is not working properly.*

## 15.3 I/O Signals

The following paragraphs describe the signals shown in Figure 15-1.

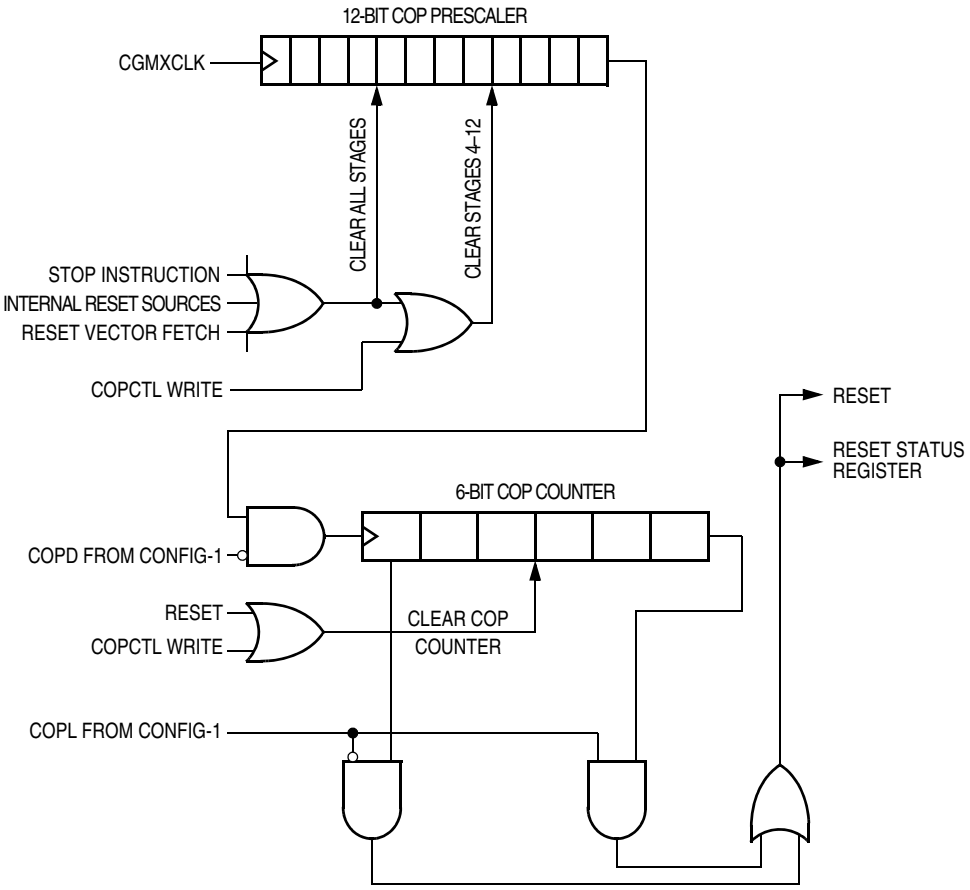



**Figure 15-1. COP Block Diagram**

### 15.3.1 CGMXCLK

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

### 15.3.2 STOP Instruction

The STOP instruction clears the COP prescaler.

### 15.3.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see 15.4 COP Control Register), clears the COP counter and clears stages 12 through 4 of the COP prescaler. Reading the COP control register returns the reset vector.

### 15.3.4 Power-On Reset

The power-on reset (POR) circuit clears the COP prescaler 4096 CGMXCLK cycles after power-up.

### 15.3.5 Internal Reset

An internal reset clears the COP prescaler and the COP counter.

### 15.3.6 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.

### 15.3.7 COPD

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register. (See Chapter 11 Configuration Register (CONFIG-1)).

### 15.3.8 COPL

The COPL signal reflects the state of the COP rate select bit. (COPL) in the configuration register. (See Chapter 11 Configuration Register (CONFIG-1)).

## 15.4 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 15-2. COP Control Register (COPCTL)**

## 15.5 Interrupts

The COP does not generate CPU interrupt requests.

## 15.6 Monitor Mode

The COP is disabled in monitor mode when $V_{Hi}$ is present on the $\overline{IRQ}$ pin or on the $\overline{RST}$ pin.

## 15.7 Low-Power Modes

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

### 15.7.1 Wait Mode

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

### 15.7.2 Stop Mode

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

The STOP bit in the configuration register (CONFIG) enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by clearing the STOP bit.

## 15.8 COP Module During Break Interrupts

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

# Chapter 16
# Low-Voltage Inhibit (LVI)

## 16.1 Introduction

This chapter describes the low-voltage inhibit module (LVI47, Version A), which monitors the voltage on the $V_{DD}$ pin and can force a reset when the $V_{DD}$ voltage falls to the LVI trip voltage.

## 16.2 Features

Features of the LVI module include:

- Programmable LVI Reset
- Programmable Power Consumption
- Digital Filtering of $V_{DD}$ Pin Level

***NOTE***

*If a low voltage interrupt (LVI) occurs during programming of EEPROM or Flash memory, then adequate programming time may not have been allowed to ensure the integrity and retention of the data. It is the responsibility of the user to ensure that in the event of an LVI any addresses being programmed receive specification programming conditions.*

## 16.3 Functional Description

Figure 16-1 shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. The LVI power bit, LVIPWR, enables the LVI to monitor $V_{DD}$ voltage. The LVI reset bit, LVIRST, enables the LVI module to generate a reset when $V_{DD}$ falls below a voltage, $LVI_{TRIPF}$, and remains at or below that level for nine or more consecutive CPU cycles.

Note that short $V_{DD}$ spikes may not trip the LVI. It is the user's responsibility to ensure a clean $V_{DD}$ signal within the specified operating voltage range if normal microcontroller operation is to be guaranteed.

LVISTOP, enables the LVI module during stop mode. This will ensure when the STOP instruction is implemented, the LVI will continue to monitor the voltage level on $V_{DD}$. LVIPWR, LVISTOP, and LVIRST are in the configuration register, CONFIG-1 (see Chapter 11 Configuration Register (CONFIG-1)).

Once an LVI reset occurs, the MCU remains in reset until $V_{DD}$ rises above a voltage, $LVI_{TRIPR}$. $V_{DD}$ must be above $LVI_{TRIPR}$ for only one CPU cycle to bring the MCU out of reset (see 16.3.2 Forced Reset Operation). The output of the comparator controls the state of the LVIOUT flag in the LVI status register (LVISR).

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

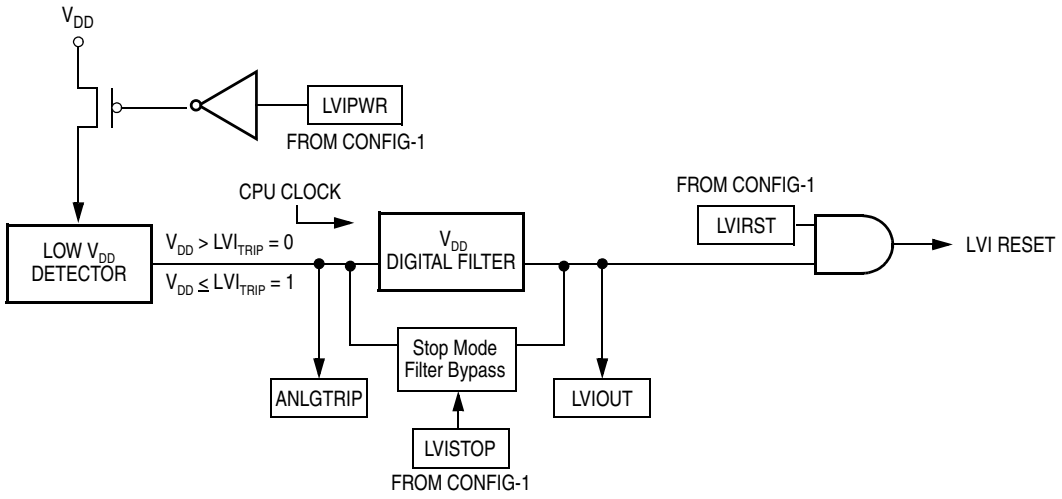



**Figure 16-1. LVI Module Block Diagram**

| 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: | | | | | | | | |

= Unimplemented

**Figure 16-2. LVI I/O Register Summary**

### 16.3.1 Polled LVI Operation

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

### 16.3.2 Forced Reset Operation

In applications that require $V_{DD}$ to remain above the $LVI_{TRIPF}$ level, enabling LVI resets allows the LVI module to reset the MCU when $V_{DD}$ falls to the $LVI_{TRIPF}$ level and remains at or below that level for nine or more consecutive CPU cycles. In the configuration register, the LVIPWR and LVIRST bits must be at logic 1 to enable the LVI module and to enable LVI resets.

### 16.3.3 False Reset Protection

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

## 16.4 LVI Status Register

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

Address: $FE0F

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

= Unimplemented

**Figure 16-3. LVI Status Register (LVISR)**

**LVIOUT — LVI Output Bit**

This read-only flag becomes set when the $V_{DD}$ voltage falls below the $LVI_{TRIPF}$ voltage for 32 to 40 CGMXCLK cycles. (See Table 16-1). Reset clears the LVIOUT bit.

**Table 16-1. LVIOUT Bit Indication**

| $V_{DD}$ | | LVIOUT |
|---|---|---|
| **At Level:** | **For Number of CGMXCLK Cycles:** | |
| $V_{DD} > {}^{LVI}TRIPR$ | Any | 0 |
| $V_{DD} < {}^{LVI}TRIPF$ | < 32 CGMXCLK Cycles | 0 |
| $V_{DD} < {}^{LVI}TRIPF$ | Between 32 and 40 CGMXCLK Cycles | 0 or 1 |
| $V_{DD} < {}^{LVI}TRIPF$ | > 40 CGMXCLK Cycles | 1 |
| ${}^{LVI}TRIPF < V_{DD} < {}^{LVI}TRIPR$ | Any | Previous Value |

## 16.5 LVI Interrupts

The LVI module does not generate interrupt requests.

## 16.6 Low-Power Modes

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

### 16.6.1 Wait Mode

With the LVIPWR bit in the configuration register programmed to 1, the LVI module is active after a WAIT instruction.

With the LVIRST bit in the configuration register programmed to 1, the LVI module can generate a reset and bring the MCU out of wait mode.

### 16.6.2 Stop Mode

With the LVISTOP and LVIPWR bits in the configuration register programmed to a logic 1, the LVI module will be active after a STOP instruction. Because CPU clocks are disabled during stop mode, the LVI trip must bypass the digital filter to generate a reset and bring the MCU out of stop.

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

***NOTE***

*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 specified operating voltage $V_{DD}$.*

# Chapter 17
# External Interrupt Module (IRQ)

## 17.1 Introduction

This chapter describes the nonmaskable external interrupt (IRQ) input.

## 17.2 Features

Features include:

- Dedicated External Interrupt Pin ($\overline{\text{IRQ}}$)
- Hysteresis Buffer
- Programmable Edge-Only or Edge- and Level-Interrupt Sensitivity
- Automatic Interrupt Acknowledge

## 17.3 Functional Description

A falling edge applied to the external interrupt pin can latch a CPU interrupt request. Figure 17-1 shows the structure of the IRQ module.

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

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the latch that caused the vector fetch.
- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (ISCR). Writing a logic 1 to the ACK bit clears the IRQ latch.
- Reset — A reset automatically clears both interrupt latches.

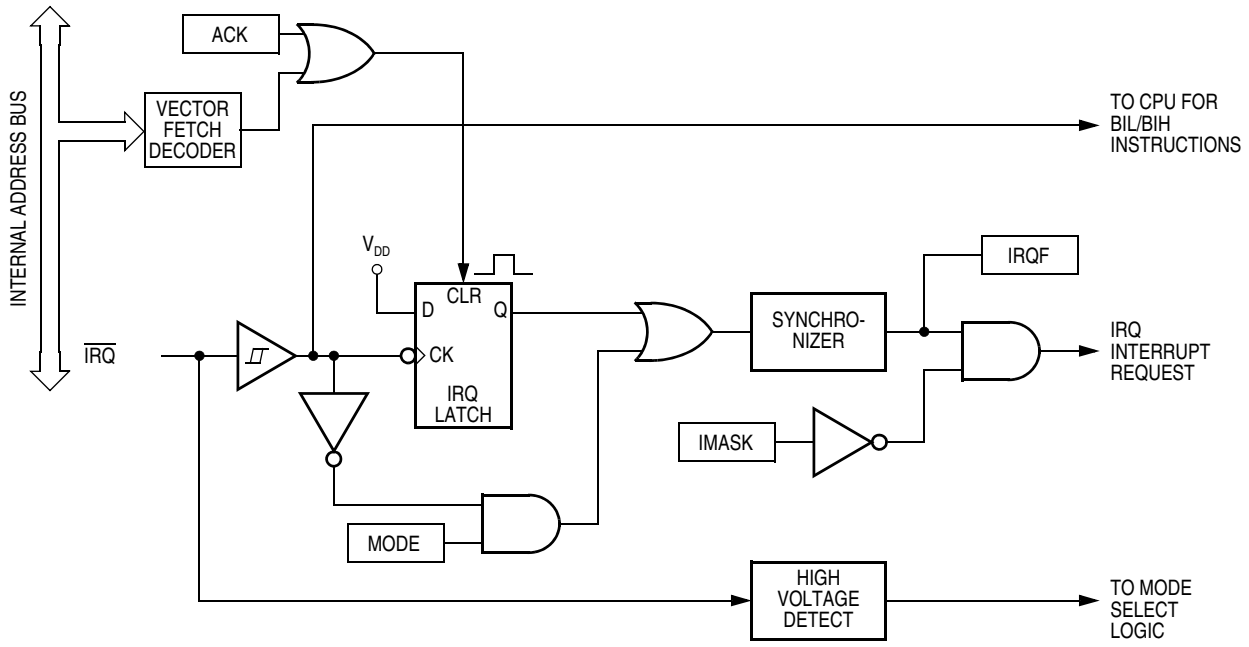



**Figure 17-1. IRQ Block Diagram**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $001A | IRQ Status/Control Register (ISCR) | Read: | 0 | 0 | 0 | 0 | IRQF | 0 | IMASK | MODE |
| | | Write: | R | R | R | R | R | ACK | | |

R = Reserved

**Figure 17-2. IRQ I/O Register Summary**

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

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

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

- Vector fetch or software clear
- Return of the interrupt pin to a high level

The vector fetch or software clear may occur before or after the interrupt pin returns to a high level. 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 ISCR masks all external interrupt requests. A latched interrupt request is not presented to the interrupt priority logic unless the corresponding IMASK bit is clear.

***NOTE***

*The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests. (See Figure 17-3).*

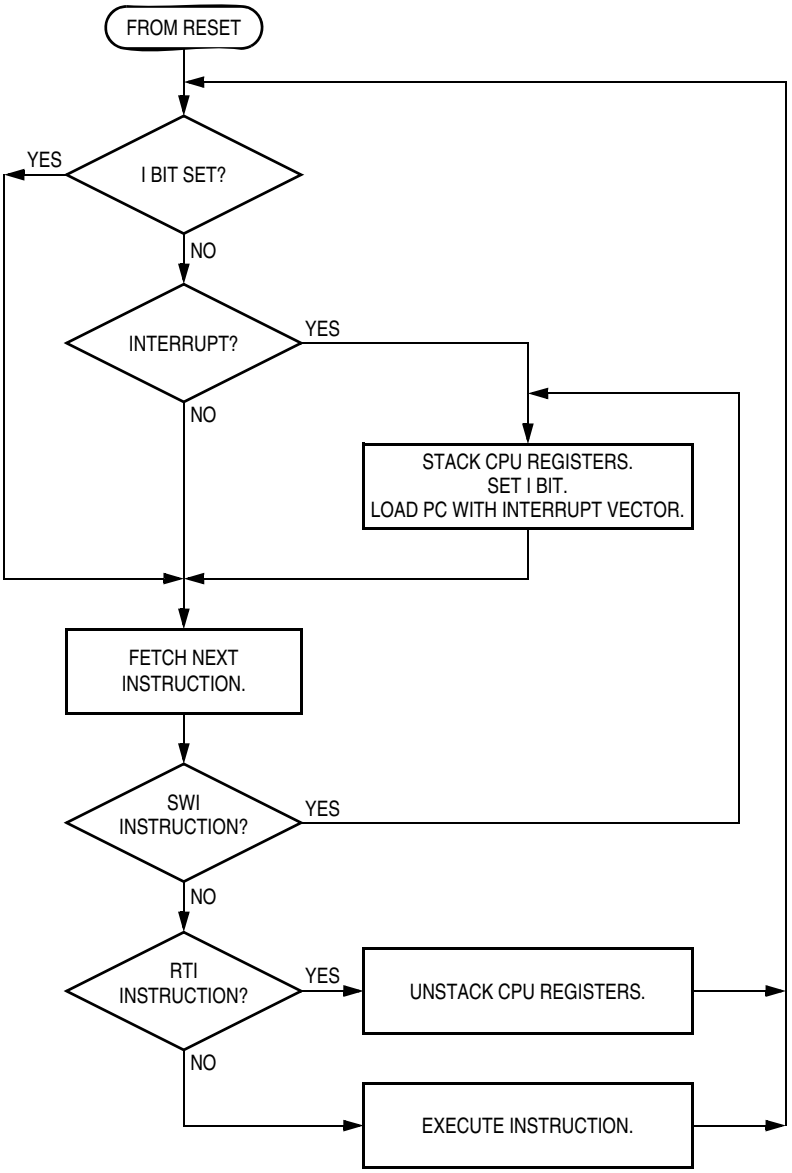



**Figure 17-3. IRQ Interrupt Flowchart**

## 17.4 $\overline{\text{IRQ}}$ Pin

A falling edge 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 the IRQ latch:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a 1 to the ACK bit in the interrupt status and control register (ISCR). The ACK bit is useful in applications that poll the $\overline{\text{IRQ}}$ pin and require software to clear the IRQ latch. Writing to the ACK bit can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the $\overline{\text{IRQ}}$ pin. A falling edge on $\overline{\text{IRQ}}$ that occurs after writing to the ACK bit latches another interrupt request. If the IRQ mask bit, IMASK, is clear, the CPU loads the program counter with the vector address at locations $FFFA and $FFFB.
- Return of the $\overline{\text{IRQ}}$ pin to a high level — As long as the $\overline{\text{IRQ}}$ pin is low, the IRQ1 latch remains set.

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

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

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

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

***NOTE***

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

## 17.5 IRQ Module During Break Interrupts

The system integration module (SIM) controls whether the IRQ interrupt latch can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latches during the break state. (See 9.7.3 SIM Break Flag Control Register.)

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

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

## 17.6 IRQ Status and Control Register

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

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

Address: $001A

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

= Unimplemented  R = Reserved

**Figure 17-4. IRQ Status and Control Register (ISCR)**

**IRQF — $\overline{\text{IRQ}}$ Flag Bit**

This read-only status bit is high when the $\overline{\text{IRQ}}$ interrupt is pending.

1 = $\overline{\text{IRQ}}$ interrupt pending
0 = $\overline{\text{IRQ}}$ interrupt not pending

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

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

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

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

1 = $\overline{\text{IRQ}}$ interrupt requests disabled
0 = $\overline{\text{IRQ}}$ interrupt requests enabled

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

This read/write bit controls the triggering sensitivity of the $\overline{\text{IRQ}}$ pin. Reset clears MODE.

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

# Chapter 18
# Serial Communications Interface (SCI)

## 18.1 Introduction

The SCI allows asynchronous communications with peripheral devices and other MCUs.

## 18.2 Features

The SCI module's features include:

- Full Duplex Operation
- Standard Mark/Space Non-Return-to-Zero (NRZ) Format
- 32 Programmable Baud Rates
- Programmable 8-Bit or 9-Bit Character Length
- Separately Enabled Transmitter and Receiver
- Separate Receiver and Transmitter 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

## 18.3 Pin Name Conventions

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

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

SCI I/O lines are implemented by sharing parallel I/O port pins. The full name of an SCI input or output reflects the name of the shared port pin. Table 18-1 shows the full names and the generic names of the SCI I/O pins.The generic pin names appear in the text of this subsection.

**Table 18-1. Pin Name Conventions**

| | | |
|---|---|---|
| **Generic Pin Names** | RxD | TxD |
| **Full Pin Names** | PTE1/SCRxD | PTE0/SCTxD |

## 18.4 Functional Description

Figure 18-1 shows the structure of the SCI module. The SCI allows full-duplex, asynchronous, NRZ serial communication between the MCU and remote devices, including other MCUs. The transmitter and receiver of the SCI operate independently, although they use the same baud rate generator. During normal operation, the CPU monitors the status of the SCI, writes the data to be transmitted, and processes received data.

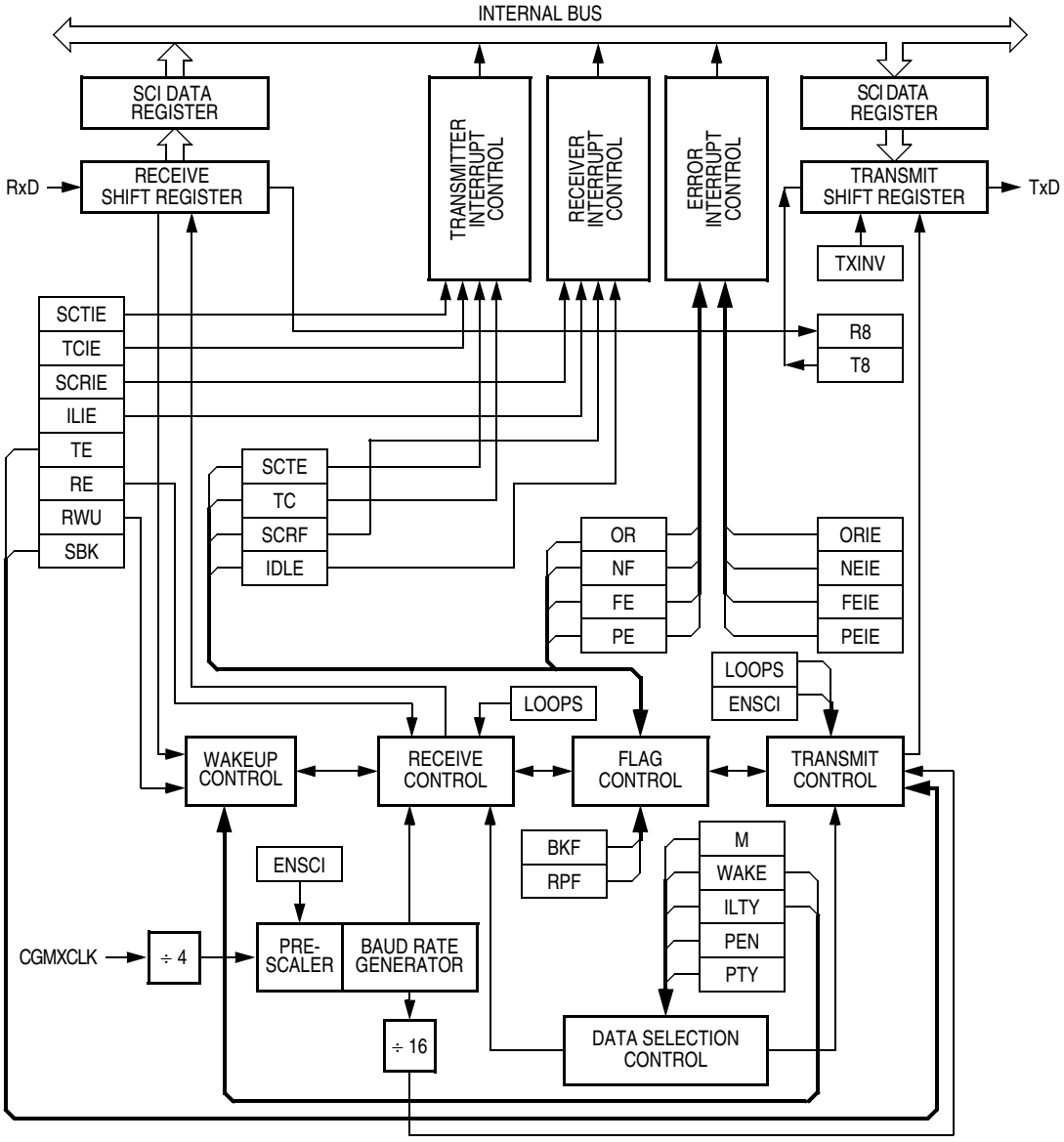



**Figure 18-1. SCI Module Block Diagram**

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

= Unimplemented U = Unaffected R = Reserved

**Figure 18-2. SCI I/O Register Summary**

**Table 18-2. SCI I/O Register Address Summary**

| Register | SCC1 | SCC2 | SCC3 | SCS1 | SCS2 | SCDR | SCBR |
|---|---|---|---|---|---|---|---|
| Address | $0013 | $0014 | $0015 | $0016 | $0017 | $0018 | $0019 |

### 18.4.1 Data Format

The SCI uses the standard non-return-to-zero mark/space data format illustrated in Figure 18-3.

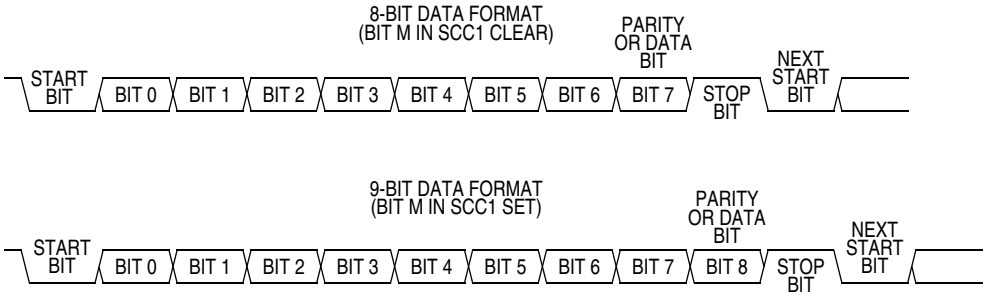



**Figure 18-3. SCI Data Formats**

### 18.4.2 Transmitter

Figure 18-4 shows the structure of the SCI transmitter.

#### *18.4.2.1 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).

#### *18.4.2.2 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 1 to the enable SCI bit (ENSCI) in SCI control register 1 (SCC1).
2. Enable the transmitter by writing a 1 to the transmitter enable bit (TE) in SCI control register 2 (SCC2).
3. Clear the SCI transmitter empty bit (SCTE) 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 1s. After the preamble shifts out, control logic transfers the SCDR data into the transmit shift register. A 0 start bit automatically goes into the least significant bit position of the transmit shift register. A 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, 1. If at any time software clears the ENSCI bit in SCI control register 1 (SCC1), the transmitter and receiver relinquish control of the port E pins.

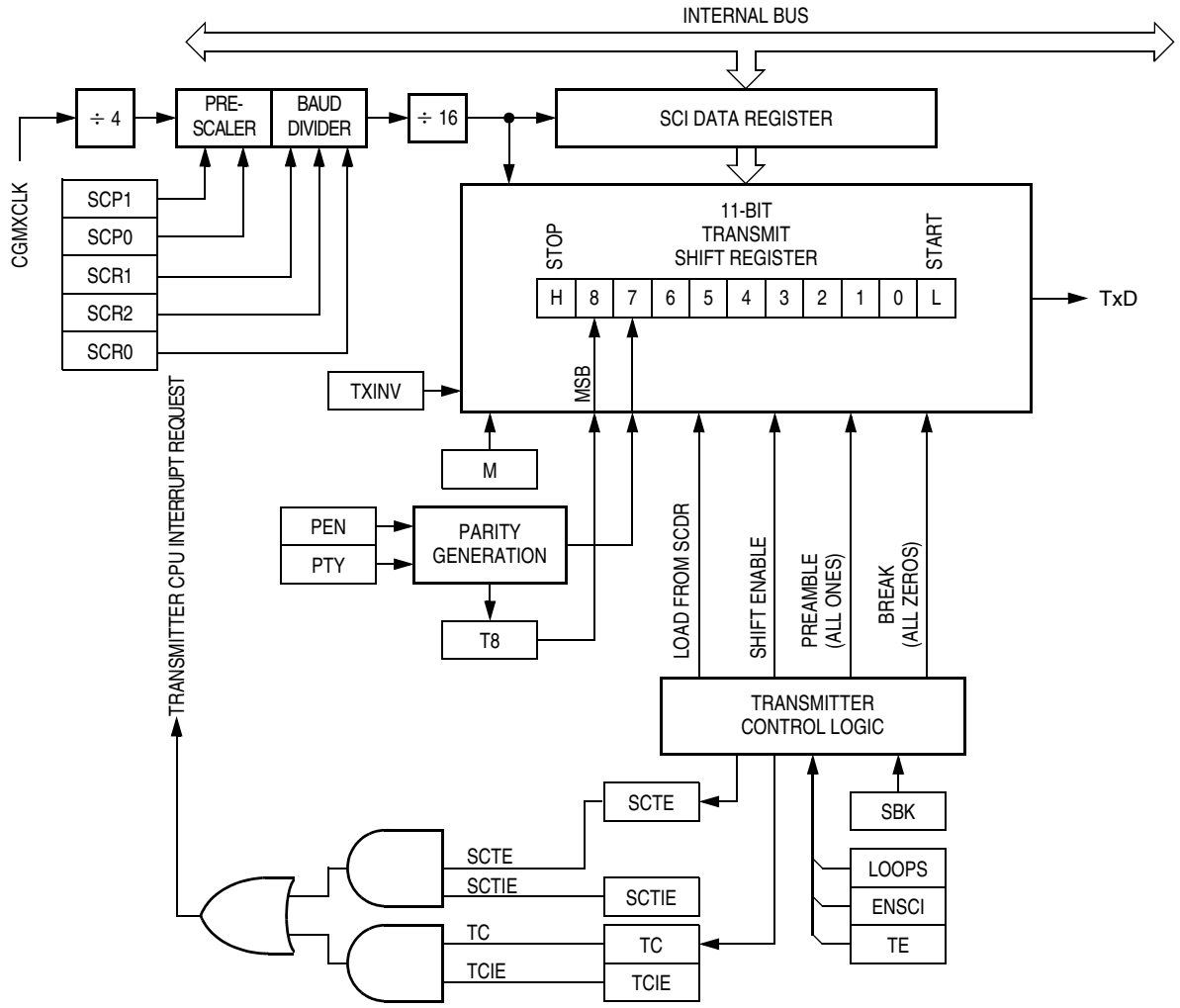



**Figure 18-4. SCI Transmitter**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| SCI Control Register 1 (SCC1) | Read: Write: | LOOPS | ENSCI | TXINV | M | WAKE | ILTY | PEN | PTY |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SCI Control Register 2 (SCC2) | Read: Write: | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SCI Control Register 3 (SCC3) | Read: | R8 | T8 | R | R | ORIE | NEIE | FEIE | PEIE |
| | Write: | | | | | | | | |
| | Reset: | U | U | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented U = Unaffected R = Reserved

**Figure 18-5. SCI Transmitter I/O Register Summary**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| SCI Status Register 1 (SCS1) | Read: | SCTE | TC | SCRF | IDLE | OR | NF | FE | PE |
| | Write: | | | | | | | | |
| | Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| SCI Data Register (SCDR) | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| | Reset: | Unaffected by Reset | | | | | | | |
| SCI Baud Rate Register (SCBR) | Read: | 0 | 0 | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| | Write: | | | | | | | | |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented U = Unaffected R = Reserved

**Figure 18-5. SCI Transmitter I/O Register Summary (Continued)**

**Table 18-3. SCI Transmitter I/O Address Summary**

| Register | SCC1 | SCC2 | SCC3 | SCS1 | SCDR | SCBR |
|---|---|---|---|---|---|---|
| Address | $0013 | $0014 | $0015 | $0016 | $0018 | $0019 |

### *18.4.2.3 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 1. The automatic 1 at the end of a break character guarantees the recognition of the start bit of the next character.

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

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

#### 18.4.2.4 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 a break sequence is followed immediately by an idle character, this SCI design exhibits a condition in which the break character length is reduced by one half bit time. In this instance, the break sequence will consist of a valid start bit, eight or nine data bits (as defined by the M bit in SCC1) of logic 0 and one half data bit length of logic 0 in the stop bit position followed immediately by the idle character. To ensure a break character of the proper length is transmitted, always queue up a byte of data to be transmitted while the final break sequence is in progress.*

**NOTE**

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

*A good time to toggle the TE bit for a queued idle character is when the SCTE bit becomes set and just before writing the next byte to the SCDR.*

#### 18.4.2.5 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 18.8.1 SCI Control Register 1.)

#### 18.4.2.6 Transmitter Interrupts

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

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

### 18.4.3 Receiver

Figure 18-6 shows the structure of the SCI receiver.

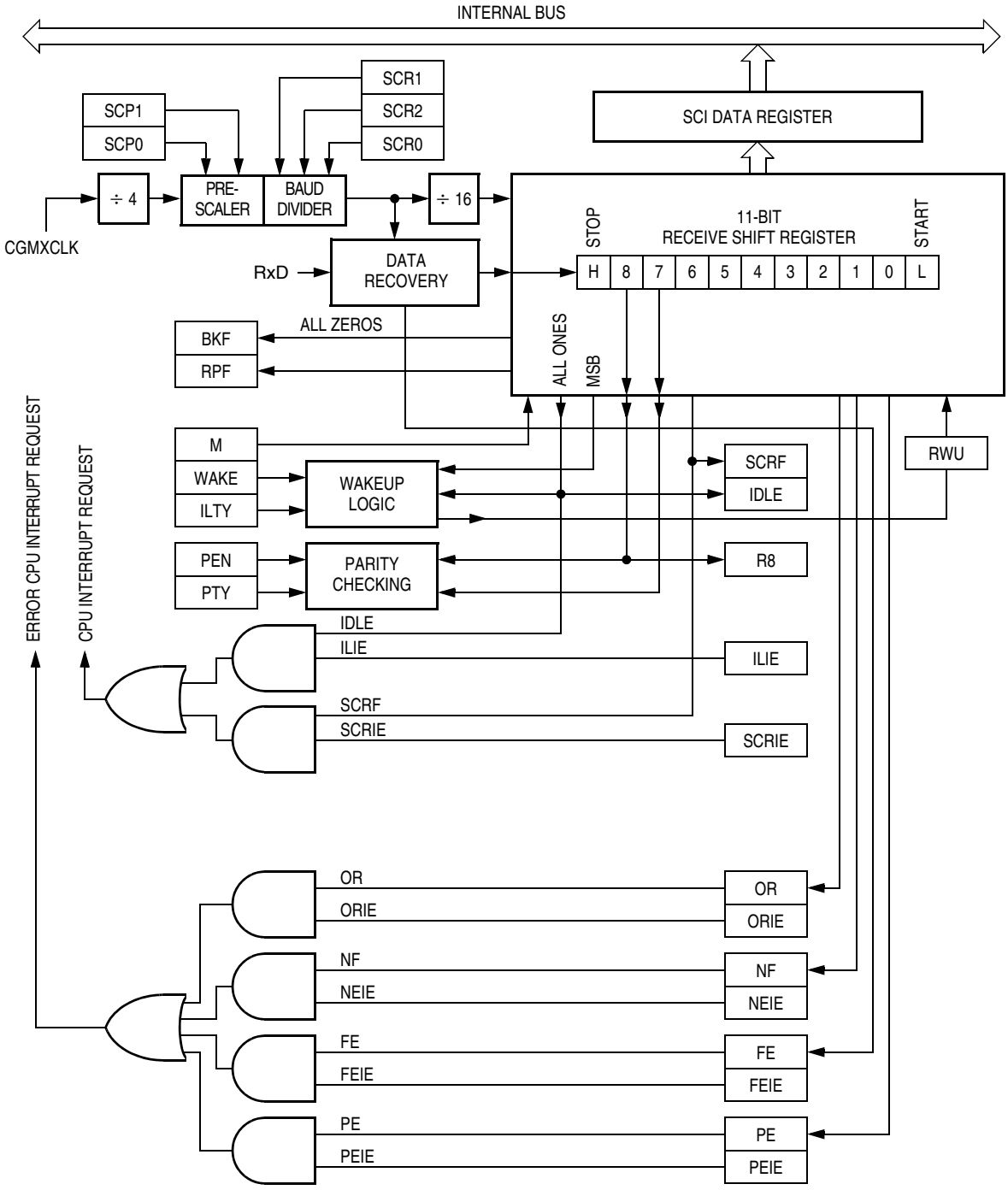



**Figure 18-6. SCI Receiver Block Diagram**

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

= Unimplemented  U = Unaffected  R = Reserved

**Figure 18-7. SCI I/O Receiver Register Summary**

**Table 18-4. SCI Receiver I/O Address Summary**

| **Register** | SCC1 | SCC2 | SCC3 | SCS1 | SCS2 | SCDR | SCBR |
|---|---|---|---|---|---|---|---|
| **Address** | $0013 | $0014 | $0015 | $0016 | $0017 | $0018 | $0019 |

#### 18.4.3.1 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).

#### 18.4.3.2 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.

#### 18.4.3.3 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 18-8):

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

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

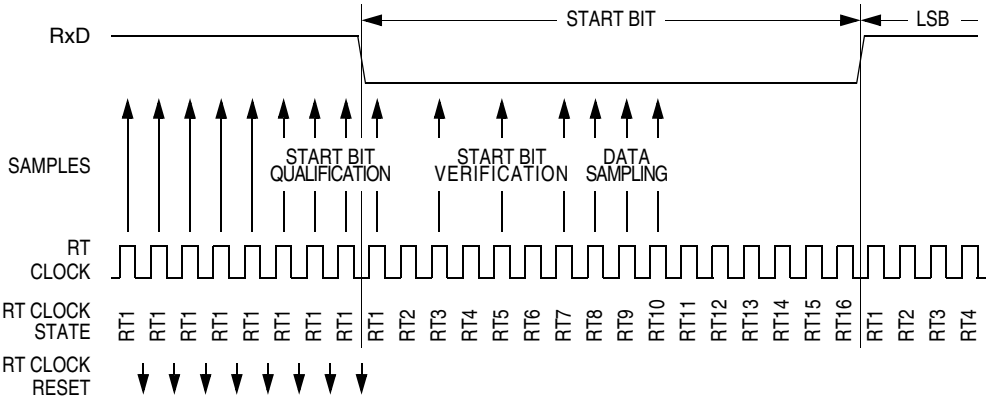



**Figure 18-8. Receiver Data Sampling**

To verify the start bit and to detect noise, data recovery logic takes samples at RT3, RT5, and RT7. Table 18-5 summarizes the results of the start bit verification samples.

**Table 18-5. Start Bit Verification**

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

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

To determine the value of a data bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 18-6 summarizes the results of the data bit samples.

**Table 18-6. Data Bit Recovery**

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

***NOTE***

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

To verify a stop bit and to detect noise, recovery logic takes samples at RT8, RT9, and RT10. Table 18-7 summarizes the results of the stop bit samples.

**Table 18-7. Stop Bit Recovery**

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

#### *18.4.3.4 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.

#### *18.4.3.5 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 18-9 shows how much a slow received character can be misaligned without causing a noise error or a framing error. The slow stop bit begins at RT8 instead of RT1 but arrives in time for the stop bit data samples at RT8, RT9, and RT10.

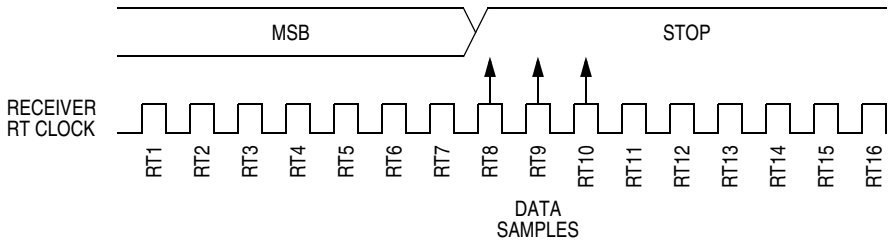



**Figure 18-9. Slow Data**

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.
With the misaligned character shown in Figure 18-9, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 9 bit times × 16 RT cycles + 3 RT cycles = 147 RT cycles.
The maximum percent difference between the receiver count and the transmitter count of a slow 8-bit character with no errors is

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

For a 9-bit character, data sampling of the stop bit takes the receiver
10 bit times × 16 RT cycles + 10 RT cycles = 170 RT cycles.
With the misaligned character shown in Figure 18-9, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 10 bit
times × 16 RT cycles + 3 RT cycles = 163 RT cycles.
The maximum percent difference between the receiver count and the transmitter count of a slow 9-bit character with no errors is

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

### Fast Data Tolerance

Figure 18-10 shows how much a fast received character can be misaligned without causing a noise error or a framing error. The fast stop bit ends at RT10 instead of RT16 but is still there for the stop bit data samples at RT8, RT9, and RT10.

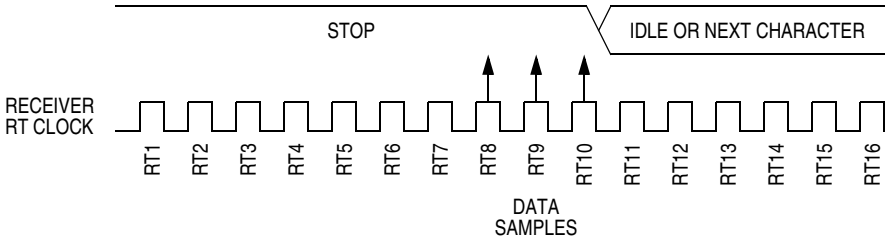



**Figure 18-10. Fast Data**

For an 8-bit character, data sampling of the stop bit takes the receiver
9 bit times × 16 RT cycles + 10 RT cycles = 154 RT cycles.
With the misaligned character shown in Figure 18-10, the receiver counts 154 RT cycles at the point when the count of the transmitting device is 10 bit times × 16 RT cycles = 160 RT cycles.
The maximum percent difference between the receiver count and the transmitter count of a fast 8-bit character with no errors is

$$\left|\frac{154 - 160}{154}\right| \times 100 = 3.90\%.$$

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 18-10, the receiver counts 170 RT cycles at the point when the count of the transmitting device is 11 bit times × 16 RT cycles = 176 RT cycles.

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

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

#### 18.4.3.6 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.*

#### 18.4.3.7 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.

#### 18.4.3.8 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.

## 18.5 Low-Power Modes

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

### 18.5.1 Wait Mode

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

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

### 18.5.2 Stop Mode

The SCI module is inactive in stop mode. The STOP instruction does not affect SCI register states. Any enabled CPU interrupt request from the SCI module does not bring the MCU out of Stop mode. SCI module operation resumes after the MCU exits stop mode.

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

## 18.6 SCI During Break Module Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. (See Chapter 13 Break Module (BRK)).

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.

## 18.7 I/O Signals

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

- PTE0/SCTxD — Transmit data
- PTE1/SCRxD — Receive data

### 18.7.1 PTE0/SCTxD (Transmit Data)

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

### 18.7.2 PTE1/SCRxD (Receive Data)

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

## 18.8 I/O Registers

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

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

### 18.8.1 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 | ILLTY | PEN | PTY |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 18-11. SCI Control Register 1 (SCC1)**

**LOOPS — Loop Mode Select Bit**

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

1 = Loop mode enabled
0 = Normal operation enabled

**ENSCI — Enable SCI Bit**

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

1 = SCI enabled
0 = SCI disabled

**TXINV — Transmit Inversion Bit**

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

1 = Transmitter output inverted
0 = Transmitter output not inverted

***NOTE***

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

**M — Mode (Character Length) Bit**

This read/write bit determines whether SCI characters are eight or nine bits long. (See Table 18-8).The ninth bit can serve as an extra stop bit, as a receiver wakeup signal, or as a parity bit. Reset clears the M bit.

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

**WAKE — Wakeup Condition Bit**

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

1 = Address mark wakeup
0 = Idle line wakeup

**ILTY — Idle Line Type Bit**

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

1 = Idle character bit count begins after stop bit

0 = Idle character bit count begins after start bit

**PEN — Parity Enable Bit**

This read/write bit enables the SCI parity function. (See Table 18-8). When enabled, the parity function inserts a parity bit in the most significant bit position. (See Table 18-7). 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 18-8). Reset clears the PTY bit.

1 = Odd parity
0 = Even parity

***NOTE***

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

**Table 18-8. Character Format Selection**

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

## 18.8.2 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 18-12. SCI Control Register 2 (SCC2)**

**SCTIE — SCI Transmit Interrupt Enable Bit**

This read/write bit enables the SCTE bit to generate SCI transmitter CPU interrupt requests. Setting the SCTIE bit in SCC3 enables the SCTE bit to generate 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. Setting the SCRIE bit in SCC3 enables the SCRF bit to generate 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 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 (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 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.*

### 18.8.3 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 the following interrupts:
  - Receiver overrun interrupts
  - Noise error interrupts
  - Framing error interrupts
  - Parity error interrupts

Address: $0015

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

= Unimplemented  R = Reserved  U = Unaffected

**Figure 18-13. SCI Control Register 3 (SCC3)**

**R8 — Received Bit 8**

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

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

**T8 — Transmitted Bit 8**

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

**ORIE — Receiver Overrun Interrupt Enable Bit**

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

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

**NEIE — Receiver Noise Error Interrupt Enable Bit**

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

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

**FEIE — Receiver Framing Error Interrupt Enable Bit**

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

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

**PEIE — Receiver Parity Error Interrupt Enable Bit**

This read/write bit enables SCI receiver CPU interrupt requests generated by the parity error bit, PE. Reset clears PEIE.

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

### 18.8.4 SCI Status Register 1

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

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

Address: $0016

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

= Unimplemented

**Figure 18-14. SCI Status Register 1 (SCS1)**

**SCTE — SCI Transmitter Empty Bit**

This clearable, read-only bit is set when the SCDR transfers a character to the transmit shift register. SCTE can generate an SCI transmitter CPU interrupt request. 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 cleared automatically when data, preamble, or break is queued and ready to be sent. There may be up to 1.5 transmitter clocks of latency between queueing data, preamble, and break and the transmission actually starting. Reset sets the TC bit.

1 = No transmission in progress
0 = Transmission in progress

**SCRF — SCI Receiver Full Bit**

This clearable, read-only bit is set when the data in the receive shift register transfers to the SCI data register. SCRF can generate an SCI receiver CPU interrupt request. When the SCRIE bit in SCC2 is set the 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 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 18-15 shows the normal flag-clearing sequence and an example of an overrun caused by a delayed flag-clearing sequence. The delayed read of SCDR does not clear the OR bit because OR was not set when SCS1 was read. Byte 2 caused the overrun and is lost. The next flag-clearing sequence reads byte 3 in the SCDR instead of byte 2.

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

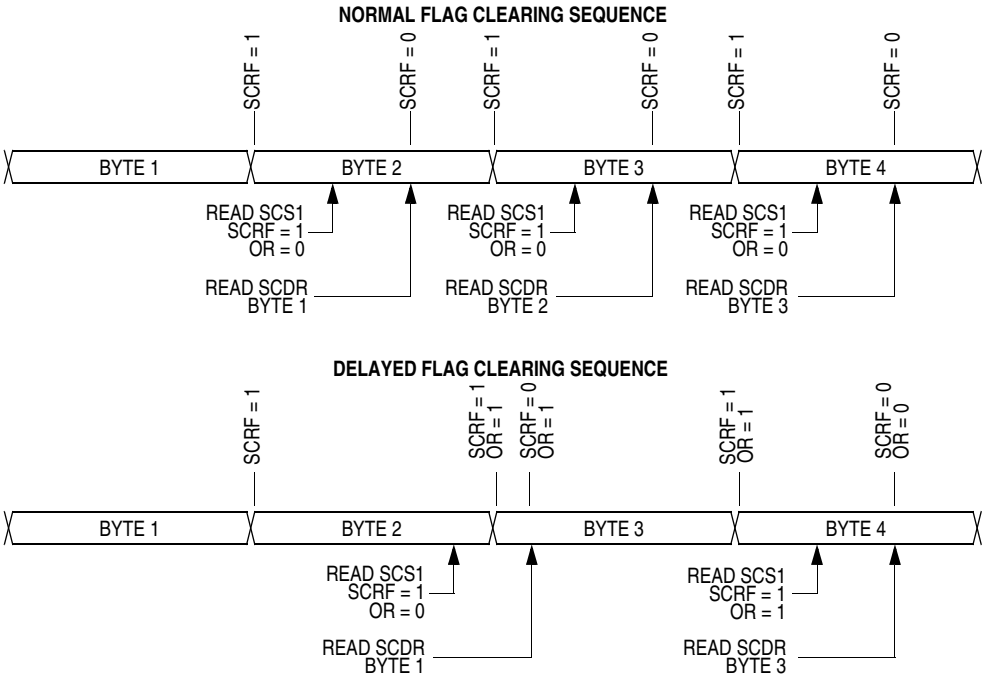



**Figure 18-15. Flag Clearing Sequence**

### NF — Receiver Noise Flag Bit

This clearable, read-only bit is set when the SCI detects noise on the RxD pin. NF generates an 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

**PE — Receiver Parity Error Bit**

This clearable, read-only bit is set when the SCI detects a parity error in incoming data. PE generates a SCI receiver 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

### 18.8.5 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: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 18-16. SCI Status Register 2 (SCS2)**

**BKF — Break Flag Bit**

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

### 18.8.6 SCI Data Register

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

Address: $0018

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

**Figure 18-17. SCI Data Register (SCDR)**

**R7/T7:R0/T0 — Receive/Transmit Data Bits**

Reading address $0018 accesses the read-only received data bits, R7:R0. Writing to address $0018 writes the data to be transmitted, T7:T0. Reset has no effect on the SCI data register.

***NOTE***

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

### 18.8.7 SCI Baud Rate Register

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

Address: $0019

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

= Unimplemented  R = Reserved

**Figure 18-18. SCI Baud Rate Register (SCBR)**

**SCP1 and SCP0 — SCI Baud Rate Prescaler Bits**

These read/write bits select the baud rate prescaler divisor as shown in Table 18-9. Reset clears SCP1 and SCP0.

**Table 18-9. SCI Baud Rate Prescaling**

| SCP[1:0] | Prescaler Divisor (PD) |
|---|---|
| 00 | 1 |
| 01 | 3 |
| 10 | 4 |
| 11 | 13 |

**SCR2–SCR0 — SCI Baud Rate Select Bits**

These read/write bits select the SCI baud rate divisor as shown in Table 18-10. Reset clears SCR2–SCR0.

**Table 18-10. SCI Baud Rate Selection**

| SCR[2:1:0] | Baud Rate Divisor (BD) |
|---|---|
| 000 | 1 |
| 001 | 2 |
| 010 | 4 |
| 011 | 8 |
| 100 | 16 |
| 101 | 32 |
| 110 | 64 |
| 111 | 128 |

Use the following formula to calculate the SCI baud rate:

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

where:

$f_{Crystal}$ = crystal frequency
PD = prescaler divisor
BD = baud rate divisor

Table 18-11 shows the SCI baud rates that can be generated with a 4.9152-MHz crystal.

**Table 18-11. SCI Baud Rate Selection Examples**

| SCP[1:0] | Prescaler Divisor (PD) | SCR[2:1:0] | Baud Rate Divisor (BD) | Baud Rate ($f_{Crystal}$ = 4.9152 MHz) |
|---|---|---|---|---|
| 00 | 1 | 000 | 1 | 76,800 |
| 00 | 1 | 001 | 2 | 38,400 |
| 00 | 1 | 010 | 4 | 19,200 |
| 00 | 1 | 011 | 8 | 9600 |
| 00 | 1 | 100 | 16 | 4800 |
| 00 | 1 | 101 | 32 | 2400 |
| 00 | 1 | 110 | 64 | 1200 |
| 00 | 1 | 111 | 128 | 600 |
| 01 | 3 | 000 | 1 | 25,600 |
| 01 | 3 | 001 | 2 | 12,800 |
| 01 | 3 | 010 | 4 | 6400 |
| 01 | 3 | 011 | 8 | 3200 |
| 01 | 3 | 100 | 16 | 1600 |
| 01 | 3 | 101 | 32 | 800 |
| 01 | 3 | 110 | 64 | 400 |
| 01 | 3 | 111 | 128 | 200 |
| 10 | 4 | 000 | 1 | 19,200 |
| 10 | 4 | 001 | 2 | 9600 |
| 10 | 4 | 010 | 4 | 4800 |
| 10 | 4 | 011 | 8 | 2400 |
| 10 | 4 | 100 | 16 | 1200 |
| 10 | 4 | 101 | 32 | 600 |
| 10 | 4 | 110 | 64 | 300 |
| 10 | 4 | 111 | 128 | 150 |
| 11 | 13 | 000 | 1 | 5908 |
| 11 | 13 | 001 | 2 | 2954 |
| 11 | 13 | 010 | 4 | 1477 |
| 11 | 13 | 011 | 8 | 739 |
| 11 | 13 | 100 | 16 | 369 |
| 11 | 13 | 101 | 32 | 185 |
| 11 | 13 | 110 | 64 | 92 |
| 11 | 13 | 111 | 128 | 46 |

# Chapter 19
# Serial Peripheral Interface (SPI)

## 19.1 Introduction

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

## 19.2 Features

Features of the SPI module include:

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

## 19.3 Pin Name and Register Name Conventions

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

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

The SPI shares four I/O pins with a parallel I/O port. The full name of an SPI pin reflects the name of the shared port pin. Table 19-1 shows the full names of the SPI I/O pins. The generic pin names appear in the text that follows.

**Table 19-1. Pin Name Conventions**

| **SPI Generic Pin Name** | MISO | MOSI | $\overline{SS}$ | SPSCK |
|---|---|---|---|---|
| **Full SPI Pin Name** | PTE5/MISO | PTE6/MOSI | PTE4/$\overline{SS}$ | PTE7/SPSCK |

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

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

Table 19-2 shows the names and the addresses of the SPI I/O registers.

**Table 19-2. I/O Register Addresses**

| Register Name | Address |
|---|---|
| SPI Control Register (SPCR) | $0010 |
| SPI Status and Control Register (SPSCR) | $0011 |
| SPI Data Register (SPDR) | $0012 |

## 19.4 Functional Description

Figure 19-1 summarizes the SPI I/O registers and Figure 19-2 shows the structure of the SPI module.

| Addr | Register Name | R/W | 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 | | | | | | | |

R = Reserved  = Unimplemented

**Figure 19-1. SPI I/O Register Summary**

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

The following paragraphs describe the operation of the SPI module.

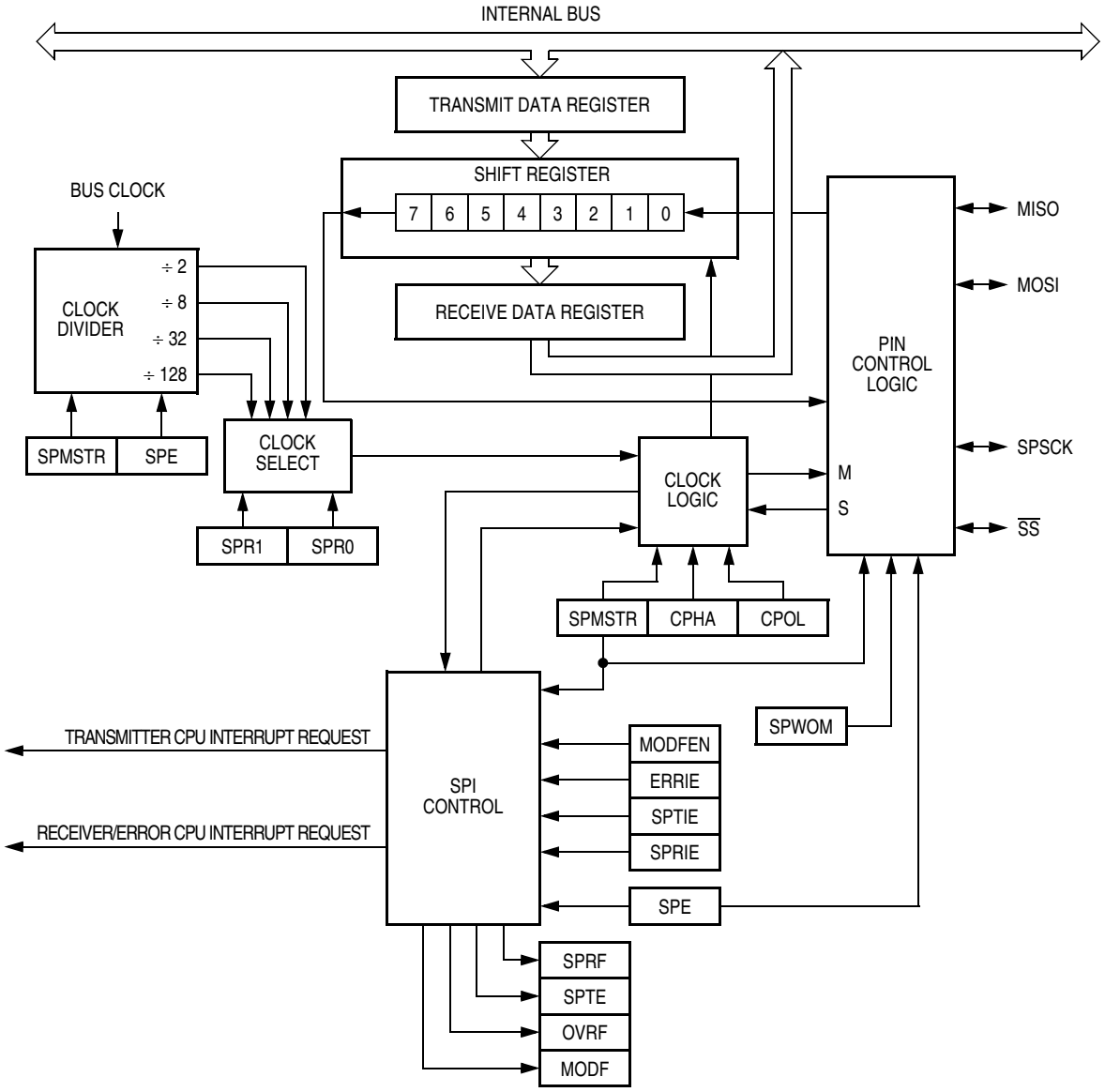



**Figure 19-2. SPI Module Block Diagram**

### 19.4.1 Master Mode

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

***NOTE***

*Configure the SPI modules as master and slave before enabling them. Enable the master SPI before enabling the slave SPI. Disable the slave SPI before disabling the master SPI. See 19.13.1 SPI Control Register.*

Only a master SPI module can initiate transmissions. Software begins the transmission from a master SPI module by writing to the SPI data register. If the shift register is empty, the byte immediately transfers to the shift register, setting the SPI transmitter empty bit, SPTE (SPSCR $0011). The byte begins shifting out on the MOSI pin under the control of the serial clock. (See Table 19-3).

The SPR1 and SPR0 bits control the baud rate generator and determine the speed of the shift register. (See 19.13.2 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.

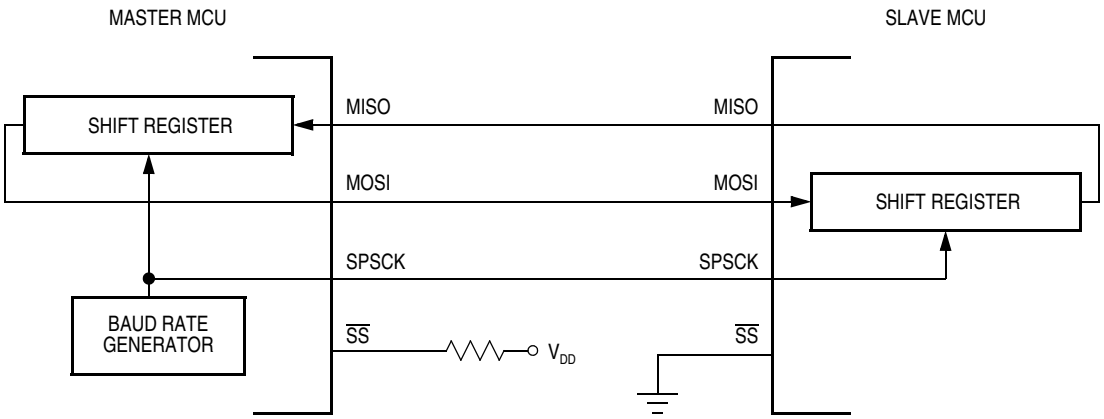



**Figure 19-3. Full-Duplex Master-Slave Connections**

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

### 19.4.2 Slave Mode

The SPI operates in slave mode when the SPMSTR bit (SPCR, $0010) is clear. In slave mode the SPSCK pin is the input for the serial clock from the master MCU. Before a data transmission occurs, the $\overline{SS}$ pin of the slave MCU must be low. $\overline{SS}$ must remain low until the transmission is complete. (See 19.6.2 Mode Fault Error).

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

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

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

When the clock phase bit (CPHA) is set, the first edge of SPSCK starts a transmission. When CPHA is clear, the falling edge of $\overline{SS}$ starts a transmission. See 19.5 Transmission Formats.

If the write to the data register is late, the SPI transmits the data already in the shift register from the previous transmission.

***NOTE***

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

## 19.5 Transmission Formats

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

### 19.5.1 Clock Phase and Polarity Controls

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

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

***NOTE***

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

### 19.5.2 Transmission Format When CPHA = 0

Figure 19-4 shows an SPI transmission in which CPHA (SPCR) is logic 0. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ 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 low, 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 19.6.2 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 transmission. The $\overline{SS}$ pin must be toggled high and then low again between each byte transmitted.

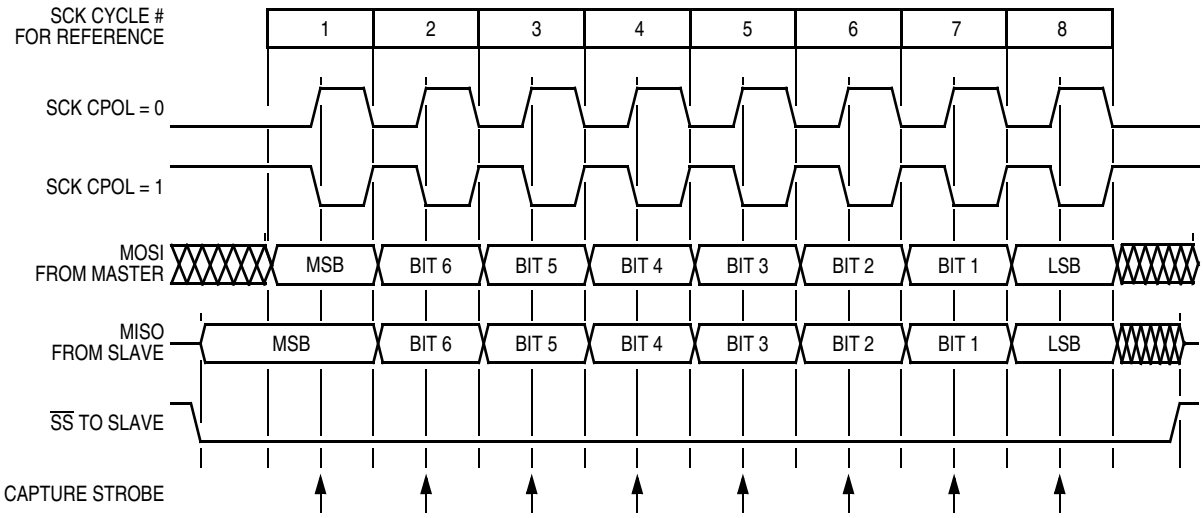



**Figure 19-4. Transmission Format (CPHA = 0)**

### 19.5.3 Transmission Format When CPHA = 1

Figure 19-5 shows an SPI transmission in which CPHA (SPCR) is logic 1. The figure should not be used as a replacement for data sheet parametric information. Two waveforms are shown for SCK: one for CPOL = 0 and another for CPOL = 1. The diagram may be interpreted as a master or slave timing diagram since the serial clock (SCK), master in/slave out (MISO), and master out/slave in (MOSI) pins are directly connected between the master and the slave. The MISO signal is the output from the slave, and the MOSI signal is the output from the master. The $\overline{SS}$ line is the slave select input to the slave. The slave SPI drives its MISO output only when its slave select input ($\overline{SS}$) is low, 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 19.6.2 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.

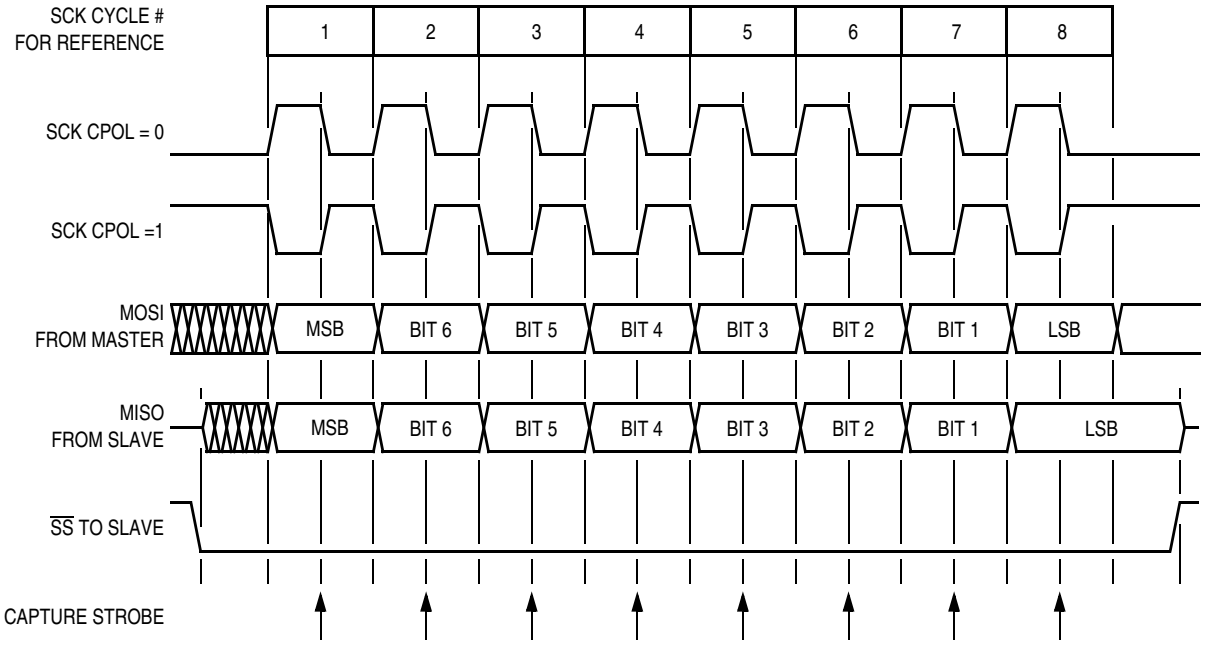



**Figure 19-5. Transmission Format (CPHA = 1)**

### 19.5.4 Transmission Initiation Latency

When the SPI is configured as a master (SPMSTR = 1), transmissions are started by a software write to the SPDR ($0012). CPHA has no effect on the delay to the start of the transmission, but it does affect the initial state of the SCK signal. When CPHA = 0, the SCK signal remains inactive for the first half of the first SCK cycle. When CPHA = 1, the first SCK cycle begins with an edge on the SCK line from its inactive to its active level. The SPI clock rate (selected by SPR1–SPR0) affects the delay from the write to SPDR and the start of the SPI transmission. (See Figure 19-6). The internal SPI clock in the master is a free-running derivative of the internal MCU clock. It is only enabled when both the SPE and SPMSTR bits (SPCR) are set to conserve power. SCK edges occur half way through the low time of the internal MCU clock. Since the SPI clock is free-running, it is uncertain where the write to the SPDR will occur relative to the slower SCK. This uncertainty causes the variation in the initiation delay shown in Figure 19-6. This

delay will be no longer than a single SPI bit time. That is, the maximum delay between the write to SPDR and the start of the SPI transmission is two MCU bus cycles for DIV2, eight MCU bus cycles for DIV8, 32 MCU bus cycles for DIV32, and 128 MCU bus cycles for DIV128.

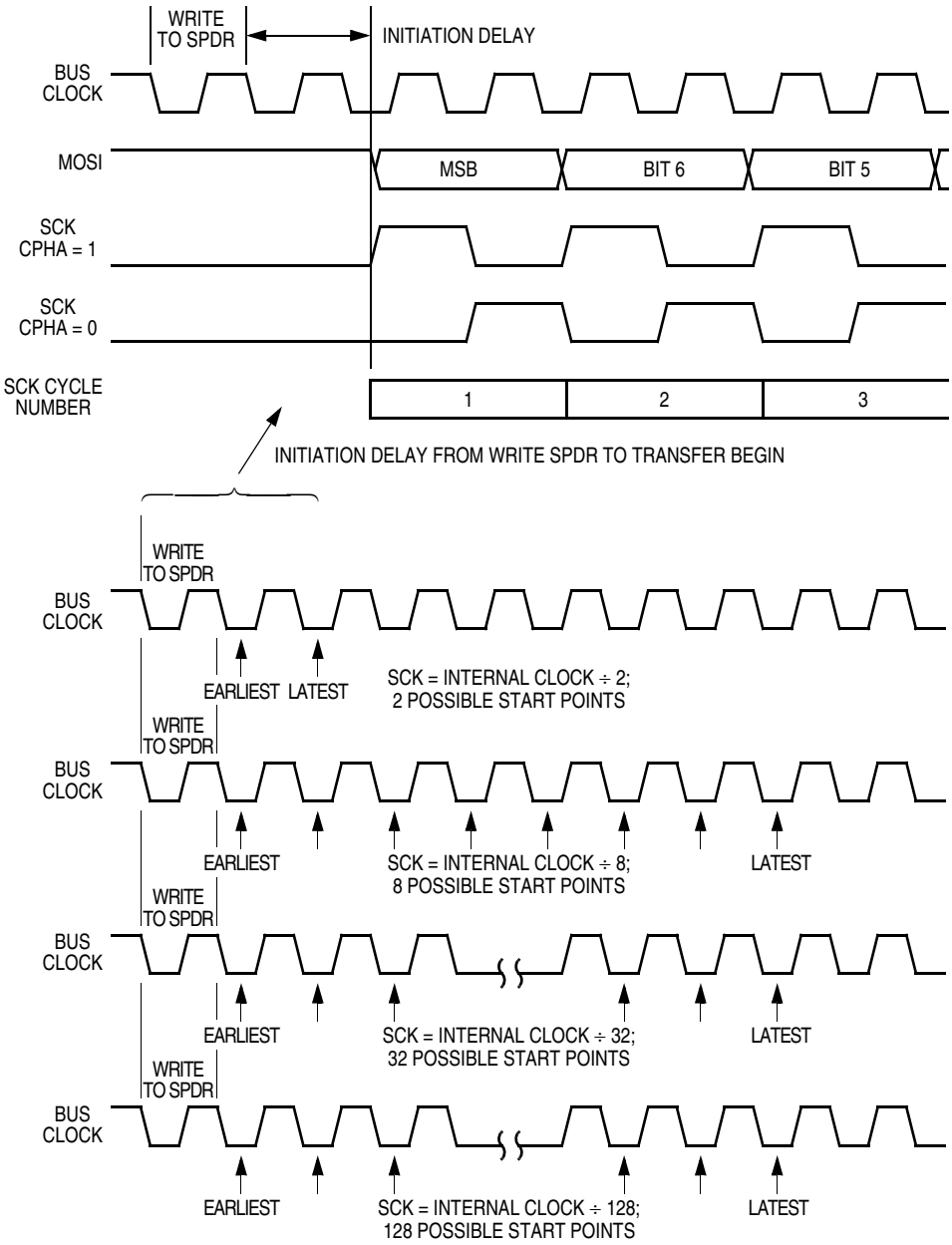



**Figure 19-6. Transmission Start Delay (Master)**

## 19.6 Error Conditions

Two flags signal SPI error conditions:

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

### 19.6.1 Overflow Error

The overflow flag (OVRF in SPSCR) becomes set if the SPI receive data register still has unread data from a previous transmission when the capture strobe of bit 1 of the next transmission occurs. (See Figure 19-4 and Figure 19-5.) If an overflow occurs, the data being received is not transferred to the receive data register so that the unread data can still be read. Therefore, an overflow error always indicates the loss of data.

OVRF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 19-9). It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

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

If the CPU SPRF interrupt is enabled and the OVRF interrupt is not, watch for an overflow condition. Figure 19-7 shows how it is possible to miss an overflow.

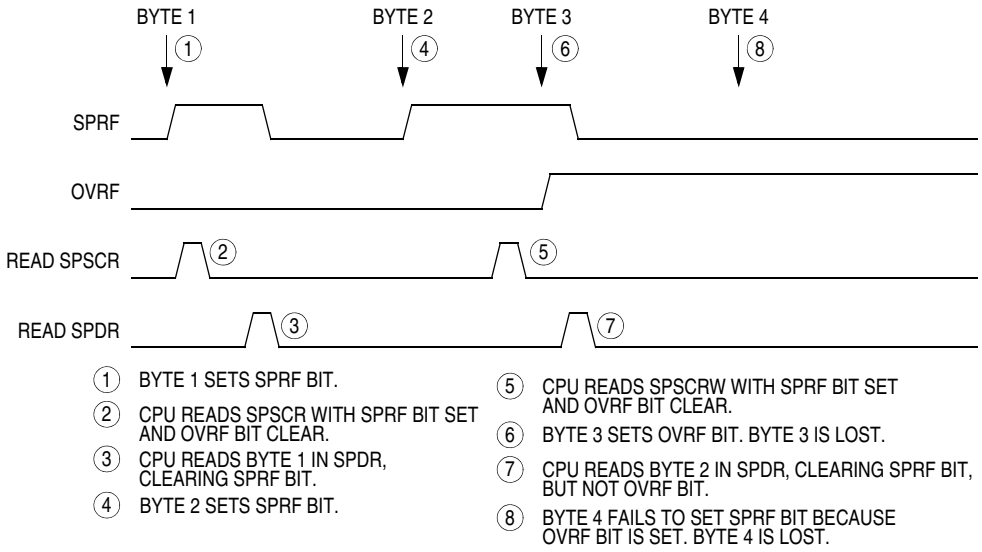



**Figure 19-7. Missed Read of Overflow Condition**

The first part of Figure 19-7 shows how to read the SPSCR and SPDR to clear the SPRF without problems. However, as illustrated by the second transmission example, the OVRF flag can be set in between the time that SPSCR and SPDR are read.

In this case, an overflow can be easily missed. Since no more SPRF interrupts can be generated until this OVRF is serviced, it will not be obvious that bytes are being lost as more transmissions are completed. To prevent this, either enable the OVRF interrupt or do another read of the SPSCR after the read of the SPDR. This ensures that the OVRF was not set before the SPRF was cleared and that future transmissions will complete with an SPRF interrupt. Figure 19-8 illustrates this process. Generally, to avoid this second SPSCR read, enable the OVRF to the CPU by setting the ERRIE bit (SPSCR).

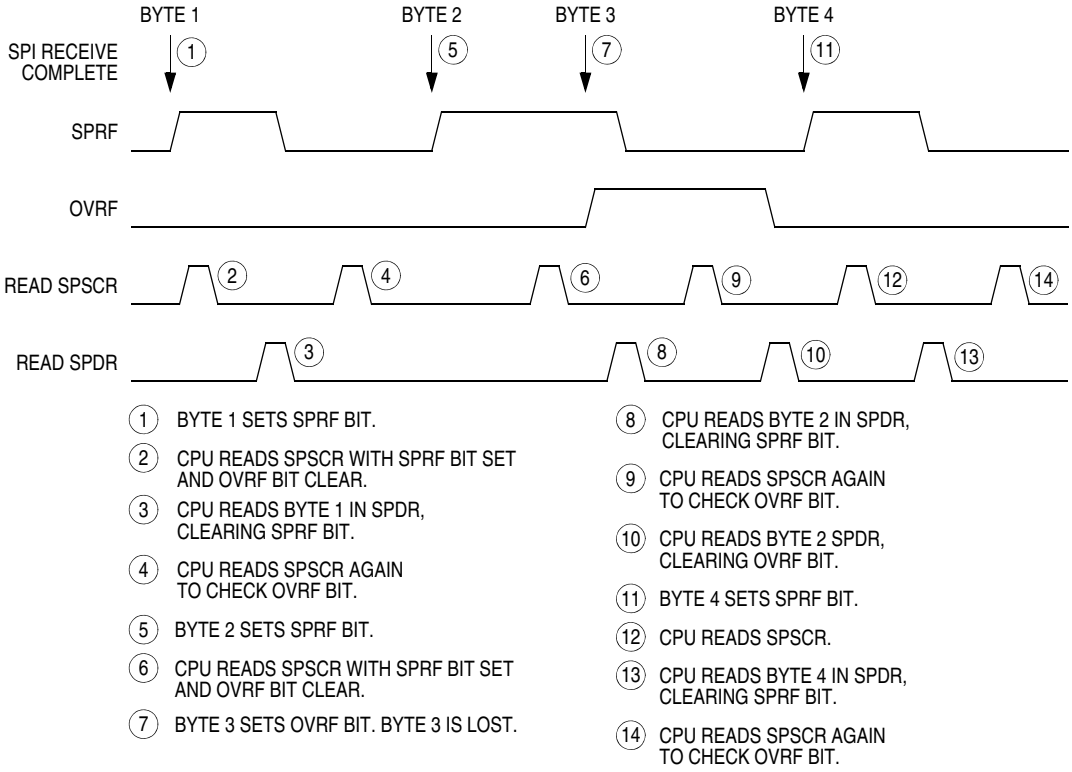



**Figure 19-8. Clearing SPRF When OVRF Interrupt Is Not Enabled**

### 19.6.2 Mode Fault Error

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

MODF generates a receiver/error CPU interrupt request if the error interrupt enable bit (ERRIE in SPSCR) is also set. The SPRF, MODF, and OVRF interrupts share the same CPU interrupt vector. MODF and OVRF can generate a receiver/error CPU interrupt request. (See Figure 19-9). It is not possible to enable only MODF or OVRF to generate a receiver/error CPU interrupt request. However, leaving MODFEN low prevents MODF from being set.

In a master SPI with the mode fault enable bit (MODFEN) set, the mode fault flag (MODF) is set if $\overline{SS}$ goes low. 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 data direction register (DDR) bits associated with the SPI shared port pins.*

***NOTE***

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

When configured as a slave (SPMSTR = 0), the MODF flag is set if $\overline{SS}$ goes high during a transmission. When CPHA = 0, a transmission begins when $\overline{SS}$ goes low and ends once the incoming SPSCK returns to its idle level after the shift of the eighth data bit. When CPHA = 1, the transmission begins when the SPSCK leaves its idle level and $\overline{SS}$ is already low. The transmission continues until the SPSCK returns to its IDLE level after the shift of the last data bit. (See 19.5 Transmission Formats).

***NOTE***

*When CPHA = 0, a MODF occurs if a slave is selected ($\overline{SS}$ is low) and later deselected ($\overline{SS}$ is high) even if no SPSCK is sent to that slave. This happens because $\overline{SS}$ at 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 deselected with no transmission occurring. Therefore, MODF does not occur since a transmission was never begun.*

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

***NOTE***

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

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

## 19.7 Interrupts

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

**Table 19-3. SPI Interrupts**

| Flag | Request |
|---|---|
| SPTE (Transmitter Empty) | SPI Transmitter CPU Interrupt Request (SPTIE = 1) |
| SPRF (Receiver Full) | SPI Receiver CPU Interrupt Request (SPRIE = 1) |
| OVRF (Overflow) | SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1) |
| MODF (Mode Fault) | SPI Receiver/Error Interrupt Request (SPRIE = 1, ERRIE = 1, MODFEN = 1) |

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

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

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

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

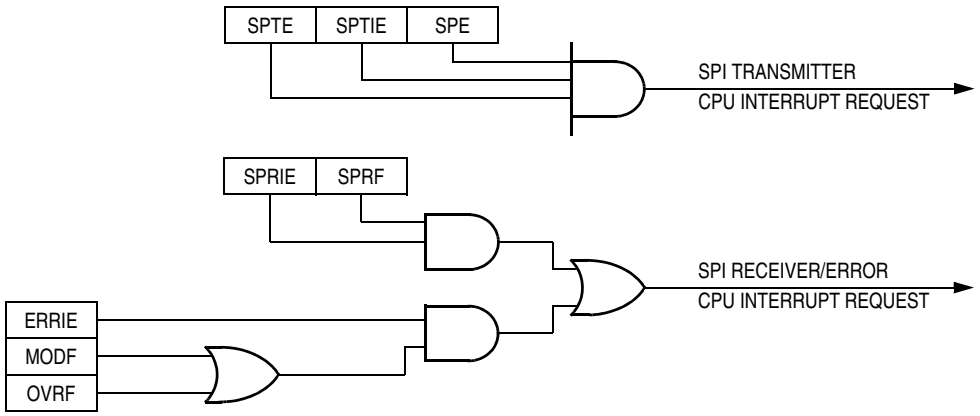



**Figure 19-9. SPI Interrupt Request Generation**

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

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

## 19.8 Queuing Transmission Data

The double-buffered transmit data register allows a data byte to be queued and transmitted. For an SPI configured as a master, a queued data byte is transmitted immediately after the previous transmission has completed. The SPI transmitter empty flag (SPTE in SPSCR) indicates when the transmit data buffer is ready to accept new data. Write to the SPI data register only when the SPTE bit is high. Figure 19-10 shows the timing associated with doing back-to-back transmissions with the SPI (SPSCK has CPHA:CPOL = 1:0).

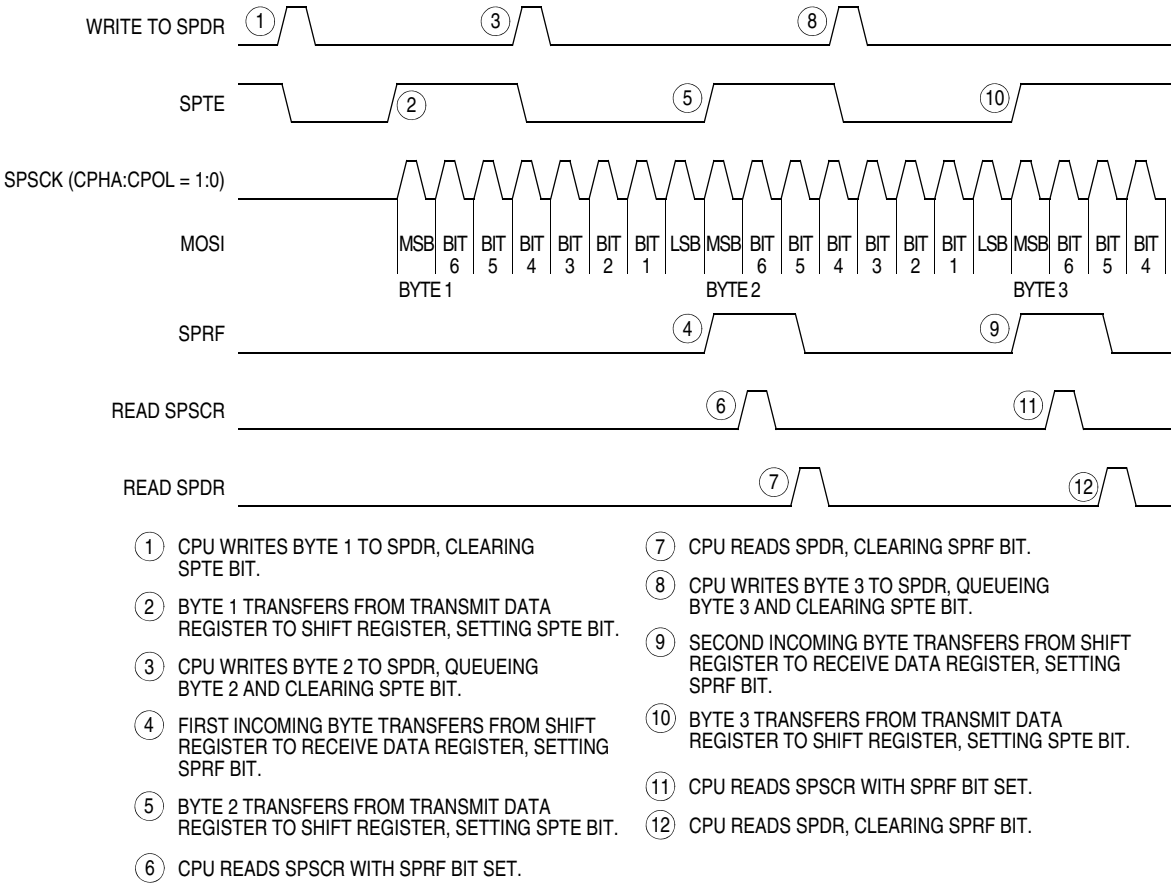



**Figure 19-10. SPRF/SPTE CPU Interrupt Timing**

For a slave, the transmit data buffer allows back-to-back transmissions to occur without the slave having to time the write of its data between the transmissions. Also, if no new data is written to the data buffer, the last value contained in the shift register will be the next data word transmitted.

## 19.9 Resetting the SPI

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

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

The following additional items are reset only by a system reset:

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

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

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

## 19.10 Low-Power Modes

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

### 19.10.1 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 19.7 Interrupts).

### 19.10.2 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 the MCU exits stop mode. If stop mode is exited by reset, any transfer in progress is aborted and the SPI is reset.

## 19.11 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, $FE03) enables software to clear status bits during the break state. (See 9.7.3 SIM Break Flag Control Register).

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

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

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

## 19.12 I/O Signals

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

- MISO — Data received
- MOSI — Data transmitted
- SPSCK — Serial clock
- $\overline{SS}$ — Slave select
- $V_{SS}$ — Clock ground

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

### 19.12.1 MISO (Master In/Slave Out)

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

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

### 19.12.2 MOSI (Master Out/Slave In)

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

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

### 19.12.3 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.

### 19.12.4 $\overline{SS}$ (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. 19.5 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 throughout the transmission for the CPHA = 1 format. See Figure 19-11.

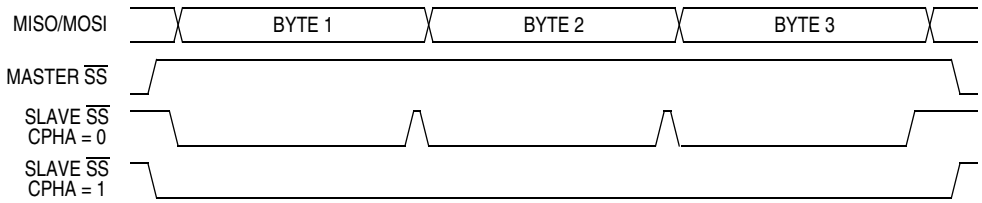



**Figure 19-11. CPHA/$\overline{SS}$ Timing**

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 19.13.2 SPI Status and Control Register).

***NOTE***

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

When an SPI is configured as a master, the $\overline{SS}$ input can be used in conjunction with the MODF flag to prevent multiple masters from driving MOSI and SPSCK. (See 19.6.2 Mode Fault Error). For the state of the $\overline{SS}$ pin to set the MODF flag, the MODFEN bit in the SPSCK register must be set. If the MODFEN bit is low for an SPI master, the $\overline{SS}$ pin can be used as a general-purpose I/O under the control of the data direction register of the shared I/O port. With MODFEN high, it is an input-only pin to the SPI regardless of the state of the data direction register of the shared I/O port.

The CPU can always read the state of the $\overline{SS}$ pin by configuring the appropriate pin as an input and reading the data register. (See Table 19-4).

**Table 19-4. SPI Configuration**

| SPE | SPMSTR | MODFEN | SPI Configuration | State of $\overline{SS}$ Logic |
|---|---|---|---|---|
| 0 | X | X | Not Enabled | General-Purpose I/O; $\overline{SS}$ Ignored by SPI |
| 1 | 0 | X | Slave | Input-Only to SPI |
| 1 | 1 | 0 | Master without MODF | General-Purpose I/O; $\overline{SS}$ Ignored by SPI |
| 1 | 1 | 1 | Master with MODF | Input-Only to SPI |

X = don't care

### 19.12.5 $V_{SS}$ (Clock Ground)

$V_{SS}$ is the ground return for the serial clock pin, SPSCK, and the ground for the port output buffers. To reduce the ground return path loop and minimize radio frequency (RF) emissions, connect the ground pin of the slave to the $V_{SS}$ pin.

## 19.13 I/O Registers

Three registers control and monitor SPI operation:

- SPI control register (SPCR $0010)
- SPI status and control register (SPSCR $0011)
- SPI data register (SPDR $0012)

### 19.13.1 SPI Control Register

The SPI control register:

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

Address: $0010

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| Reset: | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

R = Reserved

**Figure 19-12. 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 19-4 and Figure 19-5.) To transmit data between SPI modules, the SPI modules must have identical CPOL bits. Reset clears the CPOL bit.

**CPHA — Clock Phase Bit**

This read/write bit controls the timing relationship between the serial clock and SPI data. (See Figure 19-4 and Figure 19-5.) To transmit data between SPI modules, the SPI modules must have identical CPHA bits. When CPHA = 0, the $\overline{SS}$ pin of the slave SPI module must be set to logic 1 between bytes. (See Figure 19-11). Reset sets the CPHA bit.

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 data register. Therefore, the slave data register must be loaded with the desired transmit data before the falling edge of $\overline{SS}$. Any data written after the falling edge is stored in the data register and transferred to the shift register at the current transmission.

When CPHA = 1 for a slave, the first edge of the SPSCK indicates the beginning of the transmission. The same applies when $\overline{SS}$ is high for a slave. The MISO pin is held in a high-impedance state, and the incoming SPSCK is ignored. In certain cases, it may also cause the MODF flag to be set. (See 19.6.2 Mode Fault Error). A 1 on the $\overline{SS}$ pin does not in any way affect the state of the SPI state machine.

**SPWOM — SPI Wired-OR Mode Bit**

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

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

**SPE — SPI Enable Bit**

This read/write bit enables the SPI module. Clearing SPE causes a partial reset of the SPI (see 19.9 Resetting the SPI). Reset clears the SPE bit.

1 = SPI module enabled
0 = SPI module disabled

**SPTIE — SPI Transmit Interrupt Enable Bit**

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

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

### 19.13.2 SPI Status and Control Register

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

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

R = Reserved  = Unimplemented

**Figure 19-13. SPI Status and Control Register (SPSCR)**

**SPRF — SPI Receiver Full Bit**

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

Reset clears the SPRF bit.

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

**ERRIE — Error Interrupt Enable Bit**

This read-only bit enables the MODF and OVRF flags to generate CPU interrupt requests. Reset clears the ERRIE bit.

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

**OVRF — Overflow Bit**

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

1 = Overflow
0 = No overflow

**MODF — Mode Fault Bit**

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

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

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

**MODFEN — Mode Fault Enable Bit**

This read/write bit, when set to 1, allows the MODF flag to be set. If the MODF flag is set, clearing the MODFEN does not clear the MODF flag. If the SPI is enabled as a master and the MODFEN bit is low, then the $\overline{SS}$ 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 19.12.4 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 19.6.2 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 19-5. SPR1 and SPR0 have no effect in slave mode. Reset clears SPR1 and SPR0.

**Table 19-5. SPI Master Baud Rate Selection**

| SPR1: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), see Chapter 10 Clock Generator Module (CGM).
BD = baud rate divisor

### 19.13.3 SPI Data Register

The SPI data register is the read/write buffer for the receive data register and the transmit data register. Writing to the SPI data register writes data into the transmit data register. Reading the SPI data register reads data from the receive data register. The transmit data and receive data registers are separate buffers that can contain different values. (See Figure 19-2.)

Address: $0012

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

**Figure 19-14. SPI Data Register (SPDR)**

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

***NOTE***

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

# Chapter 20
# Timer Interface Module B (TIMB)

## 20.1 Introduction

This chapter describes the timer interface module (TIMB). The TIMB is a 2-channel timer that provides a timing reference with input capture, output compare and pulse width modulation functions. Figure 20-1 is a block diagram of the TIMB.

The TIMB module is feature of the MC68HC908AZ60A only.

For further information regarding timers on M68HC08 family devices, please consult the HC08 Timer Reference Manual, TIM08RM/AD.

## 20.2 Features

Features of the TIMB include:

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

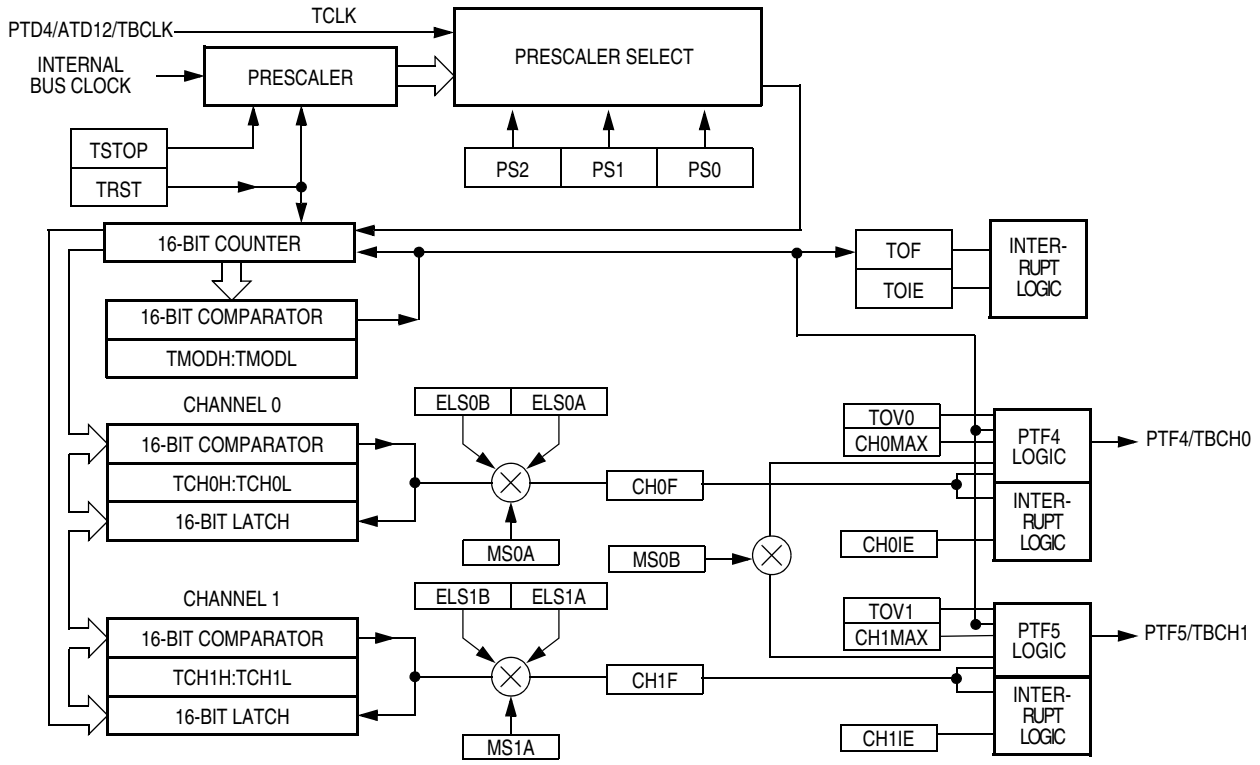



**Figure 20-1. TIMB Block Diagram**

| Addr. | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $0040 | TIMB Status/Control Register (TBSC) | TOF | TOIE | TSTOP | TRST | 0 | PS2 | PS1 | PS0 |
| $0041 | TIMB Counter Register High (TBCNTH) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0042 | TIMB Counter Register Low (TBCNTL) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0043 | TIMB Counter Modulo Reg. High (TBMODH) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0044 | TIMB Counter Modulo Reg. Low (TBMODL) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0045 | TIMB Ch. 0 Status/Control Register (TBSC0) | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| $0046 | TIMB Ch. 0 Register High (TBCH0H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0047 | TIMB Ch. 0 Register Low (TBCH0L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0048 | TIMB Ch. 1 Status/Control Register (TBSC1) | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| $0049 | TIMB Ch. 1 Register High (TBCH1H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $004A | TIMB Ch. 1 Register Low (TBCH1L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

R = Reserved

**Figure 20-2. TIMB I/O Register Summary**

## 20.3 Functional Description

Figure 20-1 shows the TIMB structure. The central component of the TIMB is the 16-bit TIMB counter that can operate as a free-running counter or a modulo up-counter. The TIMB counter provides the timing reference for the input capture and output compare functions. The TIMB counter modulo registers, TBMODH–TBMODL, control the modulo value of the TIMB counter. Software can read the TIMB counter value at any time without affecting the counting sequence.

The two TIMB channels are programmable independently as input capture or output compare channels.

### 20.3.1 TIMB Counter Prescaler

The TIMB clock source can be one of the seven prescaler outputs or the TIMB clock pin, PTD4/ATD12/TBCLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMB status and control register select the TIMB clock source.

### 20.3.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TBSC0 through TBSC1 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMB latches the contents of the TIMB counter into the TIMB channel registers, TBCHxH–TBCHxL. Input captures can generate TIMB CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIMB channel register (TBCHxH–TBCHxL, see 20.8.5 TIMB Channel Registers) on each proper signal transition regardless of whether the TIMB channel flag (CH0F–CH1F in TBSC0–TBSC1 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see 20.8.5 TIMB Channel Registers). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the input capture channel register (TBCHxH–TBCHxL).

## 20.3.3 Output Compare

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

### *20.3.3.1 Unbuffered Output Compare*

Any output compare channel can generate unbuffered output compare pulses as described in 20.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMB channel registers.

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

Use 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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

### *20.3.3.2 Buffered Output Compare*

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTF4/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The output compare value in the TIMB channel 0 registers initially controls the output on the PTF4/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the output after the TIMB overflows. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the output are the ones written to last. TBSC0 controls and monitors the buffered output compare function and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF5/TBCH1, is available as a general-purpose I/O pin.

***NOTE***

*In buffered output compare operation, do not write new output compare values to the currently active channel registers. 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.*

### 20.3.4 Pulse Width Modulation (PWM)

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

As Figure 20-3 shows, the output compare value in the TIMB channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMB to clear the channel pin on output compare if the polarity of the PWM pulse is 1. Program the TIMB to set the pin if the polarity of the PWM pulse is 0.

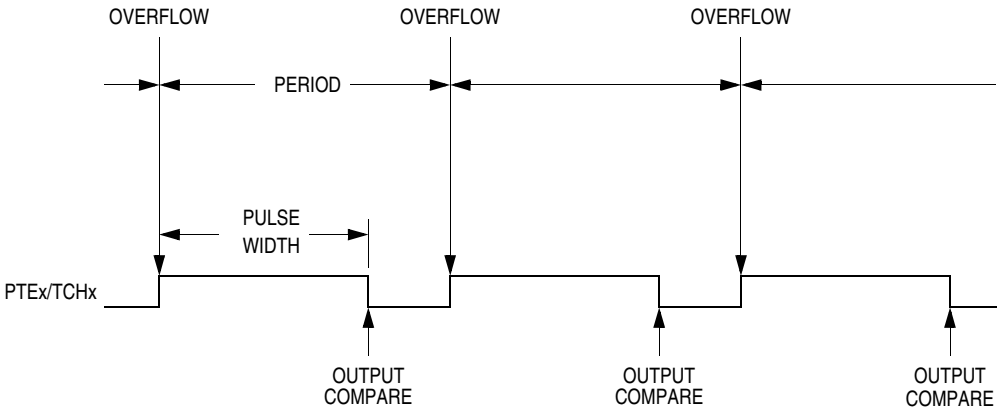



**Figure 20-3. PWM Period and Pulse Width**

The value in the TIMB counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMB counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see 20.8.1 TIMB Status and Control Register).

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

#### *20.3.4.1 Unbuffered PWM Signal Generation*

Any output compare channel can generate unbuffered PWM pulses as described in 20.3.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMB channel registers.

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

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 TIMB overflow interrupts and write the new value in the TIMB overflow interrupt routine. The TIMB overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

**NOTE**

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

#### *20.3.4.2 Buffered PWM Signal Generation*

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTF4/TBCH0 pin. The TIMB channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMB channel 0 status and control register (TBSC0) links channel 0 and channel 1. The TIMB channel 0 registers initially control the pulse width on the PTF4/TBCH0 pin. Writing to the TIMB channel 1 registers enables the TIMB channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMB channel registers (0 or 1) that control the pulse width are the ones written to last. TBSC0 controls and monitors the buffered PWM function, and TIMB channel 1 status and control register (TBSC1) is unused. While the MS0B bit is set, the channel 1 pin, PTF5/TBCH1, is available as a general-purpose I/O pin.

**NOTE**

*In buffered PWM signal generation, do not write new pulse width values to the currently active channel registers. 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.*

#### *20.3.4.3 PWM Initialization*

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

1. In the TIMB status and control register (TBSC):
   a. Stop the TIMB counter by setting the TIMB stop bit, TSTOP.
   b. Reset the TIMB counter and prescaler by setting the TIMB reset bit, TRST.
2. In the TIMB counter modulo registers (TBMODH–TBMODL) write the value for the required PWM period.
3. In the TIMB channel x registers (TBCHxH–TBCHxL) write the value for the required pulse width.
4. In TIMB channel x status and control register (TBSCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA (see Table 20-2).
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level (see Table 20-2).

   ***NOTE***

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

5. In the TIMB status control register (TBSC) clear the TIMB stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMB channel 0 registers (TBCH0H–TBCH0L) initially control the buffered PWM output. TIMB status control register 0 (TBSC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

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

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output (see 20.8.4 TIMB Channel Status and Control Registers).

## 20.4 Interrupts

The following TIMB sources can generate interrupt requests:

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

## 20.5 Low-Power Modes

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

### 20.5.1 Wait Mode

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

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

### 20.5.2 Stop Mode

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

## 20.6 TIMB During Break Interrupts

A break interrupt stops the TIMB 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state (see 9.7.3 SIM Break Flag Control Register).

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

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

## 20.7 I/O Signals

Port D shares one of its pins with the TIMB. Port F shares two of its pins with the TIMB. PTD4/ATD12/TBCLK is an external clock input to the TIMB prescaler. The two TIMB channel I/O pins are PTF4/TBCH0 and PTF5/TBCH1.

### 20.7.1 TIMB Clock Pin (PTD4/ATD12/TBCLK)

PTD4/ATD12/TBCLK is an external clock input that can be the clock source for the TIMB counter instead of the prescaled internal bus clock. Select the PTD4/ATD12/TBCLK input by writing logic 1s to the three prescaler select bits, PS[2:0] (see 20.8.1 TIMB Status and Control Register). The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

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

The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

PTD4/ATD12/TBCLK is available as a general-purpose I/O pin or ADC channel when not used as the TIMB clock input. When the PTD4/ATD12/TBCLK pin is the TIMB clock input, it is an input regardless of the state of the DDRD4 bit in data direction register D.

### 20.7.2 TIMB Channel I/O Pins (PTF5/TBCH1–PTF4/TBCH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTF4/TBCH0 and PTF5/TBCH1 can be configured as buffered output compare or buffered PWM pins.

## 20.8 I/O Registers

These I/O registers control and monitor TIMB operation:

- TIMB status and control register (TBSC)
- TIMB control registers (TBCNTH–TBCNTL)
- TIMB counter modulo registers (TBMODH–TBMODL)
- TIMB channel status and control registers (TBSC0 and TBSC1)
- TIMB channel registers (TBCH0H–TBCH0L, TBCH1H–TBCH1L)

### 20.8.1 TIMB Status and Control Register

The TIMB status and control register:

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

Address: $0040

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

R = Reserved

**Figure 20-4. TIMB Status and Control Register (TBSC)**

**TOF — TIMB Overflow Flag Bit**

This read/write flag is set when the TIMB counter reaches the modulo value programmed in the TIMB counter modulo registers. Clear TOF by reading the TIMB status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMB overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIMB counter has reached modulo value
0 = TIMB counter has not reached modulo value

**TOIE — TIMB Overflow Interrupt Enable Bit**

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

1 = TIMB overflow interrupts enabled
0 = TIMB overflow interrupts disabled

**TSTOP — TIMB Stop Bit**

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

1 = TIMB counter stopped
0 = TIMB counter active

***NOTE***

*Do not set the TSTOP bit before entering wait mode if the TIMB is required to exit wait mode. Also, when the TSTOP bit is set and the timer is configured for input capture operation, input captures are inhibited until TSTOP is cleared.*

*When using TSTOP to stop the timer counter, see if any timer flags are set. If a timer flag is set, it must be cleared by clearing TSTOP, then clearing the flag, then setting TSTOP again.*

**TRST — TIMB Reset Bit**

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

1 = Prescaler and TIMB counter cleared
0 = No effect

***NOTE***

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

**PS[2:0] — Prescaler Select Bits**

These read/write bits select either the PTD4/ATD12/TBCLK pin or one of the seven prescaler outputs as the input to the TIMB counter as Table 20-1 shows. Reset clears the PS[2:0] bits.

**Table 20-1. Prescaler Selection**

| PS[2:0] | TIMB Clock Source |
|---|---|
| 000 | Internal Bus Clock ÷1 |
| 001 | Internal Bus Clock ÷ 2 |
| 010 | Internal Bus Clock ÷ 4 |
| 011 | Internal Bus Clock ÷ 8 |
| 100 | Internal Bus Clock ÷ 16 |
| 101 | Internal Bus Clock ÷ 32 |
| 110 | Internal Bus Clock ÷ 64 |
| 111 | PTD4/ATD12/TBCLK |

## 20.8.2 TIMB Counter Registers

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

***NOTE***

*If TBCNTH is read during a break interrupt, be sure to unlatch TBCNTL by reading TBCNTL before exiting the break interrupt. Otherwise, TBCNTL retains the value latched during the break.*

Register Name and Address: TBCNTH — $0041

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

Register Name and Address: TBCNTL — $0042

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

= Unimplemented

**Figure 20-5. TIMB Counter Registers (TBCNTH and TBCNTL)**

### 20.8.3 TIMB Counter Modulo Registers

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

Register Name and Address: TBMODH — $0043

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Register Name and Address: TBMODL — $0044

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
| Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 20-6. TIMB Counter Modulo Registers (TBMODH and TBMODL)**

***NOTE***

*Reset the TIMB counter before writing to the TIMB counter modulo registers.*

### 20.8.4 TIMB Channel Status and Control Registers

Each of the TIMB channel status and control registers:

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

Register Name and Address TBSC0 — $0045

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

Register Name and Address TBSC1 — $0048

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

R = Reserved

**Figure 20-7. TIMB Channel Status and Control Registers (TBSC0–TBSC1)**

### CHxF — Channel x Flag Bit

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

When CHxIE = 1, clear CHxF by reading TIMB channel x status and control register with CHxF set, and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

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

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

### CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMB CPU interrupts on channel x.
Reset clears the CHxIE bit.

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

### MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMB channel 0.

Setting MS0B disables the channel 1 status and control register and reverts TBCH1 to general-purpose I/O.

Reset clears the MSxB bit.

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

### MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation (see Table 20-2).

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TBCHx pin once PWM, input capture or output compare operation is enabled (see Table 20-2). Reset clears the MSxA bit.

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

***NOTE***

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

### ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port F and pin PTFx/TBCHx is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture, or output compare mode is enabled. Table 20-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**Table 20-2. Mode, Edge, and Level Selection**

| MSxB | MSxA | ELSxB | ELSxA | Mode | Configuration |
|---|---|---|---|---|---|
| X | 0 | 0 | 0 | Output preset | Pin under port control; initial output level high |
| X | 1 | 0 | 0 | | Pin under port control; initial output level low |
| 0 | 0 | 0 | 1 | Input capture | Capture on rising edge only |
| 0 | 0 | 1 | 0 | | Capture on falling edge only |
| 0 | 0 | 1 | 1 | | Capture on rising or falling edge |
| 0 | 1 | 0 | 0 | Output compare or PWM | Software compare only |
| 0 | 1 | 0 | 1 | | Toggle output on compare |
| 0 | 1 | 1 | 0 | | Clear output on compare |
| 0 | 1 | 1 | 1 | | Set output on compare |
| 1 | X | 0 | 1 | Buffered output compare or buffered PWM | Toggle output on compare |
| 1 | X | 1 | 0 | | Clear output on compare |
| 1 | X | 1 | 1 | | Set output on compare |

***NOTE***

*Before enabling a TIMB channel register for input capture operation, make sure that the PTFx/TBCHx pin is stable for at least two bus clocks.*

**TOVx — Toggle-On-Overflow Bit**

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

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

***NOTE***

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

**CHxMAX — Channel x Maximum Duty Cycle Bit**

When the TOVx bit is at logic 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 20-8 shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.

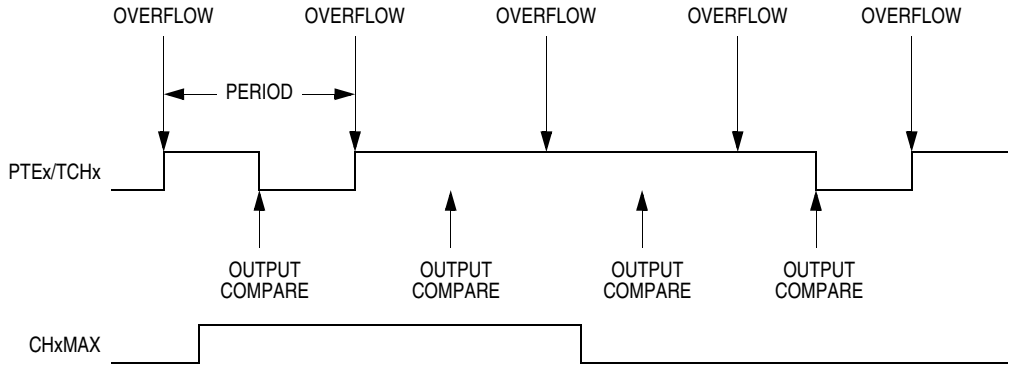



**Figure 20-8. CHxMAX Latency**

## 20.8.5 TIMB Channel Registers

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

In input capture mode (MSxB–MSxA = 0:0) reading the high byte of the TIMB channel x registers (TBCHxH) inhibits input captures until the low byte (TBCHxL) is read.

In output compare mode (MSxB–MSxA ≠ 0:0) writing to the high byte of the TIMB channel x registers (TBCHxH) inhibits output compares and the CHxF bit until the low byte (TBCHxL) is written.

Register Name and Address TBCH0H — $0046

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

Register Name and Address TBCH0L — $0047

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset: | Indeterminate after Reset | | | | | | | |

Register Name and Address TBCH1H — $0049

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

Register Name and Address TBCH1L — $004A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| Reset: | Indeterminate after Reset | | | | | | | |

**Figure 20-9. TIMB Channel Registers (TBCH0H/L–TBCH1H/L)**

# Chapter 21
# Programmable Interrupt Timer (PIT)

## 21.1 Introduction

This chapter describes the Programmable Interrupt Timer (PIT) which is a periodic interrupt timer whose counter is clocked internally via software programmable options. Figure 21-1 is a block diagram of the PIT.

For further information regarding timers on M68HC08 family devices, please consult the HC08 Timer Reference Manual, TIM08RM/AD.

## 21.2 Features

Features of the PIT include:

- Programmable PIT Clock Input
- Free-Running or Modulo Up-Count Operation
- PIT Counter Stop and Reset Bits

## 21.3 Functional Description

Figure 21-1 shows the structure of the PIT. The central component of the PIT is the 16-bit PIT counter that can operate as a free-running counter or a modulo up-counter. The counter provides the timing reference for the interrupt. The PIT counter modulo registers, PMODH–PMODL, control the modulo value of the counter. Software can read the counter value at any time without affecting the counting sequence.

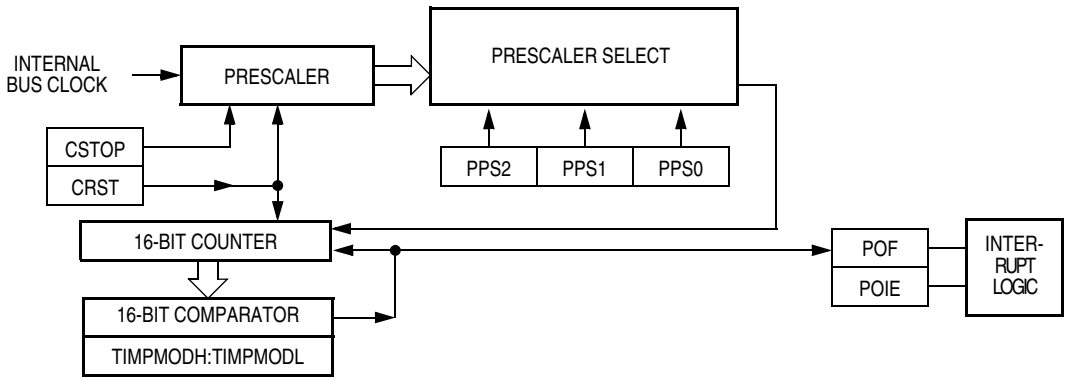



**Figure 21-1. PIT Block Diagram**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| PIT Status and Control Register (PSC) | Read: | POF | POIE | PSTOP | 0 | 0 | PPS2 | PPS1 | PPS0 |
| | Write: | 0 | | | PRST | | | | |
| | Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| PIT Counter Register High (PCNTH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | Write: | | | | | | | | |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PIT Counter Register Low (PCNTL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | Write: | | | | | | | | |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| PIT Counter Modulo Register High (PMODH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | Write: | | | | | | | | |
| | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| PIT Counter Modulo Register Low (PMODL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | Write: | | | | | | | | |
| | Reset: | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

=Unimplemented

**Figure 21-2. PIT I/O Register Summary**

**Table 21-1. PIT I/O Register Address Summary**

| Register | PSC | PCNTH | PCNTL | PMODH | PMODL |
|---|---|---|---|---|---|
| Address | $004B | $004C | $004D | $004E | $004F |

## 21.4 PIT Counter Prescaler

The 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, PPS[2:0], in the status and control register select the PIT clock source.

The value in the PIT counter modulo registers and the selected prescaler output determines the frequency of the periodic interrupt. The PIT overflow flag (POF) is set when the PIT counter value reaches the modulo value programmed in the PIT counter modulo registers. The PIT interrupt enable bit, POIE, enables PIT overflow CPU interrupt requests. POF and POIE are in the PIT status and control register.

## 21.5 Low-Power Modes

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

### 21.5.1 Wait Mode

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

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

### 21.5.2 Stop Mode

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

## 21.6 PIT During Break Interrupts

A break interrupt stops the PIT counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state (see 9.7.3 SIM Break Flag Control Register).

To allow software to clear status bits during a break interrupt, write a 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 0 to the BCFE bit. With BCFE at 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 0. After the break, doing the second step clears the status bit.

## 21.7 I/O Registers

The following I/O registers control and monitor operation of the PIT:

- PIT status and control register (PSC)
- PIT counter registers (PCNTH–PCNTL)
- PIT counter modulo registers (PMODH–PMODL)

### 21.7.1 PIT Status and Control Register

The PIT status and control register:

- Enables PIT interrupt
- Flags PIT overflows
- Stops the PIT counter
- Resets the PIT counter
- Prescales the PIT counter clock

Address: $004B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | POF | POIE | PSTOP | 0 | 0 | PPS2 | PPS1 | PPS0 |
| Write: | 0 | | | PRST | | | | |
| Reset: | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 21-3. PIT Status and Control Register (PSC)**

**POF — PIT Overflow Flag Bit**

This read/write flag is set when the PIT counter reaches the modulo value programmed in the PIT counter modulo registers. Clear POF by reading the PIT status and control register when POF is set and then writing a 0 to POF. If another PIT overflow occurs before the clearing sequence is complete, then writing 0 to POF has no effect. Therefore, a POF interrupt request cannot be lost due to inadvertent clearing of POF. Reset clears the POF bit. Writing a 1 to POF has no effect.

1 = PIT counter has reached modulo value
0 = PIT counter has not reached modulo value

**POIE — PIT Overflow Interrupt Enable Bit**

This read/write bit enables PIT overflow interrupts when the POF bit becomes set. Reset clears the POIE bit.

1 = PIT overflow interrupts enabled
0 = PIT overflow interrupts disabled

**PSTOP — PIT Stop Bit**

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

1 = PIT counter stopped
0 = PIT counter active

***NOTE***

*Do not set the PSTOP bit before entering wait mode if the PIT is required to exit wait mode.*

**PRST — PIT Reset Bit**

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

1 = Prescaler and PIT counter cleared
0 = No effect

***NOTE***

*Setting the PSTOP and PRST bits simultaneously stops the PIT counter at a value of $0000.*

**PPS[2:0] — Prescaler Select Bits**

These read/write bits select one of the seven prescaler outputs as the input to the PIT counter as Table 21-2 shows. Reset clears the PPS[2:0] bits.

**Table 21-2. Prescaler Selection**

| PPS[2:0] | PIT Clock Source |
|---|---|
| 000 | Internal Bus Clock ÷1 |
| 001 | Internal Bus Clock ÷ 2 |
| 010 | Internal Bus Clock ÷ 4 |
| 011 | Internal Bus Clock ÷ 8 |
| 100 | Internal Bus Clock ÷ 16 |
| 101 | Internal Bus Clock ÷ 32 |
| 110 | Internal Bus Clock ÷ 64 |
| 111 | Internal Bus Clock ÷ 64 |

## 21.7.2 PIT Counter Registers

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

***NOTE***

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

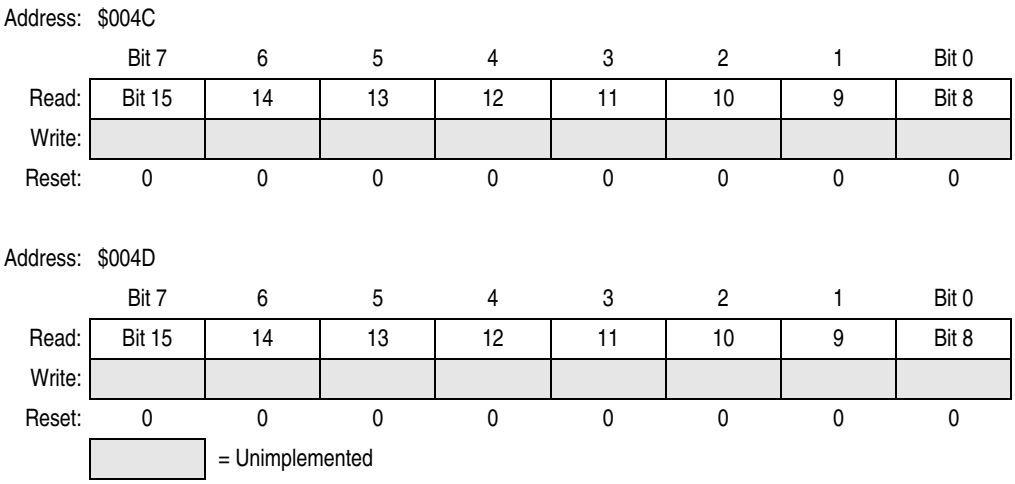



**Figure 21-4. PIT Counter Registers (PCNTH–PCNTL)**

### 21.7.3 PIT Counter Modulo Registers

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

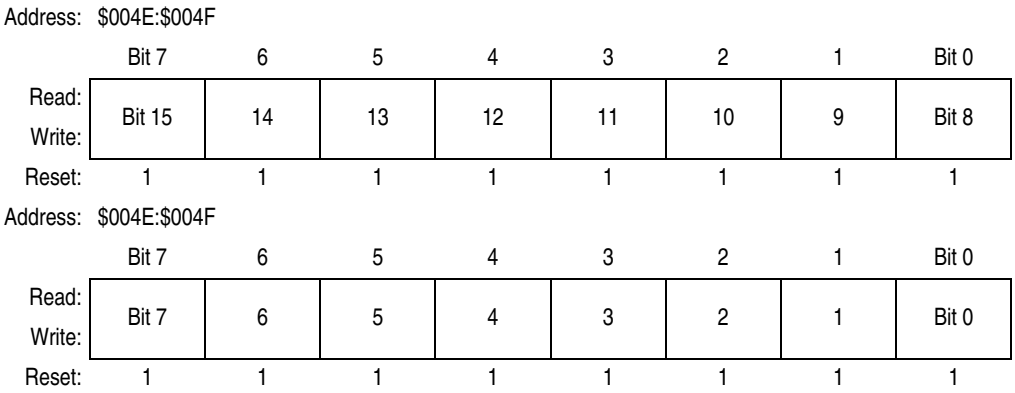



**Figure 21-5. PIT Counter Modulo Registers (PMODH–PMODL)**

***NOTE***

*Reset the PIT counter before writing to the PIT counter modulo registers.*

# Chapter 22
# Input/Output Ports

## 22.1 Introduction

On the MC68HC908AZ60A and 64-pin MC68HC908AS60A, fifty bidirectional input/output (I/O) form seven parallel ports. On the52-pin MC68HC908AS60A, forty bidirectional input/output (I/O) form six parallel ports. All I/O pins are programmable as inputs or outputs.

***NOTE***

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

| Addr. | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $0000 | Port A Data Register (PTA) | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| $0001 | Port B Data Register (PTB) | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| $0002 | Port C Data Register (PTC) | 0 | 0 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| $0003 | Port D Data Register (PTD) | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| $0004 | Data Direction Register A (DDRA) | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| $0005 | Data Direction Register B (DDRB) | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| $0006 | Data Direction Register C (DDRC) | MCLKEN | 0 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| $0007 | Data Direction Register D (DDRD) | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| $0008 | Port E Data Register (PTE) | PTE7 | PTE6 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 |
| $0009 | Port F Data Register (PTF) | 0 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| $000A | Port G Data Register (PTG) | 0 | 0 | 0 | 0 | 0 | PTG2 | PTG1 | PTG0 |
| $000B | Port H Data Register (PTH) | 0 | 0 | 0 | 0 | 0 | 0 | PTH1 | PTH0 |
| $000C | Data Direction Register E (DDRE) | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| $000D | Data Direction Register F (DDRF) | 0 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| $000E | Data Direction Register G (DDRG) | 0 | 0 | 0 | 0 | 0 | DDRG2 | DDRG1 | DDRG0 |
| $000F | Data Direction Register H (DDRH) | 0 | 0 | 0 | 0 | 0 | 0 | DDRH1 | DDRH0 |

**Figure 22-1. I/O Port Register Summary**

## 22.2 Port A

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

### 22.2.1 Port A Data Register

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

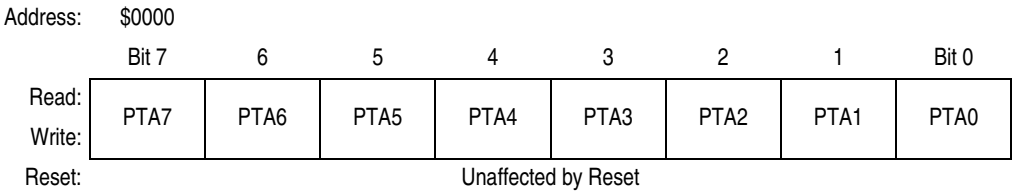



**Figure 22-2. Port A Data Register (PTA)**

**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.

### 22.2.2 Data Direction Register A

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

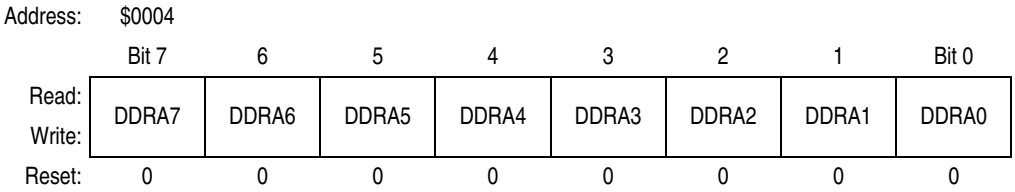



**Figure 22-3. Data Direction Register A (DDRA)**

**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 22-4 shows the port A I/O logic.

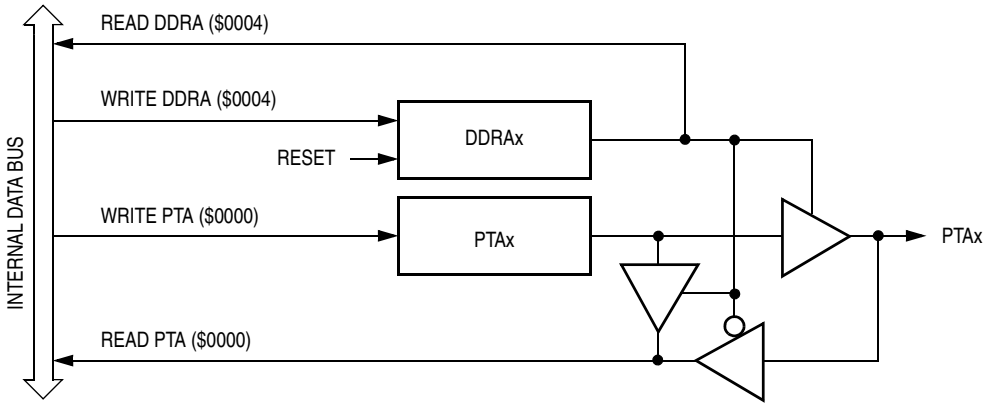



**Figure 22-4. Port A I/O Circuit**

When bit DDRAx is a logic 1, reading address $0000 reads the PTAx data latch. When bit DDRAx is a logic 0, reading address $0000 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-1 summarizes the operation of the port A pins.

**Table 22-1. Port A Pin Functions**

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

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

## 22.3 Port B

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

### 22.3.1 Port B Data Register

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

| Address: | $0001 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read:<br>Write: | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| Reset: | Unaffected by Reset | | | | | | | |
| Alternative Functions: | ATD7 | ATD6 | ATD5 | ATD4 | ATD3 | ATD2 | ATD1 | ATD0 |

**Figure 22-5. Port B Data Register (PTB)**

**PTB[7:0] — Port B Data Bits**

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

**ATD[7:0] — ADC Channels**

PTB7/ATD7–PTB0/ATD0 are eight of the analog-to-digital converter channels. The ADC channel select bits, CH[4:0], determine whether the PTB7/ATD7–PTB0/ATD0 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See Chapter 26 Analog-to-Digital Converter (ADC)). Data direction register B (DDRB) does not affect the data direction of port B pins that are being used by the ADC. However, the DDRB bits always determine whether reading port B returns to the states of the latches or 0.

### 22.3.2 Data Direction Register B

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

Address: $0005

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-6. Data Direction Register B (DDRB)**

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

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

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

***NOTE***

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

Figure 22-7 shows the port B I/O logic.

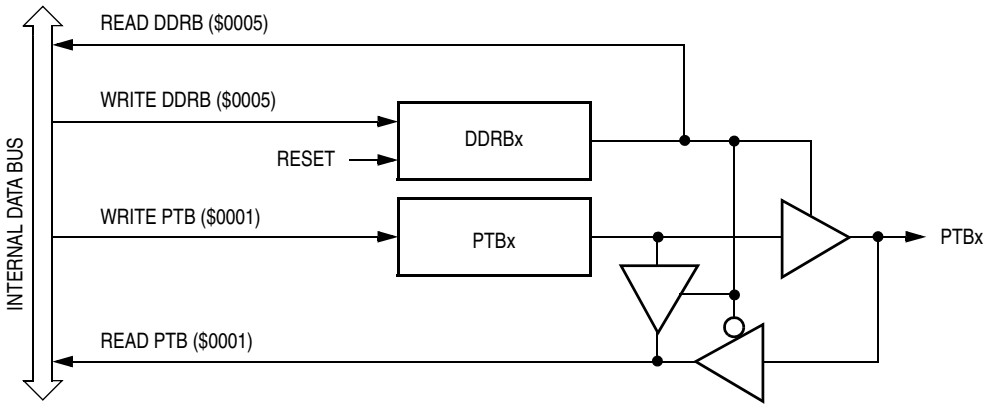



**Figure 22-7. Port B I/O Circuit**

When bit DDRBx is a logic 1, reading address $0001 reads the PTBx data latch. When bit DDRBx is a logic 0, reading address $0001 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-2 summarizes the operation of the port B pins.

**Table 22-2. Port B Pin Functions**

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

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

## 22.4 Port C

Port C is an 6-bit general-purpose bidirectional I/O port. Note that PTC5 is only available on 64-pin package options.

### 22.4.1 Port C Data Register

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

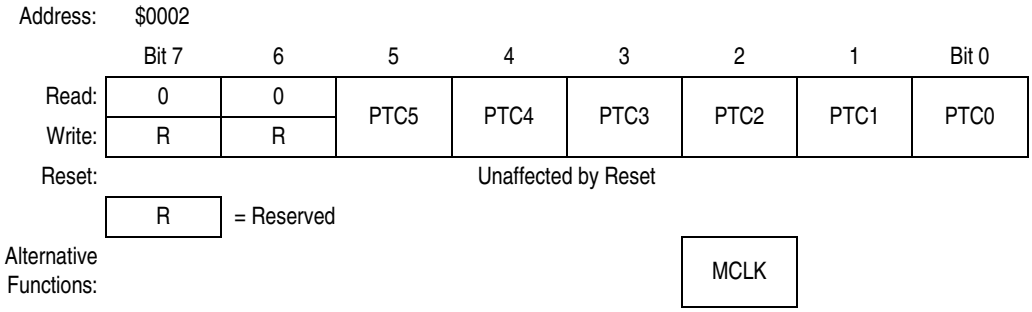



**Figure 22-8. Port C Data Register (PTC)**

**PTC[5:0] — 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 (5:0).

**MCLK — System Clock Bit**

The system clock is driven out of PTC2 when enabled by MCLKEN bit in PTCDDR7.

### 22.4.2 Data Direction Register C

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

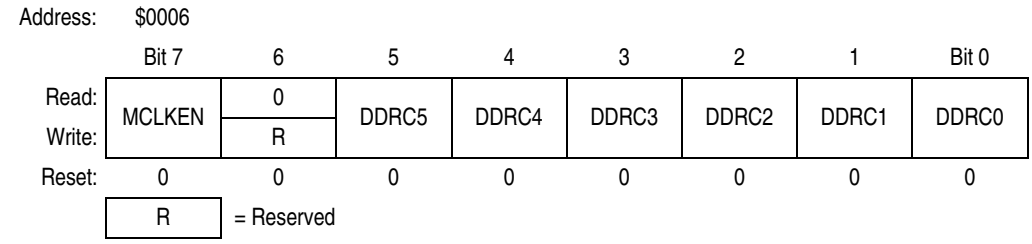



**Figure 22-9. Data Direction Register C (DDRC)**

**MCLKEN — MCLK Enable Bit**

This read/write bit enables MCLK to be an output signal on PTC2. If MCLK is enabled, DDRC2 has no effect. Reset clears this bit.

1 = MCLK output enabled
0 = MCLK output disabled

### DDRC[5:0] — 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 22-10 shows the port C I/O logic.

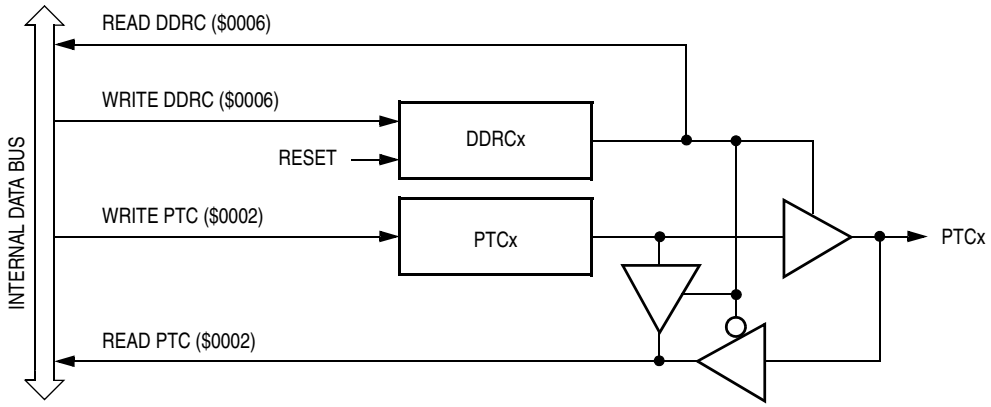



**Figure 22-10. Port C I/O Circuit**

When bit DDRCx is a logic 1, reading address $0002 reads the PTCx data latch. When bit DDRCx is a logic 0, reading address $0002 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-3 summarizes the operation of the port C pins.

**Table 22-3. Port C Pin Functions**

| Bit Value | PTC Bit | I/O Pin Mode | Accesses to DDRC | Accesses to PTC | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | 2 | Input, Hi-Z | DDRC[2] | Pin | PTC2 |
| 1 | 2 | Output | DDRC[2] | 0 | — |
| 0 | X | Input, Hi-Z | DDRC[5:0] | Pin | PTC[5:0](1) |
| 1 | X | Output | DDRC[5:0] | PTC[5:0] | PTC[5:0] |

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

## 22.5 Port D

Port D is an 8-bit general-purpose I/O port. Note that PTD7 is only available on 64-pin package options.

### 22.5.1 Port D Data Register

Port D is a 8-bit special function port that shares seven of its pins with the analog to digital converter and two with the timer interface modules.

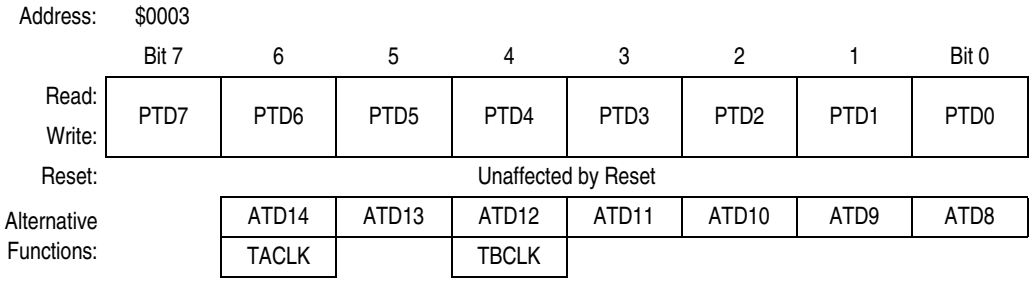



**Figure 22-11. Port D Data Register (PTD)**

**PTD[7:0] — Port D Data Bits**

PTD[7:0] are read/write, software programmable bits. Data direction of PTD[7:0] pins are under the control of the corresponding bit in data direction register D.

**ATD[14:8] — ADC Channel Status Bits**

PTD6/ATD14/TACLK–PTD0/ATD8 are seven of the 15 analog-to-digital converter channels. The ADC channel select bits, CH[4:0], determine whether the PTD6/ATD14/TACLK–PTD0/ATD8 pins are ADC channels or general-purpose I/O pins. If an ADC channel is selected and a read of this corresponding bit in the port B data register occurs, the data will be 0 if the data direction for this bit is programmed as an input. Otherwise, the data will reflect the value in the data latch. (See Chapter 26 Analog-to-Digital Converter (ADC)).

***NOTE***

*Data direction register D (DDRD) does not affect the data direction of port D pins that are being used by the TIMA or TIMB. However, the DDRD bits always determine whether reading port D returns the states of the latches or a 0.*

**TACLK/TBCLK — Timer Clock Input Bit**

The PTD6/ATD14/TACLK pin is the external clock input for the TIMA. The PTD4/ATD12/TBCLK pin is the external clock input for the TIMB. The prescaler select bits, PS[2:0], select PTD6/ATD14/TACLK or PTD4/ATD12/TBCLK as the TIM clock input. (See 25.8.4 TIMA Channel Status and Control Registers and 20.8.4 TIMB Channel Status and Control Registers). When not selected as the TIM clock, PTD6/ATD14/TACLK and PTD4/ATD12/TBCLK are available for general-purpose I/O. While TACLK/TBCLK are selected corresponding DDRD bits have no effect.

### 22.5.2 Data Direction Register D

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

Address: $0007

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDRD2 | DDRD1 | DDRD0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-12. Data Direction Register D (DDRD)**

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

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

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

***NOTE***

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

Figure 22-13 shows the port D I/O logic.

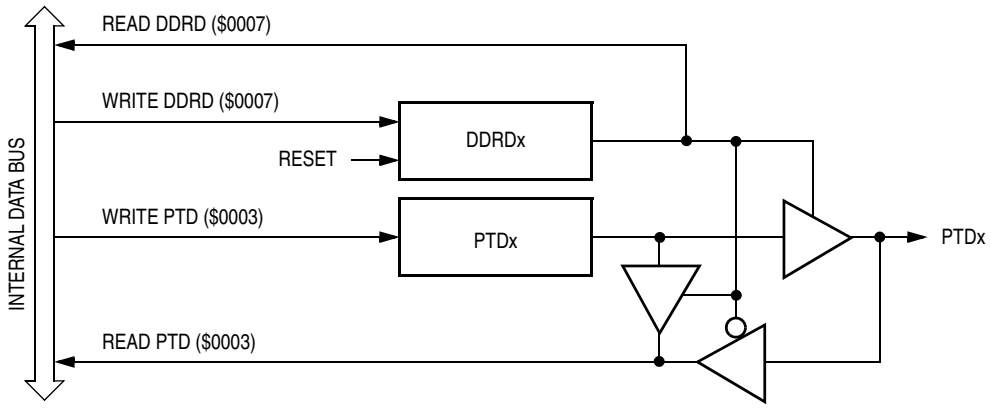



**Figure 22-13. Port D I/O Circuit**

When bit DDRDx is a logic 1, reading address $0003 reads the PTDx data latch. When bit DDRDx is a logic 0, reading address $0003 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-4 summarizes the operation of the port D pins.

**Table 22-4. Port D Pin Functions**

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

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

## 22.6 Port E

Port E is an 8-bit special function port that shares two of its pins with the timer interface module (TIMA), two of its pins with the serial communications interface module (SCI), and four of its pins with the serial peripheral interface module (SPI).

### 22.6.1 Port E Data Register

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

Address: $0008

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | PTE7 | PTE6 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 |
| Reset: | Unaffected by Reset | | | | | | | |
| Alternative Function: | SPSCK | MOSI | MISO | $\overline{SS}$ | TACH1 | TACH0 | RxD | TxD |

**Figure 22-14. Port E Data Register (PTE)**

**PTE[7:0] — Port E Data Bits**

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

**SPSCK — SPI Serial Clock Bit**

The PTE7/SPSCK pin is the serial clock input of an SPI slave module and serial clock output of an SPI master module. When the SPE bit is clear, the PTE7/SPSCK pin is available for general-purpose I/O. (See 19.13.1 SPI Control Register).

**MOSI — Master Out/Slave In Bit**

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

**MISO — Master In/Slave Out Bit**

The PTE5/MISO pin is the master in/slave out terminal of the SPI module. When the SPI enable bit, SPE, is clear, the SPI module is disabled, and the PTE5/MISO pin is available for general-purpose I/O. (See 19.13.1 SPI Control Register).

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

The PTE4/$\overline{SS}$ pin is the slave select input of the SPI module. When the SPE bit is clear, or when the SPI master bit, SPMSTR, is set and MODFEN bit is low, the PTE4/$\overline{SS}$ pin is available for general-purpose I/O. (See 19.12.4 SS (Slave Select)). When the SPI is enabled as a slave, the DDRF0 bit in data direction register E (DDRE) has no effect on the PTE4/$\overline{SS}$ pin.

***NOTE***

*Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SPI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See Table 22-5).*

**TACH[1:0] — Timer Channel I/O Bits**

The PTE3/TACH1–PTE2/TACH0 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTE3/TACH1–PTE2/TACH0 pins are timer channel I/O pins or general-purpose I/O pins. (See 25.8.4 TIMA Channel Status and Control Registers).

***NOTE***

*Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the TIM. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See Table 22-5).*

**RxD — SCI Receive Data Input Bit**

The PTE1/RxD pin is the receive data input for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE1/RxD pin is available for general-purpose I/O. (See 18.8.1 SCI Control Register 1).

**TxD — SCI Transmit Data Output**

The PTE0/TxD pin is the transmit data output for the SCI module. When the enable SCI bit, ENSCI, is clear, the SCI module is disabled, and the PTE0/TxD pin is available for general-purpose I/O. (See 18.8.1 SCI Control Register 1).

***NOTE***

*Data direction register E (DDRE) does not affect the data direction of port E pins that are being used by the SCI module. However, the DDRE bits always determine whether reading port E returns the states of the latches or the states of the pins. (See Table 22-5).*

## 22.6.2 Data Direction Register E

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

Address: $000C

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

**Figure 22-15. Data Direction Register E (DDRE)**

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

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

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

***NOTE***

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

Figure 22-16 shows the port E I/O logic.

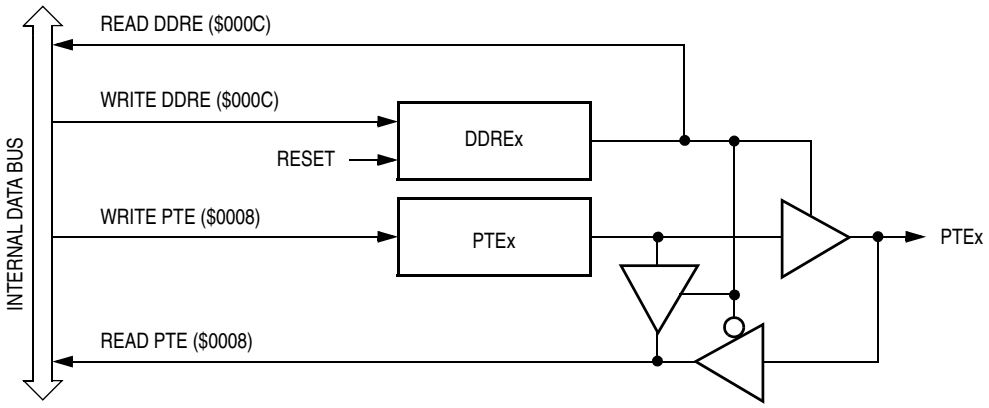



**Figure 22-16. Port E I/O Circuit**

When bit DDREx is a logic 1, reading address $0008 reads the PTEx data latch. When bit DDREx is a logic 0, reading address $0008 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-5 summarizes the operation of the port E pins.

**Table 22-5. Port E Pin Functions**

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

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

## 22.7 Port F

Port F is a 7-bit special function port that shares four of its pins with the timer interface module (TIMA-6) and two of its pins with the timer interface module (TIMB) on the MC68HC908AZ60A. Note that PTF4, PTF5 and PTF6 are only available on 64-pin package options.

### 22.7.1 Port F Data Register

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

| Address: | $0009 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | 0 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| Write: | R | | | | | | | |
| Reset: | Unaffected by Reset | | | | | | | |
| Alternative Function: | | | TBCH1 | TBCH0 | TACH5 | TACH4 | TACH3 | TACH2 |

R = Reserved

**Figure 22-17. Port F Data Register (PTF)**

**PTF[6:0] — Port F Data Bits**

These read/write bits are software programmable. Data direction of each port F pin is under the control of the corresponding bit in data direction register F. Reset has no effect on PTF[6:0].

**TACH[5:2] — Timer A Channel I/O Bits**

The PTF3–PTF0/TACH2 pins are the TIM input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF3–PTF0/TACH2 pins are timer channel I/O pins or general-purpose I/O pins. (See 25.8.1 TIMA Status and Control Register).

**TBCH[1:0] — Timer B Channel I/O Bits**

The PTF5/TBCH1–PTF4/TBCH0 pins are the TIMB input capture/output compare pins. The edge/level select bits, ELSxB:ELSxA, determine whether the PTF5/TBCH1–PTF4/TBCH0 pins are timer channel I/O pins or general-purpose I/O pins. (See 20.8.1 TIMB Status and Control Register).

***NOTE***

*Data direction register F (DDRF) does not affect the data direction of port F pins that are being used by the TIM. However, the DDRF bits always determine whether reading port F returns the states of the latches or the states of the pins. (See Table 22-6).*

### 22.7.2 Data Direction Register F

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

| Address: | $000D | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | 0 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| Write: | R | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 22-18. Data Direction Register F (DDRF)**

### DDRF[6:0] — Data Direction Register F Bits

These read/write bits control port F data direction. Reset clears DDRF[6:0], configuring all port F pins as inputs.

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

***NOTE***

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

Figure 22-19 shows the port F I/O logic.

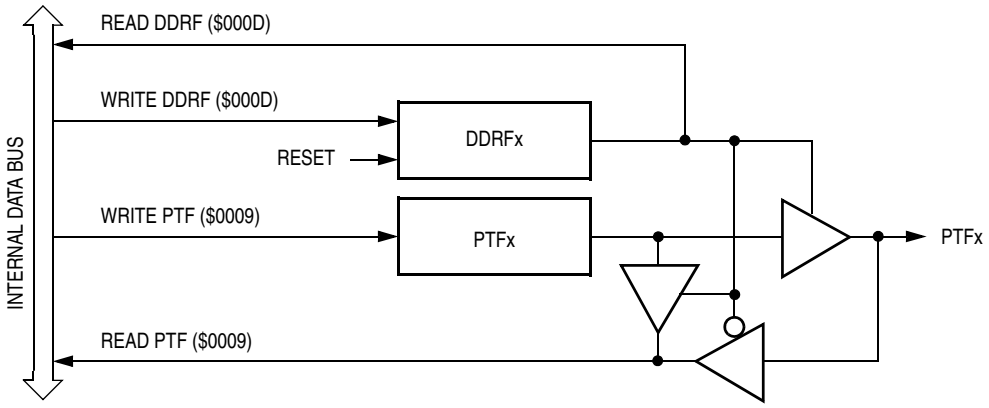



**Figure 22-19. Port F I/O Circuit**

When bit DDRFx is a logic 1, reading address $0009 reads the PTFx data latch. When bit DDRFx is a logic 0, reading address $0009 reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-6 summarizes the operation of the port F pins.

**Table 22-6. Port F Pin Functions**

| DDRF Bit | PTF Bit | I/O Pin Mode | Accesses to DDRF | Accesses to PTF | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, Hi-Z | DDRF[6:0] | Pin | PTF[6:0](1) |
| 1 | X | Output | DDRF[6:0] | PTF[6:0] | PTF[6:0] |

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

## 22.8 Port G

Port G is a 3-bit special function port that shares all of its pins with the keyboard interrupt module (KBD). Note that Port G is only available on 64-pin package options.

### 22.8.1 Port G Data Register

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

Address: $000A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | PTG2 | PTG1 | PTG0 |
| Write: | R | R | R | R | R | | | |
| Reset: | Unaffected by Reset | | | | | | | |
| Alternative Function: | | | | | | KBD2 | KBD1 | KBD0 |

R = Reserved

**Figure 22-20. Port G Data Register (PTG)**

**PTG[2:0] — Port G Data Bits**

These read/write bits are software programmable. Data direction of each port G pin is under the control of the corresponding bit in data direction register G. Reset has no effect on PTG[2:0].

**KBD[2:0] — Keyboard Wakeup pins**

The keyboard interrupt enable bits, KBIE[2:0], in the keyboard interrupt control register, enable the port G pins as external interrupt pins (See Chapter 24 Keyboard Module (KBI)). Enabling an external interrupt pin will override the corresponding DDRGx.

### 22.8.2 Data Direction Register G

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

Address: $000E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | DDRG2 | DDRG1 | DDRG0 |
| Write: | R | R | R | R | R | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 22-21. Data Direction Register G (DDRG)**

### DDRG[2:0] — Data Direction Register G Bits

These read/write bits control port G data direction. Reset clears DDRG[2:0], configuring all port G pins as inputs.

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

***NOTE***

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

Figure 22-22 shows the port G I/O logic.

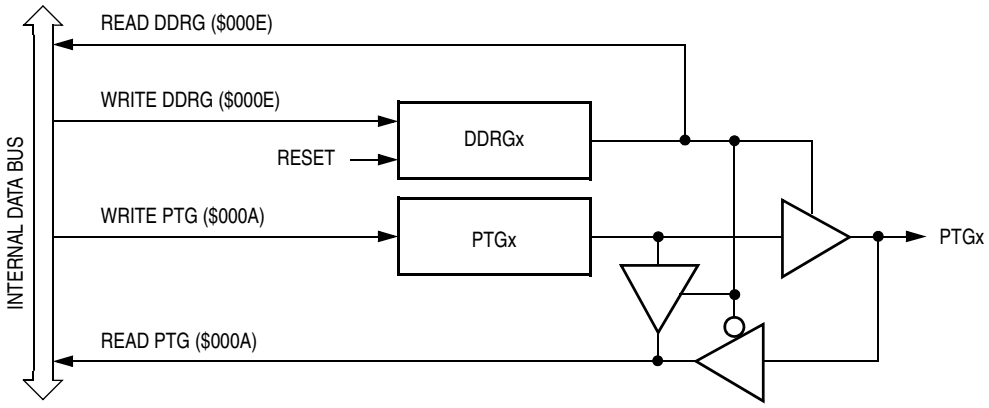



**Figure 22-22. Port G I/O Circuit**

When bit DDRGx is a logic 1, reading address $000A reads the PTGx data latch. When bit DDRGx is a logic 0, reading address $000A reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-7 summarizes the operation of the port G pins.

**Table 22-7. Port G Pin Functions**

| DDRG Bit | PTG Bit | I/O Pin Mode | Accesses to DDRG | Accesses to PTG | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, Hi-Z | DDRG[2:0] | Pin | PTG[2:0](1) |
| 1 | X | Output | DDRG[2:0] | PTG[2:0] | PTG[2:0] |

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

## 22.9 Port H

Port H is a 2-bit special function port that shares all of its pins with the keyboard interrupt module (KBD). Note that Port H is only available on 64-pin package options.

### 22.9.1 Port H Data Register

The port H data register contains a data latch for each of the two port H pins.

| Address: | $000B | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | PTH1 | PTH0 |
| Write: | R | R | R | R | R | R | | |
| Reset: | Unaffected by Reset | | | | | | | |
| Alternative Function: | | | | | | | KBD4 | KBD3 |

R = Reserved

**Figure 22-23. Port H Data Register (PTH)**

**PTH[1:0] — Port H Data Bits**

These read/write bits are software programmable. Data direction of each port H pin is under the control of the corresponding bit in data direction register H. Reset has no effect on PTH[1:0].

**KBD[4:3] — Keyboard Wake-up pins**

The keyboard interrupt enable bits, KBIE[4:3], in the keyboard interrupt control register, enable the port H pins as external interrupt pins (See Chapter 24 Keyboard Module (KBI)).

### 22.9.2 Data Direction Register H

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

| Address: | $000F | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Read: | 0 | 0 | 0 | 0 | 0 | 0 | DDRH1 | DDRH0 |
| Write: | R | R | R | R | R | R | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 22-24. Data Direction Register H (DDRH)**

**DDRH[1:0] — Data Direction Register H Bits**

These read/write bits control port H data direction. Reset clears DDRG[1:0], configuring all port H pins as inputs.

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

***NOTE***

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

Figure 22-25 shows the port H I/O logic.

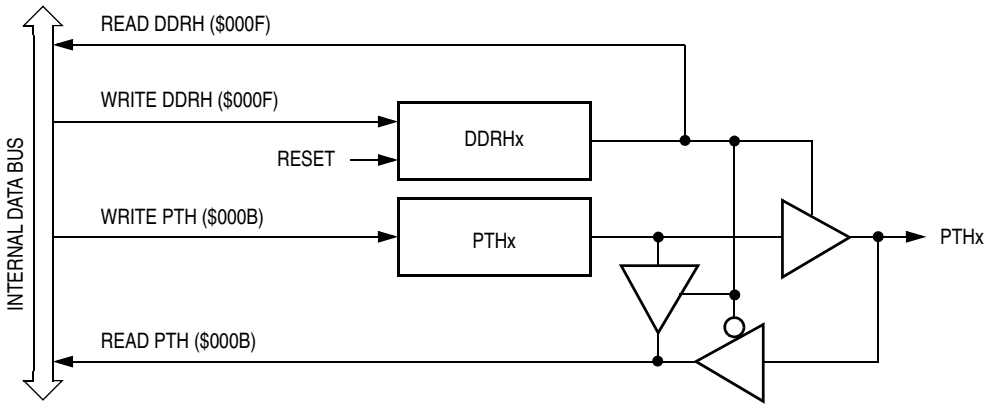



**Figure 22-25. Port H I/O Circuit**

When bit DDRHx is a logic 1, reading address $000B reads the PTHx data latch. When bit DDRHx is a logic 0, reading address $000B reads the voltage level on the pin. The data latch can always be written, regardless of the state of its data direction bit. Table 22-8 summarizes the operation of the port H pins.

**Table 22-8. Port H Pin Functions**

| DDRH Bit | PTH Bit | I/O Pin Mode | Accesses to DDRH | Accesses to PTH | |
|---|---|---|---|---|---|
| | | | Read/Write | Read | Write |
| 0 | X | Input, Hi-Z | DDRH[1:0] | Pin | PTH[1:0](1) |
| 1 | X | Output | DDRH[1:0] | PTH[1:0] | PTH[1:0] |

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

# Chapter 23
# MSCAN Controller (MSCAN08)

## 23.1 Introduction

The MSCAN08 is the specific implementation of the MSCAN concept targeted for the Freescale M68HC08 Microcontroller Family.

The module is a communication controller implementing the CAN 2.0 A/B protocol as defined in the BOSCH specification dated September 1991.

The CAN protocol was primarily, but not exclusively, designed to be used as a vehicle serial data bus, meeting the specific requirements of this field: real-time processing, reliable operation in the electromagnetic interference (EMI) environment of a vehicle, cost-effectiveness and required bandwidth.

MSCAN08 utilizes an advanced buffer arrangement, resulting in a predictable real-time behavior, and simplifies the application software.

The MSCAN08 is only available on the MC68HC908AZ60A.

## 23.2 Features

Basic features of the MSCAN08 are:

- Modular Architecture
- Implementation of the CAN Protocol — Version 2.0A/B
  - Standard and Extended Data Frames.
  - 0–8 Bytes Data Length.
  - Programmable Bit Rate up to 1 Mbps Depending on the Actual Bit Timing and the Clock Jitter of the PLL
- Support for Remote Frames
- Double-Buffered Receive Storage Scheme
- Triple-Buffered Transmit Storage Scheme with Internal Prioritisation Using a “Local Priority” Concept
- Flexible Maskable Identifier Filter Supports Alternatively One Full Size Extended Identifier Filter or Two 16-Bit Filters or Four 8-Bit Filters
- Programmable Wakeup Functionality with Integrated Low-Pass Filter
- Programmable Loop-Back Mode Supports Self-Test Operation
- Separate Signalling and Interrupt Capabilities for All CAN Receiver and Transmitter Error States (Warning, Error Passive, Bus Off)
- Programmable MSCAN08 Clock Source Either CPU Bus Clock or Crystal Oscillator Output
- Programmable Link to On-Chip Timer Interface Module (TIMB) for Time-Stamping and Network Synchronization
- Low-Power Sleep Mode

## 23.3 External Pins

The MSCAN08 uses two external pins, one input (RxCAN) and one output (TxCAN). The TxCAN output pin represents the logic level on the CAN: 0 is for a dominant state, and 1 is for a recessive state.

A typical CAN system with MSCAN08 is shown in Figure 23-1.

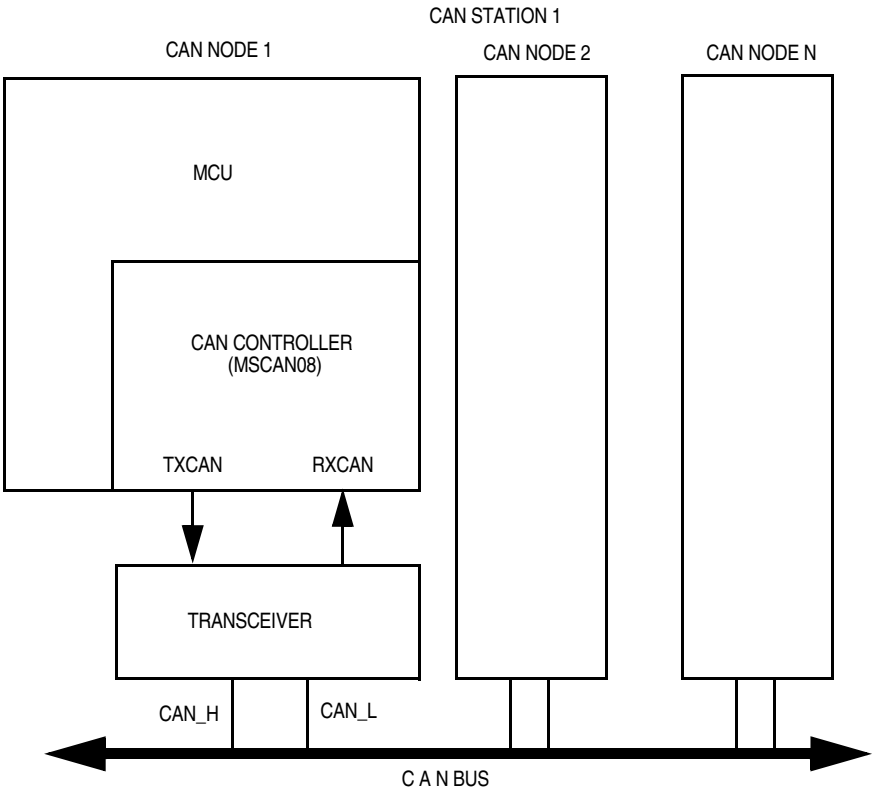



**Figure 23-1. The CAN System**

Each CAN station is connected physically to the CAN bus lines through a transceiver chip. The transceiver is capable of driving the large current needed for the CAN and has current protection against defected CAN or defected stations.

## 23.4 Message Storage

MSCAN08 facilitates a sophisticated message storage system which addresses the requirements of a broad range of network applications.

### 23.4.1 Background

Modern application layer software is built under two fundamental assumptions:

1. Any CAN node is able to send out a stream of scheduled messages without releasing the bus between two messages. Such nodes will arbitrate for the bus right after sending the previous message and will only release the bus in case of lost arbitration.
2. The internal message queue within any CAN node is organized as such that the highest priority message will be sent out first if more than one message is ready to be sent.

Above behavior cannot be achieved with a single transmit buffer. That buffer must be reloaded right after the previous message has been sent. This loading process lasts a definite amount of time and has to be completed within the inter-frame sequence (IFS) to be able to send an uninterrupted stream of messages. Even if this is feasible for limited CAN bus speeds, it requires that the CPU reacts with short latencies to the transmit interrupt.

A double buffer scheme would de-couple the re-loading of the transmit buffers from the actual message being sent and as such reduces the reactiveness requirements on the CPU. Problems may arise if the sending of a message would be finished just while the CPU re-loads the second buffer. In that case, no buffer would then be ready for transmission and the bus would be released.

At least three transmit buffers are required to meet the first of the above requirements under all circumstances. The MSCAN08 has three transmit buffers.

The second requirement calls for some sort of internal prioritisation which the MSCAN08 implements with the “local priority” concept described in 23.4.2 Receive Structures.

### 23.4.2 Receive Structures

The received messages are stored in a 2-stage input first in first out (FIFO). The two message buffers are mapped using a Ping Pong arrangement into a single memory area (see Figure 23-2). While the background receive buffer (RxBG) is exclusively associated to the MSCAN08, the foreground receive buffer (RxFG) is addressable by the CPU08. This scheme simplifies the handler software, because only one address area is applicable for the receive process.

Both buffers have a size of 13 bytes to store the CAN control bits, the identifier (standard or extended), and the data content (for details, see 23.12 Programmer’s Model of Message Storage).

The receiver full flag (RXF) in the MSCAN08 receiver flag register (CRFLG) (see 23.13.5 MSCAN08 Receiver Flag Register (CRFLG)), signals the status of the foreground receive buffer. When the buffer contains a correctly received message with matching identifier, this flag is set.

On reception, each message is checked to see if it passes the filter (for details see 23.5 Identifier Acceptance Filter) and in parallel is written into RxBG. The MSCAN08 copies the content of RxBG into RxFG[1], sets the RXF flag, and generates a receive interrupt to the CPU[2]. The user’s receive handler has to read the received message from RxFG and to reset the RXF flag to acknowledge the interrupt and to release the foreground buffer. A new message which can follow immediately after the IFS field of the CAN frame, is received into RxBG. The overwriting of the background buffer is independent of the identifier filter function.

When the MSCAN08 module is transmitting, the MSCAN08 receives its own messages into the background receive buffer, RxBG. It does NOT overwrite RxFG, generate a receive interrupt or acknowledge its own messages on the CAN bus. The exception to this rule is in loop-back mode (see 23.13.2 MSCAN08 Module Control Register 1), where the MSCAN08 treats its own messages exactly like all other incoming messages. The MSCAN08 receives its own transmitted messages in the event that it loses arbitration. If arbitration is lost, the MSCAN08 must be prepared to become receiver.

1. Only if the RXF flag is not set.
2. The receive interrupt will occur only if not masked. A polling scheme can be applied on RXF also.

An overrun condition occurs when both the foreground and the background receive message buffers are filled with correctly received messages with accepted identifiers and another message is correctly received from the bus with an accepted identifier. The latter message will be discarded and an error interrupt with overrun indication will be generated if enabled. The MSCAN08 is still able to transmit messages with both receive message buffers filled, but all incoming messages are discarded.

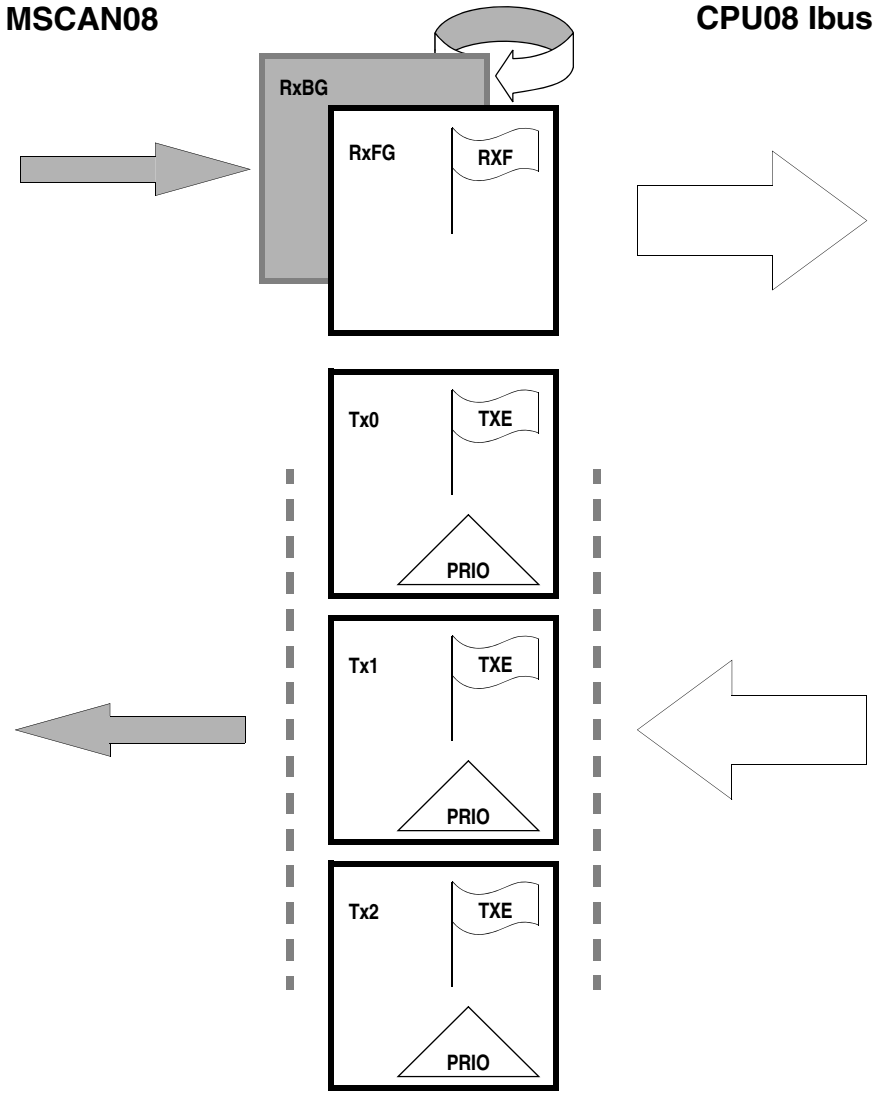



**Figure 23-2. User Model for Message Buffer Organization**

### 23.4.3 Transmit Structures

The MSCAN08 has a triple transmit buffer scheme to allow multiple messages to be set up in advance and to achieve an optimized real-time performance. The three buffers are arranged as shown in Figure 23-2.

All three buffers have a 13-byte data structure similar to the outline of the receive buffers (see 23.12 Programmer's Model of Message Storage). An additional transmit buffer priority register (TBPR) contains an 8-bit "local priority" field (PRIO) (see 23.12.5 Transmit Buffer Priority Registers).

To transmit a message, the CPU08 has to identify an available transmit buffer which is indicated by a set transmit buffer empty (TXE) flag in the MSCAN08 transmitter flag register (CTFLG) (see 23.13.7 MSCAN08 Transmitter Flag Register).

The CPU08 then stores the identifier, the control bits and the data content into one of the transmit buffers. Finally, the buffer has to be flagged ready for transmission by clearing the TXE flag.

The MSCAN08 then will schedule the message for transmission and will signal the successful transmission of the buffer by setting the TXE flag. A transmit interrupt is generated[1] when TXE is set and can be used to drive the application software to re-load the buffer.

In case more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN08 uses the local priority setting of the three buffers for prioritisation. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software sets this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being emitted from this node. The lowest binary value of the PRIO field is defined as the highest priority.

The internal scheduling process takes place whenever the MSCAN08 arbitrates for the bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message being set up in one of the three transmit buffers. As messages that are already under transmission cannot be aborted, the user has to request the abort by setting the corresponding abort request flag (ABTRQ) in the transmission control register (CTCR). The MSCAN08 will then grant the request, if possible, by setting the corresponding abort request acknowledge (ABTAK) and the TXE flag in order to release the buffer and by generating a transmit interrupt. The transmit interrupt handler software can tell from the setting of the ABTAK flag whether the message was actually aborted (ABTAK = 1) or sent (ABTAK = 0).

## 23.5 Identifier Acceptance Filter

The Identifier Acceptance Registers (CIDAR0-3) define the acceptance patterns of the standard or extended identifier (ID10-ID0 or ID28-ID0). Any of these bits can be marked ‘don’t care’ in the Identifier Mask Registers (CIDMR0-3).

A filter hit is indicated to the application on software by a set RXF (Receive Buffer Full Flag, see 23.13.5 MSCAN08 Receiver Flag Register (CRFLG)) and two bits in the Identifier Acceptance Control Register (see 23.13.9 MSCAN08 Identifier Acceptance Control Register). These Identifier Hit Flags (IDHIT1-0) clearly identify the filter section that caused the acceptance. They simplify the application software’s task to identify the cause of the receiver interrupt. In case that more than one hit occurs (two or more filters match) the lower hit has priority.

A very flexible programmable generic identifier acceptance filter has been introduced to reduce the CPU interrupt loading. The filter is programmable to operate in four different modes:

- Single identifier acceptance filter, each to be applied to a) the full 29 bits of the extended identifier and to the following bits of the CAN frame: RTR, IDE, SRR or b) the 11 bits of the standard identifier plus the RTR and IDE bits of CAN 2.0A/B messages. This mode implements a single filter for a full length CAN 2.0B compliant extended identifier. Figure 23-3 shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces a filter 0 hit.

1. The transmit interrupt will occur only if not masked. A polling scheme can be applied on TXE also.

- Two identifier acceptance filters, each to be applied to a) the 14 most significant bits of the extended identifier plus the SRR and the IDE bits of CAN2.0B messages, or b) the 11 bits of the identifier plus the RTR and IDE bits of CAN 2.0A/B messages. Figure 23-4 shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 and 1 hits.
- Four identifier acceptance filters, each to be applied to the first eight bits of the identifier. This mode implements four independent filters for the first eight bits of a CAN 2.0A/B compliant standard identifier. Figure 23-5 shows how the 32-bit filter bank (CIDAR0-3, CIDMR0-3) produces filter 0 to 3 hits.
- Closed filter. No CAN message will be copied into the foreground buffer RxFG, and the RXF flag will never be set.

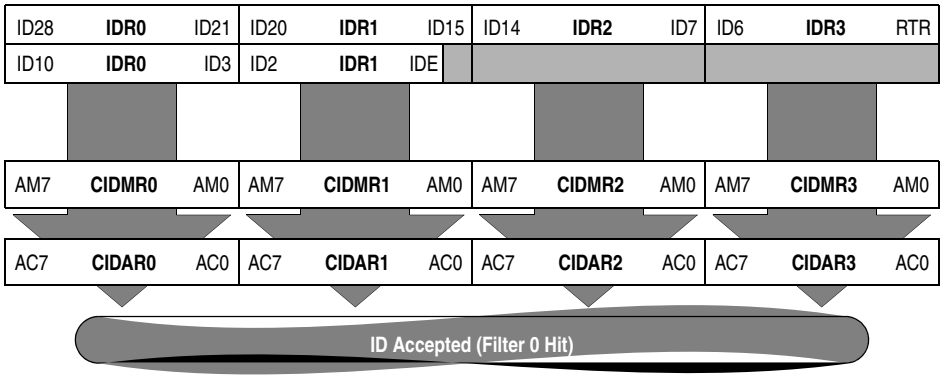



**Figure 23-3. Single 32-Bit Maskable Identifier Acceptance Filter**

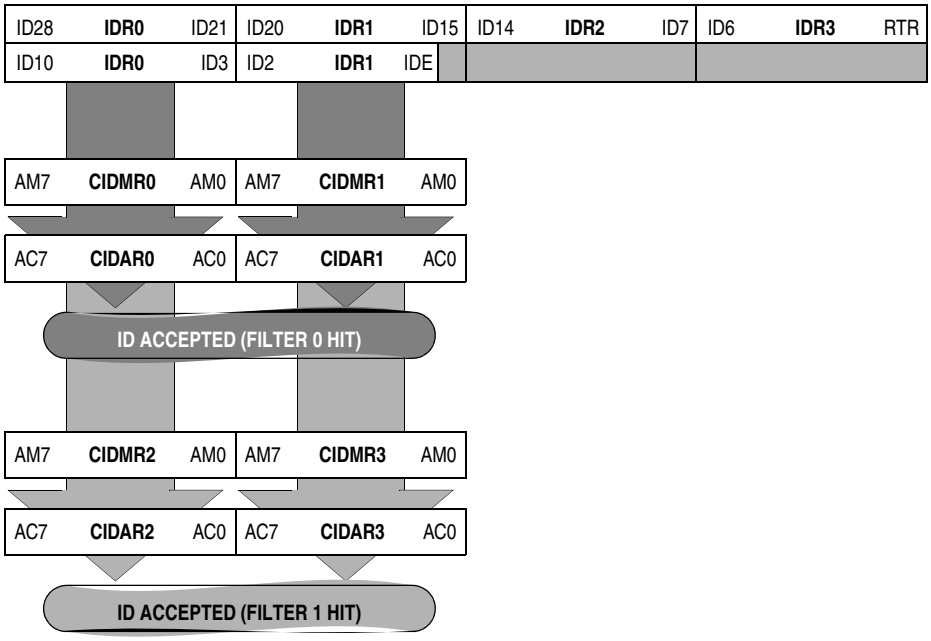



**Figure 23-4. Dual 16-Bit Maskable Acceptance Filters**

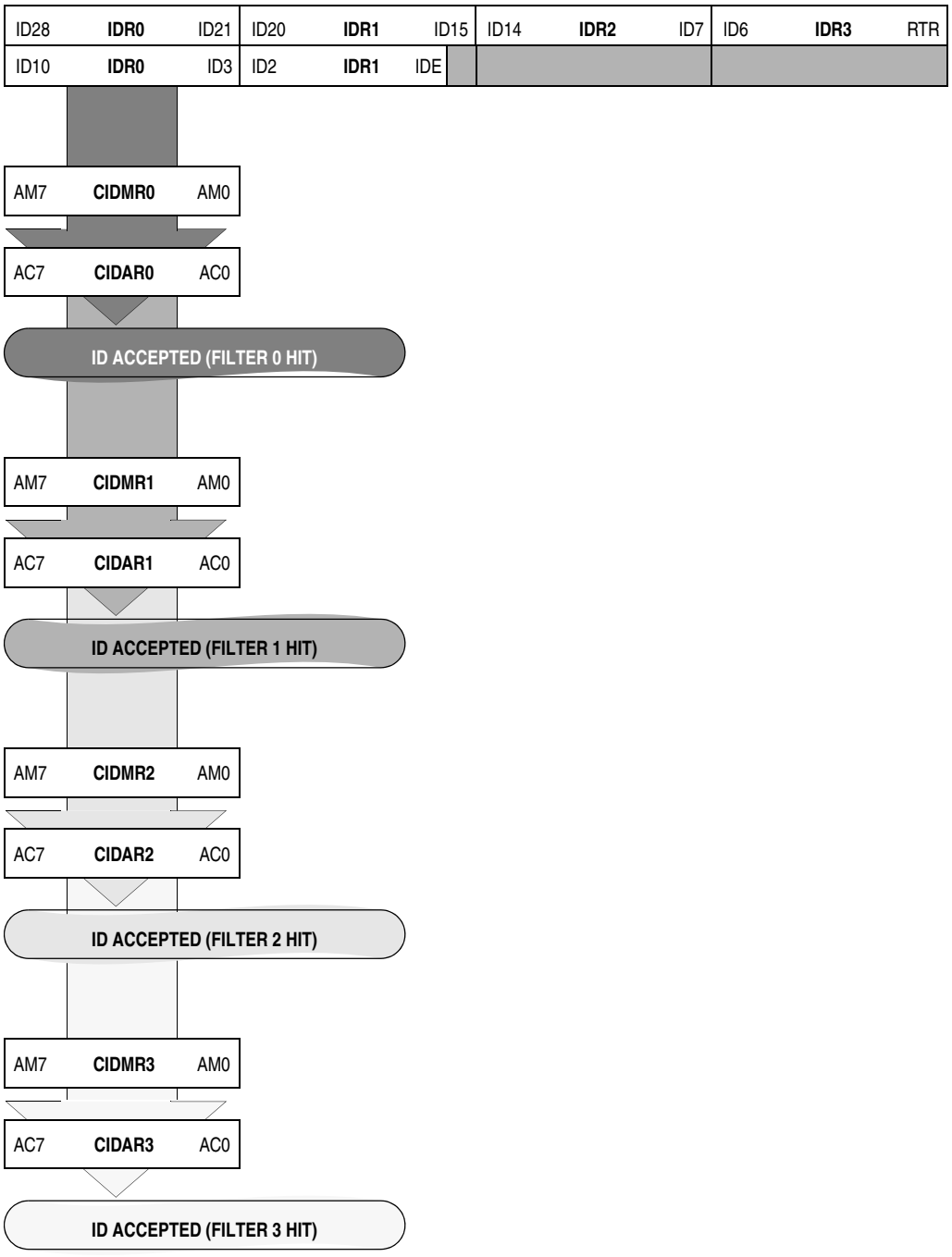



**Figure 23-5. Quadruple 8-Bit Maskable Acceptance Filters**

## 23.6 Interrupts

The MSCAN08 supports four interrupt vectors mapped onto eleven different interrupt sources, any of which can be individually masked (for details see 23.13.5 MSCAN08 Receiver Flag Register (CRFLG), to 23.13.8 MSCAN08 Transmitter Control Register).

- *Transmit Interrupt*: At least one of the three transmit buffers is empty (not scheduled) and can be loaded to schedule a message for transmission. The TXE flags of the empty message buffers are set.
- *Receive Interrupt*: A message has been received successfully and loaded into the foreground receive buffer. This interrupt will be emitted immediately after receiving the EOF symbol. The RXF flag is set.
- *Wakeup Interrupt*: An activity on the CAN bus occurred during MSCAN08 internal sleep mode or power-down mode (provided SLPAK = WUPIE = 1).
- *Error Interrupt*: An overrun, error, or warning condition occurred. The receiver flag register (CRFLG) will indicate one of the following conditions:
  - *Overrun:* An overrun condition as described in 23.4.2 Receive Structures, has occurred.
  - *Receiver Warning*: The receive error counter has reached the CPU Warning limit of 96.
  - *Transmitter Warning*: The transmit error counter has reached the CPU Warning limit of 96.
  - *Receiver Error Passive*: The receive error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
  - *Transmitter Error Passive*: The transmit error counter has exceeded the error passive limit of 127 and MSCAN08 has gone to error passive state.
  - *Bus Off*: The transmit error counter has exceeded 255 and MSCAN08 has gone to bus off state.

### 23.6.1 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in either the MSCAN08 receiver flag register (CRFLG) or the MSCAN08 transmitter flag register (CTFLG). Interrupts are pending as long as one of the corresponding flags is set. The flags in the above registers must be reset within the interrupt handler in order to handshake the interrupt. The flags are reset through writing a ‘1’ to the corresponding bit position. A flag cannot be cleared if the respective condition still prevails.

***NOTE***

*Bit manipulation instructions (BSET) shall not be used to clear interrupt flags.*

### 23.6.2 Interrupt Vectors

The MSCAN08 supports four interrupt vectors as shown in Table 23-1. The vector addresses and the relative interrupt priority are dependent on the chip integration and to be defined.

**Table 23-1. MSCAN08 Interrupt Vector Addresses**

| Function | Source | Local Mask | Global Mask |
|---|---|---|---|
| Wakeup | WUPIF | WUPIE | I Bit |
| Error Interrupts | RWRNIF | RWRNIE | |
| | TWRNIF | TWRNIE | |
| | RERRIF | RERRIE | |
| | TERRIF | TERRIE | |
| | BOFFIF | BOFFIE | |
| | OVRIF | OVRIE | |
| Receive | RXF | RXFIE | |
| Transmit | TXE0 | TXEIE0 | |
| | TXE1 | TXEIE1 | |
| | TXE2 | TXEIE2 | |

## 23.7 Protocol Violation Protection

The MSCAN08 will protect the user from accidentally violating the CAN protocol through programming errors. The protection logic implements the following features:

- The receive and transmit error counters cannot be written or otherwise manipulated.
- All registers which control the configuration of the MSCAN08 can not be modified while the MSCAN08 is on-line. The SFTRES bit in the MSCAN08 module control register (see 23.13.1 MSCAN08 Module Control Register 0) serves as a lock to protect the following registers:
  - MSCAN08 module control register 1 (CMCR1)
  - MSCAN08 bus timing register 0 and 1 (CBTR0 and CBTR1)
  - MSCAN08 identifier acceptance control register (CIDAC)
  - MSCAN08 identifier acceptance registers (CIDAR0–3)
  - MSCAN08 identifier mask registers (CIDMR0–3)
- The TxCAN pin is forced to recessive when the MSCAN08 is in any of the Low Power Modes.

## 23.8 Low Power Modes

In addition to normal mode, the MSCAN08 has three modes with reduced power consumption: Sleep, Soft Reset and Power Down modes. In Sleep and Soft Reset mode, power consumption is reduced by stopping all clocks except those to access the registers. In Power Down mode, all clocks are stopped and no power is consumed.

The WAIT and STOP instructions put the MCU in low power consumption stand-by modes.  summarizes the combinations of MSCAN08 and CPU modes. A particular combination of modes is entered for the given settings of the bits SLPAK and SFTRES. For all modes, an MSCAN wake-up interrupt can occur only if SLPAK=WUPIE=1.

**Table 23-2. MSCAN08 versus CPU Operating Modes**

| MSCAN Mode | CPU Mode | |
|---|---|---|
| | STOP | WAIT or RUN |
| Power Down | SLPAK = X[1]<br>SFTRES = X | |
| Sleep | | SLPAK = 1<br>SFTRES = 0 |
| Soft Reset | | SLPAK = 0<br>SFTRES = 1 |
| Normal | | SLPAK = 0<br>SFTRES = 0 |

1. 'X' means don't care.

### 23.8.1 MSCAN08 Sleep Mode

The CPU can request the MSCAN08 to enter the low-power mode by asserting the SLPRQ bit in the module configuration register (see Figure 23-6). The time when the MSCAN08 enters Sleep mode depends on its activity:

- if it is transmitting, it continues to transmit until there is no more message to be transmitted, and then goes into Sleep mode
- if it is receiving, it waits for the end of this message and then goes into Sleep mode
- if it is neither transmitting or receiving, it will immediately go into Sleep mode

***NOTE***

*The application software must avoid setting up a transmission (by clearing or more TXE flags) and immediately request Sleep mode (by setting SLPRQ). It then depends on the exact sequence of operations whether MSCAN08 starts transmitting or goes into Sleep mode directly.*

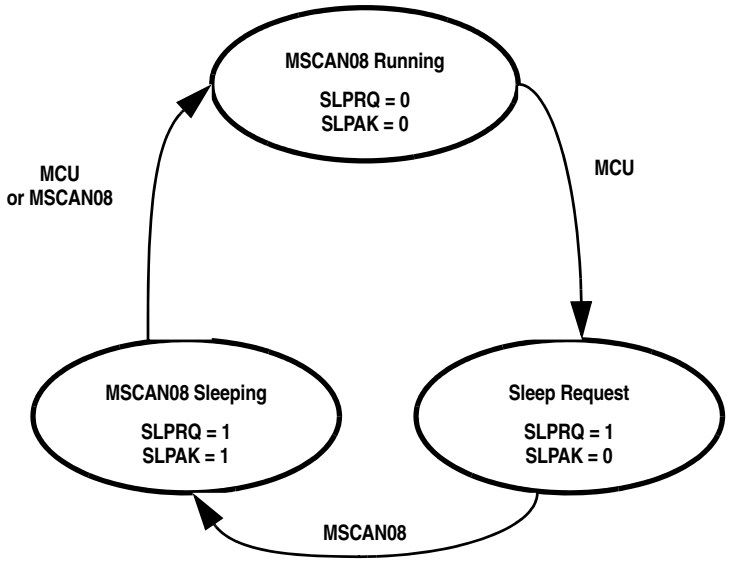



**Figure 23-6. Sleep Request/Acknowledge Cycle**

During Sleep mode, the SLPAK flag is set. The application software should use SLPAK as a handshake indication for the request (SLPRQ) to go into Sleep mode. When in Sleep mode, the MSCAN08 stops its internal clocks. However, clocks to allow register accesses still run. If the MSCAN08 is in buss-off state, it stops counting the 128*11 consecutive recessive bits due to the stopped clocks. The TxCAN pin stays in recessive state. If RXF=1, the message can be read and RXF can be cleared. Copying of RxGB into RxFG doesn't take place while in Sleep mode. It is possible to access the transmit buffers and to clear the TXE flags. No message abort takes place while in Sleep mode.

The MSCAN08 leaves Sleep mode (wake-up) when:

- bus activity occurs or
- the MCU clears the SLPRQ bit or
- the MCU sets the SFTRES bit

***NOTE***

*The MCU cannot clear the SLPRQ bit before the MSCAN08 is in Sleep mode (SLPAK=1).*

After wake-up, the MSCAN08 waits for 11 consecutive recessive bits to synchronize to the bus. As a consequence, if the MSCAN08 is woken-up by a CAN frame, this frame is not received. The receive message buffers (RxFG and RxBG) contain messages if they were received before Sleep mode was entered. All pending actions are executed upon wake-up: copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN08 is still in bus-off state after Sleep mode was left, it continues counting the 128*11 consecutive recessive bits.

### 23.8.2 MSCAN08 Soft Reset Mode

In Soft Reset mode, the MSCAN08 is stopped. Registers can still be accessed. This mode is used to initialize the module configuration, bit timing and the CAN message filter. See 23.13.1 MSCAN08 Module Control Register 0 for a complete description of the Soft Reset mode.

When setting the SFTRES bit, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

***NOTE***

*The user is responsible to take care that the MSCAN08 is not active when Soft Reset mode is entered. The recommended procedure is to bring the MSCAN08 into Sleep mode before the SFTRES bit is set.*

### 23.8.3 MSCAN08 Power Down Mode

The MSCAN08 is in Power Down mode when the CPU is in Stop mode.

When entering the Power Down mode, the MSCAN08 immediately stops all ongoing transmissions and receptions, potentially causing CAN protocol violations.

***NOTE***

*The user is responsible to take care that the MSCAN08 is not active when Power Down mode is entered. The recommended procedure is to bring the MSCAN08 into Sleep mode before the STOP instruction is executed.*

To protect the CAN bus system from fatal consequences of violations to the above rule, the MSCAN08 drives the TxCAN pin into recessive state.

In Power Down mode, no registers can be accessed.

MSCAN08 bus activity can wake the MCU from CPU Stop/MSCAN08 power-down mode. However, until the oscillator starts up and synchronisation is achieved the MSCAN08 will not respond to incoming data.

### 23.8.4 CPU Wait Mode

The MSCAN08 module remains active during CPU wait mode. The MSCAN08 will stay synchronized to the CAN bus and generates transmit, receive, and error interrupts to the CPU, if enabled. Any such interrupt will bring the MCU out of wait mode.

### 23.8.5 Programmable Wakeup Function

The MSCAN08 can be programmed to apply a low-pass filter function to the RxCAN input line while in internal sleep mode (see information on control bit WUPM in 23.13.2 MSCAN08 Module Control Register 1). This feature can be used to protect the MSCAN08 from wake-up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic inference within noisy environments.

## 23.9 Timer Link

The MSCAN08 will generate a timer signal whenever a valid frame has been received. Because the CAN specification defines a frame to be valid if no errors occurred before the EOF field has been transmitted successfully, the timer signal will be generated right after the EOF. A pulse of one bit time is generated. As the MSCAN08 receiver engine also receives the frames being sent by itself, a timer signal also will be generated after a successful transmission.

The previously described timer signal can be routed into the on-chip timer interface module (TIM).This signal is connected to the timer n channel m input[1] under the control of the timer link enable (TLNKEN) bit in the CMCR0.

After timer n has been programmed to capture rising edge events, it can be used under software control to generate 16-bit time stamps which can be stored with the received message.

## 23.10 Clock System

Figure 23-7 shows the structure of the MSCAN08 clock generation circuitry and its interaction with the clock generation module (CGM). With this flexible clocking scheme the MSCAN08 is able to handle CAN bus rates ranging from 10 kbps up to 1 Mbps.

1. The timer channel being used for the timer link is integration dependent.

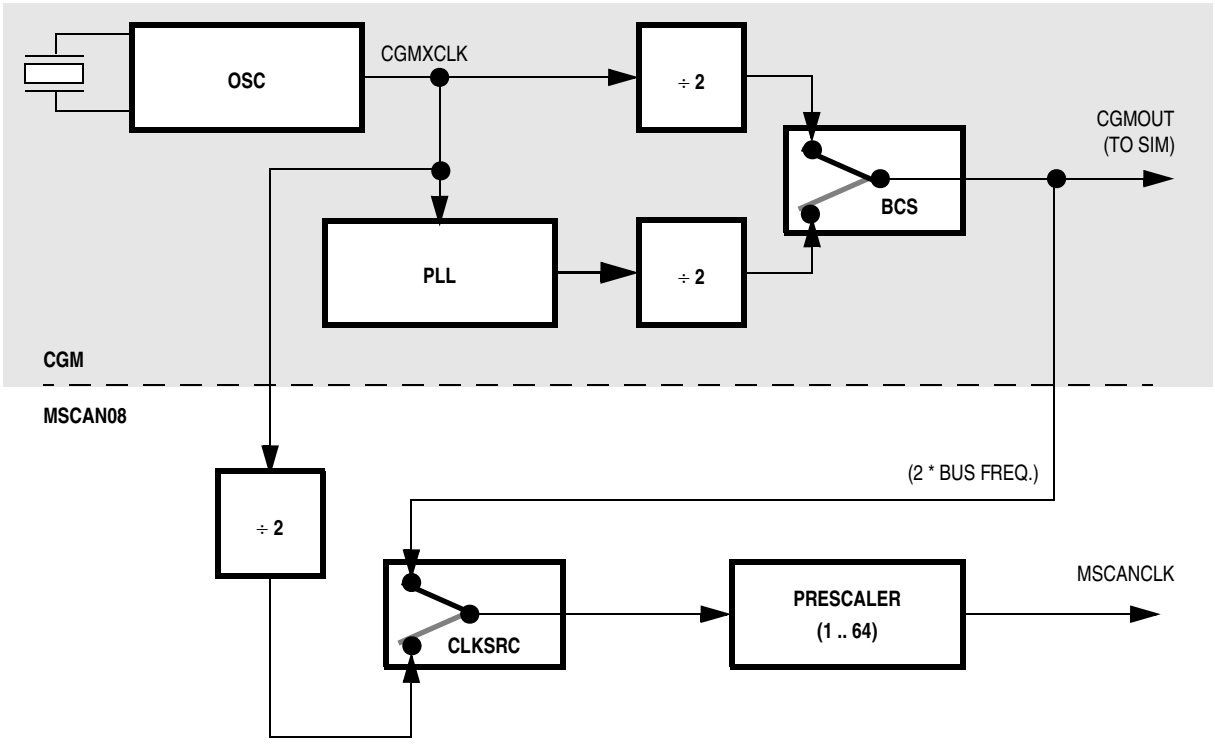



**Figure 23-7. Clocking Scheme**

The clock source bit (CLKSRC) in the MSCAN08 module control register (CMCR1) (see 23.13.1 MSCAN08 Module Control Register 0) defines whether the MSCAN08 is connected to the output of the crystal oscillator or to the PLL output.

The clock source has to be chosen such that the tight oscillator tolerance requirements (up to 0.4%) of the CAN protocol are met.

***NOTE***

*If the system clock is generated from a PLL, it is recommended to select the crystal clock source rather than the system clock source due to jitter considerations, especially at faster CAN bus rates.*

A programmable prescaler is used to generate out of the MSCAN08 clock the time quanta (Tq) clock. A time quantum is the atomic unit of time handled by the MSCAN08.

$$f_{Tq} = \frac{f_{MSCANCLK}}{Presc\ value}$$

A bit time is subdivided into three segments[1] (see Figure 23-8).

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time segment 1: This segment includes the PROP_SEG and the PHASE_SEG1 of the CAN standard. It can be programmed by setting the parameter TSEG1 to consist of 4 to 16 time quanta.

1. For further explanation of the underlying concepts please refer to ISO/DIS 11 519-1, Section 10.3.

- Time segment 2: This segment represents PHASE_SEG2 of the CAN standard. It can be programmed by setting the TSEG2 parameter to be 2 to 8 time quanta long.

$$\text{Bit rate} = \frac{f_{Tq}}{\text{No. of time quanta}}$$

The synchronization jump width (SJW) can be programmed in a range of 1 to 4 time quanta by setting the SJW parameter.

The above parameters can be set by programming the bus timing registers, CBTR0–CBTR1, see 23.13.3 MSCAN08 Bus Timing Register 0 and 23.13.4 MSCAN08 Bus Timing Register 1).

***NOTE***

*It is the user's responsibility to make sure that the bit timing settings are in compliance with the CAN standard,*

Table 23-8 gives an overview on the CAN conforming segment settings and the related parameter values.

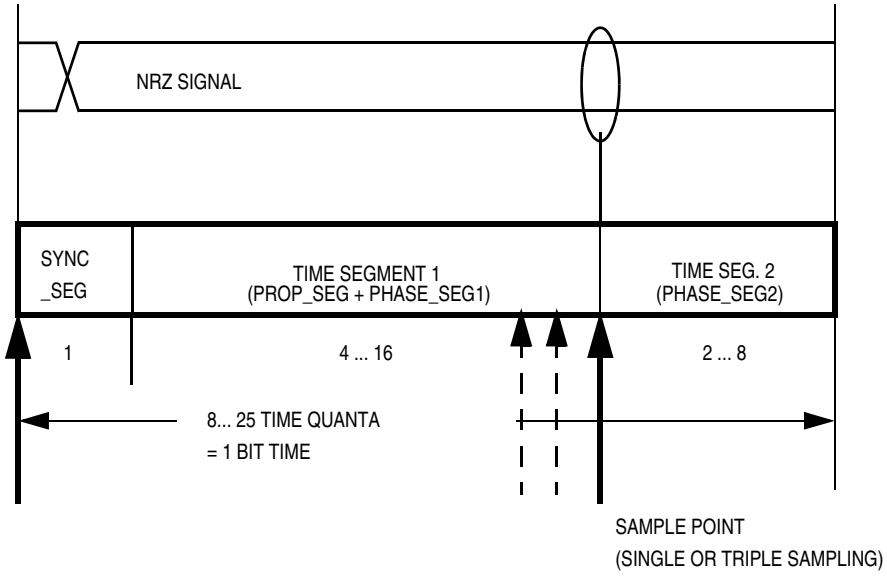



**Figure 23-8. Segments within the Bit Time**

**Table 23-3Time Segment Syntax**

| | |
|---|---|
| SYNC_SEG | System expects transitions to occur on the bus during this period. |
| Transmit point | A node in transmit mode will transfer a new value to the CAN bus at this point. |
| Sample point | A node in receive mode will sample the bus at this point. If the three samples per bit option is selected then this point marks the position of the third sample. |

**Table 23-4. CAN Standard Compliant Bit Time Segment Settings**

| Time Segment 1 | TSEG1 | Time Segment 2 | TSEG2 | Synchron. Jump Width | SJW |
|---|---|---|---|---|---|
| 5 .. 10 | 4 .. 9 | 2 | 1 | 1 .. 2 | 0 .. 1 |
| 4 .. 11 | 3 .. 10 | 3 | 2 | 1 .. 3 | 0 .. 2 |
| 5 .. 12 | 4 .. 11 | 4 | 3 | 1 .. 4 | 0 .. 3 |
| 6 .. 13 | 5 .. 12 | 5 | 4 | 1 .. 4 | 0 .. 3 |
| 7 .. 14 | 6 .. 13 | 6 | 5 | 1 .. 4 | 0 .. 3 |
| 8 .. 15 | 7 .. 14 | 7 | 6 | 1 .. 4 | 0 .. 3 |
| 9 .. 16 | 8 .. 15 | 8 | 7 | 1 .. 4 | 0 .. 3 |

## 23.11 Memory Map

The MSCAN08 occupies 128 bytes in the CPU08 memory space.

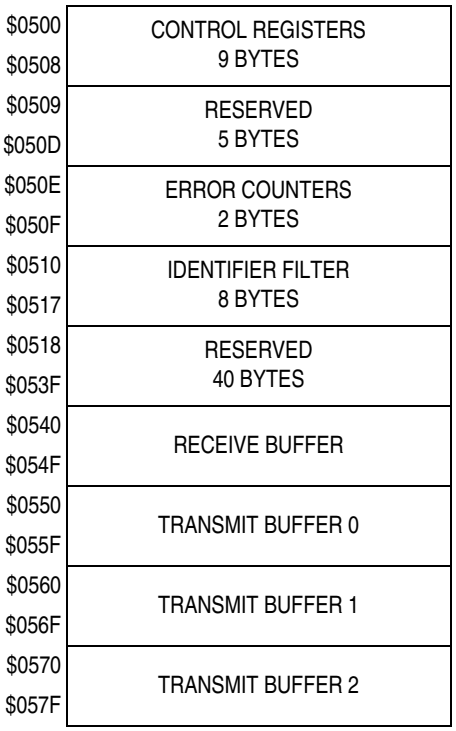



**Figure 23-9. MSCAN08 Memory Map**

## 23.12 Programmer's Model of Message Storage

This subsection details the organization of the receive and transmit message buffers and the associated control registers. For reasons of programmer interface simplification, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13-byte data structure. An additional transmit buffer priority register (TBPR) is defined for the transmit buffers.

| Addr | Register Name |
|---|---|
| $05x0 | IDENTIFIER REGISTER 0 |
| $05x1 | IDENTIFIER REGISTER 1 |
| $05x2 | IDENTIFIER REGISTER 2 |
| $05x3 | IDENTIFIER REGISTER 3 |
| $05x4 | DATA SEGMENT REGISTER 0 |
| $05x5 | DATA SEGMENT REGISTER 1 |
| $05x6 | DATA SEGMENT REGISTER 2 |
| $05x7 | DATA SEGMENT REGISTER 3 |
| $05x8 | DATA SEGMENT REGISTER 4 |
| $05x9 | DATA SEGMENT REGISTER 5 |
| $05xA | DATA SEGMENT REGISTER 6 |
| $05xB | DATA SEGMENT REGISTER 7 |
| $05xC | DATA LENGTH REGISTER |
| $05xD | TRANSMIT BUFFER PRIORITY REGISTER(1) |
| $05xE | UNUSED |
| $05xF | UNUSED |

1. Where x equals the following:
   x = 4 for receiver buffer
   x = 5 for transmit buffer 1
   x = 6 for transmit buffer 2
   x = 7 for transmit buffer 3
2. Not applicable for receive buffers

**Figure 23-10. Message Buffer Organization**

### 23.12.1 Message Buffer Outline

Figure 23-11 shows the common 13-byte data structure of receive and transmit buffers for extended identifiers. The mapping of standard identifiers into the IDR registers is shown in Figure 23-12. All bits of the 13-byte data structure are undefined out of reset.

***NOTE***

*The foreground receive buffer can be read anytime but cannot be written.*
*The transmit buffers can be read or written anytime.*

### 23.12.2 Identifier Registers

The identifiers consist of either 11 bits (ID10–ID0) for the standard, or 29 bits (ID28–ID0) for the extended format. ID10/28 is the most significant bit and is transmitted first on the bus during the arbitration procedure. The priority of an identifier is defined to be highest for the smallest binary number.

**SRR — Substitute Remote Request**

This fixed recessive bit is used only in extended format. It must be set to 1 by the user for transmission buffers and will be stored as received on the CAN bus for receive buffers.

| Addr | Register | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $05b0 | IDR0 | Read:<br>Write: | ID28 | ID27 | ID26 | ID25 | ID24 | ID23 | ID22 | ID21 |
| $05b1 | IDR1 | Read:<br>Write: | ID20 | ID19 | ID18 | SRR (=1) | IDE (=1) | ID17 | ID16 | ID15 |
| $05b2 | IDR2 | Read:<br>Write: | ID14 | ID13 | ID12 | ID11 | ID10 | ID9 | ID8 | ID7 |
| $05b3 | IDR3 | Read:<br>Write: | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | RTR |
| $05b4 | DSR0 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b5 | DSR1 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b6 | DSR2 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b7 | DSR3 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b8 | DSR4 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05b9 | DSR5 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05bA | DSR6 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05bB | DSR7 | Read:<br>Write: | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 |
| $05bC | DLR | Read:<br>Write: | | | | | DLC3 | DLC2 | DLC1 | DLC0 |

= Unimplemented

**Figure 23-11. Receive/Transmit Message Buffer Extended Identifier (IDRn)**

| Addr | Register | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $05b0 | IDR0 | Read:<br>Write: | ID10 | ID9 | ID8 | ID7 | ID6 | ID5 | ID4 | ID3 |
| $05b1 | IDR1 | Read:<br>Write: | ID2 | ID1 | ID0 | RTR | IDE(=0) | | | |
| $05b2 | IDR2 | Read:<br>Write: | | | | | | | | |
| $05b3 | IDR3 | Read:<br>Write: | | | | | | | | |

= Unimplemented

**Figure 23-12. Standard Identifier Mapping**

**IDE — ID Extended**

This flag indicates whether the extended or standard identifier format is applied in this buffer. In case of a receive buffer, the flag is set as being received and indicates to the CPU how to process the buffer identifier registers. In case of a transmit buffer, the flag indicates to the MSCAN08 what type of identifier to send.

1 = Extended format, 29 bits
0 = Standard format, 11 bits

**RTR — Remote Transmission Request**

This flag reflects the status of the remote transmission request bit in the CAN frame. In case of a receive buffer, it indicates the status of the received frame and supports the transmission of an answering frame in software. In case of a transmit buffer, this flag defines the setting of the RTR bit to be sent.

1 = Remote frame
0 = Data frame

### 23.12.3 Data Length Register (DLR)

This register keeps the data length field of the CAN frame.

**DLC3–DLC0 — Data Length Code Bits**

The data length code contains the number of bytes (data byte count) of the respective message. At transmission of a remote frame, the data length code is transmitted as programmed while the number of transmitted bytes is always 0. The data byte count ranges from 0 to 8 for a data frame. Table 23-5 shows the effect of setting the DLC bits.

**Table 23-5. Data Length Codes**

| Data Length Code | | | | Data Byte Count |
|---|---|---|---|---|
| DLC3 | DLC2 | DLC1 | DLC0 | |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 0 | 2 |
| 0 | 0 | 1 | 1 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 0 | 1 | 0 | 1 | 5 |
| 0 | 1 | 1 | 0 | 6 |
| 0 | 1 | 1 | 1 | 7 |
| 1 | 0 | 0 | 0 | 8 |

### 23.12.4 Data Segment Registers (DSRn)

The eight data segment registers contain the data to be transmitted or received. The number of bytes to be transmitted or being received is determined by the data length code in the corresponding DLR.

### 23.12.5 Transmit Buffer Priority Registers

Address: $05bD

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | PRIO7 | PRIO6 | PRIO5 | PRIO4 | PRIO3 | PRIO2 | PRIO1 | PRIO0 |
| Reset: | Unaffected by Reset | | | | | | | |

**Figure 23-13. Transmit Buffer Priority Register (TBPR)**

**PRIO7–PRIO0 — Local Priority**

This field defines the local priority of the associated message buffer. The local priority is used for the internal prioritisation process of the MSCAN08 and is defined to be highest for the smallest binary number. The MSCAN08 implements the following internal prioritisation mechanism:

- All transmission buffers with a cleared TXE flag participate in the prioritisation right before the SOF is sent.
- The transmission buffer with the lowest local priority field wins the prioritisation.
- In case more than one buffer has the same lowest priority, the message buffer with the lower index number wins.

## 23.13 Programmer's Model of Control Registers

The programmer's model has been laid out for maximum simplicity and efficiency. Figure 23-14 gives an overview on the control register block of the MSCAN08.

| Addr | Register | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0500 | CMCR0 | Read: | 0 | 0 | 0 | SYNCH | TLNKEN | SLPAK | SLPRQ | SFTRES |
| | | Write: | | | | | | | | |
| $0501 | CMCR1 | Read: | 0 | 0 | 0 | 0 | 0 | LOOPB | WUPM | CLKSRC |
| | | Write: | | | | | | | | |
| $0502 | CBTR0 | Read: | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| | | Write: | | | | | | | | |
| $0503 | CBTR1 | Read: | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| | | Write: | | | | | | | | |
| $0504 | CRFLG | Read: | WUPIF | RWRNIF | TWRNIF | RERRIF | TERRIF | BOFFIF | OVRIF | RXF |
| | | Write: | | | | | | | | |
| $0505 | CRIER | Read: | WUPIE | RWRNIE | TWRNIE | RERRIE | TERRIE | BOFFIE | OVRIE | RXFIE |
| | | Write: | | | | | | | | |
| $0506 | CTFLG | Read: | 0 | ABTAK2 | ABTAK1 | ABTAK0 | 0 | TXE2 | TXE1 | TXE0 |
| | | Write: | | | | | | | | |
| $0507 | CTCR | Read: | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| | | Write: | | | | | | | | |
| $0508 | CIDAC | Read: | 0 | 0 | IDAM1 | IDAM0 | 0 | 0 | IDHIT1 | IDHIT0 |
| | | Write: | | | | | | | | |
| $0509 | Reserved | Read: | R | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| $050E | CRXERR | Read: | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| | | Write: | | | | | | | | |
| $050F | CTXERR | Read: | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| | | Write: | | | | | | | | |
| $0510 | CIDAR0 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |
| $0511 | CIDAR1 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |
| $0512 | CIDAR2 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |
| $0513 | CIDAR3 | Read: | AC7 | AC6 | AC5 | AC4 | AC3 | AC2 | AC1 | AC0 |
| | | Write: | | | | | | | | |
| $0514 | CIDMR0 | Read: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | Write: | | | | | | | | |
| $0515 | CIDMR1 | Read: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | Write: | | | | | | | | |
| $0516 | CIDMR2 | Read: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | Write: | | | | | | | | |
| $0517 | CIDMR3 | Read: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure 23-14. MSCAN08 Control Register Structure**

### 23.13.1 MSCAN08 Module Control Register 0

Address: $0500

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | SYNCH | TLNKEN | SLPAK | SLPRQ | SFTRES |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

= Unimplemented

**Figure 23-15. Module Control Register 0 (CMCR0)**

**SYNCH — Synchronized Status**

This bit indicates whether the MSCAN08 is synchronized to the CAN bus and as such can participate in the communication process.

1 = MSCAN08 synchronized to the CAN bus
0 = MSCAN08 not synchronized to the CAN bus

**TLNKEN — Timer Enable**

This flag is used to establish a link between the MSCAN08 and the on-chip timer (see 23.9 Timer Link).

1 = The MSCAN08 timer signal output is connected to the timer input.
0 = The port is connected to the timer input.

**SLPAK — Sleep Mode Acknowledge**

This flag indicates whether the MSCAN08 is in module internal sleep mode. It shall be used as a handshake for the sleep mode request (see 23.8.1 MSCAN08 Sleep Mode). If the MSCAN08 detects bus activity while in Sleep mode, it clears the flag.

1 = Sleep – MSCAN08 in internal sleep mode
0 = Wakeup – MSCAN08 is not in Sleep mode

**SLPRQ — Sleep Request, Go to Internal Sleep Mode**

This flag requests the MSCAN08 to go into an internal power-saving mode (see 23.8.1 MSCAN08 Sleep Mode).

1 = Sleep — The MSCAN08 will go into internal sleep mode.
0 = Wakeup — The MSCAN08 will function normally.

**SFTRES — Soft Reset**

When this bit is set by the CPU, the MSCAN08 immediately enters the soft reset state. Any ongoing transmission or reception is aborted and synchronization to the bus is lost.

The following registers enter and stay in their hard reset state: CMCR0, CRFLG, CRIER, CTFLG, and CTCR.

The registers CMCR1, CBTR0, CBTR1, CIDAC, CIDAR0–3, and CIDMR0–3 can only be written by the CPU when the MSCAN08 is in soft reset state. The values of the error counters are not affected by soft reset.

When this bit is cleared by the CPU, the MSCAN08 tries to synchronize to the CAN bus. If the MSCAN08 is not in bus-off state, it will be synchronized after 11 recessive bits on the bus; if the MSCAN08 is in bus-off state, it continues to wait for 128 occurrences of 11 recessive bits.

Clearing SFTRES and writing to other bits in CMCR0 must be in separate instructions.

1 = MSCAN08 in soft reset state
0 = Normal operation

### 23.13.2 MSCAN08 Module Control Register 1

Address: $0501

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | 0 | 0 | LOOPB | WUPM | CLKSRC |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 23-16. Module Control Register (CMCR1)**

**LOOPB — Loop Back Self-Test Mode**

When this bit is set, the MSCAN08 performs an internal loop back which can be used for self-test operation: the bit stream output of the transmitter is fed back to the receiver internally. The RxCAN input pin is ignored and the TxCAN output goes to the recessive state (logic ‘1’). The MSCAN08 behaves as it does normally when transmitting and treats its own transmitted message as a message received from a remote node. In this state the MSCAN08 ignores the bit sent during the ACK slot of the CAN frame Acknowledge field to insure proper reception of its own message. Both transmit and receive interrupt are generated.

1 = Activate loop back self-test mode
0 = Normal operation

**WUPM — Wakeup Mode**

This flag defines whether the integrated low-pass filter is applied to protect the MSCAN08 from spurious wakeups (see 23.8.5 Programmable Wakeup Function).

1 = MSCAN08 will wake up the CPU only in cases of a dominant pulse on the bus which has a length of at least $t_{wup}$.
0 = MSCAN08 will wake up the CPU after any recessive to dominant edge on the CAN bus.

**CLKSRC — Clock Source**

This flag defines which clock source the MSCAN08 module is driven from (see 23.10 Clock System).

1 = The MSCAN08 clock source is CGMOUT (see Figure 23-7).
0 = The MSCAN08 clock source is CGMXCLK/2 (see Figure 23-7).

***NOTE***

*The CMCR1 register can be written only if the SFTRES bit in the MSCAN08 module control register is set*

### 23.13.3 MSCAN08 Bus Timing Register 0

Address: $0502

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | SJW1 | SJW0 | BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 23-17. Bus Timing Register 0 (CBTR0)**

**SJW1 and SJW0 — Synchronization Jump Width**

The synchronization jump width (SJW) defines the maximum number of time quanta ($T_q$) clock cycles by which a bit may be shortened, or lengthened, to achieve resynchronization on data transitions on the bus (see Table 23-6).

**Table 23-6. Synchronization Jump Width**

| SJW1 | SJW0 | Synchronization Jump Width |
|---|---|---|
| 0 | 0 | 1 $T_q$ cycle |
| 0 | 1 | 2 $T_q$ cycle |
| 1 | 0 | 3 $T_q$ cycle |
| 1 | 1 | 4 $T_q$ cycle |

**BRP5–BRP0 — Baud Rate Prescaler**

These bits determine the time quanta ($T_q$) clock, which is used to build up the individual bit timing, according to Table 23-7.

**Table 23-7. Baud Rate Prescaler**

| BRP5 | BRP4 | BRP3 | BRP2 | BRP1 | BRP0 | Prescaler Value (P) |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 2 |
| 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| : | : | : | : | : | : | : |
| : | : | : | : | : | : | : |
| 1 | 1 | 1 | 1 | 1 | 1 | 64 |

***NOTE***

*The CBTR0 register can be written only if the SFTRES bit in the MSCAN08 module control register is set.*

### 23.13.4 MSCAN08 Bus Timing Register 1

Address: $0503

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | SAMP | TSEG22 | TSEG21 | TSEG20 | TSEG13 | TSEG12 | TSEG11 | TSEG10 |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 23-18. Bus Timing Register 1 (CBTR1)**

**SAMP — Sampling**

This bit determines the number of serial bus samples to be taken per bit time. If set, three samples per bit are taken, the regular one (sample point) and two preceding samples, using a majority rule. For higher bit rates, SAMP should be cleared, which means that only one sample will be taken per bit.

1 = Three samples per bit(1)
0 = One sample per bit

**TSEG22–TSEG10 — Time Segment**

Time segments within the bit time fix the number of clock cycles per bit time and the location of the sample point. Time segment 1 (TSEG1) and time segment 2 (TSEG2) are programmable as shown in Table 23-8.

**Table 23-8. Time Segment Values**

| TSEG13 | TSEG12 | TSEG11 | TSEG10 | Time Segment 1 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 $T_q$ Cycle(1) |
| 0 | 0 | 0 | 1 | 2 $T_q$ Cycles(1) |
| 0 | 0 | 1 | 0 | 3$T_q$ Cycles(1) |
| 0 | 0 | 1 | 1 | 4 $T_q$ Cycles |
| . | . | . | . | . |
| . | . | . | . | . |
| 1 | 1 | 1 | 1 | 16 $T_q$ Cycles |

| TSEG22 | TSEG21 | TSEG20 | Time Segment 2 |
|---|---|---|---|
| 0 | 0 | 0 | 1 $T_q$ Cycle(1) |
| 0 | 0 | 1 | 2 $T_q$ Cycles |
| . | . | . | . |
| . | . | . | . |
| 1 | 1 | 1 | 8$T_q$ Cycles |

1. This setting is not valid. Please refer to Table 23-4 for valid settings.

The bit time is determined by the oscillator frequency, the baud rate prescaler, and the number of time quanta ($T_q$) clock cycles per bit as shown in Table 23-8).

$$\text{Bit time} = \frac{\text{Pres value}}{f_{MSCANCLK}} \bullet \text{number of Time Quanta}$$

***NOTE***

*The CBTR1 register can only be written if the SFTRES bit in the MSCAN08 module control register is set.*

1. In this case PHASE_SEG1 must be at least 2 time quanta.

### 23.13.5 MSCAN08 Receiver Flag Register (CRFLG)

All bits of this register are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. A flag can be cleared only when the condition which caused the setting is valid no more. Writing a 0 has no effect on the flag setting. Every flag has an associated interrupt enable flag in the CRIER register. A hard or soft reset will clear the register.

Address: $0504

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | WUPIF | RWRNIF | TWRNIF | RERRIF | TERRIF | BOFFIF | OVRIF | RXF |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 23-19. Receiver Flag Register (CRFLG)**

**WUPIF — Wakeup Interrupt Flag**

If the MSCAN08 detects bus activity while in Sleep mode, it sets the WUPIF flag. If not masked, a wake-up interrupt is pending while this flag is set.

1 = MSCAN08 has detected activity on the bus and requested wake-up.
0 = No wake-up interrupt has occurred.

**RWRNIF — Receiver Warning Interrupt Flag**

This flag is set when the MSCAN08 goes into warning status due to the receive error counter (REC) exceeding 96 and neither one of the Error Interrupt flags or the Bus-off Interrupt flag is set(1). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into receiver warning status.
0 = No receiver warning status has been reached.

**TWRNIF — Transmitter Warning Interrupt Flag**

This flag is set when the MSCAN08 goes into warning status due to the transmit error counter (TEC) exceeding 96 and neither one of the error interrupt flags or the bus-off interrupt flag is set(2). If not masked, an error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into transmitter warning status.
0 = No transmitter warning status has been reached.

**RERRIF — Receiver Error Passive Interrupt Flag**

This flag is set when the MSCAN08 goes into error passive status due to the receive error counter exceeding 127 and the bus-off interrupt flag is not set(3). If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN08 has gone into receiver error passive status.
0 = No receiver error passive status has been reached.

1. Condition to set the flag: RWRNIF = (96 ≤ REC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
2. Condition to set the flag: TWRNIF = (96 ≤ TEC) & $\overline{\text{RERRIF}}$ & $\overline{\text{TERRIF}}$ & $\overline{\text{BOFFIF}}$
3. Condition to set the flag: RERRIF = (127 ≤ REC ≤ 255) & $\overline{\text{BOFFIF}}$

**TERRIF — Transmitter Error Passive Interrupt Flag**

This flag is set when the MSCAN08 goes into error passive status due to the Transmit Error counter exceeding 127 and the Bus-off interrupt flag is not set[1]. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN08 went into transmit error passive status.
0 = No transmit error passive status has been reached.

**BOFFIF — Bus-Off Interrupt Flag**

This flag is set when the MSCAN08 goes into bus-off status, due to the transmit error counter exceeding 255. It cannot be cleared before the MSCAN08 has monitored 128 times 11 consecutive ‘recessive’ bits on the bus. If not masked, an Error interrupt is pending while this flag is set.

1 = MSCAN08has gone into bus-off status.
0 = No bus-off status has bee reached.

**OVRIF — Overrun Interrupt Flag**

This flag is set when a data overrun condition occurs. If not masked, an error interrupt is pending while this flag is set.

1 = A data overrun has been detected since last clearing the flag.
0 = No data overrun has occurred.

**RXF — Receive Buffer Full**

The RXF flag is set by the MSCAN08 when a new message is available in the foreground receive buffer. This flag indicates whether the buffer is loaded with a correctly received message. After the CPU has read that message from the receive buffer the RXF flag must be cleared to release the buffer. A set RXF flag prohibits the exchange of the background receive buffer into the foreground buffer. If not masked, a receive interrupt is pending while this flag is set.

1 = The receive buffer is full. A new message is available.
0 = The receive buffer is released (not full).

***NOTE***

*To ensure data integrity, no registers of the receive buffer shall be read while the RXF flag is cleared.*

***NOTE***

*The CRFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.*

1. Condition to set the flag: TERRIF = (128 ≤ TEC ≤ 255) & $\overline{\text{BOFFIF}}$

### 23.13.6 MSCAN08 Receiver Interrupt Enable Register

Address: $0505

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | WUPIE | RWRNIE | TWRNIE | RERRIE | TERRIE | BOFFIE | OVRIE | RXFIE |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 23-20. Receiver Interrupt Enable Register (CRIER)**

**WUPIE — Wakeup Interrupt Enable**
1 = A wakeup event will result in a wakeup interrupt.
0 = No interrupt will be generated from this event.

**RWRNIE — Receiver Warning Interrupt Enable**
1 = A receiver warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**TWRNIE — Transmitter Warning Interrupt Enable**
1 = A transmitter warning status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**RERRIE — Receiver Error Passive Interrupt Enable**
1 = A receiver error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**TERRIE — Transmitter Error Passive Interrupt Enable**
1 = A transmitter error passive status event will result in an error interrupt.
0 = No interrupt is generated from this event.

**BOFFIE — Bus-Off Interrupt Enable**
1 = A bus-off event will result in an error interrupt.
0 = No interrupt is generated from this event.

**OVRIE — Overrun Interrupt Enable**
1 = An overrun event will result in an error interrupt.
0 = No interrupt is generated from this event.

**RXFIE — Receiver Full Interrupt Enable**
1 = A receive buffer full (successful message reception) event will result in a receive interrupt.
0 = No interrupt will be generated from this event.

***NOTE***

*The CRIER register is held in the reset state when the SFTRES bit in CMCR0 is set.*

## 23.13.7 MSCAN08 Transmitter Flag Register

The Abort Acknowledge flags are read only. The Transmitter Buffer Empty flags are read and clear only. A flag can be cleared by writing a 1 to the corresponding bit position. Writing a 0 has no effect on the flag setting. The Transmitter Buffer Empty flags each have an associated interrupt enable bit in the CTCR register. A hard or soft reset will resets the register.

Address: $0506 5

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | ABTAK2 | ABTAK1 | ABTAK0 | 0 | TXE2 | TXE1 | TXE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

= Unimplemented

**Figure 23-21. Transmitter Flag Register (CTFLG)**

**ABTAK2–ABTAK0 — Abort Acknowledge**

This flag acknowledges that a message has been aborted due to a pending abort request from the CPU. After a particular message buffer has been flagged empty, this flag can be used by the application software to identify whether the message has been aborted successfully or has been sent. The ABTAKx flag is cleared implicitly whenever the corresponding TXE flag is cleared.

1 = The message has been aborted.
0 = The message has not been aborted, thus has been sent out.

**TXE2–TXE0 — Transmitter Empty**

This flag indicates that the associated transmit message buffer is empty, thus not scheduled for transmission. The CPU must handshake (clear) the flag after a message has been set up in the transmit buffer and is due for transmission. The MSCAN08 sets the flag after the message has been sent successfully. The flag is also set by the MSCAN08 when the transmission request was successfully aborted due to a pending abort request (see 23.12.5 Transmit Buffer Priority Registers). If not masked, a receive interrupt is pending while this flag is set.

Clearing a TXEx flag also clears the corresponding ABTAKx flag (ABTAK, see above). When a TXEx flag is set, the corresponding ABTRQx bit (ABTRQ, see 23.13.8 MSCAN08 Transmitter Control Register) is cleared.

1 = The associated message buffer is empty (not scheduled).
0 = The associated message buffer is full (loaded with a message due for transmission).

***NOTE***

*To ensure data integrity, no registers of the transmit buffers should be written to while the associated TXE flag is cleared.*

***NOTE***

*The CTFLG register is held in the reset state when the SFTRES bit in CMCR0 is set.*

### 23.13.8 MSCAN08 Transmitter Control Register

Address: $0507

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | ABTRQ2 | ABTRQ1 | ABTRQ0 | 0 | TXEIE2 | TXEIE1 | TXEIE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 23-22. Transmitter Control Register (CTCR)**

**ABTRQ2–ABTRQ0 — Abort Request**

The CPU sets an ABTRQx bit to request that an already scheduled message buffer (TXE = 0) be aborted. The MSCAN08 will grant the request if the message has not already started transmission, or if the transmission is not successful (lost arbitration or error). When a message is aborted the associated TXE and the abort acknowledge flag (ABTAK) (see 23.13.7 MSCAN08 Transmitter Flag Register) will be set and an TXE interrupt is generated if enabled. The CPU cannot reset ABTRQx. ABTRQx is cleared implicitly whenever the associated TXE flag is set.

1 = Abort request pending
0 = No abort request

***NOTE***

*The software must not clear one or more of the TXE flags in CTFLG and simultaneously set the respective ABTRQ bit(s).*

**TXEIE2–TXEIE0 — Transmitter Empty Interrupt Enable**

1 = A transmitter empty (transmit buffer available for transmission) event results in a transmitter empty interrupt.
0 = No interrupt is generated from this event.

***NOTE***

*The CTCR register is held in the reset state when the SFTRES bit in CMCR0 is set.*

### 23.13.9 MSCAN08 Identifier Acceptance Control Register

Address: $0508

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | IDAM1 | IDAM0 | 0 | 0 | IDHIT1 | IDHIT0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 23-23. Identifier Acceptance Control Register (CIDAC)**

**IDAM1–IDAM0— Identifier Acceptance Mode**

The CPU sets these flags to define the identifier acceptance filter organization (see 23.5 Identifier Acceptance Filter). Table 23-9 summarizes the different settings. In “filter closed” mode no messages will be accepted so that the foreground buffer will never be reloaded.

**Table 23-9. Identifier Acceptance Mode Settings**

| IDAM1 | IDAM0 | Identifier Acceptance Mode |
|---|---|---|
| 0 | 0 | Single 32-Bit Acceptance Filter |
| 0 | 1 | Two 16-Bit Acceptance Filter |
| 1 | 0 | Four 8-Bit Acceptance Filters |
| 1 | 1 | Filter Closed |

**IDHIT1–IDHIT0— Identifier Acceptance Hit Indicator**

The MSCAN08 sets these flags to indicate an identifier acceptance hit (see 23.5 Identifier Acceptance Filter). Table 23-9 summarizes the different settings.

**Table 23-10. Identifier Acceptance Hit Indication**

| IDHIT1 | IDHIT0 | Identifier Acceptance Hit |
|---|---|---|
| 0 | 0 | Filter 0 Hit |
| 0 | 1 | Filter 1 Hit |
| 1 | 0 | Filter 2 Hit |
| 1 | 1 | Filter 3 Hit |

The IDHIT indicators are always related to the message in the foreground buffer. When a message gets copied from the background to the foreground buffer, the indicators are updated as well.

***NOTE***

*The CIDAC register can be written only if the SFTRES bit in the CMCR0 is set.*

### 23.13.10 MSCAN08 Receive Error Counter

Address: $050E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | RXERR7 | RXERR6 | RXERR5 | RXERR4 | RXERR3 | RXERR2 | RXERR1 | RXERR0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 23-24. Receiver Error Counter (CRXERR)**

This register reflects the status of the MSCAN08 receive error counter. The register is read only.

### 23.13.11 MSCAN08 Transmit Error Counter

Address: $050F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | TXERR7 | TXERR6 | TXERR5 | TXERR4 | TXERR3 | TXERR2 | TXERR1 | TXERR0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 23-25. Transmit Error Counter (CTXERR)**

This register reflects the status of the MSCAN08 transmit error counter. The register is read only.

***NOTE***

*Both error counters may only be read when in Sleep or Soft Reset mode.*

## 23.13.12 MSCAN08 Identifier Acceptance Registers

On reception each message is written into the background receive buffer. The CPU is only signalled to read the message, however, if it passes the criteria in the identifier acceptance and identifier mask registers (accepted); otherwise, the message will be overwritten by the next message (dropped).

The acceptance registers of the MSCAN08 are applied on the IDR0 to IDR3 registers of incoming messages in a bit by bit manner.

For extended identifiers, all four acceptance and mask registers are applied. For standard identifiers only the first two (CIDMR0/1 and CIDAR0/1) are applied.

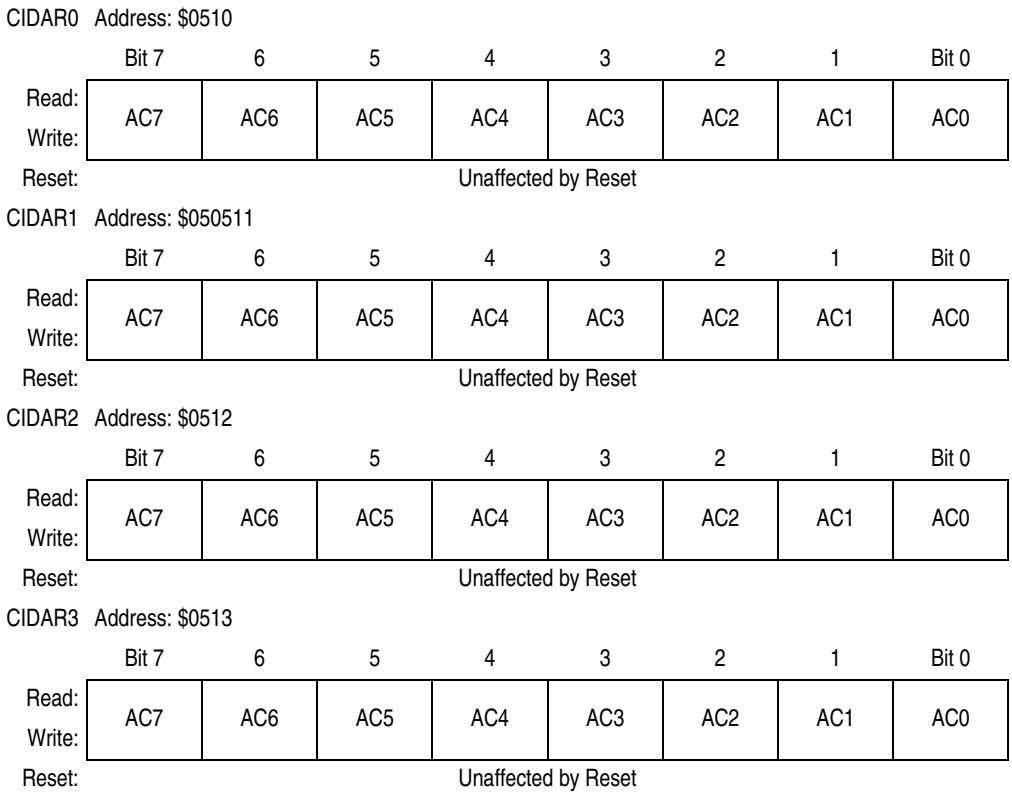



**Figure 23-26. Identifier Acceptance Registers (CIDAR0–CIDAR3)**

**AC7–AC0 — Acceptance Code Bits**

AC7–AC0 comprise a user-defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

***NOTE***

*The CIDAR0–3 registers can be written only if the SFTRES bit in CMCR0 is set*

### 23.13.13 MSCAN08 Identifier Mask Registers (CIDMR0-3)

The identifier mask registers specify which of the corresponding bits in the identifier acceptance register are relevant for acceptance filtering. For standard identifiers it is required to program the last three bits (AM2-AM0) in the mask register CIDMR1 to 'don't care'.

CIDMRO Address: $0514

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by Reset | | | | | | | |

CIDMR1 Address: $0515

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by Reset | | | | | | | |

CIDMR2 Address: $0516

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by Reset | | | | | | | |

CIDMR3 Address: $0517

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: Write: | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset: | Unaffected by Reset | | | | | | | |

**Figure 23-27. Identifier Mask Registers (CIDMR0–CIDMR3)**

**AM7–AM0 — Acceptance Mask Bits**

If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match will be detected. The message will be accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register will not affect whether or not the message is accepted.

1 = Ignore corresponding acceptance code register bit.

0 = Match corresponding acceptance code register and identifier bits.

***NOTE***

*The CIDMR0-3 registers can be written only if the SFTRES bit in the CMCR0 is set*

# Chapter 24
# Keyboard Module (KBI)

## 24.1 Introduction

The keyboard interrupt module (KBD) provides five independently maskable external interrupt pins.

This module is only available on 64-pin package options.

## 24.2 Features

KBD features include:

- Five Keyboard Interrupt Pins with Separate Keyboard Interrupt Enable Bits and One Keyboard Interrupt Mask
- Hysteresis Buffers
- Programmable Edge-Only or Edge- and Level- Interrupt Sensitivity
- Automatic Interrupt Acknowledge
- Exit from Low-Power Modes

## 24.3 Functional Description

Writing to the KBIE4–KBIE0 bits in the keyboard interrupt enable register independently enables or disables each port G or port H pin as a keyboard interrupt pin. Enabling a keyboard interrupt pin also enables its internal pullup device. A low level 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.

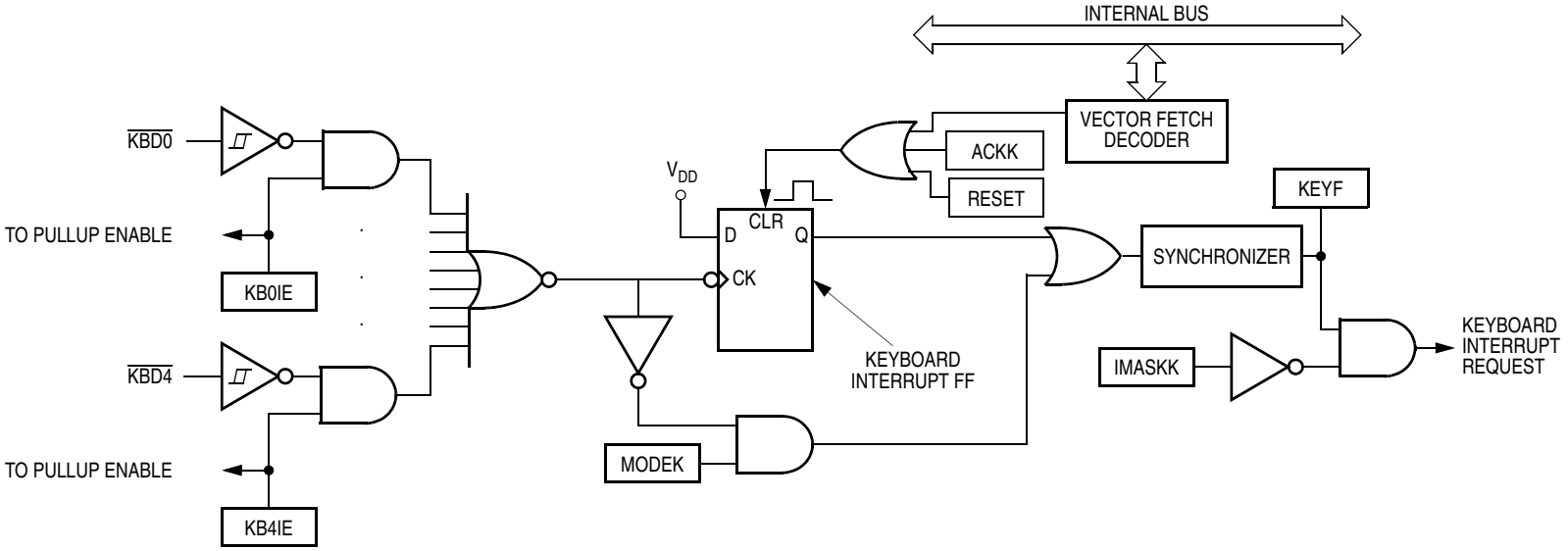



**Figure 24-1. Keyboard Module Block Diagram**

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| Keyboard Status and Control Register (KBSCR) | Read: | 0 | 0 | 0 | 0 | KEYF | 0 | IMASKK | MODEK |
| | Write: | | | | | | ACKK | | |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Keyboard Interrupt Enable Register (KBIER) | Read: | 0 | 0 | 0 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| | Write: | | | | | | | | |
| | Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 24-2. I/O Register Summary**

**Table 24-1. I/O Register Address Summary**

| Register | KBSCR | KBIER |
|---|---|---|
| Address | $001B | $0021 |

If the MODEK bit is set, the keyboard interrupt pins are both falling edge- and low level-sensitive, and both of the following actions must occur to clear a keyboard interrupt request:

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the interrupt request. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACKK bit in the keyboard status and control register (KBSCR). The ACKK bit is useful in applications that poll the keyboard interrupt pins and require software to clear the keyboard interrupt request. Writing to the ACKK bit prior to leaving an interrupt service routine also can prevent spurious interrupts due to noise. Setting ACKK does not affect subsequent transitions on the keyboard interrupt pins. A falling edge that occurs after writing to the ACKK bit latches another interrupt request. If the keyboard interrupt mask bit, IMASKK, is clear, the CPU loads the program counter with the KBI vector address.
- Return of all enabled keyboard interrupt pins to a high level. As long as any enabled keyboard interrupt pin is low, the keyboard interrupt remains set.

The vector fetch or software clear and the return of all enabled keyboard interrupt pins to a high level 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 low.

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.*

## 24.4 Keyboard Initialization

When a keyboard interrupt pin is enabled, it takes time for the internal pullup to reach a 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 DDRG bits in data direction register G.
2. Configure the keyboard pins as outputs by setting the appropriate DDRH bits in data direction register H.
3. Write logic 1s to the appropriate port G and port H data register bits.
4. Enable the KBI pins by setting the appropriate KBIEx bits in the keyboard interrupt enable register.

## 24.5 Low-Power Modes

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

### 24.5.1 Wait Mode

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

### 24.5.2 Stop Mode

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

## 24.6 Keyboard Module During Break Interrupts

The BCFE bit in the break flag control register (BFCR) enables software to clear status bits during the break state. See Chapter 13 Break Module (BRK).

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

To protect the KEYF bit during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0, writing to the keyboard acknowledge bit (ACKK) in the keyboard status and control register during the break state has no effect. See 24.7.1 Keyboard Status and Control Register.

## 24.7 I/O Registers

The following registers control and monitor operation of the keyboard module:

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

### 24.7.1 Keyboard Status and Control Register

The keyboard status and control register:

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

Address: $001B

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

= Unimplemented

**Figure 24-3. Keyboard Status and Control Register (KBSCR)**

**Bits 7–4 — Not used**

These read-only bits always read as logic 0s.

**KEYF — Keyboard Flag Bit**

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

1 = Keyboard interrupt pending
0 = No keyboard interrupt pending

**ACKK — Keyboard Acknowledge Bit**

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

**IMASKK — Keyboard Interrupt Mask Bit**

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

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

**MODEK — Keyboard Triggering Sensitivity Bit**

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

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

### 24.7.2 Keyboard Interrupt Enable Register

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

Address: $0021

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | 0 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 24-4. Keyboard Interrupt Enable Register (KBIER)**

**KBIE4–KBIE0 — Keyboard Interrupt Enable Bits**

Each of these read/write bits enables the corresponding keyboard interrupt pin to latch interrupt requests. Reset clears the keyboard interrupt enable register.

1 = PDx pin enabled as keyboard interrupt pin
0 = PDx pin not enabled as keyboard interrupt pin

# Chapter 25
# Timer Interface Module A (TIMA)

## 25.1 Introduction

This section describes the timer interface module (TIMA). The TIMA is a 6-channel timer that provides a timing reference with input capture, output compare and pulse-width-modulation functions. Figure 25-1 is a block diagram of the TIMA.

For further information regarding timers on M68HC08 family devices, please consult the HC08 Timer Reference Manual, TIM08RM/AD.

## 25.2 Features

Features of the TIMA include:

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

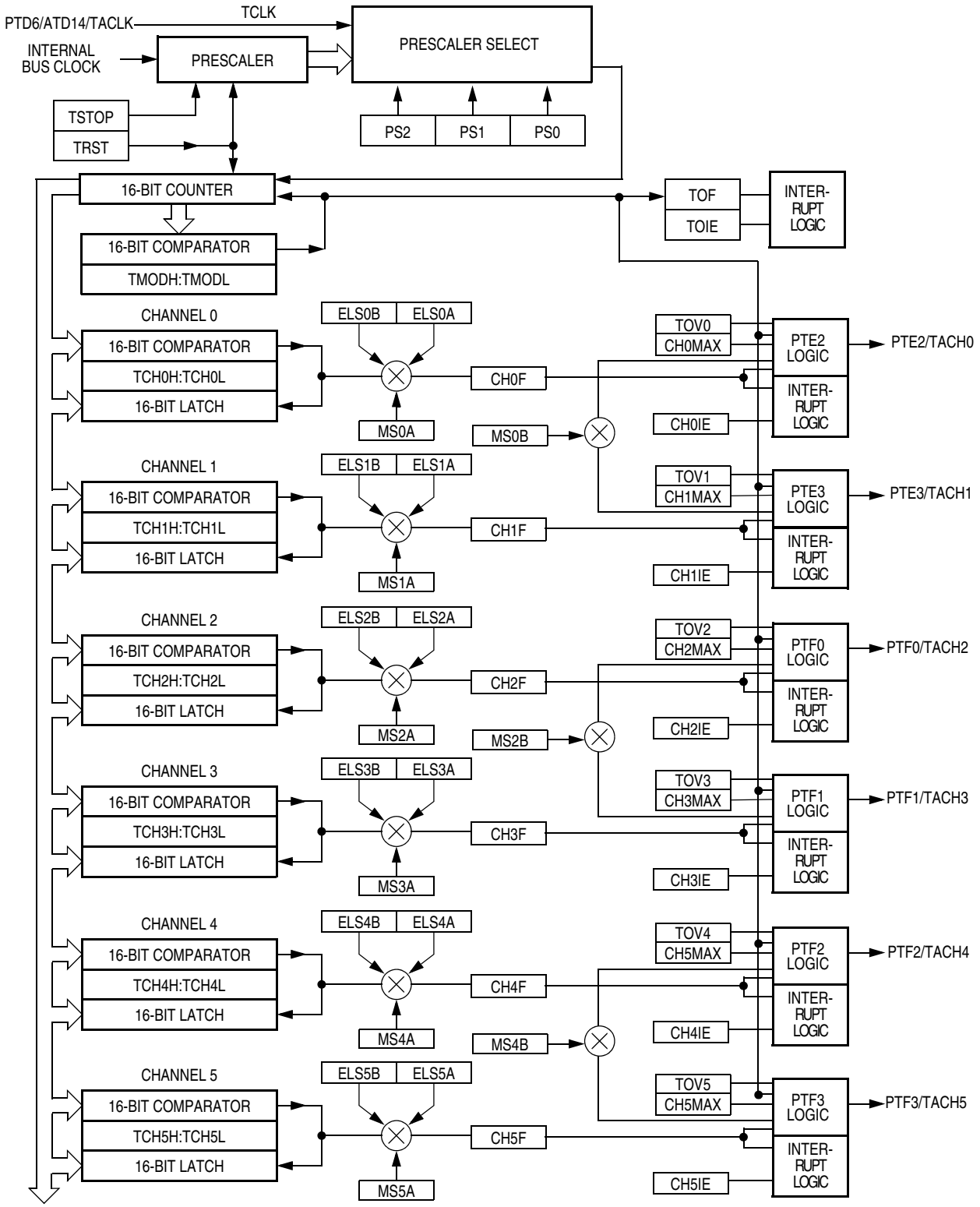



**Figure 25-1. TIMA Block Diagram**

| Addr. | Register Name | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| $0020 | TIMA Status/Control Register (TASC) | TOF | TOIE | TSTOP | TRST | 0 | PS2 | PS1 | PS0 |
| $0021 | Reserved | R | R | R | R | R | R | R | R |
| $0022 | TIMA Counter Register High (TACNTH) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0023 | TIMA Counter Register Low (TACNTL) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0024 | TIMA Counter Modulo Reg. High (TAMODH) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0025 | TIMA Counter Modulo Reg. Low (TAMODL) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0026 | TIMA Ch. 0 Status/Control Register (TASC0) | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| $0027 | TIMA Ch. 0 Register High (TACH0H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0028 | TIMA Ch. 0 Register Low (TACH0L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0029 | TIMA Ch. 1 Status/Control Register (TASC1) | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| $002A | TIMA Ch. 1 Register High (TACH1H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $002B | TIMA Ch. 1 Register Low (TACH1L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $002C | TIMA Ch. 2 Status/Control Register (TASC2) | CH2F | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| $002D | TIMA Ch. 2 Register High (TACH2H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $002E | TIMA Ch. 2 Register Low (TACH2L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $002F | TIMA Ch. 3 Status/Control Register (TASC3) | CH3F | CH3IE | 0 | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| $0030 | TIMA Ch. 3 Register High (TACH3H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0031 | TIMA Ch. 3 Register Low (TACH3L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0032 | TIMA Ch. 4 Status/Control Register (TASC4) | CH4F | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| $0033 | TIMA Ch. 4 Register High (TACH4H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0034 | TIMA Ch. 4 Register Low (TACH4L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0035 | TIMA Ch. 5 Status/Control Register (TASC5) | CH5F | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| $0036 | TIMA Ch. 5 Register High (TACH5H) | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0037 | TIMA Ch. 5 Register Low (TACH5L) | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |

R = Reserved

**Figure 25-2. TIMA I/O Register Summary**

## 25.3 Functional Description

Figure 25-1 shows the TIMA structure. The central component of the TIMA is the 16-bit TIMA counter that can operate as a free-running counter or a modulo up-counter. The TIMA counter provides the timing reference for the input capture and output compare functions. The TIMA counter modulo registers, TAMODH–TAMODL, control the modulo value of the TIMA counter. Software can read the TIMA counter value at any time without affecting the counting sequence.

The six TIMA channels are programmable independently as input capture or output compare channels.

### 25.3.1 TIMA Counter Prescaler

The TIMA clock source can be one of the seven prescaler outputs or the TIMA clock pin, PTD6/ATD14/TACLK. The prescaler generates seven clock rates from the internal bus clock. The prescaler select bits, PS[2:0], in the TIMA status and control register select the TIMA clock source.

### 25.3.2 Input Capture

An input capture function has three basic parts: edge select logic, an input capture latch and a 16-bit counter. Two 8-bit registers, which make up the 16-bit input capture register, are used to latch the value of the free-running counter after the corresponding input capture edge detector senses a defined transition. The polarity of the active edge is programmable. The level transition which triggers the counter transfer is defined by the corresponding input edge bits (ELSxB and ELSxA in TASC0 through TASC5 control registers with x referring to the active channel number). When an active edge occurs on the pin of an input capture channel, the TIMA latches the contents of the TIMA counter into the TIMA channel registers, TACHxH–TACHxL. Input captures can generate TIMA CPU interrupt requests. Software can determine that an input capture event has occurred by enabling input capture interrupts or by polling the status flag bit.

The free-running counter contents are transferred to the TIMA channel register (TACHxH–TACHxL see 25.8.5 TIMA Channel Registers) on each proper signal transition regardless of whether the TIMA channel flag (CH0F–CH5F in TASC0–TASC5 registers) is set or clear. When the status flag is set, a CPU interrupt is generated if enabled. The value of the count latched or “captured” is the time of the event. Because this value is stored in the input capture register 2 bus cycles after the actual event occurs, user software can respond to this event at a later time and determine the actual time of the event. However, this must be done prior to another input capture on the same pin; otherwise, the previous time value will be lost.

By recording the times for successive edges on an incoming signal, software can determine the period and/or pulse width of the signal. To measure a period, two successive edges of the same polarity are captured. To measure a pulse width, two alternate polarity edges are captured. Software should track the overflows at the 16-bit module counter to extend its range.

Another use for the input capture function is to establish a time reference. In this case, an input capture function is used in conjunction with an output compare function. For example, to activate an output signal a specified number of clock cycles after detecting an input event (edge), use the input capture function to record the time at which the edge occurred. A number corresponding to the desired delay is added to this captured value and stored to an output compare register (see 25.8.5 TIMA Channel Registers). Because both input captures and output compares are referenced to the same 16-bit modulo counter, the delay can be controlled to the resolution of the counter independent of software latencies.

Reset does not affect the contents of the TIMA channel register (TACHxH–TACHxL).

### 25.3.3 Output Compare

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

#### 25.3.3.1 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in 25.3.3 Output Compare. The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIMA channel registers.

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

Use 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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of the current counter overflow period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same counter overflow period.

#### 25.3.3.2 Buffered Output Compare

Channels 0 and 1 can be linked to form a buffered output compare channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The output compare value in the TIMA channel 0 registers initially controls the output on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the output are the ones written to last. TASC0 controls and monitors the buffered output compare function and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered output compare channel whose output appears on the PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The output compare value in the TIMA channel 2 registers initially controls the output on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (2 or 3) that control the output are the ones written to last. TASC2 controls and monitors the buffered output compare function, and TIMA channel 3 status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, is available as a general-purpose I/O pin.

Channels 4 and 5 can be linked to form a buffered output compare channel whose output appears on the PTF2 pin. The TIMA channel registers of the linked pair alternately control the output.

Setting the MS4B bit in TIMA channel 4 status and control register (TASC4) links channel 4 and channel 5. The output compare value in the TIMA channel 4 registers initially controls the output on the

PTF2 pin. Writing to the TIMA channel 5 registers enables the TIMA channel 5 registers to synchronously control the output after the TIMA overflows. At each subsequent overflow, the TIMA channel registers (4 or 5) that control the output are the ones written to last. TASC4 controls and monitors the buffered output compare function and TIMA channel 5 status and control register (TASC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3, 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.*

### 25.3.4 Pulse Width Modulation (PWM)

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

As Figure 25-3 shows, the output compare value in the TIMA channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIMA to clear the channel pin on output compare if the polarity of the PWM pulse is 1. Program the TIMA to set the pin if the polarity of the PWM pulse is 0.

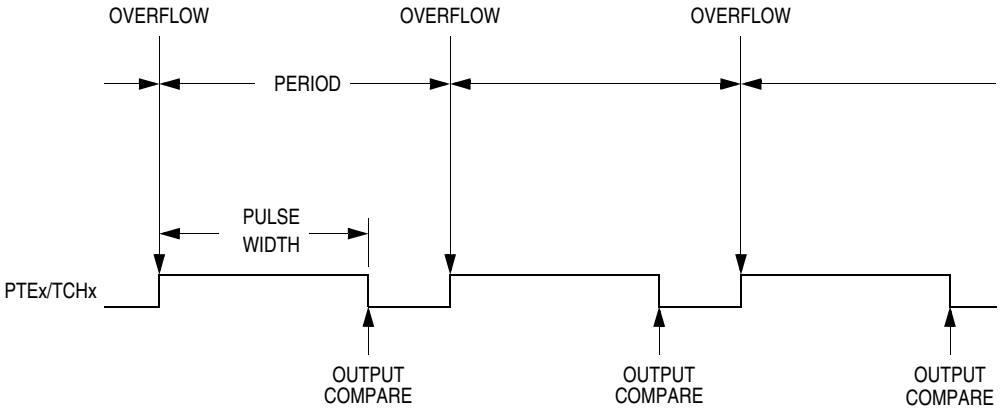



**Figure 25-3. PWM Period and Pulse Width**

The value in the TIMA counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing $00FF (255) to the TIMA counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is $000 (see 25.8.1 TIMA Status and Control Register).

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

#### 25.3.4.1 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in 25.3.4 Pulse Width Modulation (PWM). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the value currently in the TIMA channel registers.

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

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 TIMA overflow interrupts and write the new value in the TIMA overflow interrupt routine. The TIMA overflow interrupt occurs at the end of the current PWM period. Writing a larger value in an output compare interrupt routine (at the end of the current pulse) could cause two output compares to occur in the same PWM period.

**NOTE**

*In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0% 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.*

#### 25.3.4.2 Buffered PWM Signal Generation

Channels 0 and 1 can be linked to form a buffered PWM channel whose output appears on the PTE2/TACH0 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS0B bit in TIMA channel 0 status and control register (TASC0) links channel 0 and channel 1. The TIMA channel 0 registers initially control the pulse width on the PTE2/TACH0 pin. Writing to the TIMA channel 1 registers enables the TIMA channel 1 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (0 or 1) that control the pulse width are the ones written to last. TASC0 controls and monitors the buffered PWM function and TIMA channel 1 status and control register (TASC1) is unused. While the MS0B bit is set, the channel 1 pin, PTE3/TACH1, is available as a general-purpose I/O pin.

Channels 2 and 3 can be linked to form a buffered PWM channel whose output appears on the PTF0/TACH2 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS2B bit in TIMA channel 2 status and control register (TASC2) links channel 2 and channel 3. The TIMA channel 2 registers initially control the pulse width on the PTF0/TACH2 pin. Writing to the TIMA channel 3 registers enables the TIMA channel 3 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers

(2 or 3) that control the pulse width are the ones written to last. TASC2 controls and monitors the buffered PWM function and TIMA channel 3 status and control register (TASC3) is unused. While the MS2B bit is set, the channel 3 pin, PTF1/TACH3, is available as a general-purpose I/O pin.

Channels 4 and 5 can be linked to form a buffered PWM channel whose output appears on the PTF2 pin. The TIMA channel registers of the linked pair alternately control the pulse width of the output.

Setting the MS4B bit in TIMA channel 4 status and control register (TASC4) links channel 4 and channel 5. The TIMA channel 4 registers initially control the pulse width on the PTF2 pin. Writing to the TIMA channel 5 registers enables the TIMA channel 5 registers to synchronously control the pulse width at the beginning of the next PWM period. At each subsequent overflow, the TIMA channel registers (4 or 5) that control the pulse width are the ones written to last. TASC4 controls and monitors the buffered PWM function and TIMA channel 5 status and control register (TASC5) is unused. While the MS4B bit is set, the channel 5 pin, PTF3, 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.*

#### *25.3.4.3 PWM Initialization*

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

1. In the TIMA status and control register (TASC):
   a. Stop the TIMA counter by setting the TIMA stop bit, TSTOP.
   b. Reset the TIMA counter and prescaler by setting the TIMA reset bit, TRST.
2. In the TIMA counter modulo registers (TAMODH–TAMODL) write the value for the required PWM period.
3. In the TIMA channel x registers (TACHxH–TACHxL) write the value for the required pulse width.
4. In TIMA channel x status and control register (TASCx):
   a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB–MSxA (see Table 25-2).
   b. Write 1 to the toggle-on-overflow bit, TOVx.
   c. Write 1:0 (to clear output on compare) or 1:1 (to set output on compare) to the edge/level select bits, ELSxB–ELSxA. The output action on compare must force the output to the complement of the pulse width level (see Table 25-2).

   **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 TIMA status control register (TASC) clear the TIMA stop bit, TSTOP.

Setting MS0B links channels 0 and 1 and configures them for buffered PWM operation. The TIMA channel 0 registers (TACH0H–TACH0L) initially control the buffered PWM output. TIMA status control register 0 (TASC0) controls and monitors the PWM signal from the linked channels. MS0B takes priority over MS0A.

Setting MS2B links channels 2 and 3 and configures them for buffered PWM operation. The TIMA channel 2 registers (TACH2H–TACH2L) initially control the buffered PWM output. TIMA status control register 2 (TASC2) controls and monitors the PWM signal from the linked channels. MS2B takes priority over MS2A.

Setting MS4B links channels 4 and 5 and configures them for buffered PWM operation. The TIMA channel 4 registers (TACH4H–TACH4L) initially control the buffered PWM output. TIMA status control register 4 (TASC4) controls and monitors the PWM signal from the linked channels. MS4B takes priority over MS4A.

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

Setting the channel x maximum duty cycle bit (CHxMAX) and setting the TOVx bit generates a 100% duty cycle output (see 25.8.4 TIMA Channel Status and Control Registers).

## 25.4 Interrupts

The following TIMA sources can generate interrupt requests:

- TIMA overflow flag (TOF) — The TOF bit is set when the TIMA counter reaches the modulo value programmed in the TIMA counter modulo registers. The TIMA overflow interrupt enable bit, TOIE, enables TIMA overflow CPU interrupt requests. TOF and TOIE are in the TIMA status and control register.
- TIMA channel flags (CH5F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIMA CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE.

## 25.5 Low-Power Modes

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

### 25.5.1 Wait Mode

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

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

### 25.5.2 Stop Mode

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

## 25.6 TIMA During Break Interrupts

A break interrupt stops the TIMA 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 SIM break flag control register (SBFCR) enables software to clear status bits during the break state (see 9.7.3 SIM Break Flag Control Register).

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

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

## 25.7 I/O Signals

Port D shares one of its pins with the TIMA. Port E shares two of its pins with the TIMA and port F shares four of its pins with the TIMA. PTD6/ATD14/TACLK is an external clock input to the TIMA prescaler. The six TIMA channel I/O pins are PTE2/TACH0, PTE3/TACH1, PTF0/TACH2, PTF1/TACH3, PTF2, and PTF3.

### 25.7.1 TIMA Clock Pin (PTD6/ATD14/TACLK)

PTD6/ATD14/TACLK is an external clock input that can be the clock source for the TIMA counter instead of the prescaled internal bus clock. Select the PTD6/ATD14/TACLK input by writing logic 1s to the three prescaler select bits, PS[2:0] (see 25.8.1 TIMA Status and Control Register). The minimum TCLK pulse width, $TCLK_{LMIN}$ or $TCLK_{HMIN}$, is:

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

The maximum TCLK frequency is the least: 4 MHz or bus frequency ÷ 2.

PTD6/ATD14/TACLK is available as a general-purpose I/O pin or ADC channel when not used as the TIMA clock input. When the PTD6/ATD14/TACLK pin is the TIMA clock input, it is an input regardless of the state of the DDRD6 bit in data direction register D.

### 25.7.2 TIMA Channel I/O Pins (PTF3–PTF0/TACH2 and PTE3/TACH1–PTE2/TACH0)

Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. PTE2/TACH0, PTF0/TACH2 and PTF2 can be configured as buffered output compare or buffered PWM pins.

## 25.8 I/O Registers

These I/O registers control and monitor TIMA operation:

- TIMA status and control register (TASC)
- TIMA control registers (TACNTH–TACNTL)
- TIMA counter modulo registers (TAMODH–TAMODL)
- TIMA channel status and control registers (TASC0, TASC1, TASC2, TASC3, TASC4 and TASC5)
- TIMA channel registers (TACH0H–TACH0L, TACH1H–TACH1L, TACH2H–TACH2L, TACH3H–TACH3L, TACH4H–TACH4L and TACH5H–TACH5L)

### 25.8.1 TIMA Status and Control Register

The TIMA status and control register:

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

Address: $0020

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

R = Reserved

**Figure 25-4. TIMA Status and Control Register (TASC)**

**TOF — TIMA Overflow Flag Bit**

This read/write flag is set when the TIMA counter reaches the modulo value programmed in the TIMA counter modulo registers. Clear TOF by reading the TIMA status and control register when TOF is set and then writing a logic 0 to TOF. If another TIMA overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIMA counter has reached modulo value.
0 = TIMA counter has not reached modulo value.

**TOIE — TIMA Overflow Interrupt Enable Bit**

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

1 = TIMA overflow interrupts enabled
0 = TIMA overflow interrupts disabled

**TSTOP — TIMA Stop Bit**

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

1 = TIMA counter stopped
0 = TIMA counter active

***NOTE***

*Do not set the TSTOP bit before entering wait mode if the TIMA is required to exit wait mode. Also, when the TSTOP bit is set and input capture mode is enabled, input captures are inhibited until TSTOP is cleared.*

*When using the TSTOP to stop the timer counter, see if any timer flags are set. If a timer flag is set, it must be cleared by clearing the TSTOP, then clearing the flag, then setting the TSTOP again.*

**TRST — TIMA Reset Bit**

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

1 = Prescaler and TIMA counter cleared
0 = No effect

***NOTE***

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

**PS[2:0] — Prescaler Select Bits**

These read/write bits select either the PTD6/ATD14/TACLK pin or one of the seven prescaler outputs as the input to the TIMA counter as Table 25-1 shows. Reset clears the PS[2:0] bits.

**Table 25-1. Prescaler Selection**

| PS[2:0] | TIMA Clock Source |
|---|---|
| 000 | Internal Bus Clock ÷1 |
| 001 | Internal Bus Clock ÷ 2 |
| 010 | Internal Bus Clock ÷ 4 |
| 011 | Internal Bus Clock ÷ 8 |
| 100 | Internal Bus Clock ÷ 16 |
| 101 | Internal Bus Clock ÷ 32 |
| 110 | Internal Bus Clock ÷ 64 |
| 111 | PTD6/ATD14/TACLK |

## 25.8.2 TIMA Counter Registers

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

***NOTE***

*If TACNTH is read during a break interrupt, be sure to unlatch TACNTL by reading TACNTL before exiting the break interrupt. Otherwise, TACNTL retains the value latched during the break.*

Register Name and Address TACNTH — $0022

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

Register Name and Address TACNTL — $0023

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

= Unimplemented

**Figure 25-5. TIMA Counter Registers (TACNTH and TACNTL)**

### 25.8.3 TIMA Counter Modulo Registers

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

Register Name and Address TAMODH — $0024

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

Register Name and Address TAMODL — $0025

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

**Figure 25-6. TIMA Counter Modulo Registers (TAMODH and TAMODL)**

***NOTE***

*Reset the TIMA counter before writing to the TIMA counter modulo registers.*

### 25.8.4 TIMA Channel Status and Control Registers

Each of the TIMA channel status and control registers:

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

Register Name and Address: TASC0 — $0026

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

Register Name and Address: TASC1 — $0029

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

R = Reserved

Register Name and Address: TASC2 — $002C

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

Register Name and Address: TASC3 — $002F

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

Register Name and Address: TASC4 — $0032

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CH4F | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| Write: | 0 | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 25-7. TIMA Channel Status and Control Registers (TASC0–TASC5)**

Register Name and Address TASC5 — $0035

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | CH5F | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| Write: | 0 | | R | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 25-7. TIMA Channel Status and Control Registers (TASC0–TASC5) (Continued)**

### CHxF — Channel x Flag Bit

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

When CHxIE = 1, clear CHxF by reading TIMA channel x status and control register with CHxF set and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

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

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

### CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIMA CPU interrupts on channel x.
Reset clears the CHxIE bit.

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

### MSxB — Mode Select Bit B

This read/write bit selects buffered output compare/PWM operation. MSxB exists only in the TIMA channel 0, TIMA channel 2 and TIMA channel 4 status and control registers.

Setting MS0B disables the channel 1 status and control register and reverts TACH1 pin to general-purpose I/O.

Setting MS2B disables the channel 3 status and control register and reverts TACH3 pin to general-purpose I/O.

Setting MS4B disables the channel 5 status and control register and reverts TACH5 pin to general-purpose I/O.

Reset clears the MSxB bit.

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

### MSxA — Mode Select Bit A

When ELSxB:A ≠ 00, this read/write bit selects either input capture operation or unbuffered output compare/PWM operation. See Table 25-2.

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

When ELSxB:A = 00, this read/write bit selects the initial output level of the TACHx pin once PWM, output compare mode or input capture mode is enabled. See Table 25-2. Reset clears the MSxA bit.

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

***NOTE***

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

### ELSxB and ELSxA — Edge/Level Select Bits

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

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

When ELSxB and ELSxA are both clear, channel x is not connected to port E or port F and pin PTEx/TACHx or pin PTFx/TACHx is available as a general-purpose I/O pin. However, channel x is at a state determined by these bits and becomes transparent to the respective pin when PWM, input capture mode or output compare operation mode is enabled. Table 25-2 shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**Table 25-2. Mode, Edge, and Level Selection**

| MSxB | MSxA | ELSxB | ELSxA | Mode | Configuration |
|---|---|---|---|---|---|
| X | 0 | 0 | 0 | Output preset | Pin under port control; initial output level high |
| X | 1 | 0 | 0 | | Pin under port control; initial output level low |
| 0 | 0 | 0 | 1 | Input capture | Capture on rising edge only |
| 0 | 0 | 1 | 0 | | Capture on falling edge only |
| 0 | 0 | 1 | 1 | | Capture on rising or falling edge |
| 0 | 1 | 0 | 0 | Output compare or PWM | Software compare only |
| 0 | 1 | 0 | 1 | | Toggle output on compare |
| 0 | 1 | 1 | 0 | | Clear output on compare |
| 0 | 1 | 1 | 1 | | Set output on compare |
| 1 | X | 0 | 1 | Buffered output compare or buffered PWM | Toggle output on compare |
| 1 | X | 1 | 0 | | Clear output on compare |
| 1 | X | 1 | 1 | | Set output on compare |

***NOTE***

*Before enabling a TIMA channel register for input capture operation, make sure that the PTEx/TACHx pin or PTFx/TACHx pin is stable for at least two bus clocks.*

**TOVx — Toggle-On-Overflow Bit**

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

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

***NOTE***

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

**CHxMAX — Channel x Maximum Duty Cycle Bit**

When the TOVx bit is at logic 1, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As Figure 25-8 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.

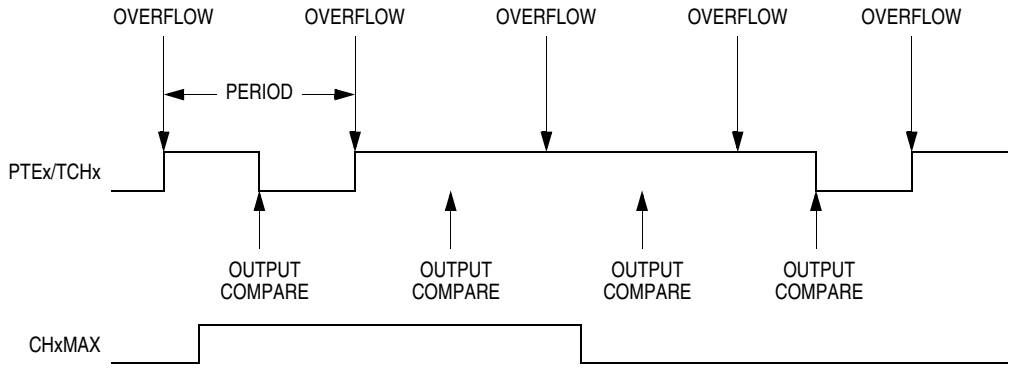



**Figure 25-8. CHxMAX Latency**

## 25.8.5 TIMA Channel Registers

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

In input capture mode (MSxB–MSxA = 0:0) reading the high byte of the TIMA channel x registers (TACHxH) inhibits input captures until the low byte (TACHxL) is read.

In output compare mode (MSxB–MSxA ≠ 0:0) writing to the high byte of the TIMA channel x registers (TACHxH) inhibits output compares and the CHxF bit until the low byte (TACHxL) is written.

Register Name and Address: TACH0H — $0027

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

**Figure 25-9. TIMA Channel Registers (TACH0H/L–TACH5H/L) (Sheet 1 of 3)**

Register Name and Address TACH0L — $0028

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

Register Name and Address TACH1H — $002A

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

Register Name and Address TACH1L — $002B

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

Register Name and Address TACH2H — $002D

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

Register Name and Address TACH2L — $002E

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

Register Name and Address TACH3H — $0030

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

Register Name and Address TACH3L — $0031

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

**Figure 25-9. TIMA Channel Registers (TACH0H/L–TACH5H/L) (Sheet 2 of 3)**

Register Name and Address TACH4H — $0033

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

Register Name and Address TACH4L — $0034

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

Register Name and Address TACH5H — $0036

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

Register Name and Address TACH5L — $0037

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

**Figure 25-9. TIMA Channel Registers (TACH0H/L–TACH5H/L) (Sheet 3 of 3)**

# Chapter 26
# Analog-to-Digital Converter (ADC)

## 26.1 Introduction

This section describes the analog-to-digital converter (ADC-15). The ADC is an 8-bit analog-to-digital converter.

For further information regarding analog-to-digital converters on Freescale microcontrollers, please consult the HC08 ADC Reference Manual, ADCRM/AD.

## 26.2 Features

Features of the ADC module include:

- 15 Channels with Multiplexed Input
- Linear Successive Approximation
- 8-Bit Resolution
- Single or Continuous Conversion
- Conversion Complete Flag or Conversion Complete Interrupt
- Selectable ADC Clock

## 26.3 Functional Description

Fifteen ADC channels are available for sampling external sources at pins PTD6/ATD14/TACLK–PTD0/ATD8 and PTB7/ATD7–PTB0/ATD0. An analog multiplexer allows the single ADC converter to select one of 15 ADC channels as ADC voltage in (ADCVIN). ADCVIN is converted by the successive approximation register-based counters. When the conversion is completed, ADC places the result in the ADC data register and sets a flag or generates an interrupt. See Figure 26-1.

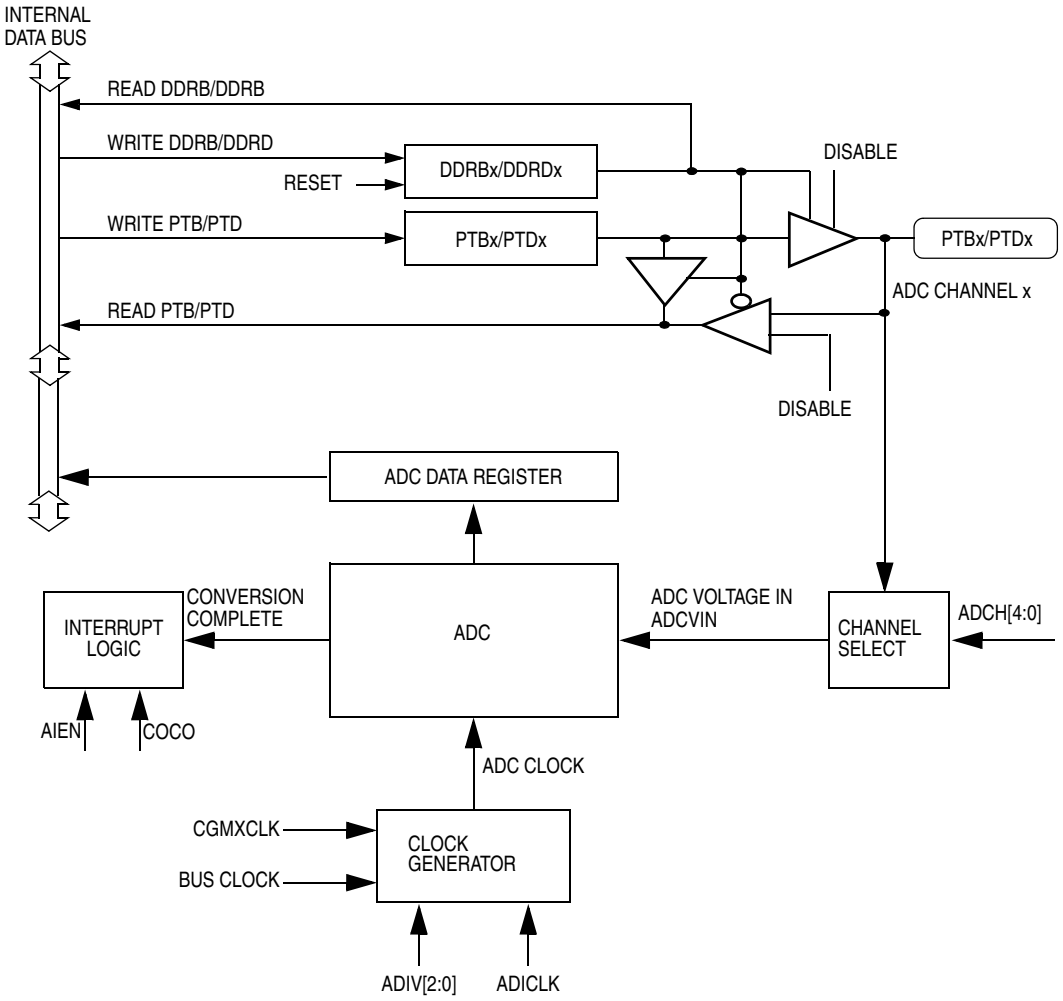



**Figure 26-1. ADC Block Diagram**

### 26.3.1 ADC Port I/O Pins

PTD6/ATD14/TACLK–PTD0/ATD8 and PTB7/ATD7–PTB0/ATD0 are general-purpose I/O pins that share with the ADC channels.

The channel select bits 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 register or DDR will not have any affect on the port pin that is selected by the ADC. Read of a port pin which is in use by the ADC will return a 0 if the corresponding DDR bit is at logic 0. If the DDR bit is at logic 1, the value in the port data latch is read.

***NOTE***

*Do not use ADC channels ATD14 or ATD12 when using the PTD6/ATD14/TACLK or PTD4/ATD12/TBCLK pins as the clock inputs for the 16-bit Timers.*

### 26.3.2 Voltage Conversion

When the input voltage to the ADC equals $V_{REFH}$ (see 28.1.6 ADC Characteristics), the ADC converts the signal to $FF (full scale). If the input voltage equals $V_{SSA,}$ the ADC converts it to $00. Input voltages between $V_{REFH}$ and $V_{SSA}$ are a straight-line linear conversion. Conversion accuracy of all other input voltages is not guaranteed. Avoid current injection on unused ADC inputs to prevent potential conversion error.

***NOTE***

*Input voltage should not exceed the analog supply voltages.*

### 26.3.3 Conversion Time

Conversion starts after a write to the ADSCR (ADC status control register, $0038), and requires between 16 and 17 ADC clock cycles to complete. Conversion time in terms of the number of bus cycles is a function of ADICLK select, CGMXCLK frequency, bus frequency, and ADIV prescaler bits. For example, with a CGMXCLK frequency of 4 MHz, bus frequency of 8 MHz, and fixed ADC clock frequency of 1 MHz, one conversion will take between 16 and 17 μs and there will be between 128 bus cycles between each conversion. Sample rate is approximately 60 kHz.

Refer to 28.1.6 ADC Characteristics.

$$\text{Conversion Time} = \frac{\text{16 to 17 ADC Clock Cycles}}{\text{ADC Clock Frequency}}$$

$$\text{Number of Bus Cycles} = \text{Conversion Time} \times \text{Bus Frequency}$$

### 26.3.4 Continuous Conversion

In the continuous conversion mode, the ADC data register will be filled 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 (ADC status control register, $0038) is cleared. The COCO bit is set after the first conversion and will stay set for the next several conversions until the next write of the ADC status and control register or the next read of the ADC data register.

### 26.3.5 Accuracy and Precision

The conversion process is monotonic and has no missing codes. See 28.1.6 ADC Characteristics for accuracy information.

## 26.4 Interrupts

When the AIEN bit is set, the ADC module is capable of generating a CPU interrupt after each ADC conversion. A CPU interrupt is generated if the COCO bit (ADC status control register, $0038) is at logic 0. The COCO bit is not used as a conversion complete flag when interrupts are enabled.

## 26.5 Low-Power Modes

The following subsections describe the low-power modes.

### 26.5.1 Wait Mode

The ADC continues normal operation during wait mode. Any enabled CPU interrupt request from the ADC can bring the MCU out of wait mode. If the ADC is not required to bring the MCU out of wait mode, power down the ADC by setting the ADCH[4:0] bits in the ADC status and control register before executing the WAIT instruction.

### 26.5.2 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.

## 26.6 I/O Signals

The ADC module has 15 channels that are shared with I/O ports B and D. Refer to 28.1.6 ADC Characteristics for voltages referenced below.

### 26.6.1 ADC Analog Power Pin ($V_{DDAREF}$)/ADC Voltage Reference Pin ($V_{REFH}$)

The ADC analog portion uses $V_{DDAREF}$ as its power pin. Connect the $V_{DDA}/V_{DDAREF}$ pin to the same voltage potential as $V_{DD}$. External filtering may be necessary to ensure clean $V_{DDAREF}$ for good results.

$V_{REFH}$ is the high reference voltage for all analog-to-digital conversions.

***NOTE***

*Route $V_{DDAREF}$ carefully for maximum noise immunity and place bypass capacitors as close as possible to the package. $V_{DDAREF}$ must be present for operation of the ADC.*

### 26.6.2 ADC Analog Ground Pin ($V_{SSA}$)/ADC Voltage Reference Low Pin ($V_{REFL}$)

The ADC analog portion uses $V_{SSA}$ as its ground pin. Connect the $V_{SSA}$ pin to the same voltage potential as $V_{SS}$.

$V_{REFL}$ is the lower reference supply for the ADC.

### 26.6.3 ADC Voltage In (ADCVIN)

ADCVIN is the input voltage signal from one of the 15 ADC channels to the ADC module.

## 26.7 I/O Registers

These I/O registers control and monitor ADC operation:

- ADC status and control register (ADSCR)
- ADC data register (ADR)
- ADC clock register (ADICLK)

### 26.7.1 ADC Status and Control Register

The following paragraphs describe the function of the ADC status and control register.

Address: $0038

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | COCO | AIEN | ADCO | CH4 | CH3 | CH2 | CH1 | CH0 |
| Write: | R | | | | | | | |
| Reset: | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

R = Reserved

**Figure 26-2. ADC Status and Control Register (ADSCR)**

**COCO — Conversions Complete Bit**

When the AIEN bit is a logic 0, the COCO is a read-only bit which is set each time a conversion is completed. This bit is cleared whenever the ADC status and control register is written or whenever the ADC data register is read.

If the AIEN bit is a logic 1, the COCO is a read/write bit which selects the CPU to service the ADC interrupt request. Reset clears this bit.

1 = conversion completed (AIEN = 0)

0 = conversion not completed (AIEN = 0)

or

CPU interrupt enabled (AIEN = 1)

**AIEN — ADC Interrupt Enable Bit**

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

1 = ADC interrupt enabled

0 = ADC interrupt disabled

**ADCO — ADC Continuous Conversion Bit**

When set, the ADC will convert samples continuously and update the ADR register at the end of each conversion. Only one conversion is allowed when this bit is cleared. Reset clears the ADCO bit.

1 = Continuous ADC conversion

0 = One ADC conversion

**ADCH[4:0] — ADC Channel Select Bits**

ADCH4, ADCH3, ADCH2, ADCH1, and ADCH0 form a 5-bit field which is used to select one of 15 ADC channels. Channel selection is detailed in the following table. 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. See Table 26-1.

The ADC subsystem is turned off when the channel select bits are all set to one. This feature allows for reduced power consumption for the MCU when the ADC is not used. Reset sets these bits.

***NOTE***

*Recovery from the disabled state requires one conversion cycle to stabilize.*

**Table 26-1. Mux Channel Select**

| ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 | Input Select |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | PTB0/ATD0 |
| 0 | 0 | 0 | 0 | 1 | PTB1/ATD1 |
| 0 | 0 | 0 | 1 | 0 | PTB2/ATD2 |
| 0 | 0 | 0 | 1 | 1 | PTB3/ATD3 |
| 0 | 0 | 1 | 0 | 0 | PTB4/ATD4 |
| 0 | 0 | 1 | 0 | 1 | PTB5/ATD5 |
| 0 | 0 | 1 | 1 | 0 | PTB6/ATD6 |
| 0 | 0 | 1 | 1 | 1 | PTB7/ATD7 |
| 0 | 1 | 0 | 0 | 0 | PTD0/ATD8/ATD8 |
| 0 | 1 | 0 | 0 | 1 | PTD1/ATD9/ATD9 |
| 0 | 1 | 0 | 1 | 0 | PTD2/ATD10/ATD10 |
| 0 | 1 | 0 | 1 | 1 | PTD3/ATD11/ATD11 |
| 0 | 1 | 1 | 0 | 0 | PTD4/ATD12/TBCLK/ATD12 |
| 0 | 1 | 1 | 0 | 1 | PTD5/ATD13/ATD13 |
| 0 | 1 | 1 | 1 | 0 | PTD6/ATD14/TACLK/ATD14 |
| Range 01111 ($0F) to 11010 ($1A) | | | | | Unused (see Note 1) |
| | | | | | Unused (see Note 1) |
| 1 | 1 | 0 | 1 | 1 | Reserved |
| 1 | 1 | 1 | 0 | 0 | Unused (see Note 1) |
| 1 | 1 | 1 | 0 | 1 | $V_{REFH}$ (see Note 2) |
| 1 | 1 | 1 | 1 | 0 | $V_{SSA}/V_{REFL}$ (see Note 2) |
| 1 | 1 | 1 | 1 | 1 | [ADC power off] |

Notes:
1. If any unused channels are selected, the resulting ADC conversion will be unknown.
2. The voltage levels supplied from internal reference nodes as specified in the table are used to verify the operation of the ADC converter both in production test and for user applications.

## 26.7.2 ADC Data Register

One 8-bit result register is provided. This register is updated each time an ADC conversion completes.

Address: $0039

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

= Unimplemented

**Figure 26-3. ADC Data Register (ADR)**

### 26.7.3 ADC Input Clock Register

This register selects the clock frequency for the ADC.

Address: $003A

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | ADIV2 | ADIV1 | ADIV0 | ADICLK | 0 | 0 | 0 | 0 |
| Write: | | | | | | | | |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented

**Figure 26-4. ADC Input Clock Register (ADICLK)**

**ADIV2–ADIV0 — ADC Clock Prescaler Bits**

ADIV2, ADIV1, and ADIV0 form a 3-bit field which selects the divide ratio used by the ADC to generate the internal ADC clock. Table 26-2 shows the available clock configurations. The ADC clock should be set to approximately 1 MHz.

**Table 26-2. ADC Clock Divide Ratio**

| ADIV2 | ADIV1 | ADIV0 | ADC Clock Rate |
|---|---|---|---|
| 0 | 0 | 0 | ADC Input Clock /1 |
| 0 | 0 | 1 | ADC Input Clock / 2 |
| 0 | 1 | 0 | ADC Input Clock / 4 |
| 0 | 1 | 1 | ADC Input Clock / 8 |
| 1 | X | X | ADC Input Clock / 16 |

X = don’t care

**ADICLK — ADC Input Clock Register Bit**

ADICLK selects either bus clock or CGMXCLK as the input clock source to generate the internal ADC clock. Reset selects CGMXCLK as the ADC clock source.

If the external clock (CGMXCLK) is equal to or greater than 1 MHz, CGMXCLK can be used as the clock source for the ADC. If CGMXCLK is less than 1 MHz, use the PLL-generated bus clock as the clock source. As long as the internal ADC clock is at approximately 1 MHz, correct operation can be guaranteed. See 28.1.6 ADC Characteristics.

1 = Internal bus clock

0 = External clock (CGMXCLK)

$$1\ \text{MHz} = \frac{f_{XCLK}\ \text{or Bus Frequency}}{\text{ADIV[2:0]}}$$

***NOTE***

*During the conversion process, changing the ADC clock will result in an incorrect conversion.*

# Chapter 27
# Byte Data Link Controller (BDLC)

## 27.1 Introduction

The byte data link controller (BDLC) provides access to an external serial communication multiplex bus, operating according to the Society of Automotive Engineers (SAE) J1850 protocol.

The BDLC-D is only available on the MC68HC908AS60A.

## 27.2 Features

Features of the BDLC module include:

- SAE J1850 class B data communications network interface compatible and ISO compatible for low speed (≤125 kbps) serial data communications in automotive applications
- 10.4 kbps variable pulse width (VPW) bit format
- Digital noise filter
- Collision detection
- Hardware cyclical redundancy check (CRC) generation and checking
- Two power-saving modes with automatic wakeup on network activity
- Polling and CPU interrupts available
- Block mode receive and transmit supported
- Supports 4X receive mode, 41.6 kbps
- Digital loopback mode
- Analog loopback mode
- In-frame response (IFR) types 0, 1, 2, and 3 supported

## 27.3 Functional Description

Figure 27-1 shows the organization of the BDLC module. The CPU interface contains the software addressable registers and provides the link between the CPU and the buffers. The buffers provide storage for data received and data to be transmitted onto the J1850 bus. The protocol handler is responsible for the encoding and decoding of data bits and special message symbols during transmission and reception. The MUX interface provides the link between the BDLC digital section and the analog physical interface. The wave shaping, driving, and digitizing of data is performed by the physical interface.

Use of the BDLC module in message networking fully implements the *SAE Standard J1850 Class B Data Communication Network Interface* specification.

***NOTE***

*It is recommended that the reader be familiar with the SAE J1850 document and ISO Serial Communication document prior to proceeding with this chapter of the specification.*

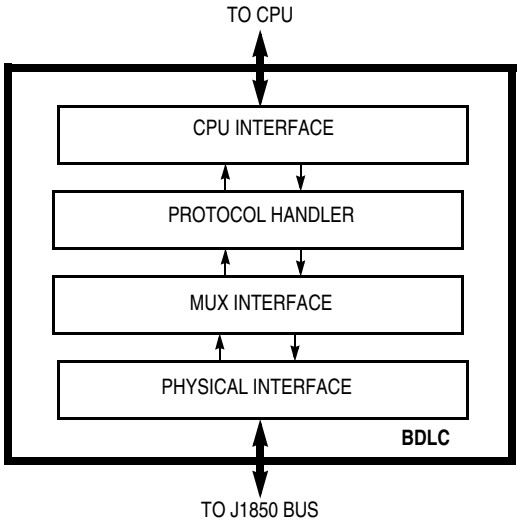



**Figure 27-1. BDLC Block Diagram**

| Addr. | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $003B | BDLC Analog and Rou5ndtrip Delay Register (BARD) | Read: | ATE | RXPOL | 0 | 0 | BO3 | BO2 | BO1 | BO0 |
| | | Write: | | | R | R | | | | |
| $003C | BDLC Control Register 1 (BCR1) | Read: | IMSG | CLKS | R1 | R0 | 0 | 0 | IE | WCM |
| | | Write: | | | | | R | R | | |
| $003D | BDLC Control Register 2 (BCR2) | Read: | ALOOP | DLOOP | RX4XE | NBFS | TEOD | TSIFR | TMIFR1 | TMIFR0 |
| | | Write: | | | | | | | | |
| $003E | BDLC State Vector Register (BSVR) | Read: | 0 | 0 | I3 | I2 | I1 | I0 | 0 | 0 |
| | | Write: | R | R | R | R | R | R | R | R |
| $003F | BDLC Data Register (BDR) | Read: | BD7 | BD6 | BD5 | BD4 | BD3 | BD2 | BD1 | BD0 |
| | | Write: | | | | | | | | |

R = Reserved

**Figure 27-2. BDLC I/O Register Summary**

### 27.3.1 BDLC Operating Modes

The BDLC has five main modes of operation which interact with the power supplies, pins, and the remainder of the MCU as shown in Figure 27-3.

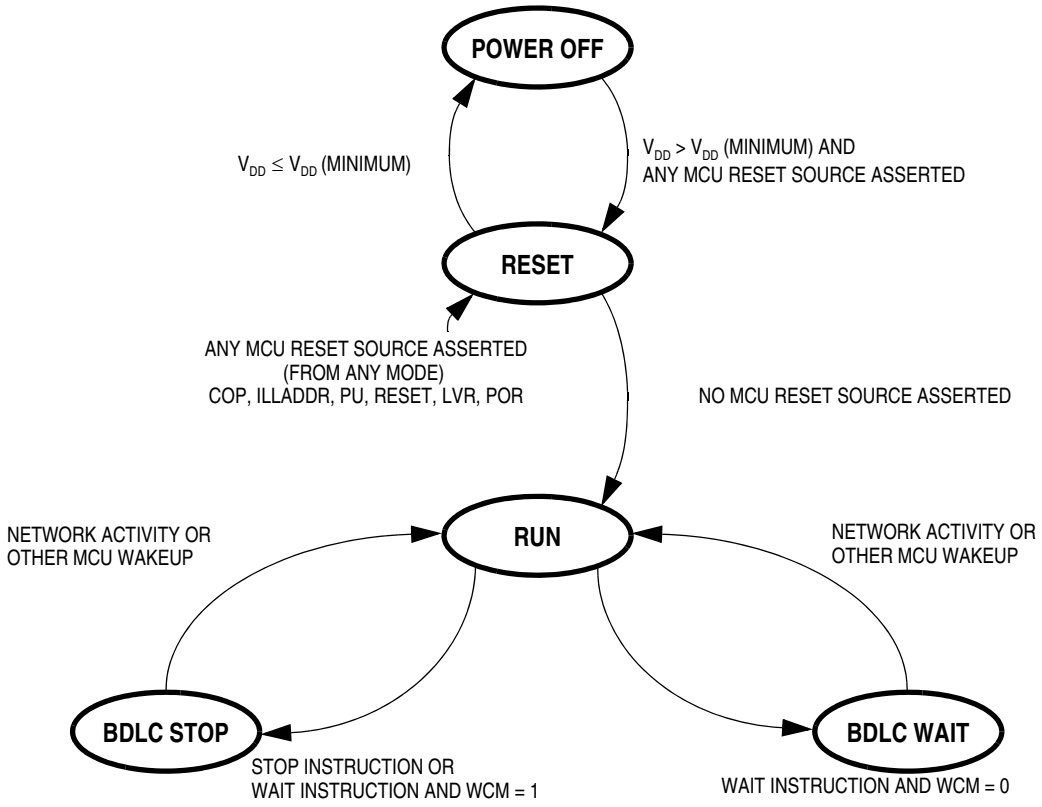



**Figure 27-3. BDLC Operating Modes State Diagram**

#### *27.3.1.1 Power Off Mode*

This mode is entered from reset mode whenever the BDLC supply voltage, $V_{DD}$, drops below its minimum specified value for the BDLC to guarantee operation. The BDLC will be placed in reset mode by low-voltage reset (LVR) before being powered down. In this mode, the pin input and output specifications are not guaranteed.

#### *27.3.1.2 Reset Mode*

This mode is entered from the power off mode whenever the BDLC supply voltage, $V_{DD}$, rises above its minimum specified value ($V_{DD}$ –10%) and some MCU reset source is asserted. The internal MCU reset must be asserted while powering up the BDLC or an unknown state will be entered and correct operation cannot be guaranteed. Reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (such as LVR, POR, COP watchdog, and reset pin, etc.) is asserted.

In reset mode, the internal BDLC voltage references are operative; $V_{DD}$ is supplied to the internal circuits which are held in their reset state; and the internal BDLC system clock is running. Registers will assume their reset condition. Outputs are held in their programmed reset state. Therefore, inputs and network activity are ignored.

#### *27.3.1.3 Run Mode*

This mode is entered from the reset mode after all MCU reset sources are no longer asserted. Run mode is entered from the BDLC wait mode whenever activity is sensed on the J1850 bus.

Run mode is entered from the BDLC stop mode whenever network activity is sensed, although messages will not be received properly until the clocks have stabilized and the CPU is in run mode also.

In this mode, normal network operation takes place. The user should ensure that all BDLC transmissions have ceased before exiting this mode.

#### *27.3.1.4 BDLC Wait Mode*

This power-conserving mode is entered automatically from run mode whenever the CPU executes a WAIT instruction and if the WCM bit in the BCR1 register is cleared previously.

In this mode, the BDLC internal clocks continue to run. The first passive-to-active transition of the bus generates a CPU interrupt request from the BDLC which wakes up the BDLC and the CPU. In addition, if the BDLC receives a valid EOF symbol while operating in wait mode, then the BDLC also will generate a CPU interrupt request which wakes up the BDLC and the CPU. See 27.7.1 Wait Mode.

#### *27.3.1.5 BDLC Stop Mode*

This power-conserving mode is entered automatically from run mode whenever the CPU executes a STOP instruction or if the CPU executes a WAIT instruction and the WCM bit in the BCR1 register is set previously.

In this mode, the BDLC internal clocks are stopped but the physical interface circuitry is placed in a low-power mode and awaits network activity. If network activity is sensed, then a CPU interrupt request will be generated, restarting the BDLC internal clocks. See 27.7.2 Stop Mode.

#### *27.3.1.6 Digital Loopback Mode*

When a bus fault has been detected, the digital loopback mode is used to determine if the fault condition is caused by failure in the node's internal circuits or elsewhere in the network, including the node's analog physical interface. In this mode, the transmit digital output pin (BDTxD) and the receive digital input pin (BDRxD) of the digital interface are disconnected from the analog physical interface and tied together to allow the digital portion of the BDLC to transmit and receive its own messages without driving the J1850 bus.

#### *27.3.1.7 Analog Loopback Mode*

Analog loopback is used to determine if a bus fault has been caused by a failure in the node's off-chip analog transceiver or elsewhere in the network. The BCLD analog loopback mode does not modify the digital transmit or receive functions of the BDLC. It does, however, ensure that once analog loopback mode is exited, the BDLC will wait for an idle bus condition before participation in network communication resumes. If the off-chip analog transceiver has a loopback mode, it usually causes the input to the output drive stage to be looped back into the receiver, allowing the node to receive messages it has transmitted without driving the J1850 bus. In this mode, the output to the J1850 bus is typically high impedance. This allows the communication path through the analog transceiver to be tested without interfering with network activity. Using the BDLC analog loopback mode in conjunction with the analog transceiver's loopback mode ensures that, once the off-chip analog transceiver has exited loopback mode, the BCLD will not begin communicating before a known condition exists on the J1850 bus.

## 27.4 BDLC MUX Interface

The MUX interface is responsible for bit encoding/decoding and digital noise filtering between the protocol handler and the physical interface.

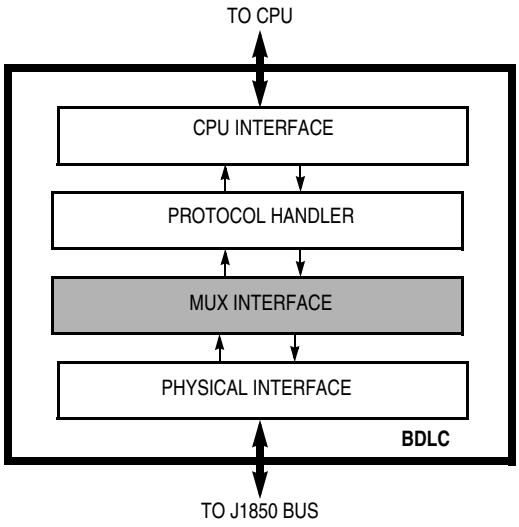



**Figure 27-4. BDLC Block Diagram**

### 27.4.1 Rx Digital Filter

The receiver section of the BDLC includes a digital low pass filter to remove narrow noise pulses from the incoming message. An outline of the digital filter is shown in Figure 27-5.

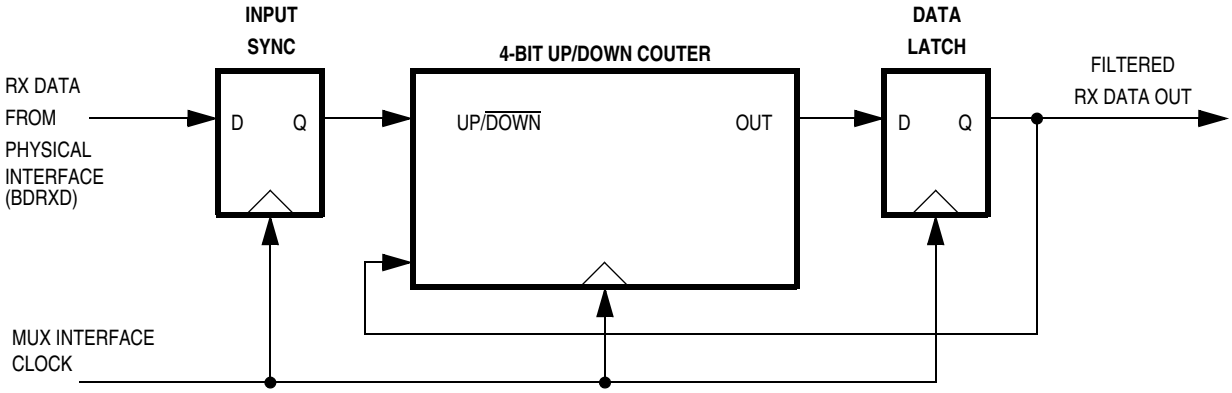



**Figure 27-5. BDLC Rx Digital Filter Block Diagram**

#### *27.4.1.1 Operation*

The clock for the digital filter is provided by the MUX interface clock (see $f_{BDLC}$ parameter in Table 27-3). At each positive edge of the clock signal, the current state of the receiver physical interface (BDRxD) signal is sampled. The BDRxD signal state is used to determine whether the counter should increment or decrement at the next negative edge of the clock signal.

The counter will increment if the input data sample is high but decrement if the input sample is low. Therefore, the counter will thus progress either up toward 15 if, on average, the BDRxD signal remains high or progress down toward 0 if, on average, the BDRxD signal remains low.

When the counter eventually reaches the value 15, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 1 and the data latch is set, causing the filtered Rx data signal to become a logic level 1. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

Alternatively, should the counter eventually reach the value 0, the digital filter decides that the condition of the BDRxD signal is at a stable logic level 0 and the data latch is reset, causing the filtered Rx data signal to become a logic level 0. Furthermore, the counter is prevented from underflowing and can only be incremented from this state.

The data latch will retain its value until the counter next reaches the opposite end point, signifying a definite transition of the signal.

#### *27.4.1.2 Performance*

The performance of the digital filter is best described in the time domain rather than the frequency domain.

If the signal on the BDRxD signal transitions, then there will be a delay before that transition appears at the filtered Rx data output signal. This delay will be between 15 and 16 clock periods, depending on where the transition occurs with respect to the sampling points. This filter delay must be taken into account when performing message arbitration.

For example, if the frequency of the MUX interface clock ($f_{BDLC}$) is 1.0486 MHz, then the period ($t_{BDLC}$) is 954 ns and the maximum filter delay in the absence of noise will be 15.259 μs.

The effect of random noise on the BDRxD signal depends on the characteristics of the noise itself. Narrow noise pulses on the BDRxD signal will be ignored completely if they are shorter than the filter delay. This provides a degree of low pass filtering.

If noise occurs during a symbol transition, the detection of that transition can be delayed by an amount equal to the length of the noise burst. This is just a reflection of the uncertainty of where the transition is truly occurring within the noise.

Noise pulses that are wider than the filter delay, but narrower than the shortest allowable symbol length, will be detected by the next stage of the BDLC's receiver as an invalid symbol.

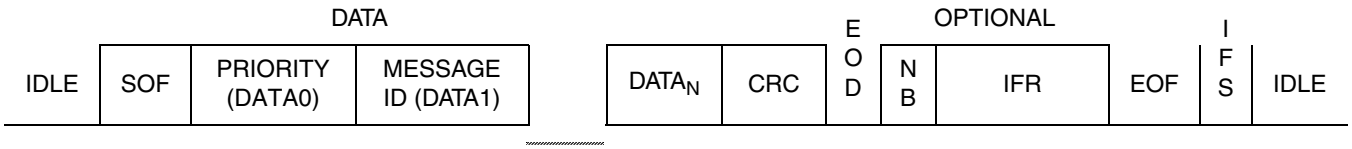
Noise pulses that are longer than the shortest allowable symbol length will be detected normally as an invalid symbol or as invalid data when the frame's CRC is checked.

### 27.4.2 J1850 Frame Format

All messages transmitted on the J1850 bus are structured using the format shown in .

J1850 states that each message has a maximum length of 101 PWM bit times or 12 VPW bytes, excluding SOF, EOD, NB, and EOF, with each byte transmitted MSB first.

All VPW symbol lengths in the following descriptions are typical values at a 10.4 kbps bit rate.




. J1850 Bus Message Format (VPW)

**SOF — Start-of-Frame Symbol**

All messages transmitted onto the J1850 bus must begin with a long-active 200-µs period SOF symbol. This indicates the start of a new message transmission. The SOF symbol is not used in the CRC calculation.

**Data — In-Message Data Bytes**

The data bytes contained in the message include the message priority/type, message ID byte (typically the physical address of the responder), and any actual data being transmitted to the receiving node. The message format used by the BDLC is similar to the 3-byte consolidated header message format outlined by the SAE J1850 document. See *SAE J1850 — Class B Data Communications Network Interface* for more information about 1- and 3-byte headers.

Messages transmitted by the BDLC onto the J1850 bus must contain at least one data byte and, therefore, can be as short as one data byte and one CRC byte. Each data byte in the message is eight bits in length and is transmitted MSB to LSB.

**CRC — Cyclical Redundancy Check Byte**

This byte is used by the receiver(s) of each message to determine if any errors have occurred during the transmission of the message. The BDLC calculates the CRC byte and appends it onto any messages transmitted onto the J1850 bus. It also performs CRC detection on any messages it receives from the J1850 bus.

CRC generation uses the divisor polynomial $X^8 + X^4 + X^3 + X^2 + 1$. The remainder polynomial initially is set to all ones. Each byte in the message after the start of frame (SOF) symbol is processed serially through the CRC generation circuitry. The one's complement of the remainder then becomes the 8-bit CRC byte, which is appended to the message after the data bytes in MSB-to-LSB order.

When receiving a message, the BDLC uses the same divisor polynomial. All data bytes, excluding the SOF and end of data symbols (EOD) but including the CRC byte, are used to check the CRC. If the message is error free, the remainder polynomial will equal $X^7 + X^6 + X^2$ = $C4, regardless of the data contained in the message. If the calculated CRC does not equal $C4, the BDLC will recognize this as a CRC error and set the CRC error flag in the BSVR.

**EOD — End-of-Data Symbol**

The EOD symbol is a long 200-µs passive period on the J1850 bus used to signify to any recipients of a message that the transmission by the originator has completed. No flag is set upon reception of the EOD symbol.

### IFR — In-Frame Response Bytes

The IFR section of the J1850 message format is optional. Users desiring further definition of in-frame response should review the *SAE J1850 — Class B Data Communications Network Interface* specification.

### EOF — End-of-Frame Symbol

This symbol is a long 280-µs passive period on the J1850 bus and is longer than an end-of-data (EOD) symbol, which signifies the end of a message. Since an EOF symbol is longer than a 200-µs EOD symbol, if no response is transmitted after an EOD symbol, it becomes an EOF, and the message is assumed to be completed. The EOF flag is set upon receiving the EOF symbol.

### IFS — Inter-Frame Separation Symbol

The IFS symbol is a 20-µs passive period on the J1850 bus which allows proper synchronization between nodes during continuous message transmission. The IFS symbol is transmitted by a node after the completion of the end-of-frame (EOF) period and, therefore, is seen as a 300-µs passive period.

When the last byte of a message has been transmitted onto the J1850 bus and the EOF symbol time has expired, all nodes then must wait for the IFS symbol time to expire before transmitting a start-of-frame (SOF) symbol, marking the beginning of another message.

However, if the BDLC is waiting for the IFS period to expire before beginning a transmission and a rising edge is detected before the IFS time has expired, it will synchronize internally to that edge. If a write to the BDR register (for instance, to initiate transmission) occurred on or before 104 • $t_{BDLC}$ from the received rising edge, then the BDLC will transmit and arbitrate for the bus. If a CPU write to the BDR register occurred after 104 • $t_{BDLC}$ from the detection of the rising edge, then the BDLC will not transmit, but will wait for the next IFS period to expire before attempting to transmit the byte.

A rising edge may occur during the IFS period because of varying clock tolerances and loading of the J1850 bus, causing different nodes to observe the completion of the IFS period at different times. To allow for individual clock tolerances, receivers must synchronize to any SOF occurring during an IFS period.

***NOTE***

*If two messages are received with a 300µs (± 1µs) interframe separation (IFS) as measured at the RX pin, the start-of-frame (SOF) symbol of the second message will generate an invalid symbol interrupt. This interrupt results in the second message being lost and will therefore be unavailable to the application software. Implementations of this BDLC design on silicon have not been exposed to interframe separation rates faster than 320µs in practical application and have therefore previously not exhibited this behavior. Ensuring that no nodes on the J1850 network transmit messages at 300µs (± 1µs) IFS will avoid this missed message frame. In addition, developing application software to robustly handle lost messages will minimize application impact.*

### BREAK — Break

The BDLC cannot transmit a BREAK symbol.

If the BDLC is transmitting at the time a BREAK is detected, it treats the BREAK as if a transmission error had occurred and halts transmission.

If the BDLC detects a BREAK symbol while receiving a message, it treats the BREAK as a reception error and sets the invalid symbol flag in the BSVR, also ignoring the frame it was receiving. If while receiving a message in 4X mode, the BDLC detects a BREAK symbol, it treats the BREAK as a reception error, sets the invalid symbol flag, and exits 4X mode (for example, the RX4XE bit in BCR2 is cleared automatically). If bus control is required after the BREAK symbol is received and the IFS time has elapsed, the programmer must resend the transmission byte using highest priority.

***NOTE***

*The J1850 protocol BREAK symbol is not related to the HC08 break module. Chapter 13 Break Module (BRK)*

**IDLE — Idle Bus**

An idle condition exists on the bus during any passive period after expiration of the IFS period (for instance, ≥ 300 μs). Any node sensing an idle bus condition can begin transmission immediately.

### 27.4.3 J1850 VPW Symbols

Huntsinger's variable pulse width modulation (VPW) is an encoding technique in which each bit is defined by the time between successive transitions and by the level of the bus between transitions (for instance, active or passive). Active and passive bits are used alternately. This encoding technique is used to reduce the number of bus transitions for a given bit rate.

Each logic 1 or logic 0 contains a single transition and can be at either the active or passive level and one of two lengths, either 64 μs or 128 μs ($t_{NOM}$ at 10.4 kbps baud rate), depending upon the encoding of the previous bit. The start-of-frame (SOF), end-of-data (EOD), end-of-frame (EOF), and inter-frame separation (IFS) symbols always will be encoded at an assigned level and length. See Figure 27-6.

Each message will begin with an SOF symbol an active symbol and, therefore, each data byte (including the CRC byte) will begin with a passive bit, regardless of whether it is a logic 1 or a logic 0.

All VPW bit lengths stated in the following descriptions are typical values at a 10.4 kbps bit rate.

**Logic 0**

A logic 0 is defined as either:

- An active-to-passive transition followed by a passive period 64 μs in length, or
- A passive-to-active transition followed by an active period 128 μs in length

See Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times **(a)**.

**Logic 1**

A logic 1 is defined as either:

- An active-to-passive transition followed by a passive period 128 μs in length, or
- A passive-to-active transition followed by an active period 64 μs in length

See Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times (b).

**Normalization Bit (NB)**

The NB symbol has the same property as a logic 1 or a logic 0. It is only used in IFR message responses.

**Break Signal (BREAK)**

The BREAK signal is defined as a passive-to-active transition followed by an active period of at least 240 μs (See Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times (c)).

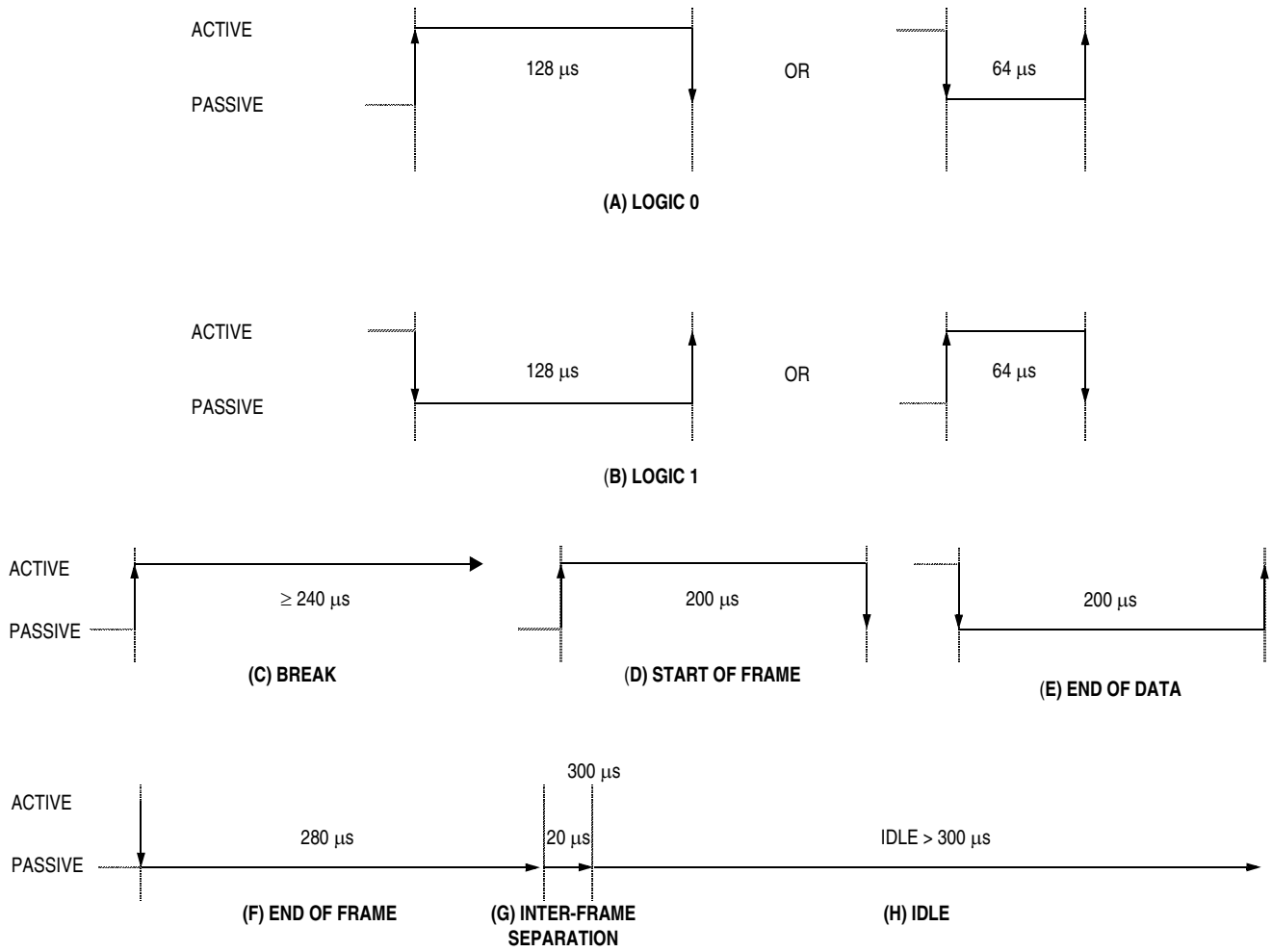



**Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times**

### Start-of-Frame Symbol (SOF)

The SOF symbol is defined as passive-to-active transition followed by an active period 200 µs in length (See Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times (d)). This allows the data bytes which follow the SOF symbol to begin with a passive bit, regardless of whether it is a logic 1 or a logic 0.

### End-of-Data Symbol (EOD)

The EOD symbol is defined as an active-to-passive transition followed by a passive period 200 µs in length (See Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times (e)).

### End-of-Frame Symbol (EOF)

The EOF symbol is defined as an active-to-passive transition followed by a passive period 280 µs in length (See Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times (f)). If no IFR byte is transmitted after an EOD symbol is transmitted, after another 80 µs the EOD becomes an EOF, indicating completion of the message.

### Inter-Frame Separation Symbol (IFS)

The IFS symbol is defined as a passive period 300 µs in length. The 20-µs IFS symbol contains no transition, since when used it always appends to an EOF symbol (See Figure 27-6. J1850 VPW Symbols with Nominal Symbol Times (g)).

**Idle**

An idle is defined as a passive period greater than 300 μs in length.

### 27.4.4 J1850 VPW Valid/Invalid Bits and Symbols

The timing tolerances for **receiving** data bits and symbols from the J1850 bus have been defined to allow for variations in oscillator frequencies. In many cases the maximum time allowed to define a data bit or symbol is equal to the minimum time allowed to define another data bit or symbol.

Since the minimum resolution of the BDLC for determining what symbol is being received is equal to a single period of the MUX interface clock ($t_{BDLC}$), an apparent separation in these maximum time/minimum time concurrences equal to one cycle of $t_{BDLC}$ occurs.

This one clock resolution allows the BDLC to differentiate properly between the different bits and symbols. This is done without reducing the valid window for receiving bits and symbols from transmitters onto the J1850 bus which have varying oscillator frequencies.

In Huntsinger's' variable pulse width (VPW) modulation bit encoding, the tolerances for both the passive and active data bits received and the symbols received are defined with no gaps between definitions. For example, the maximum length of a passive logic 0 is equal to the minimum length of a passive logic 1, and the maximum length of an active logic 0 is equal to the minimum length of a valid SOF symbol.

**Invalid Passive Bit**

See Figure 27-7 (1). If the passive-to-active received transition beginning the next data bit or symbol occurs between the active-to-passive transition beginning the current data bit (or symbol) and **a**, the current bit would be invalid.

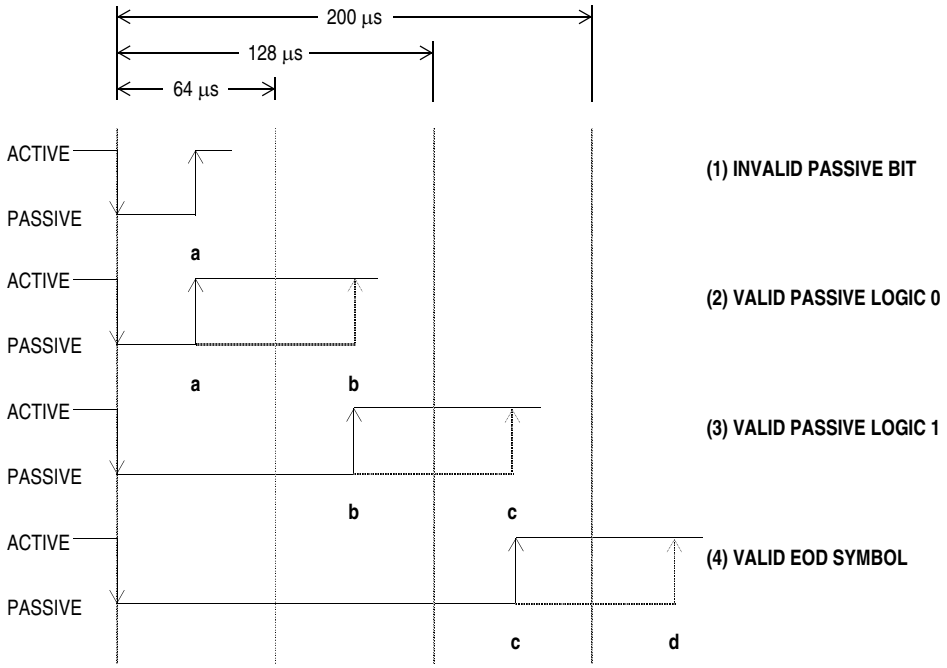



**Figure 27-7. J1850 VPW Received Passive Symbol Times**

### Valid Passive Logic 0

See Figure 27-7 (2). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit would be considered a logic 0.

### Valid Passive Logic 1

See Figure 27-7 (3). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit would be considered a logic 1.

### Valid EOD Symbol

See Figure 27-7 (4). If the passive-to-active received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol would be considered a valid end-of-data symbol (EOD).

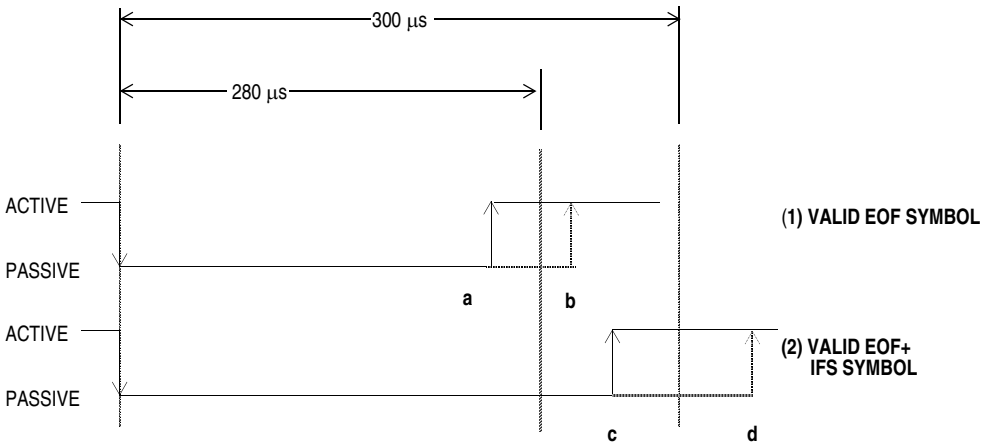



**Figure 27-8. J1850 VPW Received Passive EOF and IFS Symbol Times**

### Valid EOF and IFS Symbol

In Figure 27-8 (1), if the passive-to-active received transition beginning the SOF symbol of the next message occurs between **a** and **b**, the current symbol will be considered a valid end-of-frame (EOF) symbol.

See Figure 27-8 (2). If the passive-to-active received transition beginning the SOF symbol of the next message occurs between c and d, the current symbol will be considered a valid EOF symbol followed by a valid inter-frame separation symbol (IFS). All nodes must wait until a valid IFS symbol time has expired before beginning transmission. However, due to variations in clock frequencies and bus loading, some nodes may recognize a valid IFS symbol before others and immediately begin transmitting. Therefore, any time a node waiting to transmit detects a passive-to-active transition once a valid EOF has been detected, it should immediately begin transmission, initiating the arbitration process.

### Idle Bus

In Figure 27-8 (2), if the passive-to-active received transition beginning the start-of-frame (SOF) symbol of the next message does not occur before **d,** the bus is considered to be idle, and any node wishing to transmit a message may do so immediately.

### Invalid Active Bit

In Figure 27-9 (1), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between the passive-to-active transition beginning the current data bit (or symbol) and **a**, the current bit would be invalid.

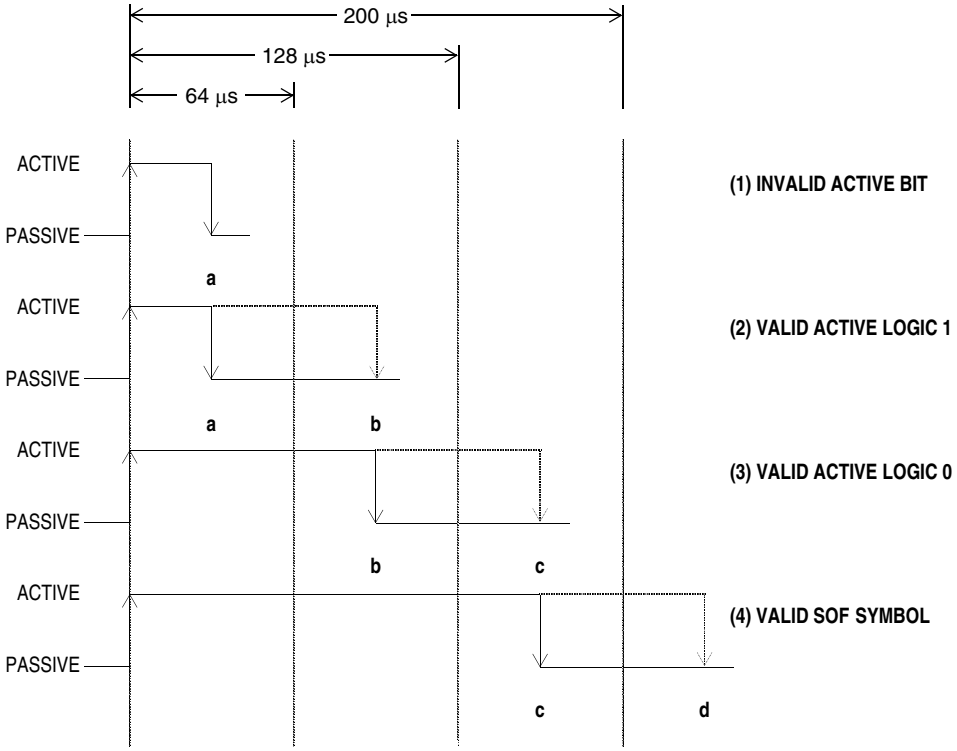



**Figure 27-9. J1850 VPW Received Active Symbol Times**

### Valid Active Logic 1

In Figure 27-9 (2), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **a** and **b**, the current bit would be considered a logic 1.

### Valid Active Logic 0

In Figure 27-9 (3), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **b** and **c**, the current bit would be considered a logic 0.

### Valid SOF Symbol

In Figure 27-9 (4), if the active-to-passive received transition beginning the next data bit (or symbol) occurs between **c** and **d**, the current symbol would be considered a valid SOF symbol.

### Valid BREAK Symbol

In Figure 27-10, if the next active-to-passive received transition does not occur until after **e**, the current symbol will be considered a valid BREAK symbol. A BREAK symbol should be followed by a start-of-frame (SOF) symbol beginning the next message to be transmitted onto the J1850 bus. See **J1850 Frame Format** for BDLC response to BREAK symbols.

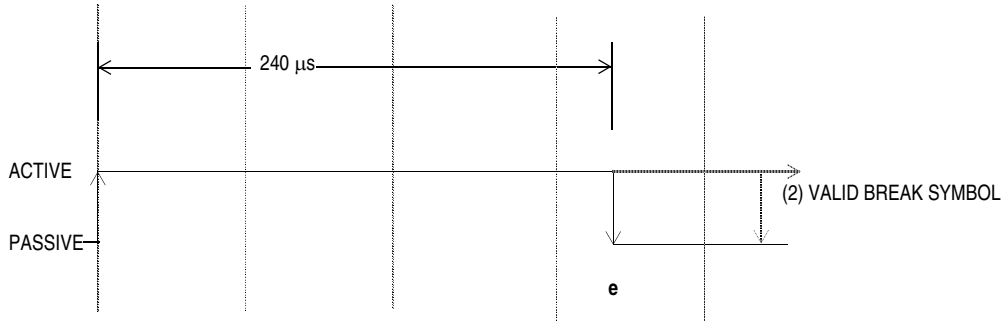



**Figure 27-10. J1850 VPW Received BREAK Symbol Times**

### 27.4.5 Message Arbitration

Message arbitration on the J1850 bus is accomplished in a non-destructive manner, allowing the message with the highest priority to be transmitted, while any transmitters which lose arbitration simply stop transmitting and wait for an idle bus to begin transmitting again.

If the BDLC wants to transmit onto the J1850 bus, but detects that another message is in progress, it waits until the bus is idle. However, if multiple nodes begin to transmit in the same synchronization window, message arbitration will occur beginning with the first bit after the SOF symbol and will continue with each bit thereafter.

The variable pulse width modulation (VPW) symbols and J1850 bus electrical characteristics are chosen carefully so that a logic 0 (active or passive type) will always dominate over a logic 1 (active or passive type) that is simultaneously transmitted. Hence, logic 0s are said to be dominant and logic 1s are said to be recessive.

Whenever a node detects a dominant bit on BDRxD when it transmitted a recessive bit, the node loses arbitration and immediately stops transmitting. This is known as bitwise arbitration.

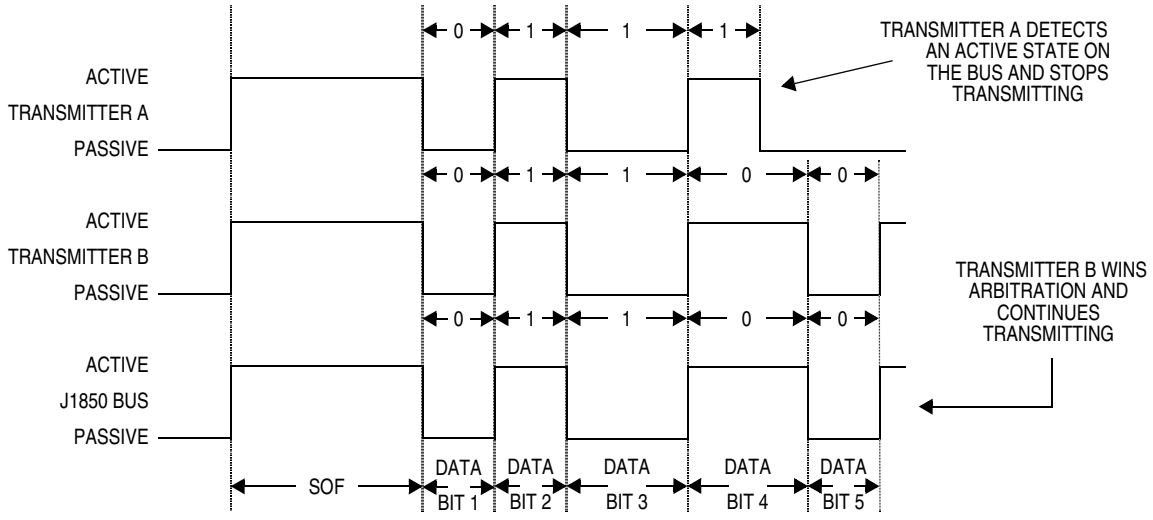



**Figure 27-11. J1850 VPW Bitwise Arbitrations**

Since a logic 0 dominates a logic 1, the message with the lowest value will have the highest priority and will always win arbitration. For instance, a message with priority 000 will win arbitration over a message with priority 011.

This method of arbitration will work no matter how many bits of priority encoding are contained in the message.

During arbitration, or even throughout the transmitting message, when an opposite bit is detected, transmission is stopped immediately unless it occurs on the 8th bit of a byte. In this case, the BDLC automatically will append up to two extra logic 1 bits and then stop transmitting. These two extra bits will be arbitrated normally and thus will not interfere with another message. The second logic 1 bit will not be sent if the first loses arbitration. If the BDLC has lost arbitration to another valid message, then the two extra logic 1s will not corrupt the current message. However, if the BDLC has lost arbitration due to noise on the bus, then the two extra logic 1s will ensure that the current message will be detected and ignored as a noise-corrupted message.

## 27.5 BDLC Protocol Handler

The protocol handler is responsible for framing, arbitration, CRC generation/checking, and error detection. The protocol handler conforms to *SAE J1850 — Class B Data Communications Network Interface*.

***NOTE***

*Freescale assumes that the reader is familiar with the J1850 specification before this protocol handler description is read.*

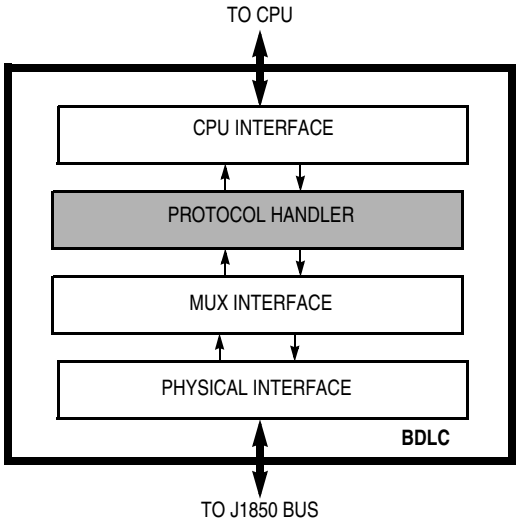



**Figure 27-12. BDLC Block Diagram**

### 27.5.1 Protocol Architecture

The protocol handler contains the state machine, Rx shadow register, Tx shadow register, Rx shift register, Tx shift register, and loopback multiplexer as shown in Figure 27-13.

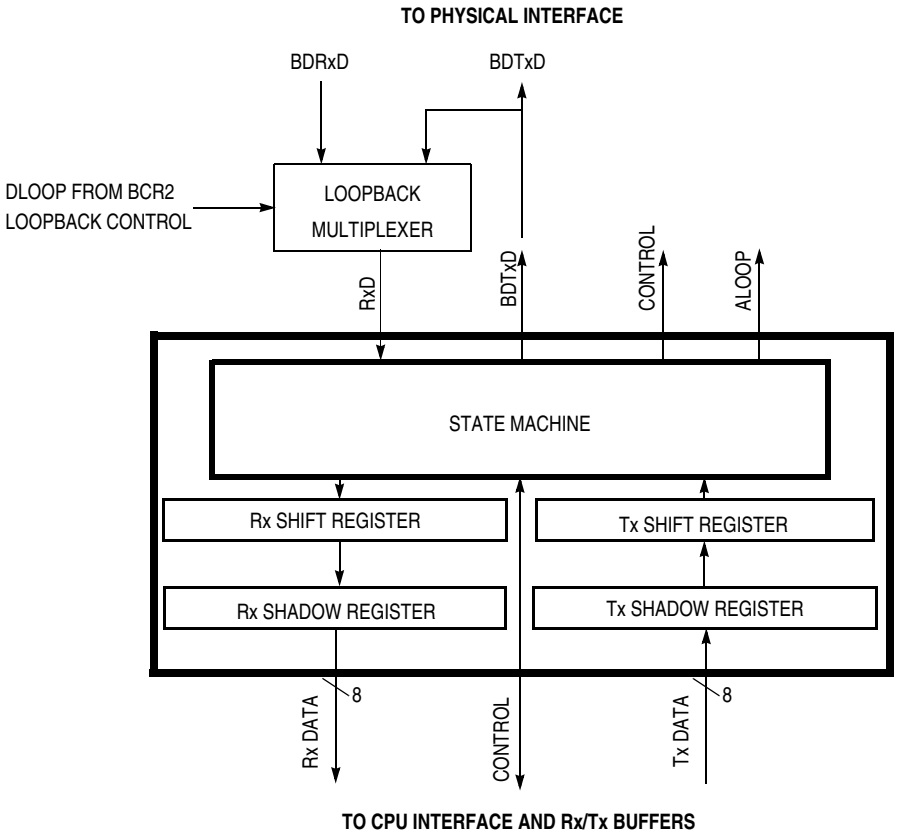



**Figure 27-13. BDLC Protocol Handler Outline**

### 27.5.2 Rx and Tx Shift Registers

The Rx shift register gathers received serial data bits from the J1850 bus and makes them available in parallel form to the Rx shadow register. The Tx shift register takes data, in parallel form, from the Tx shadow register and presents it serially to the state machine so that it can be transmitted onto the J1850 bus.

### 27.5.3 Rx and Tx Shadow Registers

Immediately after the Rx shift register has completed shifting in a byte of data, this data is transferred to the Rx shadow register and RDRF or RXIFR is set (see 27.6.4 BDLC State Vector Register) and an interrupt is generated if the interrupt enable bit (IE) in BCR1 is set. After the transfer takes place, this new data byte in the Rx shadow register is available to the CPU interface, and the Rx shift register is ready to shift in the next byte of data. Data in the Rx shadow register must be retrieved by the CPU before it is overwritten by new data from the Rx shift register.

Once the Tx shift register has completed its shifting operation for the current byte, the data byte in the Tx shadow register is loaded into the Tx shift register. After this transfer takes place, the Tx shadow register is ready to accept new data from the CPU when TDRE flag in BSVR is set.

### 27.5.4 Digital Loopback Multiplexer

The digital loopback multiplexer connects RxD to either BDTxD or BDRxD, depending on the state of the DLOOP bit in the BCR2 register (see 27.6.3 BDLC Control Register 2).

### 27.5.5 State Machine

All of the functions associated with performing the protocol are executed or controlled by the state machine. The state machine is responsible for framing, collision detection, arbitration, CRC generation/checking, and error detection. The following sections describe the BDLC's actions in a variety of situations.

#### *27.5.5.1 4X Mode*

The BDLC can exist on the same J1850 bus as modules which use a special 4X (41.6 kbps) mode of J1850 variable pulse width modulation (VPW) operation. The BDLC cannot transmit in 4X mode, but can receive messages in 4X mode, if the RX4X bit is set in BCR2 register. If the RX4X bit is not set in the BCR2 register, any 4X message on the J1850 bus is treated as noise by the BDLC and is ignored.

#### *27.5.5.2 Receiving a Message in Block Mode*

Although not a part of the SAE J1850 protocol, the BDLC does allow for a special block mode of operation of the receiver. As far as the BDLC is concerned, a block mode message is simply a long J1850 frame that contains an indefinite number of data bytes. All of the other features of the frame remain the same, including the SOF, CRC, and EOD symbols.

Another node wishing to send a block mode transmission must first inform all other nodes on the network that this is about to happen. This is usually accomplished by sending a special predefined message.

#### *27.5.5.3 Transmitting a Message in Block Mode*

A block mode message is transmitted inherently by simply loading the bytes one by one into the BDR register until the message is complete. The programmer should wait until the TDRE flag (see 27.6.4 BDLC State Vector Register) is set prior to writing a new byte of data into the BDR register. The BDLC does not contain any predefined maximum J1850 message length requirement.

#### *27.5.5.4 J1850 Bus Errors*

The BDLC detects several types of transmit and receive errors which can occur during the transmission of a message onto the J1850 bus.

**Transmission Error**

If the message transmitted by the BDLC contains invalid bits or framing symbols on non-byte boundaries, this constitutes a transmission error. When a transmission error is detected, the BDLC immediately will cease transmitting. The error condition ($1C) is reflected in the BSVR register (see Table 27-5). If the interrupt enable bit (IE in BCR1) is set, a CPU interrupt request from the BDLC is generated.

### CRC Error

A cyclical redundancy check (CRC) error is detected when the data bytes and CRC byte of a received message are processed and the CRC calculation result is not equal to $C4. The CRC code will detect any single and 2-bit errors, as well as all 8-bit burst errors and almost all other types of errors. The CRC error flag ($18 in BSVR) is set when a CRC error is detected. (See 27.6.4 BDLC State Vector Register.)

### Symbol Error

A symbol error is detected when an abnormal (invalid) symbol is detected in a message being received from the J1850 bus. However, if the BDLC is transmitting when this happens, it will be treated as a loss of arbitration ($14 in BSVR) rather than a transmitter error. The ($1C) symbol invalid or the out-of-range flag is set when a symbol error is detected. Therefore, ($1C) symbol invalid flag is stacked behind the ($14) LOA flag during a transmission error process. (See 27.6.4 BDLC State Vector Register.)

### Framing Error

A framing error is detected if an EOD or EOF symbol is detected on a non-byte boundary from the J1850 bus. A framing error also is detected if the BDLC is transmitting the EOD and instead receives an active symbol. The ($1C) symbol invalid or the out-of-range flag is set when a framing error is detected. (See 27.6.4 BDLC State Vector Register.)

### Bus Fault

If a bus fault occurs, the response of the BDLC will depend upon the type of bus fault.
If the bus is shorted to battery, the BDLC will wait for the bus to fall to a passive state before it will attempt to transmit a message. As long as the short remains, the BDLC will never attempt to transmit a message onto the J1850 bus.

If the bus is shorted to ground, the BDLC will see an idle bus, begin to transmit the message, and then detect a transmission error ($1C in BSVR), since the short to ground would not allow the bus to be driven to the active (dominant) SOF state. The BDLC will abort that transmission and wait for the next CPU command to transmit.

In any case, if the bus fault is temporary, as soon as the fault is cleared, the BDLC will resume normal operation. If the bus fault is permanent, it may result in permanent loss of communication on the J1850 bus. (See BDLC State Vector Register.)

### BREAK — Break

If a BREAK symbol is received while the BDLC is transmitting or receiving, an invalid symbol ($1C in BSVR) interrupt will be generated. Reading the BSVR register (see 27.6.4 BDLC State Vector Register.) will clear this interrupt condition. The BDLC will wait for the bus to idle, then wait for a start-of-frame (SOF) symbol.

The BDLC cannot transmit a BREAK symbol. It can only receive a BREAK symbol from the J1850 bus.

#### *27.5.5.5 Summary*

**Table 27-1. BDLC J1850 Bus Error Summary**

| Error Condition | BDLC Function |
|---|---|
| Transmission Error | For invalid bits or framing symbols on non-byte boundaries, invalid symbol interrupt will be generated. BDLC stops transmission. |
| Cyclical Redundancy Check (CRC) Error | CRC error interrupt will be generated. The BDLC will wait for SOF. |
| Invalid Symbol: BDLC Receives Invalid Bits (Noise) | The BDLC will abort transmission immediately. Invalid symbol interrupt will be generated. |
| Framing Error | Invalid symbol interrupt will be generated. The BDLC will wait for start-of-frame (SOF). |
| Bus Short to $V_{DD}$ | The BDLC will not transmit until the bus is idle. |
| Bus Short to GND | Thermal overload will shut down physical interface. Fault condition is reflected in BSVR as an invalid symbol. |
| BDLC Receives BREAK Symbol. | The BDLC will wait for the next valid SOF. Invalid symbol interrupt will be generated. |

## 27.6 BDLC CPU Interface

The CPU interface provides the interface between the CPU and the BDLC and consists of five user registers.

- BDLC analog and roundtrip delay register (BARD)
- BDLC control register 1 (BCR1)
- BDLC control register 2 (BCR2)
- BDLC state vector register (BSVR)
- BDLC data register (BDR)

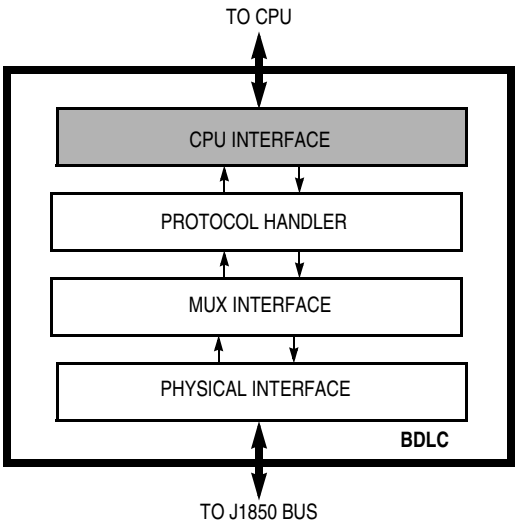



**Figure 27-14. BDLC Block Diagram**

### 27.6.1 BDLC Analog and Roundtrip Delay Register

This register programs the BDLC to compensate for various delays of different external transceivers. The default delay value is16 μs. Timing adjustments from 9 μs to 24 μs in steps of 1 μs are available. The BARD register can be written only once after each reset, after which they become read-only bits. The register may be read at any time.

Address: $003B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | ATE | RXPOL | 0 | 0 | BO3 | BO2 | BO1 | BO0 |
| Write: | | | R | R | | | | |
| Reset: | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

R = Reserved

**Figure 27-15. BDLC Analog and Roundtrip Delay Register (BARD)**

**ATE — Analog Transceiver Enable Bit**

The analog transceiver enable (ATE) bit is used to select either the on-board or an off-chip analog transceiver.

1 = Select on-board analog transceiver
0 = Select off-chip analog transceiver

***NOTE***

*This device does not contain an on-board transceiver. This bit should be programmed to a 0 for proper operation.*

**RXPOL — Receive Pin Polarity Bit**

The receive pin polarity (RXPOL) bit is used to select the polarity of an incoming signal on the receive pin. Some external analog transceivers invert the receive signal from the J1850 bus before feeding it back to the digital receive pin.

1 = Select normal/true polarity; true non-inverted signal from the J1850 bus; for example, the external transceiver does not invert the receive signal
0 = Select inverted polarity, where an external transceiver inverts the receive signal from the J1850 bus

**B03–B00 — BARD Offset Bits**

Table 27-2 shows the expected transceiver delay with respect to BARD offset values.

**Table 27-2. BDLC Transceiver Delay**

| BARD Offset Bits B0[3:0] | Corresponding Expected Transceiver's Delays (μs) |
|---|---|
| 0000 | 9 |
| 0001 | 10 |
| 0010 | 11 |
| 0011 | 12 |
| 0100 | 13 |
| 0101 | 14 |
| 0110 | 15 |
| 0111 | 16 |

**Table 27-2. BDLC Transceiver Delay (Continued)**

| BARD Offset Bits B0[3:0] | Corresponding Expected Transceiver's Delays (μs) |
|---|---|
| 1000 | 17 |
| 1001 | 18 |
| 1010 | 19 |
| 1011 | 20 |
| 1100 | 21 |
| 1101 | 22 |
| 1110 | 23 |
| 1111 | 24 |

## 27.6.2 BDLC Control Register 1

This register is used to configure and control the BDLC.

Address: $003C

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | IMSG | CLKS | R1 | R0 | 0 | 0 | IE | WCM |
| Write: | | | | | R | R | | |
| Reset: | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 27-16. BDLC Control Register 1 (BCR1)**

### IMSG — Ignore Message Bit

This bit is used to disable the receiver until a new start-of-frame (SOF) is detected.

1 = Disable receiver. When set, all BDLC interrupt requests will be masked and the status bits will be held in their reset state. If this bit is set while the BDLC is receiving a message, the rest of the incoming message will be ignored.

0 = Enable receiver. This bit is cleared automatically by the reception of an SOF symbol or a BREAK symbol. It will then generate interrupt requests and will allow changes of the status register to occur. However, these interrupts may still be masked by the interrupt enable (IE) bit.

### CLKS — Clock Bit

The nominal BDLC operating frequency ($f_{BDLC}$) must always be 1.048576 MHz or 1 MHz for J1850 bus communications to take place. The CLKS register bit allows the user to select the frequency (1.048576 MHz or 1 MHz) used to adjust symbol timing automatically.

1 = Binary frequency (1.048576 MHz) selected for $f_{BDLC}$

0 = Integer frequency (1 MHz) selected for $f_{BDLC}$

### R1 and R0 — Rate Select Bits

These bits determine the amount by which the frequency of the MCU CGMXCLK signal is divided to form the MUX interface clock ($f_{BDLC}$) which defines the basic timing resolution of the MUX interface. They may be written only once after reset, after which they become read-only bits.

The nominal frequency of $f_{BDLC}$ must always be 1.048576 MHz or 1.0 MHz for J1850 bus communications to take place. Hence, the value programmed into these bits is dependent on the chosen MCU system clock frequency per Table 27-3

**Table 27-3. BDLC Rate Selection**

| $f_{XCLK}$ Frequency | R1 | R0 | Division | $f_{BDLC}$ |
|---|---|---|---|---|
| 1.049 MHz | 0 | 0 | 1 | 1.049 MHz |
| 2.097 MHz | 0 | 1 | 2 | 1.049 MHz |
| 4.194 MHz | 1 | 0 | 4 | 1.049 MHz |
| 8.389 MHz | 1 | 1 | 8 | 1.049 MHz |
| 1.000 MHz | 0 | 0 | 1 | 1.00 MHz |
| 2.000 MHz | 0 | 1 | 2 | 1.00 MHz |
| 4.000 MHz | 1 | 0 | 4 | 1.00 MHz |
| 8.000 MHz | 1 | 1 | 8 | 1.00 MHz |

**IE— Interrupt Enable Bit**

This bit determines whether the BDLC will generate CPU interrupt requests in run mode. It does not affect CPU interrupt requests when exiting the BDLC stop or BDLC wait modes. Interrupt requests will be maintained until all of the interrupt request sources are cleared by performing the specified actions upon the BDLC's registers. Interrupts that were pending at the time that this bit is cleared may be lost.

1 = Enable interrupt requests from BDLC
0 = Disable interrupt requests from BDLC

If the programmer does not wish to use the interrupt capability of the BDLC, the BDLC state vector register (BSVR) can be polled periodically by the programmer to determine BDLC states. See 27.6.4 BDLC State Vector Register for a description of the BSVR.

**WCM — Wait Clock Mode Bit**

This bit determines the operation of the BDLC during CPU wait mode. See Stop Mode and Wait Mode for more details on its use.

1 = Stop BDLC internal clocks during CPU wait mode
0 = Run BDLC internal clocks during CPU wait mode

## 27.6.3 BDLC Control Register 2

This register controls transmitter operations of the BDLC. It is recommended that BSET and BCLR instructions be used to manipulate data in this register to ensure that the register's content does not change inadvertently.

Address: $003D

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | ALOOP | DLOOP | RX4XE | NBFS | TEOD | TSIFR | TMIFR1 | TMIFR0 |
| Reset: | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 27-17. BDLC Control Register 2 (BCR2)**

**ALOOP — Analog Loopback Mode Bit**

This bit determines whether the J1850 bus will be driven by the analog physical interface's final drive stage. The programmer can use this bit to reset the BDLC state machine to a known state after the off-chip analog transceiver is placed in loopback mode. When the user clears ALOOP, to indicate that

the off-chip analog transceiver is no longer in loopback mode, the BDLC waits for an EOF symbol before attempting to transmit.

1 = Input to the analog physical interface's final drive stage is looped back to the BDLC receiver. The J1850 bus is not driven.

0 = The J1850 bus will be driven by the BDLC. After the bit is cleared, the BDLC requires the bus to be idle for a minimum of end-of-frame symbol time ($t_{TRV4}$) before message reception or a minimum of inter-frame symbol time ($t_{TRV6}$) before message transmission. (See 28.1.15 BDLC Transmitter VPW Symbol Timings.

**DLOOP — Digital Loopback Mode Bit**

This bit determines the source to which the digital receive input (BDRxD) is connected and can be used to isolate bus fault conditions (see Figure 27-13). If a fault condition has been detected on the bus, this control bit allows the programmer to connect the digital transmit output to the digital receive input. In this configuration, data sent from the transmit buffer will be reflected back into the receive buffer. If no faults exist in the BDLC, the fault is in the physical interface block or elsewhere on the J1850 bus.

1 = When set, BDRxD is connected to BDTxD. The BDLC is now in digital loopback mode.

0 = When cleared, BDTxD is not connected to BDRxD. The BDLC is taken out of digital loopback mode and can now drive the J1850 bus normally.

**RX4XE — Receive 4X Enable Bit**

This bit determines if the BDLC operates at normal transmit and receive speed (10.4 kbps) or receive only at 41.6 kbps. This feature is useful for fast download of data into a J1850 node for diagnostic or factory programming of the node.

1 = When set, the BDLC is put in 4X receive-only operation.

0 = When cleared, the BDLC transmits and receives at 10.4 kbps.

**NBFS — Normalization Bit Format Select Bit**

This bit controls the format of the normalization bit (NB). (See **Figure 27-18**.) SAE J1850 strongly encourages using an active long (logic 0) for in-frame responses containing cyclical redundancy check (CRC) and an active short (logic 1) for in-frame responses without CRC.

1 = NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) does not end with a CRC byte.

0 = NB that is received or transmitted is a 1 when the response part of an in-frame response (IFR) ends with a CRC byte. NB that is received or transmitted is a 0 when the response part of an in-frame response (IFR) does not end with a CRC byte.

**TEOD — Transmit End of Data Bit**

This bit is set by the programmer to indicate the end of a message is being sent by the BDLC. It will append an 8-bit CRC after completing transmission of the current byte. This bit also is used to end an in-frame response (IFR). If the transmit shadow register is full when TEOD is set, the CRC byte will be transmitted after the current byte in the Tx shift register and the byte in the Tx shadow register have been transmitted. (See 27.5.3 Rx and Tx Shadow Registers for a description of the transmit shadow register.) Once TEOD is set, the transmit data register empty flag (TDRE) in the BDLC state vector register (BSVR) is cleared to allow lower priority interrupts to occur. (See 27.6.4 BDLC State Vector Register.)

1 = Transmit end-of-data (EOD) symbol

0 = The TEOD bit will be cleared automatically at the rising edge of the first CRC bit that is sent or if an error is detected. When TEOD is used to end an IFR transmission, TEOD is cleared when the BDLC receives back a valid EOD symbol or an error condition occurs.

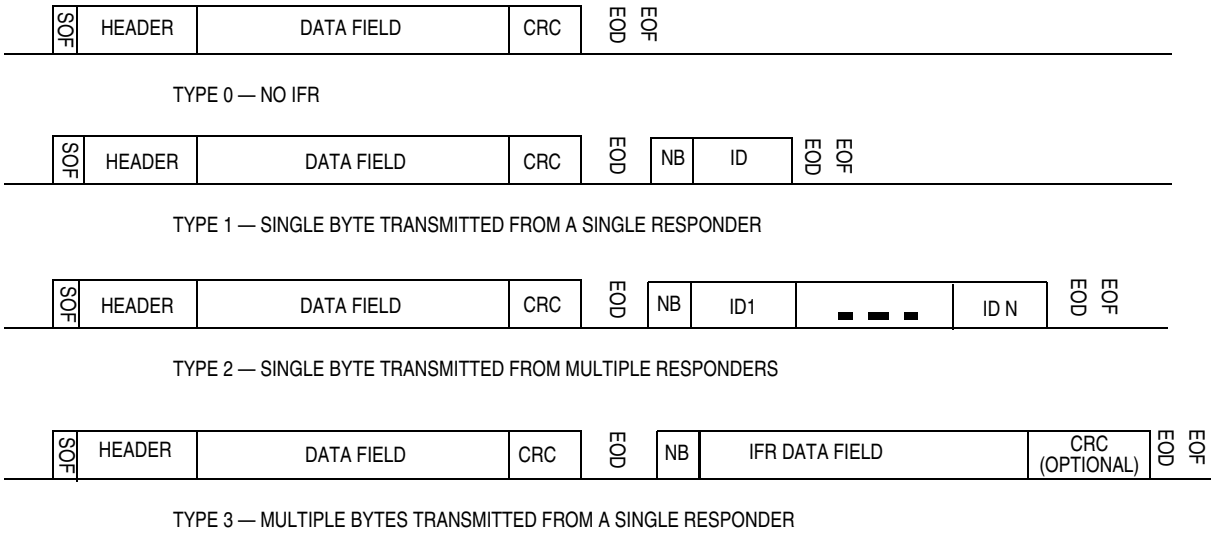
### TSIFR, TMIFR1, and TMIFR0 — Transmit In-Frame Response Control Bits

These three bits control the type of in-frame response being sent. The programmer should not set more than one of these control bits to a 1 at any given time. However, if more than one of these three control bits are set to 1, the priority encoding logic will force these register bits to a known value as shown in Table 27-4. For example, if 011 is written to TSIFR, TMIFR1, and TMIFR0, then internally they will be encoded as 010. However, when these bits are read back, they will read 011.

**Table 27-4. BDLC Transmit In-Frame Response Control Bit Priority Encoding**

| Write/Read TSIFR | Write/Read TMIFR1 | Write/Read TMIFR0 | Actual TSIFR | Actual TMIFR1 | Actual TMIFR0 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | X | X | 1 | 0 | 0 |
| 0 | 1 | X | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |

The BDLC supports the in-frame response (IFR) feature of J1850 by setting these bits correctly. The four types of J1850 IFR are shown below. The purpose of the in-frame response modes is to allow multiple nodes to acknowledge receipt of the data by responding with their personal ID or physical address in a concatenated manner after they have seen the EOD symbol. If transmission arbitration is lost by a node while sending its response, it continues to transmit its ID/address until observing its unique byte in the response stream. For VPW modulation, because the first bit of the IFR is always passive, a normalization bit (active) must be generated by the responder and sent prior to its ID/address byte. When there are multiple responders on the J1850 bus, only one normalization bit is sent which assists all other transmitting nodes to sync up their response.




NB = Normalization Bit
ID = Identifier (usually the physical address of the responder(s))
HEADER = Specifies one of three frame lengths

**Figure 27-18. Types of In-Frame Response (IFR)**

**TSIFR — Transmit Single Byte IFR with No CRC (Type 1 or 2) Bit**

The TSIFR bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR, $003F) as a single byte IFR with no CRC. Typically, the byte transmitted is a unique identifier or address of the transmitting (responding) node. See Figure 27-18.

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC will attempt to transmit the appropriate normalization bit followed by the byte in the BDR.

0 = The TSIFR bit will be cleared automatically, once the BDLC has successfully transmitted the byte in the BDR onto the bus, or TEOD is set, or an error is detected on the bus.

If the programmer attempts to set the TSIFR bit immediately after the EOD symbol has been received from the bus, the TSIFR bit will remain in the reset state and no attempt will be made to transmit the IFR byte.

If a loss of arbitration occurs when the BDLC attempts to transmit and after the IFR byte winning arbitration completes transmission, the BDLC will again attempt to transmit the BDR (with no normalization bit). The BDLC will continue transmission attempts until an error is detected on the bus, or TEOD is set, or the BDLC transmission is successful.

If loss or arbitration occurs in the last two bits of the IFR byte, two additional 1 bits **will not** be sent out because the BDLC will attempt to retransmit the byte in the transmit shift register after the IRF byte winning arbitration completes transmission.

**TMIFR1 — Transmit Multiple Byte IFR with CRC (Type 3) Bit**

The TMIFR1 bit requests the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR with CRC or as a single byte IFR with CRC. Response IFR bytes are still subject to J1850 message length maximums (see J1850 Frame Format and Figure 27-18).

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR register. After TEOD has been set and the last IFR byte has been transmitted, the CRC byte is transmitted.

0 = The TMIFR1 bit will be cleared automatically – once the BDLC has successfully transmitted the CRC byte and EOD symbol – by the detection of an error on the multiplex bus or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

If the TMIFR1 bit is set, the BDLC will attempt to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see 27.6.4 BDLC State Vector Register) will occur similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BDLC control register 2 (BCR2). This will instruct the BDLC to transmit a CRC byte once the byte in the BDR is transmitted and then transmit an EOD symbol, indicating the end of the IFR portion of the message frame.

However, if the programmer wishes to transmit a single byte followed by a CRC byte, the programmer should load the byte into the BDR before the EOD symbol has been received, and then set the TMIFR1 bit. Once the TDRE interrupt occurs, the programmer should then set the TEOD bit in the BCR2. This will result in the byte in the BDR being the only byte transmitted before the IFR CRC byte, and no TDRE interrupt will be generated.

If the programmer attempts to set the TMIFR1 bit immediately after the EOD symbol has been received from the bus, the TMIFR1 bit will remain in the reset state, and no attempt will be made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting any byte of a multiple byte IFR, the BDLC will go to the loss of arbitration state, set the appropriate flag, and cease transmission.
If the BDLC loses arbitration during the IFR, the TMIFR1 bit will be cleared and no attempt will be made to retransmit the byte in the BDR. If loss of arbitration occurs in the last two bits of the IFR byte, two additional 1 bits will be sent out.

***NOTE***

*The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise from going onto the J1850 bus from a corrupted message.*

**TMIFR0 — Transmit Multiple Byte IFR without CRC (Type 3) Bit**

The TMIFR0 bit is used to request the BDLC to transmit the byte in the BDLC data register (BDR) as the first byte of a multiple byte IFR without CRC. Response IFR bytes are still subject to J1850 message length maximums (see J1850 Frame Format and Figure 27-18).

1 = If this bit is set prior to a valid EOD being received with no CRC error, once the EOD symbol has been received the BDLC will attempt to transmit the appropriate normalization bit followed by IFR bytes. The programmer should set TEOD after the last IFR byte has been written into the BDR register. After TEOD has been set, the last IFR byte to be transmitted will be the last byte which was written into the BDR register.

0 = The TMIFR0 bit will be cleared automatically; once the BDLC has successfully transmitted the EOD symbol; by the detection of an error on the multiplex bus; or by a transmitter underrun caused when the programmer does not write another byte to the BDR after the TDRE interrupt.

If the TMIFR0 bit is set, the BDLC will attempt to transmit the normalization symbol followed by the byte in the BDR. After the byte in the BDR has been loaded into the transmit shift register, a TDRE interrupt (see27.6.4 BDLC State Vector Register) will occur similar to the main message transmit sequence. The programmer should then load the next byte of the IFR into the BDR for transmission. When the last byte of the IFR has been loaded into the BDR, the programmer should set the TEOD bit in the BCR2. This will instruct the BDLC to transmit an EOD symbol once the byte in the BDR is transmitted, indicating the end of the IFR portion of the message frame. The BDLC will not append a CRC when the TMIFR0 is set.

If the programmer attempts to set the TMIFR0 bit after the EOD symbol has been received from the bus, the TMIFR0 bit will remain in the reset state, and no attempt will be made to transmit an IFR byte.

If a loss of arbitration occurs when the BDLC is transmitting, the TMIFR0 bit will be cleared and no attempt will be made to retransmit the byte in the BDR. If loss of arbitration occurs in the last two bits of the IFR byte, two additional 1 bits (active short bits) will be sent out.

***NOTE***

*The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise from going onto the J1850 bus from a corrupted message.*

### 27.6.4 BDLC State Vector Register

This register is provided to substantially decrease the CPU overhead associated with servicing interrupts while under operation of a multiplex protocol. It provides an index offset that is directly related to the BDLC's current state, which can be used with a user-supplied jump table to rapidly enter an interrupt service routine. This eliminates the need for the user to maintain a duplicate state machine in software.

Address: $003E

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read: | 0 | 0 | I3 | I2 | I1 | I0 | 0 | 0 |
| Write: | R | R | R | R | R | R | R | R |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure 27-19. BDLC State Vector Register (BSVR)**

**I0, I1, I2, and I3 — Interrupt Source Bits**

These bits indicate the source of the interrupt request that currently is pending. The encoding of these bits are listed in Table 27-5.

**Table 27-5. BDLC Interrupt Sources**

| BSVR | I3 | I2 | I1 | I0 | Interrupt Source | Priority |
|---|---|---|---|---|---|---|
| $00 | 0 | 0 | 0 | 0 | No Interrupts Pending | 0 (Lowest) |
| $04 | 0 | 0 | 0 | 1 | Received EOF | 1 |
| $08 | 0 | 0 | 1 | 0 | Received IFR Byte (RXIFR) | 2 |
| $0C | 0 | 0 | 1 | 1 | BDLC Rx Data Register Full (RDRF) | 3 |
| $10 | 0 | 1 | 0 | 0 | BDLC Tx Data Register Empty (TDRE) | 4 |
| $14 | 0 | 1 | 0 | 1 | Loss of Arbitration | 5 |
| $18 | 0 | 1 | 1 | 0 | Cyclical Redundancy Check (CRC) Error | 6 |
| $1C | 0 | 1 | 1 | 1 | Symbol Invalid or Out of Range | 7 |
| $20 | 1 | 0 | 0 | 0 | Wakeup | 8 (Highest) |

Bits I0, I1, I2, and I3 are cleared by a read of the BSVR except when the BDLC data register needs servicing (RDRF, RXIFR, or TDRE conditions). RXIFR and RDRF can be cleared only by a read of the BSVR followed by a read of the BDLC data register (BDR). TDRE can either be cleared by a read of the BSVR followed by a write to the BDLC BDR or by setting the TEOD bit in BCR2.

Upon receiving a BDLC interrupt, the user can read the value within the BSVR, transferring it to the CPU's index register. The value can then be used to index into a jump table, with entries four bytes apart, to quickly enter the appropriate service routine. For example:

```
Service     LDX    BSVR          Fetch State Vector Number
            JMP    JMPTAB,X      Enter service routine,
*                                (must end in RTI)
*
JMPTAB      JMP    SERVE0        Service condition #0
            NOP
            JMP    SERVE1        Service condition #1
            NOP
            JMP    SERVE2        Service condition #2
            NOP
*
            JMP    SERVE8        Service condition #8
            END
```

***NOTE***

*The NOPs are used only to align the JMPs onto 4-byte boundaries so that the value in the BSVR can be used intact. Each of the service routines must end with an RTI instruction to guarantee correct continued operation of the device. Note also that the first entry can be omitted since it corresponds to no interrupt occurring.*

The service routines should clear all of the sources that are causing the pending interrupts. Note that the clearing of a high priority interrupt may still leave a lower priority interrupt pending, in which case bits I0, I1, and I2 of the BSVR will then reflect the source of the remaining interrupt request.

If fewer states are used or if a different software approach is taken, the jump table can be made smaller or omitted altogether.

## 27.6.5 BDLC Data Register

Address: $003F

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| Reset: | Unaffected by Reset | | | | | | | |

**Figure 27-20. BDLC Data Register (BDR)**

This register is used to pass the data to be transmitted to the J1850 bus from the CPU to the BDLC. It is also used to pass data received from the J1850 bus to the CPU. Each data byte (after the first one) should be written only after a Tx data register empty (TDRE) state is indicated in the BSVR.

Data read from this register will be the last data byte received from the J1850 bus. This received data should only be read after an Rx data register full (RDRF) interrupt has occurred. (See 27.6.4 BDLC State Vector Register)

The BDR is double buffered via a transmit shadow register and a receive shadow register. After the byte in the transmit shift register has been transmitted, the byte currently stored in the transmit shadow register is loaded into the transmit shift register. Once the transmit shift register has shifted the first bit out, the TDRE flag is set, and the shadow register is ready to accept the next data byte. The receive shadow register works similarly. Once a complete byte has been received, the receive shift register stores the newly received byte into the receive shadow register. The RDRF flag is set to indicate that a new byte of data has been received. The programmer has one BDLC byte reception time to read the shadow register and clear the RDRF flag before the shadow register is overwritten by the newly received byte.

To abort an in-progress transmission, the programmer should stop loading data into the BDR. This will cause a transmitter underrun error and the BDLC automatically will disable the transmitter on the next non-byte boundary. This means that the earliest a transmission can be halted is after at least one byte plus two extra logic 1s have been transmitted. The receiver will pick this up as an error and relay it in the state vector register as an invalid symbol error.

***NOTE***

The extra logic 1s are an enhancement to the J1850 protocol which forces a byte boundary condition fault. This is helpful in preventing noise from going onto the J1850 bus from a corrupted message.

## 27.7 Low-Power Modes

The following information concerns wait mode and stop mode.

### 27.7.1 Wait Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a WAIT instruction and the WCM bit in BDLC control register 1 (BCR1) is previously clear. In BDLC wait mode, the BDLC cannot drive any data.

A subsequent successfully received message, including one that is in progress at the time that this mode is entered, will cause the BDLC to wake up and generate a CPU interrupt request if the interrupt enable (IE) bit in the BDLC control register 1 (BCR1) is previously set. (See 27.6.2 BDLC Control Register 1 for a better understanding of IE.) This results in less of a power saving, but the BDLC is guaranteed to receive correctly the message which woke it up, since the BDLC internal operating clocks are kept running.

***NOTE***

*Ensuring that all transmissions are complete or aborted before putting the BDLC into wait mode is important.*

### 27.7.2 Stop Mode

This power-conserving mode is entered automatically from run mode whenever the CPU executes a STOP instruction or if the CPU executes a WAIT instruction and the WCM bit in the BDLC control register 1 (BCR1) is previously set. This is the lowest power mode that the BDLC can enter.

A subsequent passive-to-active transition on the J1850 bus will cause the BDLC to wake up and generate a non-maskable CPU interrupt request. When a STOP instruction is used to put the BDLC in stop mode, the BDLC is not guaranteed to correctly receive the message which woke it up, since it may take some time for the BDLC internal operating clocks to restart and stabilize. If a WAIT instruction is used to put the BDLC in stop mode, the BDLC is guaranteed to correctly receive the byte which woke it up, if and only if an end-of-frame (EOF) has been detected prior to issuing the WAIT instruction by the CPU. Otherwise, the BDLC will not correctly receive the byte that woke it up.

If this mode is entered while the BDLC is receiving a message, the first subsequent received edge will cause the BDLC to wake up immediately, generate a CPU interrupt request, and wait for the BDLC internal operating clocks to restart and stabilize before normal communications can resume. Therefore, the BDLC is not guaranteed to receive that message correctly.

***NOTE***

*It is important to ensure all transmissions are complete or aborted prior to putting the BDLC into stop mode.*

# Chapter 28
# Electrical Specifications

## 28.1 Electrical Specifications

### 28.1.1 Maximum Ratings

Maximum ratings are the extreme limits to which the microcontroller unit (MCU) can be exposed without permanently damaging it.

***NOTE***

*This device is not guaranteed to operate properly at the maximum ratings. Refer to 28.1.4 5.0 Volt DC Electrical Characteristics for guaranteed operating conditions.*

| Rating[1] | Symbol | Value | Unit |
|---|---|---|---|
| Supply Voltage | $V_{DD}$ | –0.3 to +6.0 | V |
| Input Voltage | $V_{IN}$ | $V_{SS}$ –0.3 to $V_{DD}$ +0.3 | V |
| Maximum Current Per Pin Excluding $V_{DD}$ and $V_{SS}$ | I | ± 25 | mA |
| Storage Temperature | $T_{STG}$ | –55 to +150 | °C |
| Maximum Current out of $V_{SS}$ | $I_{MVSS}$ | 100 | mA |
| Maximum Current into $V_{DD}$ | $I_{MVDD}$ | 100 | mA |
| Reset and $\overline{IRQ}$ Input Voltage | $V_{HI}$ | $V_{DD}$ + 4.5 | V |

1. Voltages are referenced to $V_{SS}$.

***NOTE***

*This device contains circuitry to protect the inputs against damage due to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum-rated voltages to this high-impedance circuit. For proper operation, it is recommended that $V_{IN}$ and $V_{OUT}$ be constrained to the range $V_{SS} \le (V_{IN}$ or $V_{OUT}) \le V_{DD}$. Reliability of operation is enhanced if unused inputs are connected to an appropriate logic voltage level (for example, either $V_{SS}$ or $V_{DD}$).*

## 28.1.2 Functional Operating Range

| Rating | Symbol | Value | Unit |
|---|---|---|---|
| Operating Temperature Range(1) | $T_A$ | –40 to $T_A$(MAX) | °C |
| Operating Voltage Range | $V_{DD}$ | 5.0 ± 0.5 | V |

1. $T_A$(MAX) = 125°C for part suffix MFU/MFN
   $T_A$(MAX) = 105°C for part suffix VFU/VFN
   $T_A$(MAX) = 85°C for part suffix CFU/CFN

***NOTE***

*For applications which use the LVI, Freescale guarantees the functionality of the device down to the LVI trip point ($V_{LVI}$) within the constraints outlined in Chapter 16 Low-Voltage Inhibit (LVI).*

## 28.1.3 Thermal Characteristics

| Characteristic | Symbol | Value | Unit |
|---|---|---|---|
| Thermal Resistance QFP (64 Pins) | $\theta_{JA}$ | 70 | °C/W |
| Thermal Resistance PLCC (52 Pins) | $\theta_{JA}$ | 50 | °C/W |
| I/O Pin Power Dissipation | $P_{I/O}$ | User Determined | W |
| Power Dissipation (see Note 1) | $P_D$ | $P_D = (I_{DD} \times V_{DD}) + P_{I/O} = K/(T_J + 273\ °C)$ | W |
| Constant (see Note 2) | K | $P_D \times (T_A + 273\ °C) + (P_D^2 \times \theta_{JA})$ | W/°C |
| Average Junction Temperature | $T_J$ | $T_A + P_D \times \theta_{JA}$ | °C |

1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined from a known $T_A$ and measured $P_D$. With this value of K, $P_D$ and $T_J$ can be determined for any value of $T_A$.

## 28.1.4 5.0 Volt DC Electrical Characteristics

| Characteristic[1] | Symbol | Min | Typical | Max | Unit |
|---|---|---|---|---|---|
| Output High Voltage<br>($I_{LOAD}$ = –2.0 mA) All Ports<br>($I_{LOAD}$ = –5.0 mA) All Ports | $V_{OH}$ | <br>$V_{DD}$ –0.8<br>$V_{DD}$ –1.5 | <br>—<br>— | <br>—<br>— | V |
| Total source current | $I_{OH}$(TOT) | — | — | 10 | mA |
| Output Low Voltage<br>($I_{LOAD}$ = 1.6 mA) All Ports<br>($I_{LOAD}$ = 10.0 mA) All Ports | $V_{OL}$ | <br>—<br>— | <br>—<br>— | <br>0.4<br>1.5 | V |
| Total sink current | $I_{OL}$(TOT) | — | — | 15 | mA |
| Input High Voltage<br>All Ports, $\overline{IRQ}$s, $\overline{RST}$, OSC1 | $V_{IH}$ | 0.7 x $V_{DD}$ | — | $V_{DD}$ | V |
| Input Low Voltage<br>All Ports, $\overline{IRQ}$s, $\overline{RST}$, OSC1 | $V_{IL}$ | $V_{SS}$ | — | 0.3 x $V_{DD}$ | V |
| $V_{DD}$ Supply Current<br>Run[2]<br>Wait [3]<br>Stop[4]<br>LVI enabled, $T_A$ = 25°C<br>LVI disabled, $T_A$ = 25°C<br>LVI enabled, –40°C to +125°C<br>LVI disabled, –40°C to +125°C | $I_{DD}$[5] | <br>—<br>—<br><br>—<br>—<br>—<br>— | <br>25<br>14<br><br>100<br>35 | <br>35<br>20<br><br>400<br>50<br>500<br>100 | <br>mA<br>mA<br><br>µA<br>µA<br>µA<br>µA |
| I/O Ports Hi-Z Leakage Current | $I_L$ | –1 | — | 1 | µA |
| Input Current | $I_{IN}$ | –1 | — | 1 | µA |
| Capacitance<br>Ports (As Input or Output) | $C_{OUT}$<br>$C_{IN}$ | —<br>— | — | 12<br>8 | pF |
| Low-Voltage Reset Inhibit (trip)<br>(recover) | $V_{LVI}$ | 3.80<br>— | —<br>— | —<br>4.49 | V |
| POR ReArm Voltage[6] | $V_{POR}$ | 0 | — | 200 | mV |
| POR Reset Voltage[7] | $V_{PORRST}$ | 0 | — | 800 | mV |
| POR Rise Time Ramp Rate[8] | $R_{POR}$ | 0.02 | — | — | V/ms |
| High COP Disable Voltage[9] | $V_{HI}$ | $V_{DD}$ + 3.0 | — | $V_{DD}$ + 4.5 | V |
| Monitor mode entry voltage on $\overline{IRQ}$[10] | $V_{HI}$ | $V_{DD}$ + 3.0 | — | $V_{DD}$ + 4.5 | V |
| Pull resistor (KBD[4:0]) | $R_{PU}$ | — | 100 | — | kΩ |

1. $V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A$ = –40°C to +$T_A$(MAX), unless otherwise noted.
2. Run (Operating) $I_{DD}$ measured using external square wave clock source ($f_{BUS}$ = 8.4 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. $C_L$ = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run $I_{DD}$. Measured with all modules enabled. Typical values at midpoint of voltage range, 25°C only.
3. Wait $I_{DD}$ measured using external square wave clock source ($f_{BUS}$ = 8.4 MHz). All inputs 0.2 Vdc from rail. No dc loads. Less than 100 pF on all outputs, $C_L$ = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait $I_{DD}$. Measured with all modules enabled. Typical values at midpoint of voltage range, 25°C only.
4. Stop $I_{DD}$ measured with OSC1 = $V_{SS}$.
5. Although $I_{DD}$ is proportional to bus frequency, a current of several mA is present even at very low frequencies.
6. Maximum is highest voltage that POR is guaranteed.
7. Maximum is highest voltage that POR is possible.
8. If minimum $V_{DD}$ is not reached before the internal POR reset is released, $\overline{RST}$ must be driven low externally until minimum $V_{DD}$ is reached.
9. See 15.8 COP Module During Break Interrupts. $V_{HI}$ applied to $\overline{RST}$.
10. See Monitor mode description within Chapter 15 Computer Operating Properly (COP). $V_{HI}$ applied to $\overline{IRQ}$ or $\overline{RST}$

### 28.1.5 Control Timing

| Characteristic[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Bus Operating Frequency (4.5–5.5 V — $V_{DD}$ Only) | $f_{BUS}$ | — | 8.4 | MHz |
| $\overline{RST}$ Pulse Width Low | $t_{RL}$ | 1.5 | — | $t_{cyc}$ |
| $\overline{IRQ}$ Interrupt Pulse Width Low (Edge-Triggered) | $t_{ILHI}$ | 1.5 | — | $t_{cyc}$ |
| $\overline{IRQ}$ Interrupt Pulse Period | $t_{ILIL}$ | Note 4 | — | $t_{cyc}$ |
| 16-Bit Timer[2]<br>Input Capture Pulse Width[3]<br>Input Capture Period | <br>$t_{TH}$, $t_{TL}$<br>$t_{TLTL}$ | <br>2<br>Note[4] | <br>—<br>— | $t_{cyc}$ |
| MSCAN Wake-up Filter Pulse Width[5] | $t_{WUP}$ | 2 | 5 | µs |

1. $V_{DD}$ = 5.0 Vdc ± 0.5v, $V_{SS}$ = 0 Vdc, $T_A$ = –40 °C to $T_A$(MAX), unless otherwise noted.
2. The 2-bit timer prescaler is the limiting factor in determining timer resolution.
3. Refer to Table 25-2. Mode, Edge, and Level Selection and supporting note.
4. The minimum period $t_{TLTL}$ or $t_{ILIL}$ should not be less than the number of cycles it takes to execute the capture interrupt service routine plus TBD $t_{cyc}$.
5. The minimum pulse width to wake up the MSCAN module is guaranteed by design but not tested.

### 28.1.6 ADC Characteristics

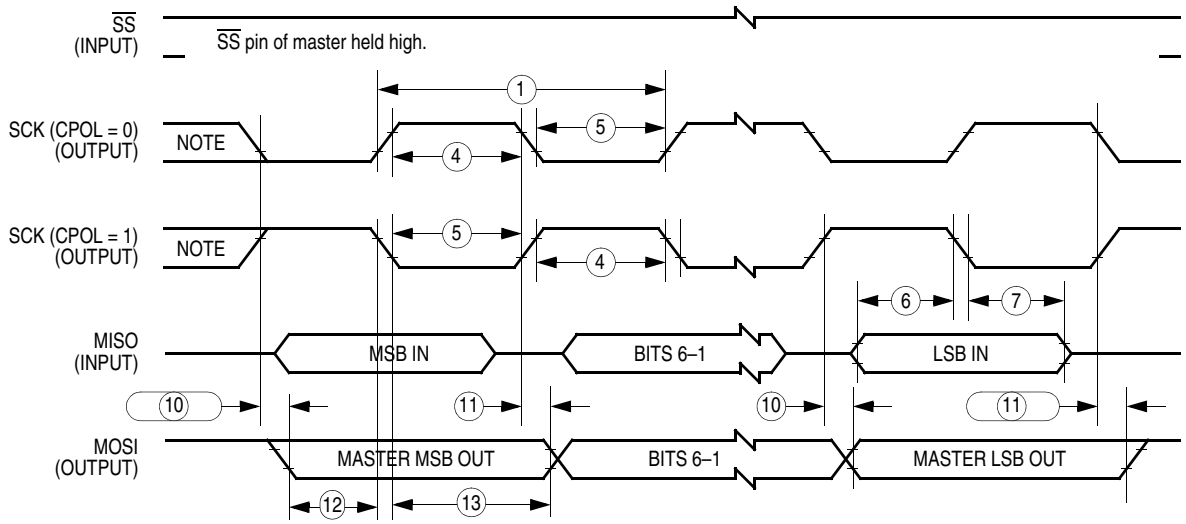
| Characteristic[1] | Min | Max | Unit | Comments |
|---|---|---|---|---|
| Resolution | 8 | 8 | Bits | |
| Absolute Accuracy<br>($V_{REFL}$ = 0 V, $V_{DDA}/V_{DDAREF}$ = $V_{REFH}$ = 5 V ± 0.5 V) | –1 | +1 | LSB | Includes Quantization |
| Conversion Range[2] | $V_{REFL}$ | $V_{REFH}$ | V | $V_{REFL}$ = $V_{SSA}$ |
| Power-Up Time | 16 | 17 | µs | Conversion Time Period |
| Input Leakage[3] (Ports B and D) | –1 | 1 | µA | |
| Conversion Time | 16 | 17 | ADC Clock Cycles | Includes Sampling Time |
| Monotonicity | Inherent within Total Error | | | |
| Zero Input Reading | 00 | 01 | Hex | $V_{IN}$ = $V_{REFL}$ |
| Full-Scale Reading | FE | FF | Hex | $V_{IN}$ = $V_{REFH}$ |
| Sample Time[2] | 5 | — | ADC Clock Cycles | |
| Input Capacitance | — | 8 | pF | Not Tested |
| ADC Internal Clock | 500 k | 1.048 M | Hz | Tested Only at 1 MHz |
| Analog Input Voltage | $V_{REFL}$ | $V_{REFH}$ | V | |

1. $V_{DD}$ = 5.0 Vdc ± 0.5 V, $V_{SS}$ = 0 Vdc, $V_{DDA}/V_{DDAREF}$ = 5.0 Vdc ± 0.5 V, $V_{SSA}$ = 0 Vdc, $V_{REFH}$ = 5.0 Vdc ± 0.5 V
2. Source impedances greater than 10 kΩ adversely affect internal RC charging time during input sampling.
3. The external system error caused by input leakage current is approximately equal to the product of R source and input current.

### 28.1.7 5.0 Vdc ± 0.5 V Serial Peripheral Interface (SPI) Timing

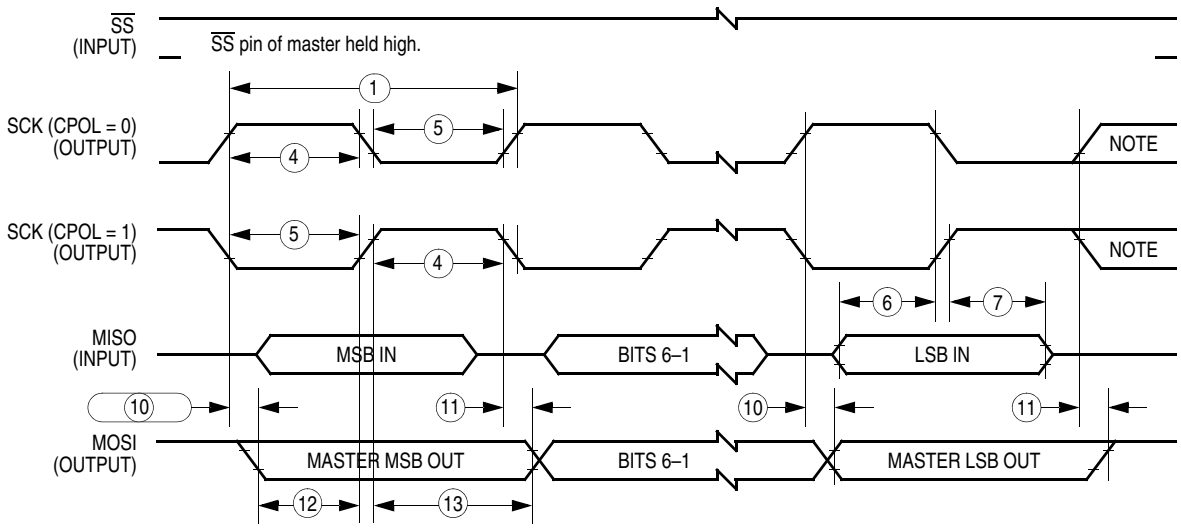
| Num(1) | Characteristic(2) | Symbol | Min | Max | Unit |
|---|---|---|---|---|---|
| | Operating Frequency(3)<br>Master<br>Slave | $f_{BUS(M)}$<br>$f_{BUS(S)}$ | $f_{BUS}/128$<br>dc | $f_{BUS}/2$<br>$f_{BUS}$ | MHz |
| 1 | Cycle Time<br>Master<br>Slave | $t_{cyc(M)}$<br>$t_{cyc(S)}$ | 2<br>1 | 128<br>— | $t_{cyc}$ |
| 2 | Enable Lead Time | $t_{Lead}$ | 15 | — | ns |
| 3 | Enable Lag Time | $t_{Lag}$ | 15 | — | ns |
| 4 | Clock (SCK) High Time<br>Master<br>Slave | $t_{W(SCKH)M}$<br>$t_{W(SCKH)S}$ | 100<br>50 | —<br>— | ns |
| 5 | Clock (SCK) Low Time<br>Master<br>Slave | $t_{W(SCKL)M}$<br>$t_{W(SCKL)S}$ | 100<br>50 | —<br>— | ns |
| 6 | Data Setup Time (Inputs)<br>Master<br>Slave | $t_{SU(M)}$<br>$t_{SU(S)}$ | 45<br>5 | —<br>— | ns |
| 7 | Data Hold Time (Inputs)<br>Master<br>Slave | $t_{H(M)}$<br>$t_{H(S)}$ | 0<br>15 | —<br>— | ns |
| 8 | Access Time, Slave(4)<br>CPHA = 0<br>CPHA = 1 | $t_{A(CP0)}$<br>$t_{A(CP1)}$ | 0<br>0 | 40<br>20 | ns |
| 9 | Slave Disable Time (Hold Time to High-Impedance State) | $t_{DIS}$ | — | 25 | ns |
| 10 | Enable Edge Lead Time to Data Valid(5)<br>Master<br>Slave | $t_{EV(M)}$<br>$t_{EV(S)}$ | —<br>— | 10<br>40 | ns |
| 11 | Data Hold Time (Outputs, after Enable Edge)<br>Master<br>Slave | $t_{HO(M)}$<br>$t_{HO(S)}$ | 0<br>5 | —<br>— | ns |
| 12 | Data Valid<br>Master (Before Capture Edge) | $t_{V(M)}$ | 90 | — | ns |
| 13 | Data Hold Time (Outputs)<br>Master (Before Capture Edge) | $t_{HO(M)}$ | 100 | — | ns |

1. Item numbers refer to dimensions in Figure 28-1 and Figure 28-2.
2. All timing is shown with respect to 30% $V_{DD}$ and 70% $V_{DD}$, unless otherwise noted; assumes 100 pF load on all SPI pins.
3. $f_{BUS}$ = the currently active bus frequency for the microcontroller.
4. Time to data active from high-impedance state.
5. With 100 pF on all SPI pins.




NOTE: This first clock edge is generated internally, but is not seen at the SCK pin.

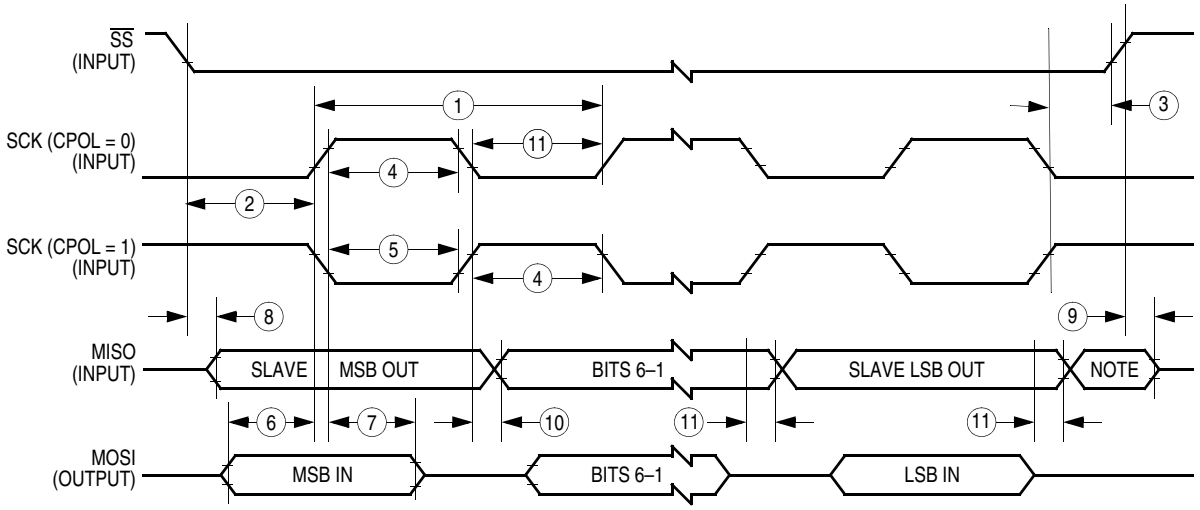
**a) SPI Master Timing (CPHA = 0)**




NOTE: This last clock edge is generated internally, but is not seen at the SCK pin.

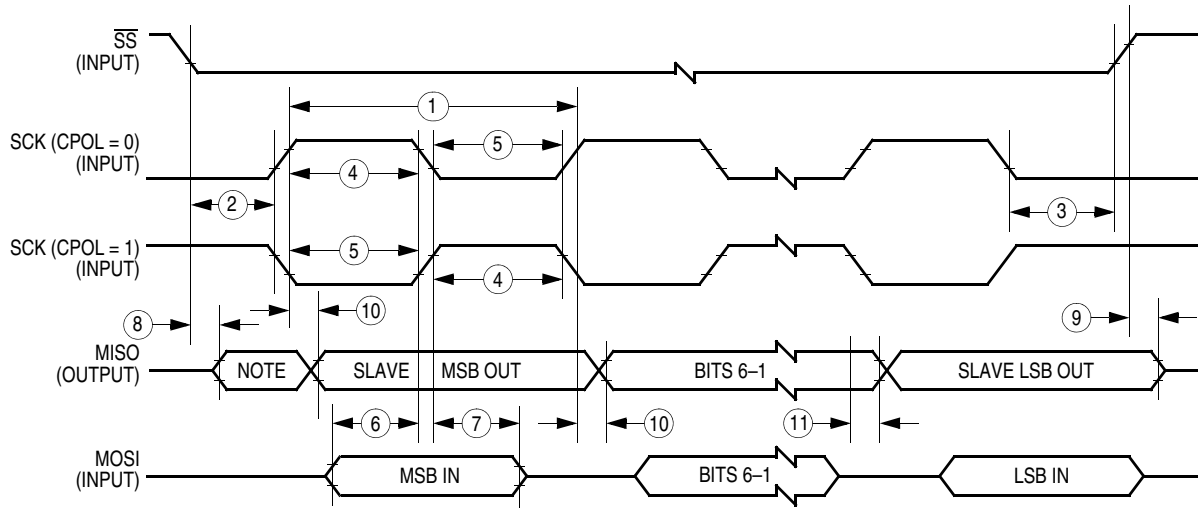
**b) SPI Master Timing (CPHA = 1)**

**Figure 28-1. SPI Master Timing Diagram**




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 28-2. SPI Slave Timing Diagram**

### 28.1.8 CGM Operating Conditions

| Characteristic | Symbol | Min | Typ | Max | Unit | Comments |
|---|---|---|---|---|---|---|
| Operating Voltage | $V_{DDA}$ | $V_{DD}$–0.3 | — | $V_{DD}$+0.3 | V | |
| | $V_{SSA}$ | $V_{SS}$–0.3 | — | $V_{SS}$+0.3 | V | |
| Crystal Reference Frequency | $f_{CGMRCLK}$ | 1 | 4.9152 | 8 | MHz | |
| Module Crystal Reference Frequency | $f_{CGMXCLK}$ | — | 4.9152 | — | MHz | Same Frequency as $f_{CGMRCLK}$ |
| Range Nom. Multiplier | $f_{NOM}$ | — | 4.9152 | — | MHz | |
| VCO Center-of-Range Frequency | $f_{CGMVRS}$ | 4.9152 | — | Note(1) | MHz | |
| VCO Operating Frequency | $f_{CGMVCLK}$ | 4.9152 | — | 32.0 | | |

1. $f_{CGMVRS}$ is a nominal value described and calculated as an example in Chapter 10 Clock Generator Module (CGM) for the desired VCO operating frequency, $f_{CGMVCLK}$.

### 28.1.9 CGM Component Information

| Description | Symbol | Min | Typ | Max | Unit | Comments |
|---|---|---|---|---|---|---|
| Crystal Load Capacitance | $C_L$ | — | — | — | — | Consult Crystal Manufacturer's Data |
| Crystal Fixed Capacitance | C1 | — | 2 x CL | — | — | Consult Crystal Manufacturer's Data |
| Crystal Tuning Capacitance | C2 | — | 2 x CL | — | — | Consult Crystal Manufacturer's Data |
| Filter Capacitor Multiply Factor | $C_{fact}$ | — | 0.0154 | — | F/s V | |
| Filter Capacitor | $C_F$ | — | $C_{FACT}$ x ($V_{DDA}/f_{XCLK}$) | — | — | See 10.4.3 External Filter Capacitor Pin (CGMXFC) |
| Bypass Capacitor | $C_{BYP}$ | — | 0.1 | — | µF | CBYP must provide low AC impedance from f = $f_{CGMXCLK}$/100 to 100 x $f_{CGMVCLK}$, so series resistance must be considered. |

### 28.1.10 CGM Acquisition/Lock Time Information

| Description[1] | Symbol | Min | Typ[2] | Max[2] | Unit | Notes |
|---|---|---|---|---|---|---|
| Manual Mode Time to Stable | $t_{ACQ}$ | — | $(8 \times V_{DDA}) / (f_{CGMXCLK} \times K_{ACQ})$ | — | s | If $C_F$ Chosen Correctly |
| Manual Stable to Lock Time | $t_{AL}$ | — | $(4 \times V_{DDA}) / (f_{CGMXCLK} \times K_{TRK})$ | — | s | If $C_F$ Chosen Correctly |
| Manual Acquisition Time | $t_{LOCK}$ | — | $t_{ACQ}+t_{AL}$ | — | s | |
| Tracking Mode Entry Frequency Tolerance | $D_{TRK}$ | 0 | — | ± 3.6 | % | |
| Acquisition Mode Entry Frequency Tolerance | $D_{UNT}$ | ± 6.3 | — | ± 7.2 | % | |
| LOCK Entry Freq. Tolerance | $D_{LOCK}$ | 0 | — | ± 0.9 | % | |
| LOCK Exit Freq. Tolerance | $D_{UNL}$ | ± 0.9 | — | ± 1.8 | % | |
| Reference Cycles per Acquisition Mode Measurement | $n_{ACQ}$ | — | 32 | — | — | |
| Reference Cycles per Tracking Mode Measurement | $n_{TRK}$ | — | 128 | — | — | |
| Automatic Mode Time to Stable | $t_{ACQ}$ | $n_{ACQ}/f_{XCLK}$ | $(8 \times V_{DDA}) / (f_{XCLK} \times K_{ACQ})$ | | s | If $C_F$ Chosen Correctly |
| Automatic Stable to Lock Time | $t_{AL}$ | $n_{TRK}/f_{XCLK}$ | $(4 \times V_{DDA}) / (f_{XCLK} \times K_{TRK})$ | — | s | If $C_F$ Chosen Correctly |
| Automatic Lock Time | $t_{LOCK}$ | — | 0.65 | 25 | ms | |
| PLL Jitter, Deviation of Average Bus Frequency over 2 ms[3] | | 0 | — | ± ($f_{CRYS}$) x (.025%) x (N/4) | % | N = VCO Freq. Mult. |
| K value for automatic mode time to stable | $K_{acq}$ | — | 0.2 | — | — | |
| K value | $K_{trk}$ | — | 0.004 | — | — | |

1. $V_{DD}$ = 5.0 Vdc ± 0.5 V, $V_{SS}$ = 0 Vdc, $T_A$ = –40°C to $T_A$ (MAX), unless otherwise noted.
2. Conditions for typical and maximum values are for Run mode with $f_{CGMXCLK}$ = 8 MHz, $f_{BUSDES}$ = 8 MHz, N = 4, L = 7, discharged $C_F$ = 15 nF, $V_{DD}$ = 5 Vdc.
3. Guaranteed by not tested. Refer to Chapter 10 Clock Generator Module (CGM) for guidance on the use of the PLL.

### 28.1.11 Timer Module Characteristics

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| Input Capture Pulse Width | $t_{TIH}$, $t_{TIL}$ | 125 | — | ns |
| Input Clock Pulse Width | $t_{TCH}$, $t_{TCL}$ | $(1/f_{OP}) + 5$ | — | ns |

### 28.1.12 RAM Memory Characteristics

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| RAM Data Retention Voltage | $V_{RDR}$ | 0.7 | — | V |

### 28.1.13 EEPROM Memory Characteristics

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| EEPROM Programming Time per Byte | $t_{EEPGM}$ | 10 | — | ms |
| EEPROM Erasing Time per Byte | $t_{EEBYTE}$ | 10 | — | ms |
| EEPROM Erasing Time per Block | $t_{EEBLOCK}$ | 10 | — | ms |
| EEPROM Erasing Time per Bulk | $t_{EEBULK}$ | 10 | — | ms |
| EEPROM Programming Voltage Discharge Period | $t_{EEFPV}$ | 100 | — | µs |
| Number of Programming Operations to the Same EEPROM Byte Before Erase[1] | — | — | 8 | — |
| EEPROM Write/Erase Cycles @ 10 ms Write Time | — | 10,000 | — | Cycles |
| EEPROM Data Retention After 10,000 Write/Erase Cycles | — | 10 | — | Years |
| EEPROM Programming Maximum Time to ‘AUTO’ Bit Set | — | — | 500 | µs |
| EEPROM Erasing Maximum Time to ‘AUTO’ Bit Set | — | — | 8 | ms |

1. Programming a byte more times than the specified maximum may affect the data integrity of that byte. The byte must be erased before it can be programmed again.

### 28.1.14 FLASH Memory Characteristics

| Characteristic | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| FLASH Program Bus Clock Frequency | — | 1 | — | MHz |
| FLASH Read Bus Clock Frequency | $f_{READ}$ (1) | 32K | 8.4M | Hz |
| FLASH Page Erase Time | $t_{ERASE}$ (2) | 1 | — | ms |
| FLASH Mass Erase Time | $t_{MERASE}$ (3) | 4 | — | ms |
| FLASH PGM/ERASE to HVEN Set Up Time | $t_{NVS}$ | 10 | — | µs |
| FLASH High Voltage Hold Time | $t_{NVH}$ | 5 | — | µs |
| FLASH High Voltage Hold Time (Mass) | $t_{NVHL}$ | 100 | — | µs |
| FLASH Program Hold Time | $t_{PGS}$ | 5 | — | µs |
| FLASH Program Time | $t_{PROG}$ | 30 | 40 | µs |
| FLASH Return to Read Time | $t_{RCV}$ (4) | 1 | | µs |
| FLASH Cumulative Program HV Period | $t_{HV}$ (5) | — | 4 | ms |
| FLASH Row Erase Endurance (6) | | 10,000 | — | cycles |
| FLASH Row Program Endurance (7) | | 10,000 | — | cycles |
| FLASH 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 $t_{MERASE}$(MIN), there is no erase-disturb, but it reduces the endurance of the FLASH memory.
4. $t_{RCV}$ 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 defined as the cumulative high voltage programming time to the same row before next erase. $t_{HV}$ must satisfy this condition: $t_{NVS}$+ $t_{NVH}$ + $t_{PGS}$ + ($t_{PROG}$X 64) ≥ $t_{HV}$ max.
6. The minimum row erase endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase cycles.
7. The minimum row program endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many program cycles.
8. The FLASH is guaranteed to retain data over the entire operating temperature range for at least the minimum time specified.

## 28.1.15 BDLC Transmitter VPW Symbol Timings

| Characteristic[1], [2] [3] | Number | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Passive Logic 0 | 10 | $t_{TVP1}$ | 62 | 64 | 66 | µs |
| Passive Logic 1 | 11 | $t_{TVP2}$ | 126 | 128 | 130 | µs |
| Active Logic 0 | 12 | $t_{TVA1}$ | 126 | 128 | 130 | µs |
| Active Logic 1 | 13 | $t_{TVA2}$ | 62 | 64 | 66 | µs |
| Start-of-Frame (SOF) | 14 | $t_{TVA3}$ | 198 | 200 | 202 | µs |
| End-of-Data (EOD) | 15 | $t_{TVP3}$ | 198 | 200 | 202 | µs |
| End-of-Frame (EOF) | 16 | $t_{TV4}$ | 278 | 280 | 282 | µs |
| Inter-Frame Separator (IFS) | 17 | $t_{TV6}$ | 298 | 300 | — | µs |

1. $f_{BDLC}$ = 1.048576 or 1.0 MHz, $V_{DD}$ = 5.0 V ± 10%, $V_{SS}$ = 0 V
2. See Figure 28-3.
3. Transmit timing dependent upon BARD register matching physical transceiver timing.

## 28.1.16 BDLC Receiver VPW Symbol Timings

| Characteristic[1], [2], [3] | Number | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| Passive Logic 0 | 10 | $t_{TRVP1}$ | 34 | 64 | 96 | µs |
| Passive Logic 1 | 11 | $t_{TRVP2}$ | 96 | 128 | 163 | µs |
| Active Logic 0 | 12 | $t_{TRVA1}$ | 96 | 128 | 163 | µs |
| Active Logic 1 | 13 | $t_{TRVA2}$ | 34 | 64 | 96 | µs |
| Start-of-Frame (SOF) | 14 | $t_{TRVA3}$ | 163 | 200 | 239 | µs |
| End-of-Data (EOD) | 15 | $t_{TRVP3}$ | 163 | 200 | 239 | µs |
| End-of-Frame (EOF) | 16 | $t_{TRV4}$ | 239 | 280 | 320 | µs |
| Break | 18 | $t_{TRV6}$ | 280 | — | — | µs |

1. $f_{BDLC}$ = 1.048576 or 1.0 MHz, $V_{DD}$ = 5.0 V ± 10%, $V_{SS}$ = 0 V
2. The receiver symbol timing boundaries are subject to an uncertainty of 1 $t_{BDLC}$ µs due to sampling considerations.
3. See Figure 28-3.

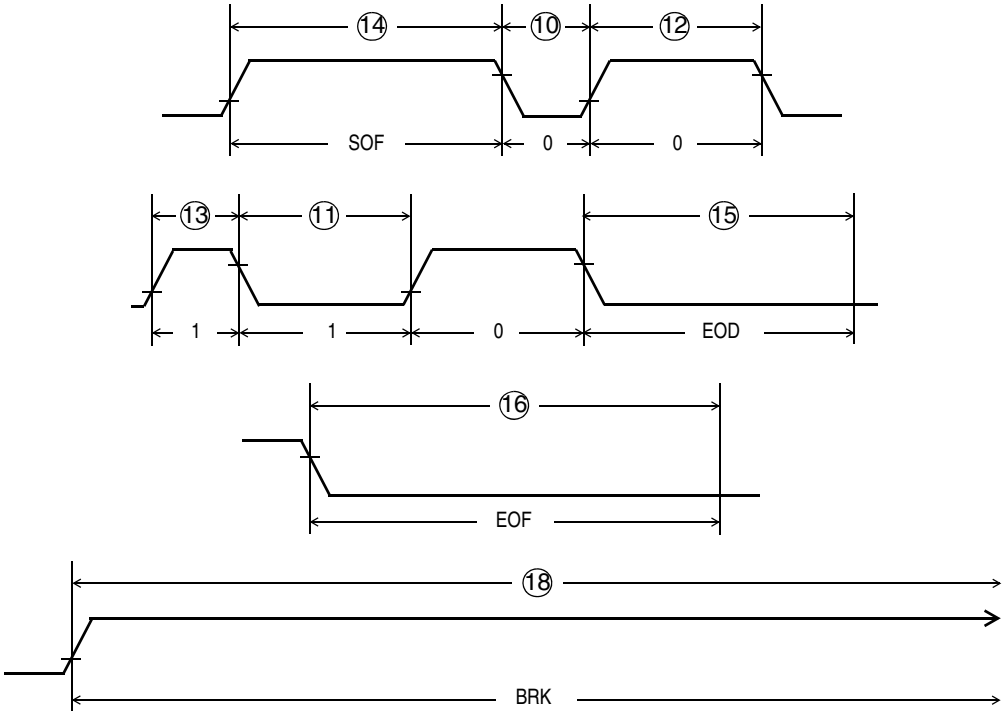



**Figure 28-3. BDLC Variable Pulse Width Modulation (VPW) Symbol Timing**

## 28.1.17 BDLC Transmitter DC Electrical Characteristics

| Characteristic[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| BDTxD Output Low Voltage (IBDTxD = 1.6 mA) | $V_{OLTX}$ | — | 0.4 | V |
| BDTxD Output High Voltage (IBDTx = –800 µA) | $V_{OHTX}$ | $V_{DD}$ –0.8 | — | V |

1. $V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A$ = –40 °C to +125 °C, unless otherwise noted

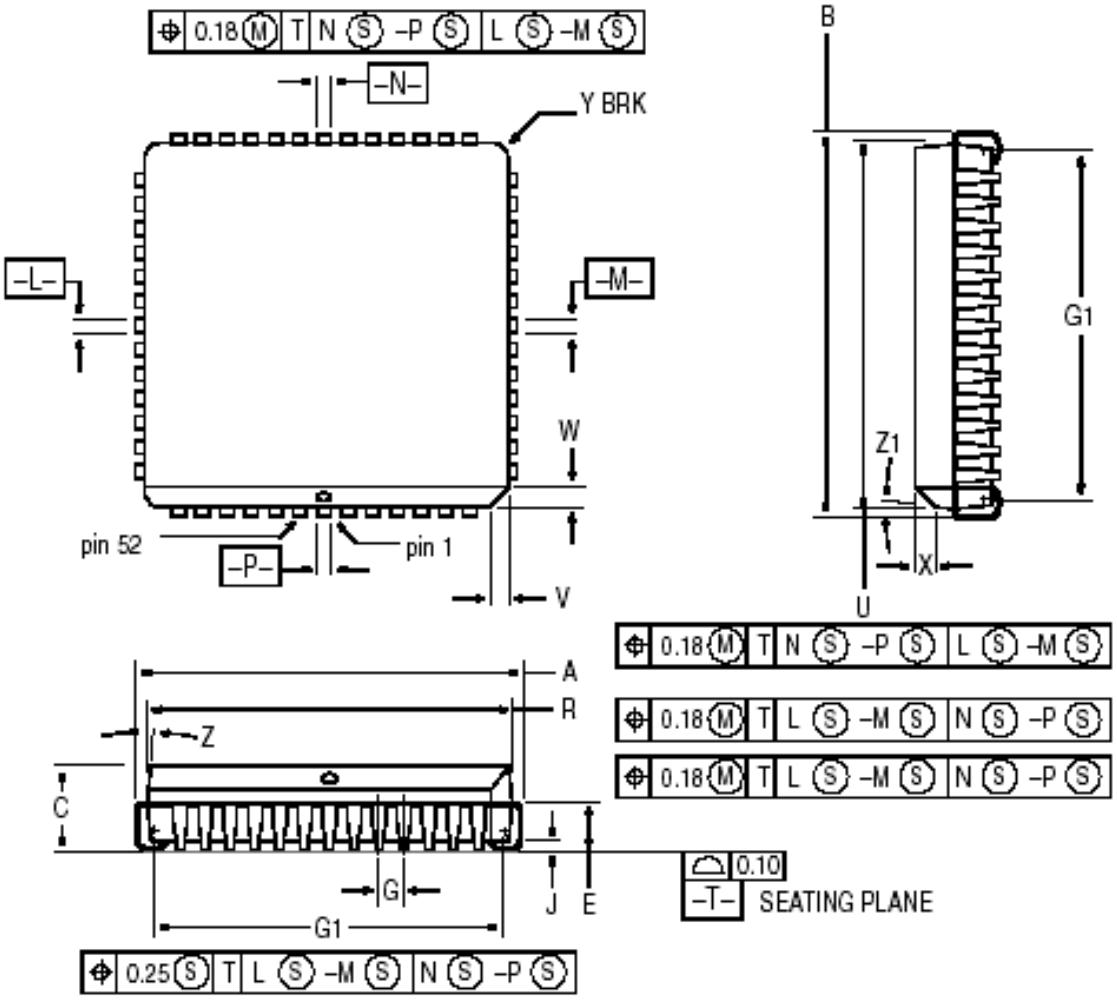
## 28.1.18 BDLC Receiver DC Electrical Characteristics

| Characteristic[1] | Symbol | Min | Max | Unit |
|---|---|---|---|---|
| BDRxD Input Low Voltage | $V_{ILRX}$ | $V_{SS}$ | 0.3 x $V_{DD}$ | V |
| BDRxD Input High Voltage | $V_{IHRX}$ | 0.7 x $V_{DD}$ | $V_{DD}$ | V |
| BDRxD Input Low Current | $I_{ILBDRXI}$ | –1 | +1 | µA |
| BDRxD Input High Current | $I_{HBDRX}$ | –1 | +1 | µA |

1. $V_{DD}$ = 5.0 Vdc ± 10%, $V_{SS}$ = 0 Vdc, $T_A$ = –40 °C to +125 °C, unless otherwise noted

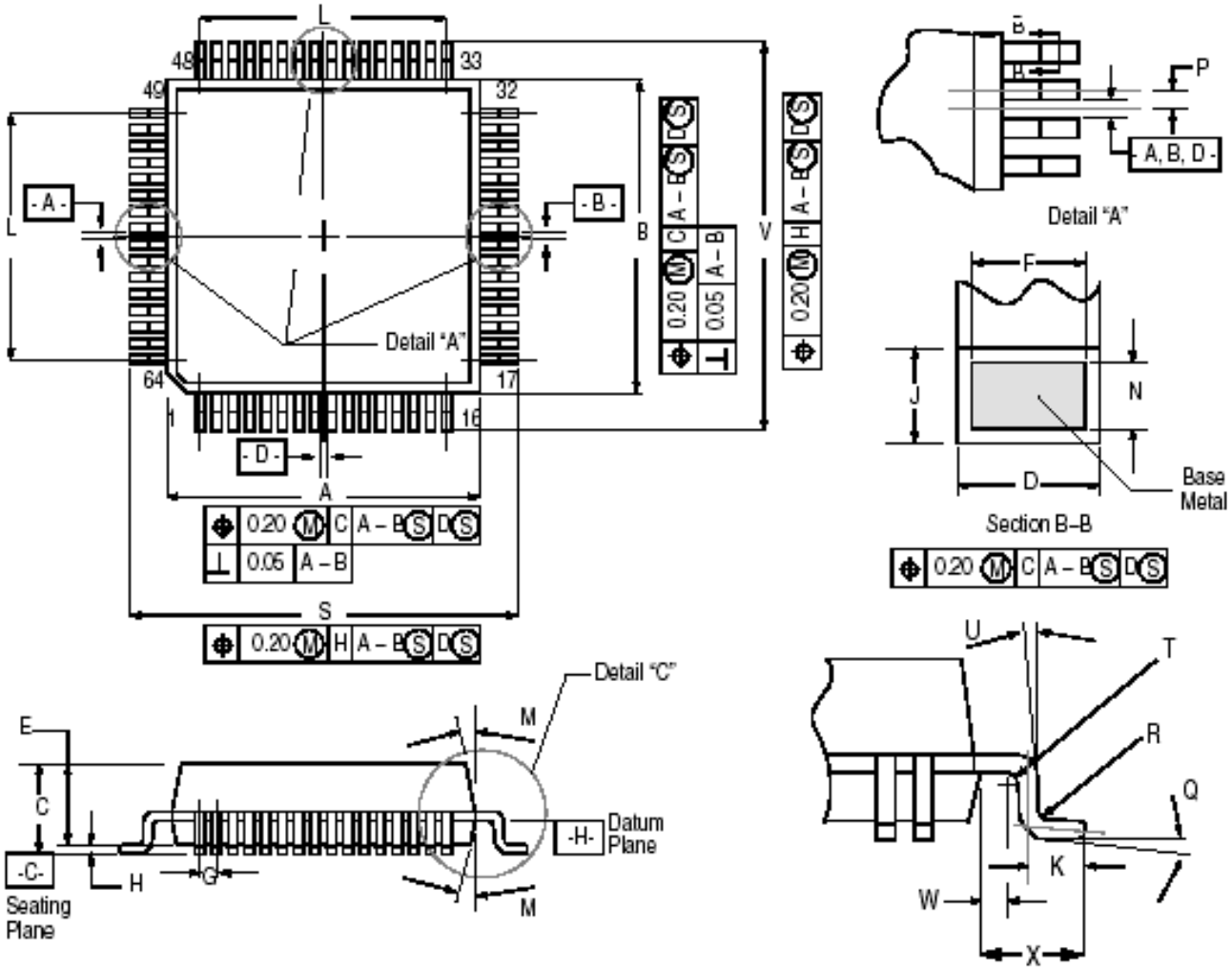
## 28.2 Mechanical Specifications

### 28.2.1 51-Pin Plastic Leaded Chip Carrier (PLCC)




| Dim. | Min. | Max. | Notes | Dim. | Min. | Max. |
|---|---|---|---|---|---|---|
| A | 19.94 | 20.19 | | U | 19.05 | 19.20 |
| B | 19.94 | 20.19 | | V | 1.07 | 1.21 |
| C | 4.20 | 4.57 | 1. Datums –L–, –M–, –N– and –P– are determined where top of lead shoulder exits plastic body at mould parting line. | W | 1.07 | 1.21 |
| E | 2.29 | 2.79 | 2. Dimension G1, true position to be measured at datum –T– (seating plane). | X | 1.07 | 1.42 |
| F | 0.33 | 0.48 | | Y | — | 0.50 |
| G | 1.27 BSC | | 3. Dimensions R and U do not include mould protrusion. Allowable mould protrusion is 0.25mm per side. | Z | 2° | 10° |
| H | 0.66 | 0.81 | 4. Dimensions and tolerancing per ANSI Y 14.5M, 1982. | G1 | 18.04 | 18.54 |
| J | 0.51 | — | 5. All dimensions in mm. | K1 | 1.02 | — |
| K | 0.64 | — | | Z1 | 2° | 10° |
| R | 19.05 | 19.20 | | | | |

## 28.2.2 64-Pin Quad Flat Pack (QFP)




| Dim. | Min. | Max. |
|---|---|---|
| A | 13.90 | 14.10 |
| B | 13.90 | 14.10 |
| C | 2.15 | 2.45 |
| D | 0.30 | 0.45 |
| E | 2.00 | 2.40 |
| F | 0.30 | 0.40 |
| G | 0.80 BSC | |
| H | — | 0.25 |
| J | 0.13 | 0.23 |
| K | 0.65 | 0.95 |
| L | 12.00 REF | |

Notes:

1. 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.
2. Datums A–B and –D to be determined at Datum Plane –H–.
3. Dimensions S and V to be determined at seating plane –C–.
4. Dimensions A and B do not include mould protrusion. Allowable mould protrusion is 0.25mm per side. Dimensions A and B do include mould mismatch and are determined at Datum Plane –H–.
5. Dimension D does not include dambar protrusion. Allowable dambar protrusion shall be 0.08 total in excess of the D dimension at maximum material condition. Dambar cannot be located on the lower radius or the foot.
6. Dimensions and tolerancing per ANSI Y 14.5M, 1982.
7. All dimensions in mm.

| Dim. | Min. | Max. |
|---|---|---|
| M | 5° | 10° |
| N | 0.13 | 0.17 |
| P | 0.40 BSC | |
| Q | 0° | 7° |
| R | 0.13 | 0.30 |
| S | 16.95 | 17.45 |
| T | 0.13 | — |
| U | 0° | — |
| V | 16.95 | 17.45 |
| W | 0.35 | 0.45 |
| X | 1.6 REF | |

# Appendix A
# MC68HC908AS60 and MC68HC908AZ60

## A.1 Changes from the MC68HC908AS60 and MC68HC908AZ60 (non-A suffix devices)

### A.1.1 Specification

Specifications for MC68HC908AS60A and MC68HC908AZ60A devices have been integrated, reflecting the many commonalties.

### A.1.2 FLASH

#### *A.1.2.1 FLASH Architecture*

FLASH-1 and FLASH-2 are made from a new nonvolatile memory (NVM) technology. The architecture is now arranged in pages of 128 bytes and 2 rows per page. Programming is now carried out on a whole row (64 bytes) at a time. Erasing is now carried out on a whole page (128 bytes) at a time. In this new technology an erased bit now reads as a logic 1 and a programmed bit now reads as a logic 0.

#### *A.1.2.2 FLASH Control Registers*

FLASH-1 control register is moved from $FE0B to $FF88. FLASH-2 control register is moved from $FE11 to $FE08. Bits 4 to 7 in the FLASH control registers are no longer used since clock control is now achieved automatically and erasing of variable block sizes is no longer a feature. Bit 2 of the FLASH control registers no longer activates a so-called ‘margin read’ operation but instead is the bit that controls a mass (bulk) erase operation.

#### *A.1.2.3 FLASH Programming Procedure*

Programming of the FLASH is largely as before within the new architecture constraints outlined above. However, an extra dummy write operation to any address in the page is required prior to programming data into one of the two rows in the page. Margin reading of programmed data is no longer required. Nor is read / verify / re-pulse of the programming a requirement.

#### *A.1.2.4 FLASH Programming Time*

The most significant change resulting from the new FLASH technology is that the byte programming time is reduced to a maximum of 40us. This represents several orders of magnitude improvement from the previous technology.

#### *A.1.2.5 FLASH Block Protection*

The FLASH block protect registers are now 8-bit registers in place of 4-bit protecting array ranges that can be incremented by as little as 1 page (128 bytes) at a time as opposed to 8 Kbytes at a time on previous MCUs. Users making use of the block protect feature must change their block protect register.

A further significant change is that high voltage ($V_{HI}$) is no longer needed on the $\overline{IRQ}$ pin to program or erase the FLASH block protect registers.

#### *A.1.2.6 FLASH Endurance*

The FLASH endurance is now specified as 10,000 write / erase cycles as opposed to less than 1000 before.

### A.1.3 EEPROM

#### *A.1.3.1 EEPROM Architecture*

Like the FLASH, EEPROM-1 and EEPROM-2 are also made from a new NVM technology. However, unlike the FLASH, the bit polarity remains the same i.e. programmed=0, erased=1. The architecture and basic programming and erase operations are unchanged.

#### *A.1.3.2 EEPROM Clock Source and Prescaler*

The first major difference on the new EEPROM is that it requires a constant time base source to ensure secure programming and erase operations. This is done by firstly selecting which clock source is going to drive the EEDIVG clock divider input using a new bit 7 introduced onto the CONFIG-2 register $FE09. Next the divide ratio from this source has to be set by programming an 11-bit time base pre-scalar into bits spread over two new registers, EEDIVxH and EEDIVxL (where x=1 or 2 for EEPROM-1 or EEPROM-2 arrays).

The EEDIVxH and EEDIVxL registers are volatile. However, they are loaded upon reset by the contents of duplicate nonvolatile EEDIVxHNVR and EEDIVxLNVR registers much in the same way as the array control registers (EEACRx) interact with the nonvolatile registers (EENVRx) for configuration control on the existing revision. As a result of the new EEDIV clock described above bit 7 (EEBCLK) of the EEPROM control registers (EECRx) is no longer used.

#### *A.1.3.3 EEPROM AUTO Programming & Erasing*

The second major change to the EEPROM is the inclusion in the EEPROM control registers (EECRx) of an AUTO function using previously unused bit 1 of these registers.

The AUTO function enables the logic of the MCU to automatically use the optimum programming or erasing time for the EEPROM. If using AUTO the user does not need to wait for the normal minimum specified programming or erasing time. After setting the EEPGM bit as normal the user just has to poll that bit again, waiting for the MCU to clear it indicating that programming or erasing is complete.

### A.1.4 CONFIG-2

CONFIG-2 register $FE09 has 2 new bits activated. Bit 3 is now a silicon hard set bit, which identifies this new A-suffix silicon (1) from the previous non-A suffix silicon (0). Bit 7 is now an EEPROM time base divider clock select bit selecting the reference clock source for the EEPROM time base divider module (refer to EEPROM changes described above).

### A.1.5 Keyboard Interrupt

The keyboard module is now a feature of the MC68HC908AS60A in 64-qfp package whereas previously it was only a feature of the AZ device. Vector addresses $FFD2 and $FFD3 are now in the AS memory map in support of this option.

### A.1.6 Current Consumption

Current consumption will be significantly lower in many applications. Although maximum specifications are still very dependent upon fabrication process variation and configuration of the MCU in the target application, additional values have been added to the $I_{DD}$ specifications to provide typical current consumption data. Please see Chapter 28 Electrical Specifications for further details.

### A.1.7 Illegal Address Reset

Only an opcode fetch from an illegal address will generate an illegal address reset. Data fetches from unmapped addresses will not generate a reset.

### A.1.8 Monitor Mode Entry and COP Disable Voltage

The monitor mode entry and COP disable voltage specifications ($V_{HI}$) have been increased. Please see Chapter 28 Electrical Specifications for details.

### A.1.9 Low-Voltage Inhibit (LVI)

The Low-Voltage Inhibit (LVI) specifications for trip and recovery voltage ($V_{LVI}$) have been altered based upon module performance on silicon. Please see for Chapter 28 Electrical Specifications details.

# Appendix B
# MC68HC908AZ60E

## B.1 Introduction

The MC68HC908AZ60E is a reduced EMC version of the MC68HC908AZ60A. Every care has been taken to insure compatibility with the MC68HC908AZ60A. Some additional features are available, however the default state of all affected modules match the MC68HC908AZ60A functionality. The reset state of all MC68HC908AZ60E registers match the MC68HC908AZ60A except for some reserved memory locations. Although significant design changes have been made to improve the radiated RF emissions from the MCU, all electrical specifications are equal to or better than the MC68HC908AZ60A.

Slew rate controlled outputs have been added to all the general purpose I/O pins as well as the PTC2/MCLK, PTE5/MISO, PTE6/MOSI, and PTE7/SPSCK pins.

**Table B-1. External Pins Summary (Sheet 1 of 3)**

| Pin Name | Function | Driver Type | Hysteresis[1] | Reset State |
|---|---|---|---|---|
| PTA7–PTA0 | General-Purpose I/O | Dual State | No | Input Hi-Z |
| PTB7/ATD7–PTB0/ATD0 | General-Purpose I/O<br>ADC Channels | Dual State | No | Input Hi-Z |
| PTC5–PTC3 | General-Purpose I/O | Dual State | No | Input Hi-Z |
| PTC2/MCLK | General-Purpose I/O<br>MCLK output | Dual State | No | Input Hi-Z |
| PTC1–PTC0 | General-Purpose I/O | Dual State | No | Input Hi-Z |
| PTD7 | General Purpose I/O | Dual State | No | Input Hi-Z |
| PTD6/ATD14/TACLK ADC Channel | General-Purpose I/O<br>ADC Channel/Timer<br>External Input Clock | Dual State | Yes, TACLK | Input Hi-Z |
| PTD5/ATD13 ADC Channel | General-Purpose I/O<br>ADC Channel | Dual State | No | Input Hi-Z |
| PTD4/ATD12/TBCLK ADC Channel | General-Purpose I/O<br>ADC Channel/Timer<br>External Input Clock | Dual State | Yes, TBCLK | Input Hi-Z |
| PTD3/ATD11–PTD0/ATD8 ADC Channels | General-Purpose I/O<br>ADC Channel | Dual State | No | Input Hi-Z |
| PTE7/SPSCK | General-Purpose I/O<br>SPI Clock | Dual State<br>Open Drain | Yes, SPSCK | Input Hi-Z |
| PTE6/MOSI | General-Purpose I/O<br>SPI Data Path | Dual State<br>Open Drain | Yes, MOSI | Input Hi-Z |
| PTE5/MISO | General-Purpose I/O<br>SPI Data Path | Dual State<br>Open Drain | Yes, MISO | Input Hi-Z |

## Table B-1. External Pins Summary (Sheet 2 of 3)

| Pin Name | Function | Driver Type | Hysteresis[1] | Reset State |
|---|---|---|---|---|
| PTE4/$\overline{SS}$ | General-Purpose I/O<br>SPI Slave Select | Dual State | Yes, $\overline{SS}$ | Input Hi-Z |
| PTE3/TACH1 | General-Purpose I/O<br>Timer A Channel 1 | Dual State | Yes, TACH1 | Input Hi-Z |
| PTE2/TACH0 | General-Purpose I/O<br>Timer A Channel 0 | Dual State | Yes, TACH0 | Input Hi-Z |
| PTE1/RxD | General-Purpose I/O<br>SCI Receive Data | Dual State | Yes, RxD | Input Hi-Z |
| PTE0/TxD | General-Purpose I/O<br>SCI Transmit Data | Dual State | No | Input Hi-Z |
| PTF6 | General-Purpose I/O | Dual State | No | Input Hi-Z |
| PTF5/TBCH1 | General-Purpose I/O<br>Timer B Channel 1 | Dual State | Yes, TBCH1 | Input Hi-Z |
| PTF4/TBCH0 | General-Purpose I/O<br>Timer B Channel 0 | Dual State | Yes, TBCH0 | Input Hi-Z |
| PTF3/TACH5 | General-Purpose I/O<br>Timer A Channel 5 | Dual State | Yes, TACH5 | Input Hi-Z |
| PTF2/TACH4 | General-Purpose I/O<br>Timer A Channel 4 | Dual State | Yes, TACH4 | Input Hi-Z |
| PTF1/TACH3 | General-Purpose I/O<br>Timer A Channel 3 | Dual State | Yes, TACH3 | Input Hi-Z |
| PTF0/TACH2 | General-Purpose I/O<br>Timer A Channel 2 | Dual State | Yes, TACH2 | Input Hi-Z |
| PTG2/KBD2–PTG0/KBD0 | General-Purpose I/O<br>Keyboard Wakeup Pin | Dual State | Yes, KBD | Input Hi-Z |
| PTH1/KBD4–PTH0/KBD3 | General-Purpose I/O<br>Keyboard Wakeup Pin | Dual State | Yes, KBD | Input Hi-Z |
| $V_{DD}$ | Chip Power Supply | N/A | N/A | N/A |
| $V_{SS}$ | Chip Ground | N/A | N/A | N/A |
| $V_{DDA}$ | CGM Analog Power Supply | N/A | N/A | N/A |
| $V_{SSA}$ | CGM Analog Ground | N/A | N/A | N/A |
| $V_{DDAREF}$ | ADC Power Supply | N/A | N/A | N/A |
| $A_{VSS}/V_{REFL}$ | ADC Ground/ADC Reference Low Voltage | N/A | N/A | N/A |
| $V_{REFH}$ | ADC Reference High Voltage | N/A | N/A | N/A |
| OSC1 | External Clock In | N/A | N/A | Input Hi-Z |
| OSC2 | External Clock Out | N/A | N/A | Output |
| CGMXFC | PLL Loop Filter Cap | N/A | N/A | N/A |

**Table B-1. External Pins Summary (Sheet 3 of 3)**

| Pin Name | Function | Driver Type | Hysteresis[1] | Reset State |
|---|---|---|---|---|
| $\overline{IRQ}$ | External Interrupt Request | N/A | Yes | Input Hi-Z |
| $\overline{RST}$ | External Reset | Open Drain | Yes | Output Low |
| CANRx | CAN Serial Input | N/A | Yes | Input Hi-Z |
| CANTx | CAN Serial Output | Output | No | Output Hi-Z |

1. Hysteresis is not 100% tested but is typically a minimum of 300 mV.

## B.2 Detailed Memory Map Changes (MC68HC908AS60A references have been removed)

Every care has been taken to insure compatibility with the MC68HC908AZ60A; however, the following memory map changes have been made.

$FE0B is now Configuration Register 3 (CONFIG3)

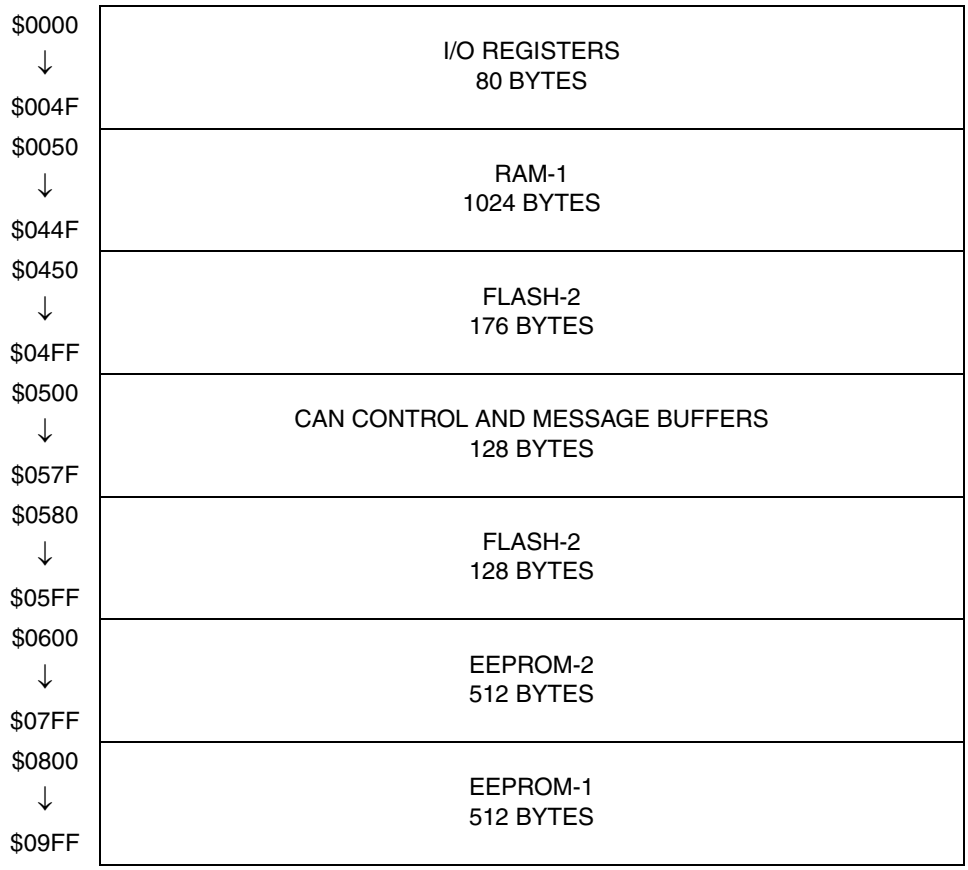



**Figure B-1. MC68HC908AZ60E Memory Map (Sheet 1 of 3)**

| | |
|---|---|
| $0A00<br>↓<br>$0DFF | RAM-2<br>1024 BYTES |
| $0E00<br>↓<br>$7FFF | FLASH-2<br>29,184 BYTES |
| $8000<br>↓<br>$FDFF | FLASH-1<br>32,256 BYTES |
| $FE00 | SIM BREAK STATUS REGISTER (SBSR) |
| $FE01 | SIM RESET STATUS REGISTER (SRSR) |
| $FE02 | RESERVED |
| $FE03 | SIM BREAK FLAG CONTROL REGISTER (SBFCR) |
| $FE04 | RESERVED |
| $FE05 | RESERVED |
| $FE06 | RESERVED |
| $FE07 | RESERVED |
| $FE08 | FLASH-2 CONTROL REGISTER (FL2CR) |
| $FE09 | CONFIGURATION WRITE-ONCE REGISTER (CONFIG-2) |
| $FE0A | RESERVED |
| $FE0B | CONFIGURATION WRITE-ONCE REGISTER (CONFIG-3) |
| $FE0C | BREAK ADDRESS REGISTER HIGH (BRKH) |
| $FE0D | BREAK ADDRESS REGISTER LOW (BRKL) |
| $FE0E | BREAK STATUS AND CONTROL REGISTER (BSCR) |
| $FE0F | LVI STATUS REGISTER (LVISR) |
| $FE10 | EEPROM-1EEDIVH NONVOLATILE REGISTER (EE1DIVHNVR) |
| $FE11 | EEPROM-1EEDIVL NONVOLATILE REGISTER (EE1DIVLNVR) |
| $FE12 | RESERVED |
| $FE13 | RESERVED |
| $FE14 | RESERVED |
| $FE15 | RESERVED |
| $FE16 | RESERVED |
| $FE17 | RESERVED |
| $FE18 | RESERVED |
| $FE19 | RESERVED |
| $FE1A | EEPROM-1 EE DIVIDER HIGH REGISTER(EE1DIVH) |
| $FE1B | EEPROM-1 EE DIVIDER LOW REGISTER(EE1DIVL) |
| $FE1C | EEPROM-1 EEPROM NONVOLATILE REGISTER (EE1NVR) |
| $FE1D | EEPROM-1 EEPROM CONTROL REGISTER (EE1CR) |
| $FE1E | RESERVED |
| $FE1F | EEPROM-1 EEPROM ARRAY CONFIGURATION REGISTER (EE1ACR) |

**Figure B-1. MC68HC908AZ60E Memory Map (Sheet 2 of 3)**

| | |
|---|---|
| $FE20<br>↓<br>$FF1F | MONITOR ROM<br>256BYTES |
| $FF20<br>↓<br>$FF6F | UNIMPLEMENTED<br>80 BYTES |
| $FF70 | EEPROM-2 EEDIVH NONVOLATILE REGISTER (EE2DIVHNVR) |
| $FF71 | EEPROM-2 EEDIVL NONVOLATILE REGISTER (EE2DIVLNVR) |
| $FF72 | RESERVED |
| $FF73 | RESERVED |
| $FF74 | RESERVED |
| $FF75 | RESERVED |
| $FF76 | RESERVED |
| $FF77 | RESERVED |
| $FF78 | RESERVED |
| $FF79 | RESERVED |
| $FF7A | EEPROM-2 EE DIVIDER HIGH REGISTER (EE2DIVH) |
| $FF7B | EEPROM-2 EE DIVIDER LOW REGISTER (EE2DIVL) |
| $FF7C | EEPROM-2 EEPROM NONVOLATILE REGISTER (EE2NVR) |
| $FF7D | EEPROM-2 EEPROM CONTROL REGISTER (EE2CR) |
| $FF7E | RESERVED |
| $FF7F | EEPROM-2 EEPROM ARRAY CONFIGURATION REGISTER (EE2ACR) |
| $FF80 | FLASH-1 BLOCK PROTECT REGISTER (FL1BPR) |
| $FF81 | FLASH-2 BLOCK PROTECT REGISTER (FL2BPR) |
| $FF82<br>↓<br>$FF87 | RESERVED<br>6 BYTES |
| $FF88 | FLASH-1 CONTROL REGISTER (FL1CR) |
| $FF89 | RESERVED |
| $FF8A | RESERVED |
| $FF8B<br>↓<br>$FFCB | RESERVED<br>64 BYTES |
| $FFCC<br>↓<br>$FFFF | VECTORS<br>52 BYTES<br>See Table B-2 |

**Figure B-1. MC68HC908AZ60E Memory Map (Sheet 3 of 3)**

## B.3 I/O Section

Addresses $0000–$004F, shown in Figure B-2, contain the I/O Data, Status, and Control Registers.

Differences from the MC68HC908AZ60A are shown in bold text.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0000 | Port A Data Register (PTA) | Read: | PTA7 | PTA6 | PTA5 | PTA4 | PTA3 | PTA2 | PTA1 | PTA0 |
| | | Write: | | | | | | | | |
| $0001 | Port B Data Register (PTB) | Read: | PTB7 | PTB6 | PTB5 | PTB4 | PTB3 | PTB2 | PTB1 | PTB0 |
| | | Write: | | | | | | | | |
| $0002 | Port C Data Register (PTC) | Read: | 0 | 0 | PTC5 | PTC4 | PTC3 | PTC2 | PTC1 | PTC0 |
| | | Write: | | | | | | | | |
| $0003 | Port D Data Register (PTD) | Read: | PTD7 | PTD6 | PTD5 | PTD4 | PTD3 | PTD2 | PTD1 | PTD0 |
| | | Write: | | | | | | | | |
| $0004 | Data Direction Register A (DDRA) | Read: | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| | | Write: | | | | | | | | |
| $0005 | Data Direction Register B (DDRB) | Read: | DDRB7 | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
| | | Write: | | | | | | | | |
| $0006 | Data Direction Register C (DDRC) | Read: | MCLKEN | 0 | DDRC5 | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
| | | Write: | | | | | | | | |
| $0007 | Data Direction Register D (DDRD) | Read: | DDRD7 | DDRD6 | DDRD5 | DDRD4 | DDRD3 | DDR2 | DDRD1 | DDRD0 |
| | | Write: | | | | | | | | |
| $0008 | Port E Data Register (PTE) | Read: | PTE7 | PTE6 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 |
| | | Write: | | | | | | | | |
| $0009 | Port F Data Register (PTF) | Read: | 0 | PTF6 | PTF5 | PTF4 | PTF3 | PTF2 | PTF1 | PTF0 |
| | | Write: | | | | | | | | |
| $000A | Port G Data Register (PTG) | Read: | 0 | 0 | 0 | 0 | 0 | PTG2 | PTG1 | PTG0 |
| | | Write: | | | | | | | | |
| $000B | Port H Data Register (PTH) | Read: | 0 | 0 | 0 | 0 | 0 | 0 | PTH1 | PTH0 |
| | | Write: | | | | | | | | |
| $000C | Data Direction Register E (DDRE) | Read: | DDRE7 | DDRE6 | DDRE5 | DDRE4 | DDRE3 | DDRE2 | DDRE1 | DDRE0 |
| | | Write: | | | | | | | | |
| $000D | Data Direction Register F (DDRF) | Read: | 0 | DDRF6 | DDRF5 | DDRF4 | DDRF3 | DDRF2 | DDRF1 | DDRF0 |
| | | Write: | | | | | | | | |
| $000E | Data Direction Register G (DDRG) | Read: | 0 | 0 | 0 | 0 | 0 | DDRG2 | DDRG1 | DDRG0 |
| | | Write: | | | | | | | | |
| $000F | Data Direction Register H (DDRH) | Read: | 0 | 0 | 0 | 0 | 0 | 0 | DDRH1 | DDRH0 |
| | | Write: | | | | | | | | |
| $0010 | SPI Control Register (SPCR) | Read: | SPRIE | R | SPMSTR | CPOL | CPHA | SPWOM | SPE | SPTIE |
| | | Write: | | | | | | | | |
| $0011 | SPI Status and Control Register (SPSCR) | Read: | SPRF | ERRIE | OVRF | MODF | SPTE | MODFEN | SPR1 | SPR0 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure B-2. I/O Data, Status and Control Registers (Sheet 1 of 4)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0012 | SPI Data Register (SPDR) | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| $0013 | SCI Control Register 1 (SCC1) | Read: | LOOPS | ENSCI | TXINV | M | WAKE | ILTY | PEN | PTY |
| | | Write: | | | | | | | | |
| $0014 | SCI Control Register 2 (SCC2) | Read: | SCTIE | TCIE | SCRIE | ILIE | TE | RE | RWU | SBK |
| | | Write: | | | | | | | | |
| $0015 | SCI Control Register 3 (SCC3) | Read: | R8 | T8 | R | R | ORIE | NEIE | FEIE | PEIE |
| | | Write: | | | | | | | | |
| $0016 | SCI Status Register 1 (SCS1) | Read: | SCTE | TC | SCRF | IDLE | OR | NF | FE | PE |
| | | Write: | | | | | | | | |
| $0017 | SCI Status Register 2 (SCS2) | Read: | 0 | 0 | 0 | 0 | 0 | 0 | BKF | RPF |
| | | Write: | | | | | | | | |
| $0018 | SCI Data Register (SCDR) | Read: | R7 | R6 | R5 | R4 | R3 | R2 | R1 | R0 |
| | | Write: | T7 | T6 | T5 | T4 | T3 | T2 | T1 | T0 |
| $0019 | SCI Baud Rate Register (SCBR) | Read: | 0 | 0 | SCP1 | SCP0 | R | SCR2 | SCR1 | SCR0 |
| | | Write: | | | | | | | | |
| $001A | IRQ Status and Control Register (ISCR) | Read: | 0 | 0 | 0 | 0 | IRQF | 0 | IMASK | MODE |
| | | Write: | | | | | R | ACK | | |
| $001B | Keyboard Status and Control Register (KBSCR) | Read: | 0 | 0 | 0 | 0 | KEYF | 0 | IMASKK | MODEK |
| | | Write: | | | | | | ACKK | | |
| $001C | PLL Control Register (PCTL) | Read: | PLLIE | PLLF | PLLON | BCS | 1 | 1 | 1 | 1 |
| | | Write: | | | | | | | | |
| $001D | PLL Bandwidth Control Register (PBWC) | Read: | AUTO | LOCK | $\overline{\text{ACQ}}$ | XLD | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| $001E | PLL Programming Register (PPG) | Read: | MUL7 | MUL6 | MUL5 | MUL4 | VRS7 | VRS6 | VRS5 | VRS4 |
| | | Write: | | | | | | | | |
| $001F | Configuration Write-Once Register (CONFIG-1) | Read: | LVISTOP | R | LVIRST | LVIPWR | SSREC | COPL | STOP | COPD |
| | | Write: | | | | | | | | |
| $0020 | Timer A Status and Control Register (TASC) | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | R | | | |
| $0021 | Keyboard Interrupt Enable Register (KBIER) | Read: | 0 | 0 | 0 | KBIE4 | KBIE3 | KBIE2 | KBIE1 | KBIE0 |
| | | Write: | | | | | | | | |
| $0022 | Timer A Counter Register High (TACNTH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0023 | Timer A Counter Register Low (TACNTL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0024 | Timer A Modulo Register High (TAMODH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure B-2. I/O Data, Status and Control Registers (Sheet 2 of 4)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0025 | Timer A Modulo Register Low (TAMODL) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0026 | Timer A Channel 0 Status and Control Register (TASC0) | Read:<br>Write: | CH0F<br>0 | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| $0027 | Timer A Channel 0 Register High (TACH0H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0028 | Timer A Channel 0 Register Low (TACH0L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0029 | Timer A Channel 1 Status and Control Register (TASC1) | Read:<br>Write: | CH1F<br>0 | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| $002A | Timer A Channel 1 Register High (TACH1H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $002B | Timer A Channel 1 Register Low (TACH1L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $002C | Timer A Channel 2 Status and Control Register (TASC2) | Read:<br>Write: | CH2F<br>0 | CH2IE | MS2B | MS2A | ELS2B | ELS2A | TOV2 | CH2MAX |
| $002D | Timer A Channel 2 Register High (TACH2H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $002E | Timer A Channel 2 Register Low (TACH2L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $002F | Timer A Channel 3 Status and Control Register (TASC3) | Read:<br>Write: | CH3F<br>0 | CH3IE | 0 | MS3A | ELS3B | ELS3A | TOV3 | CH3MAX |
| $0030 | Timer A Channel 3 Register High (TACH3H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0031 | Timer A Channel 3 Register Low (TACH3L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0032 | Timer A Channel 4 Status and Control Register (TASC4) | Read:<br>Write: | CH4F<br>0 | CH4IE | MS4B | MS4A | ELS4B | ELS4A | TOV4 | CH4MAX |
| $0033 | Timer A Channel 4 Register High (TACH4H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0034 | Timer A Channel 4 Register Low (TACH4L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| $0035 | Timer A Channel 5 Status and Control Register (TASC5) | Read:<br>Write: | CH5F<br>0 | CH5IE | 0 | MS5A | ELS5B | ELS5A | TOV5 | CH5MAX |
| $0036 | Timer A Channel 5 Register High (TACH5H) | Read:<br>Write: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| $0037 | Timer A Channel 5 Register Low (TACH5L) | Read:<br>Write: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| **$0038** | **Analog-to-Digital Status and Control Register (ADSCR)** | Read:<br>Write: | **COCO** | AIEN | ADCO | ADCH4 | ADCH3 | ADCH2 | ADCH1 | ADCH0 |

= Unimplemented  R = Reserved

**Figure B-2. I/O Data, Status and Control Registers (Sheet 3 of 4)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $0039 | Analog-to-Digital Data Register (ADR) | Read: | AD7 | AD6 | AD5 | AD4 | AD3 | AD2 | AD1 | AD0 |
| | | Write: | | | | | | | | |
| $003A | Analog-to-Digital Input Clock Register (ADICLK) | Read: | ADIV2 | ADIV1 | ADIV0 | ADICLK | 0 | 0 | 0 | 0 |
| | | Write: | | | | | R | R | R | R |
| $0040 | Timer B Status and Control Register (TBSCR) | Read: | TOF | TOIE | TSTOP | 0 | 0 | PS2 | PS1 | PS0 |
| | | Write: | 0 | | | TRST | R | | | |
| $0041 | Timer B Counter Register High (TBCNTH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0042 | Timer B Counter Register Low (TBCNTL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0043 | Timer B Modulo Register High (TBMODH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0044 | Timer B Modulo Register Low (TBMODL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0045 | Timer B CH0 Status and Control Register (TBSC0) | Read: | CH0F | CH0IE | MS0B | MS0A | ELS0B | ELS0A | TOV0 | CH0MAX |
| | | Write: | 0 | | | | | | | |
| $0046 | Timer B CH0 Register High (TBCH0H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $0047 | Timer B CH0 Register Low (TBCH0L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $0048 | Timer B CH1 Status and Control Register (TBSC1) | Read: | CH1F | CH1IE | 0 | MS1A | ELS1B | ELS1A | TOV1 | CH1MAX |
| | | Write: | 0 | | | | | | | |
| $0049 | Timer B CH1 Register High (TBCH1H) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $004A | Timer B CH1 Register Low (TBCH1L) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $004B | PIT Status and Control Register (PSC) | Read: | POF | POIE | PSTOP | 0 | 0 | PPS2 | PPS1 | PPS0 |
| | | Write: | 0 | | | PRST | | | | |
| $004C | PIT Counter Register High (PCNTH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $004D | PIT Counter Register Low (PCNTL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $004E | PIT Modulo Register High (PMODH) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $004F | PIT Modulo Register Low (PMODL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure B-2. I/O Data, Status and Control Registers (Sheet 4 of 4)**

## B.4 Additional Status and Control Registers

Selected addresses in the range $FE00 to $FF88 contain additional Status and Control registers as shown in Figure B-3. A noted exception is the COP Control Register (COPCTL) at address $FFFF.

Differences from the MC68HC908AZ60A are shown in bold text.

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FE00 | SIM Break Status Register (SBSR) | Read: | R | R | R | R | R | R | BW | R |
| | | Write: | | | | | | | 0 | |
| $FE01 | SIM Reset Status Register (SRSR) | Read: | POR | PIN | COP | ILOP | ILAD | 0 | LVI | 0 |
| | | Write: | | | | | | | | |
| $FE03 | SIM Break Flag Control Register (SBFCR) | Read: | BCFE | R | R | R | R | R | R | R |
| | | Write: | | | | | | | | |
| $FE08 | FLASH-2 Control Register (FL2CR) | Read: | 0 | 0 | 0 | 0 | HVEN | VERF | ERASE | PGM |
| | | Write: | | | | | | | | |
| $FE09 | Configuration Write-Once Register (CONFIG-2) | Read: | EEDIV-CLK | R | R | MSCAND | AT60A | R | R | AZxx |
| | | Write: | | | | | R | | | |
| **$FE0B** | **Configuration Write-Once Register (CONFIG-3)** | Read: | **R** | **R** | **R** | **R** | **R** | **R** | **SPISRD** | **R** |
| | | Write: | | | | | | | | |
| $FE0C | Break Address Register Low (BRKL) | Read: | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | Write: | | | | | | | | |
| $FE0D | Break Address Register Low (BRKL) | Read: | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | Write: | | | | | | | | |
| $FE0E | Break Status and Control Register (BRKSCR) | Read: | BRKE | BRKA | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| $FE0F | LVI Status Register (LVISR) | Read: | LVIOUT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | Write: | | | | | | | | |
| $FE10 | EE1DIV Hi Nonvolatile Register (EE1DIVHNVR) | Read: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| $FE11 | EE1DIV Lo Nonvolatile Register (EE1DIVLNVR) | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| $FE1A | EE1DIV Divider High Register (EE1DIVH) | Read: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| $FE1B | EE1DIV Divider Low Register (EE1DIVL) | Read: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| | | Write: | | | | | | | | |
| $FE1C | EEPROM-1 Nonvolatile Register (EE1NVR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| $FE1D | EEPROM-1 Control Register (EE1CR) | Read: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| | | Write: | | | | | | | | |
| $FE1F | EEPROM-1 Array Configuration Register (EE1ACR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |

= Unimplemented  R = Reserved

**Figure B-3. Additional Status and Control Registers (Sheet 1 of 2)**

| Addr. | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| $FF70 | EE2DIV Hi Nonvolatile Register (EE2DIVHNVR) | Read: | EEDIVS-ECD | R | R | R | R | EEDIV10 | EEDIV9 | EEDIV8 |
| $FF71 | EE2DIV Lo Nonvolatile Register (EE2DIVLNVR) | Read: Write: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| $FF7A | EE2DIV Divider High Register (EE2DIVH) | Read: | EEDIVS-ECD | 0 | 0 | 0 | 0 | EEDIV10 | EEDIV9 | EEDIV8 |
| | | Write: | | | | | | | | |
| $FF7B | EE2DIV Divider Low Register (EE2DIVL) | Read: Write: | EEDIV7 | EEDIV6 | EEDIV5 | EEDIV4 | EEDIV3 | EEDIV2 | EEDIV1 | EEDIV0 |
| $FE7C | EEPROM-2 Nonvolatile Register (EE2NVR) | Read: Write: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| $FE7D | EEPROM-2 Control Register (EE2CR) | Read: | UNUSED | 0 | EEOFF | EERAS1 | EERAS0 | EELAT | AUTO | EEPGM |
| | | Write: | | | | | | | | |
| $FE7F | EEPROM-2 Array Configuration Register (EE2ACR) | Read: | UNUSED | UNUSED | UNUSED | EEPRTCT | EEBP3 | EEBP2 | EEBP1 | EEBP0 |
| | | Write: | | | | | | | | |
| $FF80 | FLASH-1 Block Protect Register (FL1BPR) | Read: Write: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| $FF81 | FLASH-2 Block Protect Register (FL2BPR) | Read: Write: | BPR7 | BPR6 | BPR5 | BPR4 | BPR3 | BPR2 | BPR1 | BPR0 |
| $FF88 | FLASH-1 Control Register (FL1CR) | Read: | 0 | 0 | 0 | 0 | HVEN | VERF | ERASE | PGM |
| | | Write: | | | | | | | | |
| $FFFF | COP Control Register (COPCTL) | Read: | LOW BYTE OF RESET VECTOR | | | | | | | |
| | | Write: | WRITING TO $FFFF CLEARS COP COUNTER | | | | | | | |

= Unimplemented R = Reserved

**Figure B-3. Additional Status and Control Registers (Sheet 2 of 2)**

## B.5 Vector Addresses and Priority

Addresses in the range $FFCC to $FFFF contain the user-specified vector locations. The vector addresses are shown in Table B-2.

| | Address | Vector<br>MC68HC908AZ60E |
|---|---|---|
| Lowest Priority | $FFCC | TIMA Channel 5 Vector (High) |
| | $FFCD | TIMA Channel 5 Vector (Low) |
| | $FFCE | TIMA Channel 4 Vector (High) |
| | $FFCF | TIMA Channel 4 Vector (Low) |
| | $FFD0 | ADC Vector (High) |
| | $FFD1 | ADC Vector (Low) |
| | $FFD2 | Keyboard Vector (High) |
| | $FFD3 | Keyboard Vector (Low) |
| | $FFD4 | SCI Transmit Vector (High) |
| | $FFD5 | SCI Transmit Vector (Low) |
| | $FFD6 | SCI Receive Vector (High) |
| | $FFD7 | SCI Receive Vector (Low) |
| | $FFD8 | SCI Error Vector (High) |
| | $FFD9 | SCI Error Vector (Low) |
| | $FFDA | CAN Transmit Vector (High) |
| | $FFDB | CAN Transmit Vector (Low) |
| | $FFDC | CAN Receive Vector (High) |
| | $FFDD | CAN Receive Vector (Low) |
| | $FFDE | CAN Error Vector (High) |
| | $FFDF | CAN Error Vector (Low) |
| | $FFE0 | CAN Wakeup Vector (High) |
| | $FFE1 | CAN Wakeup Vector (Low) |
| | $FFE2 | SPI Transmit Vector (High) |
| | $FFE3 | SPI Transmit Vector (Low) |
| | $FFE4 | SPI Receive Vector (High) |
| | $FFE5 | SPI Receive Vector (Low) |
| | $FFE6 | TIMB Overflow Vector (High) |
| | $FFE7 | TIMB Overflow Vector (Low) |
| | $FFE8 | TIMB CH1 Vector (High) |
| | $FFE9 | TIMB CH1 Vector (Low) |
| | $FFEA | TIMB CH0 Vector (High) |
| | $FFEB | TIMB CH0 Vector (Low) |
| | $FFEC | TIMA Overflow Vector (High) |
| | $FFED | TIMA Overflow Vector (Low) |

**Table B-2. Vector Addresses**

| Address | Vector<br>MC68HC908AZ60E |
|---|---|
| $FFEE | TIMA CH3 Vector (High) |
| $FFEF | TIMA CH3 Vector (Low) |
| $FFF0 | TIMA CH2 Vector (High) |
| $FFF1 | TIMA CH2 Vector (Low) |
| $FFF2 | TIMA CH1 Vector (High) |
| $FFF3 | TIMA CH1 Vector (Low) |
| $FFF4 | TIMA CH0 Vector (High) |
| $FFF5 | TIMA CH0 Vector (Low) |
| $FFF6 | PIT Vector (High) |
| $FFF7 | PIT Vector (Low) |
| $FFF8 | PLL Vector (High) |
| $FFF9 | PLL Vector (Low) |
| $FFFA | IRQ1 Vector (High) |
| $FFFB | IRQ1 Vector (Low) |
| $FFFC | SWI Vector (High) |
| $FFFD | SWI Vector (Low) |
| $FFFE | Reset Vector (High) |
| $FFFF | Reset Vector (Low) |

**Table B-2. Vector Addresses (Continued)**

## B.6 Ordering Information

This section contains instructions for ordering the MC68HC908AZ60E.

### B.6.1 MC Order Numbers

**Table B-3. MC Order Numbers**

| MC Order Number | Operating Temperature Range |
|---|---|
| MC68HC908AZ60ECFU (64-Pin QFP) | –40°C to + 85°C |
| MC68HC908AZ60EVFU (64-Pin QFP) | –40°C to + 105°C |
| MC68HC908AZ60EMFU (64-Pin QFP) | –40°C to + 125°C |

## B.7 Configuration Register (CONFIG-3)

This section describes the configuration register (CONFIG-3). This register is unused on the MC68HC908AZ60A. This register contains one bit that configures the following option:

Disables slew rate control for the SPI pins

The configuration register is a write-once register. Once the register is written, further writes will have no effect until a reset occurs.

Address: $FE0B

| | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| Read:<br>Write: | R | R | R | R | R | R | SPISRD | R |
| Reset: | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

R = Reserved

**Figure B-4. Configuration Register (CONFIG-3)**

**SPISRD — SPI Slew Rate Disable**

This bit disables the slew rate controlled outputs for SCK, MOSI, and MISO pins.

1 = SPI slew rate is disabled
0 = SPI slew rate is enabled

## B.8 SCI

The incorrect operation, signified by the "Note" in the Idle Characters paragraph of the SCI section of this document has been corrected. The following note does not apply to the MC68HC908AZ60E.

***NOTE***

*When a break sequence is followed immediately by an idle character, this SCI design exhibits a condition in which the break character length is reduced by one half bit time. In this instance, the break sequence will consist of a valid start bit, eight or nine data bits (as defined by the M bit in SCC1) of logic 0 and one half data bit length of logic 0 in the stop bit position followed immediately by the idle character. To ensure a break character of the proper length is transmitted, always queue up a byte of data to be transmitted while the final break sequence is in progress.*

## B.9 MSCAN

The MSCAN08 errata on the MC68HC908AZ60A has been fixed on the MC68HC908AZ60E. For 32-bit and 16-bit identifier acceptance modes, an extended ID CAN frame with a stuff bit between ID16 and ID15 will not be rejected. No software work around is required.

## B.10 ADC

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

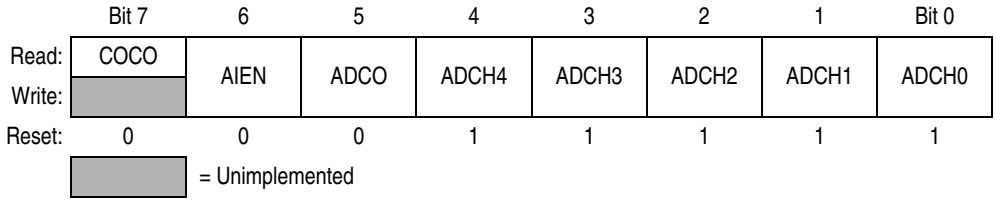



**Figure B-5. ADC10 Status and Control Register (ADCSC)**

**COCO — Conversion Complete Bit**

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

1 = Conversion completed

0 = Conversion not completed

# Revision History

## Major Change Between Revision 5.0 and Revision 6.0

The following table lists the major change between the current revision of the MC68HC908AZ60A Technical Data Book, Rev 6.0, and the previous revision, Rev 5.0.

| Section affected | Description of change |
|---|---|
| Appendix B. MC68HC908AZ60E | Added chapter describing the MC68HC908AZ60E. |

## Major Changes Between Revision 5.0 and Revision 4.0

The following table lists the major changes between the current revision of the MC68HC908AZ60A Technical Data Book, Rev 5.0, and the previous revision, Rev 4.0.

| Section affected | Description of change |
|---|---|
| Throughout | Updated to meet Freescale identity guidelines. |
| Chapter 24 Keyboard Module (KBI) | Addresses for KBSCR and KBIER registers corrected to $001B and $0021 repectively. |
| Chapter 28 Electrical Specifications | Updated values for 28.1.8 CGM Operating Conditions. |

## Major Changes Between Revision 4.0 and Revision 3.0

The following table lists the major changes between the current revision of the MC68HC908AZ60A Technical Data Book, Rev 4.0, and the previous revision, Rev 3.0.

| Section affected | Description of change |
|---|---|
| Electrical Specifications | Updated case outline drawing for 64-Pin Quad Flat Pack (Case 840B) |

## Major Changes Between Revision 3.0 and Revision 2.0

The following table lists the major changes between the current revision of the MC68HC908AZ60A Technical Data Book, Rev 3.0, and the previous revision, Rev 2.0.

| Section affected | Description of change |
|---|---|
| Keyboard Module (KBD) | In Table 24-1, addresses for KBSCR and KBIER registers corrected to $001B and $0021 repectively<br>In first bullet on page 333, vector addresses corrected to $00D2 and $00D3. |

## Major Changes Between Revision 2.0 and Revision 1.0

The following table lists the major changes between the current revision of the MC68HC908AZ60A Technical Data Book, Rev 2.0, and the previous revision, Rev 1.0.

| Section affected | Description of change |
|---|---|
| Timer Interface Module B (TIMB) | Various changes for clarification. |
| Programmable Interrupt Timer (PIT) | |
| Timer Interface Module A (TIMA) | |

## Major Changes Between Revision 1.0 and Revision 0.0

The following table lists the major changes between the current revision of the MC68HC908AZ60A Technical Data Book, Rev 1.0, and the previous revision, Rev 0.0.

| Section affected | Description of change |
|---|---|
| General Description | Highlighted that Keyboard Interrupt Module only available in 64 QFP.<br>Corrected device name in Figure 5 title.<br>Added ADC supply and reference pins to pin descriptions.<br>Corrected text in numerous pin descriptions.<br>Added VDDA and VSSA pins to Table 1-External Pins Summary.<br>Added Table 2-Clock Signal Naming Conventions.<br>Added FLASH and RAM to Table 3-Clock Source Summary.<br>Corrected part numbers in Table 4-MC Order Numbers. |
| Memory Map | Corrected type errors.<br>Corrected various addresses and register names in Figure 1-Memory Map.<br>Corrected numerous register bit descriptions in Figure 2-I/O Data, Status and Control Registers to match module sections.<br>Added Additional Status and Control Registers section and moved register descriptions accordingly. Corrected bit descriptions to match module sections.<br>Added Vector Addresses and Priority section and moved Table 4-Vector Addresses accordingly. |
| FLASH-1 and FLASH-2 | Both sections altered significantly to better align module descriptions across groups within Freescale using 0.5µ TSMC/SST FLASH. Numerous additions submitted by applications engineering for further clarification of functional operation. |
| EEPROM-1 and EEPROM-2 | Both sections altered significantly to better align module descriptions across groups within Freescale using 0.5µ TSMC/SST FLASH. Numerous additions submitted by applications engineering for further clarification of functional operation. |
| Clock Generator Module (CGM) | Corrected clock signal names and associated timing parameters for consistency and to match signal naming conventions.<br>Additional textual description added to Reaction Time Calculation subsection. |
| Configuration Register 2 (CONFIG-2) | Corrected Figure 1-Configuration Register reserved bit descriptions for consistency. |

| Section affected | Description of change |
|---|---|
| Monitor ROM (MON) | Modified Figure 1-Monitor Mode Circuit based upon recommendations from applications engineering.<br>Correct text of Note 1 to Table 2-Mode Differences.<br>Corrected type errors.<br>Corrected text describing state of unprogrammed FLASH in Security subsection.<br>Corrected Figure 6-Monitor Mode Entry Timing. |
| Computer Operating Properly (COP) | Corrected state of COPL bit in Functional Description subsection. |
| Timer Interface Module B (TIMB) | Corrected numerous type and grammatical errors.<br>Corrected numerous pin and register name errors within text.<br>Corrected references to TIMB overflow interrupts (removed "channel x" references as they are incorrect). |
| Programmable Interrupt Timer (PIT) | Corrected type and grammatical errors.<br>Corrected PIT Overflow Interrupt Enable Bit acronym from PIE to POIE. |
| Keyboard Module (KBD) | Corrected addresses of KBSCR and KBIER within text. |
| Timer Interface Module A (TIMA-6) | Corrected numerous type and grammatical errors.<br>Corrected numerous pin and register name errors within text.<br>Corrected references to TIMA overflow interrupts (removed "channel x" references as they are incorrect).<br>Corrected functional description of TOF flag. |
| Electrical Specifications | Corrected type errors.<br>Increased VHI specification in Maximum Ratings to VDD + 4.5V.<br>Corrected formula for Average Junction Temperature in Thermal Characteristics.<br>Added column for typical VDD Supply Current values in 5.0 Volt DC Electrical Characteristics.<br>Decreased LVI trip voltage specification to 3.80V and increased LVI recovery voltage to 4.49V in 5.0 Volt DC Electrical Characteristics.<br>Increased VHI specification to minimum of VDD + 3.0V and maximum of VDD + 4.5V in 5.0 Volt DC Electrical Characteristics.<br>Added Unit columns to all CGM specification tables and adjusted text accordingly.<br>Corrected Operating Voltage specification in CGM Operating Conditions.<br>Added typical specifications for Kacq and Ktrk parameters in CGM Acquisition/Lock Time Information.<br>Split Memory Characteristics table into separate RAM Memory Characteristics, EEPROM Memory Characteristics and FLASH Memory Characteristics tables.<br>Added maximum specification for EEPROM AUTO bit set for each of program and erase operation in EEPROM Memory Characteristics.<br>Corrected NOTES section of FLASH Memory Characteristics.<br>Added Note 3 to BDLC Transmitter VPW Symbol Timings table. |
| Appendix A | Added text describing elimination of need for VHI on IRQ pin to program/erase FLASH block protect registers.<br>Added subsection highlighting change of Monitor Mode entry and COP disable voltage change.<br>Added subsection highlighting change in LVI trip and recovery voltage specifications. |

# Glossary

**A** — See “accumulator (A).”

**accumulator (A)** — An 8-bit general-purpose register in the CPU08. The CPU08 uses the accumulator to hold operands and results of arithmetic and logic operations.

**acquisition mode** — A mode of PLL operation during startup before the PLL locks on a frequency. Also see "tracking mode."

**address bus** — The set of wires that the CPU or DMA uses to read and write memory locations.

**addressing mode** — The way that the CPU determines the operand address for an instruction. The M68HC08 CPU has 16 addressing modes.

**ALU** — See “arithmetic logic unit (ALU).”

**arithmetic logic unit (ALU)** — The portion of the CPU that contains the logic circuitry to perform arithmetic, logic, and manipulation operations on operands.

**asynchronous** — Refers to logic circuits and operations that are not synchronized by a common reference signal.

**baud rate** — The total number of bits transmitted per unit of time.

**BCD** — See “binary-coded decimal (BCD).”

**binary** — Relating to the base 2 number system.

**binary number system** — The base 2 number system, having two digits, 0 and 1. Binary arithmetic is convenient in digital circuit design because digital circuits have two permissible voltage levels, low and high. The binary digits 0 and 1 can be interpreted to correspond to the two digital voltage levels.

**binary-coded decimal (BCD)** — A notation that uses 4-bit binary numbers to represent the 10 decimal digits and that retains the same positional structure of a decimal number. For example,

234 (decimal) = 0010 0011 0100 (BCD)

**bit** — A binary digit. A bit has a value of either logic 0 or logic 1.

**branch instruction** — An instruction that causes the CPU to continue processing at a memory location other than the next sequential address.

**break module** — A module in the M68HC08 Family. The break module allows software to halt program execution at a programmable point in order to enter a background routine.

**breakpoint** — A number written into the break address registers of the break module. When a number appears on the internal address bus that is the same as the number in the break address registers, the CPU executes the software interrupt instruction (SWI).

**break interrupt** — A software interrupt caused by the appearance on the internal address bus of the same value that is written in the break address registers.

**bus** — A set of wires that transfers logic signals.

**bus clock** — The bus clock is derived from the CGMOUT output from the CGM. The bus clock frequency, $f_{op}$, is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

**byte** — A set of eight bits.

**C** — The carry/borrow bit in the condition code register. The CPU08 sets the carry/borrow bit when an addition operation produces a carry out of bit 7 of the accumulator or when a subtraction operation requires a borrow. Some logical operations and data manipulation instructions also clear or set the carry/borrow bit (as in bit test and branch instructions and shifts and rotates).

**CCR** — See “condition code register.”

**central processor unit (CPU)** — The primary functioning unit of any computer system. The CPU controls the execution of instructions.

**CGM** — See “clock generator module (CGM).”

**clear** — To change a bit from logic 1 to logic 0; the opposite of set.

**clock** — A square wave signal used to synchronize events in a computer.

**clock generator module (CGM)** — A module in the M68HC08 Family. The CGM generates a base clock signal from which the system clocks are derived. The CGM may include a crystal oscillator circuit and or phase-locked loop (PLL) circuit.

**comparator** — A device that compares the magnitude of two inputs. A digital comparator defines the equality or relative differences between two binary numbers.

**computer operating properly module (COP)** — A counter module in the M68HC08 Family that resets the MCU if allowed to overflow.

**condition code register (CCR)** — An 8-bit register in the CPU08 that contains the interrupt mask bit and five bits that indicate the results of the instruction just executed.

**control bit** — One bit of a register manipulated by software to control the operation of the module.

**control unit** — One of two major units of the CPU. The control unit contains logic functions that synchronize the machine and direct various operations. The control unit decodes instructions and generates the internal control signals that perform the requested operations. The outputs of the control unit drive the execution unit, which contains the arithmetic logic unit (ALU), CPU registers, and bus interface.

**COP** — See "computer operating properly module (COP)."

**counter clock** — The input clock to the TIM counter. This clock is the output of the TIM prescaler.

**CPU** — See “central processor unit (CPU).”

**CPU08** — The central processor unit of the M68HC08 Family.

**CPU clock** — The CPU clock is derived from the CGMOUT output from the CGM. The CPU clock frequency is equal to the frequency of the oscillator output, CGMXCLK, divided by four.

**CPU cycles** — A CPU cycle is one period of the internal bus clock, normally derived by dividing a crystal oscillator source by two or more so the high and low times will be equal. The length of time required to execute an instruction is measured in CPU clock cycles.

**CPU registers** — Memory locations that are wired directly into the CPU logic instead of being part of the addressable memory map. The CPU always has direct access to the information in these registers. The CPU registers in an M68HC08 are:

- A (8-bit accumulator)
- H:X (16-bit index register)
- SP (16-bit stack pointer)
- PC (16-bit program counter)
- CCR (condition code register containing the V, H, I, N, Z, and C bits)

**CSIC** — customer-specified integrated circuit

**cycle time** — The period of the operating frequency: $t_{CYC} = 1/f_{OP}$.

**decimal number system** — Base 10 numbering system that uses the digits zero through nine.

**direct memory access module (DMA)** — A M68HC08 Family module that can perform data transfers between any two CPU-addressable locations without CPU intervention. For transmitting or receiving blocks of data to or from peripherals, DMA transfers are faster and more code-efficient than CPU interrupts.

**DMA** — See "direct memory access module (DMA)."

**DMA service request** — A signal from a peripheral to the DMA module that enables the DMA module to transfer data.

**duty cycle** — A ratio of the amount of time the signal is on versus the time it is off. Duty cycle is usually represented by a percentage.

**EEPROM** — Electrically erasable, programmable, read-only memory. A nonvolatile type of memory that can be electrically reprogrammed.

**EPROM** — Erasable, programmable, read-only memory. A nonvolatile type of memory that can be erased by exposure to an ultraviolet light source and then reprogrammed.

**exception** — An event such as an interrupt or a reset that stops the sequential execution of the instructions in the main program.

**external interrupt module (IRQ)** — A module in the M68HC08 Family with both dedicated external interrupt pins and port pins that can be enabled as interrupt pins.

**fetch** — To copy data from a memory location into the accumulator.

**firmware** — Instructions and data programmed into nonvolatile memory.

**free-running counter** — A device that counts from zero to a predetermined number, then rolls over to zero and begins counting again.

**full-duplex transmission** — Communication on a channel in which data can be sent and received simultaneously.

**H** — The upper byte of the 16-bit index register (H:X) in the CPU08.

**H** — The half-carry bit in the condition code register of the CPU08. This bit indicates a carry from the low-order four bits of the accumulator value to the high-order four bits. The half-carry bit is required for binary-coded decimal arithmetic operations. The decimal adjust accumulator (DAA) instruction uses the state of the H and C bits to determine the appropriate correction factor.

**hexadecimal** — Base 16 numbering system that uses the digits 0 through 9 and the letters A through F.

**high byte** — The most significant eight bits of a word.

**illegal address** — An address not within the memory map

**illegal opcode** — A nonexistent opcode.

**I** — The interrupt mask bit in the condition code register of the CPU08. When I is set, all interrupts are disabled.

**index register (H:X)** — A 16-bit register in the CPU08. The upper byte of H:X is called H. The lower byte is called X. In the indexed addressing modes, the CPU uses the contents of H:X to determine the effective address of the operand. H:X can also serve as a temporary data storage location.

**input/output (I/O)** — Input/output interfaces between a computer system and the external world. A CPU reads an input to sense the level of an external signal and writes to an output to change the level on an external signal.

**instructions** — Operations that a CPU can perform. Instructions are expressed by programmers as assembly language mnemonics. A CPU interprets an opcode and its associated operand(s) and instruction.

**interrupt** — A temporary break in the sequential execution of a program to respond to signals from peripheral devices by executing a subroutine.

**interrupt request** — A signal from a peripheral to the CPU intended to cause the CPU to execute a subroutine.

**I/O** — See “input/output (I/0).”

**IRQ** — See "external interrupt module (IRQ)."

**jitter** — Short-term signal instability.

**latch** — A circuit that retains the voltage level (logic 1 or logic 0) written to it for as long as power is applied to the circuit.

**latency** — The time lag between instruction completion and data movement.

**least significant bit (LSB)** — The rightmost digit of a binary number.

**logic 1** — A voltage level approximately equal to the input power voltage ($V_{DD}$).

**logic 0** — A voltage level approximately equal to the ground voltage ($V_{SS}$).

**low byte** — The least significant eight bits of a word.

**low voltage inhibit module (LVI)** — A module in the M68HC08 Family that monitors power supply voltage.

**LVI** — See "low voltage inhibit module (LVI)."

**M68HC08** — A Freescale family of 8-bit MCUs.

**mark/space** — The logic 1/logic 0 convention used in formatting data in serial communication.

**mask** — 1. A logic circuit that forces a bit or group of bits to a desired state. 2. A photomask used in integrated circuit fabrication to transfer an image onto silicon.

**mask option** — A optional microcontroller feature that the customer chooses to enable or disable.

**mask option register (MOR)** — An EPROM location containing bits that enable or disable certain MCU features.

**MCU** — Microcontroller unit. See “microcontroller.”

**memory location** — Each M68HC08 memory location holds one byte of data and has a unique address. To store information in a memory location, the CPU places the address of the location on the address bus, the data information on the data bus, and asserts the write signal. To read information from a memory location, the CPU places the address of the location on the address bus and asserts the read signal. In response to the read signal, the selected memory location places its data onto the data bus.

**memory map** — A pictorial representation of all memory locations in a computer system.

**microcontroller** — Microcontroller unit (MCU). A complete computer system, including a CPU, memory, a clock oscillator, and input/output (I/O) on a single integrated circuit.

**modulo counter** — A counter that can be programmed to count to any number from zero to its maximum possible modulus.

**monitor ROM** — A section of ROM that can execute commands from a host computer for testing purposes.

**MOR** — See "mask option register (MOR)."

**most significant bit (MSB)** — The leftmost digit of a binary number.

**multiplexer** — A device that can select one of a number of inputs and pass the logic level of that input on to the output.

**N** — The negative bit in the condition code register of the CPU08. The CPU sets the negative bit when an arithmetic operation, logical operation, or data manipulation produces a negative result.

**nibble** — A set of four bits (half of a byte).

**object code** — The output from an assembler or compiler that is itself executable machine code, or is suitable for processing to produce executable machine code.

**opcode** — A binary code that instructs the CPU to perform an operation.

**open-drain** — An output that has no pullup transistor. An external pullup device can be connected to the power supply to provide the logic 1 output voltage.

**operand** — Data on which an operation is performed. Usually a statement consists of an operator and an operand. For example, the operator may be an add instruction, and the operand may be the quantity to be added.

**oscillator** — A circuit that produces a constant frequency square wave that is used by the computer as a timing and sequencing reference.

**OTPROM** — One-time programmable read-only memory. A nonvolatile type of memory that cannot be reprogrammed.

**overflow** — A quantity that is too large to be contained in one byte or one word.

**page zero** — The first 256 bytes of memory (addresses $0000–$00FF).

**parity** — An error-checking scheme that counts the number of logic 1s in each byte transmitted. In a system that uses odd parity, every byte is expected to have an odd number of logic 1s. In an even parity system, every byte should have an even number of logic 1s. In the transmitter, a parity generator appends an extra bit to each byte to make the number of logic 1s odd for odd parity or even for even parity. A parity checker in the receiver counts the number of logic 1s in each byte. The parity checker generates an error signal if it finds a byte with an incorrect number of logic 1s.

**PC** — See “program counter (PC).”

**peripheral** — A circuit not under direct CPU control.

**phase-locked loop (PLL)** — A oscillator circuit in which the frequency of the oscillator is synchronized to a reference signal.

**PLL** — See "phase-locked loop (PLL)."

**pointer** — Pointer register. An index register is sometimes called a pointer register because its contents are used in the calculation of the address of an operand, and therefore points to the operand.

**polarity** — The two opposite logic levels, logic 1 and logic 0, which correspond to two different voltage levels, $V_{DD}$ and $V_{SS}$.

**polling** — Periodically reading a status bit to monitor the condition of a peripheral device.

**port** — A set of wires for communicating with off-chip devices.

**prescaler** — A circuit that generates an output signal related to the input signal by a fractional scale factor such as 1/2, 1/8, 1/10 etc.

**program** — A set of computer instructions that cause a computer to perform a desired operation or operations.

**program counter (PC)** — A 16-bit register in the CPU08. The PC register holds the address of the next instruction or operand that the CPU will use.

**pull** — An instruction that copies into the accumulator the contents of a stack RAM location. The stack RAM address is in the stack pointer.

**pullup** — A transistor in the output of a logic gate that connects the output to the logic 1 voltage of the power supply.

**pulse-width** — The amount of time a signal is on as opposed to being in its off state.

**pulse-width modulation (PWM)** — Controlled variation (modulation) of the pulse width of a signal with a constant frequency.

**push** — An instruction that copies the contents of the accumulator to the stack RAM. The stack RAM address is in the stack pointer.

**PWM period** — The time required for one complete cycle of a PWM waveform.

**RAM** — Random access memory. All RAM locations can be read or written by the CPU. The contents of a RAM memory location remain valid until the CPU writes a different value or until power is turned off.

**RC circuit** — A circuit consisting of capacitors and resistors having a defined time constant.

**read** — To copy the contents of a memory location to the accumulator.

**register** — A circuit that stores a group of bits.

**reserved memory location** — A memory location that is used only in special factory test modes. Writing to a reserved location has no effect. Reading a reserved location returns an unpredictable value.

**reset** — To force a device to a known condition.

**ROM** — Read-only memory. A type of memory that can be read but cannot be changed (written). The contents of ROM must be specified before manufacturing the MCU.

**SCI** — See "serial communication interface module (SCI)."

**serial** — Pertaining to sequential transmission over a single line.

**serial communications interface module (SCI)** — A module in the M68HC08 Family that supports asynchronous communication.

**serial peripheral interface module (SPI)** — A module in the M68HC08 Family that supports synchronous communication.

**set** — To change a bit from logic 0 to logic 1; opposite of clear.

**shift register** — A chain of circuits that can retain the logic levels (logic 1 or logic 0) written to them and that can shift the logic levels to the right or left through adjacent circuits in the chain.

**signed** — A binary number notation that accommodates both positive and negative numbers. The most significant bit is used to indicate whether the number is positive or negative, normally logic 0 for positive and logic 1 for negative. The other seven bits indicate the magnitude of the number.

**software** — Instructions and data that control the operation of a microcontroller.

**software interrupt (SWI)** — An instruction that causes an interrupt and its associated vector fetch.

**SPI** — See "serial peripheral interface module (SPI)."

**stack** — A portion of RAM reserved for storage of CPU register contents and subroutine return addresses.

**stack pointer (SP)** — A 16-bit register in the CPU08 containing the address of the next available storage location on the stack.

**start bit** — A bit that signals the beginning of an asynchronous serial transmission.

**status bit** — A register bit that indicates the condition of a device.

**stop bit** — A bit that signals the end of an asynchronous serial transmission.

**subroutine** — A sequence of instructions to be used more than once in the course of a program. The last instruction in a subroutine is a return from subroutine (RTS) instruction. At each place in the main program where the subroutine instructions are needed, a jump or branch to subroutine (JSR or BSR) instruction is used to call the subroutine. The CPU leaves the flow of the main program to execute the instructions in the subroutine. When the RTS instruction is executed, the CPU returns to the main program where it left off.

**synchronous** — Refers to logic circuits and operations that are synchronized by a common reference signal.

**TIM** — See "timer interface module (TIM)."

**timer interface module (TIM)** — A module used to relate events in a system to a point in time.

**timer** — A module used to relate events in a system to a point in time.

**toggle** — To change the state of an output from a logic 0 to a logic 1 or from a logic 1 to a logic 0.

**tracking mode** — Mode of low-jitter PLL operation during which the PLL is locked on a frequency. Also see "acquisition mode."

**two's complement** — A means of performing binary subtraction using addition techniques. The most significant bit of a two's complement number indicates the sign of the number (1 indicates negative). The two's complement negative of a number is obtained by inverting each bit in the number and then adding 1 to the result.

**unbuffered** — Utilizes only one register for data; new data overwrites current data.

**unimplemented memory location** — A memory location that is not used. Writing to an unimplemented location has no effect. Reading an unimplemented location returns an unpredictable value. Executing an opcode at an unimplemented location causes an illegal address reset.

**V** —The overflow bit in the condition code register of the CPU08. The CPU08 sets the V bit when a two's complement overflow occurs. The signed branch instructions BGT, BGE, BLE, and BLT use the overflow bit.

**variable** — A value that changes during the course of program execution.

**VCO** — See "voltage-controlled oscillator."

**vector** — A memory location that contains the address of the beginning of a subroutine written to service an interrupt or reset.

**voltage-controlled oscillator (VCO)** — A circuit that produces an oscillating output signal of a frequency that is controlled by a dc voltage applied to a control input.

**waveform** — A graphical representation in which the amplitude of a wave is plotted against time.

**wired-OR** — Connection of circuit outputs so that if any output is high, the connection point is high.

**word** — A set of two bytes (16 bits).

**write** — The transfer of a byte of data from the CPU to a memory location.

**X** — The lower byte of the index register (H:X) in the CPU08.

**Z** — The zero bit in the condition code register of the CPU08. The CPU08 sets the zero bit when an arithmetic operation, logical operation, or data manipulation produces a result of $00.

## *How to Reach Us:*


**Home Page:**
www.freescale.com

**E-mail:**
support@freescale.com

**USA/Europe or Locations Not Listed:**
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

**Europe, Middle East, and Africa:**
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

**Japan:**
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

**Asia/Pacific:**
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

***For Literature Requests Only:***
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com


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

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


Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

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

Freescale™ and the Freescale logo are trademarks of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners.

© Freescale Semiconductor, Inc. 2006. All rights reserved.



MC68HC908AZ60A
Rev. 6, 05/2006


freescale™
semiconductor