

# 68HC08TV24 68HC908TV24 Advance Information

**M68HC08  
Microcontrollers**

**Rev. 2.1  
MC68HC908TV24/D  
August 16, 2005**

*freescale.com*





## List of Sections

|   |     |
|---|-----|
| Section 1. General Description . . . . .                      | 29  |
| Section 2. Memory Map . . . . .                               | 41  |
| Section 3. Low-Power Modes . . . . .                          | 57  |
| Section 4. Resets and Interrupts . . . . .                    | 67  |
| Section 5. Analog-to-Digital Converter (ADC4) . . . . .       | 81  |
| Section 6. Break Module (BRK) . . . . .                       | 87  |
| Section 7. Clock Generator Module (CGMC) . . . . .            | 97  |
| Section 8. Closed-Caption Data Slicer (DSL) . . . . .         | 127 |
| Section 9. Configuration Register (CONFIG) . . . . .          | 143 |
| Section 10. Computer Operating Properly (COP) . . . . .       | 147 |
| Section 11. Central Processor Unit (CPU) . . . . .            | 153 |
| Section 12. User FLASH Memory . . . . .                       | 171 |
| Section 13. On-Screen Display (OSD)<br>FLASH Memory . . . . . | 185 |
| Section 14. External Interrupt (IRQ) . . . . .                | 195 |
| Section 15. Low-Voltage Inhibit (LVI) . . . . .               | 201 |
| Section 16. Monitor ROM (MON) . . . . .                       | 207 |
| Section 17. On-Screen Display Module (OSD) . . . . .          | 221 |

|  |     |
|--|-----|
| Section 18. Input/Output (I/O) Ports . . . . .                     | 267 |
| Section 19. Random-Access Memory (RAM) . . . . .                   | 277 |
| Section 20. System Integration Module (SIM) . . . . .              | 279 |
| Section 21. Serial Synchronous Interface<br>Module (SSI) . . . . . | 305 |
| Section 22. Timebase Module (TBM) . . . . .                        | 319 |
| Section 23. Timer Interface Module (TIM) . . . . .                 | 325 |
| Section 24. ROM Version Overview (ROM) . . . . .                   | 349 |
| Section 25. Preliminary Electrical Specifications . . . . .        | 353 |
| Section 26. Mechanical Specifications . . . . .                    | 363 |
| Section 27. Ordering Information . . . . .                         | 365 |

## Table of Contents

### Section 1. General Description

|        |  |    |
|--------|--|----|
| 1.1    | Contents . . . . .   | 29 |
| 1.2    | Introduction . . . . .   | 30 |
| 1.3    | Features . . . . .   | 30 |
| 1.3.1  | Standard Features . . . . .  | 30 |
| 1.3.2  | Television Application-Specific Features . . . . .                                   | 32 |
| 1.3.3  | CPU08 Features . . . . .   | 33 |
| 1.4    | MCU Block Diagram . . . . .  | 33 |
| 1.5    | Pin Assignments . . . . .  | 35 |
| 1.6    | Pin Functions . . . . .  | 36 |
| 1.6.1  | Power Supply Pins ( $V_{DD}$ and $V_{SS}$ ) . . . . .                                | 36 |
| 1.6.2  | Oscillator Pins (OSC1 and OSC2) . . . . .  | 37 |
| 1.6.3  | External Reset Pin ( $\overline{RST}$ ) . . . . .                                    | 37 |
| 1.6.4  | External Interrupt Pin ( $\overline{IRQ}$ ) . . . . .                                | 37 |
| 1.6.5  | CGM Power Supply Pins ( $V_{DDCGM}$ and $V_{SSCGM}$ ) . . . . .                      | 37 |
| 1.6.6  | External Filter Capacitor Pin (CGMXFC) . . . . .                                     | 37 |
| 1.6.7  | Power Supply Pins for On-Screen Display<br>( $V_{DDOSD}$ and $V_{SSOSD}$ ) . . . . . | 37 |
| 1.6.8  | External Filter Pins for On-Screen Display<br>(OSDVCO and OSDPMP) . . . . .          | 38 |
| 1.6.9  | Synchronism Signals (HSYNC and VSYNC) . . . . .                                      | 38 |
| 1.6.10 | Color Encoded Pixel Signals (R, G, B, and I) . . . . .                               | 38 |
| 1.6.11 | Fast Blanking Signal (FBKG) . . . . .  | 38 |
| 1.6.12 | Serial Synchronous Interface Clocks<br>(SCL1 and SCL2) . . . . .                     | 38 |
| 1.6.13 | Serial Synchronous Interface Data Lines<br>(SDA1 and SDA2) . . . . .                 | 38 |
| 1.6.14 | Video Input (VIDEO) . . . . .  | 39 |
| 1.6.15 | A/D Converter Input (ADCIN) . . . . .  | 39 |
| 1.6.16 | Timer I/O Pins (TCH1 and TCH0) . . . . .   | 39 |

|        |                                   |    |
|--------|-----------------------------------|----|
| 1.6.17 | Timer External Clock Input (TCLK) | 39 |
| 1.6.18 | Port A I/O Pins (PTA7–PTA0)       | 39 |
| 1.6.19 | Port B I/O Pins (PTB7–PTB0)       | 39 |
| 1.6.20 | Port C I/O Pins (PTC4–PTC0)       | 40 |
| 1.6.21 | Scan Enable (SCANEN)              | 40 |

## Section 2. Memory Map

|     |                                |    |
|-----|--------------------------------|----|
| 2.1 | Contents                       | 41 |
| 2.2 | Introduction                   | 41 |
| 2.3 | Unimplemented Memory Locations | 41 |
| 2.4 | Reserved Memory Locations      | 42 |
| 2.5 | Input/Output (I/O) Section     | 42 |

## Section 3. Low-Power Modes

|       |  |    |
|-------|--|----|
| 3.1   | Contents                                 | 57 |
| 3.2   | Introduction                             | 58 |
| 3.2.1 | Wait Mode                                | 58 |
| 3.2.2 | Stop Mode                                | 58 |
| 3.3   | A/D Converter (ADC4)                     | 59 |
| 3.4   | Break Module (BRK)                       | 59 |
| 3.4.1 | Wait Mode                                | 59 |
| 3.4.2 | Stop Mode                                | 59 |
| 3.5   | Central Processor Unit (CPU)             | 59 |
| 3.5.1 | Wait Mode                                | 59 |
| 3.5.2 | Stop Mode                                | 60 |
| 3.6   | Clock Generator Module (CGM)             | 60 |
| 3.6.1 | Wait Mode                                | 60 |
| 3.6.2 | Stop Mode                                | 60 |
| 3.7   | Closed-Caption Data Slicer Module (DSL)  | 61 |
| 3.8   | Computer Operating Properly Module (COP) | 61 |
| 3.8.1 | Wait Mode                                | 61 |
| 3.8.2 | Stop Mode                                | 61 |

|        |   |    |
|--------|---|----|
| 3.9    | External Interrupt Module (IRQ)           | 61 |
| 3.9.1  | Wait Mode                                 | 61 |
| 3.9.2  | Stop Mode                                 | 62 |
| 3.10   | Low-Voltage Inhibit Module (LVI)          | 62 |
| 3.10.1 | Wait Mode                                 | 62 |
| 3.10.2 | Stop Mode                                 | 62 |
| 3.11   | On-Screen Display Module (OSD)            | 62 |
| 3.11.1 | Wait Mode                                 | 62 |
| 3.11.2 | Stop Mode                                 | 62 |
| 3.12   | Serial Synchronous Interface Module (SSI) | 63 |
| 3.13   | Timer Interface Module (TIM)              | 63 |
| 3.13.1 | Wait Mode                                 | 63 |
| 3.13.2 | Stop Mode                                 | 63 |
| 3.14   | Timebase Module (TBM)                     | 63 |
| 3.14.1 | Wait Mode                                 | 63 |
| 3.14.2 | Stop Mode                                 | 64 |
| 3.15   | Exiting Wait Mode                         | 64 |
| 3.16   | Exiting Stop Mode                         | 65 |

## Section 4. Resets and Interrupts

|       |   |    |
|-------|---|----|
| 4.1   | Contents  | 67 |
| 4.2   | Introduction  | 68 |
| 4.3   | Resets  | 68 |
| 4.3.1 | Effects   | 68 |
| 4.3.2 | External Reset  | 68 |
| 4.3.3 | Internal Reset  | 68 |
| 4.3.4 | System Integration Module (SIM) Reset Status Register | 71 |
| 4.4   | Interrupts  | 73 |
| 4.4.1 | Effects   | 73 |
| 4.4.2 | Sources   | 76 |
| 4.4.3 | Interrupt Status Registers                            | 79 |

## Section 5. Analog-to-Digital Converter (ADC4)

|       |   |    |
|-------|---|----|
| 5.1   | Contents . . . . .                        | 81 |
| 5.2   | Introduction . . . . .                    | 81 |
| 5.3   | Feature . . . . .                         | 81 |
| 5.4   | Overview . . . . .                        | 82 |
| 5.5   | Programming Guidelines . . . . .          | 82 |
| 5.5.1 | Setup . . . . .                           | 82 |
| 5.5.2 | Conversions . . . . .                     | 83 |
| 5.5.3 | Input/Output . . . . .                    | 83 |
| 5.6   | ADC Control and Status Register . . . . . | 84 |
| 5.7   | Low-Power Modes . . . . .                 | 85 |
| 5.8   | Interrupts and Resets . . . . .           | 85 |

## Section 6. Break Module (BRK)

|       |   |    |
|-------|---|----|
| 6.1   | Contents . . . . .                                | 87 |
| 6.2   | Introduction . . . . .                            | 87 |
| 6.3   | Features . . . . .                                | 88 |
| 6.4   | Functional Description . . . . .                  | 88 |
| 6.4.1 | Flag Protection During Break Interrupts . . . . . | 90 |
| 6.4.2 | CPU During Break Interrupts . . . . .             | 90 |
| 6.4.3 | TIM During Break Interrupts . . . . .             | 90 |
| 6.4.4 | COP During Break Interrupts . . . . .             | 90 |
| 6.5   | Low-Power Modes . . . . .                         | 90 |
| 6.5.1 | Wait Mode . . . . .                               | 91 |
| 6.5.2 | Stop Mode . . . . .                               | 91 |
| 6.6   | Break Module Registers . . . . .                  | 91 |
| 6.6.1 | Break Status and Control Register . . . . .       | 92 |
| 6.6.2 | Break Address Registers . . . . .                 | 93 |
| 6.6.3 | SIM Break Status Register . . . . .               | 94 |
| 6.6.4 | SIM Break Flag Control Register . . . . .         | 95 |



## Section 7. Clock Generator Module (CGMC)

|       |   |     |
|-------|---|-----|
| 7.1   | Contents . . . . .                                  | 97  |
| 7.2   | Introduction . . . . .                              | 98  |
| 7.3   | Features . . . . .                                  | 98  |
| 7.4   | Functional Description . . . . .                    | 99  |
| 7.4.1 | Crystal Oscillator Circuit . . . . .                | 99  |
| 7.4.2 | Phase-Locked Loop Circuit (PLL) . . . . .           | 101 |
| 7.4.3 | PLL Circuits . . . . .                              | 101 |
| 7.4.4 | Acquisition and Tracking Modes . . . . .            | 102 |
| 7.4.5 | Manual and Automatic PLL Bandwidth Modes . . . . .  | 103 |
| 7.4.6 | Programming the PLL . . . . .                       | 104 |
| 7.4.7 | Special Programming Exceptions . . . . .            | 108 |
| 7.4.8 | Base Clock Selector Circuit . . . . .               | 108 |
| 7.4.9 | CGMC External Connections . . . . .                 | 109 |
| 7.5   | Input/Output Signals . . . . .                      | 110 |
| 7.5.1 | Crystal Amplifier Input Pin (OSC1) . . . . .        | 110 |
| 7.5.2 | Crystal Amplifier Output Pin (OSC2) . . . . .       | 110 |
| 7.5.3 | External Filter Capacitor Pin (CGMXFC) . . . . .    | 111 |
| 7.5.4 | PLL Analog Power Pin ( $V_{DDCGM}$ ) . . . . .      | 111 |
| 7.5.5 | PLL Analog Ground Pin ( $V_{SSCGM}$ ) . . . . .     | 111 |
| 7.5.6 | Crystal Output Frequency Signal (CGMXCLK) . . . . . | 111 |
| 7.5.7 | CGMC Base Clock Output (CGMOUT) . . . . .           | 112 |
| 7.5.8 | CGMC CPU Interrupt (CGMINT) . . . . .               | 112 |
| 7.6   | CGMC Registers . . . . .                            | 112 |
| 7.6.1 | PLL Control Register . . . . .                      | 114 |
| 7.6.2 | PLL Bandwidth Control Register . . . . .            | 117 |
| 7.6.3 | PLL Multiplier Select Register High . . . . .       | 118 |
| 7.6.4 | PLL Multiplier Select Register Low . . . . .        | 119 |
| 7.6.5 | PLL VCO Range Select Register . . . . .             | 120 |
| 7.6.6 | PLL Reference Divider Select Register . . . . .     | 121 |
| 7.7   | Interrupts . . . . .                                | 122 |
| 7.8   | Special Modes . . . . .                             | 122 |
| 7.8.1 | Wait Mode . . . . .                                 | 122 |
| 7.8.2 | Stop Mode . . . . .                                 | 123 |
| 7.8.3 | CGMC During Break Interrupts . . . . .              | 123 |
| 7.9   | Acquisition/Lock Time Specifications . . . . .      | 123 |

|       |  |     |
|-------|--|-----|
| 7.9.1 | Acquisition/Lock Time Definitions . . . . .      | 124 |
| 7.9.2 | Parametric Influences on Reaction Time . . . . . | 124 |
| 7.9.3 | Choosing a Filter . . . . .                      | 125 |

## Section 8. Closed-Caption Data Slicer (DSL)

|       |  |     |
|-------|--|-----|
| 8.1   | Contents . . . . .                                   | 127 |
| 8.2   | Introduction . . . . .                               | 128 |
| 8.3   | Features . . . . .                                   | 128 |
| 8.4   | Functional Description . . . . .                     | 128 |
| 8.4.1 | Slicer . . . . .                                     | 128 |
| 8.4.2 | Line and Field Detection . . . . .                   | 129 |
| 8.4.3 | Data Sampling . . . . .                              | 130 |
| 8.5   | Programming Guidelines . . . . .                     | 130 |
| 8.5.1 | Setup . . . . .                                      | 131 |
| 8.5.2 | Interrupt Servicing . . . . .                        | 131 |
| 8.5.3 | Debugging . . . . .                                  | 131 |
| 8.6   | Input/Output . . . . .                               | 132 |
| 8.6.1 | Video . . . . .                                      | 132 |
| 8.6.2 | VSYNC, HSYNC, and Video Input Requirements . . . . . | 132 |
| 8.7   | Registers . . . . .                                  | 134 |
| 8.7.1 | DSL Character Registers . . . . .                    | 135 |
| 8.7.2 | DSL Control Register 1 . . . . .                     | 136 |
| 8.7.3 | DSL Control Register 2 . . . . .                     | 138 |
| 8.7.4 | DSL Status Register . . . . .                        | 141 |
| 8.8   | Low-Power Modes . . . . .                            | 142 |
| 8.9   | Interrupts and Resets . . . . .                      | 142 |

## Section 9. Configuration Register (CONFIG)

|     |                                  |     |
|-----|----------------------------------|-----|
| 9.1 | Contents . . . . .               | 143 |
| 9.2 | Introduction . . . . .           | 143 |
| 9.3 | Functional Description . . . . . | 143 |

## Section 10. Computer Operating Properly (COP)

|        |                              |     |
|--------|------------------------------|-----|
| 10.1   | Contents                     | 147 |
| 10.2   | Introduction                 | 147 |
| 10.3   | Functional Description       | 148 |
| 10.4   | I/O Signals                  | 149 |
| 10.4.1 | CGMXCLK                      | 149 |
| 10.4.2 | STOP Instruction             | 149 |
| 10.4.3 | COPCTL Write                 | 149 |
| 10.4.4 | Power-On Reset               | 149 |
| 10.4.5 | Internal Reset               | 150 |
| 10.4.6 | Reset Vector Fetch           | 150 |
| 10.4.7 | COPD (COP Disable)           | 150 |
| 10.4.8 | COPRS (COP Rate Select)      | 150 |
| 10.5   | COP Control Register         | 150 |
| 10.6   | Interrupts                   | 151 |
| 10.7   | Monitor Mode                 | 151 |
| 10.8   | Low-Power Modes              | 151 |
| 10.8.1 | Wait Mode                    | 151 |
| 10.8.2 | Stop Mode                    | 151 |
| 10.9   | COP Module During Break Mode | 152 |

## Section 11. Central Processor Unit (CPU)

|        |                             |     |
|--------|-----------------------------|-----|
| 11.1   | Contents                    | 153 |
| 11.2   | Introduction                | 153 |
| 11.3   | Features                    | 154 |
| 11.4   | CPU Registers               | 155 |
| 11.4.1 | Accumulator                 | 155 |
| 11.4.2 | Index Register              | 156 |
| 11.4.3 | Stack Pointer               | 157 |
| 11.4.4 | Program Counter             | 158 |
| 11.4.5 | Condition Code Register     | 159 |
| 11.5   | Arithmetic/Logic Unit (ALU) | 161 |

|        |                             |     |
|--------|-----------------------------|-----|
| 11.6   | Low-Power Modes             | 161 |
| 11.6.1 | Wait Mode                   | 161 |
| 11.6.2 | Stop Mode                   | 161 |
| 11.7   | CPU During Break Interrupts | 162 |
| 11.8   | Instruction Set Summary     | 163 |
| 11.9   | Opcode Map                  | 169 |

## Section 12. User FLASH Memory

|        |                                     |     |
|--------|-------------------------------------|-----|
| 12.1   | Contents                            | 171 |
| 12.2   | Introduction                        | 171 |
| 12.3   | Functional Description              | 171 |
| 12.4   | FLASH Control Register              | 173 |
| 12.5   | Charge Pump                         | 175 |
| 12.5.1 | FLASH Charge Pump Frequency Control | 175 |
| 12.6   | FLASH Erase Operation               | 176 |
| 12.7   | FLASH Program/Margin Read Operation | 178 |
| 12.8   | User FLASH Block Protection         | 181 |
| 12.9   | User FLASH Block Protect Register   | 182 |
| 12.10  | Wait Mode                           | 183 |
| 12.11  | Stop Mode                           | 183 |

## Section 13. On-Screen Display (OSD)

### FLASH Memory

|      |                                     |     |
|------|-------------------------------------|-----|
| 13.1 | Contents                            | 185 |
| 13.2 | Introduction                        | 185 |
| 13.3 | Functional Description              | 186 |
| 13.4 | OSD FLASH Control Register          | 187 |
| 13.5 | Charge Pump Frequency Control       | 189 |
| 13.6 | FLASH Erase Operation               | 190 |
| 13.7 | FLASH Program/Margin Read Operation | 192 |

|      |  |     |
|------|--|-----|
| 13.8 | Block Protection . . . . .                 | 193 |
| 13.9 | OSD FLASH Block Protect Register . . . . . | 193 |

## Section 14. External Interrupt (IRQ)

|      |  |     |
|------|--|-----|
| 14.1 | Contents . . . . .                           | 195 |
| 14.2 | Introduction . . . . .                       | 195 |
| 14.3 | Features . . . . .                           | 195 |
| 14.4 | Functional Description . . . . .             | 196 |
| 14.5 | $\overline{\text{IRQ}}$ Pin . . . . .        | 198 |
| 14.6 | IRQ Module During Break Interrupts . . . . . | 199 |
| 14.7 | IRQ Status and Control Register . . . . .    | 199 |

## Section 15. Low-Voltage Inhibit (LVI)

|        |   |     |
|--------|---|-----|
| 15.1   | Contents . . . . .                      | 201 |
| 15.2   | Introduction . . . . .                  | 201 |
| 15.3   | Features . . . . .                      | 202 |
| 15.4   | Functional Description . . . . .        | 202 |
| 15.4.1 | Polled LVI Operation . . . . .          | 204 |
| 15.4.2 | Forced Reset Operation . . . . .        | 204 |
| 15.4.3 | Voltage Hysteresis Protection . . . . . | 204 |
| 15.4.4 | LVI Trip Selection . . . . .            | 204 |
| 15.5   | LVI Status Register . . . . .           | 205 |
| 15.6   | LVI Interrupts . . . . .                | 205 |
| 15.7   | Low-Power Modes . . . . .               | 206 |
| 15.7.1 | Wait Mode . . . . .                     | 206 |
| 15.7.2 | Stop Mode . . . . .                     | 206 |

## Section 16. Monitor ROM (MON)

|        |                        |     |
|--------|------------------------|-----|
| 16.1   | Contents               | 207 |
| 16.2   | Introduction           | 207 |
| 16.3   | Features               | 208 |
| 16.4   | Functional Description | 208 |
| 16.4.1 | Entering Monitor Mode  | 210 |
| 16.4.2 | Data Format            | 212 |
| 16.4.3 | Break Signal           | 213 |
| 16.4.4 | Baud Rate              | 213 |
| 16.4.5 | Commands               | 214 |
| 16.4.6 | Security               | 218 |

## Section 17. On-Screen Display Module (OSD)

|        |                              |     |
|--------|------------------------------|-----|
| 17.1   | Contents                     | 221 |
| 17.2   | Introduction                 | 222 |
| 17.3   | Features                     | 222 |
| 17.4   | Overview                     | 224 |
| 17.4.1 | Single-Row Architecture      | 225 |
| 17.4.2 | Display Timing               | 225 |
| 17.4.3 | Registers and Pixel Memory   | 226 |
| 17.4.4 | OSD Output Logic             | 227 |
| 17.5   | Display Characteristics      | 228 |
| 17.5.1 | Closed-Caption Mode          | 228 |
| 17.5.2 | On-Screen Display Mode       | 230 |
| 17.6   | FLASH Programming Guidelines | 234 |
| 17.7   | Programming Guidelines       | 239 |
| 17.7.1 | Setup                        | 239 |
| 17.7.2 | Interrupt Servicing          | 240 |
| 17.7.3 | Software Controlled Features | 242 |
| 17.8   | Input and Output             | 242 |
| 17.8.1 | Power Supply Pins            | 243 |
| 17.8.2 | Input Pins                   | 243 |
| 17.8.3 | PLL Filter Pins              | 243 |
| 17.8.4 | Output Pins                  | 244 |

|         |                                 |     |
|---------|---------------------------------|-----|
| 17.9    | Registers                       | 244 |
| 17.9.1  | OSD Character Registers         | 245 |
| 17.9.2  | OSD Vertical Delay Register     | 251 |
| 17.9.3  | OSD Horizontal Delay Register   | 252 |
| 17.9.4  | OSD Foreground Control Register | 252 |
| 17.9.5  | OSD Background Control Register | 254 |
| 17.9.6  | OSD Border Control Register     | 255 |
| 17.9.7  | OSD Enable Control Register     | 257 |
| 17.9.8  | OSD Event Line Register         | 259 |
| 17.9.9  | OSD Event Count Register        | 260 |
| 17.9.10 | OSD Output Control Register     | 260 |
| 17.9.11 | OSD Status Register             | 262 |
| 17.9.12 | OSD Matrix Start Register       | 263 |
| 17.9.13 | OSD Matrix End Register         | 265 |
| 17.10   | Low Power Modes                 | 265 |
| 17.10.1 | Wait Mode                       | 265 |
| 17.10.2 | Stop Mode                       | 266 |
| 17.11   | Interrupts and Resets           | 266 |

## Section 18. Input/Output (I/O) Ports

|        |                           |     |
|--------|---------------------------|-----|
| 18.1   | Contents                  | 267 |
| 18.2   | Introduction              | 267 |
| 18.3   | Port A                    | 269 |
| 18.3.1 | Port A Data Register      | 269 |
| 18.3.2 | Data Direction Register A | 270 |
| 18.4   | Port B                    | 272 |
| 18.4.1 | Port B Data Register      | 272 |
| 18.4.2 | Data Direction Register B | 273 |
| 18.5   | Port C                    | 274 |
| 18.5.1 | Port C Data Register      | 274 |
| 18.5.2 | Data Direction Register C | 275 |

## Section 19. Random-Access Memory (RAM)

|      |                        |     |
|------|------------------------|-----|
| 19.1 | Contents               | 277 |
| 19.2 | Introduction           | 277 |
| 19.3 | Functional Description | 277 |

## Section 20. System Integration Module (SIM)

|        |                                       |     |
|--------|---------------------------------------|-----|
| 20.1   | Contents                              | 279 |
| 20.2   | Introduction                          | 280 |
| 20.3   | SIM Bus Clock Control and Generation  | 283 |
| 20.3.1 | Bus Timing                            | 284 |
| 20.3.2 | Clock Startup from POR or LVI Reset   | 284 |
| 20.3.3 | Clocks in Stop Mode and Wait Mode     | 284 |
| 20.4   | Reset and System Initialization       | 284 |
| 20.4.1 | External Pin Reset                    | 285 |
| 20.4.2 | Active Resets from Internal Sources   | 286 |
| 20.5   | SIM Counter                           | 289 |
| 20.5.1 | SIM Counter During Power-On Reset     | 289 |
| 20.5.2 | SIM Counter During Stop Mode Recovery | 290 |
| 20.5.3 | SIM Counter and Reset States          | 290 |
| 20.6   | Exception Control                     | 290 |
| 20.6.1 | Interrupts                            | 291 |
| 20.6.2 | Reset                                 | 296 |
| 20.6.3 | Break Interrupts                      | 296 |
| 20.6.4 | Status Flag Protection in Break Mode  | 296 |
| 20.7   | Low-Power Modes                       | 297 |
| 20.7.1 | Wait Mode                             | 297 |
| 20.7.2 | Stop Mode                             | 298 |
| 20.8   | SIM Registers                         | 300 |
| 20.8.1 | SIM Break Status Register             | 300 |
| 20.8.2 | SIM Reset Status Register             | 301 |
| 20.8.3 | SIM Break Flag Control Register       | 303 |



## Section 21. Serial Synchronous Interface Module (SSI)

|        |                                  |     |
|--------|----------------------------------|-----|
| 21.1   | Contents .....                   | 305 |
| 21.2   | Introduction .....               | 306 |
| 21.3   | Features .....                   | 306 |
| 21.4   | Overview .....                   | 307 |
| 21.4.1 | START Signal .....               | 308 |
| 21.4.2 | Slave Address Transmission ..... | 309 |
| 21.4.3 | Data Transfer .....              | 309 |
| 21.4.4 | STOP Signal .....                | 310 |
| 21.4.5 | Clock Synchronization .....      | 310 |
| 21.4.6 | Handshaking .....                | 310 |
| 21.4.7 | Clock Stretching .....           | 311 |
| 21.5   | Programming Guidelines .....     | 311 |
| 21.5.1 | Setup .....                      | 311 |
| 21.5.2 | Interrupt Serving .....          | 312 |
| 21.6   | Registers .....                  | 313 |
| 21.6.1 | SSI Control Register .....       | 313 |
| 21.6.2 | SSI Status Register .....        | 315 |
| 21.6.3 | SSI Data Register .....          | 316 |
| 21.7   | Low-Power Modes .....            | 317 |
| 21.8   | Interrupt .....                  | 317 |

## Section 22. Timebase Module (TBM)

|      |                                     |     |
|------|-------------------------------------|-----|
| 22.1 | Contents .....                      | 319 |
| 22.2 | Introduction .....                  | 319 |
| 22.3 | Features .....                      | 319 |
| 22.4 | Functional Description .....        | 320 |
| 22.5 | Timebase Register Description ..... | 321 |
| 22.6 | Interrupts .....                    | 322 |
| 22.7 | Low-Power Modes .....               | 323 |

## Section 23. Timer Interface Module (TIM)

|        |  |     |
|--------|--|-----|
| 23.1   | Contents                                 | 325 |
| 23.2   | Introduction                             | 326 |
| 23.3   | Features                                 | 326 |
| 23.4   | Functional Description                   | 326 |
| 23.4.1 | TIM Counter Prescaler                    | 329 |
| 23.4.2 | Input Capture                            | 329 |
| 23.4.3 | Output Compare                           | 329 |
| 23.4.4 | Unbuffered Output Compare                | 330 |
| 23.4.5 | Buffered Output Compare                  | 330 |
| 23.4.6 | Pulse Width Modulation (PWM)             | 331 |
| 23.4.7 | Unbuffered PWM Signal Generation         | 332 |
| 23.4.8 | Buffered PWM Signal Generation           | 333 |
| 23.4.9 | PWM Initialization                       | 334 |
| 23.5   | Interrupts                               | 335 |
| 23.6   | Low-Power Modes                          | 335 |
| 23.6.1 | Wait Mode                                | 336 |
| 23.6.2 | Stop Mode                                | 336 |
| 23.7   | TIM During Break Interrupts              | 336 |
| 23.8   | I/O Signals                              | 337 |
| 23.9   | I/O Registers                            | 337 |
| 23.9.1 | TIM Status and Control Register          | 337 |
| 23.9.2 | TIM Counter Registers                    | 340 |
| 23.9.3 | TIM Counter Modulo Registers             | 341 |
| 23.9.4 | TIM Channel Status and Control Registers | 342 |
| 23.9.5 | TIM Channel Registers                    | 346 |

## Section 24. ROM Version Overview (ROM)

|      |                                    |     |
|------|------------------------------------|-----|
| 24.1 | Contents                           | 349 |
| 24.2 | Introduction                       | 349 |
| 24.3 | FLASH for ROM Substitution         | 349 |
| 24.4 | Configuration Register Programming | 351 |

## Section 25. Preliminary Electrical Specifications

|         |   |     |
|---------|---|-----|
| 25.1    | Contents . . . . .                                | 353 |
| 25.2    | Introduction . . . . .                            | 353 |
| 25.3    | Absolute Maximum Ratings . . . . .                | 354 |
| 25.4    | Functional Operating Range . . . . .              | 355 |
| 25.5    | Thermal Characteristics . . . . .                 | 355 |
| 25.6    | 5.0-V DC Electrical Characteristics . . . . .     | 356 |
| 25.7    | 5.0-V Control Timing . . . . .                    | 358 |
| 25.8    | ADC4 Characteristics . . . . .                    | 359 |
| 25.9    | Timer Interface Module Characteristics . . . . .  | 359 |
| 25.10   | Clock Generation Module Characteristics . . . . . | 359 |
| 25.10.1 | CGM Component Specifications . . . . .            | 359 |
| 25.10.2 | CGM Electrical Specifications . . . . .           | 360 |
| 25.11   | Memory Characteristics . . . . .                  | 361 |
| 25.12   | 5.0-V SSI Characteristics . . . . .               | 362 |

## Section 26. Mechanical Specifications

|      |  |     |
|------|--|-----|
| 26.1 | Contents . . . . .                             | 363 |
| 26.2 | Introduction . . . . .                         | 363 |
| 26.3 | 52-Pin Plastic Quad Flat Pack (PQFP) . . . . . | 364 |

## Section 27. Ordering Information

|      |                            |     |
|------|----------------------------|-----|
| 27.1 | Contents . . . . .         | 365 |
| 27.2 | Introduction . . . . .     | 365 |
| 27.3 | MC Order Numbers . . . . . | 365 |

# Table of Contents

## List of Figures

| Figure | Title  | Page |
|--------|--|------|
| 1-1    | MC68HC908TV24 Block Diagram . . . . .                | 34   |
| 1-2    | Pin Assignments. . . . .                             | 35   |
| 1-3    | Power Supply Bypassing . . . . .                     | 36   |
| 2-1    | Memory Map . . . . .                                 | 43   |
| 2-2    | Control, Status, and Data Registers. . . . .         | 45   |
| 4-1    | Internal Reset Timing . . . . .                      | 69   |
| 4-2    | Power-On Reset Recovery. . . . .                     | 70   |
| 4-3    | SIM Reset Status Register (SRSR) . . . . .           | 72   |
| 4-4    | Interrupt Stacking Order . . . . .                   | 74   |
| 4-5    | Interrupt Recognition Example . . . . .              | 74   |
| 4-6    | Interrupt Processing . . . . .                       | 75   |
| 4-7    | Interrupt Status Register 1 (INT1) . . . . .         | 80   |
| 4-8    | Interrupt Status Register 2 (INT2) . . . . .         | 80   |
| 5-1    | ADC4 Block Diagram . . . . .                         | 82   |
| 5-2    | ADC Conversion Code Example . . . . .                | 83   |
| 5-3    | ADC Control and Status Register (ADCCSR) . . . . .   | 84   |
| 6-1    | Break Module Block Diagram. . . . .                  | 89   |
| 6-2    | I/O Register Summary . . . . .                       | 89   |
| 6-3    | Break Status and Control Register (BRKSCR) . . . . . | 92   |
| 6-4    | Break Address Register High (BRKH) . . . . .         | 93   |
| 6-5    | Break Address Register Low (BRKL). . . . .           | 93   |
| 6-7    | Example Code . . . . .                               | 94   |
| 6-6    | SIM Break Status Register (SBSR) . . . . .           | 94   |
| 6-8    | SIM Break Flag Control Register (SBFCR) . . . . .    | 95   |

## List of Figures

| Figure | Title  | Page |
|--------|--|------|
| 7-1    | CGMC Block Diagram . . . . .                           | 100  |
| 7-2    | CGMC External Connections . . . . .                    | 110  |
| 7-3    | CGMC I/O Register Summary . . . . .                    | 113  |
| 7-4    | PLL Control Register (PCTL) . . . . .                  | 114  |
| 7-5    | PLL Bandwidth Control Register (PBWC) . . . . .        | 117  |
| 7-6    | PLL Multiplier Select Register High (PMSH) . . . . .   | 118  |
| 7-7    | PLL Multiplier Select Register Low (PMSL) . . . . .    | 119  |
| 7-8    | PLL VCO Range Select Register (PMRS) . . . . .         | 120  |
| 7-9    | PLL Reference Divider Select Register (PMDS) . . . . . | 121  |
| 7-10   | PLL Filter . . . . .                                   | 126  |
| 8-1    | Data Slicer Block Diagram . . . . .                    | 129  |
| 8-2    | Slicer Input and Output. . . . .                       | 130  |
| 8-3    | Video Input Circuit . . . . .                          | 133  |
| 8-4    | DSL Input Timing . . . . .                             | 133  |
| 8-5    | DSL Character Registers (DSLCH1 and DSLCH2) . . . . .  | 135  |
| 8-6    | DSL Control Register 1 (DSLCR1) . . . . .              | 136  |
| 8-7    | DSL Control Register 2 (DSLCR2) . . . . .              | 138  |
| 8-8    | Data and Sync Slicing . . . . .                        | 139  |
| 8-9    | Conditions for Setting Vertical Pulse Delay . . . . .  | 140  |
| 8-10   | DSL Status Register (DSLRSR) . . . . .                 | 141  |
| 9-1    | Configuration Register (CONFIG) . . . . .              | 144  |
| 10-1   | COP Block Diagram . . . . .                            | 148  |
| 10-2   | COP Control Register (COPCTL) . . . . .                | 150  |
| 11-1   | CPU Registers . . . . .                                | 155  |
| 11-2   | Accumulator (A) . . . . .                              | 155  |
| 11-3   | Index Register (H:X) . . . . .                         | 156  |
| 11-4   | Stack Pointer (SP) . . . . .                           | 157  |
| 11-5   | Program Counter (PC) . . . . .                         | 158  |
| 11-6   | Condition Code Register (CCR) . . . . .                | 159  |
| 12-1   | User FLASH Control Register (FL1CR) . . . . .          | 173  |
| 12-2   | Smart Programming Algorithm Flowchart . . . . .        | 180  |

| <b>Figure</b> | <b>Title</b>   | <b>Page</b> |
|---------------|--|-------------|
| 12-3          | User FLASH Block Protect Register (FL1BPR) . . . . .         | 182         |
| 13-1          | Connection of the OSD FLASH Memory . . . . .                 | 186         |
| 13-2          | FLASH Control Register (FL2CR) . . . . .                     | 187         |
| 13-3          | OSD FLASH Block Protect Register (FL2BPR) . . . . .          | 193         |
| 14-1          | IRQ Module Block Diagram . . . . .                           | 197         |
| 14-2          | IRQ I/O Register Summary . . . . .                           | 197         |
| 14-3          | IRQ Status and Control Register (INTSCR) . . . . .           | 200         |
| 15-1          | LVI Module Block Diagram . . . . .                           | 203         |
| 15-2          | LVI I/O Register Summary . . . . .                           | 203         |
| 15-3          | LVI Status Register (LVISR) . . . . .                        | 205         |
| 16-1          | Monitor Mode Circuit . . . . .                               | 209         |
| 16-2          | Monitor Data Format . . . . .                                | 212         |
| 16-3          | Sample Monitor Waveforms . . . . .                           | 212         |
| 16-4          | Break Transaction . . . . .                                  | 213         |
| 16-5          | Read Transaction . . . . .                                   | 214         |
| 16-6          | READ (Read Memory) Command . . . . .                         | 215         |
| 16-7          | WRITE (Write Memory) Command . . . . .                       | 215         |
| 16-8          | IREAD (Indexed Read) Command . . . . .                       | 216         |
| 16-9          | IWRITE (Indexed Write) Command . . . . .                     | 216         |
| 16-10         | READSP (Read Stack Pointer) Command . . . . .                | 217         |
| 16-11         | RUN (Run User Program) Command . . . . .                     | 217         |
| 16-12         | Monitor Mode Entry Timing . . . . .                          | 218         |
| 17-1          | OSD Block Diagram . . . . .                                  | 224         |
| 17-2          | Active Display Area . . . . .                                | 226         |
| 17-3          | CC-ROM Pixel Matrix . . . . .                                | 228         |
| 17-4          | Boundary Conditions with Italics . . . . .                   | 230         |
| 17-5          | OSD-ROM Pixel Matrix with Northwest Shadow . . . . .         | 231         |
| 17-6          | Example of 3D Shadow . . . . .                               | 231         |
| 17-7          | Behavior of Control Characters in OSD Mode . . . . .         | 232         |
| 17-8          | Overlaying Characters to Get Two Foreground Colors . . . . . | 233         |
| 17-9          | OSD FLASH Memory Map . . . . .                               | 234         |

## List of Figures

| Figure | Title  | Page |
|--------|--|------|
| 17-10  | OSD Pixel Matrix in FLASH Memory . . . . .             | 235  |
| 17-11  | CC Pixel Matrix in FLASH Memory . . . . .              | 236  |
| 17-12  | Spaces and Box-Making Characters in CC Mode . . . . .  | 237  |
| 17-13  | Pixel Matrices Organization in FLASH . . . . .         | 238  |
| 17-14  | PLL Filter Circuit . . . . .                           | 243  |
| 17-15  | OSD Character Registers (OSDCHAR1–OSDCHAR34) . . . . . | 245  |
| 17-16  | CC Mode Control Character — Format A . . . . .         | 246  |
| 17-17  | CC Mode Control Character — Format B . . . . .         | 246  |
| 17-18  | CC Mode Control Character — Format C . . . . .         | 246  |
| 17-19  | OSD Mode Control Character — Format A . . . . .        | 248  |
| 17-20  | OSD Mode Control Character — Format B . . . . .        | 248  |
| 17-21  | OSD Mode Control Character — Format C . . . . .        | 248  |
| 17-22  | OSD Vertical Delay Register (OSDVDR) . . . . .         | 251  |
| 17-23  | OSD Horizontal Delay Register (OSDHDR) . . . . .       | 252  |
| 17-24  | OSD Foreground Control Register (OSDFCR) . . . . .     | 253  |
| 17-25  | OSD Background Control Register (OSDBKCR) . . . . .    | 254  |
| 17-26  | OSD Border Control Register (OSDBCR) . . . . .         | 255  |
| 17-27  | OSD Enable Control Register (OSDECTR) . . . . .        | 257  |
| 17-28  | OSD Event Line Register (OSDELRL) . . . . .            | 259  |
| 17-29  | OSD Event Count Register (OSDECR) . . . . .            | 260  |
| 17-30  | OSD Output Control Register (OSDOCR) . . . . .         | 260  |
| 17-31  | OSD Status Register (OSDSR) . . . . .                  | 262  |
| 17-32  | OSD Matrix Start Register (OSDMSR) . . . . .           | 264  |
| 17-33  | OSD Matrix End Register (OSDMER) . . . . .             | 265  |
|        |  |      |
| 18-1   | I/O Port Register Summary . . . . .                    | 268  |
| 18-2   | Port A Data Register (PTA) . . . . .                   | 269  |
| 18-3   | Data Direction Register A (DDRA) . . . . .             | 270  |
| 18-4   | Port A I/O Circuit . . . . .                           | 271  |
| 18-5   | Port B Data Register (PTB) . . . . .                   | 272  |
| 18-6   | Data Direction Register B (DDRB) . . . . .             | 273  |
| 18-7   | Port B I/O Circuit . . . . .                           | 273  |
| 18-8   | Port C Data Register (PTC) . . . . .                   | 274  |
| 18-9   | Data Direction Register C (DDRC) . . . . .             | 275  |
| 18-10  | Port C I/O Circuit . . . . .                           | 276  |



| <b>Figure</b> | <b>Title</b>  | <b>Page</b> |
|---------------|---|-------------|
| 20-1          | SIM Block Diagram . . . . .                         | 281         |
| 20-2          | SIM I/O Register Summary . . . . .                  | 282         |
| 20-3          | CGM Clock Signals . . . . .                         | 283         |
| 20-4          | External Reset Timing . . . . .                     | 285         |
| 20-5          | Internal Reset Timing . . . . .                     | 286         |
| 20-6          | Sources of Internal Reset. . . . .                  | 286         |
| 20-7          | POR Recovery . . . . .                              | 287         |
| 20-8          | Interrupt Entry Timing. . . . .                     | 291         |
| 20-9          | Interrupt Recovery Timing . . . . .                 | 291         |
| 20-10         | Interrupt Processing . . . . .                      | 292         |
| 20-11         | Interrupt Recognition Example. . . . .              | 293         |
| 20-12         | Interrupt Status Register 1 (INT1) . . . . .        | 295         |
| 20-13         | Interrupt Status Register 2 (INT2) . . . . .        | 295         |
| 20-14         | Wait Mode Entry Timing. . . . .                     | 297         |
| 20-15         | Wait Recovery from Interrupt or Break. . . . .      | 298         |
| 20-16         | Wait Recovery from Internal Reset . . . . .         | 298         |
| 20-17         | Stop Mode Entry Timing. . . . .                     | 299         |
| 20-18         | Stop Mode Recovery from Interrupt or Break. . . . . | 299         |
| 20-19         | SIM Break Status Register (SBSR) . . . . .          | 300         |
| 20-20         | SIM Reset Status Register (SRSR) . . . . .          | 301         |
| 20-21         | SIM Break Flag Control Register (SBFCR) . . . . .   | 303         |
|               |   |             |
| 21-1          | SSI Block Diagram . . . . .                         | 307         |
| 21-2          | Transmission Signal . . . . .                       | 308         |
| 21-3          | SSI Control Register (SSICR) . . . . .              | 313         |
| 21-4          | SSI Status Register (SSISR) . . . . .               | 315         |
| 21-5          | SSI Data Register (SSIDR) . . . . .                 | 316         |
|               |   |             |
| 22-1          | Timebase Block Diagram . . . . .                    | 320         |
| 22-2          | Timebase Control Register (TBCR) . . . . .          | 321         |
|               |   |             |
| 23-1          | TIM Block Diagram. . . . .                          | 327         |
| 23-2          | TIM I/O Register Summary. . . . .                   | 328         |
| 23-3          | PWM Period and Pulse Width . . . . .                | 332         |
| 23-4          | TIM Status and Control Register (TSC) . . . . .     | 338         |
| 23-5          | TIM Counter Register High (TCNTH). . . . .          | 340         |

## List of Figures

| Figure | Title  | Page |
|--------|--|------|
| 23-6   | TIM Counter Register Low (TCNTL) . . . . .                 | 340  |
| 23-7   | TIM Counter Modulo Register High (TMODH) . . . . .         | 341  |
| 23-8   | TIM Counter Modulo Register Low (TMODL) . . . . .          | 341  |
| 23-9   | TIM Channel 0 Status and Control Register (TSC0) . . . . . | 342  |
| 23-10  | TIM Channel 1 Status and Control Register (TSC1) . . . . . | 342  |
| 23-11  | CHxMAX Latency . . . . .                                   | 345  |
| 23-12  | TIM Channel 0 Register High (TCH0H) . . . . .              | 346  |
| 23-13  | TIM Channel 0 Register Low (TCH0L) . . . . .               | 346  |
| 23-14  | TIM Channel 1 Register High (TCH1H) . . . . .              | 347  |
| 23-15  | TIM Channel 1 Register Low (TCH1L) . . . . .               | 347  |
| 24-1   | MC68HC08TV24 Block Diagram . . . . .                       | 350  |
| 25-1   | SSI Timing . . . . .                                       | 362  |

## List of Tables

| Table | Title                                  | Page |
|-------|--|------|
| 2-1   | Vector Addresses . . . . .             | 55   |
| 4-1   | Interrupt Sources . . . . .            | 76   |
| 4-2   | Interrupt Source Flags . . . . .       | 79   |
| 7-1   | Numeric Example . . . . .              | 108  |
| 7-2   | PRE 1 and PRE0 Programming . . . . .   | 116  |
| 7-3   | VPR1 and VPR0 Programming . . . . .    | 116  |
| 8-1   | Line Selection . . . . .               | 137  |
| 8-2   | Data Slicer Bias Voltage . . . . .     | 138  |
| 8-3   | Minimum Vertical Pulse Width . . . . . | 139  |
| 11-1  | Instruction Set Summary . . . . .      | 163  |
| 11-2  | Opcode Map . . . . .                   | 170  |
| 12-1  | Charge Pump Clock Frequency . . . . .  | 175  |
| 12-2  | Erase Block Sizes . . . . .            | 177  |
| 13-1  | Charge Pump Clock Frequency . . . . .  | 189  |
| 13-2  | Erase Block Sizes . . . . .            | 191  |
| 15-1  | LVIOUT Bit Indication . . . . .        | 205  |
| 16-1  | Mode Selection . . . . .               | 210  |
| 16-2  | Mode Differences . . . . .             | 212  |
| 17-1  | SHAD1 and SHAD0 Meaning . . . . .      | 250  |
| 17-2  | Memory Test Mode Bits . . . . .        | 258  |

## List of Tables

| Table | Title   | Page |
|-------|---|------|
| 18-1  | Port A Pin Functions . . . . .                          | 271  |
| 18-2  | Port B Pin Functions . . . . .                          | 274  |
| 18-3  | Port C Pin Functions . . . . .                          | 276  |
| 20-1  | Signal Name Conventions . . . . .                       | 281  |
| 20-2  | PIN Bit Set Timing . . . . .                            | 285  |
| 20-3  | Interrupt Sources . . . . .                             | 294  |
| 20-4  | SIM Registers . . . . .                                 | 300  |
| 21-1  | SCL Rates (Hz) at Bus Frequency . . . . .               | 315  |
| 22-1  | Timebase Rate Selection for OSC1 = 32.768 kHz . . . . . | 321  |
| 23-1  | Prescaler Selection . . . . .                           | 339  |
| 23-2  | Mode, Edge, and Level Selection . . . . .               | 344  |
| 27-1  | MC Order Numbers . . . . .                              | 365  |

## Section 1. General Description

### 1.1 Contents

|        |  |    |
|--------|--|----|
| 1.2    | Introduction   | 30 |
| 1.3    | Features   | 30 |
| 1.3.1  | Standard Features  | 30 |
| 1.3.2  | Television Application Specific Features                                   | 32 |
| 1.3.3  | CPU08 Features   | 33 |
| 1.4    | MCU Block Diagram  | 33 |
| 1.5    | Pin Assignments  | 35 |
| 1.6    | Pin Functions  | 36 |
| 1.6.1  | Power Supply Pins ( $V_{DD}$ and $V_{SS}$ )                                | 36 |
| 1.6.2  | Oscillator Pins (OSC1 and OSC2)  | 37 |
| 1.6.3  | External Reset Pin ( $\overline{RST}$ )                                    | 37 |
| 1.6.4  | External Interrupt Pin ( $\overline{IRQ}$ )                                | 37 |
| 1.6.5  | CGM Power Supply Pins ( $V_{DDCGM}$ and $V_{SSCGM}$ )                      | 37 |
| 1.6.6  | External Filter Capacitor Pin (CGMXFC)                                     | 37 |
| 1.6.7  | Power Supply Pins for On-Screen Display<br>( $V_{DDOSD}$ and $V_{SSOSD}$ ) | 37 |
| 1.6.8  | External Filter Pins for On-Screen Display<br>(OSDVCO and OSDPMP)          | 38 |
| 1.6.9  | Synchronism Signals (HSYNC and VSYNC)                                      | 38 |
| 1.6.10 | Color Encoded Pixel Signals (R, G, B, and I)                               | 38 |
| 1.6.11 | Fast Blanking Signal (FBKG)  | 38 |
| 1.6.12 | Serial Synchronous Interface Clocks (SCL1 and SCL2)                        | 38 |
| 1.6.13 | Serial Synchronous Interface Data Lines<br>(SDA1 and SDA2)                 | 38 |
| 1.6.14 | Video Input (VIDEO)  | 39 |
| 1.6.15 | A/D Converter Input (ADCIN)  | 39 |
| 1.6.16 | Timer I/O Pins (TCH1 and TCH0)   | 39 |
| 1.6.17 | Timer External Clock Input (TCLK)  | 39 |
| 1.6.18 | Port A I/O Pins (PTA7–PTA0)  | 39 |

|        |                                   |    |
|--------|-----------------------------------|----|
| 1.6.19 | Port B I/O Pins (PTB7–PTB0) ..... | 39 |
| 1.6.20 | Port C I/O Pins (PTC4–PTC0) ..... | 40 |
| 1.6.21 | Scan Enable (SCANEN).....         | 40 |

## 1.2 Introduction

The MC68HC908TV24 is a member of the low-cost, high-performance M68HC08 Family of 8-bit microcontroller units (MCUs). All MCUs in the family use the enhanced M68HC08 central processor unit (CPU08) and are available with a variety of modules, memory sizes and types, and package types.

This part contains an embedded on-screen display module and a closed-caption controller, making it highly suitable for use as a low-cost TV or VCR microcontroller.

## 1.3 Features

For convenience, features of the MC68HC908TV24 have been organized to reflect:

- Standard features
- Television application-specific features
- CPU08 features

### 1.3.1 Standard Features

- High-performance M68HC08 architecture
- Fully upward-compatible object code with M6805, M146805, and M68HC05 Families
- 8-MHz internal bus frequency
- 24 Kbytes of on-chip FLASH memory
- 608 bytes of on-chip random-access memory (RAM)
- 21 general-purpose input/output (I/O) pins

- FLASH program memory security<sup>1</sup>
- System protection features:
  - Optional computer operating properly (COP) reset
  - Low-voltage detection with optional reset and selectable trip points for 3.0-V and 5.0-V operation
  - Illegal opcode detection with reset
  - Illegal address detection with reset
- Low-power design; fully static with stop and wait modes
- Standard low-power modes of operation:
  - Wait mode
  - Stop mode
- 16-bit, 2-channel timer interface module (TIM) with selectable input capture, output compare, and pulse-width modulation (PWM) capability on each channel
- BREAK module (BRK) to allow single breakpoint setting during in-circuit debugging
- Clock generator module with on-chip 32-kHz crystal compatible PLL (phase-lock loop)
- Timebase module with clock prescaler circuitry for eight user selectable periodic real-time interrupts with active clock source during stop mode for periodic wakeup from stop using an external 32-kHz crystal
- Master reset pin and power-on reset (POR)
- 52-pin plastic quad flat pack (PQFP)

---

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

### 1.3.2 Television Application-Specific Features

- On-screen display (OSD) controller
  - Works with either 525- or 625-line systems
  - Pixel matrix FLASH memory capable of storing 192 9 x 13 closed-caption characters and 128 12 x 18 on-screen display characters
  - 16 rows by 34 columns in closed-caption mode
  - 12 rows by 24 columns in on-screen display mode
  - Software selectable attributes: 16 foreground and background colors, black outline, 3D-shadow, underline, and italics
  - Two foreground colors are possible in one character by the use of backspace overlaying of two characters
  - Three on-screen display (OSD) character sizes selectable on a row by row basis:  
(1 x 1), (1.5 x 2), and (2 x 2)
  - Programmable vertical and horizontal position
  - Software controlled features: soft scrolling and blinking
  - Support for video muting
  - Outputs: R, G, B, I, and FBKG, all with software programmable polarities
- Closed-caption data slicer (DSL):
  - FCC/EIA-744 V-chip compatible
  - FCC/EIA-608 line 21 format data extraction on both field 1 and field 2
  - Software programmable line selection (lines 14 to 29)
  - Hardware parity checking
  - Software programmable data slicing level
- Serial synchronous interface (SSI) — Two independent data ports with single master I<sup>2</sup>C compatibility (both ports cannot be used simultaneously)
- Analog-to-digital converter (ADC) — 4-bits resolution



### 1.3.3 CPU08 Features

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  $8 \times 8$  multiply instruction
- Fast 16/8 divide instruction
- Binary-coded decimal (BCD) instructions
- Optimization for controller applications
- Efficient C language support

## 1.4 MCU Block Diagram

**Figure 1-1** shows the structure of the MC68HC908TV24.

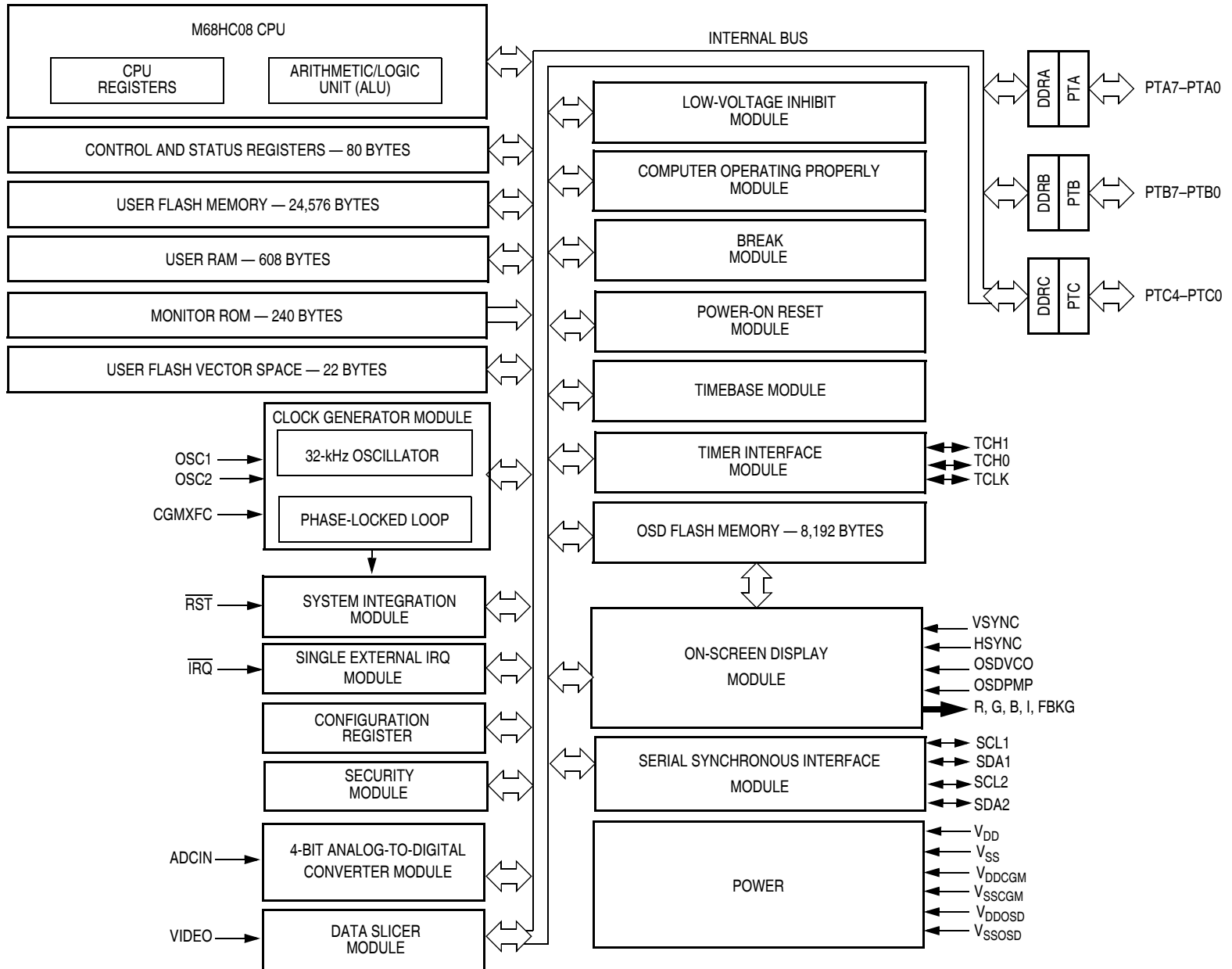
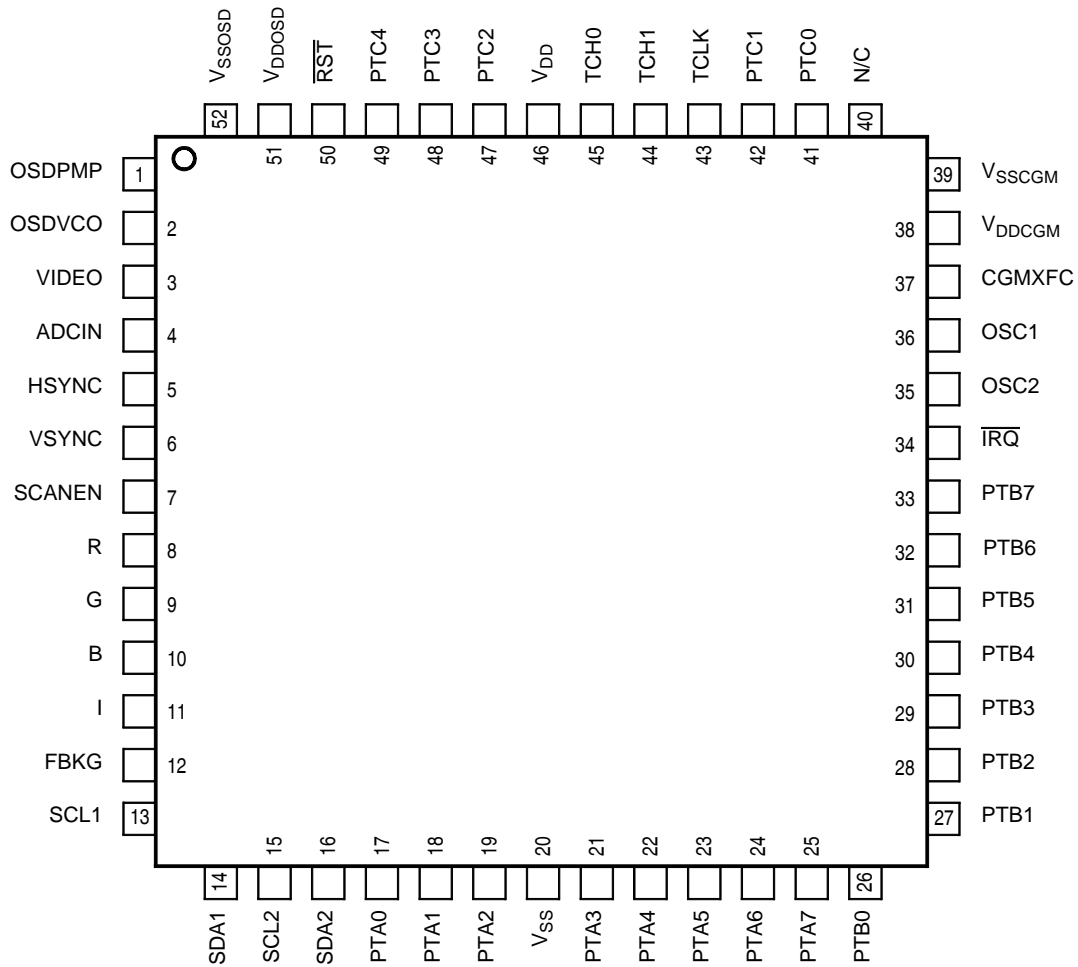


Figure 1-1. MC68HC908TV24 Block Diagram

## 1.5 Pin Assignments



**Figure 1-2. Pin Assignments**

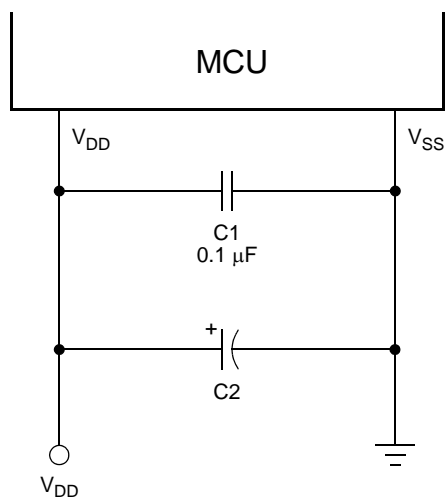
## 1.6 Pin Functions

Descriptions of the pin functions are provided here.

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

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

Fast signal transitions on MCU pins place high, short-duration current demands on the power supply. To prevent noise problems, take special care to provide power supply bypassing at the MCU as [Figure 1-3](#) shows. Place the C1 bypass capacitor as close to the MCU as possible. Use a high-frequency-response ceramic capacitor for C1. C2 is an optional bulk current bypass capacitor for use in applications that require the port pins to source high current levels.



Note: Component values shown represent typical applications.

**Figure 1-3. Power Supply Bypassing**

### 1.6.2 Oscillator Pins (OSC1 and OSC2)

The OSC1 and OSC2 pins are the connections for the on-chip oscillator circuit. See [Section 7. Clock Generator Module \(CGMC\)](#).

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

A logic 0 on the  $\overline{\text{RST}}$  pin forces the MCU to a known startup state.  $\overline{\text{RST}}$  is bidirectional, allowing a reset of the entire system. It is driven low when any internal reset source is asserted. See [Section 20. System Integration Module \(SIM\)](#).

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

$\overline{\text{IRQ}}$  is an asynchronous external interrupt pin. See [Section 14. External Interrupt \(IRQ\)](#).

### 1.6.5 CGM Power Supply Pins ( $V_{\text{DDCGM}}$ and $V_{\text{SSCGM}}$ )

$V_{\text{DDCGM}}$  and  $V_{\text{SSCGM}}$  are the power supply pins for the analog portion of the clock generator module (CGM). Decoupling of these pins should be as per the digital supply. See [Section 7. Clock Generator Module \(CGMC\)](#).

### 1.6.6 External Filter Capacitor Pin (CGMXFC)

CGMXFC is an external filter capacitor connection for the CGM. See [Section 7. Clock Generator Module \(CGMC\)](#).

### 1.6.7 Power Supply Pins for On-Screen Display ( $V_{\text{DDOSD}}$ and $V_{\text{SSOSD}}$ )

$V_{\text{DDOSD}}$  and  $V_{\text{SSOSD}}$  are the power supply and ground pins for the analog part of the on-screen display module (OSD). Decoupling of these pins should be as per the digital supply. See [Section 17. On-Screen Display Module \(OSD\)](#).

### 1.6.8 External Filter Pins for On-Screen Display (OSDVCO and OSDPMP)

OSDVCO and OSDPMP are two external filter pins used by the PLL that extracts the OSD clock from the horizontal sync signal. See [Section 17. On-Screen Display Module \(OSD\)](#).

### 1.6.9 Synchronism Signals (HSYNC and VSYNC)

These are two input-only pins used by the on-screen display module to synchronize to the television signal. These pins contain internal Schmitt triggers to improve noise immunity. See [Section 17. On-Screen Display Module \(OSD\)](#).

### 1.6.10 Color Encoded Pixel Signals (R, G, B, and I)

These are the red, green, blue and intensity output signals from the on-screen display module. See [Section 17. On-Screen Display Module \(OSD\)](#).

### 1.6.11 Fast Blanking Signal (FBKG)

FBKG is an output pin of the on-screen display module. It is used to blank the TV video when a pixel is being outputted on the R, G, B, and I pins. See [Section 17. On-Screen Display Module \(OSD\)](#).

### 1.6.12 Serial Synchronous Interface Clocks (SCL1 and SCL2)

SCL1 and SCL2 are open-drain I/O pins. They are the bidirectional clock lines of the serial synchronous interface module. See [Section 21. Serial Synchronous Interface Module \(SSI\)](#).

### 1.6.13 Serial Synchronous Interface Data Lines (SDA1 and SDA2)

SDA1 and SDA2 are open-drain I/O pins. They are the bidirectional data lines of the serial synchronous interface module. See [Section 21. Serial Synchronous Interface Module \(SSI\)](#).

#### 1.6.14 Video Input (VIDEO)

This input-only analog pin is the composite base-band video input signal for the data slicer module. See [Section 8. Closed-Caption Data Slicer \(DSL\)](#).

#### 1.6.15 A/D Converter Input (ADCIN)

This is the analog input to the analog-to-digital converter module. See [Section 5. Analog-to-Digital Converter \(ADC4\)](#).

#### 1.6.16 Timer I/O Pins (TCH1 and TCH0)

These pins are used by the timer interface module. They can be programmed independently as input capture or output compare pins. See [Section 23. Timer Interface Module \(TIM\)](#).

#### 1.6.17 Timer External Clock Input (TCLK)

This is an external clock input for the timer module that can be used instead of the prescaled internal bus clock. See [Section 23. Timer Interface Module \(TIM\)](#).

#### 1.6.18 Port A I/O Pins (PTA7–PTA0)

PTA7–PTA0 are general-purpose, bidirectional I/O port pins. See [Section 18. Input/Output \(I/O\) Ports](#).

#### 1.6.19 Port B I/O Pins (PTB7–PTB0)

PTB7–PTB0 are general-purpose, bidirectional I/O port pins. See [Section 18. Input/Output \(I/O\) Ports](#).

### 1.6.20 Port C I/O Pins (PTC4–PTC0)

PTC4–PTC0 are general-purpose, bidirectional I/O port pins. See [Section 18. Input/Output \(I/O\) Ports](#).

### 1.6.21 Scan Enable (SCANEN)

This pin is used only during production test and should be connected to  $V_{SS}$  in all user applications.

**NOTE:** *Any unused inputs and I/O ports should be tied to an appropriate logic level (either  $V_{DD}$  or  $V_{SS}$ ). Although the I/O ports of the MC68HC908TV24 do not require termination, termination is recommended to reduce the possibility of static damage.*



## Section 2. Memory Map

### 2.1 Contents

|     |  |    |
|-----|--|----|
| 2.2 | Introduction . . . . .                   | 41 |
| 2.3 | Unimplemented Memory Locations . . . . . | 41 |
| 2.4 | Reserved Memory Locations . . . . .      | 42 |
| 2.5 | Input/Output (I/O) Section. . . . .      | 42 |

### 2.2 Introduction

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

- 24,064 bytes of user FLASH memory
- 8 Kbytes of on-screen display (OSD) FLASH memory
- 608 bytes of random-access memory (RAM)
- 22 bytes of user-defined vectors
- 240 bytes of monitor read-only memory (ROM)

### 2.3 Unimplemented Memory Locations

Accessing an unimplemented location can cause an illegal address reset if illegal address resets are enabled. In the memory map (**Figure 2-1**) and in register figures in this document, unimplemented locations are shaded.

## 2.4 Reserved Memory Locations

Accessing a reserved location can have unpredictable effects on MCU operation. In [Figure 2-1](#) and in register figures in this document, reserved locations are marked with the word Reserved or with the letter R.

## 2.5 Input/Output (I/O) Section

Addresses \$0000–\$004F, shown in [Figure 2-2](#), contain most of the control, status, and data registers. Additional input/output (I/O) registers have these addresses:

- \$FE00; SIM break status register, SBSR
- \$FE01; SIM reset status register, SRSR
- \$FE02; reserved, SUBAR
- \$FE03; SIM break flag control register, SBFCR
- \$FE04; interrupt status register 1, INT1
- \$FE05; interrupt status register 2, INT2
- \$FE06; reserved, FL1TCR
- \$FE07; user FLASH control register, FL1CR
- \$FE08; reserved, FL2TCR
- \$FE09; OSD FLASH control register, FL2CR
- \$FE0A; OSD output control register, OSDOCR
- \$FE0B; OSD enable control register, OSDECTR
- \$FE0C; break address register high, BRKH
- \$FE0D; break address register low, BRKL
- \$FE0E; break status and control register, BRKSCR
- \$FE0F; LVI status register, LVISR
- \$FF80; user FLASH block protect register, FL1BPR
- \$FF81; OSD FLASH block protect register, FL2BPR
- \$FFFF; COP control register, COPCTL

[Table 2-1](#) is a list of vector locations.

|        |   |
|--------|---|
| \$0000 | I/O Registers<br>80 Bytes               |
| ↓      |   |
| \$004F | RAM<br>608 Bytes                        |
| ↓      |   |
| \$0050 | Unimplemented<br>32,080 Bytes           |
| ↓      |   |
| \$02AF | OSD FLASH Memory<br>8,192 Bytes         |
| ↓      |   |
| \$02B0 | User FLASH Memory<br>24,064 Bytes       |
| ↓      |   |
| \$7FFF | SIM Break Status Register (SBSR)        |
| \$8000 |   |
| ↓      | SIM Reset Status Register (SRSR)        |
| \$9FFF |   |
| \$A000 | Reserved (SUBAR)                        |
| ↓      |   |
| \$FDFF | SIM Break Flag Control Register (SBFCR) |
| \$FE00 |   |
| \$FE01 | Interrupt Status Register 1 (INT1)      |
| \$FE02 |   |
| \$FE03 | Interrupt Status Register 2 (INT2)      |
| \$FE04 |   |
| \$FE05 | Reserved (FL1TCR)                       |
| \$FE06 |   |
| \$FE07 | User FLASH Control Register (FL1CR)     |
| \$FE08 |   |
| \$FE09 | Reserved (FL2TCR)                       |
| \$FE0A |   |
| \$FE0B | OSD FLASH Control Register (FL2CR)      |
| \$FE0C |   |
| \$FE0D | OSD Output Control Register (OSDOCR)    |
| \$FE0E |   |
| \$FE0F | OSD Enable Control Register (OSDECTR)   |
| \$FE10 |   |
| \$FE11 | Break Address Register High (BRKH)      |
| \$FE12 |   |
| \$FE13 | Break Address Register Low (BRKL)       |
| \$FE14 |   |

**Figure 2-1. Memory Map**

# Memory Map

|             |  |
|-------------|--|
| \$FE0E      | Break Status and Control Register (BRKSCR) |
| \$FE0F      | LVI Status Register (LVISR)                |
| \$FE10<br>↓ | Monitor ROM<br>240 Bytes                   |
| \$FEFF      |  |
| \$FF00<br>↓ | Unimplemented<br>128 Bytes                 |
| \$FF7F      |  |
| \$FF80      | User FLASH Block Protect Register (FL1BPR) |
| \$FF81      | OSD FLASH Block Protect Register (FL2BPR)  |
| \$FF82<br>↓ | Reserved<br>104 Bytes                      |
| \$FFE9      |  |
| \$FFEA<br>↓ | FLASH Vectors<br>22 Bytes                  |
| \$FFFF      |  |

**Figure 2-1. Memory Map (Continued)**

| Addr.  | Register Name  | Bit 7  | 6                   | 5     | 4                | 3     | 2     | 1     | Bit 0 |       |
|--------|--|--------|---------------------|-------|------------------|-------|-------|-------|-------|-------|
| \$0000 | Port A Data Register (PTA)<br>See <a href="#">page 269</a> .               | Read:  | PTA7                | PTA6  | PTA5             | PTA4  | PTA3  | PTA2  | PTA1  | PTA0  |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | Unaffected by reset |       |                  |       |       |       |       |       |
| \$0001 | Port B Data Register (PTB)<br>See <a href="#">page 272</a> .               | Read:  | PTB7                | PTB6  | PTB5             | PTB4  | PTB3  | PTB2  | PTB1  | PTB0  |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | Unaffected by reset |       |                  |       |       |       |       |       |
| \$0002 | Port C Data Register (PTC)<br>See <a href="#">page 274</a> .               | Read:  | 0                   | 0     | 0                | PTC4  | PTC3  | PTC2  | PTC1  | PTC0  |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | Unaffected by reset |       |                  |       |       |       |       |       |
| \$0003 | Unimplemented  | Read:  |                     |       |                  |       |       |       |       |       |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | Unaffected by reset |       |                  |       |       |       |       |       |
| \$0004 | Data Direction Register A (DDRA)<br>See <a href="#">page 270</a> .         | Read:  | DDRA7               | DDRA6 | DDRA5            | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 0                | 0     | 0     | 0     | 0     | 0     |
| \$0005 | Data Direction Register B (DDRB)<br>See <a href="#">page 273</a> .         | Read:  | DDRB7               | DDRB6 | DDRB5            | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 0                | 0     | 0     | 0     | 0     | 0     |
| \$0006 | Data Direction Register C (DDRC)<br>See <a href="#">page 275</a> .         | Read:  | 0                   | 0     | 0                | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 0                | 0     | 0     | 0     | 0     | 0     |
| \$0007 | IRQ Status and Control Register (INTSCR)<br>See <a href="#">page 199</a> . | Read:  | 0                   | 0     | 0                | 0     | IRQF  | 0     | IMASK | MODE  |
|        |  | Write: |                     |       |                  |       |       | ACK   |       |       |
|        |  | Reset: | 0                   | 0     | 0                | 0     | 0     | 0     | 0     | 0     |
| \$0008 | PLL Control Register (PCTL)<br>See <a href="#">page 114</a> .              | Read:  | PLLIE               | PLLIF | PLLON            | BCS   | PRE1  | PRE0  | VPR1  | VPR0  |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 1                | 0     | 0     | 0     | 0     | 0     |
| \$0009 | PLL Bandwidth Control Register (PBWC)<br>See <a href="#">page 117</a> .    | Read:  | AUTO                | LOCK  | $\overline{ACQ}$ | 0     | 0     | 0     | 0     | R     |
|        |  | Write: |                     |       |                  |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 0                | 0     | 0     | 0     | 0     | 0     |


= Unimplemented   
 R = Reserved   
 U = Unaffected   
 X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 1 of 10)**

# Memory Map

| Addr.  | Register Name  | Bit 7  | 6       | 5        | 4       | 3       | 2     | 1     | Bit 0 |       |
|--------|--|--------|---------|----------|---------|---------|-------|-------|-------|-------|
| \$000A | PLL Multiplier Select High Register (PMSH)<br>See <a href="#">page 118</a> .   | Read:  | 0       | 0        | 0       | 0       | MUL11 | MUL10 | MUL9  | MUL8  |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 0       | 0        | 0       | 0       | 0     | 0     | 0     | 0     |
| \$000B | PLL Multiplier Select Low Register (PMSL)<br>See <a href="#">page 119</a> .    | Read:  | MUL7    | MUL6     | MUL5    | MUL4    | MUL3  | MUL2  | MUL1  | MUL0  |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 0       | 1        | 0       | 0       | 0     | 0     | 0     | 0     |
| \$000C | PLL VCO Select Range Register (PMRS)<br>See <a href="#">page 120</a> .         | Read:  | VRS7    | VRS6     | VRS5    | VRS4    | VRS3  | VRS2  | VRS1  | VRS0  |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 0       | 1        | 0       | 0       | 0     | 0     | 0     | 0     |
| \$000D | PLL Reference Divider Select Register (PMDS)<br>See <a href="#">page 121</a> . | Read:  | 0       | 0        | 0       | 0       | RDS3  | RDS2  | RDS1  | RDS0  |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 0       | 0        | 0       | 0       | 0     | 0     | 0     | 1     |
| \$000E | Configuration Register† (CONFIG)<br>See <a href="#">page 144</a> .             | Read:  | LVISTOP | LVI5OR3† | LVIRSTD | LVIPWRD | SSREC | COPRS | STOP  | COPD  |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 0       | 0        | 0       | 0       | 0     | 0     | 0     | 0     |
| \$000F | Timer Status and Control Register (TSC)<br>See <a href="#">page 337</a> .      | Read:  | TOF     | TOIE     | TSTOP   | 0       | 0     | PS2   | PS1   | PS0   |
|        |  | Write: | 0       |          |         | TRST    |       |       |       |       |
|        |  | Reset: | 0       | 0        | 1       | 0       | 0     | 0     | 0     | 0     |
| \$0010 | Timer Counter Register High (TCNTH)<br>See <a href="#">page 340</a> .          | Read:  | Bit 15  | 14       | 13      | 12      | 11    | 10    | 9     | Bit 8 |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 0       | 0        | 0       | 0       | 0     | 0     | 0     | 0     |
| \$0011 | Timer Counter Register Low (TCNTL)<br>See <a href="#">page 340</a> .           | Read:  | Bit 7   | 6        | 5       | 4       | 3     | 2     | 1     | Bit 0 |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 0       | 0        | 0       | 0       | 0     | 0     | 0     | 0     |
| \$0012 | Timer Counter Modulo Register High (TMODH)<br>See <a href="#">page 341</a> .   | Read:  | Bit 15  | 14       | 13      | 12      | 11    | 10    | 9     | Bit 8 |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 1       | 1        | 1       | 1       | 1     | 1     | 1     | 1     |
| \$0013 | Timer Counter Modulo Register Low (TMODL)<br>See <a href="#">page 341</a> .    | Read:  | Bit 7   | 6        | 5       | 4       | 3     | 2     | 1     | Bit 0 |
|        |  | Write: |         |          |         |         |       |       |       |       |
|        |  | Reset: | 1       | 1        | 1       | 1       | 1     | 1     | 1     | 1     |

† One-time writable register after each reset, except LVI5OR3 bit. The LVI5OR3 bit is only reset via POR (power-on reset).

 = Unimplemented    R = Reserved    U = Unaffected    X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 2 of 10)**

| Addr.  | Register Name   | Bit 7  | 6                         | 5     | 4     | 3    | 2     | 1     | Bit 0 |        |
|--------|---|--------|---------------------------|-------|-------|------|-------|-------|-------|--------|
| \$0014 | Timer Channel 0 Status and Control Register (TSC0) See <a href="#">page 342</a> . | Read:  | CH0F                      | CH0IE | MS0B  | MS0A | ELS0B | ELS0A | TOV0  | CH0MAX |
|        |   | Write: | 0                         |       |       |      |       |       |       |        |
|        |   | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$0015 | Timer Channel 0 Register High (TCH0H) See <a href="#">page 346</a> .              | Read:  | Bit 15                    | 14    | 13    | 12   | 11    | 10    | 9     | Bit 8  |
|        |   | Write: |                           |       |       |      |       |       |       |        |
|        |   | Reset: | Indeterminate after reset |       |       |      |       |       |       |        |
| \$0016 | Timer Channel 0 Register Low (TCH0L) See <a href="#">page 346</a> .               | Read:  | Bit 7                     | 6     | 5     | 4    | 3     | 2     | 1     | Bit 0  |
|        |   | Write: |                           |       |       |      |       |       |       |        |
|        |   | Reset: | Indeterminate after reset |       |       |      |       |       |       |        |
| \$0017 | Timer Channel 1 Status and Control Register (TSC1) See <a href="#">page 342</a> . | Read:  | CH1F                      | CH1IE | 0     | MS1A | ELS1B | ELS1A | TOV1  | CH1MAX |
|        |   | Write: | 0                         |       |       |      |       |       |       |        |
|        |   | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$0018 | Timer Channel 1 Register High (TCH1H) See <a href="#">page 346</a> .              | Read:  | Bit 15                    | 14    | 13    | 12   | 11    | 10    | 9     | Bit 8  |
|        |   | Write: |                           |       |       |      |       |       |       |        |
|        |   | Reset: | Indeterminate after reset |       |       |      |       |       |       |        |
| \$0019 | Timer Channel 1 Register Low (TCH1L) See <a href="#">page 346</a> .               | Read:  | Bit 7                     | 6     | 5     | 4    | 3     | 2     | 1     | Bit 0  |
|        |   | Write: |                           |       |       |      |       |       |       |        |
|        |   | Reset: | Indeterminate after reset |       |       |      |       |       |       |        |
| \$001A | Timebase Control Register (TBCR) See <a href="#">page 321</a> .                   | Read:  | TBIF                      | TBR2  | TBR1  | TBR0 | 0     | TBIE  | TBON  | R      |
|        |   | Write: |                           |       |       |      | TACK  |       |       |        |
|        |   | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$001B | OSD Horizontal Delay Register (OSDHDR) See <a href="#">page 252</a> .             | Read:  | 0                         | 0     | 0     | HD4  | HD3   | HD2   | HD1   | HD0    |
|        |   | Write: |                           |       |       |      |       |       |       |        |
|        |   | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$001C | OSD Foreground Control Register (OSDFCR) See <a href="#">page 252</a> .           | Read:  | CHHS                      | CHWS  | RNDEN | BOEN | FGI   | FGR   | FGB   | FGG    |
|        |   | Write: |                           |       |       |      |       |       |       |        |
|        |   | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$001D | OSD Background Control Register (OSDBKCR) See <a href="#">page 254</a> .          | Read:  | BKHT                      | SHAD1 | SHAD0 | BKS  | BKI   | BKR   | BKB   | BKG    |
|        |   | Write: |                           |       |       |      |       |       |       |        |
|        |   | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |

= Unimplemented    R = Reserved    U = Unaffected    X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 3 of 10)**

# Memory Map

| Addr.  | Register Name  | Bit 7  | 6                   | 5     | 4   | 3   | 2   | 1   | Bit 0 |     |
|--------|--|--------|---------------------|-------|-----|-----|-----|-----|-------|-----|
| \$001E | OSD Matrix Start Register (OSDMSR)<br>See <a href="#">page 263</a> . | Read:  | DMODE               | BLINK | 0   | MS4 | MS3 | MS2 | MS1   | MS0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | 0                   | 0     | 0   | 0   | 0   | 0   | 0     | 0   |
| \$001F | OSD Matrix End Register (OSDMER)<br>See <a href="#">page 265</a> .   | Read:  | 0                   | 0     | 0   | ME4 | ME3 | ME2 | ME1   | ME0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | 0                   | 0     | 0   | 0   | 0   | 0   | 0     | 0   |
| \$0020 | OSD Character 1 (OSDCHAR1)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |
| \$0021 | OSD Character 2 (OSDCHAR2)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |
| \$0022 | OSD Character 3 (OSDCHAR3)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |
| \$0023 | OSD Character 4 (OSDCHAR4)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |
| \$0024 | OSD Character 5 (OSDCHAR5)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |
| \$0025 | OSD Character 6 (OSDCHAR6)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |
| \$0026 | OSD Character 7 (OSDCHAR7)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |
| \$0027 | OSD Character 8 (OSDCHAR8)<br>See <a href="#">page 245</a> .         | Read:  | CH7                 | CH6   | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |  | Write: |                     |       |     |     |     |     |       |     |
|        |  | Reset: | Unaffected by reset |       |     |     |     |     |       |     |

= Unimplemented    R = Reserved    U = Unaffected    X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 4 of 10)**



| Addr.  | Register Name   | Bit 7  | 6                   | 5   | 4   | 3   | 2   | 1   | Bit 0 |     |
|--------|---|--------|---------------------|-----|-----|-----|-----|-----|-------|-----|
| \$0028 | OSD Character 9<br>(OSDCHAR9)<br>See <a href="#">page 245</a> .   | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0029 | OSD Character 10<br>(OSDCHAR10)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$002A | OSD Character 11<br>(OSDCHAR11)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$002B | OSD Character 12<br>(OSDCHAR12)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$002C | OSD Character 13<br>(OSDCHAR13)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$002D | OSD Character 14<br>(OSDCHAR14)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$002E | OSD Character 15<br>(OSDCHAR15)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$002F | OSD Character 16<br>(OSDCHAR16)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0030 | OSD Character 17<br>(OSDCHAR17)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0031 | OSD Character 18<br>(OSDCHAR18)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |

= Unimplemented    R = Reserved    U = Unaffected    X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 5 of 10)**

# Memory Map

| Addr.  | Register Name   | Bit 7  | 6                   | 5   | 4   | 3   | 2   | 1   | Bit 0 |     |
|--------|---|--------|---------------------|-----|-----|-----|-----|-----|-------|-----|
| \$0032 | OSD Character 19<br>(OSDCHAR19)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0033 | OSD Character 20<br>(OSDCHAR20)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0034 | OSD Character 21<br>(OSDCHAR21)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0035 | OSD Character 22<br>(OSDCHAR22)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0036 | OSD Character 23<br>(OSDCHAR23)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0037 | OSD Character 24<br>(OSDCHAR24)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0038 | OSD Character 25<br>(OSDCHAR25)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$0039 | OSD Character 26<br>(OSDCHAR26)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$003A | OSD Character 27<br>(OSDCHAR27)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |
| \$003B | OSD Character 28<br>(OSDCHAR28)<br>See <a href="#">page 245</a> . | Read:  | CH7                 | CH6 | CH5 | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                     |     |     |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset |     |     |     |     |     |       |     |

= Unimplemented   
 R = Reserved   
 U = Unaffected   
 X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 6 of 10)**

| Addr.  | Register Name   | Bit 7  | 6                         | 5    | 4     | 3   | 2   | 1   | Bit 0 |     |
|--------|---|--------|---------------------------|------|-------|-----|-----|-----|-------|-----|
| \$003C | OSD Character 29<br>(OSDCHAR29)<br>See <a href="#">page 245</a> .         | Read:  | CH7                       | CH6  | CH5   | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset       |      |       |     |     |     |       |     |
| \$003D | OSD Character 30<br>(OSDCHAR30)<br>See <a href="#">page 245</a> .         | Read:  | CH7                       | CH6  | CH5   | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset       |      |       |     |     |     |       |     |
| \$003E | OSD Character 31<br>(OSDCHAR31)<br>See <a href="#">page 245</a> .         | Read:  | CH7                       | CH6  | CH5   | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset       |      |       |     |     |     |       |     |
| \$003F | OSD Character 32<br>(OSDCHAR32)<br>See <a href="#">page 245</a> .         | Read:  | CH7                       | CH6  | CH5   | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset       |      |       |     |     |     |       |     |
| \$0040 | OSD Character 33<br>(OSDCHAR33)<br>See <a href="#">page 245</a> .         | Read:  | CH7                       | CH6  | CH5   | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset       |      |       |     |     |     |       |     |
| \$0041 | OSD Character 34<br>(OSDCHAR34)<br>See <a href="#">page 245</a> .         | Read:  | CH7                       | CH6  | CH5   | CH4 | CH3 | CH2 | CH1   | CH0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | Unaffected by reset       |      |       |     |     |     |       |     |
| \$0042 | OSD Vertical Delay<br>Register (OSDVDR)<br>See <a href="#">page 251</a> . | Read:  | 0                         | 0    | VD5   | VD4 | VD3 | VD2 | VD1   | VD0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | 0                         | 0    | 0     | 0   | 0   | 0   | 0     | 0   |
| \$0043 | OSD Border Control<br>Register (OSDBCR)<br>See <a href="#">page 255</a> . | Read:  | BLINKEN                   | BOHT | VMUTE | BOS | BOI | BOR | BOB   | BOG |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | 0                         | 0    | 0     | 0   | 0   | 0   | 0     | 0   |
| \$0044 | OSD Event Line Register<br>(OSDELR)<br>See <a href="#">page 259</a> .     | Read:  | EL7                       | EL6  | EL5   | EL4 | EL3 | EL2 | EL1   | EL0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | 0                         | 0    | 0     | 0   | 0   | 0   | 0     | 0   |
| \$0045 | OSD Event Count<br>Register (OSDECR)<br>See <a href="#">page 260</a> .    | Read:  | EV7                       | EV6  | EV5   | EV4 | EV3 | EV2 | EV1   | EV0 |
|        |   | Write: |                           |      |       |     |     |     |       |     |
|        |   | Reset: | Indeterminate after reset |      |       |     |     |     |       |     |

= Unimplemented   
 R = Reserved   
 U = Unaffected   
 X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 7 of 10)**

# Memory Map

| Addr.  | Register Name   | Bit 7  | 6      | 5     | 4      | 3     | 2     | 1       | Bit 0   |         |
|--------|---|--------|--------|-------|--------|-------|-------|---------|---------|---------|
| \$0046 | OSD Status Register (OSDSR)<br>See <a href="#">page 262</a> .             | Read:  | ELMF   | VSINF | HSYN   | VSYN  | 0     | VCOTST  | DSLSTST | PLLTST  |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 1      | 0     | X      | X     | 0     | 0       | 0       | 0       |
| \$0047 | DSL Character Register 1 (DSLCH1)<br>See <a href="#">page 135</a> .       | Read:  | PE     | DA6   | DA5    | DA4   | DA3   | DA2     | DA1     | DA0     |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | X     | X      | X     | X     | X       | X       | X       |
| \$0048 | DSL Character Register 2 (DSLCH2)<br>See <a href="#">page 135</a> .       | Read:  | PE     | DA6   | DA5    | DA4   | DA3   | DA2     | DA1     | DA0     |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | X     | X      | X     | X     | X       | X       | X       |
| \$0049 | DSL Control Register 1 (DSLCR1)<br>See <a href="#">page 136</a> .         | Read:  | DSIEN  | DSEN  | VVEN   | LINE4 | LINE3 | LINE2   | LINE1   | DISCLP  |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | 0     | 0      | 0     | 0     | 0       | 0       | 0       |
| \$004A | DSL Control Register 2 (DSLCR2)<br>See <a href="#">page 138</a> .         | Read:  | VR1    | VR0   | PW1    | PW0   | VPD   | SYNCOMP | DATCOMP | BLKCOMP |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | 0     | 0      | 0     | 0     | 0       | 0       | 0       |
| \$004B | DSL Status Register (DSL SR)<br>See <a href="#">page 141</a> .            | Read:  | DSFL   | OVFL  | FIELD1 | 0     | CSYNC | VPDET   | RIC1    | RIC0    |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | 0     | 0      | 0     | X     | 0       | 0       | 0       |
| \$004C | SSI Control Register (SSICR)<br>See <a href="#">page 313</a> .            | Read:  | SIE    | SE    | START  | STOP  | ACK   | SCHS    | SR1     | SR0     |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | 0     | 0      | 0     | 0     | 0       | 0       | 0       |
| \$004D | SSI Status Register (SSISR)<br>See <a href="#">page 315</a> .             | Read:  | SF     | DCOL  | 0      | 0     | 0     | 0       | 0       | 0       |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | 0     | 0      | 0     | 0     | 0       | 0       | 0       |
| \$004E | SSI Data Register (SSIDR)<br>See <a href="#">page 316</a> .               | Read:  | DA7    | DA6   | DA5    | DA4   | DA3   | DA2     | DA1     | DA0     |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | 0     | 0      | 0     | 0     | 0       | 0       | 0       |
| \$004F | ADC Control and Status Register (ADCCSR)<br>See <a href="#">page 84</a> . | Read:  | RESULT | ADON  | 0      | 0     | AD3   | AD2     | AD1     | AD0     |
|        |   | Write: |        |       |        |       |       |         |         |         |
|        |   | Reset: | 0      | 0     | 0      | 0     | 0     | 0       | 0       | 0       |

= Unimplemented   
 R = Reserved   
 U = Unaffected   
 X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 8 of 10)**

| Addr.                                | Register Name   |        | Bit 7 | 6     | 5    | 4    | 3    | 2      | 1     | Bit 0 |
|--------------------------------------|---|--------|-------|-------|------|------|------|--------|-------|-------|
| \$FE00                               | SIM Break Status Register (SBSR)<br>See <a href="#">page 300</a> .        | Read:  | 0     | 0     | 0    | 1    | 0    | 0      | BW    | 0     |
|                                      |   | Write: | R     | R     | R    | R    | R    | R      | Note  | R     |
|                                      |   | Reset: | 0     | 0     | 0    | 1    | 0    | 0      | 0     | 0     |
| Note: Writing a logic 0 clears SBSW. |   |        |       |       |      |      |      |        |       |       |
| \$FE01                               | SIM Reset Status Register (SRSR)<br>See <a href="#">page 301</a> .        | Read:  | POR   | PIN   | COP  | ILOP | ILAD | 0      | LVI   | 0     |
|                                      |   | Write: |       |       |      |      |      |        |       |       |
|                                      |   | POR:   | 1     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |
| \$FE02                               | SIM Upper Byte Address Register (SUBAR)                                   | Read:  | R     | R     | R    | R    | R    | R      | R     | R     |
|                                      |   | Write: |       |       |      |      |      |        |       |       |
|                                      |   | Reset: |       |       |      |      |      |        |       |       |
| \$FE03                               | SIM Break Flag Control Register (SBFCR)<br>See <a href="#">page 303</a> . | Read:  | BCFE  | R     | R    | R    | R    | R      | R     | R     |
|                                      |   | Write: |       |       |      |      |      |        |       |       |
|                                      |   | Reset: | 0     |       |      |      |      |        |       |       |
| \$FE04                               | Interrupt Status Register 1 (INT1)<br>See <a href="#">page 199</a> .      | Read:  | IF6   | IF5   | IF4  | IF3  | IF2  | IF1    | 0     | 0     |
|                                      |   | Write: | R     | R     | R    | R    | R    | R      | R     | R     |
|                                      |   | Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |
| \$FE05                               | Interrupt Status Register 2 (INT2)<br>See <a href="#">page 199</a> .      | Read:  | 0     | 0     | 0    | 0    | 0    | IF9    | IF8   | IF7   |
|                                      |   | Write: | R     | R     | R    | R    | R    | R      | R     | R     |
|                                      |   | Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |
| \$FE06                               | User FLASH Test Control Register (FL1TCR)                                 | Read:  | R     | R     | R    | R    | R    | R      | R     | R     |
|                                      |   | Write: |       |       |      |      |      |        |       |       |
|                                      |   | Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |
| \$FE07                               | User FLASH Control Register (FL1CR)<br>See <a href="#">page 173</a> .     | Read:  | FDIV1 | FDIV0 | BLK1 | BLK0 | HVEN | MARGIN | ERASE | PGM   |
|                                      |   | Write: |       |       |      |      |      |        |       |       |
|                                      |   | Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |
| \$FE08                               | OSD FLASH Test Control Register (FL2TCR)                                  | Read:  | R     | R     | R    | R    | R    | R      | R     | R     |
|                                      |   | Write: |       |       |      |      |      |        |       |       |
|                                      |   | Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |
| \$FE09                               | OSD FLASH Control Register (FL2CR)<br>See <a href="#">page 187</a> .      | Read:  | FDIV1 | FDIV0 | BLK1 | BLK0 | HVEN | MARGIN | ERASE | PGM   |
|                                      |   | Write: |       |       |      |      |      |        |       |       |
|                                      |   | Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |


= Unimplemented   
 R = Reserved   
 U = Unaffected   
 X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 9 of 10)**

# Memory Map


| Addr.  | Register Name   |        | Bit 7                                 | 6     | 5     | 4     | 3    | 2     | 1    | Bit 0 |
|--------|---|--------|---------------------------------------|-------|-------|-------|------|-------|------|-------|
| \$FE0A | OSD Output Control Register (OSDOCR)<br>See <a href="#">page 260</a> .                    | Read:  | 0                                     | HINV  | VINV  | FDINV | CINV | FBINV | IINV | 0     |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | 0                                     | 0     | 0     | 0     | 0    | 0     | 0    | 0     |
| \$FE0B | OSD Enable Control Register (OSDECTR)<br>See <a href="#">page 257</a> .                   | Read:  | OSDEN                                 | ELIEN | VSIEN | XFER  | PLEN | MEM1  | MEM0 | SCAN  |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | 0                                     | 0     | 0     | 0     | 0    | 0     | 0    | 0     |
| \$FE0C | Break Address Register High (BRKH)<br>See <a href="#">page 93</a> .                       | Read:  | Bit 15                                | 14    | 13    | 12    | 11   | 10    | 9    | Bit 8 |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | 0                                     | 0     | 0     | 0     | 0    | 0     | 0    | 0     |
| \$FE0D | Break Address Register Low (BRKL)<br>See <a href="#">page 93</a> .                        | Read:  | Bit 7                                 | 6     | 5     | 4     | 3    | 2     | 1    | Bit 0 |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | 0                                     | 0     | 0     | 0     | 0    | 0     | 0    | 0     |
| \$FE0E | Break Status and Control Register (BRKSCR)<br>See <a href="#">page 92</a> .               | Read:  | BRKE                                  | BRKA  | 0     | 0     | 0    | 0     | 0    | 0     |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | 0                                     | 0     | 0     | 0     | 0    | 0     | 0    | 0     |
| \$FE0F | LVI Status Register (LVISR)<br>See <a href="#">page 205</a> .                             | Read:  | LVIOUT                                | 0     | 0     | 0     | 0    | 0     | 0    | 0     |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | 0                                     | 0     | 0     | 0     | 0    | 0     | 0    | 0     |
| \$FF80 | User FLASH Block Protect Register (FL1BPR) <sup>†</sup><br>See <a href="#">page 182</a> . | Read:  | R                                     | R     | R     | R     | BPR3 | BPR2  | BPR1 | BPR0  |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | U                                     | U     | U     | U     | U    | U     | U    | U     |
| \$FF81 | OSD FLASH Block Protect Register (FL2BPR) <sup>†</sup><br>See <a href="#">page 193</a> .  | Read:  | R                                     | R     | R     | R     | BPR3 | BPR2  | BPR1 | BPR0  |
|        |   | Write: |                                       |       |       |       |      |       |      |       |
|        |   | Reset: | U                                     | U     | U     | U     | U    | U     | U    | U     |
| \$FFFF | COP Control Register (COPCTL)<br>See <a href="#">page 150</a> .                           | Read:  | Low byte of reset vector              |       |       |       |      |       |      |       |
|        |   | Write: | Writing clears COP counter, any value |       |       |       |      |       |      |       |
|        |   | Reset: | Unaffected by reset                   |       |       |       |      |       |      |       |

† Non-volatile FLASH register

 = Unimplemented    R = Reserved    U = Unaffected    X = Indeterminate

**Figure 2-2. Control, Status, and Data Registers (Sheet 10 of 10)**

**Table 2-1. Vector Addresses**

| Vector Priority   | Vector | Address             | Vector                                |
|---|--------|---------------------|---------------------------------------|
| Lowest<br><br>Highest | IF9    | \$FFEA              | SSI Vector (High)                     |
|   |        | \$FFEB              | SSI Vector (Low)                      |
|   | IF8    | \$FFEC              | DSL Vector (High)                     |
|   |        | \$FFED              | DSL Vector (Low)                      |
|   | IF7    | \$FFEE              | OSD Vector (High)                     |
|   |        | \$FFEF              | OSD Vector (Low)                      |
|   | IF6    | \$FFF0              | TIM Overflow Vector (High)            |
|   |        | \$FFF1              | TIM Overflow Vector (Low)             |
|   | IF5    | \$FFF2              | TIM Channel 1 Vector (High)           |
|   |        | \$FFF3              | TIM Channel 1 Vector (Low)            |
|   | IF4    | \$FFF4              | TIM Channel 0 Vector (High)           |
|   |        | \$FFF5              | TIM Channel 0 Vector (Low)            |
|   | IF3    | \$FFF6              | Timebase Vector (High)                |
|   |        | \$FFF7              | Timebase Vector (Low)                 |
|   | IF2    | \$FFF8              | PLL Vector (High)                     |
|   |        | \$FFF9              | PLL Vector (Low)                      |
|   | IF1    | \$FFFA              | $\overline{\text{IRQ}}$ Vector (High) |
|   |        | \$FFFFB             | $\overline{\text{IRQ}}$ Vector (Low)  |
|   | —      | \$FFFC              | SWI Vector (High)                     |
| \$FFFD  |        | SWI Vector (Low)    |                                       |
| —   | \$FFFE | Reset Vector (High) |                                       |
|   | \$FFFF | Reset Vector (Low)  |                                       |





## Section 3. Low-Power Modes

### 3.1 Contents

|        |   |    |
|--------|---|----|
| 3.2    | Introduction                              | 58 |
| 3.2.1  | Wait Mode                                 | 58 |
| 3.2.2  | Stop Mode                                 | 58 |
| 3.3    | A/D Converter (ADC4)                      | 59 |
| 3.4    | Break Module (BRK)                        | 59 |
| 3.4.1  | Wait Mode                                 | 59 |
| 3.4.2  | Stop Mode                                 | 59 |
| 3.5    | Central Processor Unit (CPU)              | 59 |
| 3.5.1  | Wait Mode                                 | 59 |
| 3.5.2  | Stop Mode                                 | 60 |
| 3.6    | Clock Generator Module (CGM)              | 60 |
| 3.6.1  | Wait Mode                                 | 60 |
| 3.6.2  | Stop Mode                                 | 60 |
| 3.7    | Closed-Caption Data Slicer Module (DSL)   | 61 |
| 3.8    | Computer Operating Properly Module (COP)  | 61 |
| 3.8.1  | Wait Mode                                 | 61 |
| 3.8.2  | Stop Mode                                 | 61 |
| 3.9    | External Interrupt Module (IRQ)           | 61 |
| 3.9.1  | Wait Mode                                 | 61 |
| 3.9.2  | Stop Mode                                 | 62 |
| 3.10   | Low-Voltage Inhibit Module (LVI)          | 62 |
| 3.10.1 | Wait Mode                                 | 62 |
| 3.10.2 | Stop Mode                                 | 62 |
| 3.11   | On-Screen Display Module (OSD)            | 62 |
| 3.11.1 | Wait Mode                                 | 62 |
| 3.11.2 | Stop Mode                                 | 62 |
| 3.12   | Serial Synchronous Interface Module (SSI) | 63 |

|        |                              |    |
|--------|------------------------------|----|
| 3.13   | Timer Interface Module (TIM) | 63 |
| 3.13.1 | Wait Mode                    | 63 |
| 3.13.2 | Stop Mode                    | 63 |
| 3.14   | Timebase Module (TBM)        | 63 |
| 3.14.1 | Wait Mode                    | 63 |
| 3.14.2 | Stop Mode                    | 64 |
| 3.15   | Exiting Wait Mode            | 64 |
| 3.16   | Exiting Stop Mode            | 65 |

## 3.2 Introduction

The MCU may enter two low-power modes: wait mode and stop mode. They are common to all HC08 MCUs and are entered through instruction execution. This section describes how each module acts in the low-power modes.

### 3.2.1 Wait Mode

The WAIT instruction puts the MCU in a low-power standby mode in which the CPU clock is disabled but the bus clock continues to run. Power consumption can be further reduced by disabling the low-voltage inhibit (LVI) module through bits in the configuration (CONFIG) register. (See [Section 9. Configuration Register \(CONFIG\)](#).)

### 3.2.2 Stop Mode

Stop mode is entered when a STOP instruction is executed. The CPU clock and the bus clock are disabled but the oscillator itself does not stop, continuing to feed clock to the timebase module.

### 3.3 A/D Converter (ADC4)

The analog-to-digital (A/D) converter module will not work in wait and stop modes. To achieve low-power consumption, it is recommended that the ADC4 module be disabled before entering wait or stop modes.

### 3.4 Break Module (BRK)

#### 3.4.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if the BW bit in the break status register is set.

#### 3.4.2 Stop Mode

The break module is inactive in stop mode. A break interrupt causes exit from stop mode and sets the BW bit in the break status register. The STOP instruction does not affect break module register states.

### 3.5 Central Processor Unit (CPU)

#### 3.5.1 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

### 3.5.2 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.

## 3.6 Clock Generator Module (CGM)

### 3.6.1 Wait Mode

The CGM remains active in wait mode. Before entering wait mode, software can disengage and turn off the phase-locked loop (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.

### 3.6.2 Stop Mode

The STOP instruction disables the PLL but the oscillator will continue to operate.

If the STOP instruction is executed with the VCO clock, CGMVCLK, divided by two driving CGMOUT, the PLL automatically clears the BCS bit in the PLL control register (PCTL), thereby selecting the crystal clock, CGMXCLK, divided by two as the source of CGMOUT. When the MCU recovers from STOP, the crystal clock divided by two drives CGMOUT and BCS remains clear.

### 3.7 Closed-Caption Data Slicer Module (DSL)

The DSL remains active during wait mode and stop mode; however, it will be unable to interrupt the CPU and bring it out of these modes. It is recommended that the DSL be disabled during wait mode or stop mode.

### 3.8 Computer Operating Properly Module (COP)

#### 3.8.1 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.

#### 3.8.2 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 CONFIG register enables the STOP instruction. To prevent inadvertently turning off the COP with a STOP instruction, disable the STOP instruction by clearing the STOP bit.

### 3.9 External Interrupt Module (IRQ)

#### 3.9.1 Wait Mode

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

### 3.9.2 Stop Mode

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

## 3.10 Low-Voltage Inhibit Module (LVI)

### 3.10.1 Wait Mode

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

### 3.10.2 Stop Mode

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

## 3.11 On-Screen Display Module (OSD)

### 3.11.1 Wait Mode

The OSD remains active during wait mode, but it will be unable to interrupt the CPU and bring it out of this mode. It is recommended that the OSD and PLL be disabled before entering wait mode, unless a single row of fixed video output is desired to be displayed constantly.

### 3.11.2 Stop Mode

Although the OSD module and the PLL are not automatically disabled in stop mode, the FLASH memory will be disabled. As a consequence, the OSD module will not work. It is recommended that the OSD and PLL be disabled before entering stop mode.

## 3.12 Serial Synchronous Interface Module (SSI)

In wait mode or stop mode, the SSI halts operation. Pins SDA1, SCL1, SDA2, and SCL2 will maintain their states.

If the SSI is nearing completion of a transfer when wait mode or stop mode is entered, it is possible for the SSI to generate an interrupt request and thus cause the processor to exit wait mode or stop mode immediately. To prevent this occurrence, the programmer should ensure that all transfers are complete before entering wait mode or stop mode.

## 3.13 Timer Interface Module (TIM)

### 3.13.1 Wait Mode

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

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

### 3.13.2 Stop Mode

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

## 3.14 Timebase Module (TBM)

### 3.14.1 Wait Mode

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

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

### 3.14.2 Stop Mode

The timebase module remains active after execution of the STOP instruction. The timebase module can be used in this mode to generate a periodic wakeup from stop mode. In stop mode, the timebase register is not accessible by the CPU.

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

### 3.15 Exiting Wait Mode

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

- External reset — A logic 0 on the  $\overline{\text{RST}}$  pin resets the MCU and loads the program counter with the contents of \$FFFE and \$FFFF.
- External interrupt — A high-to-low transition on the external interrupt pin ( $\overline{\text{IRQ}}$ ) loads the program counter with the contents of \$FFFA and \$FFFB.
- Break interrupt — A break interrupt loads the program counter with the contents of \$FFFC and \$FFFD.
- Computer operating properly module (COP) reset — A timeout of the COP counter resets the MCU and loads the program counter with the contents of \$FFFE and \$FFFF.
- Low-voltage inhibit module (LVI) reset — A power supply voltage below the  $V_{\text{TRIPF}}$  voltage resets the MCU and loads the program counter with the contents of \$FFFE and \$FFFF.
- Clock generator module (CGM) interrupt — A CPU interrupt request from the phase-locked loop (PLL) loads the program counter with the contents of \$FFF8 and \$FFF9.



- Timer interface module (TIM) interrupt — A CPU interrupt request from the TIM loads the program counter with the contents of:
  - \$FFF0 and \$FFF1; TIM1 overflow
  - \$FFF2 and \$FFF3; TIM1 channel 1
  - \$FFF4 and \$FFF5; TIM1 channel 0
- Timebase module (TBM) interrupt — A CPU interrupt request from the TBM loads the program counter with the contents of \$FFF6 and \$FFF7.

### 3.16 Exiting Stop Mode

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

- External reset — A logic 0 on the  $\overline{\text{RST}}$  pin resets the MCU and loads the program counter with the contents of \$FFFE and \$FFFF.
- External interrupt — A high-to-low transition on  $\overline{\text{IRQ}}$  pin loads the program counter with the contents of \$FFFA and \$FFFB.
- Low-voltage inhibit (LVI) reset — A power supply voltage below the  $\text{LVI}_{\text{TRIPF}}$  voltage resets the MCU and loads the program counter with the contents of \$FFFE and \$FFFF.
- Break interrupt — A break interrupt loads the program counter with the contents of \$FFFC and \$FFFD.
- Timebase module (TBM) interrupt — A TBM interrupt loads the program counter with the contents of \$FFF6 and \$FFF7 when the timebase counter has rolled over. This allows the TBM to generate a periodic wakeup from stop mode.

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

The short stop recovery bit, SSREC, in the configuration register controls the oscillator stabilization delay during stop recovery. Setting

SSREC reduces stop recovery time from 4096 CGMXCLK cycles to 32 CGMXCLK cycles.

**NOTE:** *Use the full stop recovery time (SSREC = 0) in applications that use an external crystal.*

## Section 4. Resets and Interrupts

### 4.1 Contents

|         |  |    |
|---------|--|----|
| 4.2     | Introduction   | 68 |
| 4.3     | Resets   | 68 |
| 4.3.1   | Effects  | 68 |
| 4.3.2   | External Reset   | 68 |
| 4.3.3   | Internal Reset   | 68 |
| 4.3.3.1 | Power-On Reset (POR)                                     | 69 |
| 4.3.3.2 | Computer Operating Properly (COP) Reset                  | 70 |
| 4.3.3.3 | Low-Voltage Inhibit (LVI) Reset                          | 70 |
| 4.3.3.4 | Illegal Opcode Reset                                     | 71 |
| 4.3.3.5 | Illegal Address Reset                                    | 71 |
| 4.3.4   | System Integration Module (SIM) Reset<br>Status Register | 71 |
| 4.4     | Interrupts   | 73 |
| 4.4.1   | Effects  | 73 |
| 4.4.2   | Sources  | 76 |
| 4.4.2.1 | SWI Instruction  | 76 |
| 4.4.2.2 | Break Interrupt  | 76 |
| 4.4.2.3 | $\overline{\text{IRQ}}$ Pin                              | 77 |
| 4.4.2.4 | Clock Generator Module (CGM)                             | 77 |
| 4.4.2.5 | Timebase Module (TBM)                                    | 77 |
| 4.4.2.6 | Timer (TIM)  | 77 |
| 4.4.2.7 | On-Screen Display (OSD)                                  | 78 |
| 4.4.2.8 | Data Slicer (DSL) Module                                 | 78 |
| 4.4.2.9 | Serial Synchronous Interface (SSI) Module                | 78 |
| 4.4.3   | Interrupt Status Registers                               | 79 |
| 4.4.3.1 | Interrupt Status Register 1                              | 80 |
| 4.4.3.2 | Interrupt Status Register 2                              | 80 |

## 4.2 Introduction

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

## 4.3 Resets

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

### 4.3.1 Effects

A reset:

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

### 4.3.2 External Reset

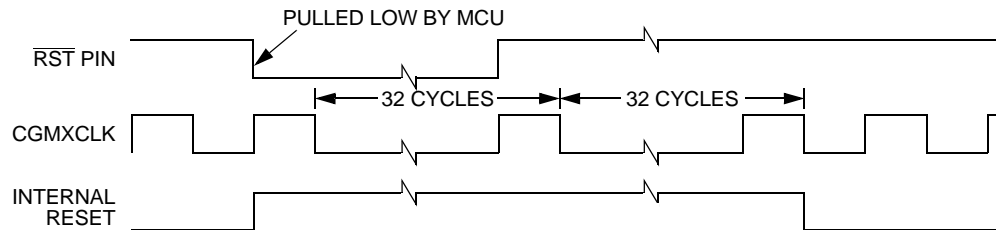
A logic 0 applied to the  $\overline{\text{RST}}$  pin for a time,  $t_{\text{IRL}}$ , generates an external reset. An external reset sets the PIN bit in the system integration module (SIM) reset status register.

### 4.3.3 Internal Reset

Sources:

- Power-on reset (POR)
- Computer operating properly (COP)
- Low-voltage inhibit (LVI)
- Illegal opcode
- Illegal address

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



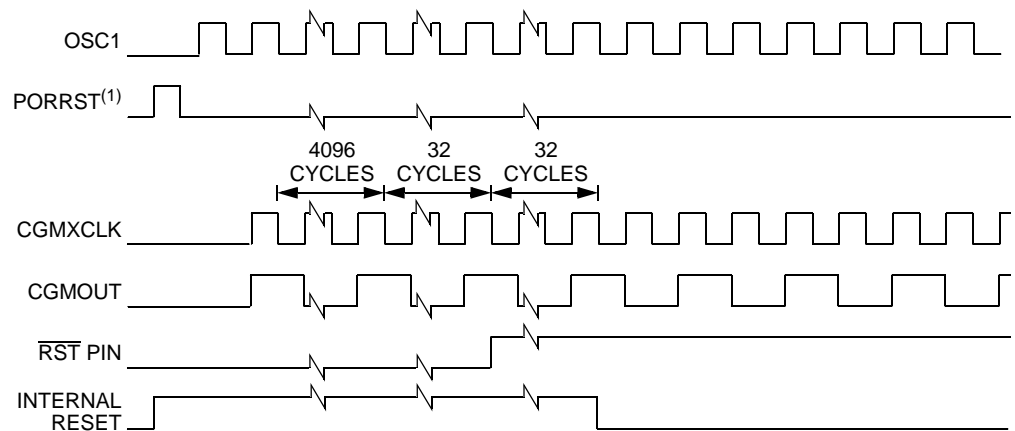
**Figure 4-1. Internal Reset Timing**

#### 4.3.3.1 Power-On Reset (POR)

A power-on reset (POR) is an internal reset caused by a positive transition on the  $V_{DD}$  pin.  $V_{DD}$  at the POR must go completely to 0 V to reset the MCU. This distinguishes between a reset and a POR. The POR is not a brown-out detector, low-voltage detector, or glitch detector.

A power-on reset:

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



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

**Figure 4-2. Power-On Reset Recovery**

### 4.3.3.2 Computer Operating Properly (COP) Reset

A COP reset is an internal reset caused by an overflow of the COP counter. A COP reset sets the COP bit in the system integration module (SIM) reset status register.

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

### 4.3.3.3 Low-Voltage Inhibit (LVI) Reset

An LVI reset is an internal reset caused by a drop in the power supply voltage to the  $LVI_{TRIPF}$  voltage.

An LVI reset:

- Holds the clocks to the CPU and modules inactive for an oscillator stabilization delay of 4096 CGMXCLK cycles after the power supply voltage rises to the  $LVI_{TRIPR}$  voltage
- Drives the  $\overline{RST}$  pin low for as long as  $V_{DD}$  is below the  $LVI_{TRIPR}$  voltage and during the oscillator stabilization delay
- Releases the  $\overline{RST}$  pin 32 CGMXCLK cycles after the oscillator stabilization delay

- Releases the CPU to begin the reset vector sequence  
64 CGMXCLK cycles after the oscillator stabilization delay
- Sets the LVI bit in the SIM reset status register

#### 4.3.3.4 Illegal Opcode Reset

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

If the stop enable bit, STOP, in the mask option register (MOR) is a logic 0, the STOP instruction causes an illegal opcode reset.

#### 4.3.3.5 Illegal Address Reset

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

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

### 4.3.4 System Integration Module (SIM) Reset Status Register


The SIM reset status register (SRSR) is a read-only register containing flags to show reset sources. All flag bits are automatically cleared following a read of the register. Reset service can read the SIM reset status register to clear the register after power-on reset and to determine the source of any subsequent reset.

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

**NOTE:** Only a read of the SIM reset status register clears all reset flags. After multiple resets from different sources without reading the register, multiple flags remain set.

Address: \$FE01

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

 = Unimplemented

**Figure 4-3. SIM Reset Status Register (SRSR)**

**POR** — Power-On Reset Flag Bit

- 1 = Power-on reset since last read of SRSR
- 0 = Read of SRSR since last power-on reset

**PIN** — External Reset Flag Bit

- 1 = External reset via  $\overline{RST}$  pin since last read of SRSR
- 0 = POR or read of SRSR since last external reset

**COP** — Computer Operating Properly Reset Bit

- 1 = Last reset caused by timeout of 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

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

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

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



## 4.4 Interrupts

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

### 4.4.1 Effects

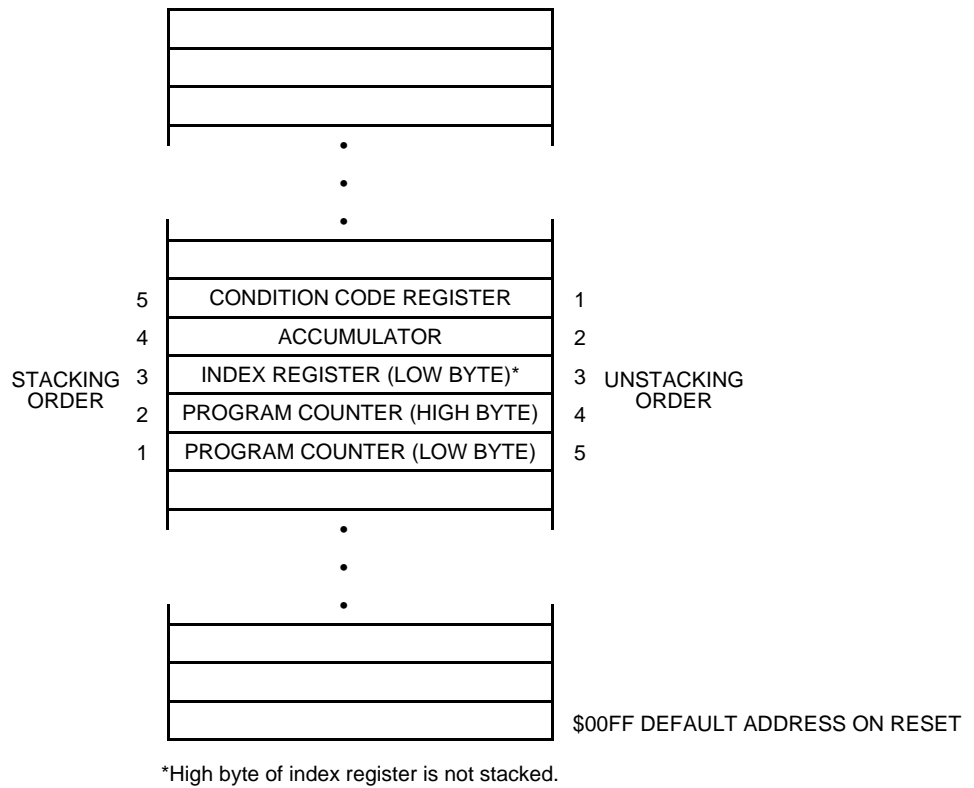
An interrupt:

- Saves the CPU registers on the stack. At the end of the interrupt, the RTI instruction recovers the CPU registers from the stack so that normal processing can resume. (See [Figure 4-4](#).)
- Sets the interrupt mask (I bit) to prevent additional interrupts. Once an interrupt is latched, no other interrupt can take precedence, regardless of its priority.
- Loads the program counter with a user-defined vector address

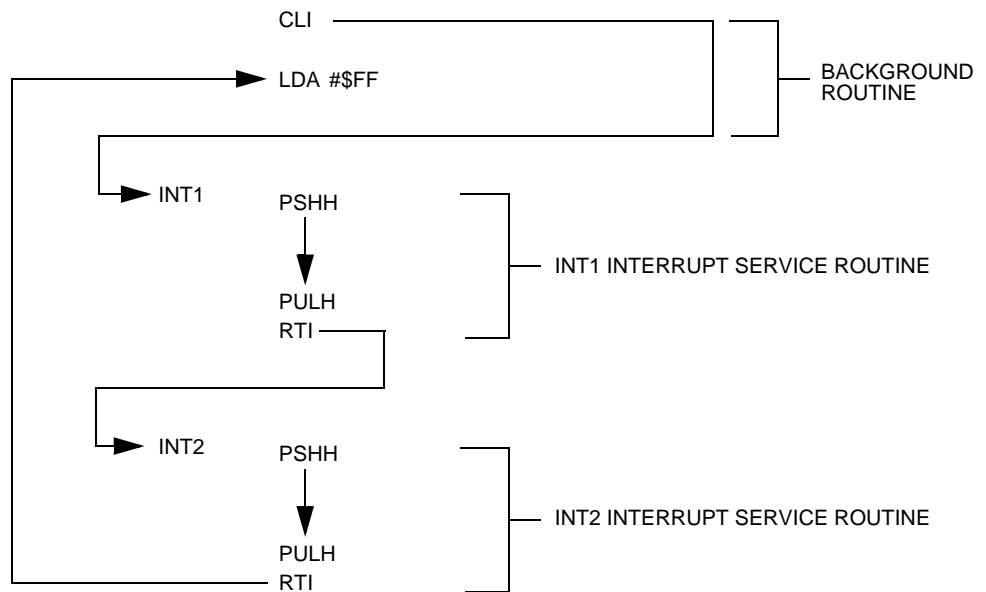
After every instruction, the CPU checks all pending interrupts if the I bit is not set. If more than one interrupt is pending when an instruction is done, the highest priority interrupt is serviced first. In the example shown in [Figure 4-5](#), if an interrupt is pending upon exit from the interrupt service routine, the pending interrupt is serviced before the load accumulator (LDA) instruction is executed.

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

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



**Figure 4-4. Interrupt Stacking Order**



**Figure 4-5. Interrupt Recognition Example**

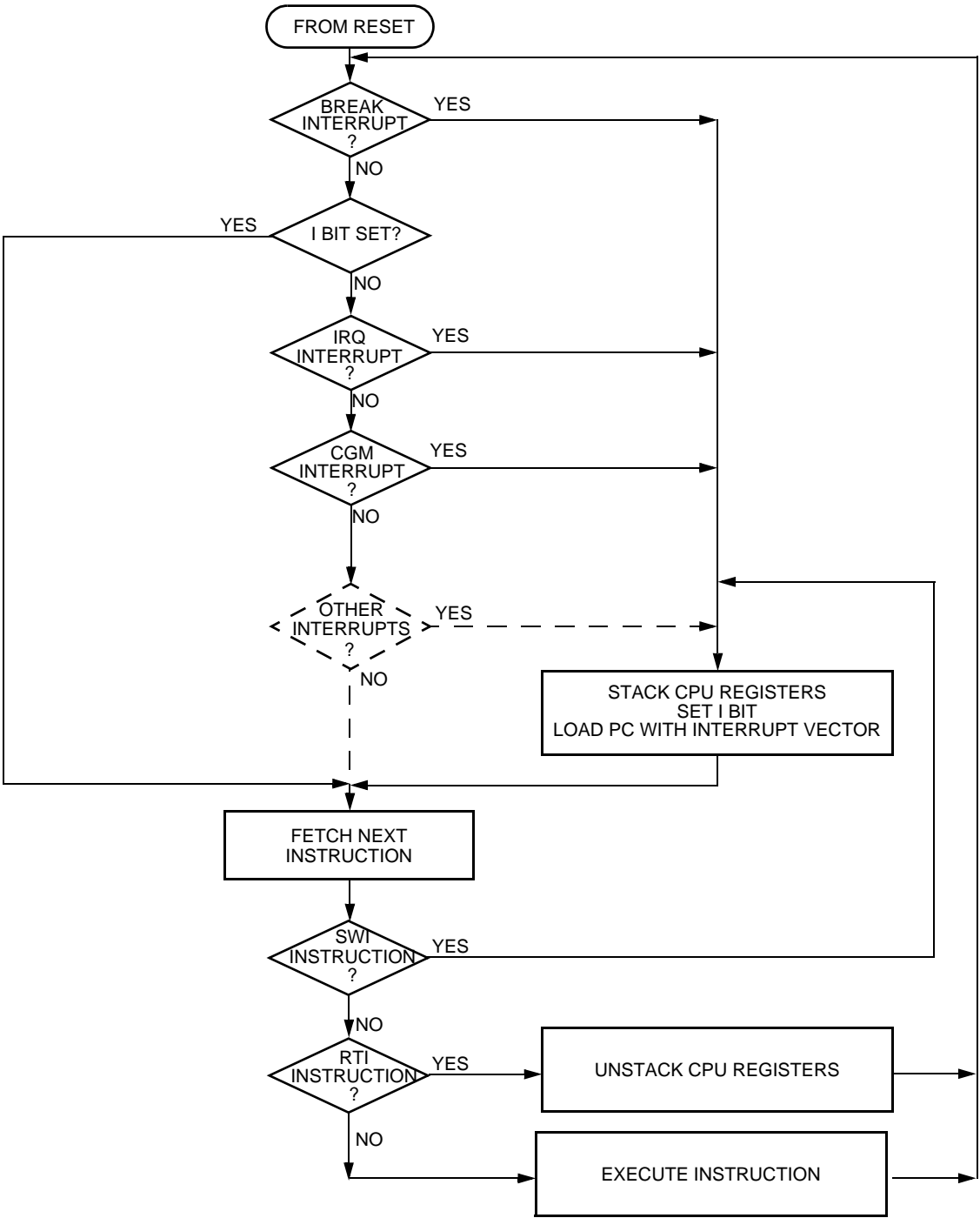


Figure 4-6. Interrupt Processing

## 4.4.2 Sources

The sources in [Table 4-1](#) can generate CPU interrupt requests.

**Table 4-1. Interrupt Sources**

| Source                       | Flag  | Mask <sup>(1)</sup> | INT Register Flag | Priority <sup>(2)</sup> | Vector Address |
|------------------------------|-------|---------------------|-------------------|-------------------------|----------------|
| Reset                        | None  | None                | None              | 0                       | \$FFFE–\$FFFF  |
| SWI instruction              | None  | None                | None              | 0                       | \$FFFC–\$FFFD  |
| IRQ pin                      | IRQF  | IMASK1              | IF1               | 1                       | \$FFFA–\$FFFB  |
| CGM (PLL)                    | PLLF  | PLLIE               | IF2               | 2                       | \$FFF8–\$FFF9  |
| Timebase                     | TBIF  | TBIE                | IF3               | 3                       | \$FFF6–\$FFF7  |
| TIM channel 0                | CH0F  | CH0IE               | IF4               | 4                       | \$FFF4–\$FFF5  |
| TIM channel 1                | CH1F  | CH1IE               | IF5               | 5                       | \$FFF2–\$FFF3  |
| TIM overflow                 | TOF   | TOIE                | IF6               | 6                       | \$FFF0–\$FFF1  |
| OSD event line match         | ELMF  | ELIEN               | IF7               | 7                       | \$FFEE–\$FFEF  |
| OSD vertical sync pulse      | VSINF | VSIEN               |                   |                         |                |
| Closed-caption data slicer   | DSFL  | DSIEN               | IF8               | 8                       | \$FFEC–\$FFED  |
| Serial synchronous interface | SF    | SIE                 | IF9               | 9                       | \$FFEA–\$FFEB  |

Notes:

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

### 4.4.2.1 SWI Instruction

The software interrupt instruction (SWI) causes a non-maskable interrupt.

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

### 4.4.2.2 Break Interrupt

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

#### 4.4.2.3 $\overline{IRQ}$ Pin

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

#### 4.4.2.4 Clock Generator Module (CGM)

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

#### 4.4.2.5 Timebase Module (TBM)

The timebase module (TBM) can interrupt the CPU on a regular basis with a rate defined by TBR2–TBR0. When the timebase counter chain rolls over, the TBIF flag is set. If the TBIE bit is set, enabling the timebase interrupt, the counter chain overflow will generate a CPU interrupt request.

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

#### 4.4.2.6 Timer (TIM)

Timer (TIM) CPU interrupt sources:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter value rolls over to \$0000 after matching the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH1F–CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. The channel x interrupt enable bit, CHxIE, enables channel x TIM CPU interrupt requests. CHxF and CHxIE are in the TIM channel x status and control register.

### 4.4.2.7 On-Screen Display (OSD)

On-screen display (OSD) CPU interrupt sources:

- Event line match flag (ELMF) — This flag is located on the OSD status register (OSDSR). It is set when the OSD scan line counter matches the value programmed on the event line register (OSDELRL). This flag generates an interrupt if the ELIEN bit of the OSD enable control register (OSDECTR) is set.
- Vertical sync flag (VSINF) — This flag is located on the OSD status register (OSDSR). It is set at every starting edge of VSYNC pulses. This flag generates an interrupt if the VSIEN bit of the OSD enable control register (OSDECTR) is set.

Interrupts must be acknowledged by writing any value to the OSD status register (OSDSR).

### 4.4.2.8 Data Slicer (DSL) Module

The data slicer (DSL) module can interrupt the CPU to indicate that it has sampled 16 bits of closed-caption data (including parity) and has placed them in registers DSLCH1 and DSLCH2. When this happens, the DSFL bit of the DSL status register (DSLRSR) is asserted and an interrupt is issued if the DSIEN bit on the DSL control register 1 (DSLCR1) is set.

Interrupts must be acknowledged by writing any value to the DSL status register (OSDSR).

### 4.4.2.9 Serial Synchronous Interface (SSI) Module

The serial synchronous interface (SSI) module can interrupt the CPU to indicate that one complete transfer has occurred. When this happens, the SF bit of the SSI status register (SSISR) is asserted and an interrupt is issued if the SIE bit of the SSI control register (SSICR) is set.

The SF flag must always be cleared between transfers. This can be done in three ways:

1. By reading the SSI status register with SF set, followed by writing or reading the SSI data register (SSIDR)
2. By a reset
3. By disabling the SSI

**NOTE:** *If the SF flag is cleared by resetting or disabling the SSI before issuing a STOP bit, the slave device may enter into an indeterminate state. The first method of clearing SF is always the best.*

### 4.4.3 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. [Table 4-2](#) summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

**Table 4-2. Interrupt Source Flags**

| Interrupt Source             | Interrupt Status Register Flag |
|------------------------------|--------------------------------|
| Reset                        | —                              |
| SWI instruction              | —                              |
| $\overline{\text{IRQ}}$ pin  | IF1                            |
| CGM (PLL)                    | IF2                            |
| Timebase                     | IF3                            |
| TIM channel 0                | IF4                            |
| TIM channel 1                | IF5                            |
| TIM overflow                 | IF6                            |
| On-screen display            | IF7                            |
| Closed-caption data slicer   | IF8                            |
| Serial synchronous interface | IF9                            |

## 4.4.3.1 Interrupt Status Register 1

Address: \$FE04

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

R = Reserved

**Figure 4-7. Interrupt Status Register 1 (INT1)**

### IF6–IF1 — Interrupt Flags 6–1

These flags indicate the presence of interrupt requests from the sources shown in [Table 4-2](#).

1 = Interrupt request present

0 = No interrupt request present

Bit 1 and Bit 0 — Always read 0

## 4.4.3.2 Interrupt Status Register 2

Address: \$FE05

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

R = Reserved

**Figure 4-8. Interrupt Status Register 2 (INT2)**

### IF9–IF7 — Interrupt Flags 9–7

These flags indicate the presence of interrupt requests from the sources shown in [Table 4-2](#).

1 = Interrupt request present

0 = No interrupt request present

Bits 7–3 — Always read 0



## Section 5. Analog-to-Digital Converter (ADC4)

### 5.1 Contents

|       |   |    |
|-------|---|----|
| 5.2   | Introduction . . . . .                    | 81 |
| 5.3   | Features . . . . .                        | 81 |
| 5.4   | Overview . . . . .                        | 82 |
| 5.5   | Programming Guidelines . . . . .          | 82 |
| 5.5.1 | Setup . . . . .                           | 82 |
| 5.5.2 | Conversions . . . . .                     | 83 |
| 5.5.3 | Input/Output . . . . .                    | 83 |
| 5.6   | ADC Control and Status Register . . . . . | 84 |
| 5.7   | Low-Power Modes . . . . .                 | 85 |
| 5.8   | Interrupts and Resets . . . . .           | 85 |

### 5.2 Introduction

The analog-to-digital converter (ADC4) performs individual analog comparisons that can be used together with a software algorithm to obtain an analog-to-digital conversion.

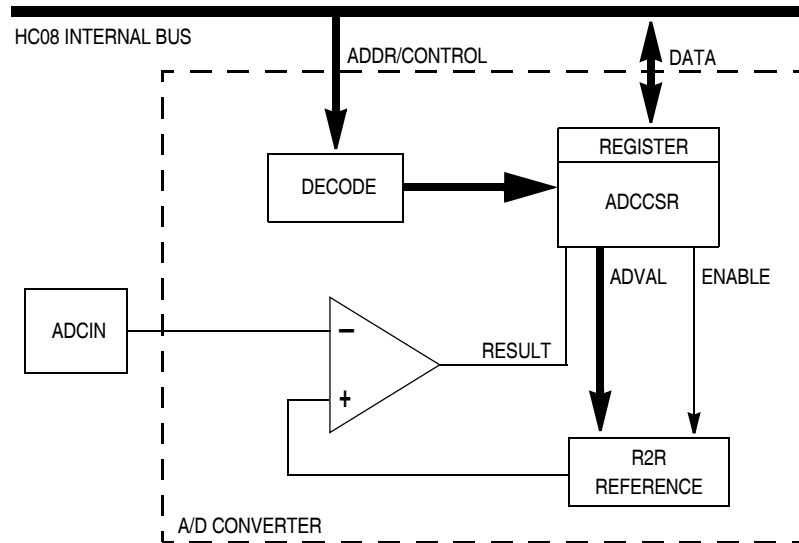
### 5.3 Feature

The ADC4 provides the following feature:

- 4-bit software analog-to-digital conversion

## 5.4 Overview

The ADC4 system consists of a single 4-bit analog-to-digital (A/D) converter and comparator with continuous conversion. A result flag indicates if the comparator output is above or below the analog input, ADCIN. (See [Figure 5-1](#).)



**Figure 5-1. ADC4 Block Diagram**

## 5.5 Programming Guidelines

The following subsections describe programming guidelines.

### 5.5.1 Setup

The ADC4 must be enabled by setting the ADON bit in the ADC control and status register (ADCCSR).

## 5.5.2 Conversions

An A/D conversion can be performed with the aid of a software algorithm. **Figure 5-2** shows an example of a conversion code.

```

        LDA#   #$40      ; Enable ADC
        STA   ADCCSR    ; and set initial A/D value = 00
        INC   ATD       ; No real function other than
        DEC   ATD       ; having the required initial delay
DTA:    STA   ATD       ; Save A/D value
        LDA   ADCCSR
        AND   #$8F     ; Read comparator result and A/D value
        CMP   #$0F     ; Check if reached analog pin
                        ; value or A/D value is maximum
        BGE   ENDC     ; If ok, end of conversion
        INC   ADCCSR   ; If not, increment A/D value
        BRA   DTA      ; return
ENDC:   ...

ATD     ...           ; analog value in ADC
                        ; AnalogIn=(ADC+1)*0.3125V with VDD=5V
  
```

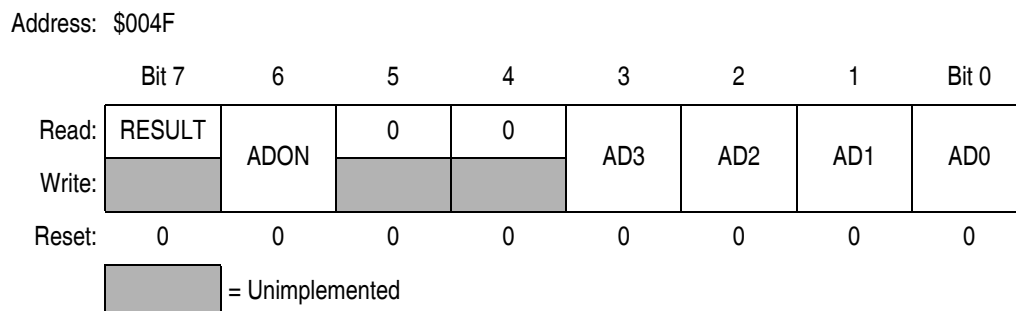
**Figure 5-2. ADC Conversion Code Example**

## 5.5.3 Input/Output

The ADC4 has one input pin (ADCIN), which is the analog input to be converted.

## 5.6 ADC Control and Status Register

The ADC control and status register (ADCCSR) contains all of the ADC4 status and control bits.



**Figure 5-3. ADC Control and Status Register (ADCCSR)**

### RESULT — Comparator Result Bit

This bit indicates the relationship of the analog input to the analog version of the AD3–AD0 value. A reset has no effect on this bit.

1 = D/A output  $\geq$  ANALOG IN

0 = D/A output  $<$  ANALOG IN

### ADON — ADC On Bit

This bit indicates whether the ADC is enabled. When enabled, the ADC supplies power to the D/A resistive ladder. A reset clears this bit.

1 = ADC enabled

0 = ADC disabled

**NOTE:** *If not in use, the input should be tied to  $V_{SS}$  and a value of \$00 should be written to the ADC control and status register.*

### AD3:AD0 — ADC Comparison Value Bits

These bits are controlled by the user to perform a successive approximation conversion in software. When a value causes the RESULT bit to change state from the value immediately before or after it, AD3:AD0 are considered to be the digital equivalent of the analog input. A reset clears these bits.

## 5.7 Low-Power Modes

The ADC4 will not work in wait mode and stop mode. To achieve low-power consumption, it is recommended that the ADC4 be disabled before entering wait mode or stop mode.

## 5.8 Interrupts and Resets

The ADC4 does not interrupt or reset the CPU.



## Section 6. Break Module (BRK)

### 6.1 Contents

|       |   |    |
|-------|---|----|
| 6.2   | Introduction . . . . .                            | 87 |
| 6.3   | Features . . . . .                                | 88 |
| 6.4   | Functional Description . . . . .                  | 88 |
| 6.4.1 | Flag Protection During Break Interrupts . . . . . | 90 |
| 6.4.2 | CPU During Break Interrupts . . . . .             | 90 |
| 6.4.3 | TIM During Break Interrupts . . . . .             | 90 |
| 6.4.4 | COP During Break Interrupts . . . . .             | 90 |
| 6.5   | Low-Power Modes . . . . .                         | 90 |
| 6.5.1 | Wait Mode . . . . .                               | 91 |
| 6.5.2 | Stop Mode . . . . .                               | 91 |
| 6.6   | Break Module Registers . . . . .                  | 91 |
| 6.6.1 | Break Status and Control Register . . . . .       | 92 |
| 6.6.2 | Break Address Registers . . . . .                 | 93 |
| 6.6.3 | SIM Break Status Register . . . . .               | 94 |
| 6.6.4 | SIM Break Flag Control Register . . . . .         | 95 |

### 6.2 Introduction

This section describes the break module (BRK). The break module can generate a break interrupt that stops normal program flow at a defined address to enter a background program.

### 6.3 Features

Features of the break module include:

- Accessible input/output (I/O) registers during the break interrupt
- CPU-generated break interrupts
- Software-generated break interrupts
- COP disabling during break interrupts

### 6.4 Functional Description

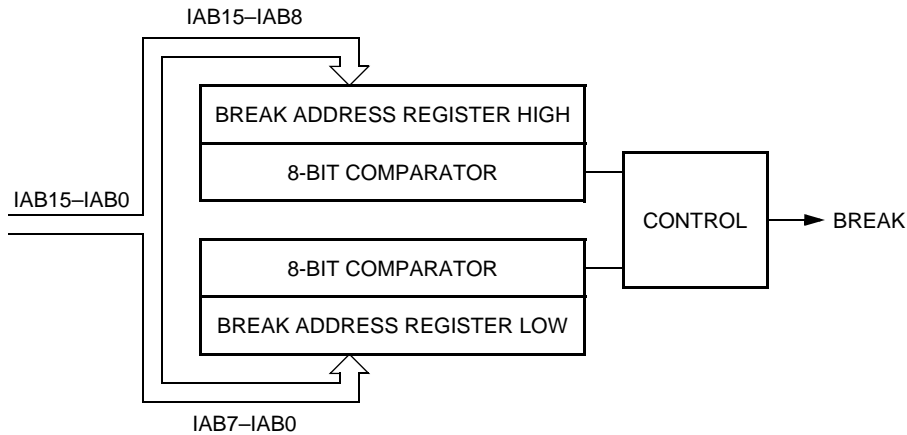
When the internal address bus matches the value written in the break address registers, the break module issues a breakpoint signal to the CPU. The CPU then loads the instruction register with a software interrupt instruction (SWI) after completion of the current CPU instruction. The program counter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

The following events can cause a break interrupt to occur:

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

When a CPU-generated address matches the contents of the break address registers, the break interrupt begins after the CPU completes its current instruction. A return-from-interrupt instruction (RTI) in the break routine ends the break interrupt and returns the MCU to normal operation. [Figure 6-1](#) shows the structure of the break module.





**Figure 6-1. Break Module Block Diagram**

| Addr.  | Register Name  | Bit 7  | 6      | 5    | 4  | 3  | 2  | 1  | Bit 0 |       |
|--------|--|--------|--------|------|----|----|----|----|-------|-------|
| \$FE00 | SIM Break Status Register (SBSR)<br>See page 94.           | Read:  | 0      | 0    | 0  | 1  | 0  | 0  | BW    | 0     |
|        |  | Write: | R      | R    | R  | R  | R  | R  | NOTE  | R     |
|        |  | Reset: | 0      | 0    | 0  | 1  | 0  | 0  | 0     | 0     |
| \$FE03 | SIM Break Flag Control Register (SBFCR)<br>See page 95.    | Read:  | BCFE   | R    | R  | R  | R  | R  | R     | R     |
|        |  | Write: |        |      |    |    |    |    |       |       |
|        |  | Reset: | 0      |      |    |    |    |    |       |       |
| \$FE0C | Break Address Register High (BRKH)<br>See page 93.         | Read:  | Bit 15 | 14   | 13 | 12 | 11 | 10 | 9     | Bit 8 |
|        |  | Write: |        |      |    |    |    |    |       |       |
|        |  | Reset: | 0      | 0    | 0  | 0  | 0  | 0  | 0     | 0     |
| \$FE0D | Break Address Register Low (BRKL)<br>See page 93.          | Read:  | Bit 7  | 6    | 5  | 4  | 3  | 2  | 1     | Bit 0 |
|        |  | Write: |        |      |    |    |    |    |       |       |
|        |  | Reset: | 0      | 0    | 0  | 0  | 0  | 0  | 0     | 0     |
| \$FE0E | Break Status and Control Register (BRKSCR)<br>See page 92. | Read:  | BRKE   | BRKA | 0  | 0  | 0  | 0  | 0     | 0     |
|        |  | Write: |        |      |    |    |    |    |       |       |
|        |  | Reset: | 0      | 0    | 0  | 0  | 0  | 0  | 0     | 0     |

Note: Writing a logic 0 clears BW.   = Unimplemented R = Reserved

**Figure 6-2. I/O Register Summary**

### 6.4.1 Flag Protection During Break Interrupts

The BCFE bit in the system integration module (SIM) break flag control register (SBFCR) enables software to clear status bits during the break state.

### 6.4.2 CPU During Break Interrupts

The CPU starts a break interrupt by:

- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC and \$FFFD (\$FEFC and \$FEFD in monitor mode)

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

### 6.4.3 TIM During Break Interrupts

A break interrupt stops the timer counters.

### 6.4.4 COP During Break Interrupts

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

## 6.5 Low-Power Modes

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

### 6.5.1 Wait Mode

If enabled, the break module is active in wait mode. In the break routine, the user can subtract one from the return address on the stack if SBSW is set. See [Section 3. Low-Power Modes](#). Clear the BW bit by writing logic 0 to it.

### 6.5.2 Stop Mode

A break interrupt causes exit from stop mode and sets the SBSW bit in the break status register.

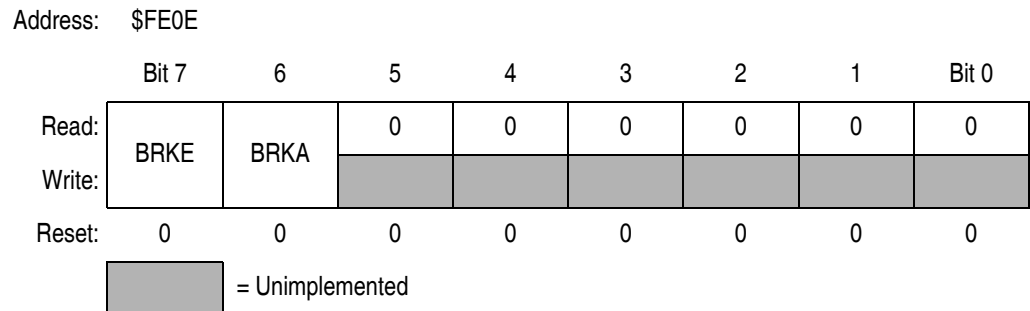
## 6.6 Break Module Registers

These registers control and monitor operation of the break module:

- Break status and control register (BRKSCR)
- Break address register high (BRKH)
- Break address register low (BRKL)
- SIM break status register (SBSR)
- SIM break flag control register (SBFCR)

## 6.6.1 Break Status and Control Register

The break status and control register (BRKSCR) contains break module enable and status bits.



**Figure 6-3. Break Status and Control Register (BRKSCR)**

### BRKE — Break Enable Bit

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

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

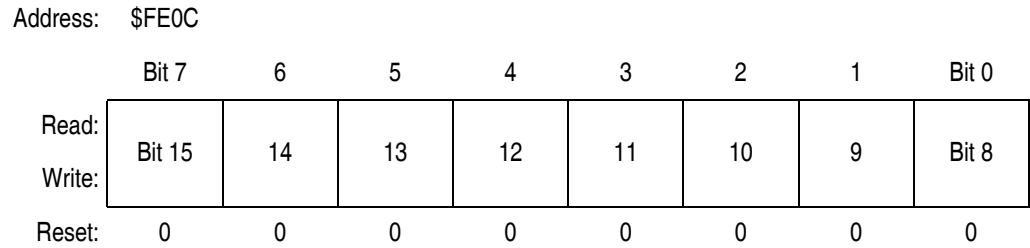
### BRKA — Break Active Bit

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

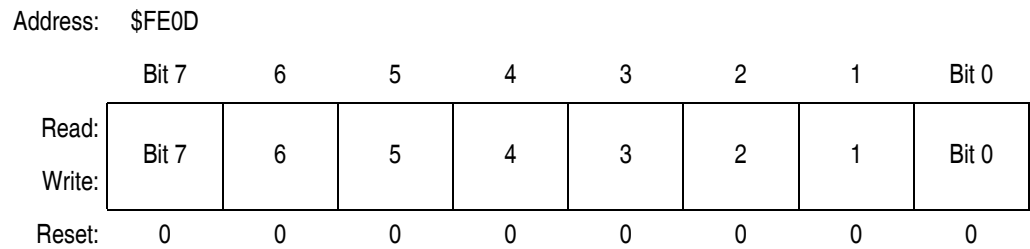
- 1 = When read, break address match
- 0 = When read, no break address match

## 6.6.2 Break Address Registers

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



**Figure 6-4. Break Address Register High (BRKH)**



**Figure 6-5. Break Address Register Low (BRKL)**

## 6.6.3 SIM Break Status Register

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

Address: \$FE00

|        | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1    | Bit 0 |
|--------|-------|---|---|---|---|---|------|-------|
| Read:  | 0     | 0 | 0 | 1 | 0 | 0 | BW   | 0     |
| Write: | R     | R | R | R | R | R | Note | R     |
| Reset: | 0     | 0 | 0 | 1 | 0 | 0 | 0    | 0     |

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

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

### BW — Break Wait Bit

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

1 = Break interrupt during wait mode

0 = No break interrupt during wait mode

BW can be read within the break interrupt routine. The user can modify the return address on the stack by subtracting 1 from it. The example code shown in [Figure 6-7](#) works if the H register was stacked in the break interrupt routine. Execute this code at the end of the break interrupt routine.

```

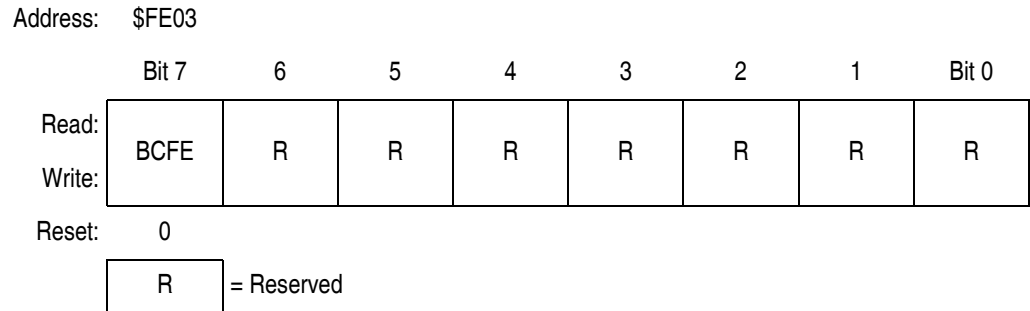
HIBYTE EQU    5
LOBYTE EQU    6

                                ; If not BW, do RTI
BRCLR BW,BSR, RETURN           ; See if wait mode or stop
                                ; mode was exited by break.
TST  LOBYTE,SP                 ; If RETURNLO is not 0,
BNE  DOLO                      ; then just decrement low byte.
DEC  HIBYTE,SP                 ; Else deal with high byte also.
DOLO DEC  LOBYTE,SP            ; Point to WAIT/STOP opcode.
RETURN PULH                     ; Restore H register.
RTI
    
```

**Figure 6-7. Example Code**

### 6.6.4 SIM Break Flag Control Register

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



**Figure 6-8. 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





## Section 7. Clock Generator Module (CGMC)

### 7.1 Contents

|       |   |     |
|-------|---|-----|
| 7.2   | Introduction . . . . .                              | 98  |
| 7.3   | Features . . . . .                                  | 98  |
| 7.4   | Functional Description . . . . .                    | 99  |
| 7.4.1 | Crystal Oscillator Circuit . . . . .                | 99  |
| 7.4.2 | Phase-Locked Loop Circuit (PLL) . . . . .           | 101 |
| 7.4.3 | PLL Circuits . . . . .                              | 101 |
| 7.4.4 | Acquisition and Tracking Modes . . . . .            | 102 |
| 7.4.5 | Manual and Automatic PLL Bandwidth Modes. . . . .   | 103 |
| 7.4.6 | Programming the PLL . . . . .                       | 104 |
| 7.4.7 | Special Programming Exceptions . . . . .            | 108 |
| 7.4.8 | Base Clock Selector Circuit . . . . .               | 108 |
| 7.4.9 | CGMC External Connections . . . . .                 | 109 |
| 7.5   | Input/Output Signals . . . . .                      | 110 |
| 7.5.1 | Crystal Amplifier Input Pin (OSC1). . . . .         | 110 |
| 7.5.2 | Crystal Amplifier Output Pin (OSC2) . . . . .       | 110 |
| 7.5.3 | External Filter Capacitor Pin (CGMXFC) . . . . .    | 111 |
| 7.5.4 | PLL Analog Power Pin ( $V_{DDCGM}$ ). . . . .       | 111 |
| 7.5.5 | PLL Analog Ground Pin ( $V_{SSCGM}$ ) . . . . .     | 111 |
| 7.5.6 | Crystal Output Frequency Signal (CGMXCLK) . . . . . | 111 |
| 7.5.7 | CGMC Base Clock Output (CGMOUT) . . . . .           | 112 |
| 7.5.8 | CGMC CPU Interrupt (CGMINT) . . . . .               | 112 |
| 7.6   | CGMC Registers. . . . .                             | 112 |
| 7.6.1 | PLL Control Register . . . . .                      | 114 |
| 7.6.2 | PLL Bandwidth Control Register . . . . .            | 117 |
| 7.6.3 | PLL Multiplier Select Register High . . . . .       | 118 |
| 7.6.4 | PLL Multiplier Select Register Low. . . . .         | 119 |
| 7.6.5 | PLL VCO Range Select Register . . . . .             | 120 |
| 7.6.6 | PLL Reference Divider Select Register . . . . .     | 121 |

|       |  |     |
|-------|--|-----|
| 7.7   | Interrupts . . . . .                             | 122 |
| 7.8   | Special Modes . . . . .                          | 122 |
| 7.8.1 | Wait Mode . . . . .                              | 122 |
| 7.8.2 | Stop Mode . . . . .                              | 123 |
| 7.8.3 | CGMC During Break Interrupts . . . . .           | 123 |
| 7.9   | Acquisition/Lock Time Specifications . . . . .   | 123 |
| 7.9.1 | Acquisition/Lock Time Definitions . . . . .      | 124 |
| 7.9.2 | Parametric Influences on Reaction Time . . . . . | 124 |
| 7.9.3 | Choosing a Filter . . . . .                      | 125 |

## 7.2 Introduction

This section describes the clock generator module (CGMC). The CGMC generates the crystal clock signal, CGMXCLK, which operates at the frequency of the crystal. The CGMC also generates the base clock signal, CGMOUT, which is based on either the crystal clock divided by two or the phase-locked loop (PLL) clock, CGMVCLK, divided by two. In user mode, CGMOUT is the clock from which the system integration module (SIM) derives the system clocks, including the bus clock, which is at a frequency of CGMOUT divided by two. In monitor mode, PTC3 determines the bus clock. The PLL is a fully functional frequency generator designed for use with crystals or ceramic resonators. The PLL can generate an 8-MHz bus frequency using a 32-kHz crystal.

## 7.3 Features

Features of the CGMC include:

- Phase-locked loop with output frequency in integer multiples of an integer dividend of the crystal reference
- Low-frequency crystal operation with low-power operation and high-output frequency resolution
- Programmable prescaler for power-of-two increases in frequency
- Programmable hardware voltage-controlled oscillator (VCO) for low-jitter operation

- Automatic bandwidth control mode for low-jitter operation
- Automatic frequency lock detector
- CPU interrupt on entry or exit from locked condition

## 7.4 Functional Description

The CGMC consists of three major sub-modules:

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

**Figure 7-1** shows the structure of the CGMC.

### 7.4.1 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 oscillator circuit works continuously even in stop mode so that modules that receive the crystal clock directly, like the timebase module (TBM), can operate in stop mode.

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 percent 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.

# Clock Generator Module (CGMC)

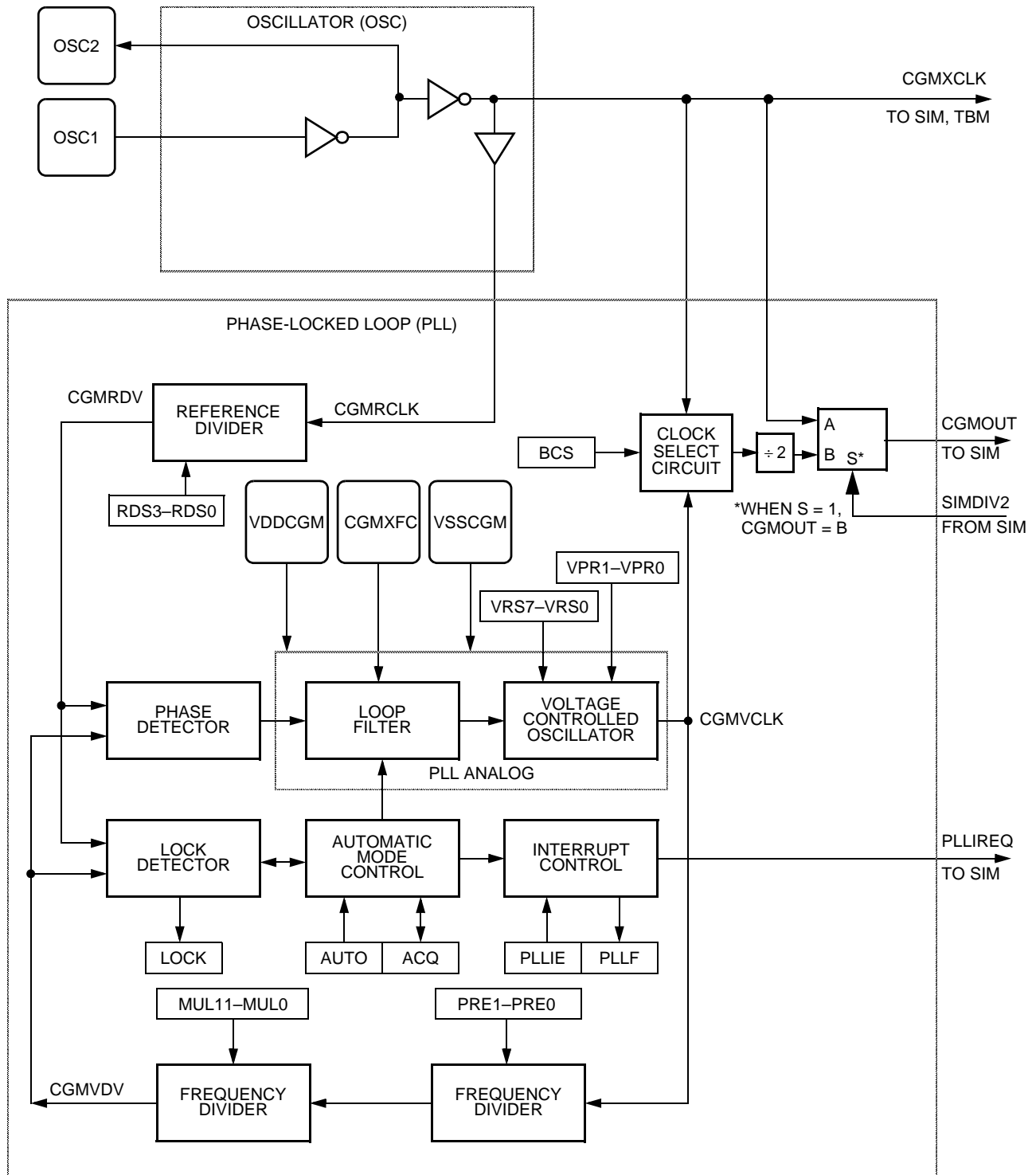


Figure 7-1. CGMC Block Diagram

## 7.4.2 Phase-Locked Loop Circuit (PLL)

The phase-locked loop (PLL) circuit 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.

## 7.4.3 PLL Circuits

The PLL consists of these circuits:

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

The operating range of the VCO is programmable for a wide range of frequencies and for maximum immunity to external noise, including supply and CGM/XFC noise. The VCO frequency is bound to a range from roughly one-half to twice the center-of-range frequency,  $f_{VRS}$ . Modulating the voltage on the CGM/XFC pin changes the frequency within this range. By design,  $f_{VRS}$  is equal to the nominal center-of-range frequency,  $f_{NOM}$ , (38.4 kHz) times a linear factor,  $L$ , and a power-of-two factor,  $E$ , or  $(L \times 2^E)f_{NOM}$ .

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

The VCO's output clock, CGMVCLK, running at a frequency,  $f_{VCLK}$ , is fed back through a programmable prescale divider and a programmable modulo divider. The prescaler divides the VCO clock by a power-of-two factor,  $P$ , and the modulo divider reduces the VCO clock by a factor,  $N$ . The dividers' output is the VCO feedback clock, CGMVDV, running at a frequency,  $f_{VDV} = f_{VCLK}/(N \times 2^P)$ . (See [7.4.6 Programming the PLL](#) for more information.)

The phase detector then compares the VCO feedback clock, CGMVDV, with the final reference clock, CGMRDV. A correction pulse is generated based on the phase difference between the two signals. The loop filter then slightly alters the DC voltage on the external capacitor connected to CGM/XFC based on the width and direction of the correction pulse. The filter can make fast or slow corrections depending on its mode, described in [7.4.4 Acquisition and Tracking Modes](#). The value of the external capacitor and the reference frequency determines the speed of the corrections and the stability of the PLL.

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

### 7.4.4 Acquisition and Tracking Modes

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

- Acquisition mode — In acquisition mode, the filter can make large frequency corrections to the VCO. This mode is used at PLL 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 [7.6.2 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 [7.4.8 Base Clock Selector Circuit](#).) The PLL is automatically in tracking mode when not in acquisition mode or when the  $\overline{ACQ}$  bit is set.

### 7.4.5 Manual and Automatic PLL Bandwidth Modes

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

In automatic bandwidth control mode (AUTO = 1), the lock detector automatically switches between acquisition and tracking modes. Automatic bandwidth control mode also is used to determine when the VCO clock, CGMVCLK, is safe to use as the source for the base clock, CGMOUT. (See [7.6.2 PLL Bandwidth Control Register](#).) If PLL interrupts are enabled, the software can wait for a PLL interrupt request and then check the LOCK bit. If interrupts are disabled, software can poll the LOCK bit continuously (during PLL 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 [7.4.8 Base Clock Selector Circuit](#).) If the VCO is selected as the source for the base clock and the LOCK bit is clear, the PLL has suffered a severe noise hit and the software must take appropriate action, depending on the application. (See [7.7 Interrupts](#) for information and precautions on using interrupts.)

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

- The  $\overline{ACQ}$  bit (See [7.6.2 PLL Bandwidth Control Register](#).) is a read-only indicator of the mode of the filter. (See [7.4.4 Acquisition and Tracking Modes](#).)
- The  $\overline{ACQ}$  bit is set when the VCO frequency is within a certain tolerance and is cleared when the VCO frequency is out of a certain tolerance. (See [7.9 Acquisition/Lock Time Specifications](#) for more information.)

- The LOCK bit is a read-only indicator of the locked state of the PLL.
- The LOCK bit is set when the VCO frequency is within a certain tolerance and is cleared when the VCO frequency is out of a certain tolerance. (See [7.9 Acquisition/Lock Time Specifications](#) for more information.)
- CPU interrupts can occur if enabled (PLLIE = 1) when the PLL's lock condition changes, toggling the LOCK bit. (See [7.6.1 PLL Control Register](#).)

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

These conditions apply when in manual mode:

- $\overline{\text{ACQ}}$  is a writable control bit that controls the mode of the filter. Before turning on the PLL in manual mode, the  $\overline{\text{ACQ}}$  bit must be clear.
- Before entering tracking mode ( $\overline{\text{ACQ}} = 1$ ), software must wait a given time,  $t_{\text{ACQ}}$  (See [7.9 Acquisition/Lock Time Specifications](#)), after turning on the PLL by setting PLLON in the PLL control register (PCTL).
- Software must wait a given time,  $t_{\text{AL}}$ , 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 CGMC are disabled.

### 7.4.6 Programming the PLL

This procedure shows how to program the PLL.

**NOTE:** *The round function in these equations means that the real number should be rounded to the nearest integer number.*



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

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

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

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

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

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

In cases where desired bus frequency has some tolerance, choose  $f_{\text{RCLK}}$  to a value determined either by other module requirements (such as modules which are clocked by CGMXCLK), cost requirements, or ideally, as high as the specified range allows. See [Section 25. Preliminary Electrical Specifications](#). Choose the reference divider,  $R = 1$ . After choosing N and P, the actual bus frequency can be determined using equation in step 2.

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

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

4. Select a VCO frequency multiplier, N.

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

Reduce N/R to the lowest possible R.

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

| Current N Value  | P |
|--|---|
| $0 < N \leq N_{\text{max}}$                                | 0 |
| $N_{\text{max}} < N \leq N_{\text{max}} \times 2$          | 1 |
| $N_{\text{max}} \times 2 < N \leq N_{\text{max}} \times 4$ | 2 |
| $N_{\text{max}} \times 4 < N \leq N_{\text{max}} \times 8$ | 3 |

Then recalculate N:

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

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

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

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

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

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

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

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

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

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

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

For proper operation,

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

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

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

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

**Table 7-1** provides numeric examples. Numbers are in hexadecimal notation.

**Table 7-1. Numeric Example**

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

## 7.4.7 Special Programming Exceptions

The programming method described in **7.4.6 Programming the PLL** does not account for three possible exceptions. A value of 0 for R, N, or L is meaningless when used in the equations given. To account for these exceptions:

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

See **7.4.8 Base Clock Selector Circuit**.

## 7.4.8 Base Clock Selector Circuit

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

one-half of the base clock frequency, is one-fourth the frequency of the selected clock (CGMXCLK or CGMVCLK).

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

### 7.4.9 CGMC External Connections

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

The crystal oscillator is normally connected in a Pierce oscillator configuration, as shown in [Figure 7-2](#). This figure shows only the logical representation of the internal components and may not represent actual circuitry. The oscillator configuration uses five components:

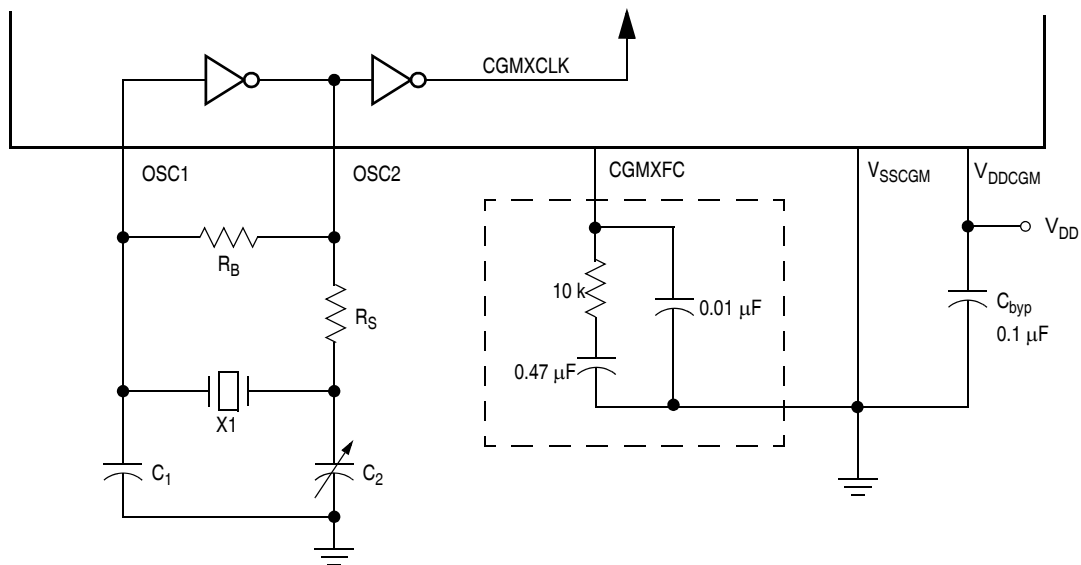
- Crystal,  $X_1$
- Fixed capacitor,  $C_1$
- Tuning capacitor,  $C_2$  (can also be a fixed capacitor)
- Feedback resistor,  $R_B$
- Series resistor,  $R_S$

The series resistor ( $R_S$ ) is included in the diagram to follow strict Pierce oscillator guidelines. Refer to the crystal manufacturer's data for more information regarding values for  $C_1$  and  $C_2$ .

[Figure 7-2](#) also shows the external components for the PLL:

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

## Clock Generator Module (CGMC)



Note: Filter network in box can be replaced with a 0.47- $\mu$ F capacitor, but it will degrade stability.

**Figure 7-2. CGMC External Connections**

Routing should be done with great care to minimize signal cross talk and noise.

See [25.10.1 CGM Component Specifications](#) for capacitor and resistor values.

## 7.5 Input/Output Signals

This subsection describes the CGMC I/O signals.

### 7.5.1 Crystal Amplifier Input Pin (OSC1)

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

### 7.5.2 Crystal Amplifier Output Pin (OSC2)

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

### 7.5.3 External Filter Capacitor Pin (CGMXFC)

The CGMXFC pin is required by the loop filter to filter out phase corrections. An external filter network is connected to this pin. (See [Figure 7-2](#).)

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

### 7.5.4 PLL Analog Power Pin ( $V_{DDCGM}$ )

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

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

### 7.5.5 PLL Analog Ground Pin ( $V_{SSCGM}$ )

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

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

### 7.5.6 Crystal Output Frequency Signal (CGMXCLK)

CGMXCLK is the crystal oscillator output signal. It runs at the full speed of the crystal ( $f_{XCLK}$ ) and comes directly from the crystal oscillator circuit. [Figure 7-2](#) 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.

### 7.5.7 CGMC Base Clock Output (CGMOUT)

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

### 7.5.8 CGMC CPU Interrupt (CGMINT)

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

## 7.6 CGMC Registers

These registers control and monitor operation of the CGMC:

- PLL control register (PCTL) — see [7.6.1 PLL Control Register](#)
- PLL bandwidth control register (PBWC) — see [7.6.2 PLL Bandwidth Control Register](#)
- PLL multiplier select register high (PMSH) — see [7.6.3 PLL Multiplier Select Register High](#)
- PLL multiplier select register low (PMSL) — see [7.6.4 PLL Multiplier Select Register Low](#)
- PLL VCO range select register (PMRS) — see [7.6.5 PLL VCO Range Select Register](#)
- PLL reference divider select register (PMDS) — see [7.6.6 PLL Reference Divider Select Register](#)

**Figure 7-3** is a summary of the CGMC registers.



| Addr.  | Register Name   | Bit 7  | 6     | 5    | 4                | 3    | 2     | 1     | Bit 0 |      |
|--------|---|--------|-------|------|------------------|------|-------|-------|-------|------|
| \$0008 | PLL Control Register (PCTL)<br>See page 114.                  | Read:  | PLLIE | PLLF | PLLON            | BCS  | PRE1  | PRE0  | VPR1  | VPR0 |
|        |   | Write: |       |      |                  |      |       |       |       |      |
|        |   | Reset: | 0     | 0    | 1                | 0    | 0     | 0     | 0     | 0    |
| \$0009 | PLL Bandwidth Control Register (PBWC)<br>See page 117.        | Read:  | AUTO  | LOCK | $\overline{ACQ}$ | 0    | 0     | 0     | 0     | R    |
|        |   | Write: |       |      |                  |      |       |       |       |      |
|        |   | Reset: | 0     | 0    | 0                | 0    | 0     | 0     | 0     | 0    |
| \$000A | PLL Multiplier Select High Register (PMSH)<br>See page 118.   | Read:  | 0     | 0    | 0                | 0    | MUL11 | MUL10 | MUL9  | MUL8 |
|        |   | Write: |       |      |                  |      |       |       |       |      |
|        |   | Reset: | 0     | 0    | 0                | 0    | 0     | 0     | 0     | 0    |
| \$000B | PLL Multiplier Select Low Register (PMSL)<br>See page 119.    | Read:  | MUL7  | MUL6 | MUL5             | MUL4 | MUL3  | MUL2  | MUL1  | MUL0 |
|        |   | Write: |       |      |                  |      |       |       |       |      |
|        |   | Reset: | 0     | 1    | 0                | 0    | 0     | 0     | 0     | 0    |
| \$000C | PLL VCO Range Select Register (PMRS)<br>See page 120.         | Read:  | VRS7  | VRS6 | VRS5             | VRS4 | VRS3  | VRS2  | VRS1  | VRS0 |
|        |   | Write: |       |      |                  |      |       |       |       |      |
|        |   | Reset: | 0     | 1    | 0                | 0    | 0     | 0     | 0     | 0    |
| \$000D | PLL Reference Divider Select Register (PMDS)<br>See page 121. | Read:  | 0     | 0    | 0                | 0    | RDS3  | RDS2  | RDS1  | RDS0 |
|        |   | Write: |       |      |                  |      |       |       |       |      |
|        |   | Reset: | 0     | 0    | 0                | 0    | 0     | 0     | 0     | 1    |

= Unimplemented     
 R = Reserved

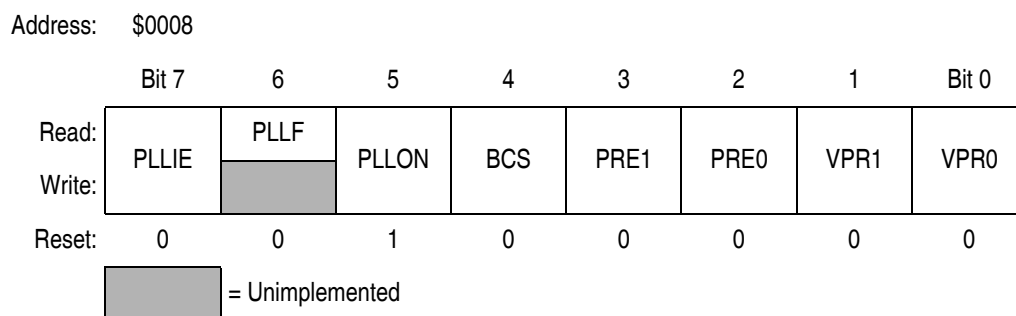
**NOTES:**

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

**Figure 7-3. CGMC I/O Register Summary**

## 7.6.1 PLL Control Register

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



**Figure 7-4. PLL Control Register (PCTL)**

### PLLIE — PLL Interrupt Enable Bit

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

1 = PLL interrupts enabled

0 = PLL interrupts disabled

### PLLF — PLL Interrupt Flag Bit

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

1 = Change in lock condition

0 = No change in lock condition

**NOTE:** Do not inadvertently clear the PLLF bit. Any read or read-modify-write operation on the PLL control register clears the PLLF bit.

#### PLLON — PLL On Bit

This read/write bit activates the PLL and enables the VCO clock, CGMVCLK. PLLON cannot be cleared if the VCO clock is driving the base clock, CGMOUT (BCS = 1). (See [7.4.8 Base Clock Selector Circuit](#).) Reset sets this bit so that the loop can stabilize as the MCU is powering up.

- 1 = PLL on
- 0 = PLL off

#### BCS — Base Clock Select Bit

This read/write bit selects either the crystal oscillator output, CGMXCLK, or the VCO clock, CGMVCLK, as the source of the CGMC 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 [7.4.8 Base Clock Selector Circuit](#).) Reset clears the BCS bit.

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

**NOTE:** *PLLON and BCS have built-in protection that prevents the base clock selector circuit from selecting the VCO clock as the source of the base clock if the PLL is off. Therefore, PLLON cannot be cleared when BCS is set, and BCS cannot be set when PLLON is clear. If the PLL is off (PLLON = 0), selecting CGMVCLK requires two writes to the PLL control register. (See [7.4.8 Base Clock Selector Circuit](#).)*

#### PRE1 and PRE0 — Prescaler Program Bits

These read/write bits control a prescaler that selects the prescaler power-of-two multiplier, P. (See [7.4.3 PLL Circuits](#) and [7.4.6 Programming the PLL](#).) PRE1 and PRE0 cannot be written when the PLLON bit is set. Reset clears these bits.

**NOTE:** *The value of P is normally 0 when using a 32.768-kHz crystal as the reference.*

**Table 7-2. PRE 1 and PRE0 Programming**

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

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

These read/write bits control the VCO's hardware power-of-two range multiplier E that, in conjunction with L, (See [7.4.3 PLL Circuits](#), [7.4.6 Programming the PLL](#), and [7.6.5 PLL VCO Range Select Register](#).) controls the hardware center-of-range frequency,  $f_{VRS}$ . VPR1 and VPR0 cannot be written when the PLLON bit is set. Reset clears these bits.

**Table 7-3. VPR1 and VPR0 Programming**

| VPR1 and VPR0 | E                | VCO Power-of-Two Range Multiplier |
|---------------|------------------|-----------------------------------|
| 00            | 0                | 1                                 |
| 01            | 1                | 2                                 |
| 10            | 2                | 4                                 |
| 11            | 3 <sup>(1)</sup> | 8                                 |

Note:

1. Do not program E to a value of 3.

## 7.6.2 PLL Bandwidth Control Register

The PLL bandwidth control register (PBWC):

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

Address: \$0009

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

= Unimplemented     
 R = Reserved

**Figure 7-5. PLL Bandwidth Control Register (PBWC)**

### AUTO — Automatic Bandwidth Control Bit

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

- 1 = Automatic bandwidth control
- 0 = Manual bandwidth control

### LOCK — Lock Indicator Bit

When the AUTO bit is set, LOCK is a read-only bit that becomes set when the VCO clock, CGMVCLK, is locked (running at the programmed frequency). When the AUTO bit is clear, LOCK reads as logic 0 and has no meaning. The write one function of this bit is reserved for test, so this bit must **always** be written a 0. Reset clears the LOCK bit.

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

## $\overline{ACQ}$ — Acquisition Mode Bit

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

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

- 1 = Tracking mode
- 0 = Acquisition mode

### 7.6.3 PLL Multiplier Select Register High

The PLL multiplier select register high (PMSH) contains the programming information for the high byte of the modulo feedback divider.

Address: \$000A

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

= Unimplemented

**Figure 7-6. PLL Multiplier Select Register High (PMSH)**

#### MUL11–MUL8 — Multiplier Select Bits

These read/write bits control the high byte of the modulo feedback divider that selects the VCO frequency multiplier N. (See [7.4.3 PLL Circuits](#) and [7.4.6 Programming the PLL](#).) A value of \$0000 in the multiplier select registers configures the modulo feedback divider the same as a value of \$0001. Reset initializes the registers to \$0040 for a default multiply value of 64.

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

PMSH7—PMSH4 — Unimplemented Bits

These bits have no function and always read as logic 0s.

### 7.6.4 PLL Multiplier Select Register Low

The PLL multiplier select register low (PMSL) contains the programming information for the low byte of the modulo feedback divider.

Address: \$000B

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

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

MUL7—MUL0 — Multiplier Select Bits

These read/write bits control the low byte of the modulo feedback divider that selects the VCO frequency multiplier, N. (See [7.4.3 PLL Circuits](#) and [7.4.6 Programming the PLL](#).) MUL7—MUL0 cannot be written when the PLLON bit in the PCTL is set. A value of \$0000 in the multiplier select registers configures the modulo feedback divider the same as a value of \$0001. Reset initializes the register to \$40 for a default multiply value of 64.

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

## 7.6.5 PLL VCO Range Select Register

**NOTE:** *PMRS may be called PVRS on other HC08 derivatives.*

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

Address: \$000C

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

**Figure 7-8. PLL VCO Range Select Register (PMRS)**

### VRS7–VRS0 — VCO Range Select Bits

These read/write bits control the hardware center-of-range linear multiplier L which, in conjunction with E (See [7.4.3 PLL Circuits](#), [7.4.6 Programming the PLL](#), and [7.6.1 PLL Control Register](#).), controls the hardware center-of-range frequency,  $f_{VRS}$ . VRS7–VRS0 cannot be written when the PLLON bit in the PCTL is set. (See [7.4.7 Special Programming Exceptions](#).) A value of \$00 in the VCO range select register disables the PLL and clears the BCS bit in the PLL control register (PCTL). (See [7.4.8 Base Clock Selector Circuit](#) and [7.4.7 Special Programming Exceptions](#).) Reset initializes the register to \$40 for a default range multiply value of 64.

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

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



### 7.6.6 PLL Reference Divider Select Register

**NOTE:** *PMDS may be called PRDS on other HC08 derivatives.*

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

Address: \$000D

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

= Unimplemented

**Figure 7-9. PLL Reference Divider Select Register (PMDS)**

#### RDS3–RDS0 — Reference Divider Select Bits

These read/write bits control the modulo reference divider that selects the reference division factor, R. (See [7.4.3 PLL Circuits](#) and [7.4.6 Programming the PLL](#).) RDS7–RDS0 cannot be written when the PLLON bit in the PCTL is set. A value of \$00 in the reference divider select register configures the reference divider the same as a value of \$01. (See [7.4.7 Special Programming Exceptions](#).) Reset initializes the register to \$01 for a default divide value of 1.

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

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

#### PMDS7–PMDS4 — Unimplemented Bits

These bits have no function and always read as logic 0s.

### 7.7 Interrupts

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

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

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

### 7.8 Special Modes

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

#### 7.8.1 Wait Mode

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

mode, such as when the PLL is first enabled and waiting for LOCK or LOCK is lost.

## 7.8.2 Stop Mode

The STOP instruction disables the phase-locked loop but the oscillator will continue to operate.

If the STOP instruction is executed with the VCO clock, CGMVCLK, divided by two driving CGMOUT, the PLL automatically clears the BCS bit in the PLL control register (PCTL), thereby selecting the crystal clock, CGMXCLK, divided by two as the source of CGMOUT. When the MCU recovers from STOP, the crystal clock divided by two drives CGMOUT and BCS remains clear.

## 7.8.3 CGMC During Break Interrupts

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. (See [20.8.3 SIM Break Flag Control Register](#).)

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

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

## 7.9 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.

### 7.9.1 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 percent acquisition time tolerance. If a command instructs the system to change from 0 Hz to 1 MHz, the acquisition time is the time taken for the frequency to reach  $1\text{ MHz} \pm 50\text{ kHz}$ . 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  $1\text{ MHz} \pm 5\text{ kHz}$ . 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.

### 7.9.2 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,  $f_{RDV}$ . This frequency is the input to the phase detector and controls how often the PLL makes corrections. For stability, the corrections must be small compared to the desired frequency, so several corrections are required to reduce the frequency error. Therefore, the slower the reference the longer it takes to make these

corrections. This parameter is under user control via the choice of crystal frequency  $f_{XCLK}$  and the R value programmed in the reference divider. (See [7.4.3 PLL Circuits](#), [7.4.6 Programming the PLL](#), and [7.6.6 PLL Reference Divider Select Register](#).)

Another critical parameter is the external filter network. The PLL modifies the voltage on the VCO by adding or subtracting charge from capacitors in this network. Therefore, the rate at which the voltage changes for a given frequency error (thus change in charge) is proportional to the capacitance. The size of the capacitor also is related to the stability of the PLL. If the capacitor is too small, the PLL cannot make small enough adjustments to the voltage and the system cannot lock. If the capacitor is too large, the PLL may not be able to adjust the voltage in a reasonable time. (See [7.9.3 Choosing a Filter](#).)

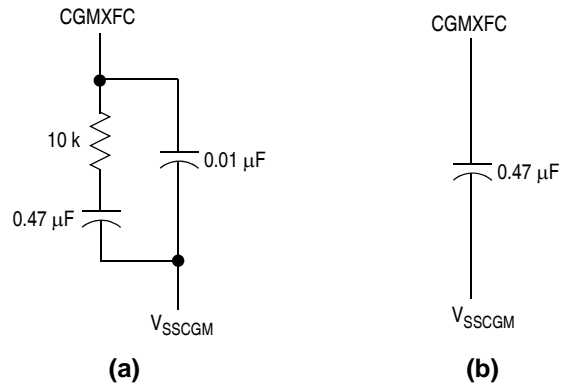
Also important is the operating voltage potential applied to  $V_{DDCGM}$ . The power supply potential alters the characteristics of the PLL. A fixed value is best. Variable supplies, such as batteries, are acceptable if they vary within a known range at very slow speeds. Noise on the power supply is 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.

### 7.9.3 Choosing a Filter

As described in [7.9.2 Parametric Influences on Reaction Time](#), the external filter network is critical to the stability and reaction time of the PLL. The PLL is also dependent on reference frequency and supply voltage.

Either of the filter networks in **Figure 7-10** is recommended when using a 32.768-kHz reference crystal. **Figure 7-10 (a)** is used for applications requiring better stability. **Figure 7-10 (b)** is used in low-cost applications where stability is not critical.



**Figure 7-10. PLL Filter**

## Section 8. Closed-Caption Data Slicer (DSL)

### 8.1 Contents

|       |  |     |
|-------|--|-----|
| 8.2   | Introduction                               | 128 |
| 8.3   | Features                                   | 128 |
| 8.4   | Functional Description                     | 128 |
| 8.4.1 | Slicer                                     | 128 |
| 8.4.2 | Line and Field Detection                   | 129 |
| 8.4.3 | Data Sampling                              | 130 |
| 8.5   | Programming Guidelines                     | 130 |
| 8.5.1 | Setup                                      | 131 |
| 8.5.2 | Interrupt Servicing                        | 131 |
| 8.5.3 | Debugging                                  | 131 |
| 8.6   | Input/Output                               | 132 |
| 8.6.1 | Video                                      | 132 |
| 8.6.2 | VSYNC, HSYNC, and Video Input Requirements | 132 |
| 8.7   | Registers                                  | 134 |
| 8.7.1 | DSL Character Registers                    | 135 |
| 8.7.2 | DSL Control Register 1                     | 136 |
| 8.7.3 | DSL Control Register 2                     | 138 |
| 8.7.4 | DSL Status Register                        | 141 |
| 8.8   | Low-Power Modes                            | 142 |
| 8.9   | Interrupts and Resets                      | 142 |

## 8.2 Introduction

The closed-caption data slicer (DSL) extracts FCC (Federal Communications Commission) closed-caption compatible data from an NTSC (National Television System Committee) or PAL (Phase Alternating Line) composite video signal for closed-caption and extended data services applications.

## 8.3 Features

The DSL provides these features:

- FCC/EIA-744 V-chip compatible
- FCC/EIA-608 line 21 format data extraction on both field 1 and field 2
- Software programmable line selection
- Hardware parity checking
- Software programmable data slicing level

## 8.4 Functional Description

The closed-caption data slicer (DSL) extracts FCC closed-caption compatible data from an NTSC composite video signal. The DSL accomplishes this task by slicing sync and data information from the incoming video; the sync information is used to locate fields and lines to trigger data sampling, and the sampled sliced data is stored in registers for CPU access. A block diagram of the DSL is shown in [Figure 8-1](#).

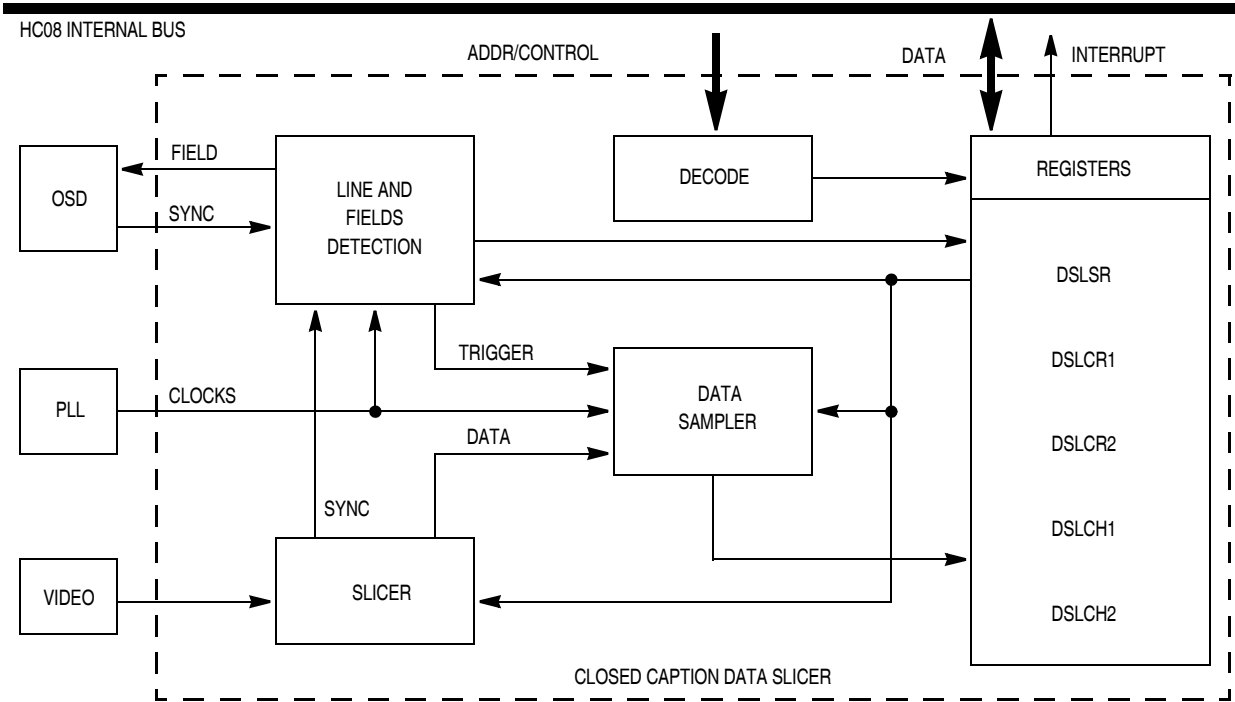
### 8.4.1 Slicer

The slicer circuitry compares the composite video input with internal reference levels to determine when sync pulses or data bits are being received. DSL control bits allow the user to select one of four reference levels for slicing data. Refer to [Figure 8-2](#).



**8.4.2 Line and Field Detection**

The line and field detection circuitry uses the separated sync output of the slicer section, along with signals derived from the HSYNC and VSYNC inputs, to determine line and field timing for the DSL and field timing for the on-screen display module (OSD). DSL control registers enable the user to select one of several lines from which to extract data in closed-caption format, and also provide flexibility for adapting to differences in chassis sync and video signal timing.



**Figure 8-1. Data Slicer Block Diagram**

## Closed-Caption Data Slicer (DSL)

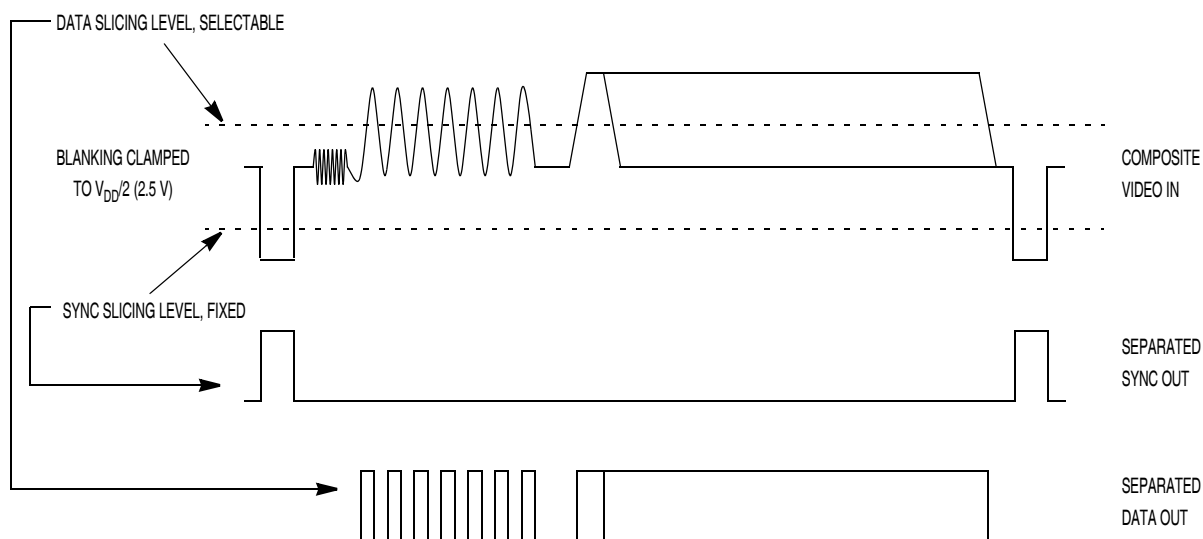


Figure 8-2. Slicer Input and Output

### 8.4.3 Data Sampling

The data sampling circuitry uses the frequency reference from the PLL (phase locked to the HSYNC input) to sample data bits from the separated data output of the slicer section. The sampling circuitry performs parity checking and stores the data in parallel format in registers which the CPU can access. The DSL sets a status register bit (DSFL) and optionally will generate an interrupt to indicate when data is available.

### 8.5 Programming Guidelines

The DSL can be used for extracting data in the FCC closed-caption format from both fields in the composite video signal. The DSL may be configured to search for data on any line from 14 to 29 (VSYNC timing may limit this range in some applications), although it cannot search on multiple lines. The DSL does not perform any closed-caption decoding or display; these functions must be provided in software with support of the OSD module.

### 8.5.1 Setup

The DSL control registers 1 and 2 must be initialized for proper operation. The DSEN bit in DSLCR1 should be set to enable the DSL, and the DSIEN bit should be set if the DSFL will be serviced through interrupts rather than polling. The desired line for decoding data should be specified in DSLCR1, and the data slicing voltage reference, vertical pulse width and delay should be initialized in DSLCR2. Some of these control bits may require adjustment during operation if video input conditions change.

### 8.5.2 Interrupt Servicing

In both the interrupt and polling methods, the DSFL flag in the DSLSR indicates that two bytes are available for reading. The FIELD1 bit in the DSLSR indicates which field the 2 bytes have been extracted from. The OVFL bit should also be checked to ensure that overflow has not occurred. Within the service routine, DSFL should be cleared by writing to the DSLSR.

Since the two character registers are shared for both field 1 and field 2, the user has approximately one video vertical scan (16.68 ms) to unload two bytes of data per field. The two bytes each contain a 7-bit code and a parity error flag. The parity error flags indicate that the data received may be invalid.

### 8.5.3 Debugging

The DSLSR status bits can be helpful when using the DSL in a new application. If the DSL does not correctly slice data, check these status bits in this order:

1. CSYNC — This bit provides visibility of the separated sync output of the slicer section. If CSYNC is not set during the time that a sync pulse is present in the incoming video, then the input level of the video signal needs adjustment to allow proper sync slicing.
2. VPDET — This bit indicates that a wide vertical sync pulse has been detected in the composite video. If VPDET is not set after the

vertical blanking interval, this indicates that either the wide vertical sync pulses in the composite video are shorter than the pulse width defined by PW1:PW0, or the VSYNC input was not active during the vertical blanking interval, or the selected decoding line was encountered before the vertical sync was seen in composite video or VSYNC.

3. RIC1 and RIC0 — These two bits are the output of a counter that counts up to three rising edges in the sliced data during the run-in-clock window. If the count is less than three, this may indicate that the incorrect line is being decoded, or that there is too much delay between composite video and HSYNC.

## 8.6 Input/Output

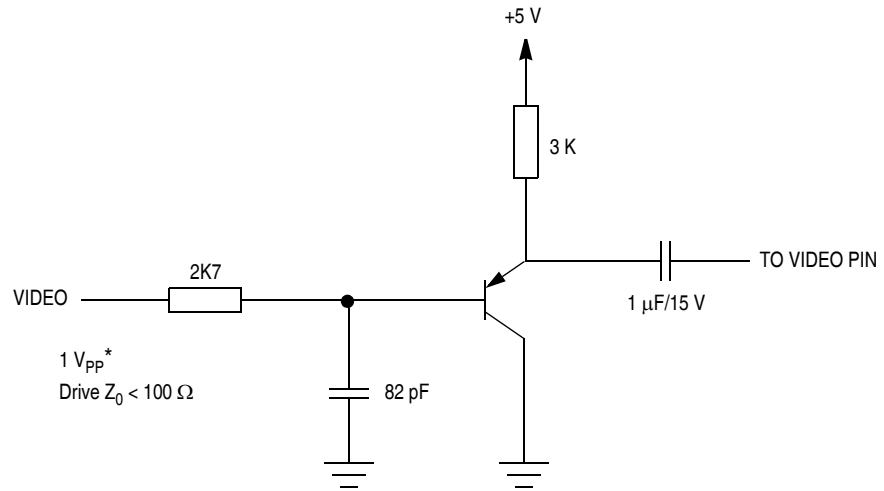
The DSL has one dedicated pin, the VIDEO input. It also uses the VSYNC and HSYNC inputs indirectly.

### 8.6.1 Video

This dedicated input provides the NTSC or PAL composite video signal from the external video system. The closed-caption data and sync information is extracted from this signal. A typical input circuit for the closed-caption video input is shown in [Figure 8-3](#). The video input uses a sync versus blanking level duty cycle clamp to set the blanking level of the incoming video to  $V_{DD}$  divided by two.

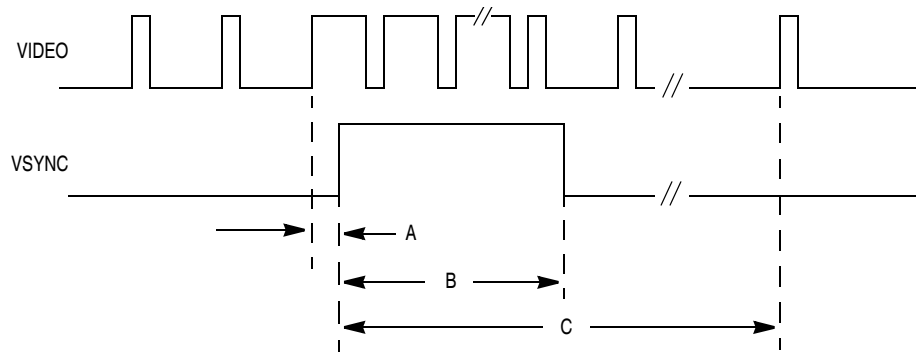
### 8.6.2 VSYNC, HSYNC, and Video Input Requirements

The DSL uses both VSYNC and HSYNC indirectly, putting some restrictions on the relationship between these two signals and the composite video. Refer to [Figure 8-4](#).



\* From negative synchronous tip to 100 IRE

**Figure 8-3. Video Input Circuit**



A: Rising edge CSYNC first wide vertical pulse to rising edge VSYNC — minimum 1  $\mu\text{s}$   
 B: VSYNC pulse width — minimum 2  $\mu\text{s}$   
 C: Rising edge VSYNC to rising edge CSYNC of line to be detected — minimum 64  $\mu\text{s}$

**Figure 8-4. DSL Input Timing**

The external VSYNC input is provided by the OSD to the DSL as an active-high signal. VSYNC is used by the DSL to:

- Detect a valid vertical blanking interval
- Generate field information for the DSL and OSD

A synthesized version (50 percent duty cycle) of external HSYNC is provided by the PLL to the DSL. The DSL also uses higher frequency PLL derivatives of HSYNC for most of its synchronization. The horizontal frequency is used to generate field information for the OSD.

The VIDEO input is separated into sync and data components. Composite sync (both vertical and horizontal sync information) is used by the DSL to:

- Detect a valid vertical blanking interval
- Disable the video input clamp during vertical blanking
- Generate field information for the DSL
- Latch an overflow condition
- Indicate when the desired decoding line has been found

Composite data is used for detecting the run-in-clock, the start bit, and sampling the data.

### 8.7 Registers

The DSL has five registers, which are described in this section.

- DSL character register 1, DSLCH1
- DSL character register 2, DSLCH2
- DSL control register 1, DSLCR1
- DSL control register 2, DSLCR2
- DSL status register, DSLSR

### 8.7.1 DSL Character Registers

These registers, DSLCH1 and DSLCH2, each contain seven data bits and a parity error flag.

Address: \$0047–\$0048

|        | Bit 7 | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|-------|-----|-----|-----|-----|-----|-----|-------|
| Read:  | PE    | DA6 | DA5 | DA4 | DA3 | DA2 | DA1 | DA0   |
| Write: |       |     |     |     |     |     |     |       |
| Reset: | 0     | X   | X   | X   | X   | X   | X   | X     |

= Unimplemented      X = Indeterminate

**Figure 8-5. DSL Character Registers (DSLCH1 and DSLCH2)**

#### PE — Parity Error Bit

This read-only bit is set when odd parity is not detected on the byte received. Writing to the DSL status registers clears this bit. A reset also clears this bit.

#### DA6–DA0 — Closed-Caption Data Bit

These read-only bits are sampled (LSB first) from the data slicer, which extracts data from the VIDEO input pin. These seven bits will define a closed-caption character. A reset has no effect on these bits.

## 8.7.2 DSL Control Register 1

Address: \$0049

|        | Bit 7 | 6    | 5    | 4     | 3     | 2     | 1     | Bit 0  |
|--------|-------|------|------|-------|-------|-------|-------|--------|
| Read:  | DSIEN | DSEN | VVEN | LINE4 | LINE3 | LINE2 | LINE1 | DISCLP |
| Write: |       |      |      |       |       |       |       |        |
| Reset: | 0     | 0    | 0    | 0     | 0     | 0     | 0     | 0      |

**Figure 8-6. DSL Control Register 1 (DSLCR1)**

### DSIEN — Data Slicer Interrupt Enable Bit

DSIEN determines if the DSFL bit in the DSLSR is enabled to generate interrupt requests to the CPU. A reset clears this bit.

- 1 = DSFL interrupt enabled
- 0 = DSFL interrupt disabled

### DSEN — Data Slicer Enable Bit

DSEN determines if the DSL is enabled. When the DSL is enabled, the data slicer voltage reference is turned on and the PLL clock outputs are input to the DSL circuitry. The PLL must be enabled (PLEN bit on the OSD enable control register) to operate the DSL. The DSL provides field information to the OSD and should not be disabled if the OSD is in use. A reset clears this bit.

- 1 = DSL enabled
- 0 = DSL disabled

### VVEN — VSYNC Verification Enable Bit

VVEN determines if VSYNC from the OSD module will be checked during the vertical blanking interval. VVEN set allows data extraction abortion if VSYNC is not detected.



### LINE4–LINE1 — Closed-Caption LINE Bits

These four bits allow the user to specify the line to be used for closed-caption decoding, according to [Table 8-1](#). A reset clears these bits.

**Table 8-1. Line Selection**

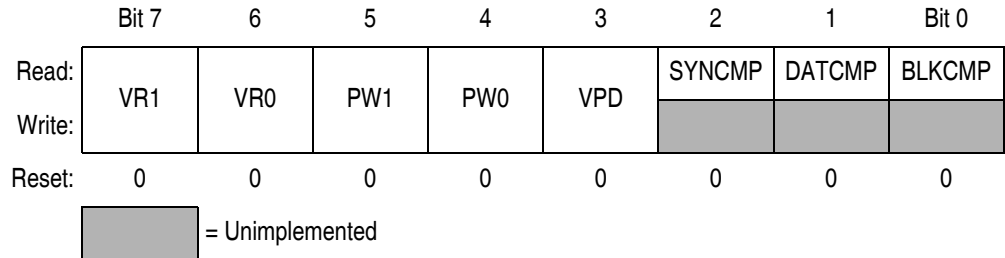
| LINE4–LINE1 | Target Line |
|-------------|-------------|
| 0000        | 14          |
| 0001        | 15          |
| 0010        | 16          |
| ...         | ...         |
| 1101        | 27          |
| 1110        | 28          |
| 1111        | 29          |

### DISCLP — Disable Clamping in Test Mode

This bit can be accessed only in DSL test mode, which is activated by setting bit DSLTST of the OSD status register (\$0046). It allows direct control over an internal signal that controls the analog portion of the data slicer. It is provided to facilitate the production test of the slicer circuit.

## 8.7.3 DSL Control Register 2

Address: \$004A



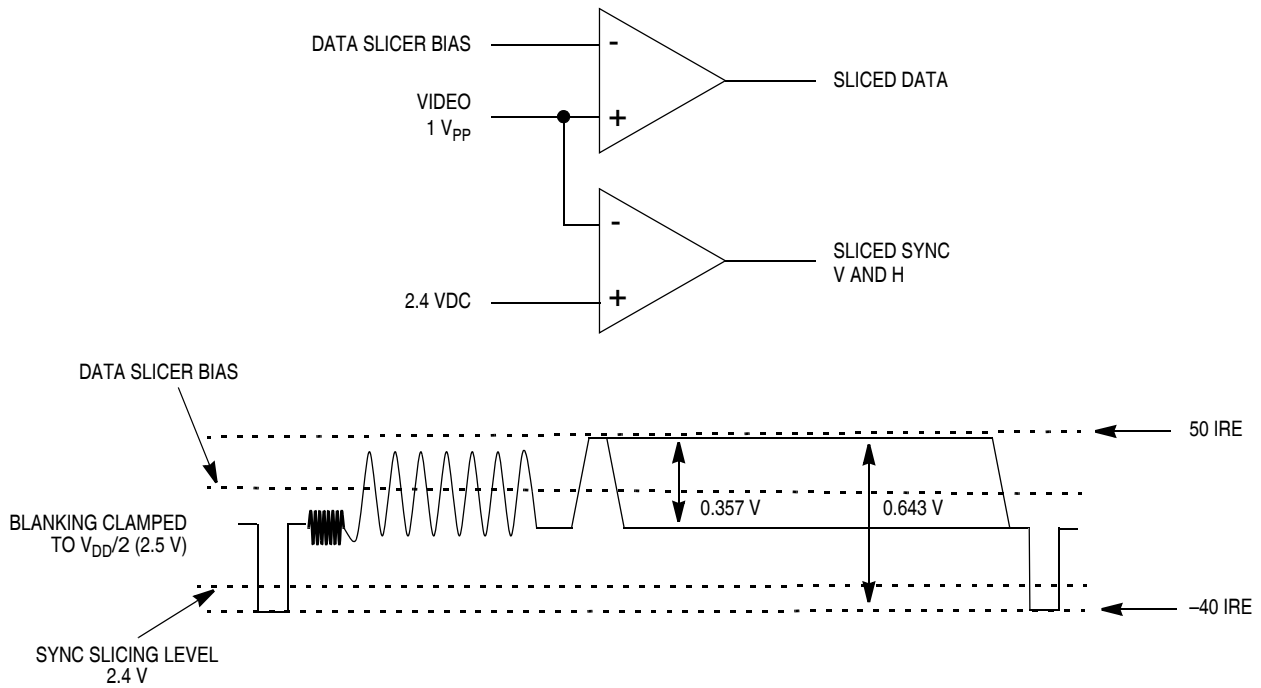
**Figure 8-7. DSL Control Register 2 (DSLCR2)**

### VR1 and VR0 — Voltage Reference Bits

These bits select the reference bias voltage for slicing data according to [Table 8-2](#). As seen in [Figure 8-8](#), the input blanking level is clamped to 2.5 V, and the sync slicing level is fixed at 2.4 V. A reset clears these bits.

**Table 8-2. Data Slicer Bias Voltage**

| VR1 | VR0 | Nominal Bias |
|-----|-----|--------------|
| 0   | 0   | 2.80 V       |
| 0   | 1   | 2.74 V       |
| 1   | 0   | 2.66 V       |
| 1   | 1   | 2.60 V       |



**Figure 8-8. Data and Sync Slicing**

**PW1 and PW0 — Vertical Sync Pulse Width Bit**

These bits define, according to [Table 8-3](#), the minimum width of an extracted sync signal pulse required to identify it as a vertical sync pulse. The standard video vertical sync interval pulse width is 29.2  $\mu\text{s}$ ; the DSL requires at least 8  $\mu\text{s}$  to recognize a vertical sync pulse. A reset clears these bits.

**Table 8-3. Minimum Vertical Pulse Width**

| PW1 | PW0 | Pulse Width      |
|-----|-----|------------------|
| 0   | 0   | 8 $\mu\text{s}$  |
| 0   | 1   | 10 $\mu\text{s}$ |
| 1   | 0   | 12 $\mu\text{s}$ |
| 1   | 1   | 14 $\mu\text{s}$ |

## VPD — Vertical Pulse Delay Bit

VPD determines whether the VSYNC input is delayed before field detection. VPD should be set when the VSYNC active edge is within  $\pm 4 \mu\text{s}$  of the HSYNC active edge for either field. See [Figure 8-9](#) for clarification. A reset clears this bit.

- 1 = VSYNC input is delayed by 12 to 22  $\mu\text{s}$  before field detection
- 0 = VSYNC input is not delayed

## SYNCMP — Synchronism Slicing Comparator Output Bit

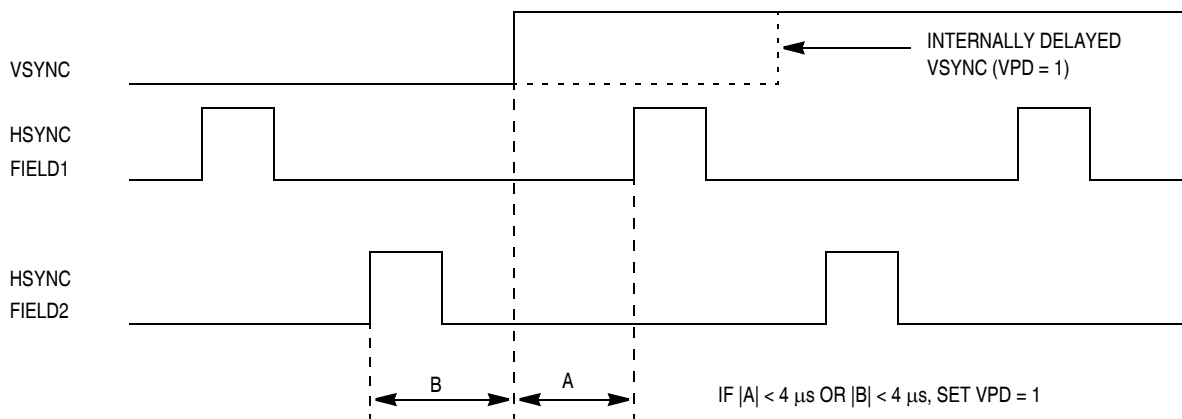
Out of test mode, this read-only bit is always 0. In DSL test mode, which is activated by setting bit DSLTST of the OSD status register (\$0046), it mirrors the state of the synchronism slicing comparator. It is provided to facilitate the production test of the data slicer circuit.

## DATCMP — Data Slicing Comparator Output Bit

This bit is also meaningful only in DSL test mode. It mirrors the state of the data slicing comparator.

## BLKCMP — Blanking Level Comparator Output Bit

This bit is also meaningful only in DSL test mode. It mirrors the state of the blanking level comparator.




**Figure 8-9. Conditions for Setting Vertical Pulse Delay**

### 8.7.4 DSL Status Register

The DSLSR contains the DSL interrupt flag and status bits and provides visibility to the slicer sync output.

Address: \$004B

|        | Bit 7 | 6    | 5      | 4 | 3     | 2     | 1    | Bit 0 |
|--------|-------|------|--------|---|-------|-------|------|-------|
| Read:  | DSFL  | OVFL | FIELD1 | 0 | CSYNC | VPDET | RIC1 | RIC0  |
| Write: |       |      |        |   |       |       |      |       |
| Reset: | 0     | 0    | 0      | 0 | X     | 0     | 0    | 0     |

 = Unimplemented

**Figure 8-10. DSL Status Register (DSL SR)**

#### DSFL — Data Sampled Flag Bit

This bit is set to indicate that the DSL has sampled 16 bits of closed-caption data (including parity) and placed them in DSLCH1 and DSLCH2. It will cause an interrupt if the DSIEN bit in DSLCR1 is set. It is cleared by writing to the DSLSR. A reset also clears this bit.

#### OVFL — Data Overflow Bit

This bit is set when new data starts to be extracted and DSFL is still set from the previous field. This condition does NOT cause an interrupt. It is cleared by writing to the DSLSR. A reset also clears this bit.

#### FIELD1 — Field 1 Indicator Bit

This bit indicates which field the closed-caption data in DSLCH1 and DSLCH2 was extracted from. A logic 1 indicates that the data in the status and character registers correspond to information retrieved from the specified line in field 1. A logic 0 indicates that the register contents correspond to information retrieved from the specified line in field 2. A reset clears this bit.

### CSYNC — Composite Separated SYNC Bit

This bit reflects the current state of the separated sync signal. If CSYNC = 1, it indicates the presence of a vertical or horizontal sync pulse in the composite video input. If CSYNC = 0, no sync pulse is currently present in the composite video input. A reset has no effect on this bit.

### VPDET – Vertical Sync Pulse Detect Bit

This bit is set when a VSYNC rising edge is found in the interval that starts in the valid vertical sync pulse detection and finishes one line before the programmed line to be extracted.

### RIC1 and RIC0 – Run-in Clock Cycle Count Bot

These bits reflect the current state of a 2-bit run-in clock cycle counter. This counter triggers off of the positive transitions of the video signal in an 8  $\mu$ s wide window starting 24  $\mu$ s after the horizontal sync pulse of the specified line. If both bits are set, this indicates that at least three positive transitions were detected. These bits are cleared by writing to the DSLSR. A reset also clears these bits.

## 8.8 Low-Power Modes

The DSL remains active during wait mode or stop mode; however, it will be unable to interrupt the CPU and bring it out of these modes. It is recommended that the DSL be disabled during wait mode or stop mode.

## 8.9 Interrupts and Resets

The DSL's only source of interrupt is the DSFL flag in the DSLSR. This interrupt is enabled by the DSIEN bit in the DSLCR1. This interrupt will cause the MCU to vector to the address stored in \$FFEC-\$FFED.

## Section 9. Configuration Register (CONFIG)

### 9.1 Contents

|     |                                  |     |
|-----|----------------------------------|-----|
| 9.2 | Introduction . . . . .           | 143 |
| 9.3 | Functional Description . . . . . | 143 |

### 9.2 Introduction

This section describes the configuration register (CONFIG). The configuration register enables or disables these options:

- Stop mode recovery time (32 CGMXCLK cycles or 4096 CGMXCLK cycles)
- COP timeout period ( $2^{18} - 2^4$  or  $2^{13} - 2^4$  CGMXCLK cycles)
- STOP instruction
- Computer operating properly module (COP)
- Low-voltage inhibit (LVI) module control and voltage trip point selection

### 9.3 Functional Description

The configuration register is used in the initialization of various options. It can be written only once after each reset. All of the configuration register bits are cleared during reset. Since the various options affect the operation of the MCU, it is recommended that this register be written immediately after reset. The configuration register is located at \$000E. It may be read at anytime.

**NOTE:** *On a FLASH device, the options except LVI5OR3 are one-time writable by the user after each reset. The LVI5OR3 bit is one-time writable by the*

## Configuration Register (CONFIG)

user only after each POR (power-on reset). The CONFIG register is not in the FLASH memory but is a special register containing one-time writable latches after each reset. Upon a reset, the CONFIG register defaults to predetermined settings as shown in [Figure 9-1](#).

Address: \$000E

|        | Bit 7   | 6                    | 5       | 4       | 3     | 2     | 1    | Bit 0 |
|--------|---------|----------------------|---------|---------|-------|-------|------|-------|
| Read:  | LVISTOP | LVI5OR3 <sup>†</sup> | LVIRSTD | LVIPWRD | SSREC | COPRS | STOP | COPD  |
| Write: |         |                      |         |         |       |       |      |       |
| Reset: | 0       | 0                    | 0       | 0       | 0     | 0     | 0    | 0     |

Note: LVI5OR3 bit is only reset via POR (power-on reset)

**Figure 9-1. Configuration Register (CONFIG)**

### LVISTOP — LVI Enable in Stop Mode Bit

When the LVIPWRD bit is clear, setting the LVISTOP bit enables the LVI to operate during stop mode. Reset clears LVISTOP. See [Section 15. Low-Voltage Inhibit \(LVI\)](#).

1 = LVI enabled during stop mode

0 = LVI disabled during stop mode

### LVI5OR3 — LVI 5-V or 3-V Operating Mode Bit

LVI5OR3 selects the voltage operating mode of the LVI module. See [Section 15. Low-Voltage Inhibit \(LVI\)](#). The voltage mode selected for the LVI should match the operating  $V_{DD}$ . See [Section 25. Preliminary Electrical Specifications](#) for the LVI's voltage trip points for each of the modes.

1 = LVI operates in 5-V mode.

0 = LVI operates in 3-V mode.

### LVIRSTD — LVI Reset Disable Bit

LVIRSTD disables the reset signal from the LVI module. See [Section 15. Low-Voltage Inhibit \(LVI\)](#).

1 = LVI module resets disabled

0 = LVI module resets enabled



LVIPWRD — LVI Power Disable Bit

LVIPWRD disables the LVI module. See [Section 15. Low-Voltage Inhibit \(LVI\)](#).

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

SSREC — Short Stop Recovery Bit

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

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

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

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

COPRS — COP Rate Select Bit

COPRS selects the COP timeout period. Reset clears COPRS. See [Section 10. Computer Operating Properly \(COP\)](#).

- 1 = COP timeout period =  $2^{13} - 2^4$  CGMXCLK cycles
- 0 = COP timeout period =  $2^{18} - 2^4$  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 [Section 10. Computer Operating Properly \(COP\)](#).

- 1 = COP module disabled
- 0 = COP module enabled

## Configuration Register (CONFIG)

## Section 10. Computer Operating Properly (COP)

### 10.1 Contents

|        |  |     |
|--------|--|-----|
| 10.2   | Introduction . . . . .                 | 147 |
| 10.3   | Functional Description . . . . .       | 148 |
| 10.4   | I/O Signals . . . . .                  | 149 |
| 10.4.1 | CGMXCLK . . . . .                      | 149 |
| 10.4.2 | STOP Instruction . . . . .             | 149 |
| 10.4.3 | COPCTL Write . . . . .                 | 149 |
| 10.4.4 | Power-On Reset. . . . .                | 149 |
| 10.4.5 | Internal Reset. . . . .                | 150 |
| 10.4.6 | Reset Vector Fetch. . . . .            | 150 |
| 10.4.7 | COPD (COP Disable). . . . .            | 150 |
| 10.4.8 | COPRS (COP Rate Select) . . . . .      | 150 |
| 10.5   | COP Control Register. . . . .          | 150 |
| 10.6   | Interrupts. . . . .                    | 151 |
| 10.7   | Monitor Mode . . . . .                 | 151 |
| 10.8   | Low-Power Modes . . . . .              | 151 |
| 10.8.1 | Wait Mode . . . . .                    | 151 |
| 10.8.2 | Stop Mode . . . . .                    | 151 |
| 10.9   | COP Module During Break Mode . . . . . | 152 |

### 10.2 Introduction

The computer operating properly (COP) module contains a free-running counter that generates a reset if allowed to overflow. The COP module helps software recover from runaway code. Prevent a COP reset by clearing the COP counter periodically. The COP module can be disabled through the COPD bit in the CONFIG register.

## 10.3 Functional Description

Figure 10-1 shows the structure of the COP module.

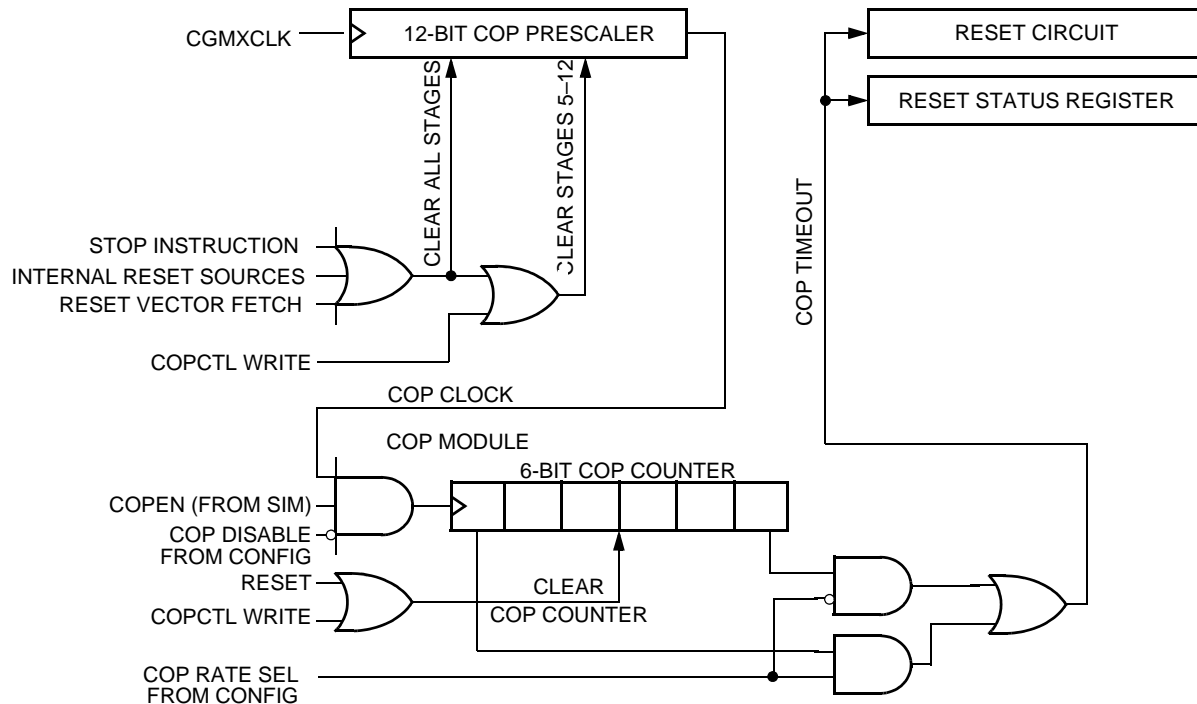


Figure 10-1. COP Block Diagram

The COP counter is a free-running 6-bit counter preceded by a 12-bit prescaler counter. If not cleared by software, the COP counter overflows and generates an asynchronous reset after  $2^{18} - 2^4$  or  $2^{13} - 2^4$  CGMXCLK cycles, depending on the state of the COP rate select bit, COPRS, in the configuration register. With a  $2^{13} - 2^4$  CGMXCLK cycle overflow option, a 32.768-kHz crystal gives a COP timeout period of 250 ms. Writing any value to location \$FFFF before an overflow occurs prevents a COP reset by clearing the COP counter and stages 12 through 5 of the prescaler.

**NOTE:** Service the COP immediately after reset and before entering or after exiting stop mode to guarantee the maximum time before the first COP counter overflow.

A COP reset pulls the  $\overline{\text{RST}}$  pin low for 32 CGMXCLK cycles and sets the COP bit in the reset status register (RSR).

In monitor mode, the COP is disabled if the  $\overline{\text{RST}}$  pin or the  $\overline{\text{IRQ}}$  is held at  $V_{\text{TST}}$ . During the break state,  $V_{\text{TST}}$  on the  $\overline{\text{RST}}$  pin disables the COP.

**NOTE:** *Place COP clearing instructions in the main program and not in an interrupt subroutine. Such an interrupt subroutine could keep the COP from generating a reset even while the main program is not working properly.*

## 10.4 I/O Signals

The following paragraphs describe the signals shown in [Figure 10-1](#).

### 10.4.1 CGMXCLK

CGMXCLK is the crystal oscillator output signal. CGMXCLK frequency is equal to the crystal frequency.

### 10.4.2 STOP Instruction

The STOP instruction clears the COP prescaler.

### 10.4.3 COPCTL Write

Writing any value to the COP control register (COPCTL) (see [10.5 COP Control Register](#)) clears the COP counter and clears bits 12 through 5 of the prescaler. Reading the COP control register returns the low byte of the reset vector.

### 10.4.4 Power-On Reset

The power-on reset (POR) circuit clears the COP prescaler 4096 CGMXCLK cycles after power-up.

## 10.4.5 Internal Reset

An internal reset clears the COP prescaler and the COP counter.

## 10.4.6 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.

## 10.4.7 COPD (COP Disable)

The COPD signal reflects the state of the COP disable bit (COPD) in the configuration register. See [Section 9. Configuration Register \(CONFIG\)](#).

## 10.4.8 COPRS (COP Rate Select)

The COPRS signal reflects the state of the COP rate select bit (COPRS) in the configuration register. See [Section 9. Configuration Register \(CONFIG\)](#).

## 10.5 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:   | Writing clears COP counter, any value                      |
| Reset:   | Unaffected by reset  |

**Figure 10-2. COP Control Register (COPCTL)**

## 10.6 Interrupts

The COP does not generate CPU interrupt requests.

## 10.7 Monitor Mode

When monitor mode is entered with  $V_{TST}$  on the  $\overline{IRQ}$  pin, the COP is disabled as long as  $V_{TST}$  remains on the  $\overline{IRQ}$  pin or the  $\overline{RST}$  pin.

## 10.8 Low-Power Modes

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

### 10.8.1 Wait Mode

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

### 10.8.2 Stop Mode

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

To prevent inadvertently turning off the COP with a STOP instruction, a configuration option is available that disables the STOP instruction. When the STOP bit in the configuration register has the STOP instruction disabled, execution of a STOP instruction results in an illegal opcode reset.

### 10.9 COP Module During Break Mode

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



## Section 11. Central Processor Unit (CPU)

### 11.1 Contents

|        |                             |     |
|--------|-----------------------------|-----|
| 11.2   | Introduction                | 153 |
| 11.3   | Features                    | 154 |
| 11.4   | CPU Registers               | 155 |
| 11.4.1 | Accumulator                 | 155 |
| 11.4.2 | Index Register              | 156 |
| 11.4.3 | Stack Pointer               | 157 |
| 11.4.4 | Program Counter             | 158 |
| 11.4.5 | Condition Code Register     | 159 |
| 11.5   | Arithmetic/Logic Unit (ALU) | 161 |
| 11.6   | Low-Power Modes             | 161 |
| 11.6.1 | Wait Mode                   | 161 |
| 11.6.2 | Stop Mode                   | 161 |
| 11.7   | CPU During Break Interrupts | 162 |
| 11.8   | Instruction Set Summary     | 163 |
| 11.9   | Opcode Map                  | 169 |

### 11.2 Introduction

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

### 11.3 Features

Features 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

### 11.4 CPU Registers

Figure 11-1 shows the five CPU registers. CPU registers are not part of the memory map.

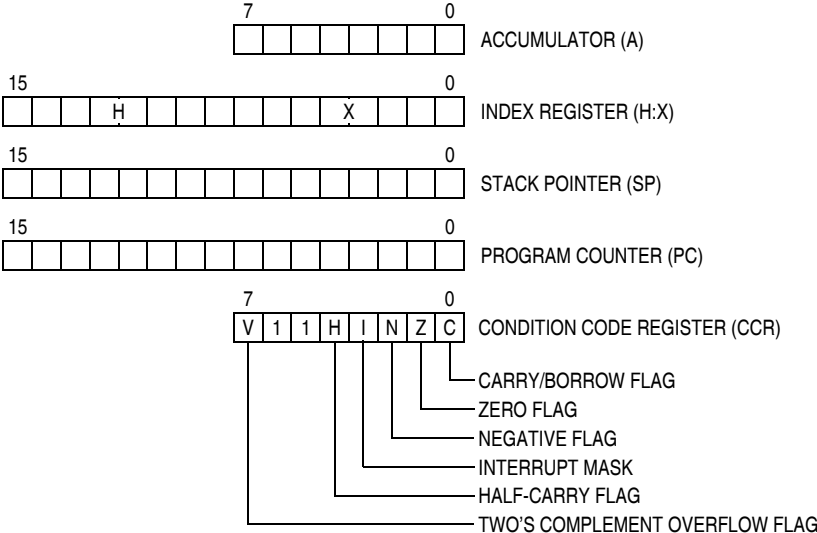


Figure 11-1. CPU Registers

#### 11.4.1 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.

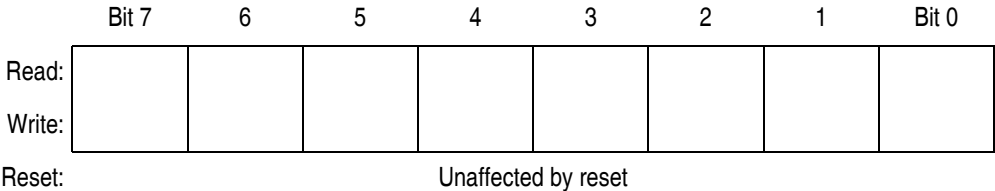


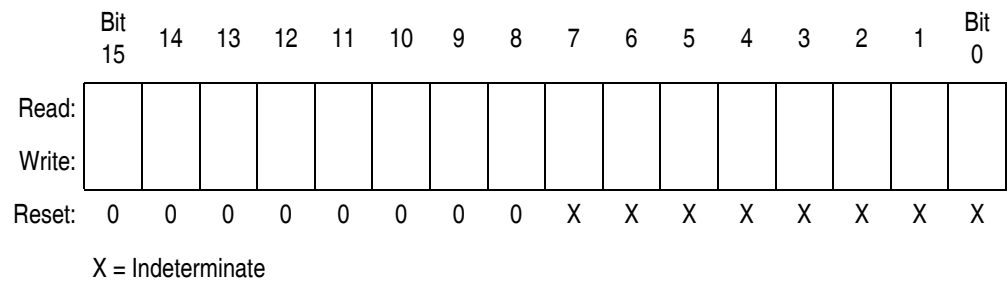
Figure 11-2. Accumulator (A)

## 11.4.2 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.

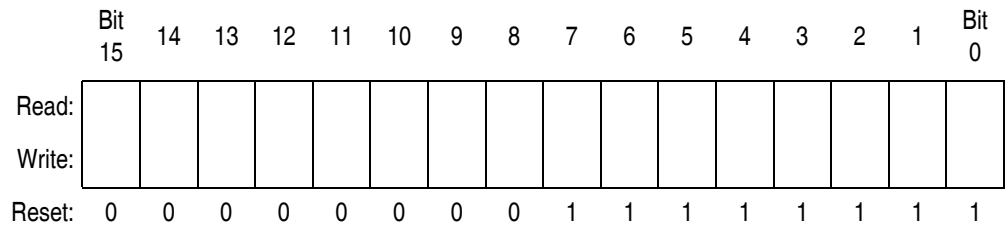


**Figure 11-3. Index Register (H:X)**

### 11.4.3 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 11-4. Stack Pointer (SP)**

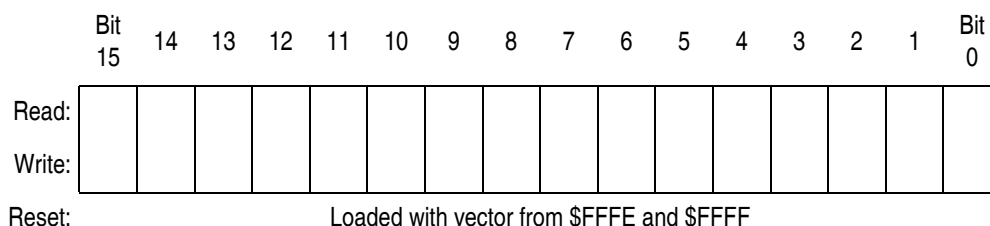
**NOTE:** *The location of the stack is arbitrary and may be relocated anywhere in RAM. Moving the SP out of page 0 (\$0000 to \$00FF) frees direct address (page 0) space. For correct operation, the stack pointer must point only to RAM locations.*

## 11.4.4 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 11-5. Program Counter (PC)**

### 11.4.5 Condition Code Register

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

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

X = Indeterminate

**Figure 11-6. Condition Code Register (CCR)**

#### V — Overflow Flag Bit

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 Bit

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 Bit

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 Bit

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 Bit

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 Bit

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



## 11.5 Arithmetic/Logic Unit (ALU)

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

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

## 11.6 Low-Power Modes

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

### 11.6.1 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

### 11.6.2 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.

### 11.7 CPU During Break Interrupts

If a break module is present on the MCU, the CPU starts a break interrupt by:

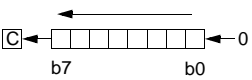
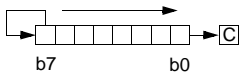
- Loading the instruction register with the SWI instruction
- Loading the program counter with \$FFFC:\$FFFD or with \$FEFC:\$FEFD in monitor mode

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.

## 11.8 Instruction Set Summary

Table 11-1. Instruction Set Summary (Sheet 1 of 7)

| 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  | $A \leftarrow (A) + (M) + (C)$  | ↑             | ↑ | - | ↑ | ↑ | ↑            | 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>ff<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                                       | $A \leftarrow (A) + (M)$  | ↑             | ↑ | - | ↑ | ↑ | ↑            | 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>ff<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                      | $SP \leftarrow (SP) + (16 \ll M)$   | -             | - | - | - | - | -            | IMM  | A7   | ii  | 2                                    |
| AIX #opr   | Add Immediate Value (Signed) to H:X                     | $H:X \leftarrow (H:X) + (16 \ll M)$   | -             | - | - | - | - | -            | 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   | $A \leftarrow (A) \& (M)$   | 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>ff<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>ff<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>ff<br>ff  | 4<br>1<br>1<br>4<br>3<br>5           |
| BCC rel  | Branch if Carry Bit Clear                               | $PC \leftarrow (PC) + 2 + rel ? (C) = 0$  | -             | - | - | - | - | -            | REL  | 24   | rr  | 3                                    |
| BCLR n, opr  | Clear Bit n in M  | $M_n \leftarrow 0$  | -             | - | - | - | - | -            | 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)                   | $PC \leftarrow (PC) + 2 + rel ? (C) = 1$  | -             | - | - | - | - | -            | REL  | 25   | rr  | 3                                    |
| BEQ rel  | Branch if Equal   | $PC \leftarrow (PC) + 2 + rel ? (Z) = 1$  | -             | - | - | - | - | -            | REL  | 27   | rr  | 3                                    |
| BGE opr  | Branch if Greater Than or Equal To<br>(Signed Operands) | $PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 0$                                   | -             | - | - | - | - | -            | REL  | 90   | rr  | 3                                    |

## Table 11-1. Instruction Set Summary (Sheet 2 of 7)

| Source Form  | Operation   | Description   | Effect on CCR |   |   |   |   | Address Mode   | Opcode  | Operand  | Cycles  |                                      |
|--|---|---|---------------|---|---|---|---|--|---|--|---|--------------------------------------|
|  |   |   | V             | H | I | N | Z |  |   |  |   | C                                    |
| BGT <i>opr</i>   | Branch if Greater Than (Signed Operands)          | $PC \leftarrow (PC) + 2 + rel ? (Z)   (N \oplus V) = 0$ | -             | - | - | - | - | REL  | 92  | rr   | 3   |                                      |
| BHCC <i>rel</i>  | Branch if Half Carry Bit Clear                    | $PC \leftarrow (PC) + 2 + rel ? (H) = 0$                | -             | - | - | - | - | REL  | 28  | rr   | 3   |                                      |
| BHCS <i>rel</i>  | Branch if Half Carry Bit Set                      | $PC \leftarrow (PC) + 2 + rel ? (H) = 1$                | -             | - | - | - | - | REL  | 29  | rr   | 3   |                                      |
| BHI <i>rel</i>   | Branch if Higher                                  | $PC \leftarrow (PC) + 2 + rel ? (C)   (Z) = 0$          | -             | - | - | - | - | REL  | 22  | rr   | 3   |                                      |
| BHS <i>rel</i>   | Branch if Higher or Same (Same as BCC)            | $PC \leftarrow (PC) + 2 + rel ? (C) = 0$                | -             | - | - | - | - | REL  | 24  | rr   | 3   |                                      |
| BIH <i>rel</i>   | Branch if $\overline{IRQ}$ Pin High               | $PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 1$     | -             | - | - | - | - | REL  | 2F  | rr   | 3   |                                      |
| BIL <i>rel</i>   | Branch if $\overline{IRQ}$ Pin Low                | $PC \leftarrow (PC) + 2 + rel ? \overline{IRQ} = 0$     | -             | - | - | - | - | REL  | 2E  | rr   | 3   |                                      |
| BIT # <i>opr</i><br>BIT <i>opr</i><br>BIT <i>opr</i><br>BIT <i>opr</i> ,X<br>BIT <i>opr</i> ,X<br>BIT ,X<br>BIT <i>opr</i> ,SP<br>BIT <i>opr</i> ,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>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| BLE <i>opr</i>   | Branch if Less Than or Equal To (Signed Operands) | $PC \leftarrow (PC) + 2 + rel ? (Z)   (N \oplus V) = 1$ | -             | - | - | - | - | REL  | 93  | rr   | 3   |                                      |
| BLO <i>rel</i>   | Branch if Lower (Same as BCS)                     | $PC \leftarrow (PC) + 2 + rel ? (C) = 1$                | -             | - | - | - | - | REL  | 25  | rr   | 3   |                                      |
| BLS <i>rel</i>   | Branch if Lower or Same                           | $PC \leftarrow (PC) + 2 + rel ? (C)   (Z) = 1$          | -             | - | - | - | - | REL  | 23  | rr   | 3   |                                      |
| BLT <i>opr</i>   | Branch if Less Than (Signed Operands)             | $PC \leftarrow (PC) + 2 + rel ? (N \oplus V) = 1$       | -             | - | - | - | - | REL  | 91  | rr   | 3   |                                      |
| BMC <i>rel</i>   | Branch if Interrupt Mask Clear                    | $PC \leftarrow (PC) + 2 + rel ? (I) = 0$                | -             | - | - | - | - | REL  | 2C  | rr   | 3   |                                      |
| BMI <i>rel</i>   | Branch if Minus                                   | $PC \leftarrow (PC) + 2 + rel ? (N) = 1$                | -             | - | - | - | - | REL  | 2B  | rr   | 3   |                                      |
| BMS <i>rel</i>   | Branch if Interrupt Mask Set                      | $PC \leftarrow (PC) + 2 + rel ? (I) = 1$                | -             | - | - | - | - | REL  | 2D  | rr   | 3   |                                      |
| BNE <i>rel</i>   | Branch if Not Equal                               | $PC \leftarrow (PC) + 2 + rel ? (Z) = 0$                | -             | - | - | - | - | REL  | 26  | rr   | 3   |                                      |
| BPL <i>rel</i>   | Branch if Plus                                    | $PC \leftarrow (PC) + 2 + rel ? (N) = 0$                | -             | - | - | - | - | REL  | 2A  | rr   | 3   |                                      |
| BRA <i>rel</i>   | Branch Always                                     | $PC \leftarrow (PC) + 2 + rel$                          | -             | - | - | - | - | REL  | 20  | rr   | 3   |                                      |
| BRCLR <i>n,opr,rel</i>   | Branch if Bit <i>n</i> in M Clear                 | $PC \leftarrow (PC) + 3 + rel ? (Mn) = 0$               | -             | - | - | - | ↓ | 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 <i>rel</i>   | Branch Never                                      | $PC \leftarrow (PC) + 2$                                | -             | - | - | - | - | REL  | 21  | rr   | 3   |                                      |
| BRSET <i>n,opr,rel</i>   | Branch if Bit <i>n</i> in M Set                   | $PC \leftarrow (PC) + 3 + rel ? (Mn) = 1$               | -             | - | - | - | ↑ | 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                  |                                      |

Table 11-1. Instruction Set Summary (Sheet 3 of 7)

| Source Form   | Operation                     | Description  | Effect on CCR |   |   |   |   | Address Mode | Opcode   | Operand  | Cycles  |                                      |
|---|-------------------------------|--|---------------|---|---|---|---|--------------|--|--|---|--------------------------------------|
|   |                               |  | V             | H | I | N | Z |              |  |  |   | C                                    |
| BSET <i>n,opr</i>   | Set Bit <i>n</i> in M         | $M_n \leftarrow 1$   | -             | - | - | - | - | -            | 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 <i>rel</i>  | Branch to Subroutine          | PC ← (PC) + 2; push (PCL)<br>SP ← (SP) - 1; push (PCH)<br>SP ← (SP) - 1<br>PC ← (PC) + <i>rel</i>  | -             | - | - | - | - | -            | REL  | AD   | rr  | 4                                    |
| CBEQ <i>opr,rel</i><br>CBEQA # <i>opr,rel</i><br>CBEQX # <i>opr,rel</i><br>CBEQ <i>opr,X+,rel</i><br>CBEQ <i>X+,rel</i><br>CBEQ <i>opr,SP,rel</i> | Compare and Branch if Equal   | PC ← (PC) + 3 + <i>rel</i> ? (A) - (M) = \$00<br>PC ← (PC) + 3 + <i>rel</i> ? (A) - (M) = \$00<br>PC ← (PC) + 3 + <i>rel</i> ? (X) - (M) = \$00<br>PC ← (PC) + 3 + <i>rel</i> ? (A) - (M) = \$00<br>PC ← (PC) + 2 + <i>rel</i> ? (A) - (M) = \$00<br>PC ← (PC) + 4 + <i>rel</i> ? (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 rr<br>ff rr    | 5<br>4<br>4<br>5<br>4<br>6           |
| CLC   | Clear Carry Bit               | $C \leftarrow 0$   | -             | - | - | - | - | 0            | INH  | 98   |   | 1                                    |
| CLI   | Clear Interrupt Mask          | $I \leftarrow 0$   | -             | - | 0 | - | - | -            | INH  | 9A   |   | 2                                    |
| CLR <i>opr</i><br>CLRA<br>CLR X<br>CLR H<br>CLR <i>opr,X</i><br>CLR ,X<br>CLR <i>opr,SP</i>   | Clear                         | $M \leftarrow \$00$<br>$A \leftarrow \$00$<br>$X \leftarrow \$00$<br>$H \leftarrow \$00$<br>$M \leftarrow \$00$<br>$M \leftarrow \$00$<br>$M \leftarrow \$00$  | 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>ff<br>ff                      | 3<br>1<br>1<br>1<br>3<br>2<br>4      |
| CMP # <i>opr</i><br>CMP <i>opr</i><br>CMP <i>opr</i><br>CMP <i>opr,X</i><br>CMP <i>opr,X</i><br>CMP ,X<br>CMP <i>opr,SP</i><br>CMP <i>opr,SP</i>  | 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>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| COM <i>opr</i><br>COMA<br>COM X<br>COM <i>opr,X</i><br>COM ,X<br>COM <i>opr,SP</i>  | Complement (One's Complement) | $M \leftarrow (\overline{M}) = \$FF - (M)$<br>$A \leftarrow (\overline{A}) = \$FF - (M)$<br>$X \leftarrow (\overline{X}) = \$FF - (M)$<br>$M \leftarrow (\overline{M}) = \$FF - (M)$<br>$M \leftarrow (\overline{M}) = \$FF - (M)$<br>$M \leftarrow (\overline{M}) = \$FF - (M)$                   | 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>ff<br>ff                          | 4<br>1<br>1<br>4<br>3<br>5           |
| CPHX # <i>opr</i><br>CPHX <i>opr</i>  | 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 # <i>opr</i><br>CPX <i>opr</i><br>CPX <i>opr</i><br>CPX ,X<br>CPX <i>opr,X</i><br>CPX <i>opr,X</i><br>CPX <i>opr,SP</i><br>CPX <i>opr,SP</i>  | 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>ff<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) <sub>10</sub>  | U             | - | - | ↑ | ↑ | ↑            | INH  | 72   |   | 2                                    |

## Table 11-1. Instruction Set Summary (Sheet 4 of 7)

| Source Form  | Operation                        | Description  | Effect on CCR |   |   |   |   | Address Mode | Opcode  | Operand  | Cycles  |                                      |
|--|----------------------------------|--|---------------|---|---|---|---|--------------|---|--|---|--------------------------------------|
|  |                                  |  | V             | H | I | N | Z |              |   |  |   | C                                    |
| DBNZ <i>opr,rel</i><br>DBNZ <i>A,rel</i><br>DBNZ <i>X,rel</i><br>DBNZ <i>opr,X,rel</i><br>DBNZ <i>X,rel</i><br>DBNZ <i>opr,SP,rel</i>                  | Decrement and Branch if Not Zero | $A \leftarrow (A) - 1$ or $M \leftarrow (M) - 1$ or $X \leftarrow (X) - 1$<br>$PC \leftarrow (PC) + 3 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 3 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 2 + rel ? (result) \neq 0$<br>$PC \leftarrow (PC) + 4 + rel ? (result) \neq 0$ | -             | - | - | - | - | -            | 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 <i>opr</i><br>DECA<br>DECX<br>DEC <i>opr,X</i><br>DEC <i>,X</i><br>DEC <i>opr,SP</i>   | Decrement                        | $M \leftarrow (M) - 1$<br>$A \leftarrow (A) - 1$<br>$X \leftarrow (X) - 1$<br>$M \leftarrow (M) - 1$<br>$M \leftarrow (M) - 1$<br>$M \leftarrow (M) - 1$   | ↓             | - | - | ↓ | ↓ | -            | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 3A<br>4A<br>5A<br>6A<br>7A<br>9E6A               | dd<br>ff<br>ff  | 4<br>1<br>1<br>4<br>3<br>5           |
| DIV  | Divide                           | $A \leftarrow (H:A)/(X)$<br>$H \leftarrow \text{Remainder}$  | -             | - | - | - | ↓ | ↓            | INH   | 52   |   | 7                                    |
| EOR <i>#opr</i><br>EOR <i>opr</i><br>EOR <i>opr</i><br>EOR <i>opr,X</i><br>EOR <i>opr,X</i><br>EOR <i>,X</i><br>EOR <i>opr,SP</i><br>EOR <i>opr,SP</i> | Exclusive OR M with A            | $A \leftarrow (A \oplus M)$  | 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>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| INC <i>opr</i><br>INCA<br>INCX<br>INC <i>opr,X</i><br>INC <i>,X</i><br>INC <i>opr,SP</i>   | Increment                        | $M \leftarrow (M) + 1$<br>$A \leftarrow (A) + 1$<br>$X \leftarrow (X) + 1$<br>$M \leftarrow (M) + 1$<br>$M \leftarrow (M) + 1$<br>$M \leftarrow (M) + 1$   | ↓             | - | - | ↓ | ↓ | -            | DIR<br>INH<br>INH<br>IX1<br>IX<br>SP1               | 3C<br>4C<br>5C<br>6C<br>7C<br>9E6C               | dd<br>ff<br>ff  | 4<br>1<br>1<br>4<br>3<br>5           |
| JMP <i>opr</i><br>JMP <i>opr</i><br>JMP <i>opr,X</i><br>JMP <i>opr,X</i><br>JMP <i>,X</i>  | Jump                             | $PC \leftarrow \text{Jump Address}$  | -             | - | - | - | - | -            | 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 <i>opr</i><br>JSR <i>opr</i><br>JSR <i>opr,X</i><br>JSR <i>opr,X</i><br>JSR <i>,X</i>  | Jump to Subroutine               | $PC \leftarrow (PC) + n$ ( $n = 1, 2,$ or $3$ )<br>Push (PCL); $SP \leftarrow (SP) - 1$<br>Push (PCH); $SP \leftarrow (SP) - 1$<br>$PC \leftarrow \text{Unconditional Address}$  | -             | - | - | - | - | -            | 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 <i>#opr</i><br>LDA <i>opr</i><br>LDA <i>opr</i><br>LDA <i>opr,X</i><br>LDA <i>opr,X</i><br>LDA <i>,X</i><br>LDA <i>opr,SP</i><br>LDA <i>opr,SP</i> | Load A from M                    | $A \leftarrow (M)$   | 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>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| LDHX <i>#opr</i><br>LDHX <i>opr</i>  | Load H:X from M                  | $H:X \leftarrow (M:M + 1)$   | 0             | - | - | ↓ | ↓ | -            | IMM<br>DIR  | 45<br>55   | ii jj<br>dd   | 3<br>4                               |
| LDX <i>#opr</i><br>LDX <i>opr</i><br>LDX <i>opr</i><br>LDX <i>opr,X</i><br>LDX <i>opr,X</i><br>LDX <i>,X</i><br>LDX <i>opr,SP</i><br>LDX <i>opr,SP</i> | Load X from M                    | $X \leftarrow (M)$   | 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>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |

Table 11-1. Instruction Set Summary (Sheet 5 of 7)

| Source Form  | Operation                           | Description   | Effect on CCR |   |   |   |   | Address Mode | Opcode  | Operand  | Cycles                                    |                                      |
|--|-------------------------------------|---|---------------|---|---|---|---|--------------|---|--|---|--------------------------------------|
|  |                                     |   | V             | H | I | N | Z |              |   |  |   | C                                    |
| LSL <i>opr</i><br>LSLA<br>LSLX<br>LSL <i>opr,X</i><br>LSL ,X<br>LSL <i>opr,SP</i>  | 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>ff<br>ff                            | 4<br>1<br>1<br>4<br>3<br>5           |
| LSR <i>opr</i><br>LSRA<br>LSRX<br>LSR <i>opr,X</i><br>LSR ,X<br>LSR <i>opr,SP</i>  | 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>ff<br>ff                            | 4<br>1<br>1<br>4<br>3<br>5           |
| MOV <i>opr,opr</i><br>MOV <i>opr,X+</i><br>MOV # <i>opr,opr</i><br>MOV X+, <i>opr</i>  | Move                                | (M) <sub>Destination</sub> ← (M) <sub>Source</sub><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 ii dd<br>dd                   | 5<br>4<br>4<br>4                     |
| MUL  | Unsigned multiply                   | X:A ← (X) × (A)   | -             | 0 | - | - | - | 0            | INH   | 42   |   | 5                                    |
| NEG <i>opr</i><br>NEGA<br>NEGX<br>NEG <i>opr,X</i><br>NEG ,X<br>NEG <i>opr,SP</i>  | 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>ff<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 # <i>opr</i><br>ORA <i>opr</i><br>ORA <i>opr</i><br>ORA <i>opr,X</i><br>ORA <i>opr,X</i><br>ORA ,X<br>ORA <i>opr,SP</i><br>ORA <i>opr,SP</i> | 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 dd<br>hh ll<br>ee ff<br>ff<br>ff 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                                    |
| 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 <i>opr</i><br>ROLA<br>ROLX<br>ROL <i>opr,X</i><br>ROL ,X<br>ROL <i>opr,SP</i>  | 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>ff<br>ff                            | 4<br>1<br>1<br>4<br>3<br>5           |
| ROR <i>opr</i><br>RORA<br>RORX<br>ROR <i>opr,X</i><br>ROR ,X<br>ROR <i>opr,SP</i>  | 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>ff<br>ff                            | 4<br>1<br>1<br>4<br>3<br>5           |

# Central Processor Unit (CPU)

Table 11-1. Instruction Set Summary (Sheet 6 of 7)

| Source Form  | Operation                                    | Description   | Effect on CCR |   |   |   |   |   | Address Mode  | Opcode   | Operand   | Cycles                               |
|--|--|---|---------------|---|---|---|---|---|---|--|---|--------------------------------------|
|  |  |   | V             | H | I | N | Z | C |   |  |   |                                      |
| RSP  | Reset Stack Pointer                          | $SP \leftarrow \$FF$  | -             | - | - | - | - | - | INH   | 9C   |   | 1                                    |
| RTI  | Return from Interrupt                        | $SP \leftarrow (SP) + 1$ ; Pull (CCR)<br>$SP \leftarrow (SP) + 1$ ; Pull (A)<br>$SP \leftarrow (SP) + 1$ ; Pull (X)<br>$SP \leftarrow (SP) + 1$ ; Pull (PCH)<br>$SP \leftarrow (SP) + 1$ ; Pull (PCL)   | ↑             | ↑ | ↑ | ↑ | ↑ | ↑ | INH   | 80   |   | 7                                    |
| RTS  | Return from Subroutine                       | $SP \leftarrow SP + 1$ ; Pull (PCH)<br>$SP \leftarrow SP + 1$ ; 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                          | $A \leftarrow (A) - (M) - (C)$  | ↑             | - | - | ↑ | ↑ | ↑ | 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>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SEC  | Set Carry Bit                                | $C \leftarrow 1$  | -             | - | - | - | - | 1 | INH   | 99   |   | 1                                    |
| SEI  | Set Interrupt Mask                           | $I \leftarrow 1$  | -             | - | 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                                 | $M \leftarrow (A)$  | 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>ff<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                               | $(M:M + 1) \leftarrow (H:X)$  | 0             | - | - | ↑ | ↑ | - | DIR   | 35   | dd  | 4                                    |
| STOP   | Enable $\overline{IRQ}$ Pin; Stop Oscillator | $I \leftarrow 0$ ; Stop Oscillator  | -             | - | 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                                 | $M \leftarrow (X)$  | 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>ff<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                                     | $A \leftarrow (A) - (M)$  | ↑             | - | - | ↑ | ↑ | ↑ | 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>ff<br>ff<br>ee ff | 2<br>3<br>4<br>4<br>3<br>2<br>4<br>5 |
| SWI  | Software Interrupt                           | $PC \leftarrow (PC) + 1$ ; Push (PCL)<br>$SP \leftarrow (SP) - 1$ ; Push (PCH)<br>$SP \leftarrow (SP) - 1$ ; Push (X)<br>$SP \leftarrow (SP) - 1$ ; Push (A)<br>$SP \leftarrow (SP) - 1$ ; Push (CCR)<br>$SP \leftarrow (SP) - 1$ ; $I \leftarrow 1$<br>PCH $\leftarrow$ Interrupt Vector High Byte<br>PCL $\leftarrow$ Interrupt Vector Low Byte | -             | - | 1 | - | - | - | INH   | 83   |   | 9                                    |
| TAP  | Transfer A to CCR                            | $CCR \leftarrow (A)$  | ↑             | ↑ | ↑ | ↑ | ↑ | ↑ | INH   | 84   |   | 2                                    |
| TAX  | Transfer A to X                              | $X \leftarrow (A)$  | -             | - | - | - | - | - | INH   | 97   |   | 1                                    |



**Table 11-1. Instruction Set Summary (Sheet 7 of 7)**

| Source Form  | Operation                 | Description                                  | Effect on CCR |   |   |   |   |   | Address Mode                          | Opcode                             | Operand        | Cycles                     |
|--|---------------------------|--|---------------|---|---|---|---|---|---------------------------------------|------------------------------------|----------------|----------------------------|
|  |                           |  | V             | H | I | N | Z | C |                                       |                                    |                |                            |
| TPA  | Transfer CCR to A         | $A \leftarrow (CCR)$                         | -             | - | - | - | - | - | INH                                   | 85                                 |                | 1                          |
| TST <i>opr</i><br>TSTA<br>TSTX<br>TST <i>opr,X</i><br>TST <i>,X</i><br>TST <i>opr,SP</i> | 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>ff<br>ff | 3<br>1<br>1<br>3<br>2<br>4 |
| TSX  | Transfer SP to H:X        | $H:X \leftarrow (SP) + 1$                    | -             | - | - | - | - | - | INH                                   | 95                                 |                | 2                          |
| TXA  | Transfer X to A           | $A \leftarrow (X)$                           | -             | - | - | - | - | - | INH                                   | 9F                                 |                | 1                          |
| TXS  | Transfer H:X to SP        | $(SP) \leftarrow (H:X) - 1$                  | -             | - | - | - | - | - | INH                                   | 94                                 |                | 2                          |

- |       |   |            |   |
|-------|---|------------|---|
| A     | Accumulator   | <i>n</i>   | Any bit                                     |
| C     | Carry/borrow bit  | <i>opr</i> | 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  | <i>rel</i> | 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                                |

## 11.9 Opcode Map

See [Table 11-2](#).

**Table 11-2. Opcode Map**

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

Low Byte of Opcode in Hexadecimal

|            |                 |
|------------|-----------------|
| MSB<br>LSB | 0               |
| 0          | BRSET0<br>3 DIR |

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

\*Pre-byte for stack pointer indexed instructions

## Section 12. User FLASH Memory

### 12.1 Contents

|        |   |     |
|--------|---|-----|
| 12.2   | Introduction . . . . .                        | 171 |
| 12.3   | Functional Description . . . . .              | 171 |
| 12.4   | FLASH Control Register . . . . .              | 173 |
| 12.5   | Charge Pump . . . . .                         | 175 |
| 12.5.1 | FLASH Charge Pump Frequency Control . . . . . | 175 |
| 12.6   | FLASH Erase Operation . . . . .               | 176 |
| 12.7   | FLASH Program/Margin Read Operation . . . . . | 178 |
| 12.8   | User FLASH Block Protection . . . . .         | 181 |
| 12.9   | User FLASH Block Protect Register . . . . .   | 182 |
| 12.10  | Wait Mode . . . . .                           | 183 |
| 12.11  | Stop Mode . . . . .                           | 183 |

### 12.2 Introduction

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

### 12.3 Functional Description

The user FLASH memory is an array of 24,064 bytes with an additional 22 bytes of user vectors and two bytes for block protection. An erased bit reads as logic 0 and a programmed bit reads as a logic 1. Program and erase operations are facilitated through control bits in a memory mapped register. Details for these operations appear later in this section.

Memory in the user FLASH array is organized into pages within rows. There are eight pages of memory per row with eight bytes per page. The minimum erase block size is a single row, 64 bytes. Programming is performed on a per page basis; eight bytes at a time. The address ranges for the user memory and vectors are:

- \$A000–\$FDFF; user memory
- \$FF80–\$FF81; block protect registers
- \$FFEA–\$FFFF; locations reserved for user-defined interrupt and reset vectors

When programming the FLASH, just enough program time must be used to program a page. Too much program time can result in a program disturb condition, in which case an erased bit on the row being programmed becomes unintentionally programmed. Program disturb is avoided by using an iterative program and margin read technique known as the smart page programming algorithm. The smart programming algorithm is required whenever programming the array (see [12.7 FLASH Program/Margin Read Operation](#)).

To avoid the program disturb issue, each row should not be programmed more than eight times before it is erased. The eight program cycle maximum per row aligns with the architecture's eight pages of storage per row. The margin read step of the smart programming algorithm is used to ensure programmed bits are programmed to sufficient margin for data retention over the device lifetime.

Row architecture for this array is:

- \$A000–\$A03F; row 0
- \$A040–\$A07F; row 1
- \$A080–\$A0BF; row 2
- -----
- \$FFC0–\$FFFF; row 383

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

**NOTE:** A security feature prevents viewing of the FLASH contents.<sup>1</sup>

## 12.4 FLASH Control Register

The user FLASH control register (FL1CR) controls FLASH program, erase, and margin read operations.

Address: \$FE07

|        | Bit 7 | 6     | 5    | 4    | 3    | 2      | 1     | Bit 0 |
|--------|-------|-------|------|------|------|--------|-------|-------|
| Read:  | FDIV1 | FDIV0 | BLK1 | BLK0 | HVEN | MARGIN | ERASE | PGM   |
| Write: |       |       |      |      |      |        |       |       |
| Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |

**Figure 12-1. User FLASH Control Register (FL1CR)**

### FDIV1 — Frequency Divide Control Bit

This read/write bit together with FDIV0 selects the value by which the charge pump clock is divided from the system clock. See [12.5.1 FLASH Charge Pump Frequency Control](#).

### FDIV0 — Frequency Divide Control Bit

This read/write bit together with FDIV1 selects the value by which the charge pump clock is divided from the system clock. See [12.5.1 FLASH Charge Pump Frequency Control](#).

### BLK1 — Block Erase Control Bit

This read/write bit together with BLK0 allows erasing of blocks of varying size. See [12.6 FLASH Erase Operation](#) for a description of available block sizes.

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

### BLK0 — Block Erase Control Bit

This read/write bit together with BLK1 allows erasing of blocks of varying size. See [12.6 FLASH Erase Operation](#) for a description of available block sizes.

### 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/margin read or erase is followed.

1 = High voltage enabled to array and charge pump on

0 = High voltage disabled to array and charge pump off

### MARGIN — Margin Read Control Bit

This read/write bit configures the memory for margin read operation. MARGIN cannot be set if the HVEN = 1. MARGIN will return to unset automatically if asserted when HVEN is set.

1 = Margin read operation selected

0 = Margin read operation unselected

### ERASE — Erase Control Bit

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

1 = Erase operation selected

0 = Erase operation 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

## 12.5 Charge Pump

The internal FLASH charge pump is an analog circuit that provides the proper voltage to the FLASH memory when reading, programming, and erasing the memory arrays. Only one charge pump circuit services both FLASH memories.

### 12.5.1 FLASH Charge Pump Frequency Control

The internal charge pump required for program, margin read, and erase operations is designed to operate most efficiently with a 2-MHz clock. The charge pump clock is derived from the bus clock. [Table 12-1](#) shows how the FDIV bits are used to select a charge pump frequency based on the bus clock frequency. Program and erase operations cannot be performed if the bus clock frequency is below 2 MHz.

**Table 12-1. Charge Pump Clock Frequency**

| FDIV1 | FDIV0 | Pump Clock Frequency   | Bus Clock Frequency |
|-------|-------|------------------------|---------------------|
| 0     | 0     | Bus frequency $\div$ 1 | 1.8 to 2.5 MHz      |
| 0     | 1     | Bus frequency $\div$ 2 | 3.6 to 5.0 MHz      |
| 1     | 0     | Bus frequency $\div$ 2 | 3.6 to 5.0 MHz      |
| 1     | 1     | Bus frequency $\div$ 4 | 7.2 to 10.0 MHz     |

**NOTE:** *Since only one charge pump circuit services both FLASH arrays, the actual FDIV bits received by the charge pump are the logical OR of the individual FDIV bits of registers FL1CR and FL2CR.*

## 12.6 FLASH Erase Operation

Use the following procedure to erase a block of FLASH memory (to read as logic 0). Values for the time parameters are specified in [25.11 Memory Characteristics](#).

1. Set the ERASE bit, the BLK0, BLK1, FDIV0, and FDIV1 bits in the user FLASH control register (\$FE07). See [Table 12-1](#) for FDIV settings. See [Table 12-2](#) for block sizes.
2. Ensure that the block to be erased is not protected by the settings in the FL1BPR register. Read the FL1BPR register. If test voltage,  $V_{TST}$ , is applied to the  $\overline{IRQ}$  pin, block protection is bypassed. This bypass is useful when the entire array (including the FL1BPR register) needs to be erased. See [12.9 User FLASH Block Protect Register](#).
3. Write to any FLASH address with any data within the block address range desired. If the address is in a protected area, the ERASE bit will be cleared and the following steps of the erase procedure are blocked.
4. Set the HVEN bit.
5. Wait for a time,  $t_{Erase}$ .
6. Clear the HVEN bit.
7. Wait for a time,  $t_{Kill}$ , for the high voltages to dissipate.
8. Clear the ERASE bit.
9. After a time,  $t_{HVD}$ , the memory can be accessed again in read mode.

**NOTE:** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Do not exceed  $t_{Erase}$  maximum.*

[Table 12-2](#) shows the various block sizes which can be erased in one erase operation.



**Table 12-2. Erase Block Sizes**

| Write to Address Bits            | Address Value                        | BLK1 | BLK0 | Desired Erase Address Range   | Array Size                    |
|----------------------------------|--------------------------------------|------|------|---|-------------------------------|
| Any FLASH address                | Any                                  | 0    | 0    | \$A000–\$FDFF and \$FFEA–\$FFFF   | Full array: 24 Kbytes         |
| A <sub>15</sub> :A <sub>14</sub> | 11                                   | 0    | 1    | \$C000–\$FDFF and \$FFEA–\$FFFF   | Upper 2/3 array:<br>16 Kbytes |
| A <sub>15</sub> :A <sub>14</sub> | 10                                   | 0    | 1    | \$A000–\$BFFF   | Lower 1/3 array:<br>8 Kbytes  |
| A <sub>15</sub> :A <sub>9</sub>  | {1,A <sub>14</sub> :A <sub>9</sub> } | 1    | 0    | {1, A <sub>14</sub> :A <sub>9</sub> , 00000000} to<br>{1, A <sub>14</sub> :A <sub>9</sub> , 11111111} | Eight rows: 512 bytes         |
| A <sub>15</sub> :A <sub>6</sub>  | {1,A <sub>14</sub> :A <sub>6</sub> } | 1    | 1    | {1, A <sub>14</sub> :A <sub>6</sub> , 000000} to<br>{1, A <sub>14</sub> :A <sub>6</sub> , 111111}     | Single row: 64 bytes          |

In step 3 of the erase operation, the desired erase addresses are latched and used to determine the location of the block to be erased. For the full array (BLK1 = BLK0 = 0), the only requirement is that the user FLASH memory is selected. Writing to any address in the range \$A000 to \$FDFF or the vectors in the address range \$FFEA to \$FFFF will enable the full array erase.

In the “upper 2/3 array” case in [Table 12-2](#) (BLK1 = 0, BLK0 = 1), the state of A<sub>15</sub>:A<sub>14</sub> = 11 determines that the range from \$C000 to \$FDFF and \$FFEA to \$FFFF is erased. For example, writing to address \$D123 will erase the range \$C000 to \$FDFF and \$FFEA to \$FFFF.

In the “lower 1/3 array” case (BLK1 = 0, BLK0 = 1), the state of A<sub>15</sub>:A<sub>14</sub> = 10 determines that the range from \$A000 to \$BFFF is erased. For example, writing to address \$B123 will erase the range \$A000 to \$BFFF.

In the “eight row case” (BLK1 = 1, BLK0 = 0), 512-byte blocks are erased as determined by upper addresses A<sub>15</sub>–A<sub>9</sub>. For example, writing to address \$FB10 will erase the range \$FA00 to \$FBFF.

In the “single row case” (BLK1 = 1, BLK0 = 1), 64-byte blocks are erased as determined by upper addresses A<sub>15</sub>–A<sub>6</sub>. For example, writing to address \$BC60 will erase the range \$BC40 to \$BC7F.

### 12.7 FLASH Program/Margin Read Operation

**NOTE:** *After a total of eight program operations have been applied to a row, the row must be erased before further programming to avoid program disturb. An erased byte will read \$00.*

Programming of the user FLASH memory is done on a page basis. A page consists of eight consecutive bytes starting from address \$XXX0 or \$XXX8. The smart programming algorithm is required to program every page in the FLASH memory. The smart programming algorithm is defined as an iterative program and margin read sequence. Every page programming pulse ( $t_{\text{Step}}$  duration) is followed by a margin read. A margin read imposes a more stringent read condition on the bitcell than an ordinary read. As part of the margin read, a built-in counter stretches the data access for an additional eight cycles to allow sensing of the lower bitcell current. During these eight stretch cycles, the COP counter continues to run. The user must account for these extra cycles within COP feed loops.

The steps of programming and margin reading repeats until the data being programmed is identical to the margin read data. This iterative process will ensure that data has been programmed with sufficient margin for long-term data retention. Note that a margin read operation can only follow a page programming operation.

**NOTE:** *To overwrite a memory location, it must first be erased to 0s then programmed to the new value. For instance, if a location previously has been programmed to \$AA (1010 1010 binary) and the value should be changed to \$55 (0101 0101 binary), it is necessary to erase \$AA to \$00 first before programming to \$55. If the erase operation in this example is not performed and \$AA is simply re-programmed to \$55, then the location will be read as \$FF (1111 1111 binary). (0s cannot be programmed. 0s only result from the erase operation.)*

The smart programming algorithm consists of these steps. A flowchart for the algorithm is shown in [Figure 12-2](#). Values for the time parameters are specified in [25.11 Memory Characteristics](#).

1. Set the PGM bit in FL1CR. This configures the memory for program operation and enables the latching of address and data for programming.

2. Ensure that the block to be programmed is not protected by the settings in the FL1BPR register. Read the FL1BPR register. If test voltage,  $V_{TST}$ , is applied to the  $\overline{IRQ}$  pin, block protection is bypassed. This bypass is useful when the entire array (including the FL1BPR register) need to be altered. See [12.9 User FLASH Block Protect Register](#).
3. Write data to the eight bytes of the page being programmed. This requires eight separate write operations.
4. Set the HVEN bit.
5. Wait for a time,  $t_{Step}$ .
6. Clear the HVEN bit.
7. Wait for a time,  $t_{HVTV}$ .
8. Set the MARGIN bit.
9. Wait for a time,  $t_{VTP}$ .
10. Clear the PGM bit.
11. Wait for a time,  $t_{HVD}$ .
12. Read the eight data locations written in step 3. This is a margin read. Each read operation is stretched by eight cycles.
13. Clear the MARGIN bit.

If margin read data is identical to write data then programming is complete. If this verify step fails, repeat from step 2.

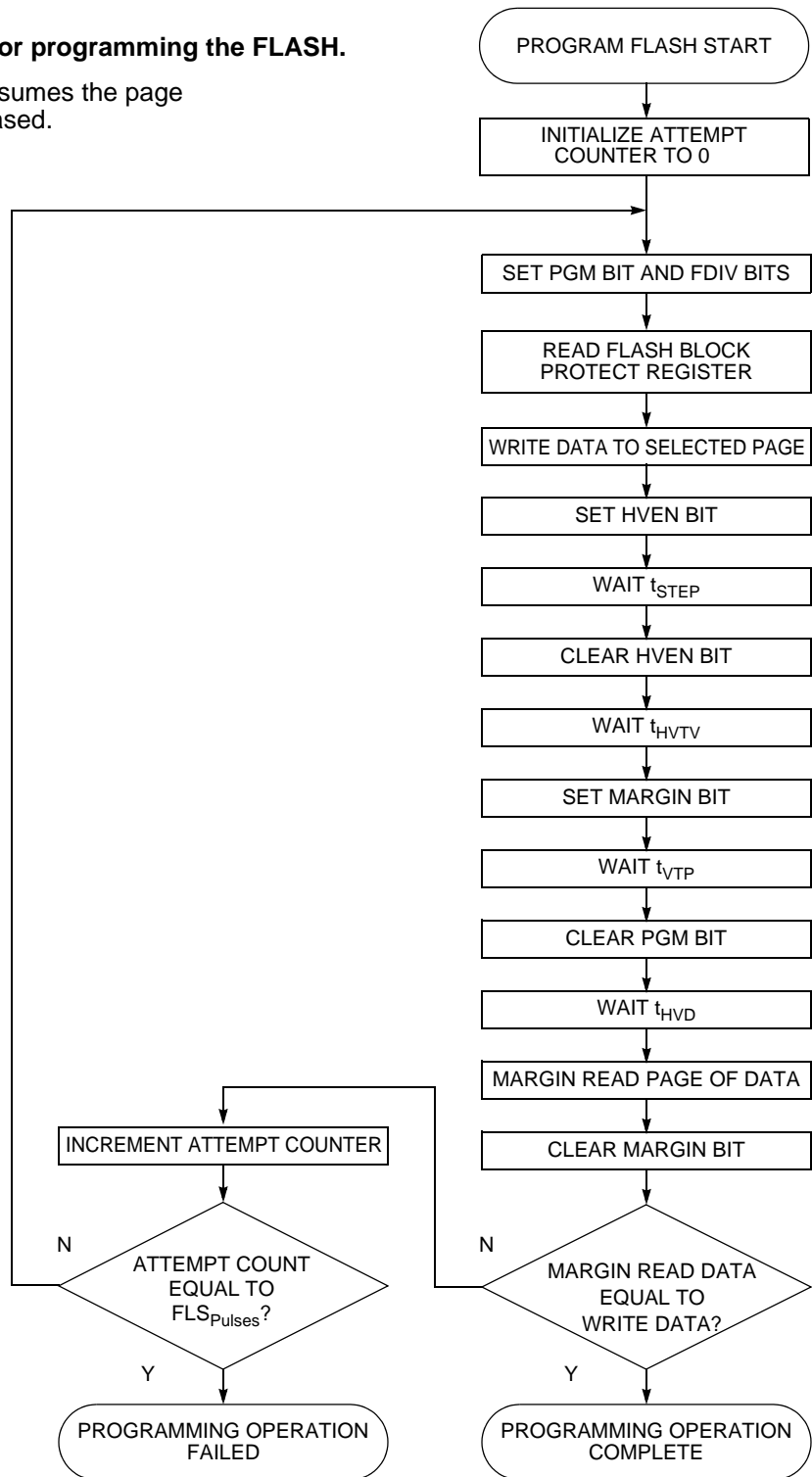
**NOTE:** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps.*

This algorithm, which should be repeated for all data to be programmed, guarantees the minimum possible program time and avoids the program disturb effect.

**NOTE:** *To ensure proper FLASH read operation after completion of the smart programming algorithm, a series of 500 FLASH dummy reads must be performed of any address before accurate data is read from the FLASH.*

**Note: This algorithm is mandatory for programming the FLASH.**

Note: This page program algorithm assumes the page to be programmed is initially erased.



**Figure 12-2. Smart Programming Algorithm Flowchart**

## 12.8 User FLASH Block Protection

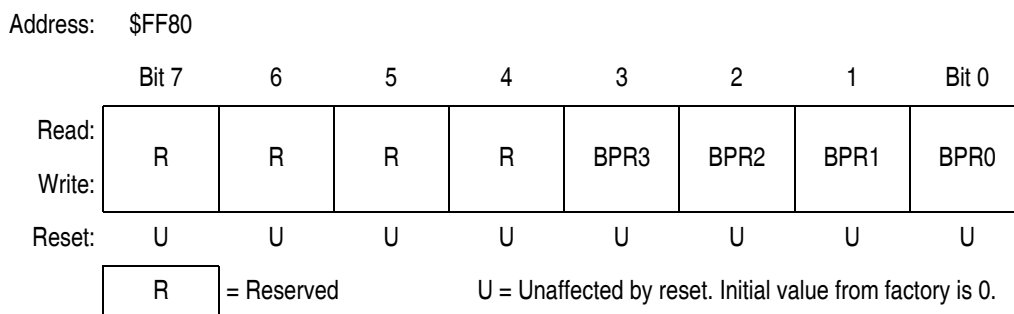
Due to the ability of the on-board charge pump to erase and program the FLASH memory in the target application, provision is made for protecting blocks of memory from unintentional erase or program operations due to system malfunction. This protection is done by reserving a location in the memory for block protect information and requiring that this location be read to enable setting of the HVEN bit. When the block protect register is read, its contents are latched by the FLASH control logic. If the address range for an erase or program operation includes a protected block, the PGM or ERASE bit is cleared which prevents the HVEN bit in the FLASH control register from being set so that no high voltage is allowed in the array.

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

When the block protect register is erased (all 0s), the entire memory is accessible for program and erase. When bits within the register are programmed, they lock blocks of memory address ranges as shown in [12.9 User FLASH Block Protect Register](#). The presence of a voltage  $V_{TST}$  on the  $\overline{IRQ}$  pin will bypass the block protection so that all of the memory, including the block protect register, is open for program and erase operations. Be aware that a  $V_{TST}$  voltage on  $\overline{IRQ}$  when coming out of reset may force entry into monitor mode. See [16.4.1 Entering Monitor Mode](#).

## 12.9 User FLASH Block Protect Register

The block protect register (FL1BPR) is implemented as a byte within the user FLASH memory. Each bit, when programmed, protects a range of addresses in the user FLASH.



**Figure 12-3. User FLASH Block Protect Register (FL1BPR)**

### BPR3 — Block Protect Register Bit 3

This bit protects the memory contents in the address range \$F000 to \$FFFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR2 — Block Protect Register Bit 2

This bit protects the memory contents in the address range \$E000 to \$FFFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR1 — Block Protect Register Bit 1

This bit protects the memory contents in the address range \$C000 to \$FFFF.

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

### BPR0 — Block Protect Register Bit 0

This bit protects the entire array (\$A000 to \$FFFF).

- 1 = Address range protected from erase or program
- 0 = Address range open to erase or program

By programming the block protect bits, a portion of the memory will be locked so that no further erase or program operations may be performed. Programming more than one bit at a time is redundant. If both BPR3 and BPR2 are set, for instance, the address range \$E000 through \$FFFF is locked. If all bits are erased, then all of the memory is available for erase and program. The presence of a voltage  $V_{TST}$  on the  $\overline{IRQ}$  pin will bypass the block protection so that all of the memory, including the block protect register, is open for program and erase operations.

## 12.10 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, but there will not be any memory activity since the CPU is inactive.

The WAIT instruction should not be executed while performing a program or erase operation on the FLASH. When the MCU is put into wait mode, the charge pump for the FLASH is disabled so that either a program or erase operation will not continue. If the memory is in either program mode (PGM = 1, HVEN = 1) or erase mode (ERASE = 1, HVEN = 1), then it will remain in that mode during wait. Exit from wait must now be done with a reset rather than an interrupt because if exiting wait with an interrupt, the memory will not be in read mode and the interrupt vector cannot be read from the memory.

## 12.11 Stop Mode

When the MCU is put into stop mode, if the FLASH is in read mode, it will be put into low-power standby. Exit from stop is possible with an external interrupt, such as  $\overline{IRQ}$ , keyboard interrupt, or reset.

The STOP instruction should not be executed while performing a program or erase operation on the FLASH. When the MCU is put into stop mode, the charge pump for the FLASH is disabled so that either a program or erase operation will not continue. If the memory is in either program mode (PGM = 1, HVEN = 1) or erase mode (ERASE = 1,

HEVEN = 1), then it will remain in that mode during stop. In this case, exit from stop must be done with a reset rather than an interrupt because if exiting stop with an interrupt, the memory will not be in read mode and the interrupt vector cannot be read from the memory.



## Section 13. On-Screen Display (OSD) FLASH Memory

### 13.1 Contents

|      |   |     |
|------|---|-----|
| 13.2 | Introduction . . . . .                        | 185 |
| 13.3 | Functional Description . . . . .              | 186 |
| 13.4 | OSD FLASH Control Register . . . . .          | 187 |
| 13.5 | Charge Pump Frequency Control . . . . .       | 189 |
| 13.6 | FLASH Erase Operation . . . . .               | 190 |
| 13.7 | FLASH Program/Margin Read Operation . . . . . | 192 |
| 13.8 | Block Protection . . . . .                    | 193 |
| 13.9 | OSD FLASH Block Protect Register . . . . .    | 193 |

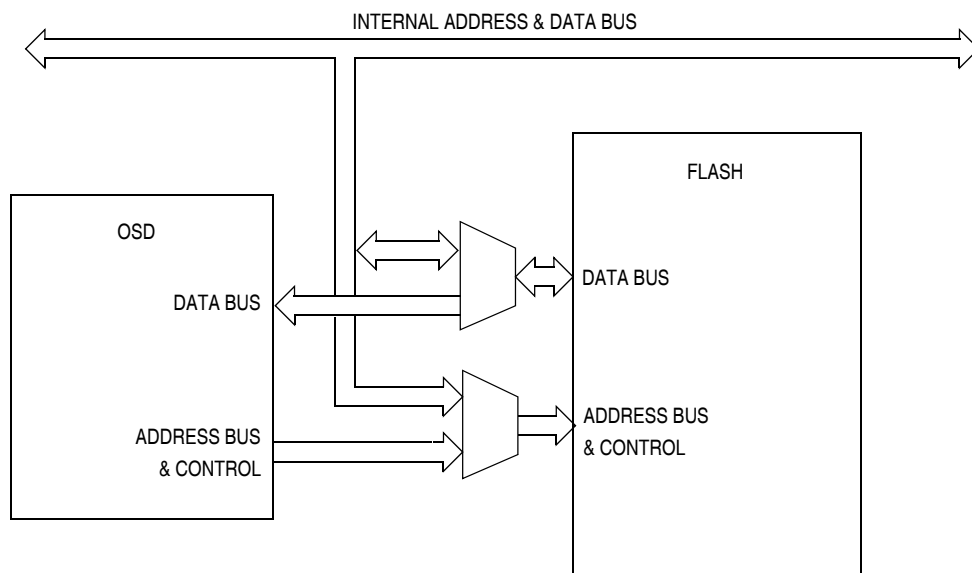
### 13.2 Introduction

This section describes the operation of the embedded OSD FLASH memory. This memory is used to store the pixel matrices of closed-caption and on-screen display (OSD) characters. It can be read, programmed, and erased from a single external supply. The program, erase, and read operations are enabled through the use of an internal charge pump.

## 13.3 Functional Description

The OSD FLASH memory is an array of 8,192 bytes used to store pixel matrices of closed-caption and OSD characters. Unlike the user FLASH, this memory is not connected directly to the internal bus. It is connected both to the OSD module and to the internal bus through a set of multiplexers as shown in **Figure 13-1**. During normal operation, the memory is always connected to the OSD module for read-only access.

The CPU can take control of the memory for programming, erasing, and reading just by selecting addresses in the memory map space reserved for the OSD FLASH (\$8000–\$9FFF).



**Figure 13-1. Connection of the OSD FLASH Memory**

**NOTE:** During normal operation the CPU should never try to access any address in the range \$8000–\$9FFF. If this happens, the OSD FLASH will be disconnected from the OSD module and the television image will be corrupted.

Since the array sizes are different, the OSD FLASH has a different page organization than the user FLASH. There are also eight pages of memory per row but only four bytes per page. The minimum erase block size is a single row, 32 bytes. Programming is performed on a per page

basis; four bytes at a time. The address range for CPU access is \$8000–\$9FFF. The same erasing and programming procedures outlined for the user FLASH in [12.3 Functional Description](#), [12.6 FLASH Erase Operation](#), and [12.7 FLASH Program/Margin Read Operation](#) must be applied to the OSD FLASH.

Although the OSD FLASH is capable of storing 8,192 bytes, the actual memory space required by the OSD module is only 6,528 bytes, occupying the range \$8000–\$9980.

**NOTE:** *This FLASH does not have security enabled. Nevertheless, it is still necessary to bypass security in order to write onto it. This is true because the block protect register (FL2BPR) is on the user FLASH, which is protected by security.*

## 13.4 OSD FLASH Control Register

The OSD FLASH control register controls FLASH program, erase, and verify operations.

Address: \$FE09

|        | Bit 7 | 6     | 5    | 4    | 3    | 2      | 1     | Bit 0 |
|--------|-------|-------|------|------|------|--------|-------|-------|
| Read:  | FDIV1 | FDIV0 | BLK1 | BLK0 | HVEN | MARGIN | ERASE | PGM   |
| Write: |       |       |      |      |      |        |       |       |
| Reset: | 0     | 0     | 0    | 0    | 0    | 0      | 0     | 0     |

**Figure 13-2. FLASH Control Register (FL2CR)**

**NOTE:** *This register is not connected directly to the internal bus. It is connected to the same bus as the OSD FLASH and thus can only be accessed if the OSD module is disabled (OSDEN = 0 in the OSD enable control register).*

**FDIV1** — Frequency Divide Control Bit

This read/write bit together with FDIV0 selects the factor by which the charge pump clock is divided from the system clock. See [13.5 Charge Pump Frequency Control](#).

### FDIV0 — Frequency Divide Control Bit

This read/write bit together with FDIV1 selects the factor by which the charge pump clock is divided from the system clock. See [13.5 Charge Pump Frequency Control](#).

### BLK1 — Block Erase Control Bit

This read/write bit together with BLK0 allows erasing of blocks of varying size. See [13.6 FLASH Erase Operation](#) for a description of available block sizes.

### BLK0 — Block Erase Control Bit

This read/write bit together with BLK1 allows erasing of blocks of varying size. See [13.6 FLASH Erase Operation](#) for a description of available block sizes.

### HVEN — High-Voltage Enable Bit

This read/write bit enables high voltage from the charge pump to the memory for either program or erase operation. It can only be set if either PGM or ERASE is high and the sequence for erase or program/margin read is followed.

1 = High voltage enabled to array and charge pump on

0 = High voltage disabled to array and charge pump off

### MARGIN — Margin Read Control Bit

This read/write bit configures the memory for margin read operation. It cannot be set if the HVEN bit is high, and if it is high when HVEN is set, it will automatically return to 0.

1 = Margin read operation selected

0 = Margin read operation unselected

### ERASE — Erase Control Bit

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

1 = Erase operation selected

0 = Erase operation unselected

### PGM — Program Control Bit

This read/write bit configures the memory for program operation. It 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

## 13.5 Charge Pump Frequency Control

The internal charge pump is designed to operate at greatest efficiency with an internal frequency of about 2 MHz. [Table 13-1](#) shows how the FDIV bits are used to select a charge pump frequency and the recommended bus frequency ranges for each configuration. Program and erase operations cannot be performed if the pump clock frequency is below 2 MHz.

**Table 13-1. Charge Pump Clock Frequency**

| FDIV1 | FDIV0 | Pump Clock Frequency | Bus Frequency |
|-------|-------|----------------------|---------------|
| 0     | 0     | Bus frequency ÷ 1    | 2 MHz ± 10%   |
| 0     | 1     | Bus frequency ÷ 2    | 4 MHz ± 10%   |
| 1     | 0     | Bus frequency ÷ 2    | 4 MHz ± 10%   |
| 1     | 1     | Bus frequency ÷ 4    | 8 MHz ± 10%   |

**NOTE:** *Since only one charge pump circuit services both FLASH arrays, the actual FDIV bits received by the charge pump is the logical OR of the individual FDIV bits of registers FL1CR and FL2CR.*

## 13.6 FLASH Erase Operation

Use the following procedure to erase a block of FLASH memory (to read as logic 0). Values for the time parameters are specified in

[25.11 Memory Characteristics](#).

1. Set the ERASE bit, the BLK0, BLK1, FDIV0, and FDIV1 bits in FL2CR. See [Table 13-1](#) for FDIV settings. See [Table 13-2](#) for block sizes.
2. Ensure that the block to be erased is not protected by the settings in the FL2BPR register. Read the FL2BPR register (\$FF81) which is physically located on the other FLASH. If test voltage,  $V_{TST}$ , is applied to the  $\overline{IRQ}$  pin, block protection is bypassed. See [13.9 OSD FLASH Block Protect Register](#).
3. Write to any FLASH address with any data within the block address range desired. If address is in a protected area, the ERASE bit will be cleared and the following steps of the erase procedure are blocked.
4. Set the HVEN bit.
5. Wait for a time,  $t_{ERASE}$ .
6. Clear the HVEN bit.
7. Wait for a time,  $t_{Kill}$ , for the high voltages to dissipate.
8. Clear the ERASE bit.
9. After a time,  $t_{HVD}$ , the memory can be accessed in read mode again.

**NOTE:** *While these operations must be performed in the order shown, other unrelated operations may occur between the steps. Do not exceed  $t_{Erase}$  maximum.*

[Table 13-2](#) shows the various block sizes which can be erased in one erase operation.

**Table 13-2. Erase Block Sizes**

| Write to Address Bits            | Address Value                           | BLK1 | BLK0 | Desired Erase Address Array  | Array Size                     |
|----------------------------------|---|------|------|--|--------------------------------|
| Any FLASH ADDR                   | Any                                     | 0    | 0    | \$8000–\$9FFF  | Full Array: 8,192 Bytes        |
| A <sub>15</sub> :A <sub>12</sub> | 1001                                    | 0    | 1    | \$9000–\$9FFF  | “Upper 1/2” Array: 4,096 Bytes |
| A <sub>15</sub> :A <sub>12</sub> | 1000                                    | 0    | 1    | \$8000–\$8FFF  | “Lower 1/2” Array: 4,096 Bytes |
| A <sub>15</sub> :A <sub>8</sub>  | {100, A <sub>12</sub> :A <sub>8</sub> } | 1    | 0    | {100, A <sub>12</sub> :A <sub>8</sub> , 00000000} to {100, A <sub>12</sub> :A <sub>8</sub> , 11111111} | Eight Rows: 256 Bytes          |
| A <sub>15</sub> :A <sub>5</sub>  | {100, A <sub>12</sub> :A <sub>5</sub> } | 1    | 1    | {100, A <sub>12</sub> :A <sub>5</sub> , 00000} to {100, A <sub>12</sub> :A <sub>5</sub> , 11111}       | Single Row: 32 Bytes           |

In step 3 of the erase operation, the desired erase addresses are latched and used to determine the location of the block to be erased. For the full array (BLK1 = 0, BLK0 = 0), the only requirement is that the FLASH memory be selected. Writing to any address in the range \$8000 to \$9FFF will enable the full array erase.

In the “upper 1/2 array” case (BLK1 = 0, BLK0 = 1), the state of A<sub>15</sub>:A<sub>12</sub> = 1001 determines that the range from \$9000 to \$9FFF is erased. For example, writing to address \$9123 will erase the range \$9000 to \$9FFF.

In the “lower 1/2 array” case (BLK1 = 0, BLK0 = 1) the state of A<sub>15</sub>:A<sub>12</sub> = 1000 determines that the range from \$8000 to \$8FFF is erased. For example, writing to address \$8623 will erase the range \$8000 to \$8FFF.

In the “eight row” erase operation (BLK1 = 1, BLK0 = 0), 256-byte blocks are erased as determined by upper addresses A<sub>15</sub>:A<sub>8</sub>. For example, writing to address \$8B10 will erase the range \$8B00 to \$8BFF.

In the “single row” erase operation (BLK1 = 1, BLK0 = 1), 32-byte blocks are erased as determined by upper addresses A<sub>15</sub>:A<sub>5</sub>. For example, writing to address \$9C60 will erase the range \$9C60 to \$9C7F.

### 13.7 FLASH Program/Margin Read Operation

Programming of the OSD FLASH memory is done using the same procedures outlined in [12.7 FLASH Program/Margin Read Operation](#). The main difference here is that a page consists of only four consecutive bytes instead of eight. A 4-byte page starts at the address {A15:A3,000} or {A15:A3,100}. The smart programming algorithm applied to this FLASH consists of the following steps:

1. Set the PGM bit in FL2CR. This configures the memory for program operation and enables the latching of address and data for programming.
2. Ensure that the block to be programmed is not protected by the settings in the FL2BPR register. Read the FL2BPR register, which is physically located on the other FLASH. If test voltage,  $V_{TST}$ , is applied to the  $\overline{IRQ}$  pin, block protection is bypassed. See [13.9 OSD FLASH Block Protect Register](#).
3. Write data to the four bytes of the page being programmed. This requires four separate write operations.
4. Set the HVEN bit.
5. Wait for a time,  $t_{STEP}$ .
6. Clear the HVEN bit.
7. Wait for a time,  $t_{HVTV}$ .
8. Set the MARGIN bit.
9. Wait for a time,  $t_{VTP}$ .
10. Clear the PGM bit.
11. Wait for a time,  $t_{HVD}$ .
12. Read the four data locations written in step 3. This is a margin read. Each read operation is stretched by eight cycles.
13. Clear the MARGIN bit.

If margin read data is identical to write data, then programming is complete. If this verify step fails, repeat from step 2.



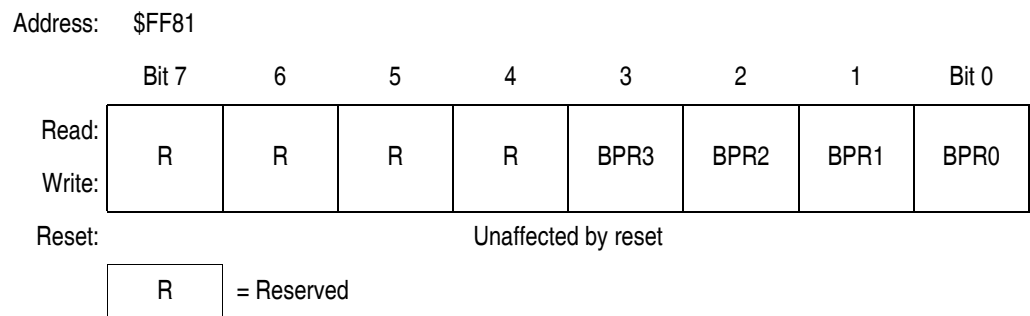
## 13.8 Block Protection

The OSD FLASH has the same block protection mechanism as the user FLASH. The difference is that the block protect register is not one of the memory positions of the OSD FLASH. Instead, it is physically located on the other FLASH memory, at location \$FF81.

When the block protect register is erased (all 0s), the entire memory is accessible for program and erase. When bits within the register are programmed, they lock blocks of memory address ranges as discussed in [13.9 OSD FLASH Block Protect Register](#). The presence of a voltage  $V_{TST}$  on the  $\overline{IRQ}$  pin will bypass the block protection mechanism so that all of the memory is open for program and erase operations. Be aware that a  $V_{TST}$  voltage on  $\overline{IRQ}$  when coming out of reset may force entry into monitor mode. See [16.4.1 Entering Monitor Mode](#).

## 13.9 OSD FLASH Block Protect Register

The block protect register is implemented as a byte within the user FLASH memory. Since the register is implemented in FLASH, upon reset the bits will come up in the state that was last defined by the user. Each bit, when programmed, protects a range of addresses in the OSD FLASH.



**Figure 13-3. OSD FLASH Block Protect Register (FL2BPR)**

### BPR3 — Block Protect Register Bit 3

This bit protects the memory contents in the address range \$9C00 to \$9FFF.

1 = Address range protected from erase or program

0 = Address range open to erase or program

### BPR2 — Block Protect Register Bit 2

This bit protects the memory contents in the address range \$9800 to \$9FFF.

1 = Address range protected from erase or program

0 = Address range open to erase or program

### BPR1 — Block Protect Register Bit 1

This bit protects the memory contents in the address range \$9000 to \$9FFF.

1 = Address range protected from erase or program

0 = Address range open to erase or program

### BPR0 — Block Protect Register Bit 0

This bit protects the entire array (\$8000 to \$9FFF).

1 = Address range protected from erase or program

0 = Address range open to erase or program

By programming the block protect bits, a portion of the memory will be locked so that no further erase or program operations may be performed. Programming more than one bit at a time is redundant. If both bit 3 and bit 2 are set, for instance, the address range \$9800 through \$9FFF is locked. If all bits are erased, then all of the memory is available for erase and program. The presence of a voltage  $V_{TST}$  will bypass the block protection so that all of the memory, including the block protect register, is open for program and erase operations.

**NOTE:** *Since the block protect register is located physically in the FLASH array occupying the address range \$A000–\$FFFF, the user must ensure that the block protect bits in FL1BPR do not prevent the programming of FL2BPR.*

## Section 14. External Interrupt (IRQ)

### 14.1 Contents

|      |                                    |     |
|------|------------------------------------|-----|
| 14.2 | Introduction                       | 195 |
| 14.3 | Features                           | 195 |
| 14.4 | Functional Description             | 196 |
| 14.5 | $\overline{\text{IRQ}}$ Pin        | 198 |
| 14.6 | IRQ Module During Break Interrupts | 199 |
| 14.7 | IRQ Status and Control Register    | 199 |

### 14.2 Introduction

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

### 14.3 Features

Features of the IRQ module include:

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

### 14.4 Functional Description

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

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

- Vector fetch — A vector fetch automatically generates an interrupt acknowledge signal that clears the latch that caused the vector fetch.
- Software clear — Software can clear an interrupt latch by writing to the appropriate acknowledge bit in the interrupt status and control register (INTSCR). Writing a logic 1 to the ACK bit clears the IRQ latch.
- Reset — A reset automatically clears the interrupt latch.

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

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

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

- Vector fetch or software clear
- Return of the interrupt pin to logic 1

The vector fetch or software clear may occur before or after the interrupt pin returns to logic 1. As long as the pin is low, the interrupt request remains pending. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

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

**NOTE:** The interrupt mask (I) in the condition code register (CCR) masks all interrupt requests, including external interrupt requests.

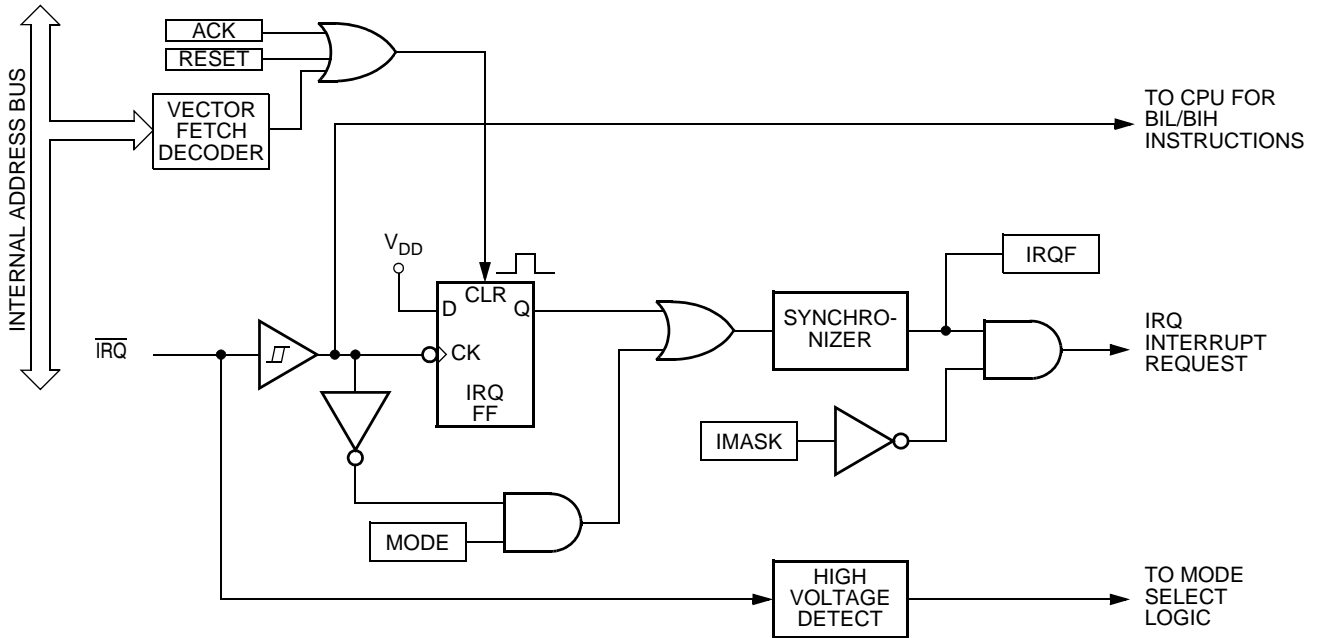


Figure 14-1. IRQ Module Block Diagram

| Addr.  | Register Name  | Bit 7  | 6 | 5 | 4 | 3 | 2    | 1   | Bit 0 |      |
|--------|--|--------|---|---|---|---|------|-----|-------|------|
| \$0007 | IRQ Status and Control Register (INTSCR)<br>See <a href="#">page 199</a> . | Read:  | 0 | 0 | 0 | 0 | IRQF | 0   | IMASK | MODE |
|        |  | Write: |   |   |   |   |      | ACK |       |      |
|        |  | Reset: | 0 | 0 | 0 | 0 | 0    | 0   | 0     | 0    |

= Unimplemented

Figure 14-2. IRQ I/O Register Summary

### 14.5 $\overline{\text{IRQ}}$ Pin

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

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

- Vector fetch or software clear — A vector fetch generates an interrupt acknowledge signal to clear the latch. Software may generate the interrupt acknowledge signal by writing a logic 1 to the ACK bit in the interrupt status and control register (INTSCR). The ACK bit is useful in applications that poll the  $\overline{\text{IRQ}}$  pin and require software to clear the IRQ latch. Writing to the ACK bit prior to leaving an interrupt service routine can also prevent spurious interrupts due to noise. Setting ACK does not affect subsequent transitions on the  $\overline{\text{IRQ}}$  pin. A falling edge occurs after writing another interrupt request to the ACK bit. If the IRQ mask bit, IMASK, is clear, the CPU loads the program counter with the vector address at locations \$FFFA and \$FFFB.
- Return of the  $\overline{\text{IRQ}}$  pin to logic 1 — As long as the  $\overline{\text{IRQ}}$  pin is at logic 0, IRQ remains active.

The vector fetch or software clear and the return of the  $\overline{\text{IRQ}}$  pin to logic 1 may occur in any order. The interrupt request remains pending as long as the  $\overline{\text{IRQ}}$  pin is at logic 0. A reset will clear the latch and the MODE control bit, thereby clearing the interrupt even if the pin stays low.

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

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

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

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

## 14.6 IRQ Module During Break Interrupts

The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear the latch during the break state. See [Section 6. Break Module \(BRK\)](#).

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

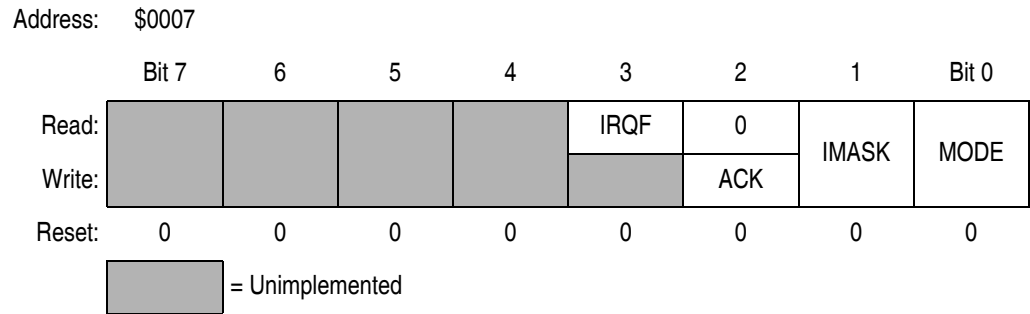
To protect CPU interrupt flags during the break state, write a logic 0 to the BCFE bit. With BCFE at logic 0 (its default state), writing to the ACK bit in the IRQ status and control register during the break state has no effect on the IRQ interrupt flags.

## 14.7 IRQ Status and Control Register

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

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

## External Interrupt (IRQ)



**Figure 14-3. IRQ Status and Control Register (INTSCR)**

### IRQF — IRQ Flag Bit

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

1 =  $\overline{\text{IRQ}}$  interrupt pending

0 =  $\overline{\text{IRQ}}$  interrupt not pending

### ACK — IRQ Interrupt Request Acknowledge Bit

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

### IMASK — IRQ Interrupt Mask Bit

Writing a logic 1 to this read/write bit disables IRQ interrupt requests. Reset clears IMASK.

1 = IRQ interrupt requests disabled

0 = IRQ interrupt requests enabled

### MODE — IRQ Edge/Level Select Bit

This read/write bit controls the triggering sensitivity of the  $\overline{\text{IRQ}}$  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



## Section 15. Low-Voltage Inhibit (LVI)

### 15.1 Contents

|        |                               |     |
|--------|-------------------------------|-----|
| 15.2   | Introduction                  | 201 |
| 15.3   | Features                      | 202 |
| 15.4   | Functional Description        | 202 |
| 15.4.1 | Polled LVI Operation          | 204 |
| 15.4.2 | Forced Reset Operation        | 204 |
| 15.4.3 | Voltage Hysteresis Protection | 204 |
| 15.4.4 | LVI Trip Selection            | 204 |
| 15.5   | LVI Status Register           | 205 |
| 15.6   | LVI Interrupts                | 205 |
| 15.7   | Low-Power Modes               | 206 |
| 15.7.1 | Wait Mode                     | 206 |
| 15.7.2 | Stop Mode                     | 206 |

### 15.2 Introduction

This section describes the low-voltage inhibit (LVI) module, which monitors the voltage on the  $V_{DD}$  pin and can force a reset when the  $V_{DD}$  voltage falls below the LVI trip falling voltage,  $V_{TRIPF}$ .

## 15.3 Features

Features of the LVI module include:

- Programmable LVI reset
- Selectable LVI trip voltage
- Programmable stop mode operation

## 15.4 Functional Description

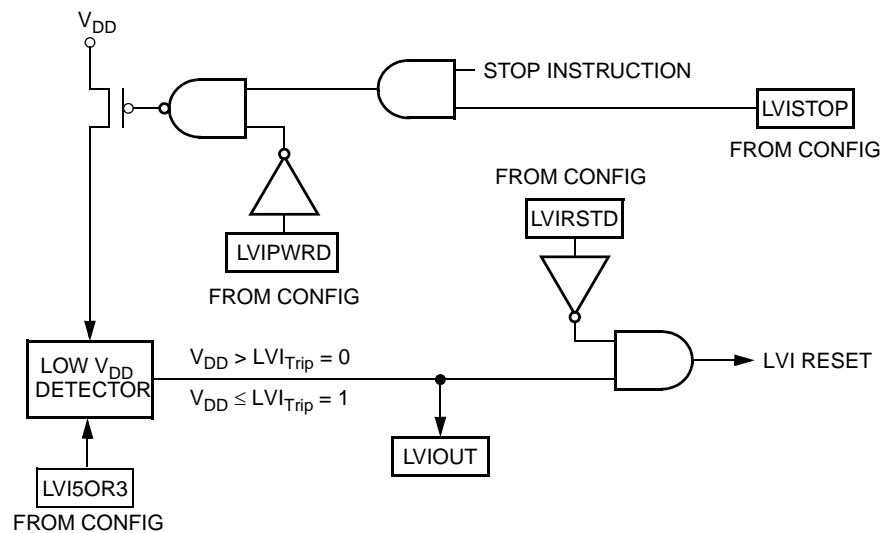
**Figure 15-1** shows the structure of the LVI module. The LVI is enabled out of reset. The LVI module contains a bandgap reference circuit and comparator. Clearing the LVI power disable bit, LVIPWRD, enables the LVI to monitor  $V_{DD}$  voltage. Clearing the LVI reset disable bit, LVIRSTD, enables the LVI module to generate a reset when  $V_{DD}$  falls below a voltage,  $V_{TRIPF}$ . Setting the LVI enable bit (LVISTOP) in stop mode bit enables the LVI to operate in stop mode. Setting the LVI 5-V or 3-V trip point bit, LVI5OR3, enables the trip point voltage,  $V_{TRIPF}$ , to be configured for 5-V operation. Clearing the LVI5OR3 bit enables the trip point voltage,  $V_{TRIPF}$ , to be configured for 3-V operation. The actual trip points are shown in **Section 25. Preliminary Electrical Specifications**.

**NOTE:** After a power-on reset (POR) the LVI's default mode of operation is 3 V. If a 5-V system is used, the user must set the LVI5OR3 bit to raise the trip point to 5-V operation. Note that this must be done after every power-on reset since the default will revert back to 3-V mode after each power-on reset. If the  $V_{DD}$  supply is below the 5-V mode trip voltage but above the 3-V mode trip voltage when POR is released, the part will operate because  $V_{TRIPF}$  defaults to 3-V mode after a POR. So, in a 5-V system care must be taken to ensure that  $V_{DD}$  is above the 5-V mode trip voltage after POR is released.

**NOTE:** If the user requires 5-V mode and sets the LVI5OR3 bit after a power-on reset while the  $V_{DD}$  supply is not above the  $V_{TRIPR}$  for 5-V mode, the MCU will immediately go into reset. The LVI in this case will hold the part in reset until either  $V_{DD}$  goes above the rising 5-V trip point,  $V_{TRIPR}$ , which will release reset or  $V_{DD}$  decreases to approximately 0 V which will re-trigger the power-on reset and reset the trip point to 3-V operation.

LVISTOP, LVIPWRD, LVI5OR3, and LVIRSTD are in the configuration register (CONFIG). See [9.3 Functional Description](#) for details of the LVI's configuration bits. Once an LVI reset occurs, the MCU remains in reset until  $V_{DD}$  rises above a voltage,  $V_{TRIPR}$ , which causes the MCU to exit reset. See [20.4.2.5 Low-Voltage Inhibit \(LVI\) Reset](#) for details of the interaction between the SIM and the LVI. The output of the comparator controls the state of the LVIOOUT 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 15-1. LVI Module Block Diagram**

| Addr.  | Register Name   | Bit 7  | 6       | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|---|--------|---------|---|---|---|---|---|-------|
| \$FE0F | LVI Status Register (LVISR)<br>See <a href="#">page 205</a> . | Read:  | LVIOOUT | 0 | 0 | 0 | 0 | 0 | 0     |
|        |   | Write: |         |   |   |   |   |   |       |
|        |   | Reset: | 0       | 0 | 0 | 0 | 0 | 0 | 0     |

= Unimplemented

**Figure 15-2. LVI I/O Register Summary**

### 15.4.1 Polled LVI Operation

In applications that can operate at  $V_{DD}$  levels below the  $V_{TRIPF}$  level, software can monitor  $V_{DD}$  by polling the LVIOOUT bit. In the configuration register, the LVIPWRD bit must be at logic 0 to enable the LVI module, and the LVIRSTD bit must be at logic 1 to disable LVI resets.

### 15.4.2 Forced Reset Operation

In applications that require  $V_{DD}$  to remain above the  $V_{TRIPF}$  level, enabling LVI reset allows the LVI module to reset the MCU when  $V_{DD}$  falls below the  $V_{TRIPF}$  level. In the configuration register, the LVIPWRD and LVIRSTD bits must be at logic 0 to enable the LVI module and to enable LVI resets.

### 15.4.3 Voltage Hysteresis Protection

Once the LVI has triggered by having  $V_{DD}$  fall below  $V_{TRIPF}$ , the LVI will maintain a reset condition until  $V_{DD}$  rises above the rising trip point voltage,  $V_{TRIPR}$ . This prevents a condition in which the MCU is continually entering and exiting reset if  $V_{DD}$  is approximately equal to  $V_{TRIPF}$ .  $V_{TRIPR}$  is greater than  $V_{TRIPF}$  by the hysteresis voltage,  $V_{HYS}$ .

### 15.4.4 LVI Trip Selection

The LVI5OR3 bit in the configuration register selects whether the LVI is configured for 5-V or 3-V protection.


**NOTE:** *Although the LVI can be configured for 3 V protection, the microcontroller itself may or may not have been designed to operate with 3 V supply voltage. See [Section 25. Preliminary Electrical Specifications](#) for minimum supply and LVI trip point voltages.)*

## 15.5 LVI Status Register

The LVI status register (LVISR) indicates if the  $V_{DD}$  voltage was detected below the  $V_{TRIPF}$  level.

Address: \$FE0F

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

 = Unimplemented

**Figure 15-3. LVI Status Register (LVISR)**

### LVIOUT — LVI Output Bit

This read-only flag becomes set when the  $V_{DD}$  voltage falls below the  $V_{TRIPF}$  trip voltage. (See [Table 15-1](#).) Reset clears the LVIOUT bit.

**Table 15-1. LVIOUT Bit Indication**

| $V_{DD}$                         | LVIOUT         |
|----------------------------------|----------------|
| $V_{DD} > V_{TRIPR}$             | 0              |
| $V_{DD} < V_{TRIPF}$             | 1              |
| $V_{TRIPF} < V_{DD} < V_{TRIPR}$ | Previous value |

## 15.6 LVI Interrupts

The LVI module does not generate interrupt requests.

### 15.7 Low-Power Modes

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

#### 15.7.1 Wait Mode

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

#### 15.7.2 Stop Mode

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

## Section 16. Monitor ROM (MON)

### 16.1 Contents

|        |                              |     |
|--------|------------------------------|-----|
| 16.2   | Introduction .....           | 207 |
| 16.3   | Features .....               | 208 |
| 16.4   | Functional Description ..... | 208 |
| 16.4.1 | Entering Monitor Mode .....  | 210 |
| 16.4.2 | Data Format .....            | 212 |
| 16.4.3 | Break Signal .....           | 213 |
| 16.4.4 | Baud Rate .....              | 213 |
| 16.4.5 | Commands .....               | 214 |
| 16.4.6 | Security .....               | 218 |

### 16.2 Introduction

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

### 16.3 Features

Features of the monitor ROM include:

- Normal user-mode pin functionality
- One pin dedicated to serial communication between monitor ROM and host computer
- Standard mark/space non-return-to-zero (NRZ) communication with host computer
- Execution of code in RAM or FLASH
- FLASH memory security feature

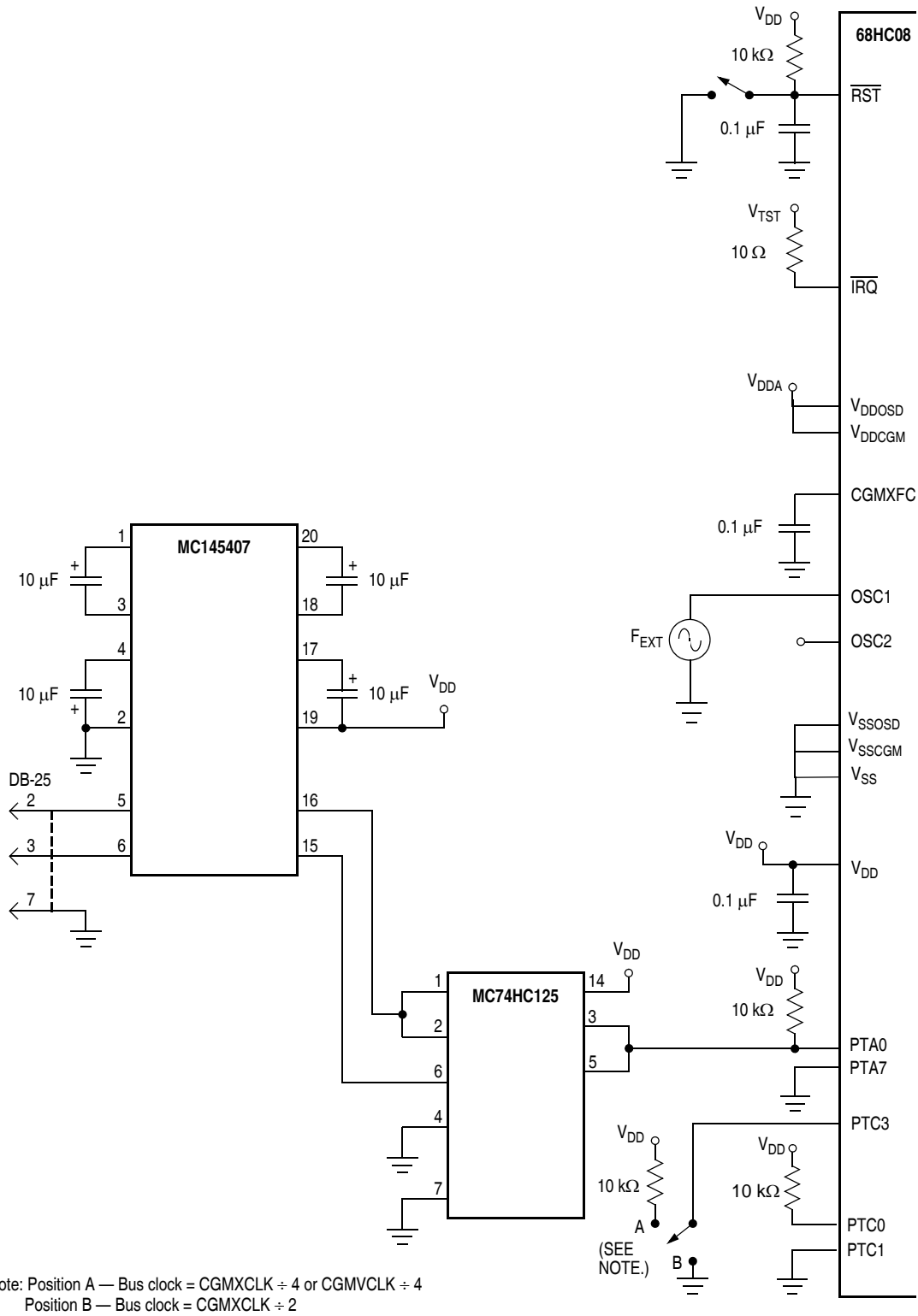
### 16.4 Functional Description

The monitor ROM receives and executes commands from a host computer. [Figure 16-1](#) shows a sample circuit used to enter monitor mode and communicate with a host computer via a standard RS-232 interface.

Simple monitor commands can access any memory address. In monitor mode, the MCU can execute code download into RAM by a host computer while most MCU pins retain normal operating mode functions. All communication between the host computer and the MCU is through the PTA0 pin. A level-shifting and multiplexing interface is required between PTA0 and the host computer. PTA0 is used in a wired-OR configuration and requires a pullup resistor.

The MC68HC908TV24 has 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 [16.4.6 Security](#)).





**Figure 16-1. Monitor Mode Circuit**

## 16.4.1 Entering Monitor Mode

**Table 16-1** shows the pin conditions for entering monitor mode.

**Table 16-1. Mode Selection**

| $\overline{\text{IRQ}}$ Pin | PTA7 Pin | PTA0 Pin | PTC3 Pin | PTC0 Pin | PTC1 Pin | Mode    | CGMOUT   | Bus Frequency             |
|-----------------------------|----------|----------|----------|----------|----------|---------|--|---------------------------|
| $V_{\text{TST}}^{(1)}$      | 0        | 1        | 1        | 1        | 0        | Monitor | $\frac{\text{CGMXCLK}}{2}$ or $\frac{\text{CGMVCLK}}{2}$ | $\frac{\text{CGMOUT}}{2}$ |
| $V_{\text{TST}}^{(1)}$      | 0        | 1        | 0        | 1        | 0        | Monitor | CGMXCLK  | $\frac{\text{CGMOUT}}{2}$ |

1. For  $V_{\text{TST}}$ , see [25.6 5.0-V DC Electrical Characteristics](#).

CGMOUT/2 is the internal bus clock frequency. If PTC3 is low upon monitor mode entry, CGMOUT is equal to the frequency of CGMXCLK, which is a buffered version of the clock on the OSC1 pin. The bus frequency in this case is a divide-by-two of the input clock. If PTC3 is high upon monitor mode entry, the bus frequency will be a divide-by-four of the input clock if the PLL is not engaged. The PLL can be engaged to multiply the bus frequency by programming the CGM. Refer to [Section 7. Clock Generator Module \(CGMC\)](#) for information on how to program the PLL. When the PLL is used, PTC3 must be logic 1 during monitor mode entry, and the bus frequency will be a divide-by-four of CGMVCLK, the output clock of the PLL.

**NOTE:** *Holding the PTC3 pin low when entering monitor mode causes a bypass of a divide-by-two stage in the oscillator. In this case, the CGMOUT frequency is equal to the CGMXCLK (external clock) frequency. The OSC1 signal must have a 50 percent duty cycle at maximum bus frequency.*

Enter monitor mode with the pin configuration shown in [Table 16-1](#) by pulling  $\overline{\text{RST}}$  low and then high. The rising edge of  $\overline{\text{RST}}$  latches monitor mode. Once monitor mode is latched, the values on the specified pins can change.

**NOTE:** *The PTA7 pin must remain at logic 0 for 24 bus cycles after the  $\overline{RST}$  pin goes high.*

Once out of reset, the MCU waits for the host to send eight security bytes (see [16.4.6 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. The break signal also serves as a timing reference to allow the host to determine the necessary baud rate

Monitor mode uses alternate vectors for reset, SWI, and break interrupt to those used in user mode. The alternate vectors are in the \$FE page instead of the \$FF page and allow code execution from the internal monitor firmware instead of user code.

When the host computer has completed downloading code into the MCU RAM, this code can be executed by driving PTA0 low while asserting  $\overline{RST}$  low and then high. The internal monitor ROM firmware will interpret the low on PTA0 as an indication to jump RAM, and execution control will then continue from RAM. The location jumped to is always the second byte of RAM (for example, the first RAM byte address + 1). Execution of an SWI from the downloaded code will return program control to the internal monitor ROM firmware. Alternatively, the host can send a RUN command, which executes an RTI, and this can be used to send control to the address on the stack pointer.

The COP module is disabled in monitor mode as long as  $V_{TST}$  is applied to either the  $\overline{IRQ}$  pin or the RESET pin.

**Table 16-2** is a summary of the differences between user mode and monitor mode.

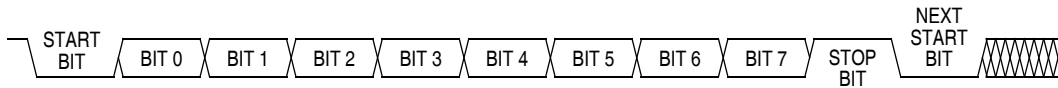
**Table 16-2. Mode Differences**

| 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 <sup>(1)</sup> | \$FEFE            | \$FEFF           | \$FEFC            | \$FEFD           | \$FEFC          | \$FEFD         |

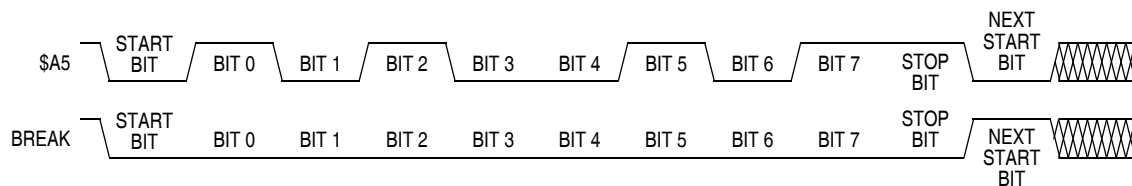
1. If the high voltage ( $V_{TST}$ ) is removed from the  $\overline{IRQ}$  pin or the  $\overline{RST}$  pin while in monitor mode, the SIM asserts its COP enable output. The COP is mask option enabled or disabled by the COPD bit in the configuration register.

## 16.4.2 Data Format

Communication with the monitor ROM is in standard non-return-to-zero (NRZ) mark/space data format. Transmit and receive baud rates must be identical. See **Figure 16-2** and **Figure 16-3**.



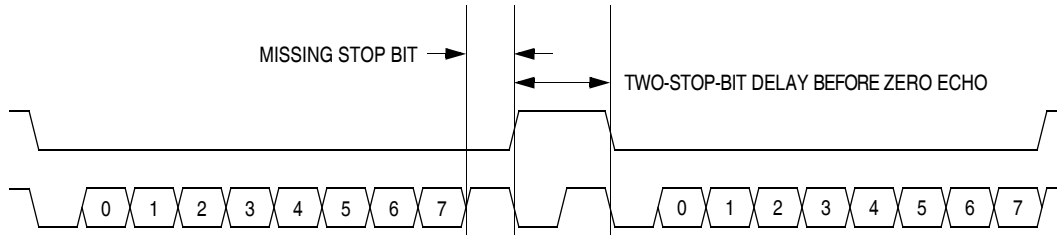
**Figure 16-2. Monitor Data Format**



**Figure 16-3. Sample Monitor Waveforms**

### 16.4.3 Break Signal

A start bit followed by nine low bits is a break signal. See [Figure 16-4](#). 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 16-4. Break Transaction**

### 16.4.4 Baud Rate

The bus clock frequency for the MCU in monitor mode is determined by the external clock frequency, the value on PTC3 during monitor mode entry, and whether or not the PLL is engaged. The internal monitor firmware performs a division by 256 (for sampling data). Therefore, the bus frequency divided by 256 is the baud rate of the monitor mode data transfer.

For example, with a 4.9152-MHz external clock and the 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.

The internal PLL can be engaged to increase the baud rate of instruction transfer between the host and the MCU and to increase the speed of program execution. Refer to [Section 7. Clock Generator Module \(CGMC\)](#) for information on how to program the PLL. If use of the PLL is desired, the monitor mode must be entered with PTC high. See [16.4.1 Entering Monitor Mode](#). Initially, the bus frequency is a divide-by-four of the input clock. After the PLL is programmed and enabled onto the bus, communication between the host and MCU must be re-established

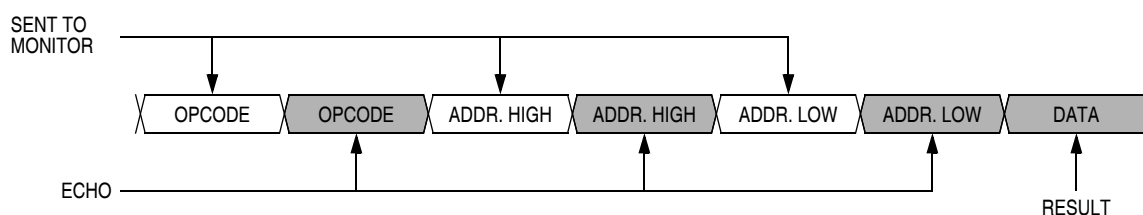
at the new baud rate. One way of accomplishing this would be for the host to download a program into the MCU RAM that would program the PLL and send a new baud rate flag to the host just prior to engaging the PLL onto the bus. Upon completion of execution, an SWI would return program control to the monitor firmware.

### 16.4.5 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

As the host computer sends commands through PTA0, the monitor ROM firmware immediately echoes each received byte back to the PTA0 pin for error checking, as shown in the example command in [Figure 16-5](#).



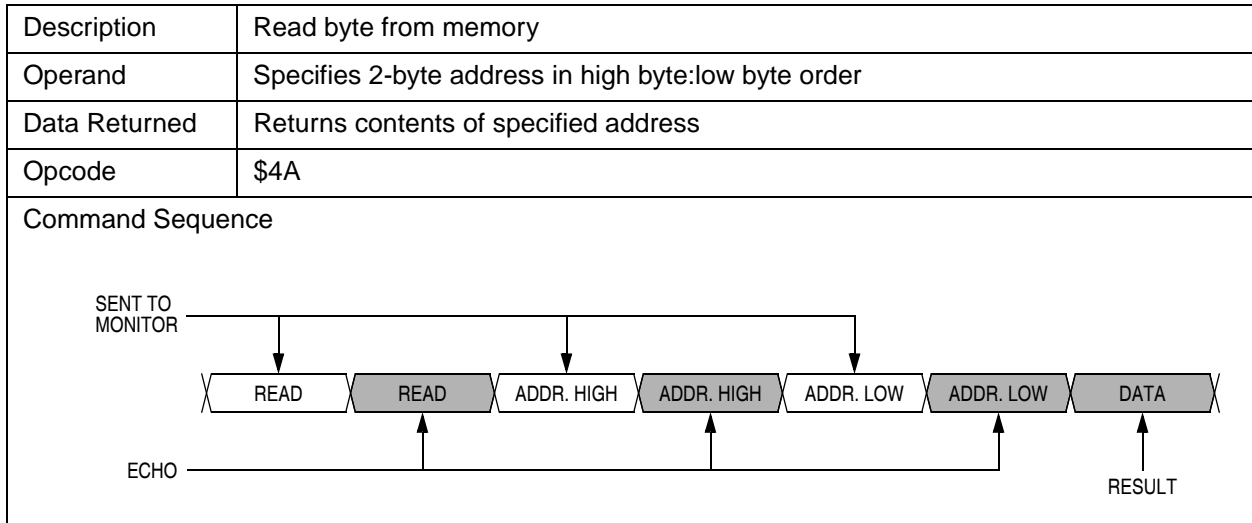
**Figure 16-5. Read Transaction**

The resultant data of a read type of command appears after the echo of the last byte of the command.

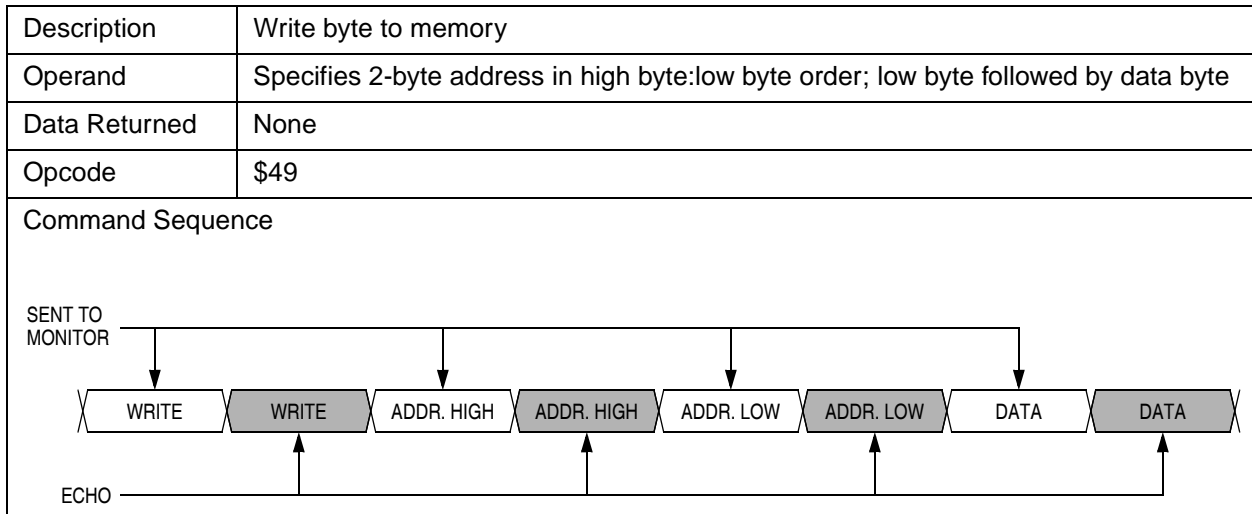
A brief description of each monitor mode command follows.

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

**Figure 16-6. READ (Read Memory) Command**



**Figure 16-7. WRITE (Write Memory) Command**



### Figure 16-8. 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 |  |

The diagram illustrates the IREAD command sequence. It consists of four blocks: IREAD, IREAD, DATA, and DATA. The first IREAD block is labeled 'SENT TO MONITOR' with an arrow pointing to it. The second IREAD block is labeled 'ECHO' with an arrow pointing to it. The first DATA block is labeled 'RESULT' with an arrow pointing to it. The second DATA block is also labeled 'RESULT' with an arrow pointing to it.

### Figure 16-9. IWRITE (Indexed Write) Command

|                  |                                    |
|------------------|------------------------------------|
| Description      | Write to last address accessed + 1 |
| Operand          | Specifies single data byte         |
| Data Returned    | None                               |
| Opcode           | \$19                               |
| Command Sequence |                                    |

The diagram illustrates the IWRITE command sequence. It consists of four blocks: IWRITE, IWRITE, DATA, and DATA. The first IWRITE block is labeled 'SENT TO MONITOR' with an arrow pointing to it. The second IWRITE block is labeled 'ECHO' with an arrow pointing to it. The first DATA block is labeled 'SENT TO MONITOR' with an arrow pointing to it. The second DATA block is labeled 'ECHO' with an arrow pointing to it.



**Figure 16-10. 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 |   |

**Figure 16-11. RUN (Run User Program) Command**

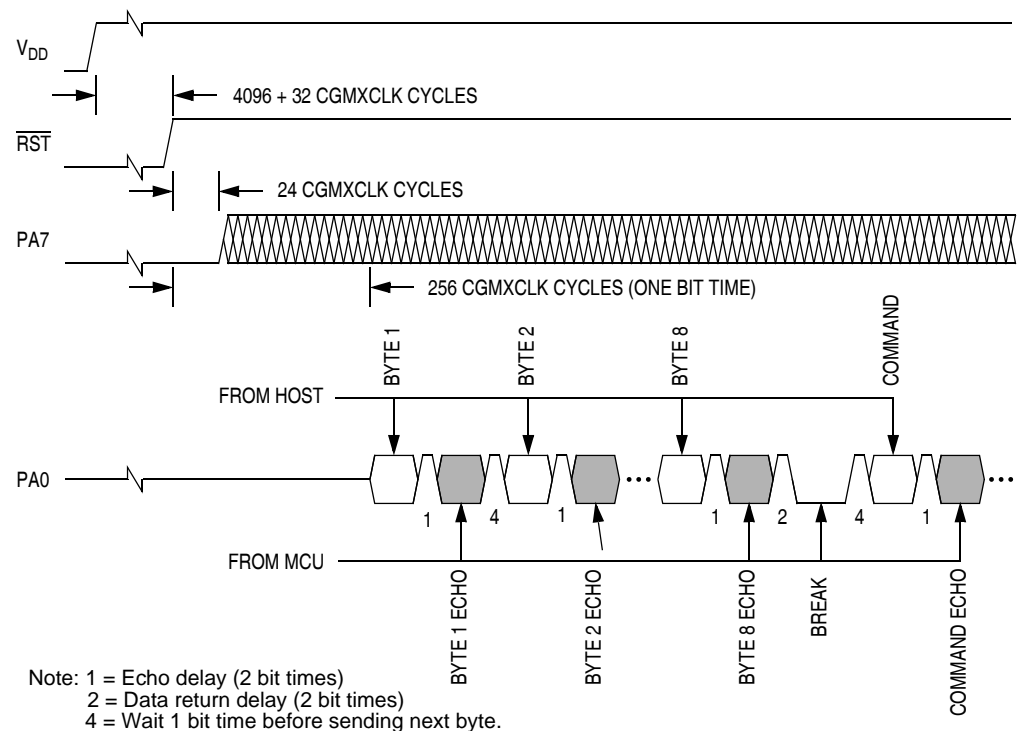
|                  |                          |
|------------------|--------------------------|
| Description      | Executes RTI instruction |
| Operand          | None                     |
| Data Returned    | None                     |
| Opcode           | \$28                     |
| Command Sequence |                          |

## 16.4.6 Security

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

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

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



**Figure 16-12. 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.*

# Monitor ROM (MON)

## Section 17. On-Screen Display Module (OSD)

### 17.1 Contents

|        |                                 |     |
|--------|---------------------------------|-----|
| 17.2   | Introduction                    | 222 |
| 17.3   | Features                        | 222 |
| 17.4   | Overview                        | 224 |
| 17.4.1 | Single-Row Architecture         | 225 |
| 17.4.2 | Display Timing                  | 225 |
| 17.4.3 | Registers and Pixel Memory      | 226 |
| 17.4.4 | OSD Output Logic                | 227 |
| 17.5   | Display Characteristics         | 228 |
| 17.5.1 | Closed-Caption Mode             | 228 |
| 17.5.2 | On-Screen Display Mode          | 230 |
| 17.6   | FLASH Programming Guidelines    | 234 |
| 17.7   | Programming Guidelines          | 239 |
| 17.7.1 | Setup                           | 239 |
| 17.7.2 | Interrupt Servicing             | 240 |
| 17.7.3 | Software Controlled Features    | 242 |
| 17.8   | Input and Output                | 242 |
| 17.8.1 | Power Supply Pins               | 243 |
| 17.8.2 | Input Pins                      | 243 |
| 17.8.3 | PLL Filter Pins                 | 243 |
| 17.8.4 | Output Pins                     | 244 |
| 17.9   | Registers                       | 244 |
| 17.9.1 | OSD Character Registers         | 245 |
| 17.9.2 | OSD Vertical Delay Register     | 251 |
| 17.9.3 | OSD Horizontal Delay Register   | 252 |
| 17.9.4 | OSD Foreground Control Register | 252 |
| 17.9.5 | OSD Background Control Register | 254 |
| 17.9.6 | OSD Border Control Register     | 255 |
| 17.9.7 | OSD Enable Control Register     | 257 |

|         |                             |     |
|---------|-----------------------------|-----|
| 17.9.8  | OSD Event Line Register     | 259 |
| 17.9.9  | OSD Event Count Register    | 260 |
| 17.9.10 | OSD Output Control Register | 260 |
| 17.9.11 | OSD Status Register         | 262 |
| 17.9.12 | OSD Matrix Start Register   | 263 |
| 17.9.13 | OSD Matrix End Register     | 265 |
| 17.10   | Low Power Modes             | 265 |
| 17.10.1 | Wait Mode                   | 265 |
| 17.10.2 | Stop Mode                   | 266 |
| 17.11   | Interrupts and Resets       | 266 |

## 17.2 Introduction

The on-screen display (OSD) module converts programmed character addresses and control information into digital color, intensity, and blanking outputs to display user-defined characters on a television screen for on-screen programming and closed-caption (CC) applications.

## 17.3 Features

The OSD module provides these features:

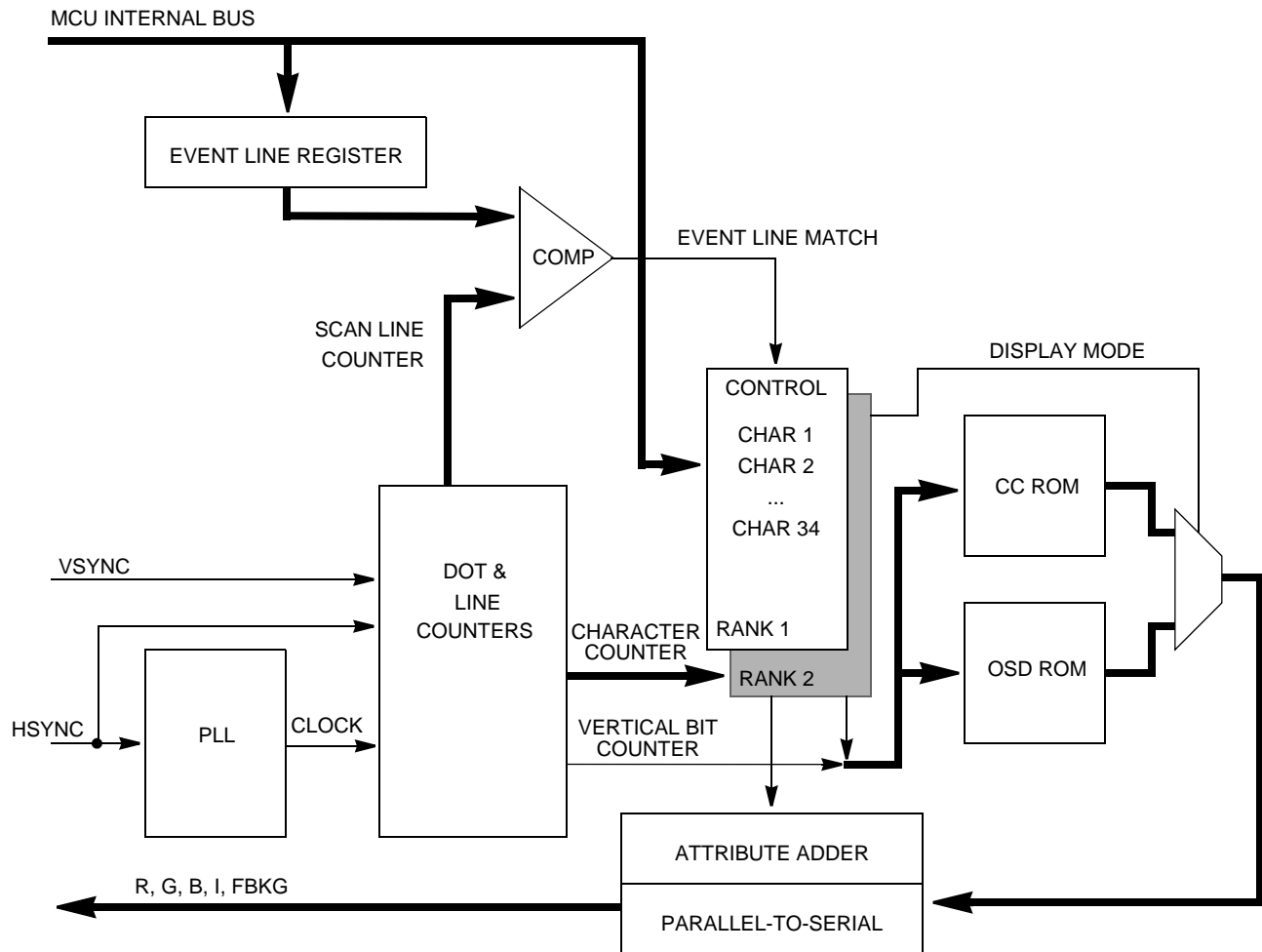
- 192 user-defined (programmable in FLASH memory)  
9 x 13 characters for use in CC mode
- 126 user-defined (programmable in FLASH memory)  
12 x 18 characters for use in OSD mode
- 34 columns by 16 rows in CC mode (standard size)
- 24 columns by 12 rows in OSD mode (standard size)

- Software selectable character attributes:
  - 16 foreground colors (8 colors and 2 intensity levels)
  - 16 background colors
  - 16 border colors (area outside foreground and background)
  - Rounding
  - Black outline
  - 3D shadow (OSD mode only)
  - Underline and italics (in CC mode only)
- In OSD mode, two foreground colors are possible in one character with the use of backspace overlaying of two characters
- Closed-caption characters are fixed size, but OSD characters may have three different sizes, selectable on a row-by-row basis: (1W x 1V), (1.5W x 2V), and (2W x 2V)
- Programmable horizontal and vertical positioning
- Software controlled features:
  - Soft scrolling
  - Blinking
- Programmable polarity of R, G, B, I, and FBKG
- Zero inter-column and inter-row spacing, so composed characters can be formed by abutment

# On-Screen Display Module (OSD)

## 17.4 Overview

The OSD uses single character row architecture instead of a full-screen display RAM in order to minimize die area and address space requirements. **Figure 17-1** shows a block diagram of the OSD.



**Figure 17-1. OSD Block Diagram**

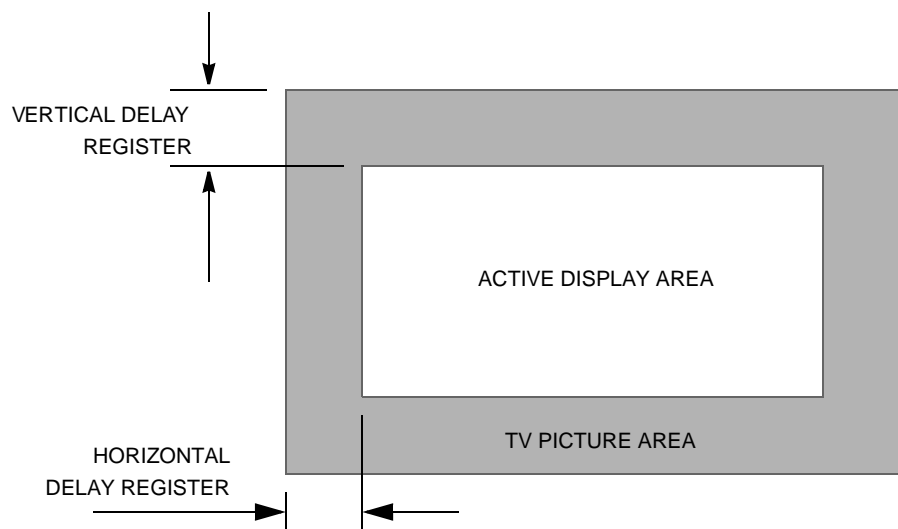


### 17.4.1 Single-Row Architecture

The basic concept of single-row architecture is to use one row of double-ranked registers to do the communication between the CPU and the display. The CPU writes the data of the next row to be displayed into the first rank of registers. It also writes the target line in which this row is to be displayed on the event line register. Meanwhile, a counter (accessible through the event register) keeps track of the current line being displayed. This line counter is continuously compared to the contents of the event line register. When a match occurs, all the external rank is loaded into the internal rank and an interrupt request (event line match interrupt) is optionally generated to notify the CPU of the availability of the external rank for the next row of characters to be displayed. For the display of one row only, the CPU can leave the registers unchanged all the way. For two adjacent rows, the CPU must update the buffer within a period of a one-row display. In CC mode, this period is  $63.5 \times 13 = 825 \mu\text{s}$ .

### 17.4.2 Display Timing

The OSD horizontal timing is based on an independent on-chip oscillator that is phase locked to the starting edge of the internal horizontal sync pulse. The OSD vertical timing is synchronized to the starting edge of the internal vertical sync pulse, so that stable display is possible with either 525 or 625 line systems. An interrupt is optionally generated at every starting edge of the vertical sync pulse. This interrupt can be used to determine which television system is being received: 60 Hz (525 lines) or 50 Hz (625 lines). Knowing which TV system is being received, the software must program the vertical delay register and the horizontal delay register to best fit the active display area on the screen, as shown in [Figure 17-2](#). The active display area occupies 16 rows by 34 columns in CC mode and 12 rows by 24 columns in OSD mode.



**Figure 17-2. Active Display Area**

### 17.4.3 Registers and Pixel Memory

The majority of the OSD registers are double-ranked because the OSD must read the values for the current row while the CPU writes the values for the next row. The external rank consists of a set of registers visible to the CPU, containing the character codes and control information relevant to a single row on the display screen. These registers appear in the register space and their bits are readable as well as writable. The internal rank is parallel shifted from the external rank whenever the OSD is ready to display the next row. The internal rank feeds the character generation logic of the OSD.

Character codes are read sequentially from the internal rank of the character registers during each scan line. Each code, along with the scan line number within the character row, is used as an address to fetch the pixel pattern from the CC ROM (in CC mode) or from the OSD ROM (in OSD mode). The CC ROM contains 192 user-defined (programmable in FLASH memory) 9 x 13 pixel matrix characters used exclusively for closed-caption display. The OSD ROM contains another 128 user-definable (also programmable in FLASH memory) 12 x 18 pixel matrix characters used for OSD display. Character codes may alternately be interpreted as control codes that alter display attributes in the middle of a row.

#### 17.4.4 OSD Output Logic

Programmable character attributes include color (foreground and background), rounding, black outline, 3D shadow (OSD mode only), blinking, underlining and italics (both in CC mode only). Rounding and outline dots are half the width and length of the basic ROM character pixel. In the vertical dimension, each pixel of the character matrix is formed by a pair of odd and even lines of different interlace fields. To produce the half-pixel rounding dot, the spatial displacement of odd and even fields of the interlaced video is used, so that just one of the fields displays the rounding or outline dot. The odd/even field indicator is extracted by the closed-caption data slicer, which must be enabled to provide field information to the OSD.

These attributes have a default set on a row basis, and many of them can be modified inside the row by control characters. The CC mode and the OSD mode have different character control codes and the display behavior is different for each mode. In CC mode, all mid-row control characters are displayed as background color spaces, whereas in OSD mode, up to two control characters can be placed together without being displayed. In OSD mode it is also possible to backspace a character and display it on top of the preceding one, allowing the use of more than one foreground color in one character.

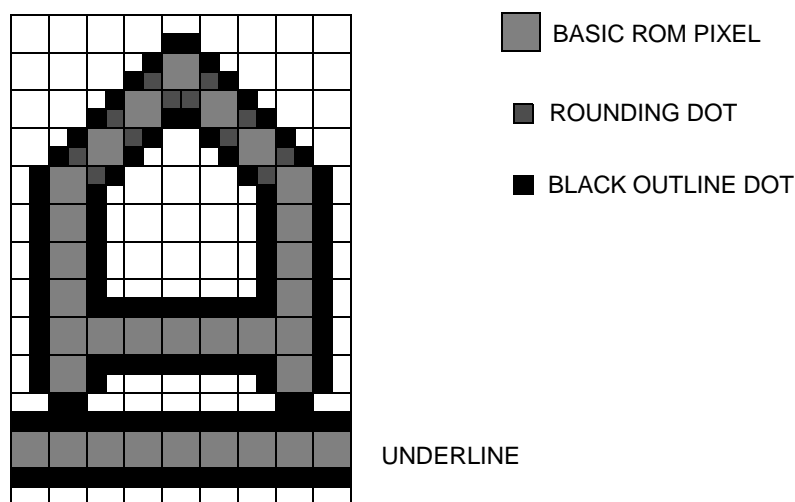
The pixel pattern data from the pixel ROM passes through circuitry which adds the selected attributes of rounding, black outline, shadow, italics and underline. The resulting signal passes through a color encoder and is converted into R, G, B, I, and FBKG signals. In OSD mode, if a backspace control code appears between two characters, the second one is read at the same time as the first, and the two of them go in parallel to the color encoder to form a composite character. In this case it is possible to apply rounding, black outline and shadow only to the first character. The second one is added with no attributes applied.

## 17.5 Display Characteristics

Two display modes affect the appearance and attributes of the characters on the screen: the CC mode and the OSD mode. The display mode can be programmed for each row by writing into bit 7 of the matrix start register.

### 17.5.1 Closed-Caption Mode

In closed-caption mode, the character is defined on a 9 x 13 pixel matrix, as shown in [Figure 17-3](#). Attributes that can be added in this mode are rounding, black outline, underline and italics. Rounding dots, which are generated to smooth diagonals, are half the size of a normal ROM pixel and are displayed in the foreground color. Black-outline dots, also half the size of a normal ROM pixel, are displayed in black to outline a character and improve readability. The underline attribute will generate a line of pixels which are of the same color and size as the ROM pixels. Underline will affect all characters with the exception of the border space (\$00), background space (\$01), and control codes. The italics attribute does not generate any additional pixels, but will slant the display of the character.



**Figure 17-3. CC-ROM Pixel Matrix**

The background is the region of the character matrix where ROM pixels are zero and there are no rounding, black outline or underline dots. Background may be a solid color or transparent, in which case the video will show through (regardless of border color selection).

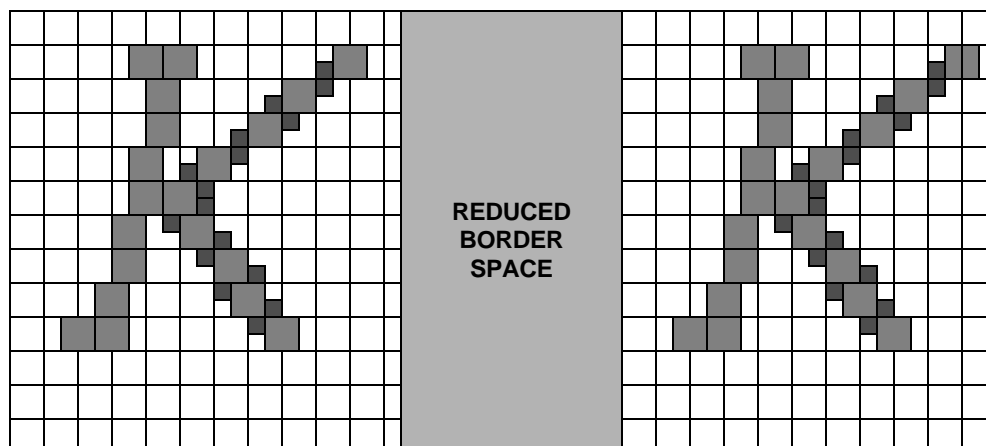
Border is displayed wherever there is no foreground or background inside the active display area. Border may also be a solid color or transparent. Border can be specified for a character within a row by using the border space character. When background or border are transparent, the video intensity can be specified to be half-tone to improve readability.

The horizontal delay of the active display area from each HSYNC pulse is given by a minimum fixed delay of 14.1  $\mu\text{s}$  plus an additional delay specified on the horizontal delay register (32 increments of 0.14  $\mu\text{s}$ ). This delay does not depend on the character size programmed for each row. The border space character can be used to provide further indentation in a row-by-row basis.

Vertical positioning of the active display area is specified in the vertical delay register. This delay is measured in lines from the starting edge of the vertical sync pulse. Vertical positioning inside the active display area is indicated by the contents of the event line register.

Scrolling can be accomplished in software by modifying the vertical positioning of a row, in conjunction with specifying the subset of horizontal lines of the row to display. The matrix start register and the matrix end register indicate, respectively, the first and last lines of a row to be displayed. All lines before and after the lines displayed are filled in with border.

Boundaries between border and character display are always vertical. If italics are not enabled, the width of a border space character is the same as the width of a ROM character. If italics are enabled, the boundaries of the border space character are not slanted, but the width occupied by this space is smaller than the width of a ROM character by five dots (see [Figure 17-4](#)).



**Figure 17-4. Boundary Conditions with Italics**

## 17.5.2 On-Screen Display Mode

In on-screen display mode, the character is defined on a 12 x 18 pixel matrix, as shown in [Figure 17-5](#). Attributes that can be added in this mode are rounding, black outline, blinking, shadow, not-pressed three-dimensional (3D) shadow and pressed 3D-shadow. Rounding, black outline, and blinking are identical as in CC mode. Shadow is generated to simulate the effect of a light source situated in the northwest corner. The not-pressed 3D-shadow simulates the effect of a 3D object by putting a white outline on the side near the light source, and a black outline on the opposite side. The pressed 3D-shadow does the contrary, putting a black outline on the side near the light source, and a white outline on the opposite side (see [Figure 17-6](#)).

Background, border, horizontal delay, vertical delay, and scrolling behave the same as in CC mode and the same registers control them. The only exception is the appearance of the control characters. In CC mode they appear as background spaces. In OSD mode it is possible to use up to two consecutive control characters without any of them being displayed. This is shown in [Figure 17-7](#). As a consequence, a single control character can not be used to add a space. One must use a border space (\$00) or a background space (\$01). If three consecutive control characters are used, a background space will be added.

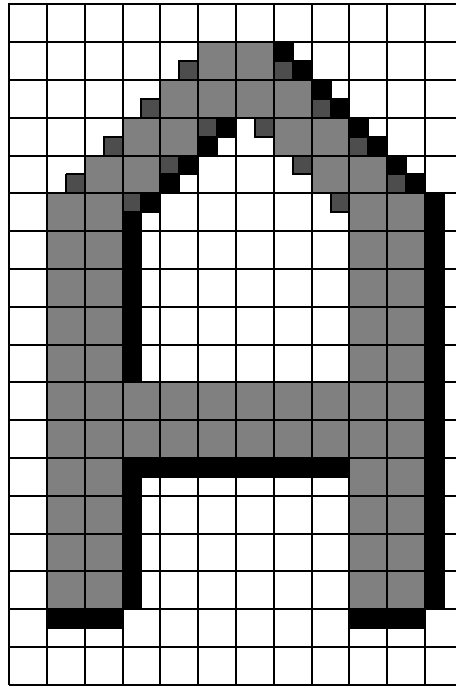


Figure 17-5. OSD-ROM Pixel Matrix with Northwest Shadow

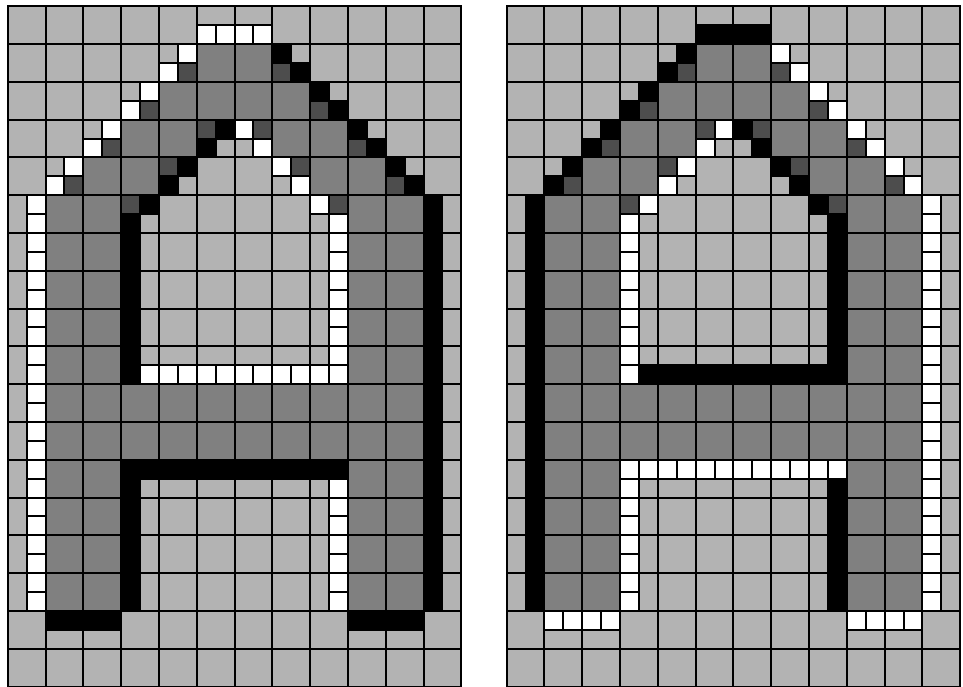
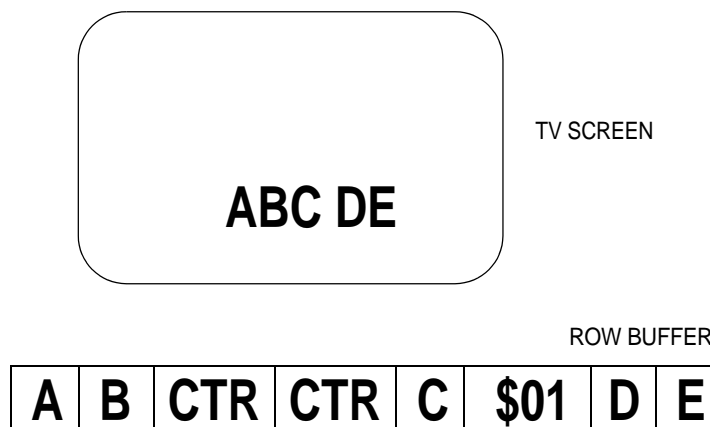


Figure 17-6. Example of 3D Shadow

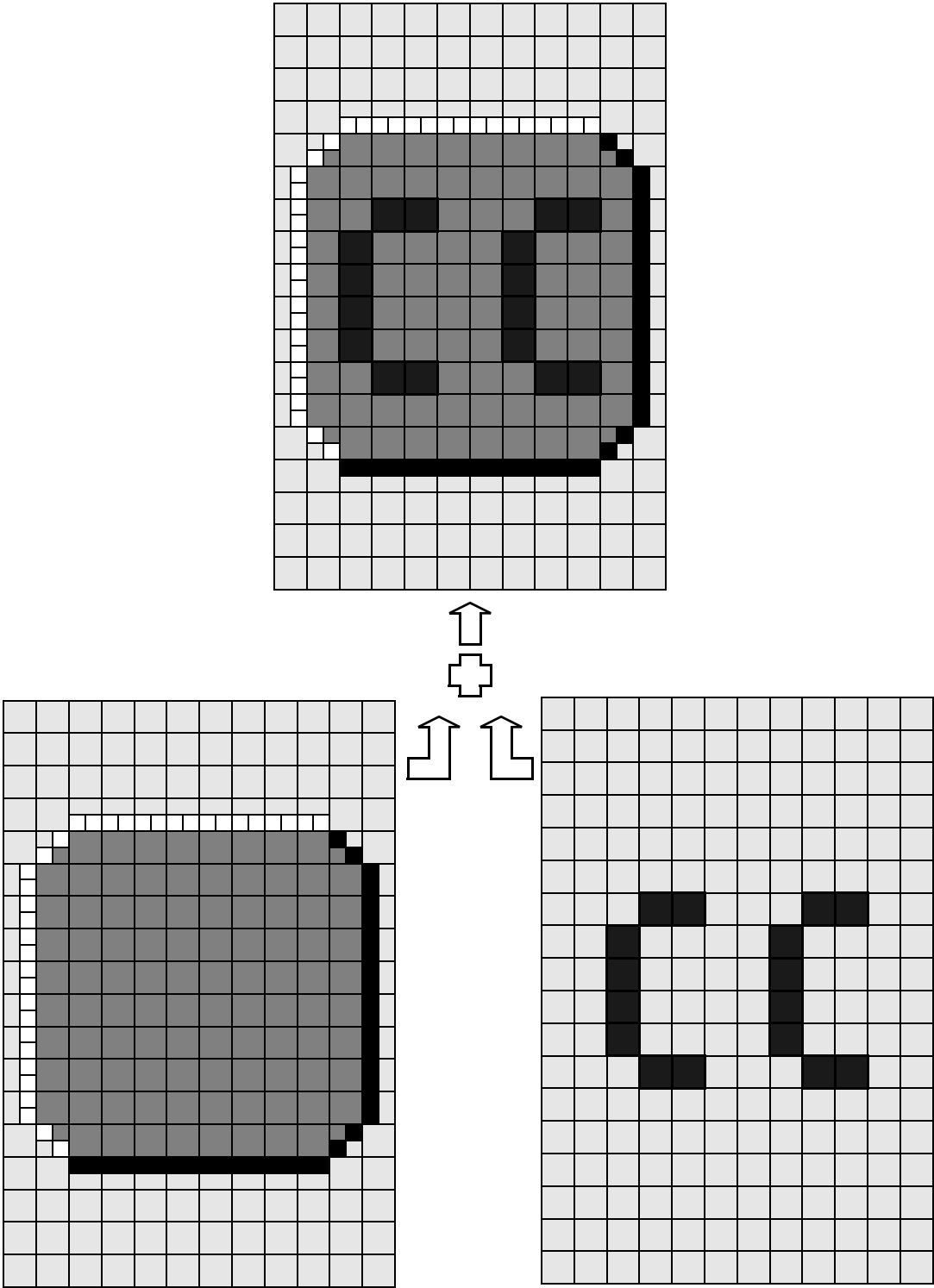


**Figure 17-7. Behavior of Control Characters in OSD Mode**

Character size is determined in the foreground control register. There are three possible sizes: 1W, 1.5W, and 2W, where W refers to the width. The heights of 1.5W and 2W are twice that of the 1W size. Size is fixed for an entire row. The 1W-size character is 12 pixels in width and 18 scan lines in height. A maximum of 24 1W-characters may be displayed per row. The 1.5W-size character is 18 pixels in width and 36 scan lines in height and has a maximum of 16 characters per row. The 2W-size character is 24 pixels in width and 36 scan lines in height, and has a maximum of 12 characters per row.

A unique feature that appears only in OSD mode is the possibility of overlaying two characters to form a composite one. This is controlled by the backspace (BS) bit, which appears in format A of the character control code. This code must be put between two characters and must be the only one between them. Its effect is to overlay the second character on top of the first one. [Figure 17-8](#) illustrates this feature. The way in which the foreground colors will be combined can be specified in one of two forms: first, the second character will always win; second, the resulting foreground color will be the X-OR of the individual RGB bits. Using this feature, it is possible to have up to three foreground colors, but there are some limitations. It is not possible to add attributes to the second character. As a consequence, only the first character can have rounding, black outline and shadow. The background color of the composite character will be the background color of the first character.

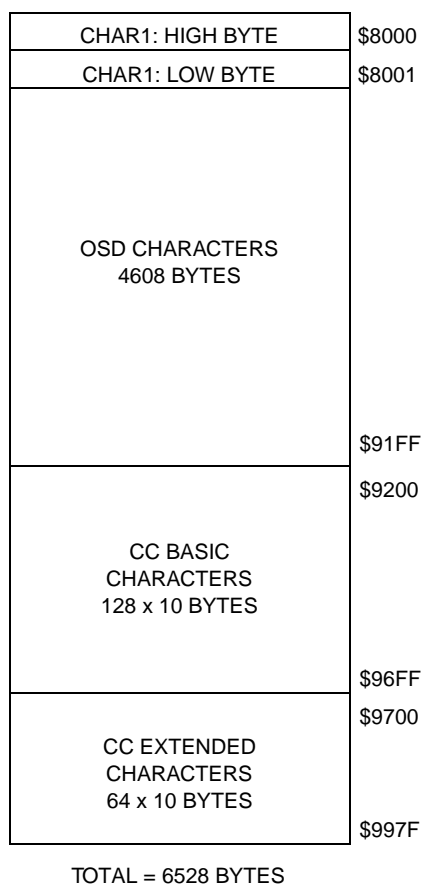




**Figure 17-8. Overlaying Characters to Get Two Foreground Colors**

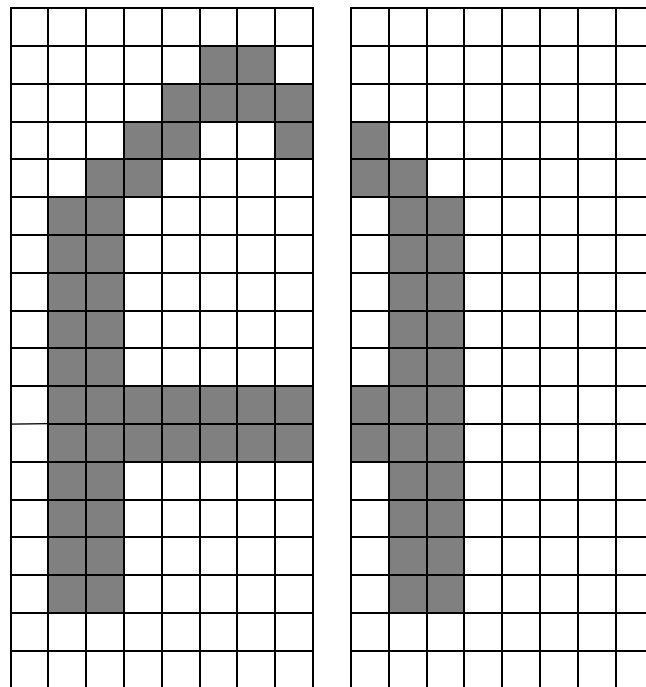
## 17.6 FLASH Programming Guidelines

The description presented here always refers to two separate ROM memories storing pixel information, one for closed-caption and one for on-screen display. Although this makes it easier to understand the module behavior, the actual physical implementation has only one programmable FLASH memory occupying the MCU memory space in the address range \$8000–\$9FFF. Although this memory has 8,192 bytes available, the OSD module will use only 6,528 bytes starting with the OSD data in address range \$8000–\$91FF, followed by the CC data in address range \$9200–\$997F. See [Figure 17-9](#).



**Figure 17-9. OSD FLASH Memory Map**

OSD characters are stored in 12 x 18 matrices. Each matrix row needs two bytes of storage space, one byte for the eight left-hand pixels and another byte for the four right-hand pixels; therefore, each OSD character takes up 36 bytes (see [Figure 17-10](#)). The OSD pixel matrix can be arbitrarily defined, except for the first two characters, which are always interpreted as border and background spaces and should be left empty.

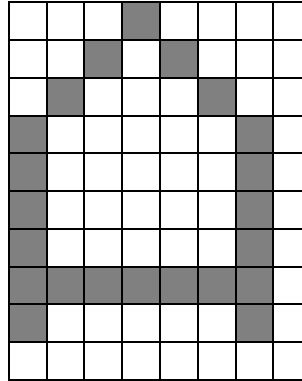


**Figure 17-10. OSD Pixel Matrix in FLASH Memory**

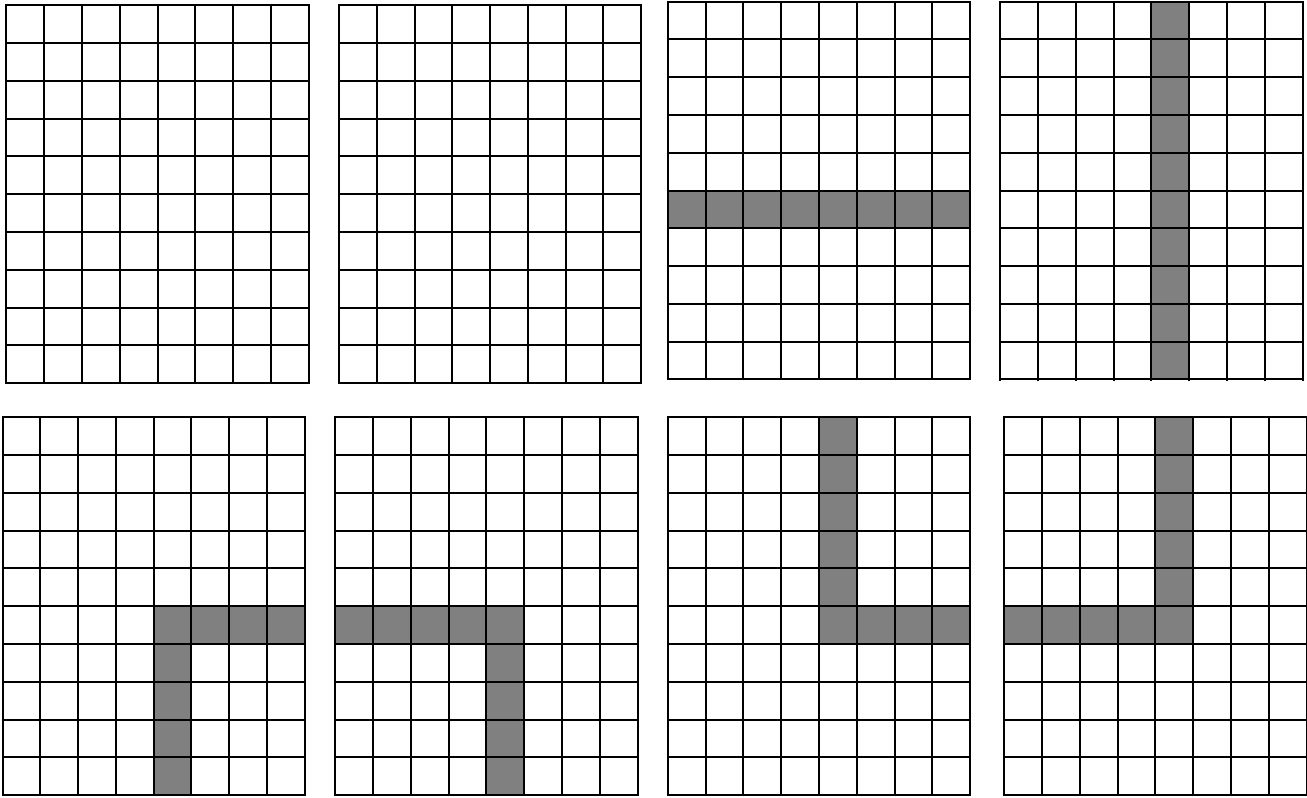
Although the CC pixel matrix size is 9 x 13, it is not necessary to store the whole matrix in memory because:

1. With very few exceptions (the six box making characters of the closed-caption standard) there is no need to form composite characters by abutment on the horizontal or vertical directions. The exceptions are treated by the output logic.
2. The last two lines can only be occupied by underline pixels, which are automatically inserted by the output logic.

As a consequence, only a reduced 8 x 10 matrix must be stored in the FLASH memory (see [Figure 17-11](#)). The first two characters are border and background spaces, so they must be left empty. Also, since the output logic must give special treatment to the six box making characters, they must be stored right after the two space characters and their pixel matrices must be identical to those shown in [Figure 17-12](#).



**Figure 17-11. CC Pixel Matrix in FLASH Memory**

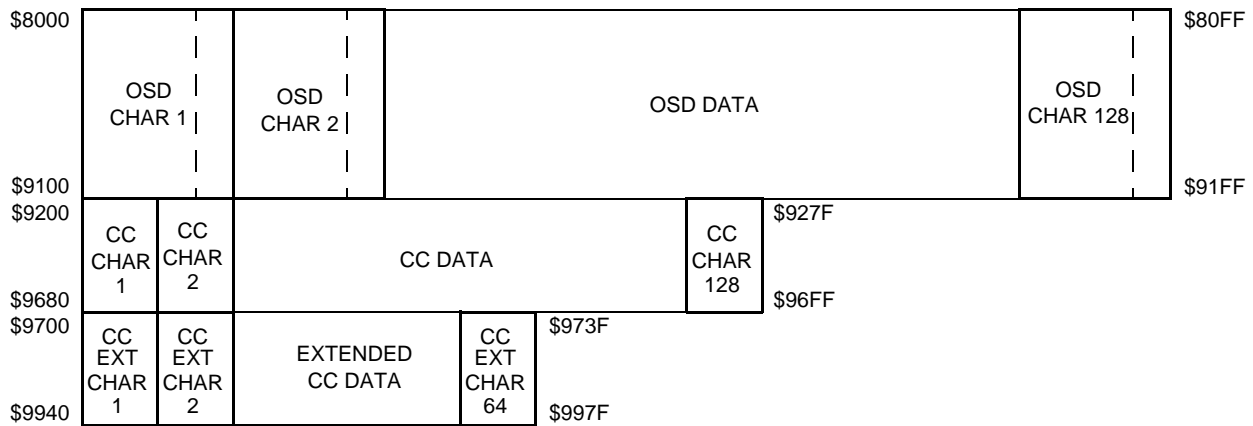


**Figure 17-12. Spaces and Box-Making Characters in CC Mode**

# On-Screen Display Module (OSD)

**Figure 17-13** shows the overall organization of the OSD FLASH memory. The OSD portion is organized as 18 consecutive blocks of 256 bytes. Each block contains one pixel matrix line of each of the 128 characters. Each line of each character uses two bytes. The first byte has the eight left-hand pixels of the line, whereas the second byte contains the four right-hand pixels.

The CC portion is organized as 10 consecutive blocks of 128 bytes, each byte being a line of a basic CC character, plus 10 consecutive blocks of 64 bytes, each byte being a line of an extended CC character.



**Figure 17-13. Pixel Matrices Organization in FLASH**

## 17.7 Programming Guidelines

This information is available to the programmer for characterization of the displayed row:

- The vertical and horizontal placement of the active display area
- The scan line number within the active display area on which the character row is to start
- The displaying mode for the row: CC mode or OSD mode
- If in OSD mode, the character size from three choices
- These attributes that should be applied on the characters of the row:
  - CC mode: rounding, black outline, underline, italics, and blinking
  - OSD mode: rounding, black outline, blinking, and shadow
- The region of scan lines of the row to display
- The foreground color from 16 choices
- The background color from 16 choices
- A string of up to 34 characters or control codes

In addition, several control registers govern when and how the OSD interacts with the rest of the system.

### 17.7.1 Setup

Initializing the OSD involves enabling the appropriate modules, identifying the TV system, defining how interrupts are to be used, enabling outputs, defining the active levels of both inputs and outputs, and setting up the default display characteristics.

For the OSD to operate, the OSD, PLL, and DSL must be enabled. The PLL provides clock signal synchronized to the television chassis horizontal, and the DSL provides the field information for interlacing. The PLL has a startup time that must elapse before the OSD should be used for display.

The received TV system must be identified in order to know if it is a 525-line or 625-line system. Upon identification, the vertical delay of the active display region should be programmed into the vertical delay register for proper centering of the display. To help the software identifying the TV system, an interrupt is optionally generated at the starting edge of the vertical sync pulse. By measuring the time between two of these interrupts, it is possible to distinguish between a 50-Hz system (625 lines) and a 60-Hz system (525 lines). After the identification, the interrupt can be disabled by resetting the VSIEN bit on the enable control register.

If the ELIEN bit of the enable control register is set, the OSD will generate an interrupt whenever the target scan line defined in the event line register matches the current scan line count in the event register. The user can specify, with the XFER bit in the enable control register, if the OSD transfers the external rank of registers to the internal rank when the interrupt occurs. If transferring is not enabled, then the OSD is interrupting only to indicate a scan line match. Transferring must be enabled to display more than one row of characters.

The active levels of HSYNC, VSYNC, FBKG, R, G, B, and I are all programmable. The character registers and display attributes must be initialized, since most of these are undefined upon reset. When the OSD is disabled (OSDEN = 0) and the PLL is enabled (PLLEN = 1), writing to the external rank of registers also writes to the internal rank. The character registers should be set to \$00 to indicate border space characters if no display is required upon startup. The border color must be defined in the border control register.

### 17.7.2 Interrupt Servicing

Two possible sources of interrupt can be masked independently:

- VSYNC interrupt, generated at the starting edge of the internal vertical sync pulse.
- Event line match interrupt, generated when the target scan line in the event line register matches the current scan line (event register).



Upon receiving an interrupt, the CPU must read the status register to identify the source of the interrupt and then clear the interrupt flags by writing any value into the status register. The flag bits corresponding to these interrupts are VSYNF (VSYNC flag) and ELMF (event line match flag).

If the received interrupt is an event line match interrupt, the CPU must update the information for the next row to be displayed. This information consists of:

- The display mode (CC mode or OSD mode)
- The target scan line
- The region of lines to display
- Character size (OSD mode)
- Character attributes valid for the entire row
- Character codes and mid-row attribute codes

The display mode is programmed on bit 7 of the matrix start register. The first scan line of the row is defined in the event line register. The region of lines to be displayed is programmed on matrix start register and matrix end register.

**NOTE:** *If the value programmed on the event line register causes the first line of the row to be after the last line of the TV picture, no ELMF interrupt will be issued because the internal line counter will never reach the value programmed on the register.*

The attributes for the characters of the row are defined using the foreground and background control registers, except for blinking, which is defined in the matrix start register.

The character codes and the mid-row attribute codes are stored in character register 1 to 34. Since in OSD mode a maximum of 24 characters can be displayed, it is not necessary to write all 34 character registers. However, the internal circuit will try to read up to three consecutive control characters past the last displayed character, in order to catch possible attribute changes after the end of the row. As a consequence, if less than three control characters (or none) will be

written after the end of the row, it is recommended to write at least one non-control character (anyone) terminating the sequence.

The worst-case time for updating the registers occurs when displaying two vertically neighbor rows. In a 525-line system with a 15,734-Hz line-rate, this interval is approximately 826  $\mu$ s. If the next row to be displayed is not immediately below the current one, this time interval is larger.

### 17.7.3 Software Controlled Features

Two features included into the OSD need software support:

- Soft scrolling
- Blinking.

Soft scrolling is accomplished by manipulating the event line register, the matrix start register and the matrix end register on a periodic basis. The event line register contains the target scan line and should be gradually decreased to scroll a display up the screen. The matrix start register contains the starting line and the matrix end register contains the ending line of a row display. When scrolling up from the bottom of the screen, the range of lines should be gradually increased from the top line of the row. When scrolling up off the top of the screen, the range of lines should be gradually decreased to the bottom line of the row.

Blinking text is achieved by using mid-row control codes and the BLINKEN bit in the border control register. Blinking text is designated by a preceding control code character with the BLINK bit set, and a following control code with BLINK cleared. The designated text will be replaced with background space characters whenever the BLINKEN bit is set, so toggling this bit at the desired blinking rate will produce the blinking effect on the text.

## 17.8 Input and Output

The OSD has 11 dedicated pins. Seven of them are digital CMOS compatible signals. Two are dedicated power pins for the analog part, and two are external filter pins for the PLL.

### 17.8.1 Power Supply Pins

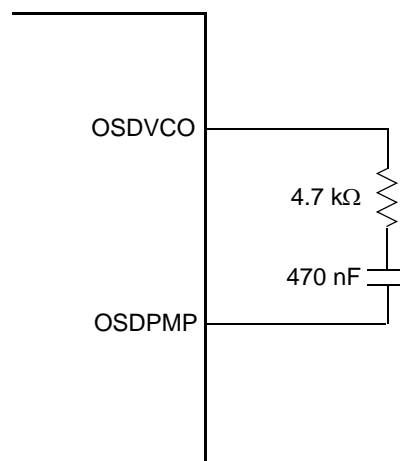
$V_{DDOSD}$  and  $V_{SSOSD}$  are two dedicated power pins that feed the analog circuits contained in the OSD and the DSL. These pins are provided to improve noise immunity of the analog circuits.

### 17.8.2 Input Pins

The input pins are HSYNC and VSYNC. These inputs provide the horizontal and vertical timing reference from the external video system. Both pins have internal Schmitt triggers to improve noise immunity. The HINV and VINV bits in the output control register select the polarity of the HSYNC and VSYNC inputs.

### 17.8.3 PLL Filter Pins

Pins OSDVCO and OSDPMP are used to tailor the 14-MHz PLL loop filter and center frequency. The recommended filter circuit for the PLL is shown in [Figure 17-14](#).



**Figure 17-14. PLL Filter Circuit**

### 17.8.4 Output Pins

The output pins are R, G, B, I, and FBKG. Signals R, G, and B are the red, green, and blue color-encoded pixel signals that form the characters to be displayed. The polarity of these bits is determined by the CINV bit in the output control register.

The FBKG (fast blanking) signal is intended to blank the external video source so that the combination of OSD and external video is non-additive. FBKG is asserted in five places:

- Character foreground
- Character rounding, if enabled
- Character black outline or shadow, if enabled
- Character background, if not transparent
- Border, if not transparent

The FBINV bit in the output control register determines FBKG output polarity.

The I (intensity) pin is intended to control the intensity of R, G, and B outputs, expanding the color palette to 16 colors. The IINV bit in the output control register determines I output polarity.

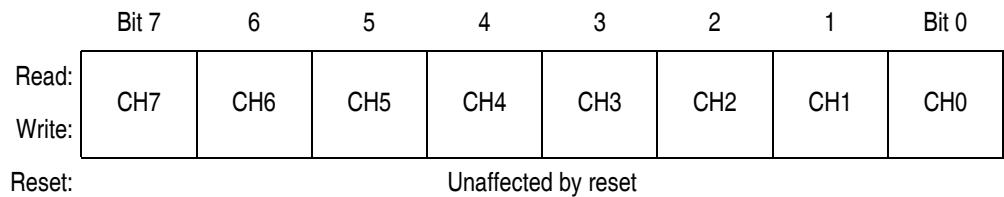
## 17.9 Registers

The OSD has 46 registers, 38 of which are double-ranked to prevent CPU access from disturbing the video display.

### 17.9.1 OSD Character Registers

The 34 double-ranked character registers contain character or control codes of an entire row. A character code specifies a position of the CC ROM or the OSD ROM, depending on the display mode programmed on the DMODE bit of the matrix start register. A video control character is used to modify display attributes in the middle of the row. The control character behaves differently and its bits have different meanings, depending on the display mode.

Address: \$0020–\$0041



**Figure 17-15. OSD Character Registers  
(OSDCHAR1–OSDCHAR34)**

#### CH7 — Character Bit 7

This is a control bit that determines whether CH[6:0] is interpreted as a normal character code or a code with special meaning:

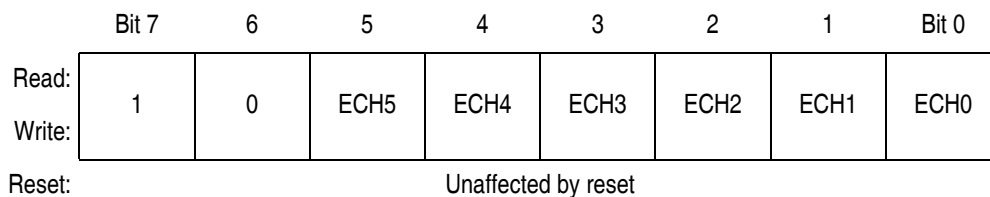
- 1 = CH[6:0] has a different meaning depending on the display mode
- 0 = CH[6:0] is a character code.

#### CH[6:0] — Character Bits 6 to 0

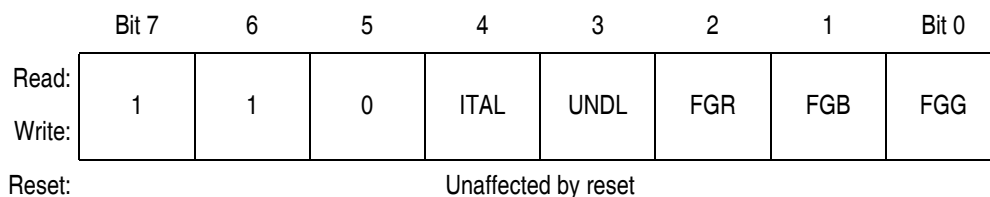
These seven bits contain context dependent information. If CH7 is 1, CH[6:0] is a special code that must be interpreted differently depending on the display mode. In OSD mode, it is always a control code that specifies mid-row changes in attributes. In CC mode, it may specify an attribute change or may contain a character of the extended set of 64 characters.

If CH7 is 0, CH[6:0] is a normal character code specifying one of the 128 OSD-ROM characters, or one of the 128 non-extended CC-ROM characters.

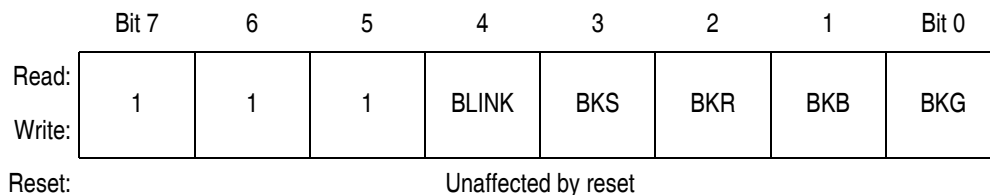
In closed-caption mode, a control character is displayed as a background space, except when it contains an extended character. The formats shown in [Figure 17-16](#), [Figure 17-17](#), and [Figure 17-18](#) are valid only in CC mode.



**Figure 17-16. CC Mode Control Character — Format A**



**Figure 17-17. CC Mode Control Character — Format B**



**Figure 17-18. CC Mode Control Character — Format C**

### ECH[5:0] — Extended Character Bits

These six bits specify an address of the extended page of the CC ROM.

### ITAL — Italics Bit

Determines when characters are displayed in italics by slanting.

1 = Subsequent foreground will be italicized

0 = Subsequent foreground will not be italicized

#### UNDL — Underline Bit

This bit determines if characters are underlined.

1 = Subsequent characters are underlined.

0 = Subsequent characters are not underlined.

#### FGR, FGB, and FGG — Foreground Color Bits

These bits define the foreground color for the subsequent characters being displayed. The intensity of the foreground color in CC mode is not modifiable inside the row. Bit FGI of the foreground control register sets the foreground color intensity for the entire row.

#### BLINK — Foreground Blink Bit

Determines, in conjunction with the BLINKEN bit in the border control register, which text is blinking on the screen. The BLINKEN bit is toggled, through software, at the desired blinking frequency.

Foreground text (including underline) that is preceded by BLINK will flash at the BLINKEN frequency.

1 = Subsequent foreground will be replaced by background color when BLINKEN = 1.

0 = Subsequent foreground will not blink.

#### BKS — Background Solid Bit

This bit determines whether the background is transparent or visible.

1 = The background is a solid color determined by BKR, BKB, and BKG

0 = The background is transparent

#### BKR, BKB, and BKG — Background Color Bits

These bits define the background color for the subsequent characters being displayed. BKS must be set to display a background color. The intensity of the background color in CC mode is not modifiable inside the row. Bit BKI of the background control register sets the background color intensity for the entire row.

In on-screen display mode, up to two consecutive control characters can be placed together without introducing a space on the screen. Therefore, attributes belonging to different registers can be modified on a character by character basis inside words. The control character bits in OSD mode are defined in [Figure 17-19](#), [Figure 17-20](#), and [Figure 17-21](#).

|        | Bit 7               | 6 | 5  | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------|---|----|-----|-----|-----|-----|-------|
| Read:  | 1                   | 0 | BS | OVM | FGI | FGR | FGB | FGG   |
| Write: |                     |   |    |     |     |     |     |       |
| Reset: | Unaffected by reset |   |    |     |     |     |     |       |

**Figure 17-19. OSD Mode Control Character — Format A**

|        | Bit 7               | 6 | 5 | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------|---|---|-----|-----|-----|-----|-------|
| Read:  | 1                   | 1 | 0 | BKS | BKI | BKR | BKB | BKG   |
| Write: |                     |   |   |     |     |     |     |       |
| Reset: | Unaffected by reset |   |   |     |     |     |     |       |

**Figure 17-20. OSD Mode Control Character — Format B**

|        | Bit 7               | 6 | 5 | 4 | 3 | 2     | 1     | Bit 0 |
|--------|---------------------|---|---|---|---|-------|-------|-------|
| Read:  | 1                   | 1 | 1 |   |   | BLINK | SHAD1 | SHAD0 |
| Write: |                     |   |   |   |   |       |       |       |
| Reset: | Unaffected by reset |   |   |   |   |       |       |       |

= Unimplemented

**Figure 17-21. OSD Mode Control Character — Format C**

## BS — Back Space Bit

Determines if the next character is going to be displayed on top of the preceding one. If this bit is set, there can be only one control character between the two characters that are to be overlaid. The second character will not have black-outline or shadow and only its foreground color and intensity can be modified. If BS is set, the foreground color specified in this register applies only to the character to be overlaid and not to subsequent characters of the row. An example of overlay is shown in [Figure 17-18](#).

- 1 = The next character will be overlaid on top of the preceding one.
- 0 = The next character will not be overlaid.



### OVM — Overlay Method Bit

If BS = 1, this bit determines how the second character will be overlaid on top of the first one.

1 = The RGB of the resulting foreground pixels will be the X-OR of the individual RGB foreground pixels, and the background will be the background of the first character.

0 = The foreground pixels of the second character will take precedence over the foreground pixels of the first character, and the background will be the background of the first character.

### FGI — Foreground Intensity Bit

This bit determines the intensity of the foreground color.

1 = Subsequent foreground pixels will have full intensity.

0 = Subsequent foreground pixels will have half intensity.

### FGR, FGB, and FGG — Foreground Color Bits

These bits define the foreground color for the subsequent characters being displayed.

### BKS — Background Solid Bit

Determines whether the background is transparent or visible.

1 = The background is a solid color determined by BKR, BKB, and BKG.

0 = The background is transparent.

### BKI — Background Intensity Bit

This bit determines the intensity of the background color.

1 = Subsequent background pixels will have full intensity.

0 = Subsequent background pixels will have half intensity.

### BKR, BKB, and BKG — Background Color Bits

These bits define the background color for the subsequent characters being displayed.

### BLINK — Blink Foreground Bit

This bit determines, in conjunction with the BLINKEN bit in the border control register, which text is blinking on the screen. The BLINKEN bit is toggled, through software, at the desired blinking frequency.

Foreground text that is preceded by BLINK will flash at the BLINKEN frequency.

1 = Subsequent foreground will be replaced by background color when BLINKEN = 1.

0 = Subsequent foreground will not blink.

### SHAD1 and SHAD0 — Shadow Control Bits

These bits specify the type of shadow that will be applied to subsequent foreground characters.

**Table 17-1. SHAD1 and SHAD0 Meaning**

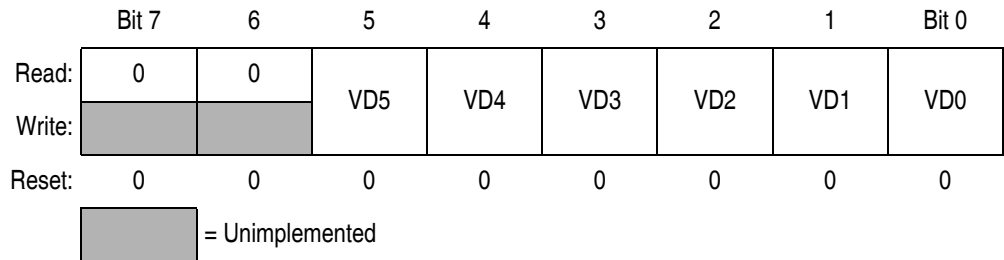
| SHAD1 and SHAD0 | Action  |
|-----------------|---|
| 00              | No shadow will be applied, but will be black-outlined if BOEN = 1 |
| 01              | Northwest shadow  |
| 10              | Not-pressed 3D shadow   |
| 11              | Pressed 3D shadow   |

### 17.9.2 OSD Vertical Delay Register

This register defines the vertical initial position in the active display area, as shown in [Figure 17-2](#). The position is specified in number of lines from the starting edge of VSYNC.

The position of the active display area should be centered on the screen by software according to the TV standard being received (525 or 625 lines). By adjusting the vertical position in this register, it is not necessary to consider different TV system timings when programming the event line register.

Address: \$0042



**Figure 17-22. OSD Vertical Delay Register (OSDVDR)**

VD[5:0] — Vertical Delay Bits


Number of lines from the starting edge of VSYNC until the beginning of the active display area

## 17.9.3 OSD Horizontal Delay Register

This register defines the horizontal delay of the active display area with respect to the starting edge of the internal (conditioned with HINV bit) HSYNC. Each increment provides approximately 0.14  $\mu$ s of delay, regardless of the character size programmed in the CHHS and CHWS bits. In addition to the specified delay, there is a built-in fixed delay of approximately 14.1  $\mu$ s.

Address: \$001B

|        | Bit 7 | 6 | 5 | 4   | 3   | 2   | 1   | Bit 0 |
|--------|-------|---|---|-----|-----|-----|-----|-------|
| Read:  | 0     | 0 | 0 | HD4 | HD3 | HD2 | HD1 | HD0   |
| Write: |       |   |   |     |     |     |     |       |
| Reset: | 0     | 0 | 0 | 0   | 0   | 0   | 0   | 0     |

 = Unimplemented

**Figure 17-23. OSD Horizontal Delay Register (OSDHDR)**

HD[4:0] — Horizontal Delay Bits

These bits define the horizontal delay of the active display area from the starting edge of HSYNC.

## 17.9.4 OSD Foreground Control Register

This double-ranked register is used to define character row attributes prior to display of the row. With the exception of CHHS and CHWS, all attributes defined by this register are applicable to both CC mode and OSD mode. The character size can be changed only in OSD mode. The foreground color can be changed mid-row by the use of control

characters. The intensity of the color, defined by bit FGI, can be modified mid-row only in OSD mode.

Address: \$001C

|        | Bit 7 | 6    | 5     | 4    | 3   | 2   | 1   | Bit 0 |
|--------|-------|------|-------|------|-----|-----|-----|-------|
| Read:  | CHHS  | CHWS | RNDEN | BOEN | FGI | FGR | FGB | FGG   |
| Write: |       |      |       |      |     |     |     |       |
| Reset: | 0     | 0    | 0     | 0    | 0   | 0   | 0   | 0     |

**Figure 17-24. OSD Foreground Control Register (OSDFCR)**

#### CHHS — Character Height Select Bit

This bit determines whether characters in OSD mode are displayed at standard height and width, or at double height with width selected by CHWS. Character height is selected for an entire row.

- 1 = 2X height characters, CHWS will determine the character width
- 0 = Standard size (height and width) characters

#### CHWS — Character Width Select Bit

This bit determines the character width when CHHS = 1. Character width is selected for an entire row.

- 1 = 2X width characters are selected.
- 0 = 1.5X width characters are selected.

#### RNDEN — Rounding Enable Bit

This bit determines if foreground characters are rounded.

- 1 = Foreground will be rounded.
- 0 = Foreground will not be rounded.

#### BOEN — Black Outline Enable Bit

This bit determines if foreground characters are black outlined when shadow is disabled (SHAD1 = 0 and SHAD2 = 0). Shadow takes precedence over black outline. If any of the forms of shadow is set, the character will not be black outlined.

- 1 = Foreground characters are black outlined if shadow is disabled.
- 0 = Foreground characters are not black outlined.

## FGI — Foreground Intensity Bit

This bit determines the intensity of the foreground color.

1 = Foreground pixels will have full intensity.

0 = Foreground pixels will have half intensity.

## FGR, FGB, and FGG — Foreground Color Bits

These bits define the foreground color for the characters of the row.

### 17.9.5 OSD Background Control Register

This double-ranked register is used to define character row attributes prior to display of the row. The shadow attribute is only applicable to OSD mode and is ignored in CC mode. The rest of the attributes are applicable to both modes. The background color can be changed mid-row by the use of control characters, but the intensity of the color, defined by bit BKI, can be modified mid-row in OSD mode only.

When background is transparent, the video image that substitutes it can be attenuated by setting bit BKHT. This setting cannot be changed mid-row.

Address: \$001D

|        | Bit 7 | 6     | 5     | 4   | 3   | 2   | 1   | Bit 0 |
|--------|-------|-------|-------|-----|-----|-----|-----|-------|
| Read:  | BKHT  | SHAD1 | SHAD0 | BKS | BKI | BKR | BKB | BKG   |
| Write: |       |       |       |     |     |     |     |       |
| Reset: | 0     | 0     | 0     | 0   | 0   | 0   | 0   | 0     |

**Figure 17-25. OSD Background Control Register (OSDBKCR)**

## BKHT — Background Half-Tone Bit

This bit controls the video intensity when transparent background is selected.

1 = Video will be attenuated in transparent areas.

0 = Video will not be attenuated.

### SHAD1 and SHAD0 — Shadow Control Bits

These bits specify the type of shadow that will be applied to all characters of the row. [Table 17-1](#) gives the meaning of these bits.

### BKS — Background Solid Bit

This bit determines whether the background is transparent or visible.

1 = Background is a solid color determined by BKR, BKB, and BKG.  
0 = Background is transparent.

### BKI — Background Intensity Bit

This bit determines the intensity of the background color.

1 = Row background color will have full intensity  
0 = Row background color will have half intensity

### BKR, BKB, and BKG — Background Color Bits

These bits define the row background color.

## 17.9.6 OSD Border Control Register

This register is used to define border characteristics.

Address: \$0043

|        | Bit 7   | 6    | 5     | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------|------|-------|-----|-----|-----|-----|-------|
| Read:  | BLINKEN | BOHT | VMUTE | BOS | BOI | BOR | BOB | BOG   |
| Write: |         |      |       |     |     |     |     |       |
| Reset: | 0       | 0    | 0     | 0   | 0   | 0   | 0   | 0     |

**Figure 17-26. OSD Border Control Register (OSDBCR)**

### BLINKEN — BLINK Enable Bit

This bit determines if text following a control code with BLINK set will be displayed. BLINKEN should be toggled in software to establish the desired blinking frequency.

1 = Text following a control code with BLINK set will not be displayed.  
0 = Text following a control code with BLINK set will be displayed.

0 = Text following a control code with BLINK set will be displayed.

### BOHT — Video Tone Bit

This bit controls the video intensity when transparent background or transparent border is selected.

1 = Video will be attenuated in transparent areas.

0 = Video will not be attenuated.

### VMUTE — Video Muting Bit

This bit controls the video muting function, in which the whole screen is filled with a solid color.

1 = The whole screen is filled with a solid color determined by BOR, BOB and BOG

0 = The screen is enabled to show video, OSD, and caption.

### BOS — Border Solid Bit

This bit determines whether the border is transparent or visible.

1 = The border is a solid color determined by BOR, BOB, and BOG.

0 = The border is transparent.

### BOI — Border Intensity Bit

This bit determines the intensity of the border color.

1 = Border pixels will have full intensity.

0 = Border pixels will have half intensity.

### BOR, BOB, and BOG — Border Color Bits

These bits define the border color.



### 17.9.7 OSD Enable Control Register

This register contains enable and control bits for the OSD.

Address: \$FE0B

|        | Bit 7 | 6     | 5     | 4    | 3     | 2    | 1    | Bit 0 |
|--------|-------|-------|-------|------|-------|------|------|-------|
| Read:  | OSDEN | ELIEN | VSIEN | XFER | PLLEN | MEM1 | MEM0 | SCAN  |
| Write: |       |       |       |      |       |      |      |       |
| Reset: | 0     | 0     | 0     | 0    | 0     | 0    | 0    | 0     |

**Figure 17-27. OSD Enable Control Register (OSDECTR)**

#### OSDEN — OSD Enable Bit

This bit determines whether the OSD is enabled or disabled. The PLLEN bit must also be set for OSD operation, and the DSL must be enabled to provide field information for interlacing.

- 1 = OSD enabled
- 0 = OSD disabled

#### ELIEN — Event Line Match Interrupt Enable Bit

This bit determines if the ELMF bit in the status register is enabled to generate interrupt requests to the CPU.

- 1 = Event line match interrupt enabled
- 0 = Event line match interrupt disabled

#### VSIEN — Vertical Sync Interrupt Enable Bit

This bit determines if the VSINF bit in the status register is enabled to generate interrupt requests to the CPU.

- 1 = Vertical sync interrupt enabled
- 0 = Vertical sync interrupt disabled

#### XFER — External to Internal Rank Transfer Enable Bit

This bit determines if the external rank of registers is transferred to the internal rank upon an event line match.

- 1 = Transfer enabled
- 0 = Transfer disabled

## PLLEN — PLL Enable Bit

This bit determines if the phase-locked loop oscillator is enabled. When enabling the PLL, the program must wait for the PLL to stabilize before activating the OSD to achieve a stable display. The PLL should be disabled before stop mode is entered. In addition, it may be useful to stop the PLL whenever OSD data are not currently being displayed, to eliminate ingress of 14-MHz interference to the RF, IF, or base-band portions of the external video chain.

1 = PLL enabled

0 = PLL disabled

## MEM[1:0] — Memory Test Mode Bits

These two bits can only be accessed in peripheral test mode (PTM) or CPU test mode (CTM). They allow to choose one of the two character register ranks to be directly connected to the internal bus, bypassing the internal OSD control. [Table 17-2](#) shows the meaning of these bits.

**Table 17-2. Memory Test Mode Bits**

| MEM[1:0] | What Can be Accessed via Internal Bus  |
|----------|--|
| 00   01  | One of the character-register ranks, determined by the internal OSD circuitry; functional behavior |
| 10       | Character registers — rank 1   |
| 11       | Character registers — rank 2   |

## SCAN — Scan Test Mode Bit

This bit also can be accessed only in one of the test modes. It causes the OSD module to reconfigure some of its internal circuitry to prepare itself for production test.

1 = OSD module is reconfigured to production test.

0 = OSD module keeps its functional structure.

### 17.9.8 OSD Event Line Register

This register contains the target scan line address for the next row to be displayed. The range of lines considered for display is inside the active display area, as shown in [Figure 17-2](#). Writing a 0 into the event line register means that the row should be displayed at the first line of the active display area. The vertical placement of the active area must be adjusted using the vertical delay register.

Address: \$0044

|        | Bit 7 | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|-------|-----|-----|-----|-----|-----|-----|-------|
| Read:  | EL7   | EL6 | EL5 | EL4 | EL3 | EL2 | EL1 | EL0   |
| Write: |       |     |     |     |     |     |     |       |
| Reset: | 0     | 0   | 0   | 0   | 0   | 0   | 0   | 0     |

**Figure 17-28. OSD Event Line Register (OSDELR)**

EL[7:0] — Event Line Number Bits

These bits define where to display the first scan line of the next character row.


**NOTE:** *If the value programmed on the event line register causes the first line of the row to be after the last line of the TV picture, no ELMF interrupt will be issued because the internal line counter will never reach the value programmed on the register.*

## 17.9.9 OSD Event Count Register

This read-only register makes the internal scan line counter visible to the CPU.

Address: \$0045

|        | Bit 7               | 6   | 5   | 4   | 3   | 2   | 1   | Bit 0 |
|--------|---------------------|-----|-----|-----|-----|-----|-----|-------|
| Read:  | EV7                 | EV6 | EV5 | EV4 | EV3 | EV2 | EV1 | EV0   |
| Write: |                     |     |     |     |     |     |     |       |
| Reset: | Unaffected by Reset |     |     |     |     |     |     |       |

 = Unimplemented

**Figure 17-29. OSD Event Count Register (OSDECR)**

### EV[7:0] — Event Count Bits


These bits mirror the internal scan line counter contents.

## 17.9.10 OSD Output Control Register

This register contains control bits for the input and output pins of the OSD.

Address: \$FE0A

|        | Bit 7 | 6    | 5    | 4     | 3    | 2     | 1    | Bit 0 |
|--------|-------|------|------|-------|------|-------|------|-------|
| Read:  | 0     | HINV | VINV | FDINV | CINV | FBINV | IINV | 0     |
| Write: |       |      |      |       |      |       |      |       |
| Reset: | 0     | 0    | 0    | 0     | 0    | 0     | 0    | 0     |

 = Unimplemented

**Figure 17-30. OSD Output Control Register (OSDOCR)**

### HINV — HSYNC Polarity Bit

This bit determines whether the HSYNC input is active low or active high.

1 = HSYNC input is active low and is inverted internally.

0 = HSYNC input is active high.

#### VINV — VSYNC Polarity Bit

This bit determines whether the VSYNC input is active low or active high.

1 = VSYNC input is active low and is inverted internally.

0 = VSYNC input is active high.

#### FDINV — Field Invert Bit

This bit determines whether the odd/even field indicator from the data slicer is inverted or not inverted.

1 = Odd/even field indicator from the data slicer is inverted.

0 = Odd/even field indicator from the data slicer is not inverted.

#### CINV — Color Invert Bit

This bit determines whether the R, G and B outputs are active low or active high.

1 = R, G, and B outputs are active low.

0 = R, G, and B outputs are active high.

#### FBINV — Fast Blanking Invert Bit

This bit determines whether the FBKG output is active low or active high.

1 = FBKG output is active low.

0 = FBKG output is active high.

#### IINV — Intensity Invert Bit

This bit determines whether the Intensity output (I) is active low or active high.

1 = I output is active low.


0 = I output is active high.

## 17.9.11 OSD Status Register

This register contains the interrupt flags and provides visibility of the input signals.

Address: \$0046

|        | Bit 7 | 6     | 5    | 4    | 3 | 2      | 1       | Bit 0  |
|--------|-------|-------|------|------|---|--------|---------|--------|
| Read:  | ELMF  | VSINF | HSYN | VSYN | 0 | VCOTST | DSLSTST | PLLTST |
| Write: |       |       |      |      |   |        |         |        |
| Reset: | 0     | 0     | U    | U    | 0 | 0      | 0       | 0      |

 = Unimplemented

**Figure 17-31. OSD Status Register (OSDSR)**

### ELMF — Event Line Match Interrupt Bit

This bit is set when the current field scan line (event register) matches the target field scan line (event line register). It will cause an interrupt if the ELIEN bit is set in the enable control register. The interrupt must be acknowledged by writing any value to the register.

### VSINF — Vertical Sync Interrupt Bit

This bit is set at every starting edge of VSYNC. It will cause an interrupt if the VSIEN bit is set in the enable control register. The interrupt must be acknowledged by writing any value to the register.

### HSYN — Horizontal Sync Pulse Bit

This read-only bit provides visibility to the HSYNC input. A logic 1 indicates the presence of a horizontal sync pulse.

### VSYN — Vertical Sync Pulse Bit

This read-only bit provides visibility to the VSYNC input. A logic 1 indicates the presence of a vertical sync pulse.

#### VCOTST — VCO Test Mode Bit

This bit can be accessed only in peripheral test mode (PTM) or CPU test mode (CTM). It is provided to facilitate the test of the VCO circuit inside the OSD-PLL, allowing observation of the PLL clock frequency through pin FBKG.

1 = A signal with half of the PLL clock frequency is connected to pin FBKG.

0 = The fast blanking signal goes to FBKG.

#### DSLSTST — Data Slicer Test Mode Bit

This bit also can be accessed only in one of the test modes. It is used to reconfigure the analog part of the data slicer (DSL) circuit to allow its own test.

1 = DSL is put into test mode

0 = DSL is in functional mode

#### PLLSTST — PLL Test Mode

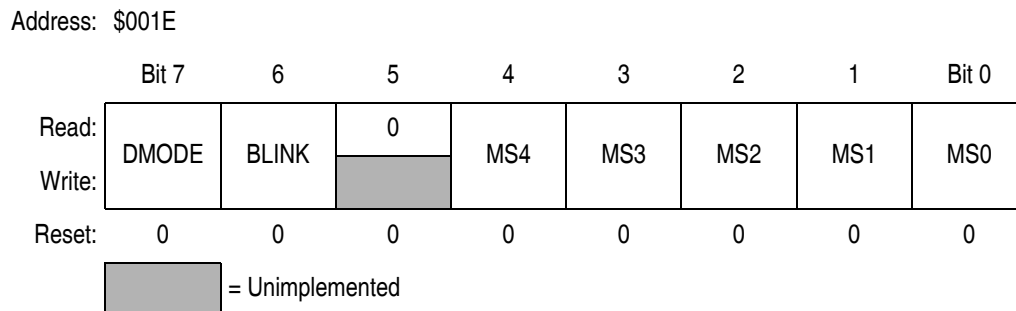
This bit also can be accessed only in one of the test modes. It is used to reconfigure the OSD-PLL to allow its own test.

1 = OSD-PLL is put into test mode.

0 = OSD-PLL is in functional mode.

### 17.9.12 OSD Matrix Start Register

This double-ranked register defines the display mode, the blinking attribute and the first line of the row to be displayed. The display mode, either CC or OSD, defines the size of the pixel matrix, what character ROM should be fetched, and what attributes can be applied to the characters along the row. It also defines how mid-row control characters must be interpreted. The blinking attribute can be modified mid-row by control characters.



**Figure 17-32. OSD Matrix Start Register (OSDMSR)**

### DMODE — Display Mode Bit

This bit defines the size of the pixel matrix, what character ROM should be fetched and what attributes can be applied to the characters along the row.

- 1 = Row will be displayed in CC mode
- 0 = Row will be displayed in OSD mode

### BLINK — Blink Foreground Bit

This bit determines, in conjunction with the BLINKEN bit in the border control register, which text is blinking on the screen. The BLINKEN bit is toggled, through software, at the desired blinking frequency. If BLINK is set in this register, the whole row will flash at the BLINKEN frequency.

- 1 = The entire row will be replaced by background color when BLINKEN = 1
- 0 = The row will not blink

### MS[4:0] — Matrix Start Line Bits

These bits set the starting scan line within the character row. MS[4:0] = 0 means that the row starts at the first scan line. Valid start numbers are 0 through 12 in CC mode and 0 through 17 in OSD mode.



### 17.9.13 OSD Matrix End Register

This double-ranked register defines the last line of the row to be displayed.

Address: \$001F

|        | Bit 7 | 6 | 5 | 4   | 3   | 2   | 1   | Bit 0 |
|--------|-------|---|---|-----|-----|-----|-----|-------|
| Read:  | 0     | 0 | 0 | ME4 | ME3 | ME2 | ME1 | ME0   |
| Write: |       |   |   |     |     |     |     |       |
| Reset: | 0     | 0 | 0 | 0   | 0   | 0   | 0   | 0     |

= Unimplemented

**Figure 17-33. OSD Matrix End Register (OSDMER)**

#### ME[4:0] — Matrix End Line Bits

These bits set the ending scan line within the character row. In CC mode, valid numbers are 0 through 12, and ME[4:0] = 12 means that the row is displayed through its last scan line. In OSD mode, valid numbers are 0 through 17, and ME[4:0] = 17 means that the row is displayed through its last scan line.

## 17.10 Low Power Modes

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

### 17.10.1 Wait Mode

The OSD remains active during wait mode, but it will be unable to interrupt the CPU and bring it out of this mode. It is recommended that the OSD and PLL be disabled before entering wait mode, unless a single row of fixed video output is desired to be displayed constantly.

### 17.10.2 Stop Mode

Although the OSD module and the PLL are not automatically disabled in stop mode, the FLASH memory will be disabled. As a consequence, the OSD module will not work. It is recommended that the OSD and PLL be disabled before entering stop mode.

### 17.11 Interrupts and Resets

The OSD has two sources of interrupts: the ELMF and VSINF flags in the OSD status register. These interrupts can be independently masked by bits ELIEN and VSIEN in the OSD enable control register. Both interrupts will cause the CPU to vector to the address stored in \$FFEE-\$FFEF.

## Section 18. Input/Output (I/O) Ports

### 18.1 Contents

|        |                           |     |
|--------|---------------------------|-----|
| 18.2   | Introduction              | 267 |
| 18.3   | Port A                    | 269 |
| 18.3.1 | Port A Data Register      | 269 |
| 18.3.2 | Data Direction Register A | 270 |
| 18.4   | Port B                    | 272 |
| 18.4.1 | Port B Data Register      | 272 |
| 18.4.2 | Data Direction Register B | 273 |
| 18.5   | Port C                    | 274 |
| 18.5.1 | Port C Data Register      | 274 |
| 18.5.2 | Data Direction Register C | 275 |


### 18.2 Introduction

Twenty-one (21) bidirectional input-output (I/O) pins form three 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.*

# Input/Output (I/O) Ports

| Addr.  | Register Name  | Bit 7  | 6                   | 5     | 4     | 3     | 2     | 1     | Bit 0 |       |
|--------|--|--------|---------------------|-------|-------|-------|-------|-------|-------|-------|
| \$0000 | Port A Data Register (PTA)<br>See <a href="#">page 269</a> .       | Read:  | PTA7                | PTA6  | PTA5  | PTA4  | PTA3  | PTA2  | PTA1  | PTA0  |
|        |  | Write: |                     |       |       |       |       |       |       |       |
|        |  | Reset: | Unaffected by reset |       |       |       |       |       |       |       |
| \$0001 | Port B Data Register (PTB)<br>See <a href="#">page 272</a> .       | Read:  | PTB7                | PTB6  | PTB5  | PTB4  | PTB3  | PTB2  | PTB1  | PTB0  |
|        |  | Write: |                     |       |       |       |       |       |       |       |
|        |  | Reset: | Unaffected by reset |       |       |       |       |       |       |       |
| \$0002 | Port C Data Register (PTC)<br>See <a href="#">page 274</a> .       | Read:  | 0                   | 0     | 0     | PTC4  | PTC3  | PTC2  | PTC1  | PTC0  |
|        |  | Write: |                     |       |       |       |       |       |       |       |
|        |  | Reset: | Unaffected by reset |       |       |       |       |       |       |       |
| \$0004 | Data Direction Register A (DDRA)<br>See <a href="#">page 270</a> . | Read:  | DDRA7               | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
|        |  | Write: |                     |       |       |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| \$0005 | Data Direction Register B (DDRB)<br>See <a href="#">page 273</a> . | Read:  | DDRB7               | DDRB6 | DDRB5 | DDRB4 | DDRB3 | DDRB2 | DDRB1 | DDRB0 |
|        |  | Write: |                     |       |       |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 0     | 0     | 0     | 0     | 0     | 0     |
| \$0006 | Data Direction Register C (DDRC)<br>See <a href="#">page 275</a> . | Read:  | 0                   | 0     | 0     | DDRC4 | DDRC3 | DDRC2 | DDRC1 | DDRC0 |
|        |  | Write: |                     |       |       |       |       |       |       |       |
|        |  | Reset: | 0                   | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

 = Unimplemented

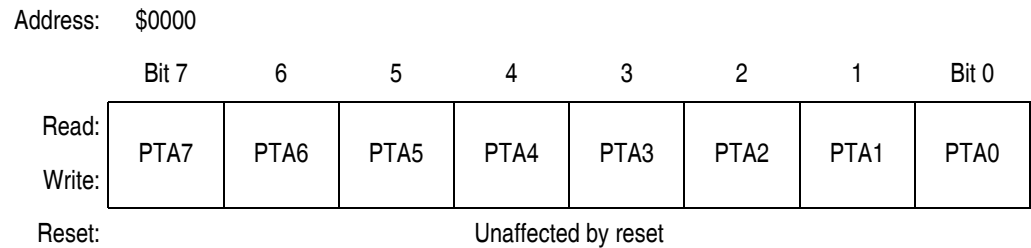
**Figure 18-1. I/O Port Register Summary**

## 18.3 Port A

Port A is an 8-bit, general-purpose, bidirectional I/O port.

### 18.3.1 Port A Data Register

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



**Figure 18-2. Port A Data Register (PTA)**

#### PTA7–PTA0 — 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.

## 18.3.2 Data Direction Register A

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

Address: \$0004

|        | Bit 7 | 6     | 5     | 4     | 3     | 2     | 1     | Bit 0 |
|--------|-------|-------|-------|-------|-------|-------|-------|-------|
| Read:  | DDRA7 | DDRA6 | DDRA5 | DDRA4 | DDRA3 | DDRA2 | DDRA1 | DDRA0 |
| Write: |       |       |       |       |       |       |       |       |
| Reset: | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

**Figure 18-3. Data Direction Register A (DDRA)**

### DDRA7–DDRA0 — Data Direction Register A Bits

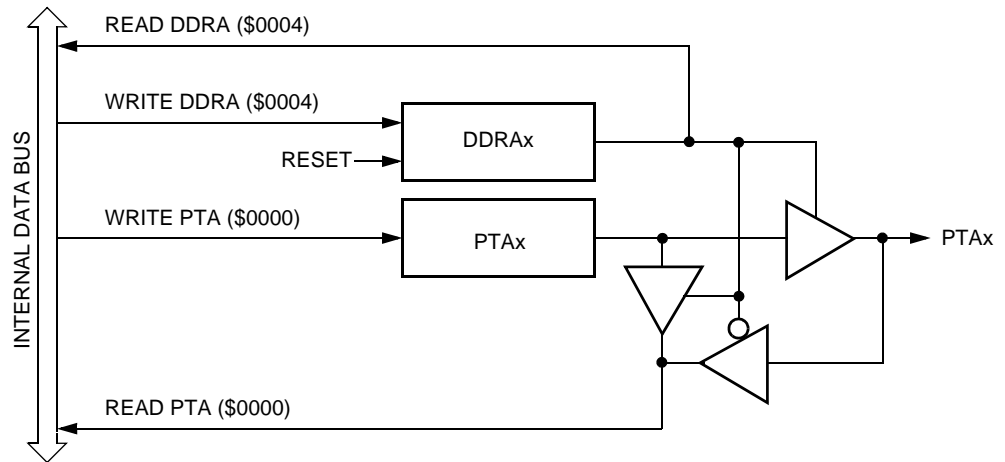
These read/write bits control port A data direction. Reset clears DDRA7–DDRA0, configuring all port A pins as inputs.

1 = Corresponding port A pin configured as output

0 = Corresponding port A pin configured as input

**NOTE:** *Avoid glitches on port A pins by writing to the port A data register before changing data direction register A bits from 0 to 1.*

**Figure 18-4** shows the port A I/O logic.



**Figure 18-4. 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 18-1](#) summarizes the operation of the port A pins.

**Table 18-1. Port A Pin Functions**

| DDRA Bit | PTA Bit          | I/O Pin Mode               | Accesses to DDRA | Accesses to PTA |                          |
|----------|------------------|----------------------------|------------------|-----------------|--------------------------|
|          |                  |                            | Read/Write       | Read            | Write                    |
| 0        | X <sup>(1)</sup> | Input, Hi-Z <sup>(2)</sup> | DDRA7–DDRA0      | Pin             | PTA7–PTA0 <sup>(3)</sup> |
| 1        | X                | Output                     | DDRA7–DDRA0      | PTA7–PTA0       | PTA7–PTA0                |

NOTES:

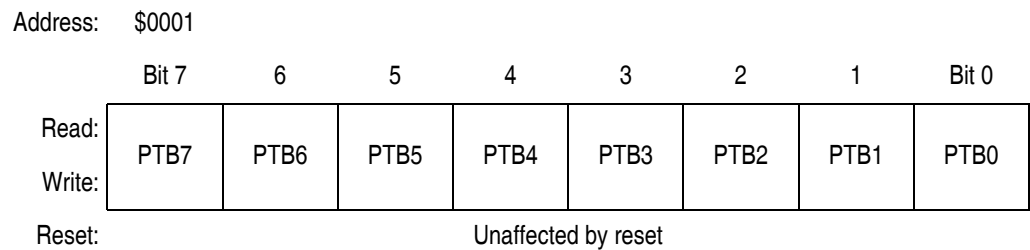
1. X = Don't care
2. Hi-Z = High impedance
3. Writing affects data register, but does not affect input.

## 18.4 Port B

Port B is an 8-bit, general-purpose, bidirectional I/O port.

### 18.4.1 Port B Data Register

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



**Figure 18-5. Port B Data Register (PTB)**

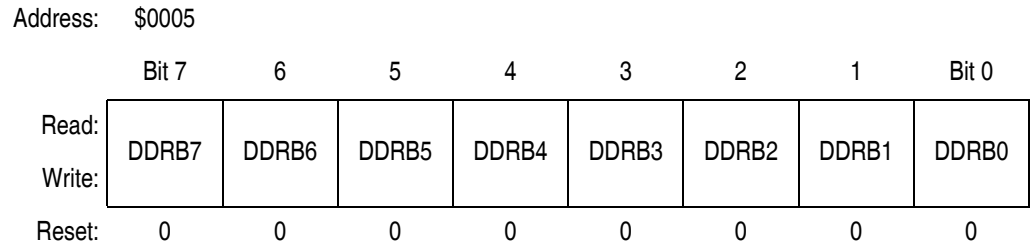
#### PTB7–PTB0 — Port B Data Bits

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.



## 18.4.2 Data Direction Register B

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



**Figure 18-6. Data Direction Register B (DDRB)**

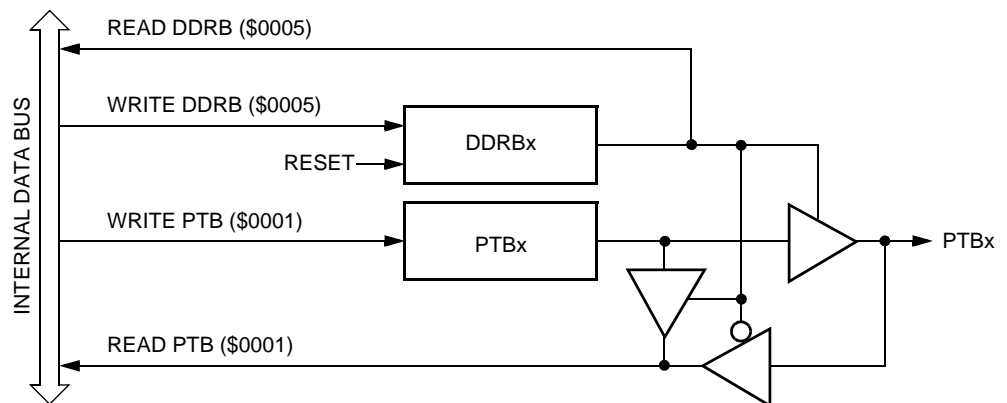
### DDRB7–DDRB0 — Data Direction Register B Bits

These read/write bits control port B data direction. Reset clears DDRB7–DDRB0, configuring all port B pins as inputs.

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

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

Figure 18-7 shows the port B I/O logic.



**Figure 18-7. 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 18-2** summarizes the operation of the port B pins.

**Table 18-2. Port B Pin Functions**

| DDRB Bit | PTB Bit          | I/O Pin Mode               | Accesses to DDRB | Accesses to PTB |                          |
|----------|------------------|----------------------------|------------------|-----------------|--------------------------|
|          |                  |                            | Read/Write       | Read            | Write                    |
| 0        | X <sup>(1)</sup> | Input, Hi-Z <sup>(2)</sup> | DDRB7–DDRB0      | Pin             | PTB7–PTB0 <sup>(3)</sup> |
| 1        | X                | Output                     | DDRB7–DDRB0      | PTB7–PTB0       | PTB7–PTB0                |

Notes:

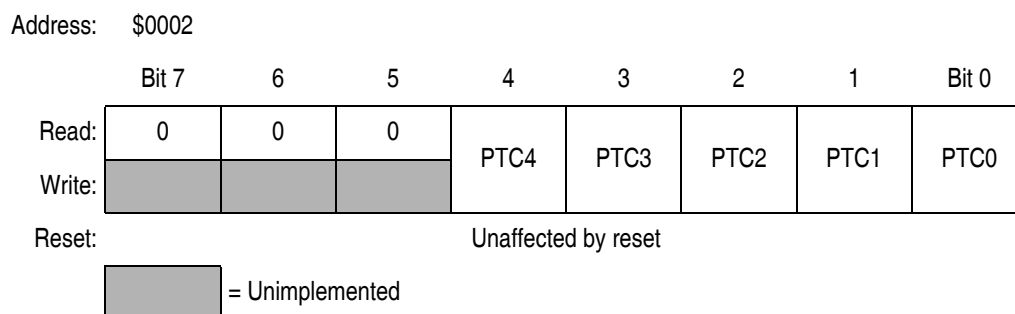
1. X = Don't care
2. Hi-Z = High impedance
3. Writing affects data register, but does not affect input.

## 18.5 Port C

Port C is a 5-bit, general-purpose, bidirectional I/O port.

### 18.5.1 Port C Data Register

The port C data register (PTC) contains a data latch for each of the five port C pins.



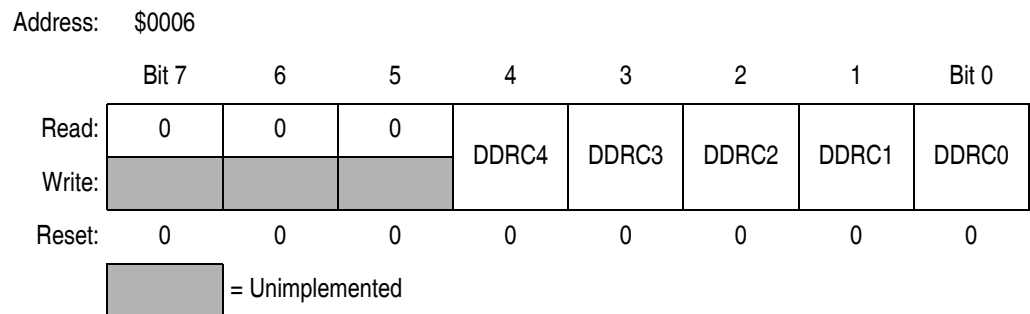
**Figure 18-8. Port C Data Register (PTC)**

### PTC4–PTC0 — 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.

## 18.5.2 Data Direction Register C

Data direction register C (DDRC) 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 18-9. Data Direction Register C (DDRC)**

### DDRC4–DDRC0 — Data Direction Register C Bits

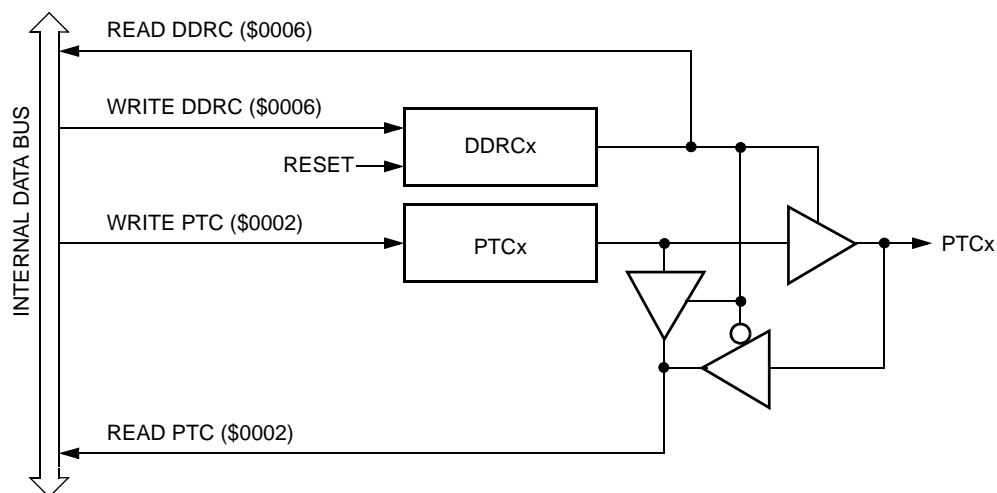
These read/write bits control port C data direction. Reset clears DDRC4–DDRC0, configuring all port C pins as inputs.

1 = Corresponding port C pin configured as output

0 = Corresponding port C pin configured as input

**NOTE:** *Avoid glitches on port C pins by writing to the port C data register before changing data direction register C bits from 0 to 1.*

**Figure 18-10** shows the port C I/O logic.



**Figure 18-10. 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 18-3](#) summarizes the operation of the port C pins.

**Table 18-3. Port C Pin Functions**

| DDRC Bit | PTC Bit          | I/O Pin Mode               | Accesses to DDRC | Accesses to PTC |                          |
|----------|------------------|----------------------------|------------------|-----------------|--------------------------|
|          |                  |                            | Read/Write       | Read            | Write                    |
| 0        | X <sup>(1)</sup> | Input, Hi-Z <sup>(2)</sup> | DDRC4–DDRC0      | Pin             | PTC4–PTC0 <sup>(3)</sup> |
| 1        | X                | Output                     | DDRC4–DDRC0      | PTC4–PTC0       | PTC4–PTC0                |

Notes:

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

## Section 19. Random-Access Memory (RAM)

### 19.1 Contents

|                                       |     |
|---------------------------------------|-----|
| 19.2 Introduction . . . . .           | 277 |
| 19.3 Functional Description . . . . . | 277 |

### 19.2 Introduction

This section describes the 602 bytes of RAM (random-access memory).

### 19.3 Functional Description

Addresses \$0050 through \$02AF are RAM locations. The location of the stack RAM is programmable. The 16-bit stack pointer allows the stack to be anywhere in the 64-Kbyte memory space.

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

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

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

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

## Random-Access Memory (RAM)

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

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

## Section 20. System Integration Module (SIM)

### 20.1 Contents

|          |  |     |
|----------|--|-----|
| 20.2     | Introduction . . . . .                           | 280 |
| 20.3     | SIM Bus Clock Control and Generation . . . . .   | 283 |
| 20.3.1   | Bus Timing . . . . .                             | 284 |
| 20.3.2   | Clock Startup from POR or LVI Reset . . . . .    | 284 |
| 20.3.3   | Clocks in Stop Mode and Wait Mode . . . . .      | 284 |
| 20.4     | Reset and System Initialization. . . . .         | 284 |
| 20.4.1   | External Pin Reset . . . . .                     | 285 |
| 20.4.2   | Active Resets from Internal Sources . . . . .    | 286 |
| 20.4.2.1 | Power-On Reset . . . . .                         | 287 |
| 20.4.2.2 | Computer Operating Properly (COP) Reset. . . . . | 288 |
| 20.4.2.3 | Illegal Opcode Reset . . . . .                   | 288 |
| 20.4.2.4 | Illegal Address Reset. . . . .                   | 289 |
| 20.4.2.5 | Low-Voltage Inhibit (LVI) Reset . . . . .        | 289 |
| 20.5     | SIM Counter . . . . .                            | 289 |
| 20.5.1   | SIM Counter During Power-On Reset . . . . .      | 289 |
| 20.5.2   | SIM Counter During Stop Mode Recovery . . . . .  | 290 |
| 20.5.3   | SIM Counter and Reset States. . . . .            | 290 |
| 20.6     | Exception Control . . . . .                      | 290 |
| 20.6.1   | Interrupts . . . . .                             | 291 |
| 20.6.1.1 | Hardware Interrupts . . . . .                    | 293 |
| 20.6.1.2 | SWI Instruction. . . . .                         | 294 |
| 20.6.1.3 | Interrupt Status Registers . . . . .             | 294 |
| 20.6.2   | Reset . . . . .                                  | 296 |
| 20.6.3   | Break Interrupts . . . . .                       | 296 |
| 20.6.4   | Status Flag Protection in Break Mode . . . . .   | 296 |
| 20.7     | Low-Power Modes . . . . .                        | 297 |
| 20.7.1   | Wait Mode . . . . .                              | 297 |
| 20.7.2   | Stop Mode . . . . .                              | 298 |

|        |                                 |     |
|--------|---------------------------------|-----|
| 20.8   | SIM Registers                   | 300 |
| 20.8.1 | SIM Break Status Register       | 300 |
| 20.8.2 | SIM Reset Status Register       | 301 |
| 20.8.3 | SIM Break Flag Control Register | 303 |

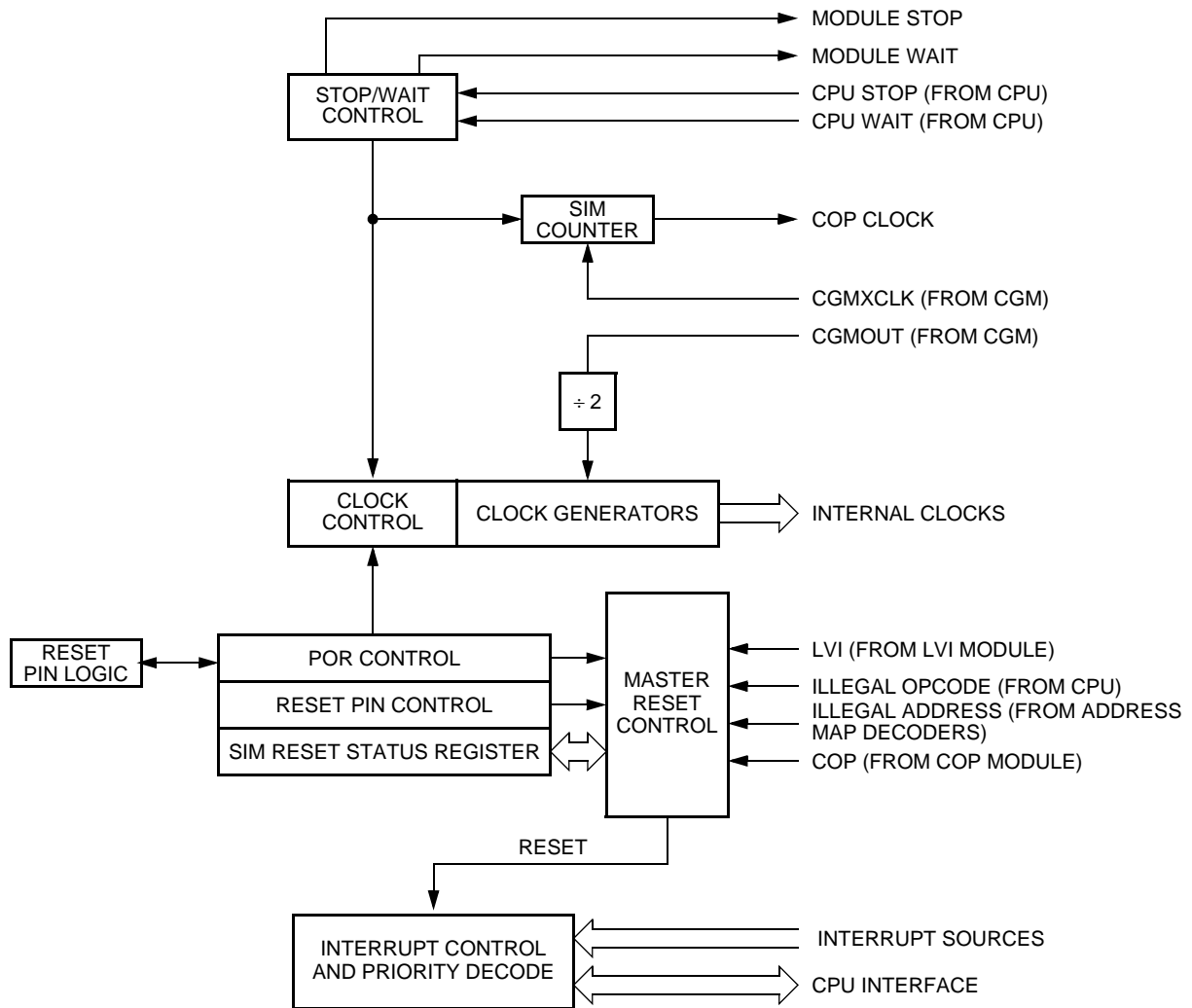
## 20.2 Introduction

This section describes the system integration module (SIM). Together with the CPU, the SIM controls all MCU activities. A block diagram of the SIM is shown in [Figure 20-1](#). [Table 20-1](#) is a summary of the SIM input/output (I/O) registers. The SIM is a system state controller that coordinates CPU and exception timing. The SIM is responsible for:

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

[Table 20-1](#) shows the internal signal names used in this section.





**Figure 20-1. SIM Block Diagram**

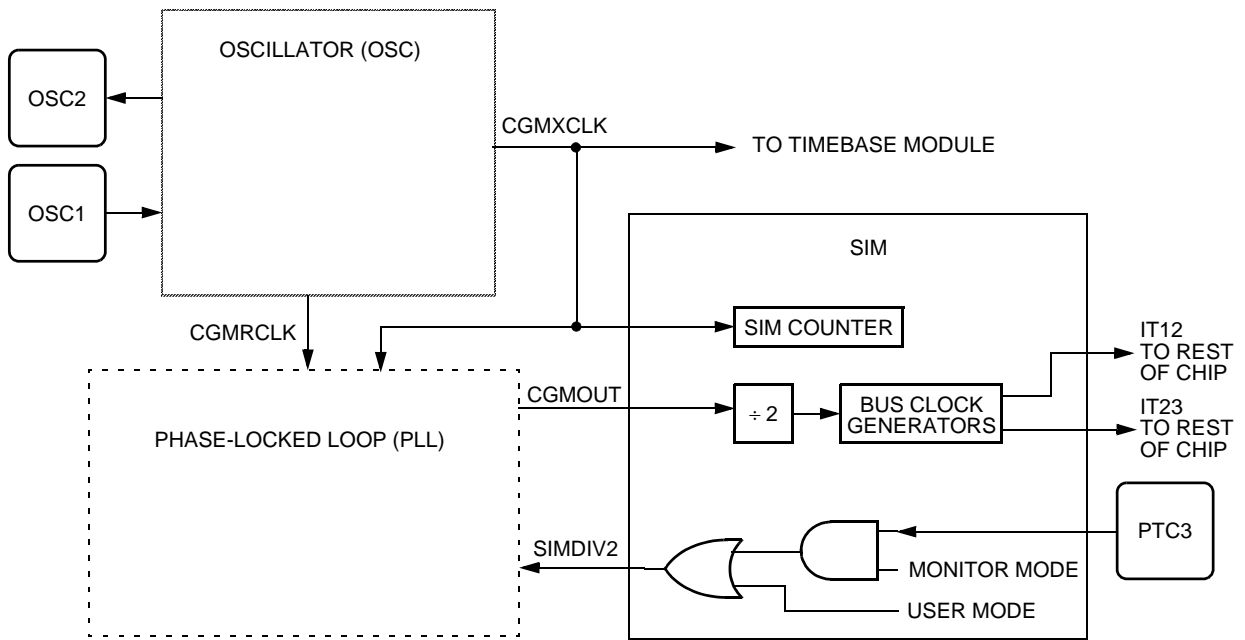
**Table 20-1. Signal Name Conventions**

| 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/ $\bar{W}$ | Read/write signal  |



## 20.3 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 20-3](#). This clock can come from either an external oscillator or from the on-chip PLL. (See [Section 7. Clock Generator Module \(CGMC\)](#).)



**Figure 20-3. CGM Clock Signals**

### 20.3.1 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 [Section 14. External Interrupt \(IRQ\)](#).

### 20.3.2 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 the 4096 CGMXCLK cycle POR timeout has completed. The  $\overline{\text{RST}}$  pin is driven low by the SIM during this entire period. The IBUS clocks start upon completion of the timeout.

### 20.3.3 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 [20.7.2 Stop Mode](#).)

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

## 20.4 Reset and System Initialization

The MCU has these reset sources:

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

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

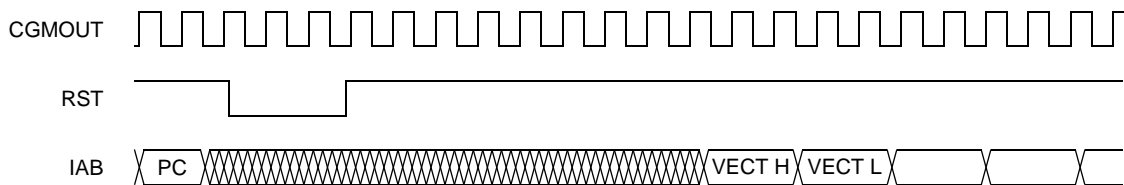
An internal reset clears the SIM counter (see [20.5 SIM Counter](#)), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). (See [20.8 SIM Registers](#).)

### 20.4.1 External Pin Reset

Pulling the asynchronous  $\overline{\text{RST}}$  pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as  $\overline{\text{RST}}$  is held low for a minimum of 67 CGMXCLK cycles, assuming that neither the POR nor the LVI was the source of the reset. See [Table 20-2](#) for details. [Figure 20-4](#) shows the relative timing.

**Table 20-2. PIN Bit Set Timing**

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

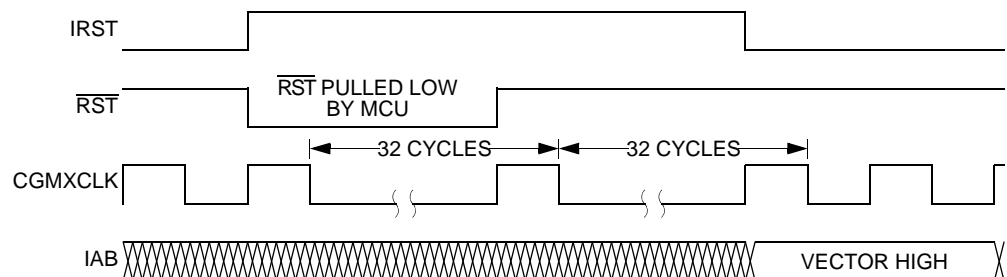


**Figure 20-4. External Reset Timing**

## 20.4.2 Active Resets from Internal Sources

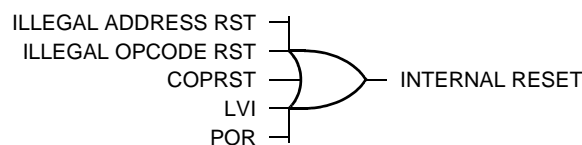
All internal reset sources actively pull the  $\overline{RST}$  pin low for 32 CGMXCLK cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles. See [Figure 20-5](#). An internal reset can be caused by an illegal address, illegal opcode, COP timeout, LVI, or POR. (See [Figure 20-6](#).)

**NOTE:** For LVI or POR resets, the SIM cycles through 4096 CGMXCLK cycles during which the SIM forces the  $\overline{RST}$  pin low. The internal reset signal then follows the sequence from the falling edge of  $\overline{RST}$  shown in [Figure 20-5](#).



**Figure 20-5. Internal Reset Timing**

The COP reset is asynchronous to the bus clock.



**Figure 20-6. Sources of Internal Reset**

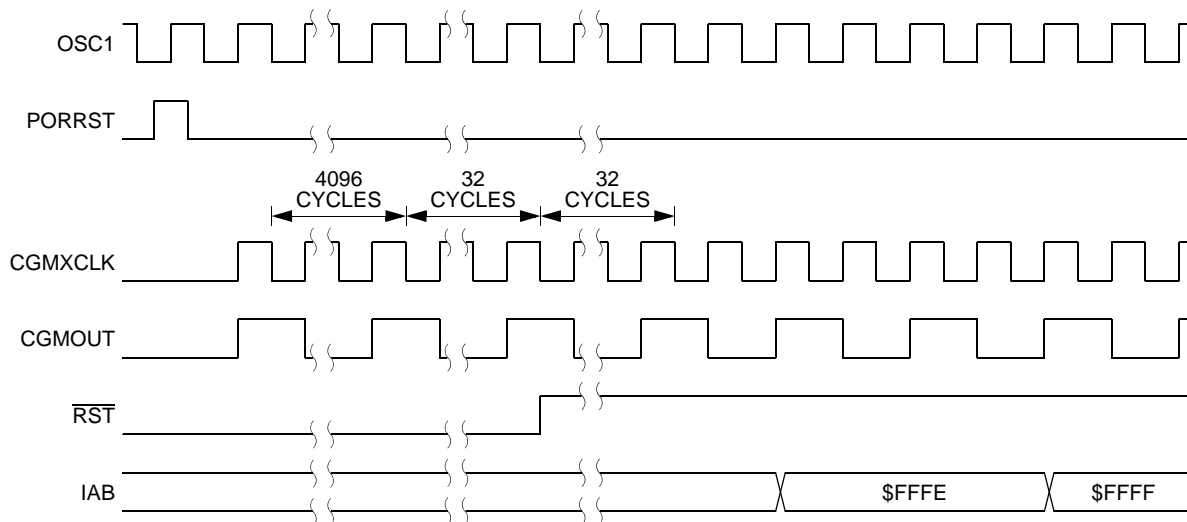
The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.

### 20.4.2.1 Power-On Reset

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

At power-on, these events occur:

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



**Figure 20-7. POR Recovery**

### 20.4.2.2 Computer Operating Properly (COP) Reset

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

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

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

### 20.4.2.3 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 configuration register is logic 0, the SIM treats the STOP instruction as an illegal opcode and causes an illegal opcode reset. The SIM actively pulls down the  $\overline{\text{RST}}$  pin for all internal reset sources.



#### 20.4.2.4 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.

#### 20.4.2.5 Low-Voltage Inhibit (LVI) Reset

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

## 20.5 SIM Counter

The SIM counter is used by the power-on reset module (POR) and in stop mode recovery to allow the oscillator time to stabilize before enabling the internal bus (IBUS) clocks. The SIM counter also serves as a prescaler for the computer operating properly module (COP). The SIM counter overflow supplies the clock for the COP module. The SIM counter is 13 bits long and is clocked by the falling edge of CGMXCLK.

### 20.5.1 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.

### 20.5.2 SIM Counter During Stop Mode Recovery

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

### 20.5.3 SIM Counter and Reset States

External reset has no effect on the SIM counter. (See [20.7.2 Stop Mode](#) for details.) The SIM counter is free-running after all reset states. (See [20.4.2 Active Resets from Internal Sources](#) for counter control and internal reset recovery sequences.)

## 20.6 Exception Control

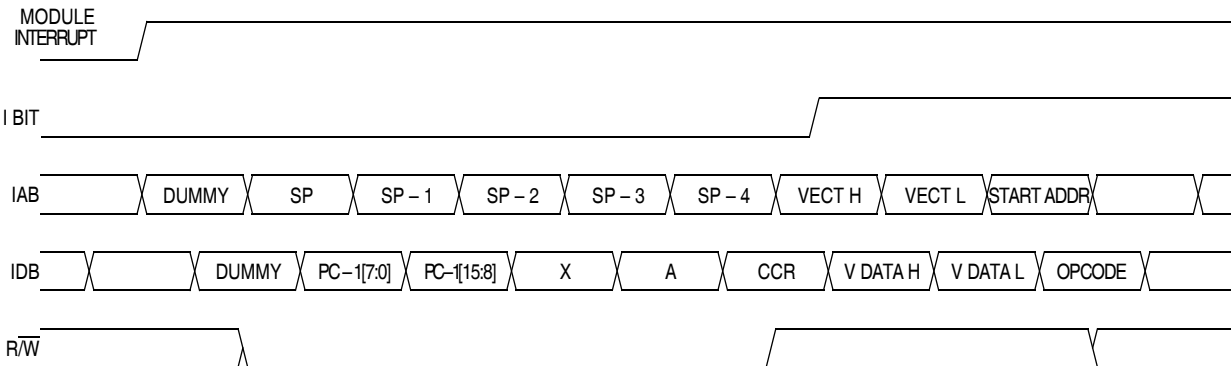
Normal, sequential program execution can be changed in three different ways:

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

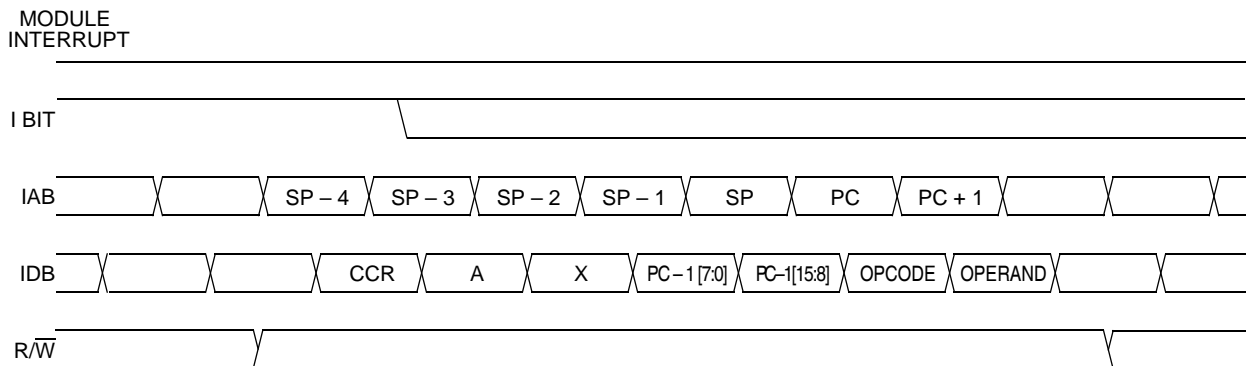
## 20.6.1 Interrupts

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 20-8](#) shows interrupt entry timing. [Figure 20-9](#) shows interrupt recovery timing.

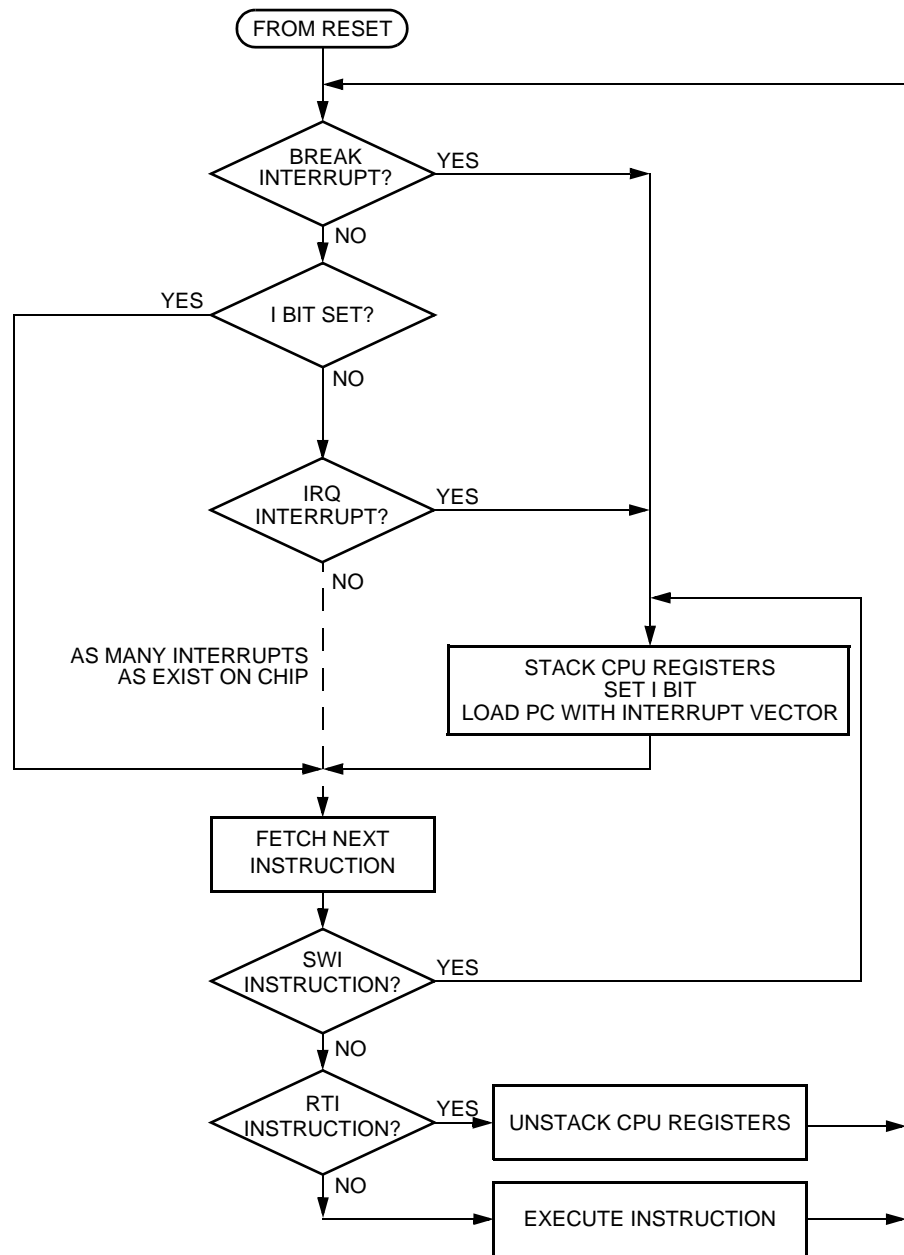
Interrupts are latched, and arbitration is performed in the SIM at the start of interrupt processing. The arbitration result is a constant that the CPU uses to determine which vector to fetch. Once an interrupt is latched by the SIM, no other interrupt can take precedence, regardless of priority, until the latched interrupt is serviced (or the I bit is cleared). (See [Figure 20-10](#).)



**Figure 20-8. Interrupt Entry Timing**



**Figure 20-9. Interrupt Recovery Timing**

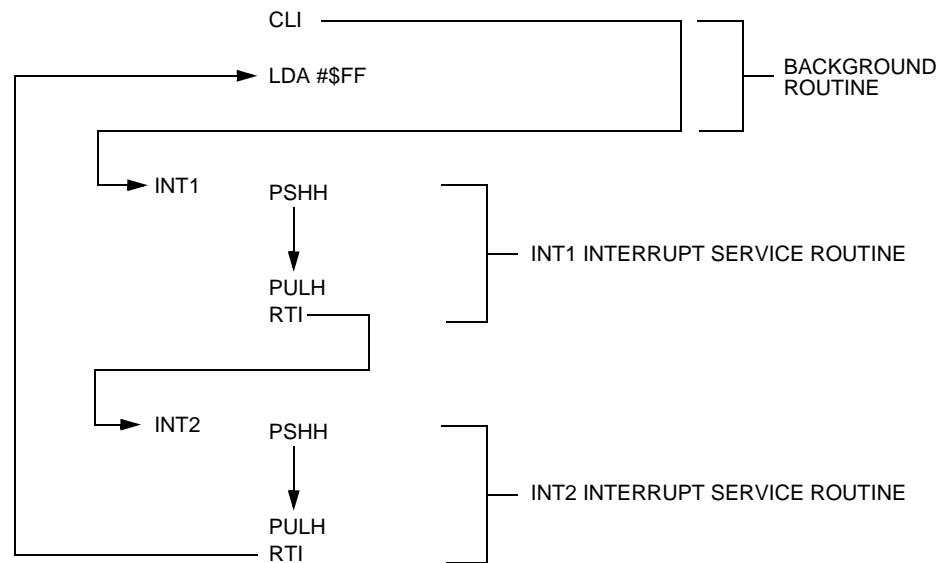


**Figure 20-10. Interrupt Processing**

### 20.6.1.1 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 20-11** 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 20-11. Interrupt Recognition Example**

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

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

## 20.6.1.2 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.

## 20.6.1.3 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. [Table 20-3](#) summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

**Table 20-3. Interrupt Sources**

| Priority | Interrupt Source            | Interrupt Status Register Flag |
|----------|-----------------------------|--------------------------------|
| Highest  | Reset                       | —                              |
|          | SWI instruction             | —                              |
|          | $\overline{\text{IRQ}}$ pin | I1                             |
|          | PLL                         | I2                             |
|          | Timebase module             | I3                             |
|          | TIM channel 0               | I4                             |
|          | TIM channel 1               | I5                             |
|          | TTIM overflow               | I6                             |
|          | OSD                         | I7                             |
|          | DSL                         | I8                             |
| Lowest   | SSI                         | I9                             |

*Interrupt Status Register 1*

Address: \$FE04

|        | Bit 7 | 6  | 5  | 4  | 3  | 2  | 1 | Bit 0 |
|--------|-------|----|----|----|----|----|---|-------|
| Read:  | I6    | I5 | I4 | I3 | I2 | I1 | 0 | 0     |
| Write: | R     | R  | R  | R  | R  | R  | R | R     |
| Reset: | 0     | 0  | 0  | 0  | 0  | 0  | 0 | 0     |

R = Reserved

**Figure 20-12. Interrupt Status Register 1 (INT1)**

**I6–I1 — Interrupt Flags 1–6**

These flags indicate the presence of interrupt requests from the sources shown in [Table 20-3](#).

- 1 = Interrupt request present
- 0 = No interrupt request present

**Bit 0 and Bit 1 — Always read 0**

*Interrupt Status Register 2*

Address: \$FE05

|        | Bit 7 | 6 | 5 | 4 | 3 | 2  | 1  | Bit 0 |
|--------|-------|---|---|---|---|----|----|-------|
| Read:  | 0     | 0 | 0 | 0 | 0 | I9 | I8 | I7    |
| Write: | R     | R | R | R | R | R  | R  | R     |
| Reset: | 0     | 0 | 0 | 0 | 0 | 0  | 0  | 0     |

R = Reserved

**Figure 20-13. Interrupt Status Register 2 (INT2)**

**I9–I7 — Interrupt Flags 9–7**

These flags indicate the presence of interrupt requests from the sources shown in [Table 20-3](#).

- 1 = Interrupt request present
- 0 = No interrupt request present

### 20.6.2 Reset

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

### 20.6.3 Break Interrupts

The break module can stop normal program flow at a software-programmable break point by asserting its break interrupt output. See [Section 6. 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.

### 20.6.4 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 2-step clearing mechanism — for example, a read of one register followed by the read or write of another — are protected, even when the first step is accomplished prior to entering break mode. Upon leaving break mode, execution of the second step will clear the flag as normal.



## 20.7 Low-Power Modes

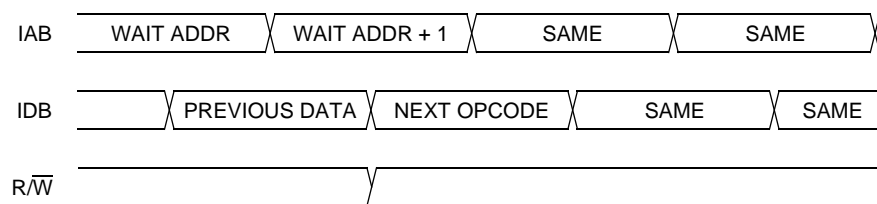
Executing the WAIT or STOP instruction puts the MCU in a low power-consumption mode for standby situations. The SIM holds the CPU in a non-clocked state. The operation of each of these modes is described in the following subsections. Both STOP and WAIT clear the interrupt mask (I) in the condition code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

### 20.7.1 Wait Mode

In wait mode, the CPU clocks are inactive while the peripheral clocks continue to run. **Figure 20-14** shows the timing for wait mode entry.

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

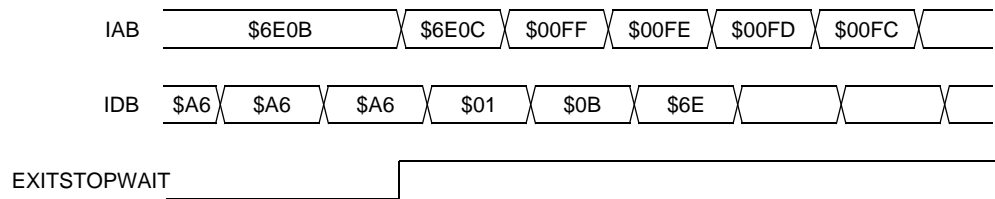
Wait mode also can be exited by a reset or break. A break interrupt during wait mode sets the SIM break stop/wait bit, SBSW, in the SIM break status register (SBSR). If the COP disable bit, COPD, in the configuration register is logic 0, then the computer operating properly module (COP) is enabled and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



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

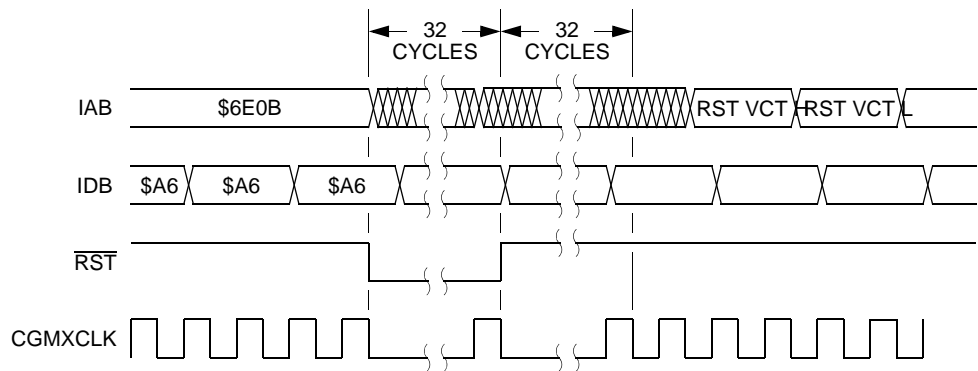
**Figure 20-14. Wait Mode Entry Timing**

**Figure 20-15** and **Figure 20-16** show the timing for WAIT recovery.



Note: EXITSTOPWAIT =  $\overline{\text{RST}}$  pin, CPU interrupt, or break interrupt

**Figure 20-15. Wait Recovery from Interrupt or Break**



**Figure 20-16. Wait Recovery from Internal Reset**

## 20.7.2 Stop Mode

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

The SIM disables the clock generator module output (CGMOUT) in stop mode, stopping the CPU and peripherals<sup>1</sup>. Stop recovery time is selectable using the SSREC bit in the configuration register (CONFIG). 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.

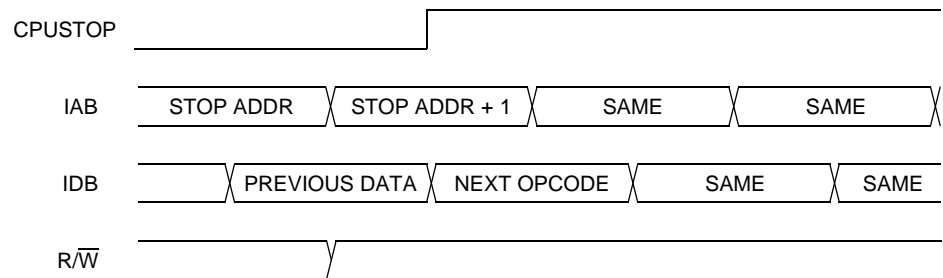
**NOTE:** *External crystal applications should use the full stop recovery time by clearing the SSREC bit.*

1. Clock CGMXCLK is not stopped. It continues to feed the timebase module.

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

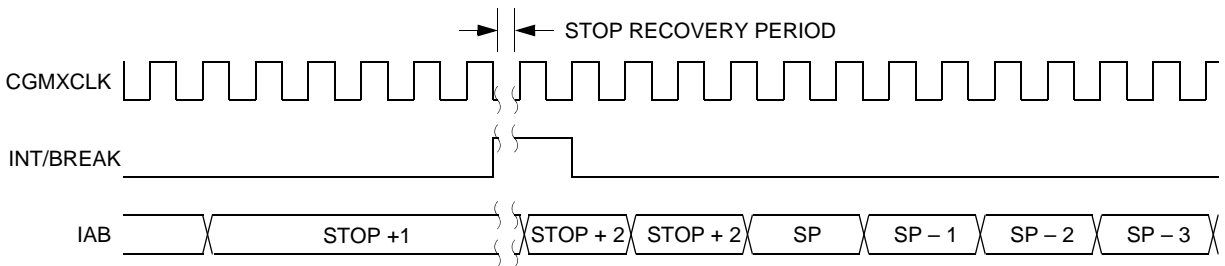
The SIM counter is held in reset from the execution of the STOP instruction until the beginning of stop recovery. It is then used to time the recovery period. **Figure 20-17** shows stop mode entry timing.

**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 20-17. Stop Mode Entry Timing**



**Figure 20-18. Stop Mode Recovery from Interrupt or Break**

## 20.8 SIM Registers

The SIM has three memory-mapped registers. [Table 20-4](#) shows the mapping of these registers.

**Table 20-4. SIM Registers**

| Address | Register | Access Mode |
|---------|----------|-------------|
| \$FE00  | SBSR     | User        |
| \$FE01  | SRSR     | User        |
| \$FE03  | SBFCR    | User        |

### 20.8.1 SIM Break Status Register

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

Address: \$FE00

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

|   |
|---|
| R |
|---|

 = Reserved

Note: Writing a logic 0 clears SBSW.

**Figure 20-19. SIM Break Status Register (SBSR)**

#### SBSW — SIM Break Stop/Wait Bit

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

1 = Stop mode or wait mode was exited by break interrupt.

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

SBSW can be read within the break state SWI routine. The user can modify the return address on the stack by subtracting one from it. The following code is an example of this. Writing 0 to the SBSW bit clears it.

```

;This code works if the H register has been pushed onto the stack in the break
;service routine software. This code should be executed at the end of the break
;service routine software.

HIBYTE EQU 5
LOBYTE EQU 6

; If not SBSW, do RTI

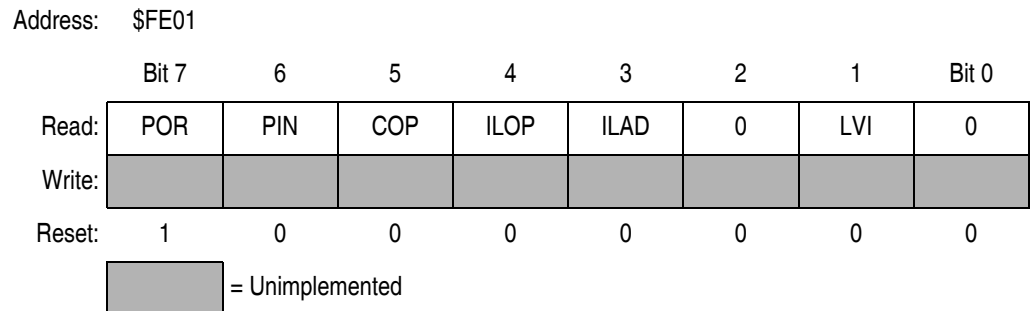
BRCLR SBSW,SBSR, RETURN ;See if wait mode or stop mode was exited by
;break.

TST LOBYTE,SP ;If RETURNLO is not zero,
BNE DOLO ;then just decrement low byte.
DEC HIBYTE,SP ;Else deal with high byte, too.
DOLO DEC LOBYTE,SP ;Point to WAIT/STOP opcode.
RETURN PULH ;Restore H register.
RTI

```

### 20.8.2 SIM Reset Status Register

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



**Figure 20-20. 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

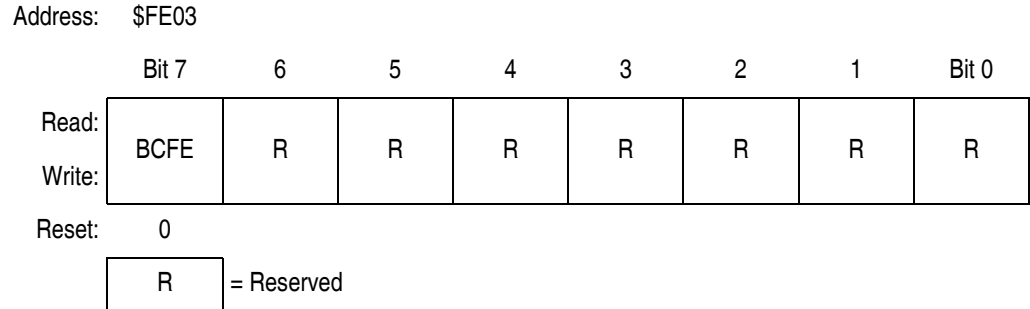
LVI — Low-Voltage Inhibit Reset Bit

1 = Last reset caused by the LVI circuit

0 = POR or read of SRSR

### 20.8.3 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 20-21. 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





## Section 21. Serial Synchronous Interface Module (SSI)

### 21.1 Contents

|        |                            |     |
|--------|----------------------------|-----|
| 21.2   | Introduction               | 306 |
| 21.3   | Features                   | 306 |
| 21.4   | Overview                   | 307 |
| 21.4.1 | START Signal               | 308 |
| 21.4.2 | Slave Address Transmission | 309 |
| 21.4.3 | Data Transfer              | 309 |
| 21.4.4 | STOP Signal                | 310 |
| 21.4.5 | Clock Synchronization      | 310 |
| 21.4.6 | Handshaking                | 310 |
| 21.4.7 | Clock Stretching           | 311 |
| 21.5   | Programming Guidelines     | 311 |
| 21.5.1 | Setup                      | 311 |
| 21.5.2 | Interrupt Serving          | 312 |
| 21.6   | Registers                  | 313 |
| 21.6.1 | SSI Control Register       | 313 |
| 21.6.2 | SSI Status Register        | 315 |
| 21.6.3 | SSI Data Register          | 316 |
| 21.7   | Low-Power Modes            | 317 |
| 21.8   | Interrupt                  | 317 |

## 21.2 Introduction

The serial synchronous interface module (SSI) transmits and receives synchronous serial data for communication with other peripherals. It offers I<sup>2</sup>C master compatibility, taking care of START and STOP signal generation and optionally producing byte-by-byte interrupts. The interface has two pairs of clock and data lines that can not operate simultaneously. In that way, sensitive devices can be isolated from each other by putting them in separate clock and data lines.

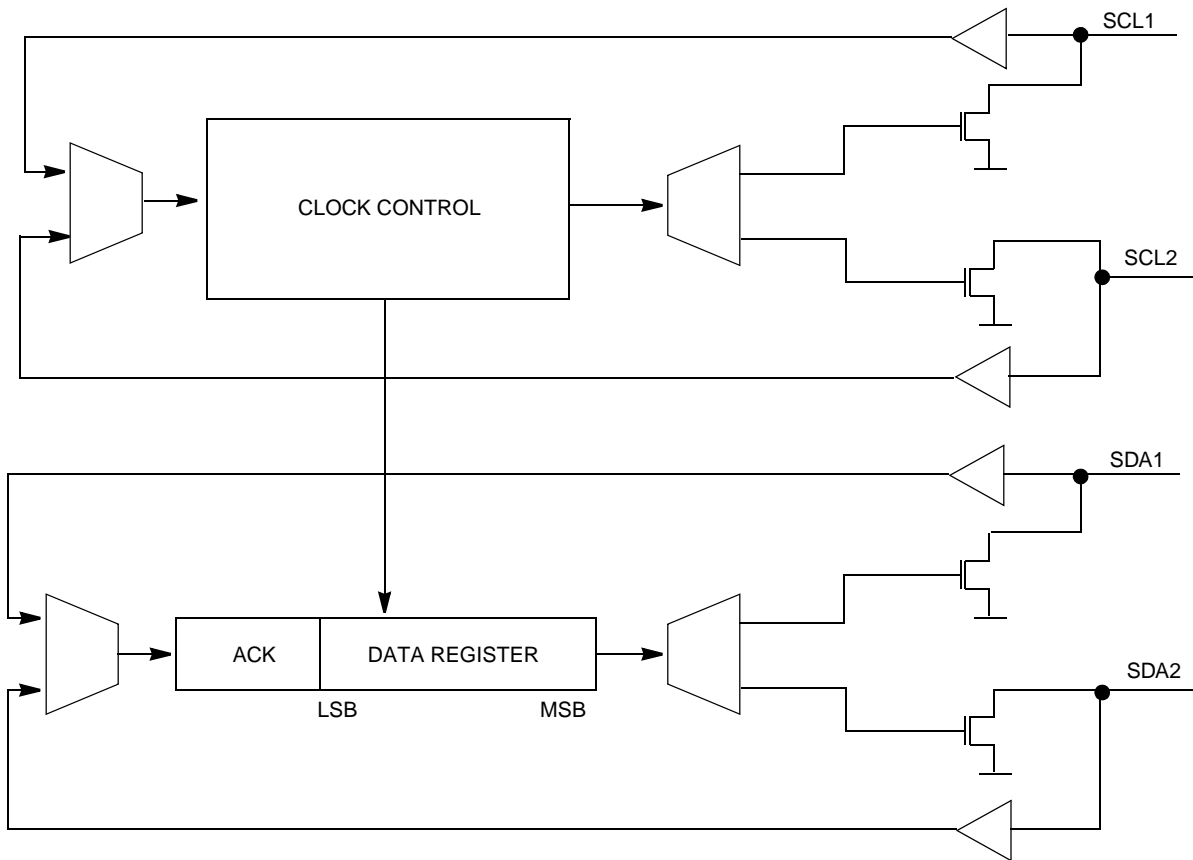
## 21.3 Features

The SSI has these key features:

- Limited master compatibility with the I<sup>2</sup>C bus standard
- High interference immunity between the clock and data lines
- Software programmable clock frequency
- Software controlled acknowledge bit generation/detection
- Start and stop signal generation
- Interrupt driven byte-by-byte data transfer
- Repeated start signal generation
- Clock stretching to allow operation of slow devices

## 21.4 Overview

The SSI uses two serial data lines (SDA1 and SDA2) and two serial clock lines (SCL1 and SCL2) for data transfer. All devices connected in these lines must have open-drain or open-collector outputs. Logic “AND” function is exercised on all four lines with external pullup resistors. Only one pair of clock and data lines is allowed to operate at any time. They share the same internal serial-to-parallel converter (see [Figure 21-1](#)).



**Figure 21-1. SSI Block Diagram**

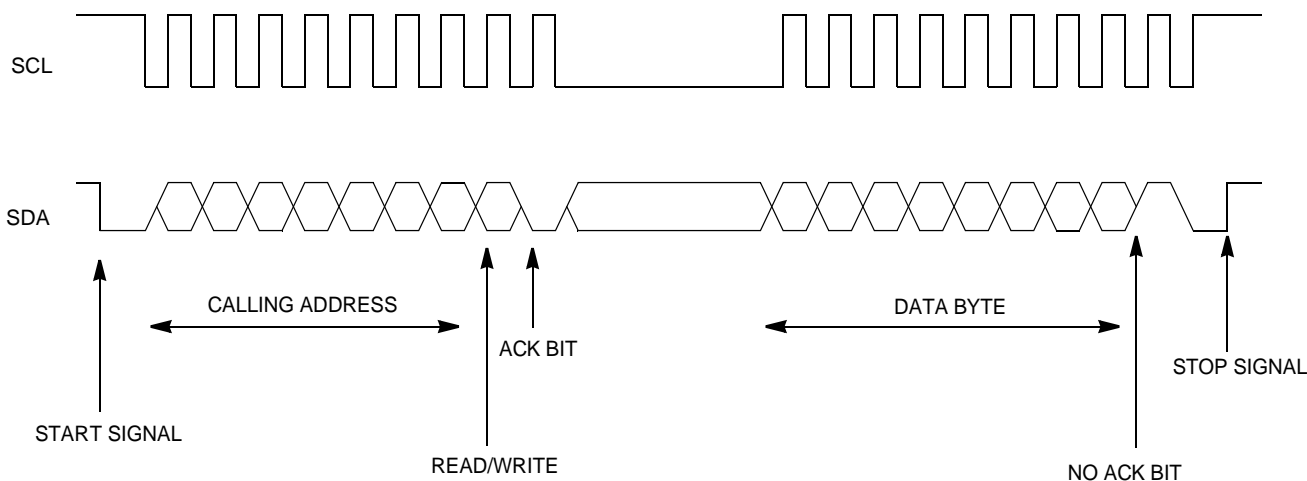
## Serial Synchronous Interface Module (SSI)

The SSI can operate in master mode only, but it can send or receive data from peripherals. Normally, a standard communication is composed of four parts: START signal, slave address transmission, data transfer and STOP signal. The START and STOP signals are generated by hardware under software control. Slave address transmission is viewed here as a normal data transmission. The data byte and the acknowledge-bit must be written or read by software. An interrupt is optionally generated on a byte-by-byte basis to help data transfer.

### 21.4.1 START Signal

For sake of simplicity, the following text refers to either SCL1 or SCL2 as just SCL, and SDA1 or SDA2 as just SDA.

To initiate a communication, the CPU will program the control and the data registers and then the SSI will send a START signal. As shown in [Figure 21-2](#), a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and wakes up all slaves.



**Figure 21-2. Transmission Signal**

## 21.4.2 Slave Address Transmission

The first byte of data transfer immediately after the START signal is the slave address transmitted by the master. This is a 7-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer:

- 1 = Read transfer, the slave transmits data to the master
- 0 = Write transfer, the master transmits data to the slave

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see [Figure 21-2](#)). No two slaves in the system may have the same address.

## 21.4.3 Data Transfer

Once slave addressing is achieved successfully, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit. All transfers that come after an address cycle are referred to as data transfers.

Each data byte is eight bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in [Figure 21-2](#). There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte has to be followed by an acknowledge bit, which is signaled from the receiving device by pulling the SDA low at the ninth clock. So one complete data byte transfer needs nine clock pulses.

If the peripheral does not acknowledge the received byte, the SDA line is left high during the acknowledge bit interval. The SSI can then generate a stop signal at the end of the next data transfer or a start signal (repeated start) to commence a new calling.

If the SSI is operating as a receiver and it does not acknowledge the slave transmitter after a byte transmission, it means “end of data” to the slave. So, the slave releases the SDA line for the SSI to generate a STOP or START signal.

### 21.4.4 STOP Signal

The SSI module can terminate the communication by generating a STOP signal that will free the bus at the end of the current byte transfer. However, the SSI may generate another START signal without generating a STOP signal first. This is called repeated START. A STOP signal is defined as a low-to-high transition of SDA while SCL is at logical 1 (see [Figure 21-2](#)).

The SSI module can generate a STOP even if the peripheral has generated an acknowledge at which point the peripheral must release the bus.

### 21.4.5 Clock Synchronization

Since wired-and logic is performed on the SCL line, peripherals can also drive the serial clock signal. A high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and once a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time. When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high.

### 21.4.6 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (nine bits). In such case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

### 21.4.7 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low, the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period, then the resulting SCL bus signal low period is stretched.

## 21.5 Programming Guidelines

Two modes of operation are possible:

- Master transmitter
- Master receiver

In both modes, the programming procedure is the same for the first byte, because the slave address is sent. After the first byte, the procedure is a little different in each case.

### 21.5.1 Setup

On both transmitter and receiver modes, a setup involves these steps:

1. Enable the SSI and the interrupts (if desired) in the control register.
2. Program the desired clock rate in the control register.
3. Set the START bit, reset the STOP bit, and set the ACK bit so that it will lose any wired-and dispute.
4. Write the slave address and the R/W bit into the data register. Since the SF flag is clear at reset, a write to the data register triggers the shifting of the data into the bus.

### 21.5.2 Interrupt Serving

The internal shift register is composed of the data register and the ACK bit, as shown in **Figure 21-1**. As the data is shifted out (MSB first), the signal on the data line, resulting from the wired-and logic, will be shifted back to the register via least significant bit. At the end of the transfer, the contents of the data register and the ACK bit of the control register reflect what was present in the external bus. In that way the CPU can read the data register and the ACK bit to find out the contents of the bus during the transfer.

After the ninth SCL falling edge, an interrupt will be generated (if enabled) and SCL will stop until the interrupt is served by reading the status register and then writing or reading the data register.

In transmitter mode, upon receiving an interrupt, the CPU has to perform these tasks:

1. Read the status register.
2. Optionally, read the ACK bit to see if the peripheral has acknowledged the transmission.
3. Optionally, read the data register to see if there was any contention on the bus, in which case the contents of the register would have changed. This operation clears the SF flag.
4. Write into the data register the next byte to be transmitted. This operation also clears the SF flag and initiates the data shifting.

In receiver mode, upon receiving an interruption, the CPU has to perform these tasks:

1. Read the status register. This operation clears the SF flag and the interrupt.
2. Read the desired data in the data register.
3. Write "0" into the ACK bit of the control register to acknowledge the reception of the next byte.
4. Write \$FF into the data register. This operation initiates the transfer of the next byte.



## 21.6 Registers

Three registers are used in the SSI module. The internal configuration of these registers is discussed here.

### 21.6.1 SSI Control Register

The SSI control register (SSICR) provides eight control bits.

Address: \$004C

|        | Bit 7 | 6  | 5     | 4    | 3   | 2    | 1   | Bit 0 |
|--------|-------|----|-------|------|-----|------|-----|-------|
| Read:  | SIE   | SE | START | STOP | ACK | SCHS | SR1 | SR0   |
| Write: |       |    |       |      |     |      |     |       |
| Reset: | 0     | 0  | 0     | 0    | 0   | 0    | 0   | 0     |

**Figure 21-3. SSI Control Register (SSICR)**

#### SIE — SSI Interrupt Enable Bit

This bit determines if the SF bit in the status register is allowed to generate interrupt requests to the CPU.

- 1 = Interrupts from the SSI module are enabled
- 0 = Interrupts from the SSI module are disabled

#### SE — SSI Enable Bit

This bit controls whether the SSI is enabled or disabled. When the SSI is disabled, the internal counters are reset, the SCL clock stops and any transmission in progress is aborted. Also, the status register is cleared and the SCL and SDA pins goes to high impedance.

- 1 = The SSI module is enabled. This bit must be set before any other bits of this register have any effect.
- 0 = The module is disabled. This is the power-on reset situation. When low, the interface is held in reset but registers can still be accessed.

### START — Start Bit

This bit determines if the following byte is going to be preceded by a START signal. This bit is set to initiate a transmission or to begin reading data from peripherals. It also can be set in the middle of a transmission order to specify a repeated START condition.

1 = The following byte will be preceded by a START signal.

0 = A START signal will not be produced.

### STOP — STOP Bit

This bit determines if the next byte is going to be followed by a STOP signal. This bit is set to finish a transmission or to finish reading data from peripherals.

1 = The next byte will be followed by a STOP signal.

0 = A STOP signal will not be produced.

### ACK — Acknowledge Bit

When receiving data, the CPU must write a logic 0 into this bit so that an acknowledge is generated to the peripheral. When transmitting data, the CPU must write a logic 1 to this bit and then read it later after the transmission has been completed to know if the peripheral has acknowledged the transfer.

### SCHS — SSI Channel Select Bit

This bit controls the multiplexing of the clock and data lines to the external pins.

1 = SCL2 and SDA2 are activated to carry the SSI signals; SCL1 and SDA1 goes to high-impedance.

0 = SCL1 and SDA1 are activated to carry the SSI signals; SCL2 and SDA2 goes to high-impedance.

**NOTE:** *It is strongly recommended that the last transfer is properly terminated by a STOP bit before changing SCHS.*

### SR1 and SR0 — SSI Clock Rate

These bits determine the serial clock frequency as shown in [Table 21-1](#)


**Table 21-1. SCL Rates (Hz) at Bus Frequency**

| SR1 and SR0 | 2.0 MHz  | 4.0 MHz | 8.0 MHz |
|-------------|----------|---------|---------|
| 00          | 15.625 k | 31.25 k | 62.5 k  |
| 01          | 31.25 k  | 62.5 k  | 125 k   |
| 10          | 62.5 k   | 125 k   | 250 k   |
| 11          | 125 k    | 250 k   | 500 k   |

### 21.6.2 SSI Status Register

Address: \$004D

|        | Bit 7 | 6    | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|--------|-------|------|---|---|---|---|---|-------|
| Read:  | SF    | DCOL | 0 | 0 | 0 | 0 | 0 | 0     |
| Write: |       |      |   |   |   |   |   |       |
| Reset: | 0     | 0    | 0 | 0 | 0 | 0 | 0 | 0     |

 = Unimplemented

**Figure 21-4. SSI Status Register (SSISR)**

#### SF — SSI Flag Bit

This read-only bit is set after the occurrence of the ninth falling clock edge and indicates that a complete transfer has occurred. If the SIE bit is set, an interrupt will also be generated. The SCL clock line will stop until the SF flag is cleared, indicating that the CPU has prepared a new byte to transfer or has read the received byte. The SF flag must always be cleared between transfers, which can be done in three ways:

- By reading the status register with SF set, followed by writing or reading the data register
- By a reset
- By disabling the SSI

**NOTE:** *If the SF flag is cleared by resetting or disabling the SSI before issuing a STOP bit, the slave device may enter into an indeterminate state. The first method of clearing SF is always the best.*

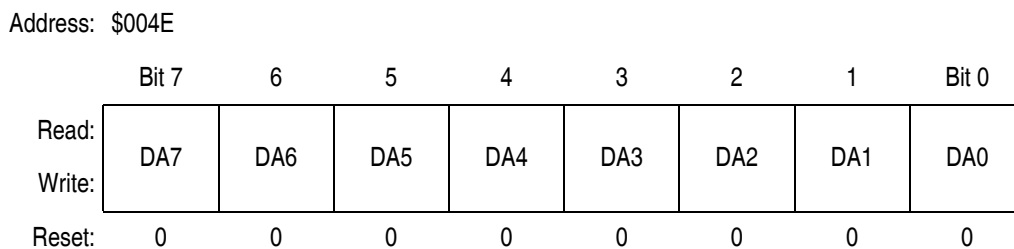
## DCOL — Data Collision Bit

This is a read-only status bit that indicates an invalid access to the data has been made. An invalid access is:

- An access to the data register in the middle of a transfer, after the first falling edge of SCL and before SF is set
- An access to the data register before an access to the status register, after SF is set

DCOL is cleared by reading the status register with SF set, followed by a read or write of the data register.

## 21.6.3 SSI Data Register



**Figure 21-5. SSI Data Register (SSIDR)**

## DA7–DA0 — SSI Data Bits

These bits are the eight data bits that have been received or are to be transferred by the SSI. These bits are not double-buffered but writes to this register are masked during transfers and will not destroy the previous contents. A transfer is triggered by a read to the status register (if SF is set) followed by a write to the data register. During a transfer, the contents of the data register plus the ACK bit are shifted (MSB first) to the data line (see [Figure 21-1](#)). The signal on the data line, resulting from the wired-and logic, will be shifted back to the register via least significant bit. At the end of the transfer, the contents of the data register and the ACK bit of the control register reflect what

was present in the external bus. In that way, the CPU can read the data register and the ACK bit to find out the contents of the bus during the transfer.

## 21.7 Low-Power Modes

In wait mode or stop mode, the SSI halts operation. Pins SDA1, SCL1, SDA2, and SCL2 will maintain their states.

If the SSI is nearing completion of a transfer when wait mode or stop mode is entered, it is possible for the SSI to generate an interrupt request and thus cause the processor to exit wait mode or stop mode immediately. To prevent this occurrence, the programmer should ensure that all transfers are complete before entering wait mode or stop mode.

## 21.8 Interrupt

The SSI has one source of interrupt, the SF flag in the SSI status register. This interrupt is enabled by bit SIE in the SSI control register. This interrupt will cause the CPU to vector to the address stored in \$FFEA–\$FFEB.



## Section 22. Timebase Module (TBM)

### 22.1 Contents

|      |   |     |
|------|---|-----|
| 22.2 | Introduction . . . . .                  | 319 |
| 22.3 | Features . . . . .                      | 319 |
| 22.4 | Functional Description . . . . .        | 320 |
| 22.5 | Timebase Register Description . . . . . | 321 |
| 22.6 | Interrupts . . . . .                    | 322 |
| 22.7 | Low-Power Modes . . . . .               | 323 |

### 22.2 Introduction

This section describes the timebase module (TBM). The TBM will generate periodic interrupts at user selectable rates using a counter clocked by the external crystal clock. This TBM version uses 15 divider stages, eight of which are user selectable.

### 22.3 Features

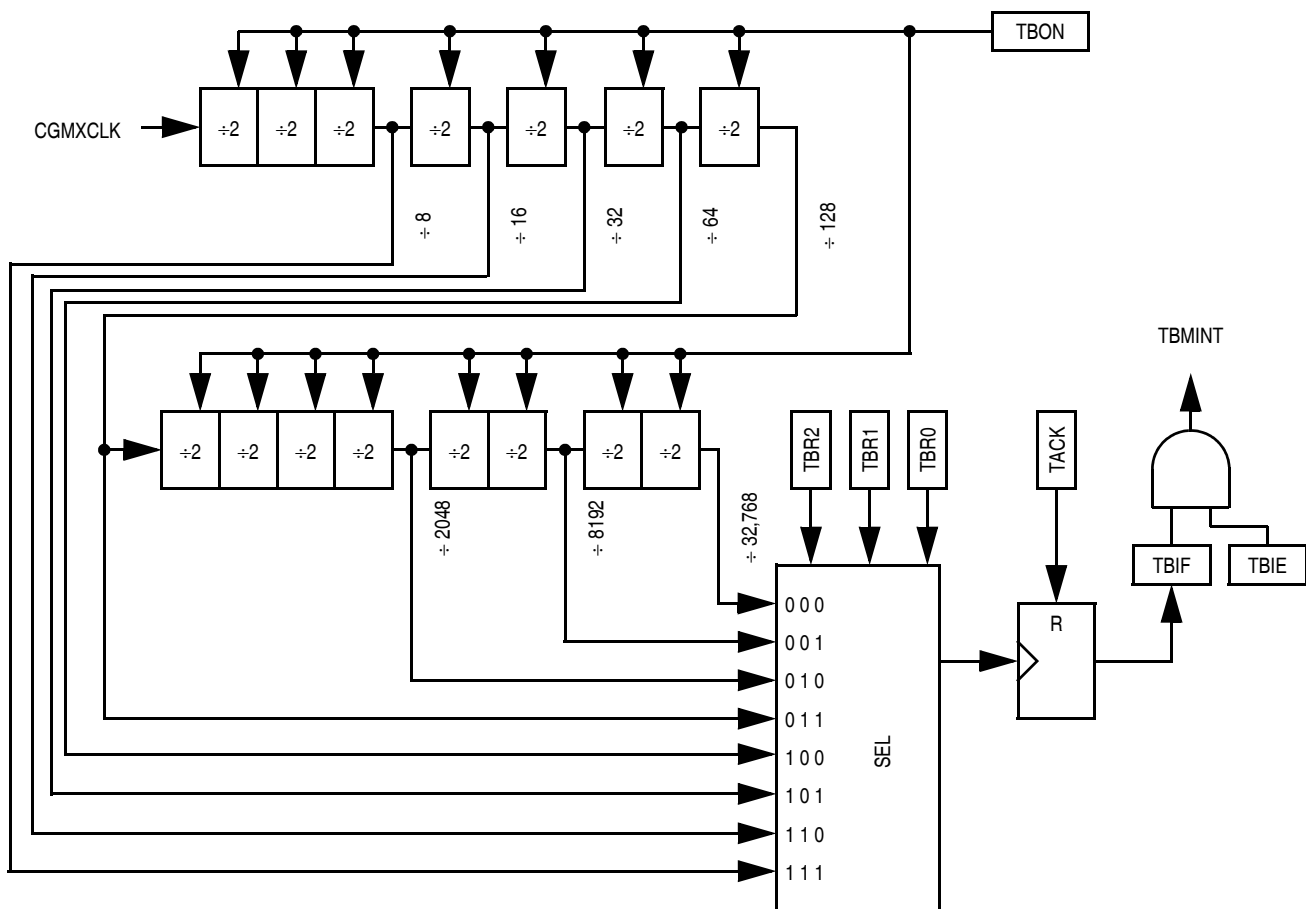
Features of the TBM module include:

- Software programmable 1-Hz, 4-Hz, 16-Hz, 256-Hz, 512-Hz, 1024-Hz, 2048-Hz, and 4096-Hz periodic interrupt using external 32.768-kHz crystal

## 22.4 Functional Description

**NOTE:** This module is designed for a 32.768-kHz oscillator.

This module can generate a periodic interrupt by dividing the crystal frequency, CGMXCLK. The counter is initialized to all 0s when TBON bit is cleared. The counter, shown in **Figure 22-1**, starts counting when the TBON bit is set. When the counter overflows at the tap selected by TBR2:TBR0, the TBIF bit gets set. If the TBIE bit is set, an interrupt request is sent to the CPU. The TBIF flag is cleared by writing a 1 to the TACK bit. The first time the TBIF flag is set after enabling the timebase module, the interrupt is generated at approximately half of the overflow period. Subsequent events occur at the exact period.



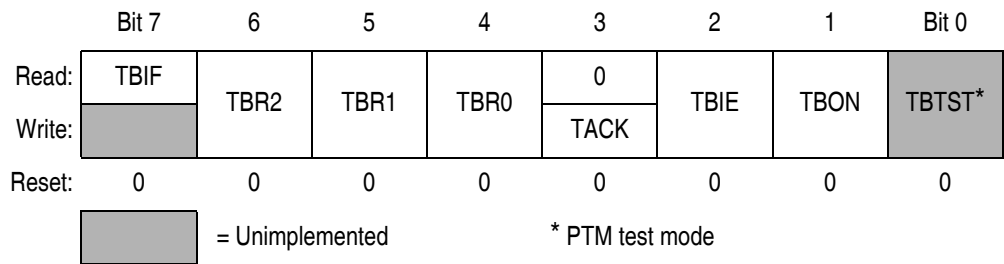
**Figure 22-1. Timebase Block Diagram**



## 22.5 Timebase Register Description

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

Address: \$001A



**Figure 22-2. Timebase Control Register (TBCR)**

**TBIF** — Timebase Interrupt Flag Bit

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

- 1 = Timebase interrupt pending
- 0 = Timebase interrupt not pending

**TBR2:TBR0** — Timebase Rate Selection Bit

These read/write bits are used to select the rate of timebase interrupts as shown in [Table 22-1](#).

**Table 22-1. Timebase Rate Selection for OSC1 = 32.768 kHz**

| TBR2 | TBR1 | TBR0 | Divider | Timebase Interrupt Rate |       |
|------|------|------|---------|-------------------------|-------|
|      |      |      |         | Hz                      | ms    |
| 0    | 0    | 0    | 32,768  | 1                       | 1000  |
| 0    | 0    | 1    | 8192    | 4                       | 250   |
| 0    | 1    | 0    | 2048    | 16                      | 62.5  |
| 0    | 1    | 1    | 128     | 256                     | ~ 3.9 |
| 1    | 0    | 0    | 64      | 512                     | ~2    |
| 1    | 0    | 1    | 32      | 1024                    | ~1    |
| 1    | 1    | 0    | 16      | 2048                    | ~0.5  |
| 1    | 1    | 1    | 8       | 4096                    | ~0.24 |

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

TACK— Timebase ACKnowledge Bit

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

1 = Clear timebase interrupt flag

0 = No effect

TBIE — Timebase Interrupt Enabled Bit

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

1 = Timebase interrupt enabled

0 = Timebase interrupt disabled

TBON — Timebase Enabled Bit

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

1 = Timebase enabled

0 = Timebase disabled and the counter initialized to 0s

## 22.6 Interrupts

The timebase module can interrupt the CPU on a regular basis with a rate defined by TBR2:TBR0. When the timebase counter chain rolls over, the TBIF flag is set. If the TBIE bit is set, enabling the timebase interrupt, the counter chain overflow will generate a CPU interrupt request.

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

## 22.7 Low-Power Modes

The timebase module remains active after execution of WAIT or STOP instructions. In wait or stop modes, the timebase register is not accessible by the CPU.

If the timebase functions are not required during wait and stop modes, reduce the power consumption by stopping the timebase before enabling the WAIT or STOP instruction.



## Section 23. Timer Interface Module (TIM)

### 23.1 Contents

|        |  |     |
|--------|--|-----|
| 23.2   | Introduction                             | 326 |
| 23.3   | Features                                 | 326 |
| 23.4   | Functional Description                   | 326 |
| 23.4.1 | TIM Counter Prescaler                    | 329 |
| 23.4.2 | Input Capture                            | 329 |
| 23.4.3 | Output Compare                           | 329 |
| 23.4.4 | Unbuffered Output Compare                | 330 |
| 23.4.5 | Buffered Output Compare                  | 330 |
| 23.4.6 | Pulse Width Modulation (PWM)             | 331 |
| 23.4.7 | Unbuffered PWM Signal Generation         | 332 |
| 23.4.8 | Buffered PWM Signal Generation           | 333 |
| 23.4.9 | PWM Initialization                       | 334 |
| 23.5   | Interrupts                               | 335 |
| 23.6   | Low-Power Modes                          | 335 |
| 23.6.1 | Wait Mode                                | 336 |
| 23.6.2 | Stop Mode                                | 336 |
| 23.7   | TIM During Break Interrupts              | 336 |
| 23.8   | I/O Signals                              | 337 |
| 23.9   | I/O Registers                            | 337 |
| 23.9.1 | TIM Status and Control Register          | 337 |
| 23.9.2 | TIM Counter Registers                    | 340 |
| 23.9.3 | TIM Counter Modulo Registers             | 341 |
| 23.9.4 | TIM Channel Status and Control Registers | 342 |
| 23.9.5 | TIM Channel Registers                    | 346 |

## 23.2 Introduction

This section describes the timer interface (TIM) module. The TIM is a 2-channel timer that provides a timing reference with input capture, output compare, and pulse-width-modulation functions. [Figure 23-1](#) is a block diagram of the TIM.

## 23.3 Features

Features of the TIM include:

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

## 23.4 Functional Description

[Figure 23-1](#) shows the structure of the TIM. The central component of the TIM is the 16-bit TIM counter that can operate as a free-running counter or a modulo up-counter. The TIM counter provides the timing reference for the input capture and output compare functions. The TIM counter modulo registers, TMODH:TMODL, control the modulo value of the TIM counter. Software can read the TIM counter value at any time without affecting the counting sequence.

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

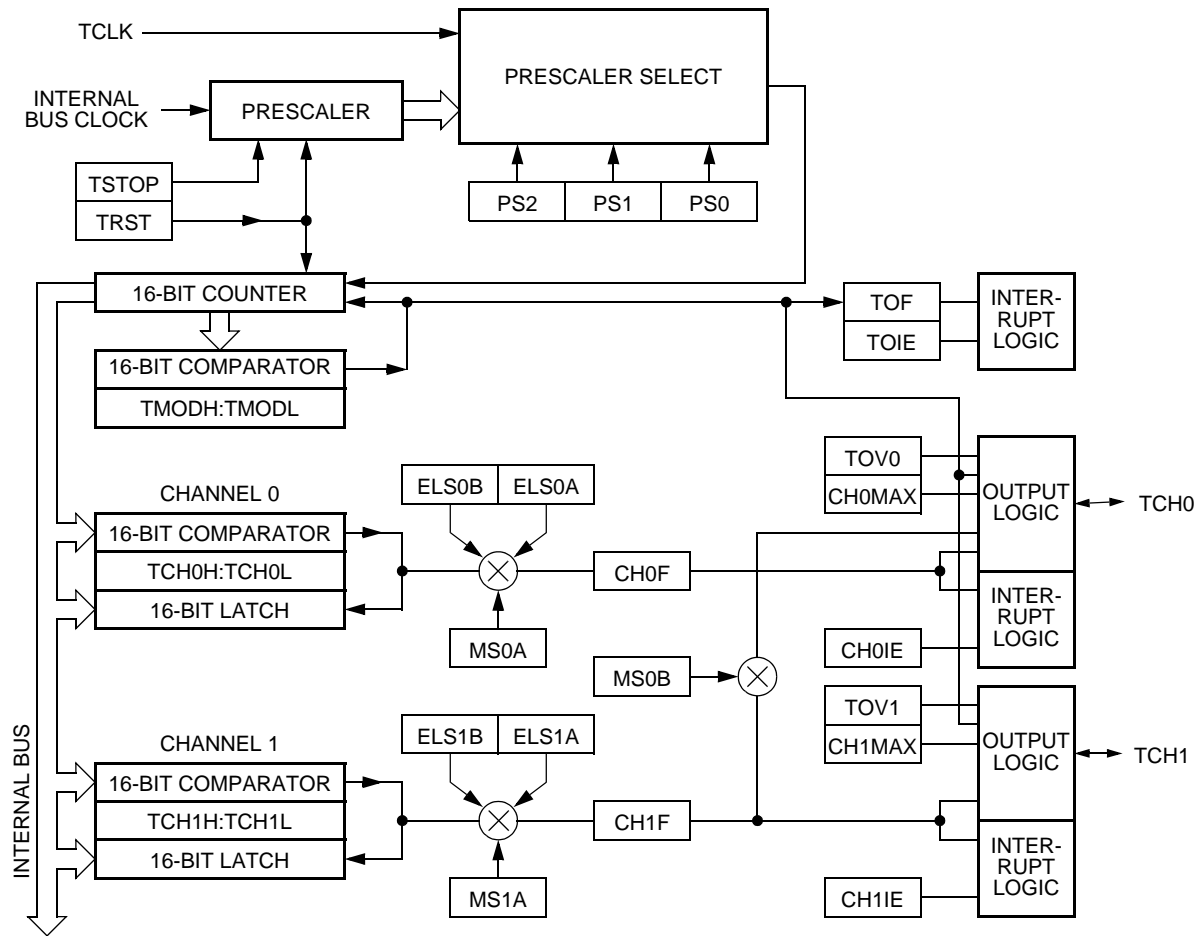


Figure 23-1. TIM Block Diagram

# Timer Interface Module (TIM)

Figure 23-2 summarizes the timer registers.

| Addr.  | Register Name  | Bit 7  | 6                         | 5     | 4     | 3    | 2     | 1     | Bit 0 |        |
|--------|--|--------|---------------------------|-------|-------|------|-------|-------|-------|--------|
| \$000F | Timer Status and Control Register (TSC)<br>See page 337.         | Read:  | TOF                       | TOIE  | TSTOP | 0    | 0     | PS2   | PS1   | PS0    |
|        |  | Write: | 0                         |       |       | TRST |       |       |       |        |
|        |  | Reset: | 0                         | 0     | 1     | 0    | 0     | 0     | 0     | 0      |
| \$0010 | Timer Counter Register High (TCNTH)<br>See page 340.             | Read:  | Bit 15                    | 14    | 13    | 12   | 11    | 10    | 9     | Bit 8  |
|        |  | Write: |                           |       |       |      |       |       |       |        |
|        |  | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$0011 | Timer Counter Register Low (TCNTL)<br>See page 340.              | Read:  | Bit 7                     | 6     | 5     | 4    | 3     | 2     | 1     | Bit 0  |
|        |  | Write: |                           |       |       |      |       |       |       |        |
|        |  | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$0012 | Timer Counter Modulo Register High (TMODH)<br>See page 341.      | Read:  | Bit 15                    | 14    | 13    | 12   | 11    | 10    | 9     | Bit 8  |
|        |  | Write: |                           |       |       |      |       |       |       |        |
|        |  | Reset: | 1                         | 1     | 1     | 1    | 1     | 1     | 1     | 1      |
| \$0013 | Timer Counter Modulo Register Low (TMODL)<br>See page 341.       | Read:  | Bit 7                     | 6     | 5     | 4    | 3     | 2     | 1     | Bit 0  |
|        |  | Write: |                           |       |       |      |       |       |       |        |
|        |  | Reset: | 1                         | 1     | 1     | 1    | 1     | 1     | 1     | 1      |
| \$0014 | Timer Channel 0 Status and Control Register (TSC0) See page 342. | Read:  | CH0F                      | CH0IE | MS0B  | MS0A | ELS0B | ELS0A | TOV0  | CH0MAX |
|        |  | Write: | 0                         |       |       |      |       |       |       |        |
|        |  | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |
| \$0015 | Timer Channel 0 Register High (TCH0H)<br>See page 346.           | Read:  | Bit 15                    | 14    | 13    | 12   | 11    | 10    | 9     | Bit 8  |
|        |  | Write: |                           |       |       |      |       |       |       |        |
|        |  | Reset: | Indeterminate after reset |       |       |      |       |       |       |        |
| \$0016 | Timer Channel 0 Register Low (TCH0L)<br>See page 346.            | Read:  | Bit 7                     | 6     | 5     | 4    | 3     | 2     | 1     | Bit 0  |
|        |  | Write: |                           |       |       |      |       |       |       |        |
|        |  | Reset: | Indeterminate after reset |       |       |      |       |       |       |        |
| \$0017 | Timer Channel 1 Status and Control Register (TSC1) See page 346. | Read:  | CH1F                      | CH1IE | 0     | MS1A | ELS1B | ELS1A | TOV1  | CH1MAX |
|        |  | Write: | 0                         |       |       |      |       |       |       |        |
|        |  | Reset: | 0                         | 0     | 0     | 0    | 0     | 0     | 0     | 0      |

= Unimplemented

Figure 23-2. TIM I/O Register Summary (Sheet 1 of 2)



| Addr.  | Register Name  | Bit 7  | 6                         | 5  | 4  | 3  | 2  | 1  | Bit 0 |       |
|--------|--|--------|---------------------------|----|----|----|----|----|-------|-------|
| \$0018 | Timer Channel 1<br>Register High (TCH1H)<br>See <a href="#">page 346</a> . | Read:  | Bit 15                    | 14 | 13 | 12 | 11 | 10 | 9     | Bit 8 |
|        |  | Write: |                           |    |    |    |    |    |       |       |
|        |  | Reset: | Indeterminate after reset |    |    |    |    |    |       |       |
| \$0019 | Timer Channel 1<br>Register Low (TCH1L)<br>See <a href="#">page 346</a> .  | Read:  | Bit 7                     | 6  | 5  | 4  | 3  | 2  | 1     | Bit 0 |
|        |  | Write: |                           |    |    |    |    |    |       |       |
|        |  | Reset: | Indeterminate after reset |    |    |    |    |    |       |       |

= Unimplemented

**Figure 23-2. TIM I/O Register Summary (Sheet 2 of 2)**

### 23.4.1 TIM Counter Prescaler

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

### 23.4.2 Input Capture

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

### 23.4.3 Output Compare

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

### 23.4.4 Unbuffered Output Compare

Any output compare channel can generate unbuffered output compare pulses as described in [23.4.3 Output Compare](#). The pulses are unbuffered because changing the output compare value requires writing the new value over the old value currently in the TIM channel registers.

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

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

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

### 23.4.5 Buffered Output Compare

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

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

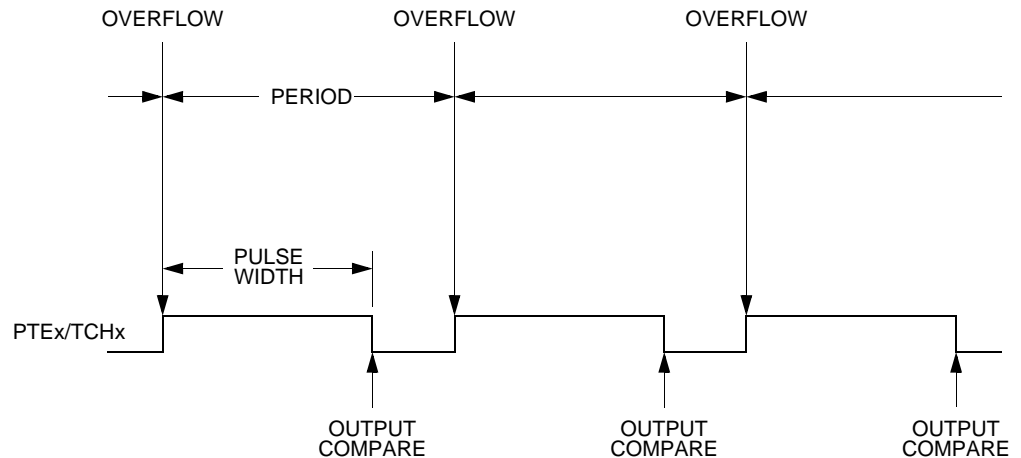
**NOTE:** *In buffered output compare operation, do not write new output compare values to the currently active channel registers. Writing to the active channel registers is the same as generating unbuffered output compares.*

### 23.4.6 Pulse Width Modulation (PWM)

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

As [Figure 23-3](#) shows, the output compare value in the TIM channel registers determines the pulse width of the PWM signal. The time between overflow and output compare is the pulse width. Program the TIM to clear the channel pin on output compare if the state of the PWM pulse is logic 1. Program the TIM to set the pin if the state of the PWM pulse is logic 0.

The value in the TIM counter modulo registers and the selected prescaler output determines the frequency of the PWM output. The frequency of an 8-bit PWM signal is variable in 256 increments. Writing \$00FF (255) to the TIM counter modulo registers produces a PWM period of 256 times the internal bus clock period if the prescaler select value is \$000. See [23.9.1 TIM Status and Control Register](#).



**Figure 23-3. PWM Period and Pulse Width**

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

### 23.4.7 Unbuffered PWM Signal Generation

Any output compare channel can generate unbuffered PWM pulses as described in [23.4.6 Pulse Width Modulation \(PWM\)](#). The pulses are unbuffered because changing the pulse width requires writing the new pulse width value over the old value currently in the TIM channel registers.

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

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

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

**NOTE:** *In PWM signal generation, do not program the PWM channel to toggle on output compare. Toggling on output compare prevents reliable 0 percent 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.*

### 23.4.8 Buffered PWM Signal Generation

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

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

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

### 23.4.9 PWM Initialization

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

1. In the TIM status and control register (TSC):
  - a. Stop the TIM counter by setting the TIM stop bit, TSTOP.
  - b. Reset the TIM counter by setting the TIM reset bit, TRST.
2. In the TIM counter modulo registers (TMODH:TMODL), write the value for the required PWM period.
3. In the TIM channel x registers (TCHxH:TCHxL), write the value for the required pulse width.
4. In TIM channel x status and control register (TSCx):
  - a. Write 0:1 (for unbuffered output compare or PWM signals) or 1:0 (for buffered output compare or PWM signals) to the mode select bits, MSxB:MSxA. See [Table 23-2](#).
  - 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 23-2](#).)

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

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

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

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

Setting the channel x maximum duty cycle bit (CHxMAX) and clearing the TOVx bit generates a 100 percent duty cycle output. (See [23.9.4 TIM Channel Status and Control Registers](#).)

## 23.5 Interrupts

The following TIM sources can generate interrupt requests:

- TIM overflow flag (TOF) — The TOF bit is set when the TIM counter value rolls over to \$0000 after matching the value in the TIM counter modulo registers. The TIM overflow interrupt enable bit, TOIE, enables TIM overflow CPU interrupt requests. TOF and TOIE are in the TIM status and control register.
- TIM channel flags (CH1F:CH0F) — The CHxF bit is set when an input capture or output compare occurs on channel x. Channel x TIM CPU interrupt requests are controlled by the channel x interrupt enable bit, CHxIE. Channel x TIM CPU interrupt requests are enabled when CHxIE = 1. CHxF and CHxIE are in the TIM channel x status and control register.

## 23.6 Low-Power Modes

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

### 23.6.1 Wait Mode

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

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

### 23.6.2 Stop Mode

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

## 23.7 TIM During Break Interrupts

A break interrupt stops the TIM counter.

The system integration module (SIM) controls whether status bits in other modules can be cleared during the break state. The BCFE bit in the SIM break flag control register (SBFCR) enables software to clear status bits during the break state. See [20.8.3 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.



## 23.8 I/O Signals

The three TIM pins are the external input clock TCLK and the two channel I/O pins, TCH0 and TCH1. Each channel I/O pin is programmable independently as an input capture pin or an output compare pin. TCH0 can be configured as buffered output compare or buffered PWM pin.

## 23.9 I/O Registers

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

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

### 23.9.1 TIM Status and Control Register


The TIM status and control register (TSC):

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

## Timer Interface Module (TIM)

Address: \$000F

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

 = Unimplemented

**Figure 23-4. TIM Status and Control Register (TSC)**

### TOF — TIM Overflow Flag Bit

This read/write flag is set when the TIM counter resets to \$0000 after reaching the modulo value programmed in the TIM counter modulo registers. Clear TOF by reading the TIM status and control register when TOF is set and then writing a logic 0 to TOF. If another TIM overflow occurs before the clearing sequence is complete, then writing logic 0 to TOF has no effect. Therefore, a TOF interrupt request cannot be lost due to inadvertent clearing of TOF. Reset clears the TOF bit. Writing a logic 1 to TOF has no effect.

1 = TIM counter has reached modulo value.

0 = TIM counter has not reached modulo value.

### TOIE — TIM Overflow Interrupt Enable Bit

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

1 = TIM overflow interrupts enabled

0 = TIM overflow interrupts disabled

### TSTOP — TIM Stop Bit

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

1 = TIM counter stopped

0 = TIM counter active

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

### TRST — TIM Reset Bit

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

- 1 = Prescaler and TIM counter cleared
- 0 = No effect

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

### PS2–PS0 — Prescaler Select Bits

These read/write bits select either the TCLK pin or one of the seven prescaler outputs as the input to the TIM counter as [Table 23-1](#) shows. Reset clears the PS[2:0] bits.

**Table 23-1. Prescaler Selection**

| PS2–PS0 | TIM 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     | Not available           |


## 23.9.2 TIM Counter Registers

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

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

Address: \$0010


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

 = Unimplemented

**Figure 23-5. TIM Counter Register High (TCNTH)**

Address: \$0011

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

 = Unimplemented

**Figure 23-6. TIM Counter Register Low (TCNTL)**

### 23.9.3 TIM Counter Modulo Registers

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

Address: \$0012

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

**Figure 23-7. TIM Counter Modulo Register High (TMODH)**

Address: \$0013

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

**Figure 23-8. TIM Counter Modulo Register Low (TMODL)**

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

## 23.9.4 TIM Channel Status and Control Registers

Each of the TIM channel status and control registers:

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

Address: \$0014

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

**Figure 23-9. TIM Channel 0 Status and Control Register (TSC0)**

Address: \$0017

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

= Unimplemented

**Figure 23-10. TIM Channel 1 Status and Control Register (TSC1)**

### CHxF — Channel x Flag Bit

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

When TIM CPU interrupt requests are enabled (CHxIE = 1), clear CHxF by reading TIM channel x status and control register with CHxF set and then writing a logic 0 to CHxF. If another interrupt request occurs before the clearing sequence is complete, then writing logic 0 to CHxF has no effect. Therefore, an interrupt request cannot be lost due to inadvertent clearing of CHxF.

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

1 = Input capture or output compare on channel x

0 = No input capture or output compare on channel x

### CHxIE — Channel x Interrupt Enable Bit

This read/write bit enables TIM CPU 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 TIM channel 0 status and control register.

Setting MS0B disables the channel 1 status and control register and renders pin TCH1 inoperable.

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 23-2](#).

[Table 23-2](#).

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 TCHx pin once PWM or output compare operation is enabled. See [Table 23-2](#). Reset clears this bit.

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

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

### ELSxB and ELSxA — Edge/Level Select Bits

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

When channel x is an output compare channel, ELSxB and ELSxA control the channel x output behavior when an output compare occurs. [Table 23-2](#) shows how ELSxB and ELSxA work. Reset clears the ELSxB and ELSxA bits.

**Table 23-2. Mode, Edge, and Level Selection**

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

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



### TOVx — Toggle On Overflow Bit

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

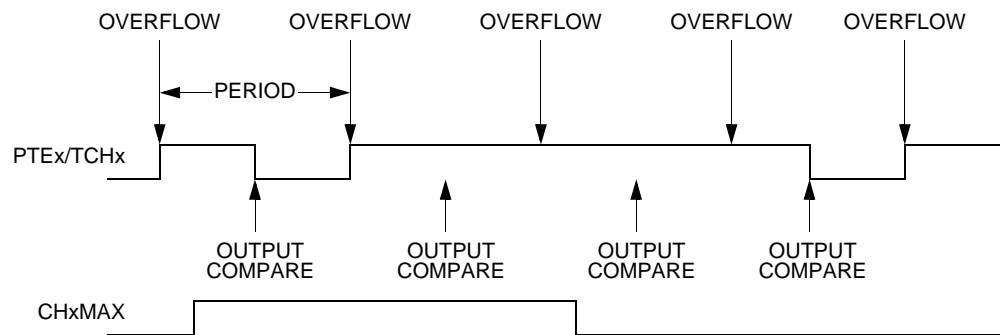
1 = Channel x pin toggles on TIM counter overflow.

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

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

### CHxMAX — Channel x Maximum Duty Cycle Bit

When the TOVx bit is at logic 0, setting the CHxMAX bit forces the duty cycle of buffered and unbuffered PWM signals to 100%. As [Figure 23-11](#) shows, the CHxMAX bit takes effect in the cycle after it is set or cleared. The output stays at the 100% duty cycle level until the cycle after CHxMAX is cleared.



**Figure 23-11. CHxMAX Latency**

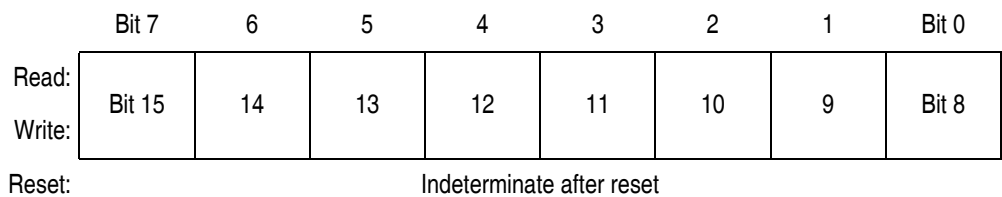
## 23.9.5 TIM Channel Registers

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

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

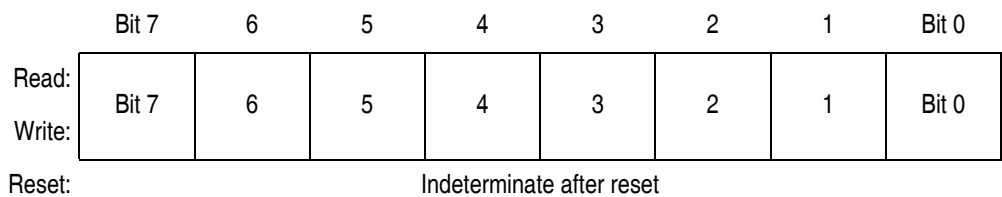
In output compare mode ( $MSxB:MSxA \neq 0:0$ ), writing to the high byte of the TIM channel x registers (TCHxH) inhibits output compares until the low byte (TCHxL) is written.

Address: \$0015



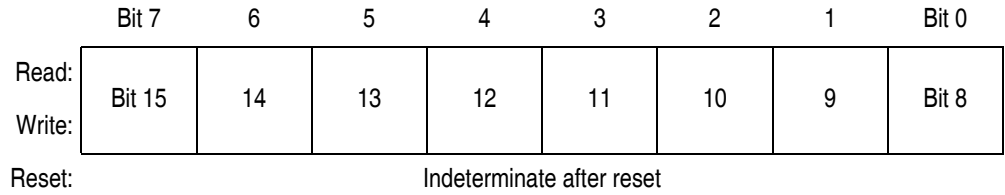
**Figure 23-12. TIM Channel 0 Register High (TCH0H)**

Address: \$0016



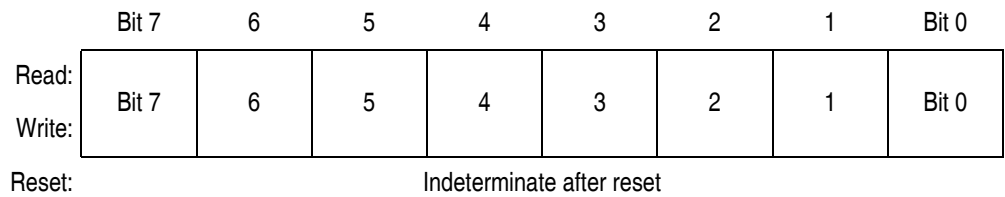
**Figure 23-13. TIM Channel 0 Register Low (TCH0L)**

Address: \$0018



**Figure 23-14. TIM Channel 1 Register High (TCH1H)**

Address: \$0019



**Figure 23-15. TIM Channel 1 Register Low (TCH1L)**



## Section 24. ROM Version Overview (ROM)

### 24.1 Contents

|      |  |     |
|------|--|-----|
| 24.2 | Introduction . . . . .                       | 349 |
| 24.3 | FLASH for ROM Substitution . . . . .         | 349 |
| 24.4 | Configuration Register Programming . . . . . | 351 |

### 24.2 Introduction

This section describes the differences between the ROM version (MC68HC08TV24) and the FLASH version (MC68HC908TV24) of the microcontroller.

Basically, the differences are:

- Use of ROM instead of FLASH in both OSD and user memories
- Configuration register not programmable on ROM version

### 24.3 FLASH for ROM Substitution

**Figure 24-1** shows the structure of MC68HC08TV24. FLASH memories and related supporting modules are replaced by ROM memories. User FLASH vector space is substituted by ROM vector space. Additionally, the following registers are not present in the ROM version:

- User FLASH test control register (FL1TCR)
- User FLASH control register (FL1CR)
- OSD FLASH test control register (FL2TCR)
- OSD FLASH control register (FL2CR)

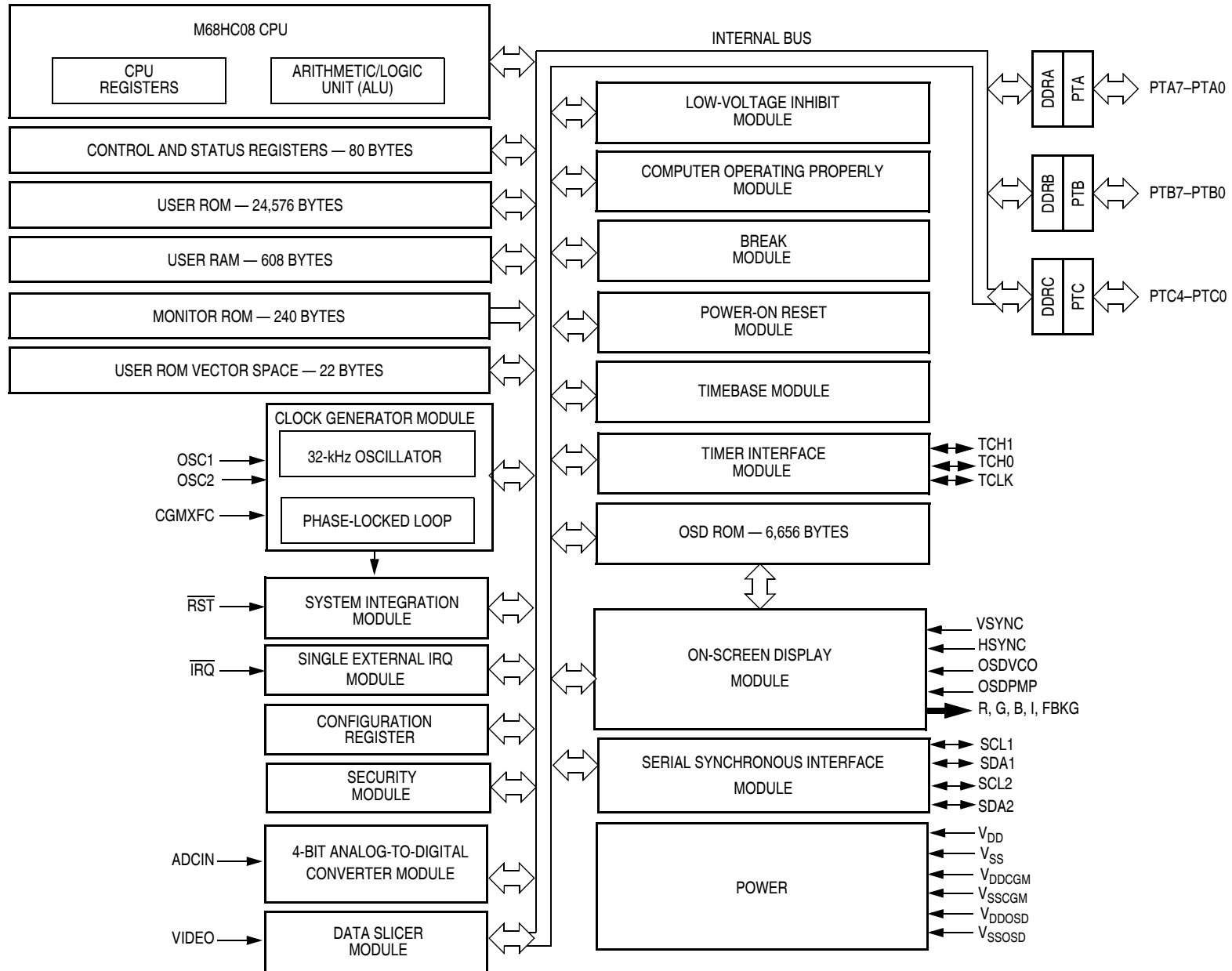


Figure 24-1. MC68HC08TV24 Block Diagram

- User FLASH block protect register (FL1BPR)
- OSD FLASH block protect register (FL2BPR)

Because of modularity reasons, the OSD ROM size is 6,656 bytes, but only 6,528 bytes will be used by the OSD module. The remaining bytes can be programmed with any value.

## 24.4 Configuration Register Programming

The configuration register (CONFIG) is not programmable in the ROM version. While in the FLASH version, the CONFIG register can be written once after each reset; in the ROM version, the options have to be specified beforehand and delivered to mask elaboration together with the ROM contents.

Some modules affected by the configuration register bits make reference to default values of these bits and have recommendation notes on programming them. These notes are not entirely applicable to the ROM version. The CONFIG bits will neither assume the described default values after reset nor be modified later under user code control.





## Section 25. Preliminary Electrical Specifications

### 25.1 Contents

|         |   |     |
|---------|---|-----|
| 25.2    | Introduction . . . . .                            | 353 |
| 25.3    | Absolute Maximum Ratings . . . . .                | 354 |
| 25.4    | Functional Operating Range. . . . .               | 355 |
| 25.5    | Thermal Characteristics . . . . .                 | 355 |
| 25.6    | 5.0-V DC Electrical Characteristics. . . . .      | 356 |
| 25.7    | 5.0-V Control Timing. . . . .                     | 358 |
| 25.8    | ADC4 Characteristics . . . . .                    | 359 |
| 25.9    | Timer Interface Module Characteristics . . . . .  | 359 |
| 25.10   | Clock Generation Module Characteristics . . . . . | 359 |
| 25.10.1 | CGM Component Specifications . . . . .            | 359 |
| 25.10.2 | CGM Electrical Specifications . . . . .           | 360 |
| 25.11   | Memory Characteristics . . . . .                  | 361 |
| 25.12   | 5.0-V SSI Characteristics . . . . .               | 362 |

### 25.2 Introduction

This section contains electrical and timing specifications. These values are design targets and have not yet been fully tested.

## 25.3 Absolute Maximum Ratings

Maximum ratings are the extreme limits to which the MCU can be exposed without permanently damaging it.

The MCU contains circuitry to protect the inputs against damage from high static voltages; however, do not apply voltages higher than those shown in the table below. Keep  $V_{In}$  and  $V_{Out}$  within the range  $V_{SS} \leq (V_{In} \text{ or } V_{Out}) \leq V_{DD}$ . Connect unused inputs to the appropriate voltage level, either  $V_{SS}$  or  $V_{DD}$ .

| Characteristic <sup>(1)</sup>                           | 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          | $\pm 25$                         | mA   |
| Maximum current into $V_{DD}$                           | $I_{mvdd}$ | 100                              | mA   |
| Maximum current out of $V_{SS}$                         | $I_{mvss}$ | 100                              | mA   |
| Storage temperature                                     | $T_{stg}$  | -55 to +150                      | °C   |

Note:

1. Voltages are referenced to  $V_{SS}$ .

**NOTE:** *This device is not guaranteed to operate properly at the maximum ratings. Refer to [25.6 5.0-V DC Electrical Characteristics](#) for guaranteed operating conditions.*

## 25.4 Functional Operating Range

| Characteristic              | Symbol   | Value          | Unit |
|-----------------------------|----------|----------------|------|
| Operating temperature range | $T_A$    | -40 to +85     | °C   |
| Operating voltage range     | $V_{DD}$ | $5.0 \pm 10\%$ | V    |

## 25.5 Thermal Characteristics

| Characteristic                      | Symbol        | Value  | Unit |
|-------------------------------------|---------------|--|------|
| Thermal resistance<br>TQFP (52-pin) | $\theta_{JA}$ | 70   | °C/W |
| I/O pin power dissipation           | $P_{I/O}$     | User-determined  | W    |
| Power dissipation <sup>(1)</sup>    | $P_D$         | $P_D = (I_{DD} \times V_{DD}) + P_{I/O}$<br>$= K/(T_J + 273 \text{ °C})$ | W    |
| Constant <sup>(2)</sup>             | K             | $P_J \times (T_A + 273 \text{ °C})$<br>$+ P_D^2 \times \theta_{JA}$      | W/°C |
| Average junction temperature        | $T_J$         | $T_A + (P_D \times \theta_{JA})$   | °C   |
| Maximum junction temperature        | $T_{JM}$      | 125  | °C   |

Notes:

1. Power dissipation is a function of temperature.
2. K is a constant unique to the device. K can be determined for a known,  $T_A$ , and measured  $P_D$ . With this value of K,  $P_D$  and  $T_J$  can be determined for any value of  $T_A$ .

## 25.6 5.0-V DC Electrical Characteristics

| Characteristic <sup>(1)</sup>   | Symbol      | Min                 | Typ <sup>(2)</sup> | Max                 | Unit    |
|---|-------------|---------------------|--------------------|---------------------|---------|
| Output high voltage<br>( $I_{Load} = -2.0$ mA) all ports, R, G, B, I, FBKG, TCH0, TCH1  | $V_{OH}$    | $V_{DD} - 0.8$      | —                  | —                   | V       |
| ( $I_{Load} = -5.0$ mA) all ports, R, G, B, I, FBKG, TCH0, TCH1   | $V_{OH}$    | $V_{DD} - 1.5$      | —                  | —                   | V       |
| Output low voltage<br>( $I_{Load} = 1.6$ mA) all ports, $\overline{RST}$ , R, G, B, I, FBKG, TCH0, TCH1, SCL1, SCL2, SDA1, SDA2 | $V_{OL}$    | —                   | —                  | 0.4                 | V       |
| ( $I_{Load} = 10$ mA) all ports, $\overline{RST}$ , R, G, B, I, FBKG, TCH0, TCH1, SCL1, SCL2, SDA1, SDA2                        | $V_{OL}$    | —                   | —                  | 1.5                 | V       |
| Input high voltage<br>All ports, $\overline{RST}$ , $\overline{IRQ}$ , VSYNC, HSYNC, SCL1, SCL2, SDA1, SDA2, TCLK, TCH0, TCH1   | $V_{IH}$    | $0.7 \times V_{DD}$ | —                  | $V_{DD}$            | V       |
| Input low voltage<br>All ports, $\overline{RST}$ , $\overline{IRQ}$ , VSYNC, HSYNC, SCL1, SCL2, SDA1, SDA2, TCLK, TCH0, TCH1    | $V_{IL}$    | $V_{SS}$            | —                  | $0.2 \times V_{DD}$ | V       |
| $V_{DD}$ supply current   |             |                     |                    |                     |         |
| Run <sup>(3)</sup>  |             | —                   | —                  | 30                  | mA      |
| Wait <sup>(4)</sup>   |             | —                   | —                  | 12                  | mA      |
| Stop <sup>(5)</sup>   |             |                     |                    |                     |         |
| 25 °C   | $I_{DD}$    | —                   | 5                  | —                   | $\mu$ A |
| 25 °C with TBM enabled <sup>(6)</sup>   |             | —                   | 20                 | —                   | $\mu$ A |
| 25 °C with LVI and TBM enabled <sup>(6)</sup>   |             | —                   | 300                | —                   | $\mu$ A |
| –40 °C to 85 °C with TBM enabled <sup>(6)</sup>   |             | —                   | 50                 | —                   | $\mu$ A |
| –40 °C to 85 °C with LVI and TBM enabled <sup>(6)</sup>   |             | —                   | 500                | —                   | $\mu$ A |
| I/O ports Hi-Z leakage current  | $I_{IL}$    | —                   | —                  | $\pm 10$            | $\mu$ A |
| Input current   | $I_{In}$    | —                   | —                  | $\pm 1$             | $\mu$ A |
| Capacitance   | $C_{Out}$   | —                   | —                  | 12                  | pF      |
| Ports (as input or output)  | $C_{In}$    | —                   | —                  | 8                   | pF      |
| Monitor mode entry voltage  | $V_{TST}$   | $V_{DD} + 2.5$      | —                  | 9                   | V       |
| Low-voltage inhibit, trip falling voltage — LVI5OR3 = 1   | $V_{TRIPF}$ | 4.13                | 4.3                | 4.35                | V       |
| Low-voltage inhibit, trip rising voltage — LVI5OR3 = 1  | $V_{TRIPR}$ | 4.23                | 4.4                | 4.45                | V       |
| Low-voltage inhibit reset/recover hysteresis<br>( $V_{TRIPF} + V_{HYS} = V_{TRIPR}$ ) — LVI5OR3 = 1                             | $V_{HYS}$   | —                   | 100                | —                   | mV      |

| Characteristic <sup>(1)</sup>   | Symbol              | Min   | Typ <sup>(2)</sup> | Max  | Unit |
|---|---------------------|-------|--------------------|------|------|
| Low-voltage inhibit, trip falling voltage — LVI5OR3 = 0   | V <sub>TRIPF</sub>  | 2.5   | 2.6                | 2.63 | V    |
| Low-voltage inhibit, trip rising voltage — LVI5OR3 = 0  | V <sub>TRIPR</sub>  | 2.6   | 2.66               | 2.73 | V    |
| Low-voltage inhibit reset/recover hysteresis<br>(V <sub>TRIPF</sub> + V <sub>HYS</sub> = V <sub>TRIPR</sub> ) — LVI5OR3 = 0 | V <sub>HYS</sub>    | —     | 60                 | —    | mV   |
| POR rearm voltage <sup>(7)</sup>  | V <sub>POR</sub>    | 0     | —                  | 100  | mV   |
| POR reset voltage <sup>(8)</sup>  | V <sub>PORRST</sub> | 0     | 700                | 800  | mV   |
| POR rise time ramp rate <sup>(9)</sup>  | R <sub>POR</sub>    | 0.035 | —                  | —    | V/ms |

Notes:

- V<sub>DD</sub> = 5.0 Vdc ± 10%, V<sub>SS</sub> = 0 Vdc, T<sub>A</sub> = T<sub>L</sub> to T<sub>H</sub>, unless otherwise noted
- Typical values reflect average measurements at midpoint of voltage range, 25 °C only.
- Run (operating) I<sub>DD</sub> measured using external square wave clock source (f<sub>OSC</sub> = 32.8 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C<sub>L</sub> = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects run I<sub>DD</sub>. Measured with all modules enabled.
- Wait I<sub>DD</sub> measured using external square wave clock source (f<sub>OSC</sub> = 32.8 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. C<sub>L</sub> = 20 pF on OSC2. All ports configured as inputs. OSC2 capacitance linearly affects wait I<sub>DD</sub>. Measured with PLL and LVI enabled.
- Stop I<sub>DD</sub> is measured with OSC1 = V<sub>SS</sub>.
- Stop I<sub>DD</sub> with TBM enabled is measured using an external square wave clock source (f<sub>OSC</sub> = 32.8 MHz). All inputs 0.2 V from rail. No dc loads. Less than 100 pF on all outputs. All inputs configured as inputs.
- Maximum is highest voltage that POR is guaranteed.
- Maximum is highest voltage that POR is possible.
- If minimum V<sub>DD</sub> is not reached before the internal POR reset is released,  $\overline{\text{RST}}$  must be driven low externally until minimum V<sub>DD</sub> is reached.

## 25.7 5.0-V Control Timing

| Characteristic <sup>(1)</sup>   | Symbol                         | Min                     | Max         | Unit            |
|---|--------------------------------|-------------------------|-------------|-----------------|
| Frequency of operation <sup>(2)</sup><br>Crystal option<br>External clock option <sup>(3)</sup> | $f_{osc}$                      | 32<br>dc <sup>(4)</sup> | 100<br>32.8 | kHz<br>MHz      |
| Internal operating frequency  | $f_{op}$                       | —                       | 8.2         | MHz             |
| Internal clock period (1/ $f_{op}$ )  | $t_{cyc}$                      | 122                     | —           | ns              |
| $\overline{RESET}$ input pulse width low <sup>(5)</sup>   | $t_{IRL}$                      | 50                      | —           | ns              |
| $\overline{IRQ}$ interrupt pulse width low <sup>(6)</sup><br>(edge-triggered)                   | $t_{LIH}$                      | 50                      | —           | ns              |
| $\overline{IRQ}$ interrupt pulse period   | $t_{LIL}$                      | TBD Note 8              | —           | $t_{cyc}$       |
| 16-bit timer <sup>(7)</sup><br>Input capture pulse width<br>Input capture period                | $t_{TH}, t_{TL}$<br>$t_{TLTL}$ | TBD<br>Note 8           | —<br>—      | ns<br>$t_{cyc}$ |

### Notes:

- $V_{SS} = 0$  Vdc; timing shown with respect to 20%  $V_{DD}$  and 70%  $V_{SS}$  unless otherwise noted
- See [25.10 Clock Generation Module Characteristics](#) for more information.
- No more than 10% duty cycle deviation from 50%
- Some modules may require a minimum frequency greater than dc for proper operation. See appropriate table for this information.
- Minimum pulse width reset is guaranteed to be recognized. It is possible for a smaller pulse width to cause a reset.
- Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
- Minimum pulse width is for guaranteed interrupt. It is possible for a smaller pulse width to be recognized.
- The minimum period,  $t_{LIL}$  or  $t_{TLTL}$ , should not be less than the number of cycles it takes to execute the interrupt service routine plus  $t_{cyc}$ .

## 25.8 ADC4 Characteristics

| Characteristic <sup>(1)</sup>  | Symbol     | Min | Max       | Unit      |
|--------------------------------|------------|-----|-----------|-----------|
| Input voltages                 | $V_{ADIN}$ | 0   | $V_{DD}$  | V         |
| D/A Resolution                 | $B_{DA}$   | 4   | 4         | Bits      |
| D/A conversion error           | $A_{DA}$   | —   | $\pm 1/2$ | LSB       |
| D/A conversion range           | $R_{DA}$   | 0   | $V_{DD}$  | V         |
| Comparator conversion time     | $t_{CONV}$ | 2   | 3         | $t_{cyc}$ |
| Comparator initialization time | $t_{ADS}$  | —   | 1.3       | $\mu s$   |

Notes:

1.  $V_{DD} = 5.0 \text{ Vdc} \pm 10\%$ ,  $V_{SS} = 0 \text{ Vdc}$ ,  $V_{DDOSD} = 5.0 \text{ Vdc} \pm 10\%$ ,  $V_{SSOSD} = 0 \text{ Vdc}$

## 25.9 Timer Interface Module Characteristics

| Characteristic            | Symbol                | Min | Max | Unit      |
|---------------------------|-----------------------|-----|-----|-----------|
| Input capture pulse width | $t_{TIH}$ , $t_{TIL}$ | 1   | —   | $t_{cyc}$ |

## 25.10 Clock Generation Module Characteristics

### 25.10.1 CGM Component Specifications

| Characteristic                             | Symbol     | Min | Typ            | Max | Unit      |
|--|------------|-----|----------------|-----|-----------|
| Crystal reference frequency <sup>(1)</sup> | $f_{XCLK}$ | 30  | 32.768         | 100 | kHz       |
| Crystal load capacitance <sup>(2)</sup>    | $C_L$      | —   | —              | —   | pF        |
| Crystal fixed capacitance <sup>(2)</sup>   | $C_1$      | 6   | $2 \times C_L$ | 40  | pF        |
| Crystal tuning capacitance <sup>(2)</sup>  | $C_2$      | 6   | $2 \times C_L$ | 40  | pF        |
| Feedback bias resistor                     | $R_B$      | 10  | 10             | 22  | $M\Omega$ |
| Series resistor                            | $R_S$      | 330 | 330            | 470 | $k\Omega$ |

Notes:

1. Fundamental mode crystals only
2. Consult crystal manufacturer's data.

# Preliminary Electrical Specifications

## 25.10.2 CGM Electrical Specifications

| Description   | Symbol     | Min    | Typ    | Max                                      | Unit |
|---|------------|--------|--------|--|------|
| Operating voltage   | $V_{DD}$   | 2.7    | —      | 5.5                                      | V    |
| Operating temperature                                       | T          | −40    | 25     | 130                                      | °C   |
| Crystal reference frequency                                 | $f_{RCLK}$ | 30     | 32.768 | 100                                      | kHz  |
| Range nominal multiplier                                    | $f_{NOM}$  | —      | 38.4   | —  | kHz  |
| VCO center-of-range frequency <sup>(1)</sup>                | $f_{VRS}$  | 38.4 k | —      | 40.0 M                                   | Hz   |
| Medium-voltage VCO center-of-range frequency <sup>(2)</sup> | $f_{VRS}$  | 38.4 k | —      | 40.0 M                                   | Hz   |
| VCO range linear range multiplier                           | L          | 1      | —      | 255                                      |      |
| VCO power-of-two range multiplier                           | $2^E$      | 1      | —      | 4  |      |
| VCO multiply factor   | N          | 1      | —      | 4095                                     |      |
| VCO prescale multiplier                                     | $2^P$      | 1      | 1      | 8  |      |
| Reference divider factor                                    | R          | 1      | 1      | 15                                       |      |
| VCO operating frequency                                     | $f_{VCLK}$ | 38.4 k | —      | 40.0 M                                   | Hz   |
| Bus operating frequency <sup>(1)</sup>                      | $f_{BUS}$  | —      | —      | 8.2                                      | MHz  |
| Bus frequency @ medium voltage <sup>(2)</sup>               | $f_{BUS}$  | —      | —      | 4.1                                      | MHz  |
| Manual acquisition time                                     | $t_{Lock}$ | —      | —      | 50                                       | ms   |
| Automatic lock time   | $t_{Lock}$ | —      | —      | 50                                       | ms   |
| PLL jitter <sup>(3)</sup>                                   | $f_J$      | 0      | —      | $f_{RCLK} \times 0.025\% \times 2^P N/4$ | Hz   |
| External clock input frequency<br>PLL disabled              | $f_{OSC}$  | dc     | —      | 32.8 M                                   | Hz   |
| External clock input frequency<br>PLL enabled               | $f_{OSC}$  | 30 k   | —      | 1.5 M                                    | Hz   |

Notes:

1.  $5.0\text{ V} \pm 10\% V_{DD}$
2.  $3.0\text{ V} \pm 10\% V_{DD}$
3. Deviation of average bus frequency over 2 ms. N = VCO multiplier.



## 25.11 Memory Characteristics

| Characteristic  | Symbol            | Min  | Typ | Max   | Unit    |
|---|-------------------|------|-----|-------|---------|
| RAM data retention voltage  | $V_{RDR}$         | 1.3  | —   | —     | V       |
| FLASH pages per row (both FLASH memories)   | —                 | 8    |     |       | Pages   |
| FLASH bytes per page (user FLASH)   | —                 | 8    |     |       | Bytes   |
| FLASH bytes per page (OSD FLASH)  | —                 | 4    |     |       | Bytes   |
| FLASH read bus clock frequency  | $f_{Read}^{(1)}$  | 32 k | —   | 8.4 M | Hz      |
| FLASH charge pump clock frequency<br>(See <a href="#">12.5.1 FLASH Charge Pump Frequency Control</a> .) | $f_{Pump}^{(2)}$  | 1.8  | —   | 2.5   | MHz     |
| FLASH block/bulk erase time   | $t_{Erase}$       | 100  | —   | —     | ms      |
| FLASH high-voltage kill time  | $t_{Kill}$        | 200  | —   | —     | $\mu$ s |
| FLASH return to read time   | $t_{HVD}$         | 50   | —   | —     | $\mu$ s |
| FLASH page program pulses   | $flsPulses^{(3)}$ | 1    | 20  | TBD   | Pulses  |
| FLASH page program step size  | $t_{PROG}^{(4)}$  | 1.0  | —   | 1.2   | ms      |
| FLASH cumulative program time per row between erase cycles  | $t_{Row}^{(5)}$   | —    | —   | TBD   | ms      |
| FLASH HVEN low to MARGIN high time  | $t_{HVTV}$        | 50   | —   | —     | $\mu$ s |
| FLASH MARGIN high to PGM low time   | $t_{VTP}$         | 150  | —   | —     | $\mu$ s |
| FLASH row erase endurance <sup>(6)</sup>  | —                 | 100  | —   | —     | Cycles  |
| FLASH row program endurance <sup>(7)</sup>  | —                 | 100  | —   | —     | Cycles  |
| FLASH data retention time <sup>(8)</sup>  | —                 | 10   | —   | —     | Years   |

Notes:

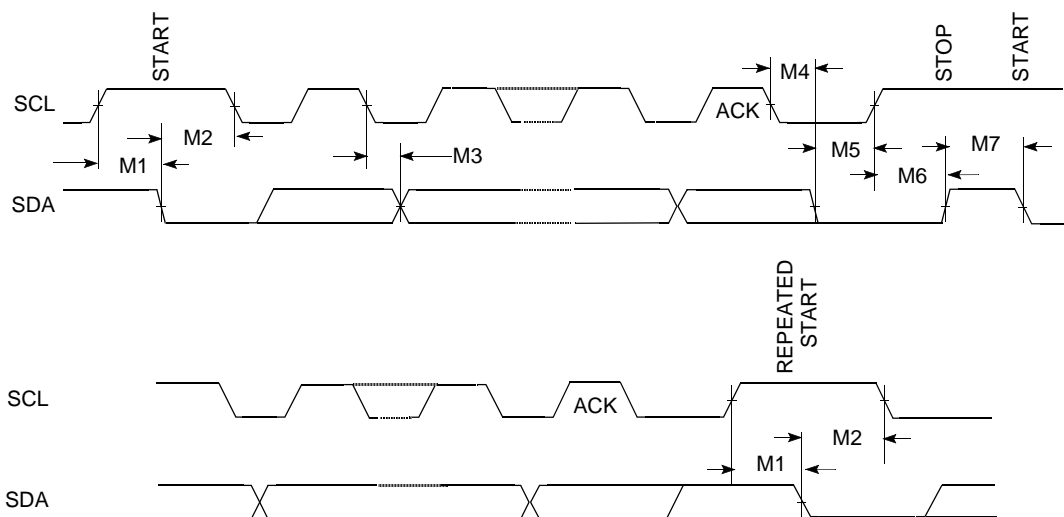
- $f_{Read}$  is defined as the frequency range for which the FLASH memory can be read.
- $f_{Pump}$  is defined as the charge pump clock frequency required for program, erase, and margin read operations.
- $flsPulses$  is defined as the number of pulses used to program the FLASH using the required smart program algorithm.
- $t_{PROG}$  is defined as the amount of time during one page program cycle that HVEN is held high.
- $t_{Row}$  is defined as the cumulative time a row can see the program voltage before the row must be erased before further programming.
- The minimum row endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase / program cycles.
- The minimum row endurance value specifies each row of the FLASH memory is guaranteed to work for at least this many erase / program cycles.
- The FLASH is guaranteed to retain data over the entire temperature range for at least the minimum time specified.

## 25.12 5.0-V SSI Characteristics

| Characteristic                                       | Diagram Symbol <sup>(1)</sup> | Min <sup>(2)</sup> | Typ | Max | Unit |
|--|-------------------------------|--------------------|-----|-----|------|
| Start condition setup time                           | M1                            | 4.7                | —   | —   | ms   |
| Start condition hold time                            | M2                            | 4.0                | —   | —   | ms   |
| Data hold time                                       | M3                            | 125                | —   | —   | ns   |
| Data falling from ninth clock falling <sup>(3)</sup> | M4                            | 4.0                | —   | —   | ms   |
| Clock rising from data falling <sup>(3)</sup>        | M5                            | 4.0                | —   | —   | ms   |
| Stop condition setup time                            | M6                            | 4.0                | —   | —   | ms   |
| Bus free time between a stop and a start condition   | M7                            | 4.7                | —   | —   | ms   |

Notes:

1. These signals are open-drain type outputs. The time required for these signals to attain steady high or low state is dependent on the external signal capacitance and pull-up resistor values.
2. All these values do not depend on output clock (SCL) selected frequency. The internal bus clock frequency used is 8 MHz.
3. M4 and M5 are not part of the I<sup>2</sup>C standard.



**Figure 25-1. SSI Timing**

## Section 26. Mechanical Specifications

### 26.1 Contents

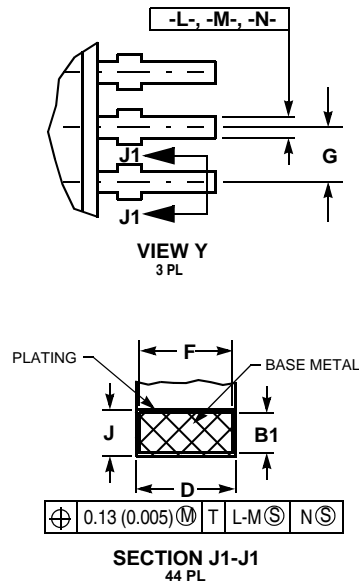
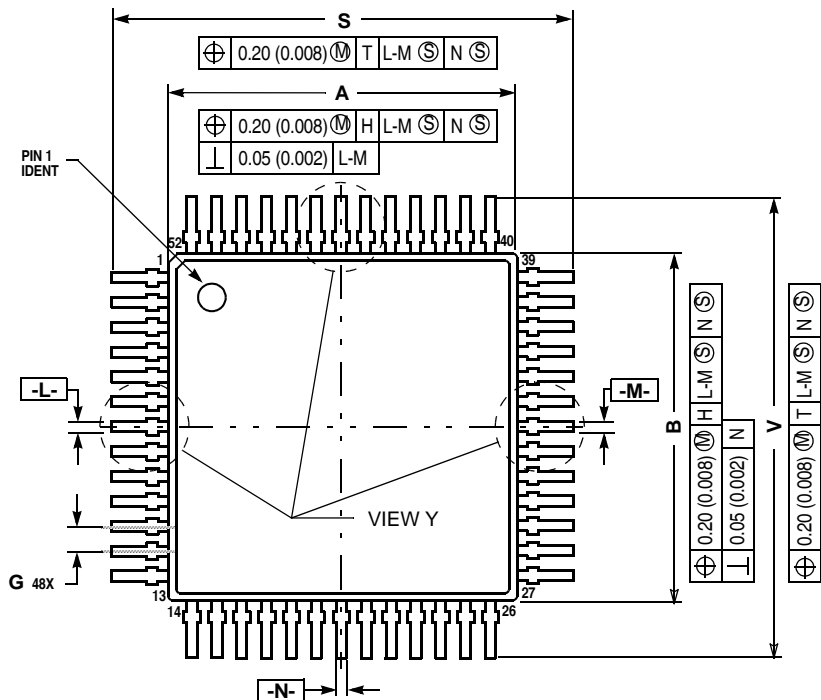
|      |  |     |
|------|--|-----|
| 26.2 | Introduction . . . . .                         | 363 |
| 26.3 | 52-Pin Plastic Quad Flat Pack (PQFP) . . . . . | 364 |

### 26.2 Introduction

This section gives the dimensions for the 52-pin thin quad flat pack (case 848D-03).

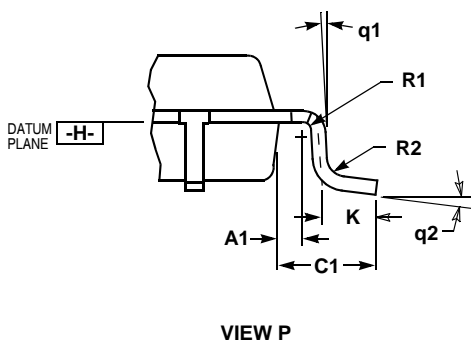
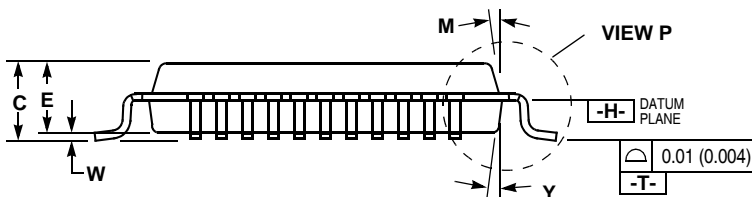
The following figure shows the latest package drawing at the time of this publication. To make sure that you have the latest package specifications, please visit the Freescale website at <http://freescale.com>. Follow Worldwide Web on-line instructions to retrieve the current mechanical specifications.

## 26.3 52-Pin Plastic Quad Flat Pack (PQFP)



**NOTES:**

- DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
- CONTROLLING DIMENSION: MILLIMETER.
- 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.
- DATUMS -L-, -M- AND -N- TO BE DETERMINED AT DATUM PLANE -H-.
- DIMENSIONS S AND V TO BE DETERMINED AT SEATING PLANE -T-.
- DIMENSIONS A AND B DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 0.25 (0.010) PER SIDE. DIMENSIONS A AND B DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE -H-.
- DIMENSION D DOES NOT INCLUDE DAMBAR PROTRUSION. DAMBAR PROTRUSION SHALL NOT CAUSE THE D DIMENSION TO EXCEED 0.46 (0.018). MINIMUM DIMENSION BETWEEN PROTRUSION AND ADJACENT LEAD OR PROTRUSION 0.07 (0.003).



| DIM | MILLIMETERS |      | INCHES |       |
|-----|-------------|------|--------|-------|
|     | MIN         | MAX  | MIN    | MAX   |
| A   | 10.00       | BSC  | 0.394  | BSC   |
| B   | 10.00       | BSC  | 0.394  | BSC   |
| C   |             | 1.70 |        | 0.067 |
| D   | 0.20        | 0.40 | 0.008  | 0.016 |
| E   | 1.30        | 1.50 | 0.051  | 0.059 |
| F   | 0.22        | 0.35 | 0.009  | 0.014 |
| G   | 0.65        | BSC  | 0.026  | BSC   |
| J   | 0.07        | 0.20 | 0.003  | 0.008 |
| K   | 0.50        | REF  | 0.020  | REF   |
| M   | 12°         | REF  | 12°    | REF   |
| S   | 12.00       | BSC  | 0.472  | BSC   |
| V   | 12.00       | BSC  | 0.472  | BSC   |
| W   | 0.05        | 0.20 | 0.002  | 0.008 |
| Y   | 12°         | REF  | 12°    | REF   |
| A1  | 0.20        | REF  | 0.008  | REF   |
| B1  | 0.09        | 0.16 | 0.004  | 0.006 |
| C1  | 1.00        | REF  | 0.039  | REF   |
| R1  | 0.08        | 0.20 | 0.003  | 0.008 |
| R2  | 0.08        | 0.20 | 0.003  | 0.008 |
| q1  | 0°          |      | 0°     |       |
| q2  | 0°          | 7°   | 0°     | 7°    |

## Section 27. Ordering Information

### 27.1 Contents

|      |                            |     |
|------|----------------------------|-----|
| 27.2 | Introduction . . . . .     | 365 |
| 27.3 | MC Order Numbers . . . . . | 365 |

### 27.2 Introduction

This section contains ordering numbers for the MC68HC908TV24 and the MC68HC08TV24.

### 27.3 MC Order Numbers

**Table 27-1. MC Order Numbers**

| MC Order Number <sup>(1)</sup> | Operating Temperature Range |
|--------------------------------|-----------------------------|
| MC68HC08TV24CFB                | –40 °C to +85 °C            |
| MC68HC908TV24CFB               | –40 °C to +85 °C            |

Note:

1. FB = Plastic quad flat pack

# Ordering Information



## **How to Reach Us:**

### **Home Page:**

[www.freescale.com](http://www.freescale.com)

### **E-mail:**

[support@freescale.com](mailto: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](mailto: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](mailto: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](mailto: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](mailto: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](mailto: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. The ARM POWERED logo is a registered trademark of ARM Limited. ARM7TDMI-S is a trademark of ARM Limited. Java and all other Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. The Bluetooth trademarks are owned by their proprietor and used by Freescale Semiconductor, Inc. under license.

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

Rev. 2.1

MC68HC908TV24/D

August 16, 2005

