# C8051F336/7/8/9

## Mixed-Signal Byte-Programmable EPROM MCU

### Analog Peripherals
- **10-Bit ADC ('F336/8 only)**
  - Up to 200 ksps
  - Up to 20 external single-ended or differential inputs
  - VREF from on-chip VREF, external pin or $V_{DD}$
  - Internal or external start of conversion source
  - Built-in temperature sensor
- **10-Bit Current Output DAC ('F336/8 only)**
- **Comparator**
  - Programmable hysteresis and response time
  - Configurable as interrupt or reset source

### On-Chip Debug
- On-chip debug circuitry facilitates full speed, non-intrusive in-system debug (no emulator required)
- Provides breakpoints, single stepping, inspect/modify memory and registers
- Superior performance to emulation systems using ICE-chips, target pods, and sockets
- Low cost, **complete** development kit

### Supply Voltage 2.7 to 3.6 V
- Built-in voltage supply monitor

### High-Speed 8051 µC Core
- Pipelined instruction architecture; executes 70% of instructions in 1 or 2 system clocks
- Up to 25 MIPS throughput with 25 MHz clock
- Expanded interrupt handler

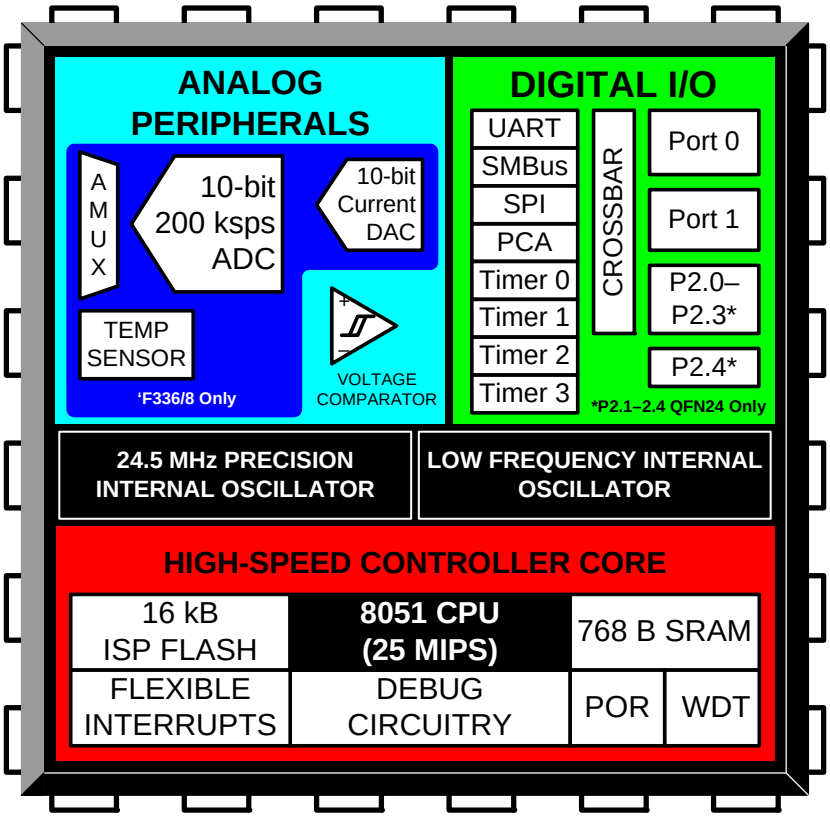
### Temperature Range: –40 to +85 °C

### Memory
- 768 bytes internal data RAM (256 + 512)
- 16 kB Flash; In-system programmable in 512-byte Sectors (512 bytes are reserved)

### Digital Peripherals
- 21 or 17 Port I/O; All 5 V tolerant with high sink current
- **Pin-compatible with C8051F330 family of MCUs**
- Hardware enhanced UART, SMBus™ ($I^2C$ compatible), and enhanced SPI™ serial ports
- Four general purpose 16-bit counter/timers
- 16-Bit programmable counter array (PCA) with three capture/compare modules and enhanced PWM functionality
- Real time clock mode using timer and crystal

### Clock Sources
- 24.5 MHz ±2% Oscillator
  - Supports crystal-less UART operation
  - Low-power suspend mode with fast wake time
- 80/20/40/10 kHz low-frequency, low-power oscillator
- External oscillator: Crystal, RC, C, or clock (1 or 2 pin modes)
- Can switch between clock sources on-the-fly; useful in power saving modes

### 20 or 24-Pin QFN (4 x 4 mm)





Copyright © 2008 by Silicon Laboratories

## Table of Contents

**1. System Overview ..... 15**
**2. Ordering Information ..... 18**
**3. Pin Definitions ..... 19**
**4. QFN-20 Package Specifications ..... 23**
**5. QFN-24 Package Specifications ..... 25**
**6. Electrical Characteristics ..... 27**
6.1. Absolute Maximum Specifications ..... 27
6.2. Electrical Characteristics ..... 28
6.3. Typical Performance Curves ..... 36
**7. 10-Bit ADC (ADC0, C8051F336/8 only) ..... 37**
7.1. Output Code Formatting ..... 38
7.2. Modes of Operation ..... 38
7.2.1. Starting a Conversion ..... 38
7.2.2. Tracking Modes ..... 39
7.2.3. Settling Time Requirements ..... 40
7.3. Programmable Window Detector ..... 44
7.3.1. Window Detector In Single-Ended Mode ..... 46
7.3.2. Window Detector In Differential Mode ..... 47
7.4. ADC0 Analog Multiplexer (C8051F336/8 only) ..... 48
**8. Temperature Sensor (C8051F336/8 only) ..... 51**
**9. 10-Bit Current Mode DAC (IDA0, C8051F336/8 only) ..... 52**
9.1. IDA0 Output Scheduling ..... 52
9.1.1. Update Output On-Demand ..... 52
9.1.2. Update Output Based on Timer Overflow ..... 53
9.1.3. Update Output Based on CNVSTR Edge ..... 53
9.2. IDAC Output Mapping ..... 53
**10. Voltage Reference (C8051F336/8 only) ..... 56**
**11. Comparator0 ..... 58**
11.1. Comparator Multiplexer ..... 63
**12. CIP-51 Microcontroller ..... 65**
12.1. Instruction Set ..... 66
12.1.1. Instruction and CPU Timing ..... 66
12.2. CIP-51 Register Descriptions ..... 71
**13. Memory Organization ..... 74**
13.1. Program Memory ..... 75
13.1.1. MOVX Instruction and Program Memory ..... 75
13.2. Data Memory ..... 75
13.2.1. Internal RAM ..... 75
13.2.1.1. General Purpose Registers ..... 76
13.2.1.2. Bit Addressable Locations ..... 76
13.2.1.3. Stack ..... 76
13.2.2. External RAM ..... 76

**14. Special Function Registers ..... 78**
**15. Interrupts ..... 82**
15.1. MCU Interrupt Sources and Vectors ..... 83
15.1.1. Interrupt Priorities ..... 83
15.1.2. Interrupt Latency ..... 83
15.2. Interrupt Register Descriptions ..... 84
15.3. External Interrupts /INT0 and /INT1 ..... 89
**16. Flash Memory ..... 91**
16.1. Programming The Flash Memory ..... 91
16.1.1. Flash Lock and Key Functions ..... 91
16.1.2. Flash Erase Procedure ..... 91
16.1.3. Flash Write Procedure ..... 92
16.2. Non-volatile Data Storage ..... 92
16.3. Security Options ..... 93
16.4. Flash Write and Erase Guidelines ..... 95
16.4.1. $V_{DD}$ Maintenance and the $V_{DD}$ monitor ..... 95
16.4.2. PSWE Maintenance ..... 95
16.4.3. System Clock ..... 96
**17. Reset Sources ..... 100**
17.1. Power-On Reset ..... 101
17.2. Power-Fail Reset / VDD Monitor ..... 102
17.3. External Reset ..... 103
17.4. Missing Clock Detector Reset ..... 103
17.5. Comparator0 Reset ..... 104
17.6. PCA Watchdog Timer Reset ..... 104
17.7. Flash Error Reset ..... 104
17.8. Software Reset ..... 104
**18. Power Management Modes ..... 106**
18.1. Idle Mode ..... 106
18.2. Stop Mode ..... 107
18.3. Suspend Mode ..... 107
**19. Oscillators and Clock Selection ..... 109**
19.1. System Clock Selection ..... 109
19.2. Programmable Internal High-Frequency (H-F) Oscillator ..... 111
19.2.1. Internal Oscillator Suspend Mode ..... 111
19.3. Programmable Internal Low-Frequency (L-F) Oscillator ..... 113
19.3.1. Calibrating the Internal L-F Oscillator ..... 113
19.4. External Oscillator Drive Circuit ..... 114
19.4.1. External Crystal Example ..... 116
19.4.2. External RC Example ..... 117
19.4.3. External Capacitor Example ..... 118
**20. Port Input/Output ..... 119**
20.1. Port I/O Modes of Operation ..... 120
20.1.1. Port Pins Configured for Analog I/O ..... 120
20.1.2. Port Pins Configured For Digital I/O ..... 120

20.1.3. Interfacing Port I/O to 5V Logic ..... 121
20.2. Assigning Port I/O Pins to Analog and Digital Functions ..... 122
20.2.1. Assigning Port I/O Pins to Analog Functions ..... 122
20.2.2. Assigning Port I/O Pins to Digital Functions ..... 122
20.2.3. Assigning Port I/O Pins to External Event Trigger Functions ..... 123
20.3. Priority Crossbar Decoder ..... 124
20.4. Port I/O Initialization ..... 126
20.5. Port Match ..... 129
20.6. Special Function Registers for Accessing and Configuring Port I/O ..... 131
**21. SMBus ..... 138**
21.1. Supporting Documents ..... 139
21.2. SMBus Configuration ..... 139
21.3. SMBus Operation ..... 139
21.3.1. Transmitter Vs. Receiver ..... 140
21.3.2. Arbitration ..... 140
21.3.3. Clock Low Extension ..... 140
21.3.4. SCL Low Timeout ..... 140
21.3.5. SCL High (SMBus Free) Timeout ..... 141
21.4. Using the SMBus ..... 141
21.4.1. SMBus Configuration Register ..... 141
21.4.2. SMB0CN Control Register ..... 145
21.4.2.1. Software ACK Generation ..... 145
21.4.2.2. Hardware ACK Generation ..... 145
21.4.3. Hardware Slave Address Recognition ..... 147
21.4.4. Data Register ..... 150
21.5. SMBus Transfer Modes ..... 151
21.5.1. Write Sequence (Master) ..... 151
21.5.2. Read Sequence (Master) ..... 152
21.5.3. Write Sequence (Slave) ..... 153
21.5.4. Read Sequence (Slave) ..... 154
21.6. SMBus Status Decoding ..... 154
**22. UART0 ..... 159**
22.1. Enhanced Baud Rate Generation ..... 160
22.2. Operational Modes ..... 161
22.2.1. 8-Bit UART ..... 161
22.2.2. 9-Bit UART ..... 162
22.3. Multiprocessor Communications ..... 163
**23. Enhanced Serial Peripheral Interface (SPI0) ..... 167**
23.1. Signal Descriptions ..... 168
23.1.1. Master Out, Slave In (MOSI) ..... 168
23.1.2. Master In, Slave Out (MISO) ..... 168
23.1.3. Serial Clock (SCK) ..... 168
23.1.4. Slave Select (NSS) ..... 168
23.2. SPI0 Master Mode Operation ..... 169
23.3. SPI0 Slave Mode Operation ..... 170

23.4. SPI0 Interrupt Sources ..... 171
23.5. Serial Clock Phase and Polarity ..... 171
23.6. SPI Special Function Registers ..... 173
**24. Timers ..... 180**
24.1. Timer 0 and Timer 1 ..... 182
24.1.1. Mode 0: 13-bit Counter/Timer ..... 182
24.1.2. Mode 1: 16-bit Counter/Timer ..... 183
24.1.3. Mode 2: 8-bit Counter/Timer with Auto-Reload ..... 184
24.1.4. Mode 3: Two 8-bit Counter/Timers (Timer 0 Only) ..... 185
24.2. Timer 2 ..... 190
24.2.1. 16-bit Timer with Auto-Reload ..... 190
24.2.2. 8-bit Timers with Auto-Reload ..... 191
24.2.3. Low-Frequency Oscillator (LFO) Capture Mode ..... 192
24.3. Timer 3 ..... 196
24.3.1. 16-bit Timer with Auto-Reload ..... 196
24.3.2. 8-bit Timers with Auto-Reload ..... 197
24.3.3. Low-Frequency Oscillator (LFO) Capture Mode ..... 198
**25. Programmable Counter Array ..... 202**
25.1. PCA Counter/Timer ..... 203
25.2. PCA0 Interrupt Sources ..... 204
25.3. Capture/Compare Modules ..... 205
25.3.1. Edge-triggered Capture Mode ..... 206
25.3.2. Software Timer (Compare) Mode ..... 207
25.3.3. High-Speed Output Mode ..... 208
25.3.4. Frequency Output Mode ..... 209
25.3.5. 8-bit, 9-bit, 10-bit and 11-bit Pulse Width Modulator Modes ..... 209
25.3.5.1. 8-bit Pulse Width Modulator Mode ..... 210
25.3.5.2. 9/10/11-bit Pulse Width Modulator Mode ..... 211
25.3.6. 16-Bit Pulse Width Modulator Mode ..... 212
25.4. Watchdog Timer Mode ..... 213
25.4.1. Watchdog Timer Operation ..... 213
25.4.2. Watchdog Timer Usage ..... 214
25.5. Register Descriptions for PCA0 ..... 215
**26. C2 Interface ..... 221**
26.1. C2 Interface Registers ..... 221
26.2. C2 Pin Sharing ..... 224
**Document Change List ..... 225**
**Contact Information ..... 226**

## List of Figures

**1. System Overview**
Figure 1.1. C8051F336/7 Block Diagram ..... 16
Figure 1.2. C8051F338/9 Block Diagram ..... 17
**2. Ordering Information**
**3. Pin Definitions**
Figure 3.1. QFN-20 Pinout Diagram (Top View) ..... 21
Figure 3.2. QFN-24 Pinout Diagram (Top View) ..... 22
**4. QFN-20 Package Specifications**
Figure 4.1. QFN-20 Package Drawing ..... 23
Figure 4.2. QFN-20 Recommended PCB Land Pattern ..... 24
**5. QFN-24 Package Specifications**
Figure 5.1. QFN-24 Package Drawing ..... 25
Figure 5.2. QFN-24 Recommended PCB Land Pattern ..... 26
**6. Electrical Characteristics**
Figure 6.1. Normal Mode Digital Supply Current vs. Frequency ..... 36
Figure 6.2. Idle Mode Digital Supply Current vs. Frequency ..... 36
**7. 10-Bit ADC (ADC0, C8051F336/8 only)**
Figure 7.1. ADC0 Functional Block Diagram ..... 37
Figure 7.2. 10-Bit ADC Track and Conversion Example Timing ..... 39
Figure 7.3. ADC0 Equivalent Input Circuits ..... 40
Figure 7.4. ADC Window Compare Example: Right-Justified Single-Ended Data .. 46
Figure 7.5. ADC Window Compare Example: Left-Justified Single-Ended Data ..... 46
Figure 7.6. ADC Window Compare Example: Right-Justified Differential Data ..... 47
Figure 7.7. ADC Window Compare Example: Left-Justified Differential Data ..... 47
Figure 7.8. ADC0 Multiplexer Block Diagram ..... 48
**8. Temperature Sensor (C8051F336/8 only)**
Figure 8.1. Temperature Sensor Transfer Function ..... 51
**9. 10-Bit Current Mode DAC (IDA0, C8051F336/8 only)**
Figure 9.1. IDA0 Functional Block Diagram ..... 52
Figure 9.2. IDA0 Data Word Mapping ..... 53
**10. Voltage Reference (C8051F336/8 only)**
Figure 10.1. Voltage Reference Functional Block Diagram ..... 56
**11. Comparator0**
Figure 11.1. Comparator0 Functional Block Diagram ..... 58
Figure 11.2. Comparator Hysteresis Plot ..... 59
Figure 11.3. Comparator Input Multiplexer Block Diagram ..... 63
**12. CIP-51 Microcontroller**
Figure 12.1. CIP-51 Block Diagram ..... 65
**13. Memory Organization**
Figure 13.1. C8051F336/7/8/9 Memory Map ..... 74
Figure 13.2. Flash Program Memory Map ..... 75
**14. Special Function Registers**

**15. Interrupts**
**16. Flash Memory**
Figure 16.1. Security Byte Decoding ..... 93
**17. Reset Sources**
Figure 17.1. Reset Sources ..... 100
Figure 17.2. Power-On and VDD Monitor Reset Timing ..... 101
**18. Power Management Modes**
**19. Oscillators and Clock Selection**
Figure 19.1. Oscillator Options ..... 109
Figure 19.2. External 32.768 kHz Quartz Crystal Oscillator Connection Diagram 117
**20. Port Input/Output**
Figure 20.1. Port I/O Functional Block Diagram ..... 119
Figure 20.2. Port I/O Cell Block Diagram ..... 121
Figure 20.3. Port I/O Overdrive Current ..... 121
Figure 20.4. Crossbar Priority Decoder - Possible Pin Assignments ..... 124
Figure 20.5. Crossbar Priority Decoder Example ..... 125
**21. SMBus**
Figure 21.1. SMBus Block Diagram ..... 138
Figure 21.2. Typical SMBus Configuration ..... 139
Figure 21.3. SMBus Transaction ..... 140
Figure 21.4. Typical SMBus SCL Generation ..... 142
Figure 21.5. Typical Master Write Sequence ..... 151
Figure 21.6. Typical Master Read Sequence ..... 152
Figure 21.7. Typical Slave Write Sequence ..... 153
Figure 21.8. Typical Slave Read Sequence ..... 154
**22. UART0**
Figure 22.1. UART0 Block Diagram ..... 159
Figure 22.2. UART0 Baud Rate Logic ..... 160
Figure 22.3. UART Interconnect Diagram ..... 161
Figure 22.4. 8-Bit UART Timing Diagram ..... 161
Figure 22.5. 9-Bit UART Timing Diagram ..... 162
Figure 22.6. UART Multi-Processor Mode Interconnect Diagram ..... 163
**23. Enhanced Serial Peripheral Interface (SPI0)**
Figure 23.1. SPI Block Diagram ..... 167
Figure 23.2. Multiple-Master Mode Connection Diagram ..... 169
Figure 23.3. 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram 170
Figure 23.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram 170
Figure 23.5. Master Mode Data/Clock Timing ..... 172
Figure 23.6. Slave Mode Data/Clock Timing (CKPHA = 0) ..... 172
Figure 23.7. Slave Mode Data/Clock Timing (CKPHA = 1) ..... 173
Figure 23.8. SPI Master Timing (CKPHA = 0) ..... 177
Figure 23.9. SPI Master Timing (CKPHA = 1) ..... 177
Figure 23.10. SPI Slave Timing (CKPHA = 0) ..... 178

Figure 23.11. SPI Slave Timing (CKPHA = 1) ..... 178
**24. Timers**
Figure 24.1. T0 Mode 0 Block Diagram ..... 183
Figure 24.2. T0 Mode 2 Block Diagram ..... 184
Figure 24.3. T0 Mode 3 Block Diagram ..... 185
Figure 24.4. Timer 2 16-Bit Mode Block Diagram ..... 190
Figure 24.5. Timer 2 8-Bit Mode Block Diagram ..... 191
Figure 24.6. Timer 2 Low-Frequency Oscillation Capture Mode Block Diagram ... 192
Figure 24.7. Timer 3 16-Bit Mode Block Diagram ..... 196
Figure 24.8. Timer 3 8-Bit Mode Block Diagram ..... 197
Figure 24.9. Timer 3 Low-Frequency Oscillation Capture Mode Block Diagram ... 198
**25. Programmable Counter Array**
Figure 25.1. PCA Block Diagram ..... 202
Figure 25.2. PCA Counter/Timer Block Diagram ..... 203
Figure 25.3. PCA Interrupt Block Diagram ..... 204
Figure 25.4. PCA Capture Mode Diagram ..... 206
Figure 25.5. PCA Software Timer Mode Diagram ..... 207
Figure 25.6. PCA High-Speed Output Mode Diagram ..... 208
Figure 25.7. PCA Frequency Output Mode ..... 209
Figure 25.8. PCA 8-Bit PWM Mode Diagram ..... 210
Figure 25.9. PCA 9, 10 and 11-Bit PWM Mode Diagram ..... 211
Figure 25.10. PCA 16-Bit PWM Mode ..... 212
Figure 25.11. PCA Module 2 with Watchdog Timer Enabled ..... 213
**26. C2 Interface**
Figure 26.1. Typical C2 Pin Sharing ..... 224

## List of Tables

**1. System Overview**
**2. Ordering Information**
Table 2.1. Product Selection Guide ..... 18
**3. Pin Definitions**
Table 3.1. Pin Definitions for the C8051F336/7/8/9 ..... 19
**4. QFN-20 Package Specifications**
Table 4.1. QFN-20 Package Dimensions ..... 23
Table 4.2. QFN-20 PCB Land Pattern Dimesions ..... 24
**5. QFN-24 Package Specifications**
Table 5.1. QFN-24 Package Dimensions ..... 25
Table 5.2. QFN-24 PCB Land Pattern Dimesions ..... 26
**6. Electrical Characteristics**
Table 6.1. Absolute Maximum Ratings ..... 27
Table 6.2. Global Electrical Characteristics ..... 28
Table 6.3. Port I/O DC Electrical Characteristics ..... 29
Table 6.4. Reset Electrical Characteristics ..... 30
Table 6.5. Flash Electrical Characteristics ..... 30
Table 6.6. Internal High-Frequency Oscillator Electrical Characteristics ..... 31
Table 6.7. Internal Low-Frequency Oscillator Electrical Characteristics ..... 31
Table 6.8. ADC0 Electrical Characteristics ..... 32
Table 6.9. Temperature Sensor Electrical Characteristics ..... 33
Table 6.10. Voltage Reference Electrical Characteristics ..... 33
Table 6.11. IDAC Electrical Characteristics ..... 34
Table 6.12. Comparator Electrical Characteristics ..... 35
**7. 10-Bit ADC (ADC0, C8051F336/8 only)**
**8. Temperature Sensor (C8051F336/8 only)**
**9. 10-Bit Current Mode DAC (IDA0, C8051F336/8 only)**
**10. Voltage Reference (C8051F336/8 only)**
**11. Comparator0**
**12. CIP-51 Microcontroller**
Table 12.1. CIP-51 Instruction Set Summary ..... 67
**13. Memory Organization**
**14. Special Function Registers**
Table 14.1. Special Function Register (SFR) Memory Map ..... 78
Table 14.2. Special Function Registers ..... 79
**15. Interrupts**
Table 15.1. Interrupt Summary ..... 84
**16. Flash Memory**
Table 16.1. Flash Security Summary ..... 94
**17. Reset Sources**
**18. Power Management Modes**
**19. Oscillators and Clock Selection**
**20. Port Input/Output**

Table 20.1. Port I/O Assignment for Analog Functions ..... 122
Table 20.2. Port I/O Assignment for Digital Functions ..... 122
Table 20.3. Port I/O Assignment for External Event Trigger Functions ..... 123
**21. SMBus**
Table 21.1. SMBus Clock Source Selection ..... 142
Table 21.2. Minimum SDA Setup and Hold Times ..... 143
Table 21.3. Sources for Hardware Changes to SMB0CN ..... 147
Table 21.4. Hardware Address Recognition Examples (EHACK = 1) ..... 148
Table 21.5. SMBus Status Decoding With Hardware ACK Generation Disabled (EHACK = 0) ..... 155
Table 21.6. SMBus Status Decoding With Hardware ACK Generation Enabled (EHACK = 1) ..... 157
**22. UART0**
Table 22.1. Timer Settings for Standard Baud Rates Using The Internal 24.5 MHz Oscillator ..... 166
Table 22.2. Timer Settings for Standard Baud Rates Using an External 22.1184 MHz Oscillator ..... 166
**23. Enhanced Serial Peripheral Interface (SPI0)**
Table 23.1. SPI Slave Timing Parameters ..... 179
**24. Timers**
**25. Programmable Counter Array**
Table 25.1. PCA Timebase Input Options ..... 203
Table 25.2. PCA0CPM and PCA0PWM Bit Settings for PCA Capture/Compare Modules ..... 205
Table 25.3. Watchdog Timer Timeout Intervals1 ..... 214
**26. C2 Interface**

## List of Registers

SFR Definition 7.1. ADC0CF: ADC0 Configuration ..... 41
SFR Definition 7.2. ADC0H: ADC0 Data Word MSB ..... 42
SFR Definition 7.3. ADC0L: ADC0 Data Word LSB ..... 42
SFR Definition 7.4. ADC0CN: ADC0 Control ..... 43
SFR Definition 7.5. ADC0GTH: ADC0 Greater-Than Data High Byte ..... 44
SFR Definition 7.6. ADC0GTL: ADC0 Greater-Than Data Low Byte ..... 44
SFR Definition 7.7. ADC0LTH: ADC0 Less-Than Data High Byte ..... 45
SFR Definition 7.8. ADC0LTL: ADC0 Less-Than Data Low Byte ..... 45
SFR Definition 7.9. AMX0P: AMUX0 Positive Channel Select ..... 49
SFR Definition 7.10. AMX0N: AMUX0 Negative Channel Select ..... 50
SFR Definition 9.1. IDA0CN: IDA0 Control ..... 54
SFR Definition 9.2. IDA0H: IDA0 Data Word MSB ..... 55
SFR Definition 9.3. IDA0L: IDA0 Data Word LSB ..... 55
SFR Definition 10.1. REF0CN: Reference Control ..... 57
SFR Definition 11.1. CPT0CN: Comparator0 Control ..... 61
SFR Definition 11.2. CPT0MD: Comparator0 Mode Selection ..... 62
SFR Definition 11.3. CPT0MX: Comparator0 MUX Selection ..... 64
SFR Definition 12.1. DPL: Data Pointer Low Byte ..... 71
SFR Definition 12.2. DPH: Data Pointer High Byte ..... 71
SFR Definition 12.3. SP: Stack Pointer ..... 72
SFR Definition 12.4. ACC: Accumulator ..... 72
SFR Definition 12.5. B: B Register ..... 72
SFR Definition 12.6. PSW: Program Status Word ..... 73
SFR Definition 13.1. EMI0CN: External Memory Interface Control ..... 77
SFR Definition 15.1. IE: Interrupt Enable ..... 85
SFR Definition 15.2. IP: Interrupt Priority ..... 86
SFR Definition 15.3. EIE1: Extended Interrupt Enable 1 ..... 87
SFR Definition 15.4. EIP1: Extended Interrupt Priority 1 ..... 88
SFR Definition 15.5. IT01CF: INT0/INT1 Configuration ..... 90
SFR Definition 16.1. PSCTL: Program Store R/W Control ..... 97
SFR Definition 16.2. FLKEY: Flash Lock and Key ..... 98
SFR Definition 16.3. FLSCL: Flash Scale ..... 99
SFR Definition 17.1. VDM0CN: VDD Monitor Control ..... 103
SFR Definition 17.2. RSTSRC: Reset Source ..... 105
SFR Definition 18.1. PCON: Power Control ..... 108
SFR Definition 19.1. CLKSEL: Clock Select ..... 110
SFR Definition 19.2. OSCICL: Internal H-F Oscillator Calibration ..... 111
SFR Definition 19.3. OSCICN: Internal H-F Oscillator Control ..... 112
SFR Definition 19.4. OSCLCN: Internal L-F Oscillator Control ..... 113
SFR Definition 19.5. OSCXCN: External Oscillator Control ..... 115
SFR Definition 20.1. XBR0: Port I/O Crossbar Register 0 ..... 127
SFR Definition 20.2. XBR1: Port I/O Crossbar Register 1 ..... 128
SFR Definition 20.3. P0MASK: Port 0 Mask Register ..... 129

SFR Definition 20.4. P0MAT: Port 0 Match Register ..... 130
SFR Definition 20.5. P1MASK: Port 1 Mask Register ..... 130
SFR Definition 20.6. P1MAT: Port 1 Match Register ..... 131
SFR Definition 20.7. P0: Port 0 ..... 132
SFR Definition 20.8. P0MDIN: Port 0 Input Mode ..... 132
SFR Definition 20.9. P0MDOUT: Port 0 Output Mode ..... 133
SFR Definition 20.10. P0SKIP: Port 0 Skip ..... 133
SFR Definition 20.11. P1: Port 1 ..... 134
SFR Definition 20.12. P1MDIN: Port 1 Input Mode ..... 134
SFR Definition 20.13. P1MDOUT: Port 1 Output Mode ..... 135
SFR Definition 20.14. P1SKIP: Port 1 Skip ..... 135
SFR Definition 20.15. P2: Port 2 ..... 136
SFR Definition 20.16. P2MDIN: Port 2 Input Mode ..... 136
SFR Definition 20.17. P2MDOUT: Port 2 Output Mode ..... 137
SFR Definition 20.18. P2SKIP: Port 2 Skip ..... 137
SFR Definition 21.1. SMB0CF: SMBus Clock/Configuration ..... 144
SFR Definition 21.2. SMB0CN: SMBus Control ..... 146
SFR Definition 21.3. SMB0ADR: SMBus Slave Address ..... 148
SFR Definition 21.4. SMB0ADM: SMBus Slave Address Mask ..... 149
SFR Definition 21.5. SMB0DAT: SMBus Data ..... 150
SFR Definition 22.1. SCON0: Serial Port 0 Control ..... 164
SFR Definition 22.2. SBUF0: Serial (UART0) Port Data Buffer ..... 165
SFR Definition 23.1. SPI0CFG: SPI0 Configuration ..... 174
SFR Definition 23.2. SPI0CN: SPI0 Control ..... 175
SFR Definition 23.3. SPI0CKR: SPI0 Clock Rate ..... 176
SFR Definition 23.4. SPI0DAT: SPI0 Data ..... 176
SFR Definition 24.1. CKCON: Clock Control ..... 181
SFR Definition 24.2. TCON: Timer Control ..... 186
SFR Definition 24.3. TMOD: Timer Mode ..... 187
SFR Definition 24.4. TL0: Timer 0 Low Byte ..... 188
SFR Definition 24.5. TL1: Timer 1 Low Byte ..... 188
SFR Definition 24.6. TH0: Timer 0 High Byte ..... 189
SFR Definition 24.7. TH1: Timer 1 High Byte ..... 189
SFR Definition 24.8. TMR2CN: Timer 2 Control ..... 193
SFR Definition 24.9. TMR2RLL: Timer 2 Reload Register Low Byte ..... 194
SFR Definition 24.10. TMR2RLH: Timer 2 Reload Register High Byte ..... 194
SFR Definition 24.11. TMR2L: Timer 2 Low Byte ..... 194
SFR Definition 24.12. TMR2H Timer 2 High Byte ..... 195
SFR Definition 24.13. TMR3CN: Timer 3 Control ..... 199
SFR Definition 24.14. TMR3RLL: Timer 3 Reload Register Low Byte ..... 200
SFR Definition 24.15. TMR3RLH: Timer 3 Reload Register High Byte ..... 200
SFR Definition 24.16. TMR3L: Timer 3 Low Byte ..... 200
SFR Definition 24.17. TMR3H Timer 3 High Byte ..... 201
SFR Definition 25.1. PCA0CN: PCA Control ..... 215
SFR Definition 25.2. PCA0MD: PCA Mode ..... 216

SFR Definition 25.3. PCA0PWM: PCA PWM Configuration ..... 217
SFR Definition 25.4. PCA0CPMn: PCA Capture/Compare Mode ..... 218
SFR Definition 25.5. PCA0L: PCA Counter/Timer Low Byte ..... 219
SFR Definition 25.6. PCA0H: PCA Counter/Timer High Byte ..... 219
SFR Definition 25.7. PCA0CPLn: PCA Capture Module Low Byte ..... 220
SFR Definition 25.8. PCA0CPHn: PCA Capture Module High Byte ..... 220
C2 Register Definition 26.1. C2ADD: C2 Address ..... 221
C2 Register Definition 26.2. DEVICEID: C2 Device ID ..... 222
C2 Register Definition 26.3. REVID: C2 Revision ID ..... 222
C2 Register Definition 26.4. FPCTL: C2 Flash Programming Control ..... 223
C2 Register Definition 26.5. FPDAT: C2 Flash Programming Data ..... 223

# 1. System Overview

C8051F336/7/8/9 devices are fully integrated mixed-signal System-on-a-Chip MCUs. Highlighted features are listed below. Refer to **Section "2. Ordering Information" on page 18** for specific product feature selection and part ordering numbers.

- High-speed pipelined 8051-compatible microcontroller core (up to 25 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- True 10-bit 200 ksps 20-channel single-ended/differential ADC with analog multiplexer
- 10-bit Current Output DAC
- Precision programmable 24.5 MHz internal oscillator
- Low-power, low-frequency oscillator
- 16 kB of on-chip Flash memory—512 bytes are reserved
- 768 bytes of on-chip RAM
- SMBus/I2C, Enhanced UART, and Enhanced SPI serial interfaces implemented in hardware
- Four general-purpose 16-bit timers
- Programmable Counter/Timer Array (PCA) with three capture/compare modules and Watchdog Timer function
- On-chip Power-On Reset, $V_{DD}$ Monitor, and Temperature Sensor
- On-chip Voltage Comparator
- 21 or 17 Port I/O (5 V tolerant)
- Low-power suspend mode with fast wake-up time

With on-chip Power-On Reset, $V_{DD}$ monitor, Watchdog Timer, and clock oscillator, the C8051F336/7/8/9 devices are truly stand-alone System-on-a-Chip solutions. The Flash memory can be reprogrammed even in-circuit, providing non-volatile data storage, and also allowing field upgrades of the 8051 firmware. User software has complete control of all peripherals, and may individually shut down any or all peripherals for power savings.

The on-chip Silicon Labs 2-Wire (C2) Development Interface allows non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production MCU installed in the final application. This debug logic supports inspection and modification of memory and registers, setting breakpoints, single stepping, run and halt commands. All analog and digital peripherals are fully functional while debugging using C2. The two C2 interface pins can be shared with user functions, allowing in-system debugging without occupying package pins.

Each device is specified for 2.7 to 3.6 V operation over the industrial temperature range (–40 to +85 °C). The Port I/O and $\overline{\text{RST}}$ pins are tolerant of input signals up to 5 V. The C8051F336/7 are available in a 20-pin QFN package and the C8051F338/9 are available in a 24-pin QFN package. Both package options are lead-free and RoHS compliant. See **Section "2. Ordering Information" on page 18** for ordering information. Block diagrams are included in Figure 1.1 and Figure 1.2.

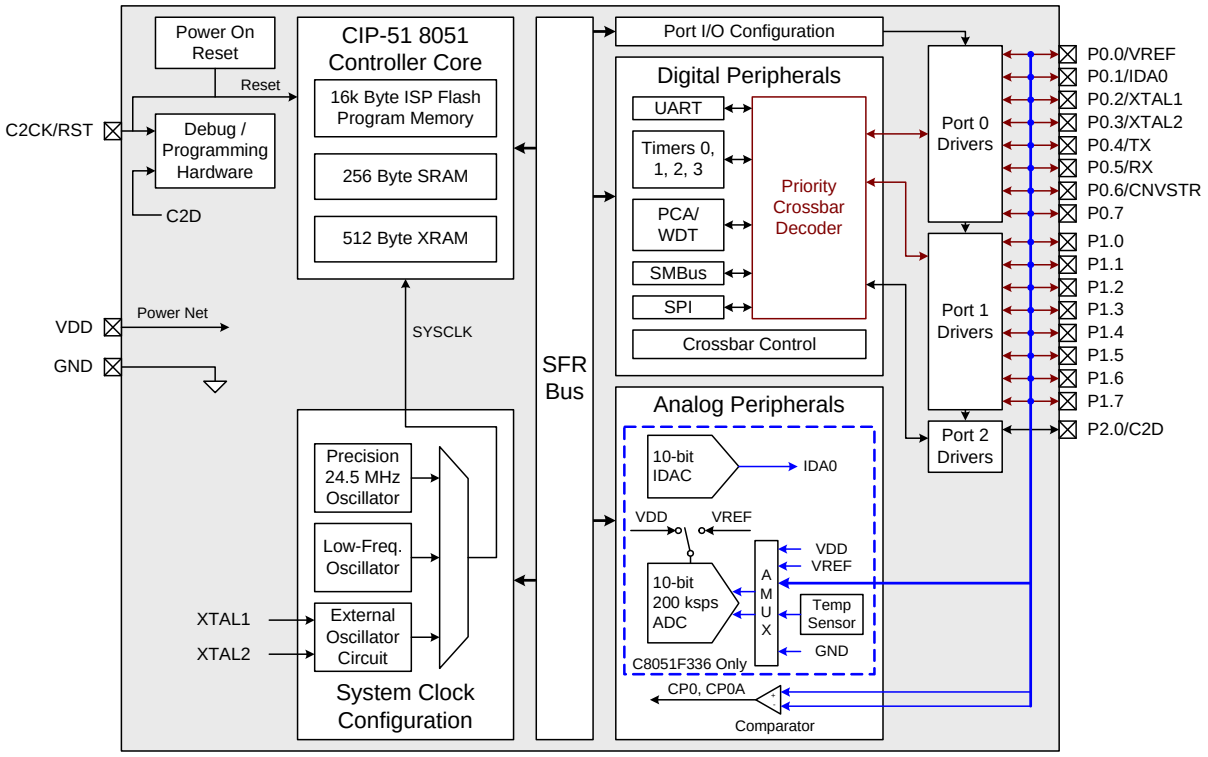



**Figure 1.1. C8051F336/7 Block Diagram**

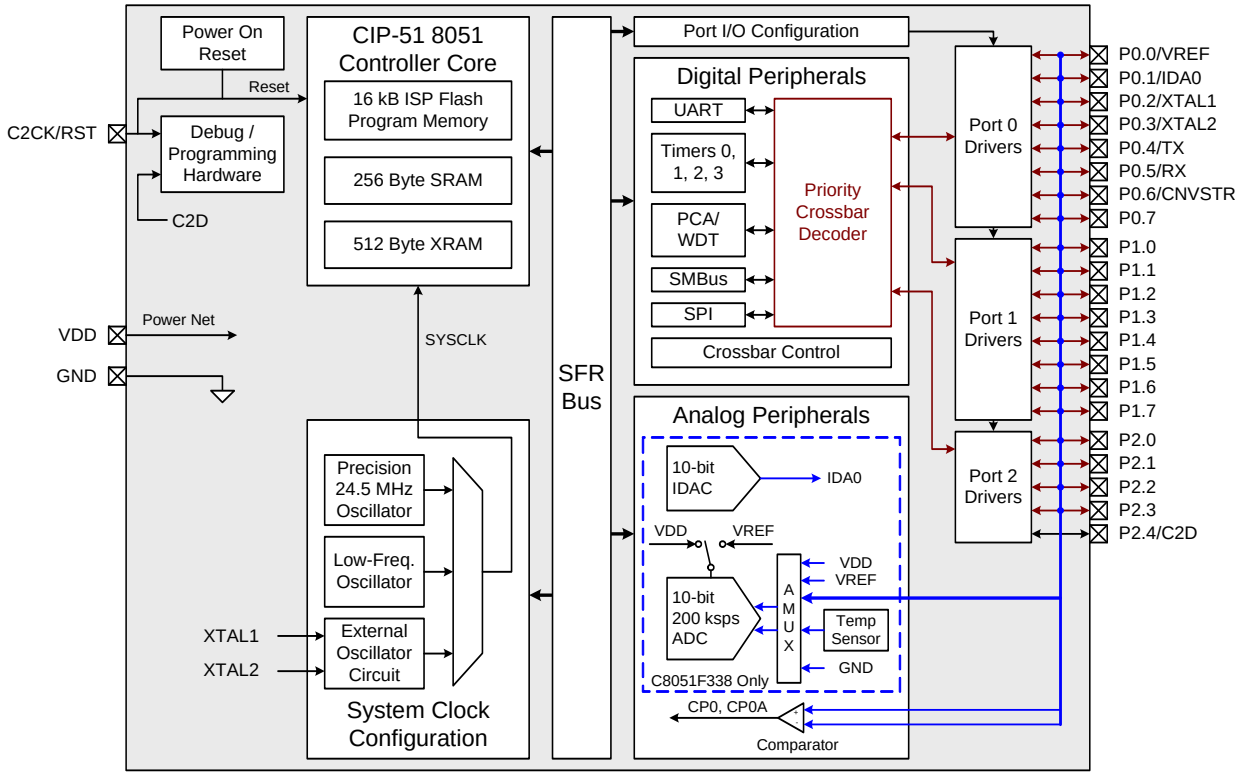



**Figure 1.2. C8051F338/9 Block Diagram**

## 2. Ordering Information

**Table 2.1. Product Selection Guide**

| Ordering Part Number | MIPS (Peak) | Flash Memory (kB) | RAM (bytes) | Calibrated Internal 24.5 MHz Oscillator | Internal 80 kHz Oscillator | SMBus/I2C | Enhanced SPI | UART | Timers (16-bit) | RTC OPeration | Programmable Counter Array | Digital Port I/Os | 10-bit 200ksps ADC | 10-bit Current Output DAC | Internal Voltage Reference | Temperature Sensor | Analog Comparator | Lead-Free / RoHS Compliant | Package |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| C8051F336-GM | 25 | 16 | 768 | Y | Y | Y | Y | Y | 4 | Y | Y | 17 | Y | Y | Y | Y | Y | Y | QFN-20 |
| C8051F337-GM | 25 | 16 | 768 | Y | Y | Y | Y | Y | 4 | Y | Y | 17 | — | — | — | — | Y | Y | QFN-20 |
| C8051F338-GM | 25 | 16 | 768 | Y | Y | Y | Y | Y | 4 | Y | Y | 21 | Y | Y | Y | Y | Y | Y | QFN-24 |
| C8051F339-GM | 25 | 16 | 768 | Y | Y | Y | Y | Y | 4 | Y | Y | 21 | — | — | — | — | Y | Y | QFN-24 |

## 3. Pin Definitions

**Table 3.1. Pin Definitions for the C8051F336/7/8/9**

| Name | Pin 'F336/7 | Pin 'F338/9 | Type | Description |
|---|---|---|---|---|
| $V_{DD}$ | 3 | 4 | | Power Supply Voltage. |
| GND | 2 | 3 | | Ground.<br>This ground connection is required. The center pad may optionally be connected to ground also. |
| $\overline{RST}$/ | 4 | 5 | D I/O | Device Reset. Open-drain output of internal POR or $V_{DD}$ monitor. An external source can initiate a system reset by driving this pin low for at least 10 µs. |
| C2CK | | | D I/O | Clock signal for the C2 Debug Interface. |
| C2D | 5 | 6 | D I/O | Bi-directional data signal for the C2 Debug Interface. Shared with P2.0 on 20-pin packaging and P2.4 on 24-pin packaging. |
| P0.0/ | 1 | 2 | D I/O or A In | Port 0.0. |
| VREF | | | A In | External VREF input. |
| P0.1 | 20 | 1 | D I/O or A In | Port 0.1. |
| IDA0 | | | A Out | IDA0 Output. |
| P0.2/ | 19 | 24 | D I/O or A In | Port 0.2. |
| XTAL1 | | | A In | External Clock Input. This pin is the external oscillator return for a crystal or resonator. |
| P0.3/ | 18 | 23 | D I/O or A In | Port 0.3. |
| XTAL2 | | | A I/O or D In | External Clock Output. For an external crystal or resonator, this pin is the excitation driver. This pin is the external clock input for CMOS, capacitor, or RC oscillator configurations. |
| P0.4 | 17 | 22 | D I/O or A In | Port 0.4. |
| P0.5 | 16 | 21 | D I/O or A In | Port 0.5. |
| P0.6/ | 15 | 20 | D I/O or A In | Port 0.6. |
| CNVSTR | | | D In | ADC0 External Convert Start or IDA0 Update Source Input. |

**Table 3.1. Pin Definitions for the C8051F336/7/8/9 (Continued)**

| Name | Pin 'F336/7 | Pin 'F338/9 | Type | Description |
|---|---|---|---|---|
| P0.7 | 14 | 19 | D I/O or A In | Port 0.7. |
| P1.0 | 13 | 18 | D I/O or A In | Port 1.0. |
| P1.1 | 12 | 17 | D I/O or A In | Port 1.1. |
| P1.2 | 11 | 16 | D I/O or A In | Port 1.2. |
| P1.3 | 10 | 15 | D I/O or A In | Port 1.3. |
| P1.4 | 9 | 14 | D I/O or A In | Port 1.4. |
| P1.5 | 8 | 13 | D I/O or A In | Port 1.5. |
| P1.6 | 7 | 12 | D I/O or A In | Port 1.6. |
| P1.7 | 6 | 11 | D I/O or A In | Port 1.7. |
| P2.0 | 5 | 10 | D I/O or A In | Port 2.0. (Also C2D on 20-pin Packaging) |
| P2.1 | — | 9 | D I/O or A In | Port 2.1. |
| P2.2 | — | 8 | D I/O or A In | Port 2.2. |
| P2.3 | — | 7 | D I/O or A In | Port 2.3. |
| P2.4 | — | 6 | D I/O | Port 2.4. (Also C2D on 24-pin Packaging) |

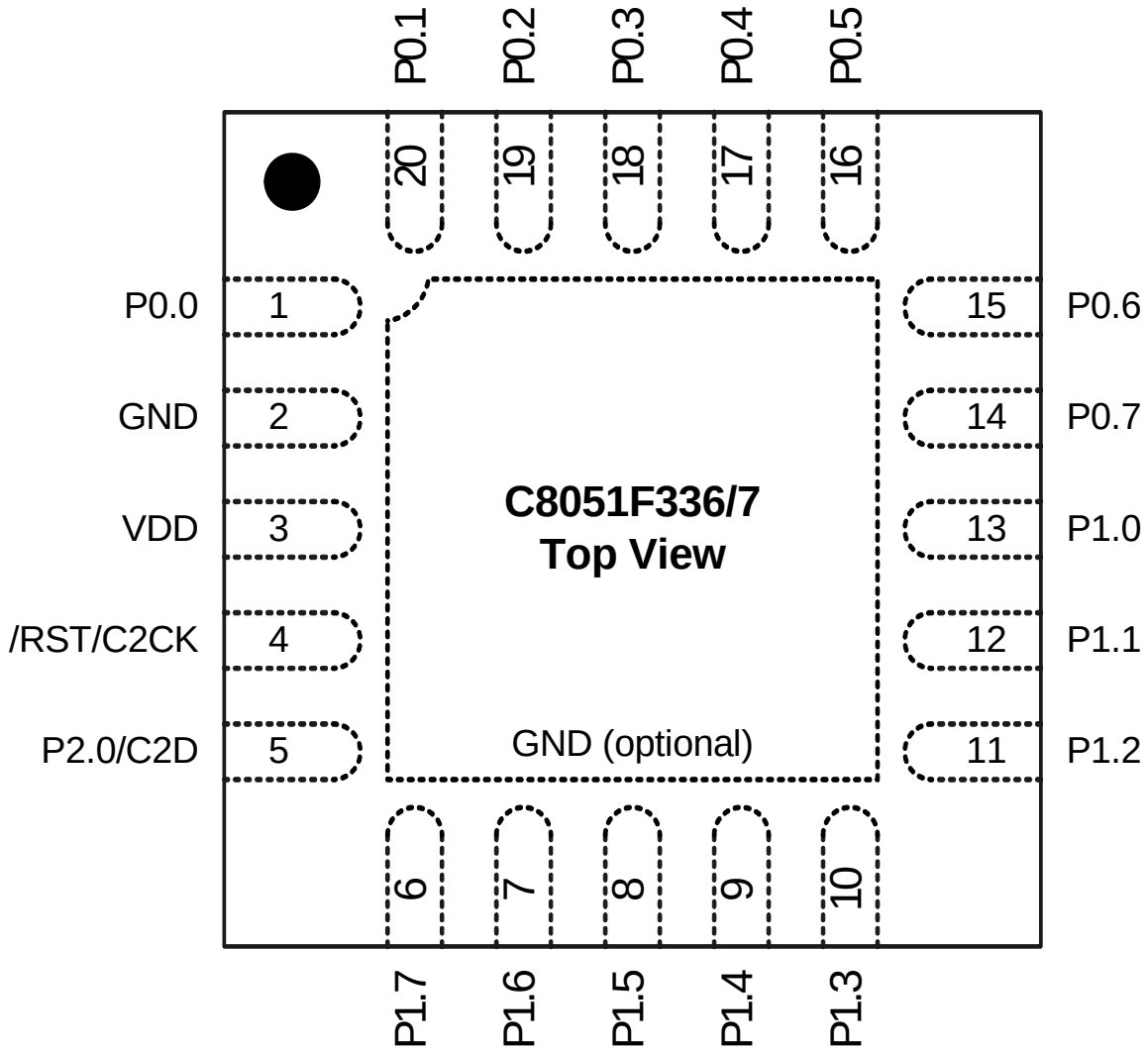



**Figure 3.1. QFN-20 Pinout Diagram (Top View)**

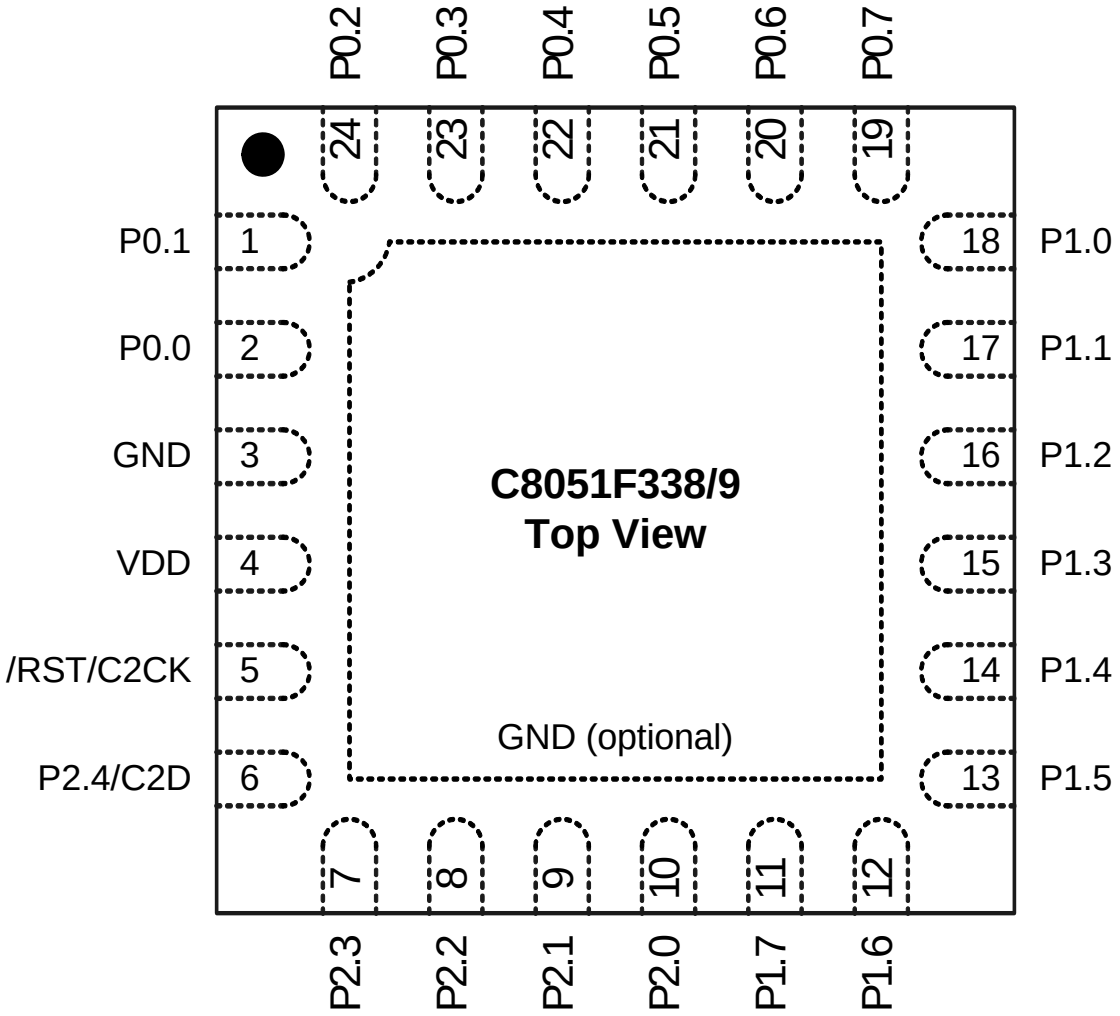



**Figure 3.2. QFN-24 Pinout Diagram (Top View)**

# 4. QFN-20 Package Specifications

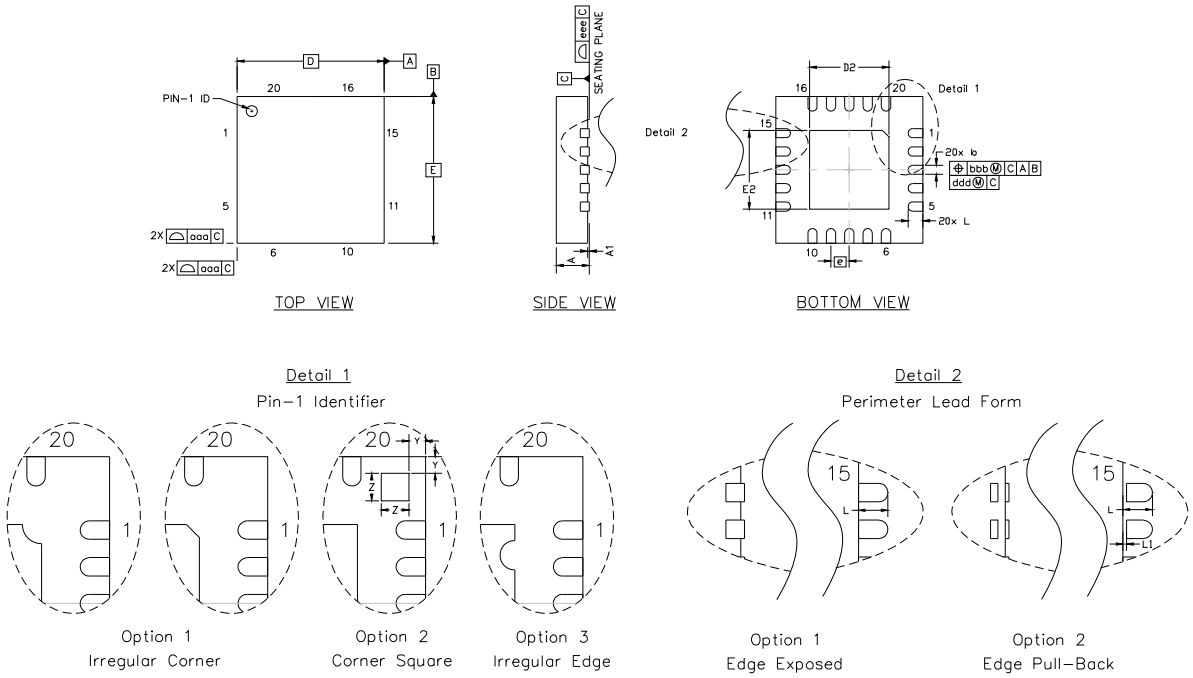



**Figure 4.1. QFN-20 Package Drawing**

**Table 4.1. QFN-20 Package Dimensions**

| Dimension | Min | Typ | Max |
|---|---|---|---|
| A | 0.80 | 0.90 | 1.00 |
| A1 | 0.00 | 0.02 | 0.05 |
| b | 0.18 | 0.25 | 0.30 |
| D | 4.00 BSC. | | |
| D2 | 2.00 | 2.15 | 2.25 |
| e | 0.50 BSC. | | |
| E | 4.00 BSC. | | |
| E2 | 2.00 | 2.15 | 2.25 |

| Dimension | Min | Typ | Max |
|---|---|---|---|
| L | 0.45 | 0.55 | 0.65 |
| L1 | 0.00 | — | 0.15 |
| aaa | — | — | 0.15 |
| bbb | — | — | 0.10 |
| ddd | — | — | 0.05 |
| eee | — | — | 0.08 |
| Z | — | 0.43 | — |
| Y | — | 0.18 | — |

**Notes:**

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing per ANSI Y14.5M-1994.
3. This drawing conforms to the JEDEC Solid State Outline MO-220, variation VGGD except for custom features D2, E2, Z, Y, and L which are toleranced per supplier designation.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020C specification for Small Body Components.

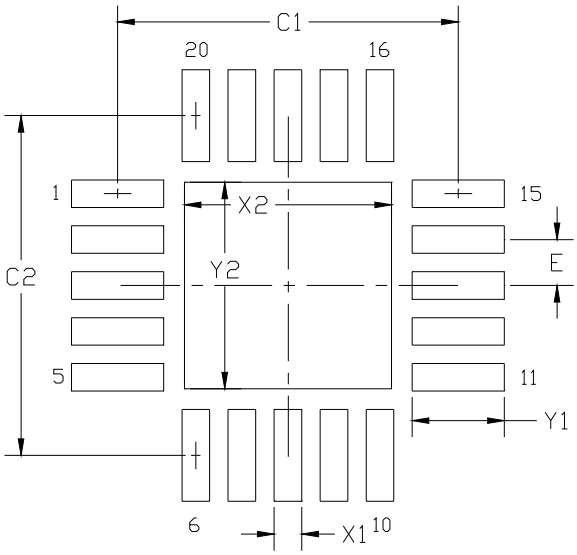



**Figure 4.2. QFN-20 Recommended PCB Land Pattern**

**Table 4.2. QFN-20 PCB Land Pattern Dimesions**

| Dimension | Min | Max |
|---|---|---|
| C1 | 3.70 | |
| C2 | 3.70 | |
| E | 0.50 | |
| X1 | 0.20 | 0.30 |

| Dimension | Min | Max |
|---|---|---|
| X2 | 2.15 | 2.25 |
| Y1 | 0.90 | 1.00 |
| Y2 | 2.15 | 2.25 |

**Notes:**

General

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing is per the ANSI Y14.5M-1994 specification.
3. This Land Pattern Design is based on the IPC-7351 guidelines.

Solder Mask Design

4. All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60µm minimum, all the way around the pad.

Stencil Design

5. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
6. The stencil thickness should be 0.125mm (5 mils).
7. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pins.
8. A 2x2 array of 0.95mm openings on a 1.1mm pitch should be used for the center pad to assure the proper paste volume (71% Paste Coverage).

Card Assembly

9. A No-Clean, Type-3 solder paste is recommended.
10. The recommended card reflow profile is per the JEDEC/IPC J-STD-020C specification for Small Body Components.

## 5. QFN-24 Package Specifications

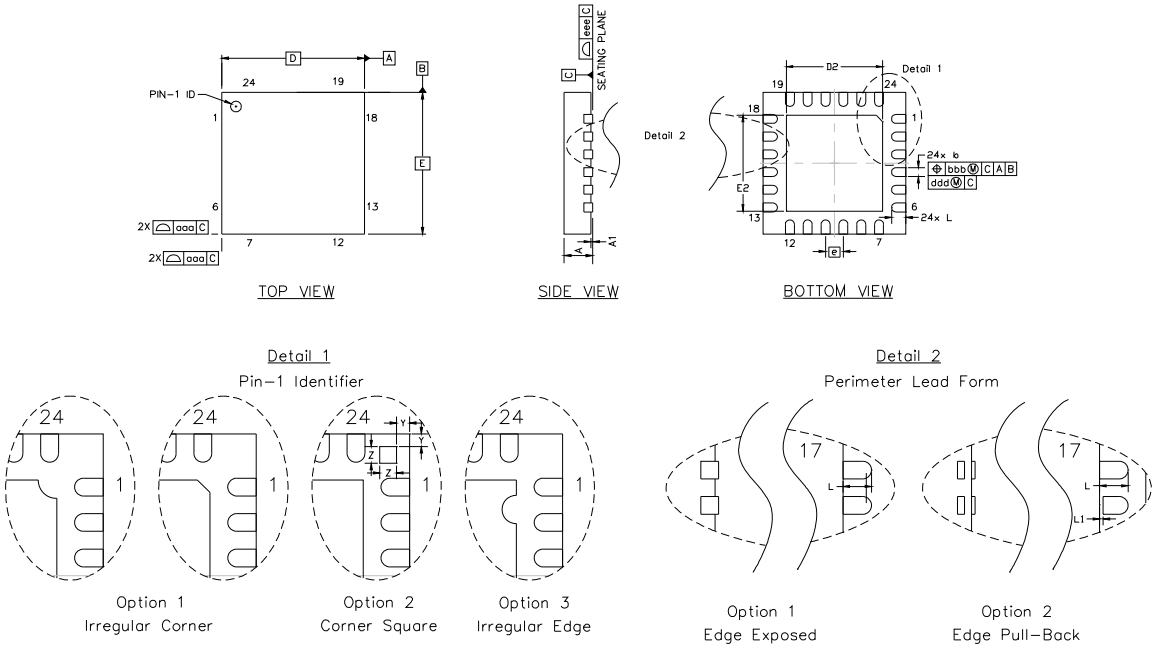



**Figure 5.1. QFN-24 Package Drawing**

**Table 5.1. QFN-24 Package Dimensions**

| Dimension | Min | Typ | Max | | Dimension | Min | Typ | Max |
|---|---|---|---|---|---|---|---|---|
| A | 0.70 | 0.75 | 0.80 | | L | 0.30 | 0.40 | 0.50 |
| A1 | 0.00 | 0.02 | 0.05 | | L1 | 0.00 | — | 0.15 |
| b | 0.18 | 0.25 | 0.30 | | aaa | — | — | 0.15 |
| D | | 4.00 BSC. | | | bbb | — | — | 0.10 |
| D2 | 2.55 | 2.70 | 2.80 | | ddd | — | — | 0.05 |
| e | | 0.50 BSC. | | | eee | — | — | 0.08 |
| E | | 4.00 BSC. | | | Z | — | 0.24 | — |
| E2 | 2.55 | 2.70 | 2.80 | | Y | — | 0.18 | — |

**Notes:**

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing per ANSI Y14.5M-1994.
3. This drawing conforms to JEDEC Solid State Outline MO-220, variation WGGD except for custom features D2, E2, Z, Y, and L which are toleranced per supplier designation.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020C specification for Small Body Components.

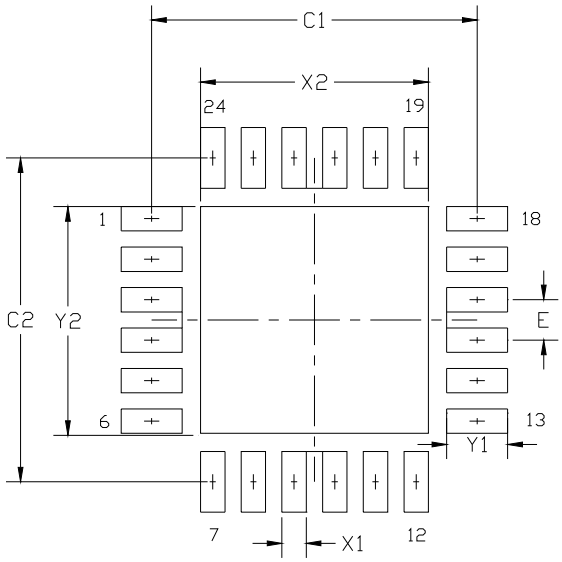



**Figure 5.2. QFN-24 Recommended PCB Land Pattern**

**Table 5.2. QFN-24 PCB Land Pattern Dimesions**

| Dimension | Min | Max |
|---|---|---|
| C1 | 3.90 | 4.00 |
| C2 | 3.90 | 4.00 |
| E | 0.50 BSC | |
| X1 | 0.20 | 0.30 |

| Dimension | Min | Max |
|---|---|---|
| X2 | 2.70 | 2.80 |
| Y1 | 0.65 | 0.75 |
| Y2 | 2.70 | 2.80 |

**Notes:**

General

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. This Land Pattern Design is based on the IPC-7351 guidelines.

Solder Mask Design

3. All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60µm minimum, all the way around the pad.

Stencil Design

4. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
5. The stencil thickness should be 0.125mm (5 mils).
6. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.
7. A 2x2 array of 1.10mm x 1.10mm openings on a 1.30mm pitch should be used for the center pad.

Card Assembly

8. A No-Clean, Type-3 solder paste is recommended.
9. The recommended card reflow profile is per the JEDEC/IPC J-STD-020C specification for Small Body Components.

# 6. Electrical Characteristics

## 6.1. Absolute Maximum Specifications

### Table 6.1. Absolute Maximum Ratings

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Ambient temperature under bias | | –55 | — | 125 | °C |
| Storage Temperature | | –65 | — | 150 | °C |
| Voltage on any Port I/O Pin or $\overline{\text{RST}}$ with respect to GND | | –0.3 | — | 5.8 | V |
| Voltage on $V_{DD}$ with respect to GND | | –0.3 | — | 4.2 | V |
| Maximum Total current through $V_{DD}$ or GND | | — | — | 500 | mA |
| Maximum output current sunk by $\overline{\text{RST}}$ or any Port pin | | — | — | 100 | mA |

**Note:** Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

## 6.2. Electrical Characteristics

### Table 6.2. Global Electrical Characteristics

–40 to +85 °C, 25 MHz system clock unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Digital Supply Voltage | Normal Operation | $V_{RST}$[1] | 3.0 | 3.6 | V |
| | Writing or Erasing Flash Memory | 2.7 | 3.0 | 3.6 | V |
| Digital Supply RAM Data Retention Voltage | | — | 1.5 | — | V |
| SYSCLK (System Clock) (Note 2) | | 0 | — | 25 | MHz |
| $T_{SYSH}$ (SYSCLK High Time) | | 18 | — | — | ns |
| $T_{SYSL}$ (SYSCLK Low Time) | | 18 | — | — | ns |
| Specified Operating Temperature Range | | –40 | — | +85 | °C |
| **Digital Supply Current—CPU Active (Normal Mode, fetching instructions from Flash)** | | | | | |
| $I_{DD}$ (Note 3) | $V_{DD}$ = 3.6 V, F = 25 MHz | — | 11.0 | 13.0 | mA |
| | $V_{DD}$ = 3.0 V, F = 25 MHz | — | 8.0 | 10.0 | mA |
| | $V_{DD}$ = 3.6 V, F = 1 MHz | — | 0.56 | — | mA |
| | $V_{DD}$ = 3.0 V, F = 1 MHz | — | 0.42 | — | mA |
| | $V_{DD}$ = 3.0 V, F = 80 kHz | — | 35 | — | µA |
| **Digital Supply Current—CPU Inactive (Idle Mode, not fetching instructions from Flash)** | | | | | |
| $I_{DD}$ (Note 3) | $V_{DD}$ = 3.6 V, F = 25 MHz | — | 5.0 | 6.0 | mA |
| | $V_{DD}$ = 3.0 V, F = 25 MHz | — | 4.1 | 5.0 | mA |
| | $V_{DD}$ = 3.6 V, F = 1 MHz | — | 0.20 | — | mA |
| | $V_{DD}$ = 3.0 V, F = 1 MHz | — | 0.16 | — | mA |
| | $V_{DD}$ = 3.0 V, F = 80 kHz | — | 13 | — | µA |
| Digital Supply Current (Stop or Suspend Mode, shut-down) | Oscillator not running, $V_{DD}$ Monitor Disabled | — | < 0.1 | — | µA |

**Notes:**
1. Given in Table 6.4 on page 30.
2. SYSCLK must be at least 32 kHz to enable debugging.
3. Based on device characterization data; Not production tested.

## Table 6.3. Port I/O DC Electrical Characteristics

$V_{DD}$ = 2.7 to 3.6 V, –40 to +85 °C unless otherwise specified.

| Parameters | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Output High Voltage | $I_{OH}$ = –3 mA, Port I/O push-pull | $V_{DD}$ – 0.7 | — | — | V |
| | $I_{OH}$ = –10 µA, Port I/O push-pull | $V_{DD}$ – 0.1 | — | — | V |
| | $I_{OH}$ = –10 mA, Port I/O push-pull | — | $V_{DD}$ – 0.8 | — | V |
| Output Low Voltage | $I_{OL}$ = 8.5 mA | — | — | 0.6 | V |
| | $I_{OL}$ = 10 µA | — | — | 0.1 | V |
| | $I_{OL}$ = 25 mA | — | 1.0 | — | V |
| Input High Voltage | | 2.0 | — | — | V |
| Input Low Voltage | | — | — | 0.8 | V |
| Input Leakage Current | Weak Pullup Off | — | — | ±1 | µA |
| | Weak Pullup On, $V_{IN}$ = 0 V | — | 50 | 100 | µA |

### Table 6.4. Reset Electrical Characteristics

–40 to +85 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $\overline{\text{RST}}$ Output Low Voltage | $I_{OL}$ = 8.5 mA, $V_{DD}$ = 2.7 V to 3.6 V | — | — | 0.6 | V |
| $\overline{\text{RST}}$ Input Low Voltage | | — | — | 0.6 | |
| $\overline{\text{RST}}$ Input Pullup Current | $\overline{\text{RST}}$ = 0.0 V | — | 50 | 100 | µA |
| $V_{DD}$ POR Threshold ($V_{RST}$) | | 2.40 | 2.55 | 2.70 | V |
| Missing Clock Detector Time-out | Time from last system clock rising edge to reset initiation | 100 | 220 | 600 | µs |
| Reset Time Delay | Delay between release of any reset source and code execution at location 0x0000 | — | — | 40 | µs |
| Minimum $\overline{\text{RST}}$ Low Time to Generate a System Reset | | 15 | — | — | µs |
| $V_{DD}$ Monitor Turn-on Time | | 100 | — | — | µs |
| $V_{DD}$ Monitor Supply Current | | — | 20 | 40 | µA |

### Table 6.5. Flash Electrical Characteristics

$V_{DD}$ = 2.7 to 3.6 V; –40 to +85 ºC unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Flash Size | | 16384* | — | — | bytes |
| Endurance | | 20 k | 100 k | — | Erase/Write |
| Erase Cycle Time | 25 MHz System Clock | 10 | 15 | 20 | ms |
| Write Cycle Time | 25 MHz System Clock | 40 | 55 | 70 | µs |
| **Note:** 512 bytes at addresses 0x3E00 to 0x3FFF are reserved. | | | | | |

## Table 6.6. Internal High-Frequency Oscillator Electrical Characteristics

$V_{DD}$ = 2.7 to 3.6 V; $T_A$ = –40 to +85 °C unless otherwise specified; Using factory-calibrated settings.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Oscillator Frequency | IFCN = 11b | 24 | 24.5 | 25 | MHz |
| Oscillator Supply Current (from $V_{DD}$) | 25 °C, $V_{DD}$ = 3.0 V, OSCICN.7 = 1, OCSICN.5 = 0 | — | 450 | 600 | µA |
| Power Supply Sensitivity | Constant Temperature | — | 0.12 | — | %/V |
| Temperature Sensitivity | Constant Supply | — | 60 | — | ppm/°C |

## Table 6.7. Internal Low-Frequency Oscillator Electrical Characteristics

$V_{DD}$ = 2.7 to 3.6 V; $T_A$ = –40 to +85 °C unless otherwise specified; Using factory-calibrated settings.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Oscillator Frequency | OSCLD = 11b | 72 | 80 | 88 | kHz |
| Oscillator Supply Current (from $V_{DD}$) | 25 °C, $V_{DD}$ = 3.0 V, OSCLCN.7 = 1 | — | 5.5 | 10 | µA |
| Power Supply Sensitivity | Constant Temperature | — | 2.4 | — | %/V |
| Temperature Sensitivity | Constant Supply | — | 30 | — | ppm/°C |

## Table 6.8. ADC0 Electrical Characteristics

$V_{DD}$ = 3.0 V, VREF = 2.40 V (REFSL=0), –40 to +85 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| **DC Accuracy** | | | | | |
| Resolution | | | 10 | | bits |
| Integral Nonlinearity | | — | ±0.5 | ±1 | LSB |
| Differential Nonlinearity | Guaranteed Monotonic | — | ±0.5 | ±1 | LSB |
| Offset Error | | –12 | 3 | 12 | LSB |
| Full Scale Error | | –5 | 1 | 5 | LSB |
| Offset Temperature Coefficient | | — | 3 | — | ppm/°C |
| **Dynamic performance (10 kHz sine-wave single-ended input, 1 dB below Full Scale, 200 ksps)** | | | | | |
| Signal-to-Noise Plus Distortion | | 53 | 58 | — | dB |
| Total Harmonic Distortion | Up to the 5th harmonic | — | –75 | — | dB |
| Spurious-Free Dynamic Range | | — | 75 | — | dB |
| **Conversion Rate** | | | | | |
| SAR Conversion Clock | | — | — | 3.125 | MHz |
| Conversion Time in SAR Clocks | | 13 | — | — | clocks |
| Track/Hold Acquisition Time | | 300 | — | — | ns |
| Throughput Rate | | — | — | 200 | ksps |
| **Analog Inputs** | | | | | |
| ADC Input Voltage Range | Single Ended (AIN+ – GND)<br>Differential (AIN+ – AIN–) | 0<br>–VREF | — | VREF<br>VREF | V<br>V |
| Absolute Pin Voltage with respect to GND | Single Ended or Differential | 0 | — | $V_{DD}$ | V |
| Sampling Capacitance ($C_{SAMPLE}$) | | — | 5 | — | pF |
| Input Multiplexer Impedance ($R_{MUX}$) | | — | 5 | — | kΩ |
| **Power Specifications** | | | | | |
| Power Supply Current ($V_{DD}$ supplied to ADC0) | Operating Mode, 200 ksps | — | 500 | 900 | µA |
| Power Supply Rejection | | — | 3 | — | mV/V |

## Table 6.9. Temperature Sensor Electrical Characteristics

$V_{DD}$ = 3.0 V, –40 to +85 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Linearity | | — | ± 0.2 | — | °C |
| Slope | | — | 2.25 | — | mV/°C |
| Slope Error* | | — | 23 | — | µV/°C |
| Offset | Temp = 0 °C | — | 785 | — | mV |
| Offset Error* | Temp = 0 °C | — | 11.6 | — | mV |
| **Note:** Represents one standard deviation from the mean. | | | | | |

## Table 6.10. Voltage Reference Electrical Characteristics

$V_{DD}$ = 3.0 V; –40 to +85 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| **Internal Reference (REFBE = 1)** | | | | | |
| Output Voltage | 25 °C ambient | 2.35 | 2.42 | 2.50 | V |
| VREF Short-Circuit Current | | — | — | 10 | mA |
| VREF Temperature Coefficient | | — | 30 | — | ppm/°C |
| Load Regulation | Load = 0 to 200 µA to AGND | — | 3 | — | µV/µA |
| VREF Turn-on Time 1 | 4.7 µF tantalum, 0.1 µF ceramic bypass | — | 7.5 | — | ms |
| VREF Turn-on Time 2 | 0.1 µF ceramic bypass | — | 200 | — | µs |
| Power Supply Rejection | | — | –0.6 | — | mV/V |
| **External Reference (REFBE = 0)** | | | | | |
| Input Voltage Range | | 0 | — | $V_{DD}$ | V |
| Input Current | Sample Rate = 200 ksps; VREF = 3.0 V | — | 3 | — | µA |
| **Power Specifications** | | | | | |
| Reference Bias Generator | REFBE = ‘1’ or TEMPE = ‘1’ | — | 30 | 50 | µA |

## Table 6.11. IDAC Electrical Characteristics

–40 to +85 °C, $V_{DD}$ = 3.0 V Full-scale output current set to 2 mA unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| **Static Performance** | | | | | |
| Resolution | | | 10 | | bits |
| Integral Nonlinearity | | — | ±0.5 | ±2 | LSB |
| Differential Nonlinearity | Guaranteed Monotonic | — | ±0.5 | ±1 | LSB |
| Output Compliance Range | | 0 | — | $V_{DD}$ – 1.2 | V |
| Offset Error | | — | 0 | — | µA |
| Full Scale Error | 2 mA Full Scale Output Current | — | 0 | ±30 | µA |
| Full Scale Error Tempco | | — | 30 | — | ppm/°C |
| $V_{DD}$ Power Supply Rejection Ratio | | — | 6 | — | µA/V |
| **Dynamic Performance** | | | | | |
| Output Settling Time to 1/2 LSB | IDA0H:L = 0x3FF to 0x000 | — | 5 | — | µs |
| Startup Time | | — | 5 | — | µs |
| Gain Variation | 1 mA Full Scale Output Current | — | ±1 | — | % |
| | 0.5 mA Full Scale Output Current | — | ±1 | — | % |
| **Power Consumption** | | | | | |
| Power Supply Current ($V_{DD}$ supplied to IDAC) | 2 mA Full Scale Output Current | — | 2100 | — | µA |
| | 1 mA Full Scale Output Current | — | 1100 | — | µA |
| | 0.5 mA Full Scale Output Current | — | 600 | — | µA |

## Table 6.12. Comparator Electrical Characteristics

$V_{DD}$ = 3.0 V, –40 to +85 °C unless otherwise noted.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Response Time<br>Mode 0, Vcm* = 1.5 V | CP0+ – CP0– = 100 mV<br>CP0+ – CP0– = –100 mV | —<br>— | 100<br>200 | —<br>— | ns<br>ns |
| Response Time<br>Mode 1, Vcm* = 1.5 V | CP0+ – CP0– = 100 mV<br>CP0+ – CP0– = –100 mV | —<br>— | 250<br>350 | —<br>— | ns<br>ns |
| Response Time<br>Mode 2, Vcm* = 1.5 V | CP0+ – CP0– = 100 mV<br>CP0+ – CP0– = –100 mV | —<br>— | 400<br>800 | —<br>— | ns<br>ns |
| Response Time<br>Mode 3, Vcm* = 1.5 V | CP0+ – CP0– = 100 mV<br>CP0+ – CP0– = –100 mV | —<br>— | 1100<br>5000 | —<br>— | ns<br>ns |
| Common-Mode Rejection Ratio | | — | 1.25 | 5 | mV/V |
| Positive Hysteresis 1 | CP0HYP1–0 = 00 | — | 0 | 1 | mV |
| Positive Hysteresis 2 | CP0HYP1–0 = 01 | 1 | 5 | 10 | mV |
| Positive Hysteresis 3 | CP0HYP1–0 = 10 | 6 | 10 | 20 | mV |
| Positive Hysteresis 4 | CP0HYP1–0 = 11 | 12 | 20 | 30 | mV |
| Negative Hysteresis 1 | CP0HYN1–0 = 00 | | 0 | 1 | mV |
| Negative Hysteresis 2 | CP0HYN1–0 = 01 | 1 | 5 | 10 | mV |
| Negative Hysteresis 3 | CP0HYN1–0 = 10 | 6 | 10 | 20 | mV |
| Negative Hysteresis 4 | CP0HYN1–0 = 11 | 12 | 20 | 30 | mV |
| Inverting or Non-Inverting Input Voltage Range | | –0.25 | — | $V_{DD}$ + 0.25 | V |
| Input Capacitance | | — | 4 | — | pF |
| Input Bias Current | | — | 0.001 | — | nA |
| Input Offset Voltage | | –5 | — | +5 | mV |
| **Power Supply** | | | | | |
| Power Supply Rejection | | — | 0.1 | — | mV/V |
| Power-up Time | | — | 10 | — | µs |
| Supply Current at DC | Mode 0<br>Mode 1<br>Mode 2<br>Mode 3 | —<br>—<br>—<br>— | 10<br>4<br>2<br>0.4 | 20<br>10<br>5<br>2.5 | µA<br>µA<br>µA<br>µA |

**Note:** Vcm is the common-mode voltage on CP0+ and CP0–.

## 6.3. Typical Performance Curves

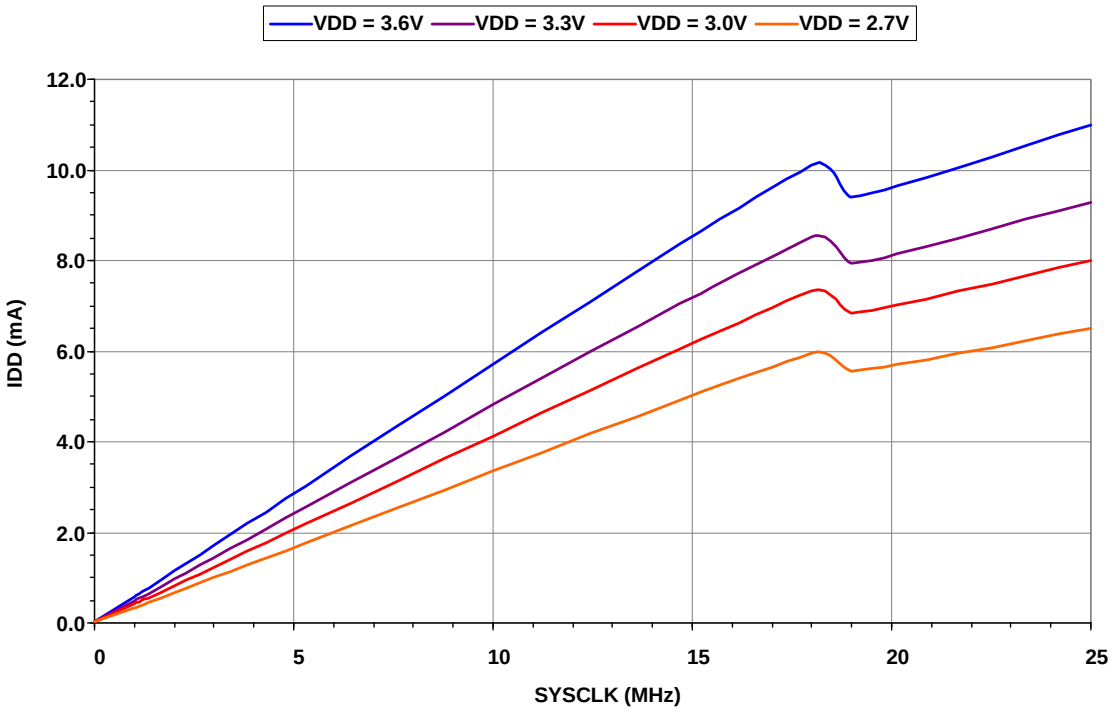



**Figure 6.1. Normal Mode Digital Supply Current vs. Frequency**

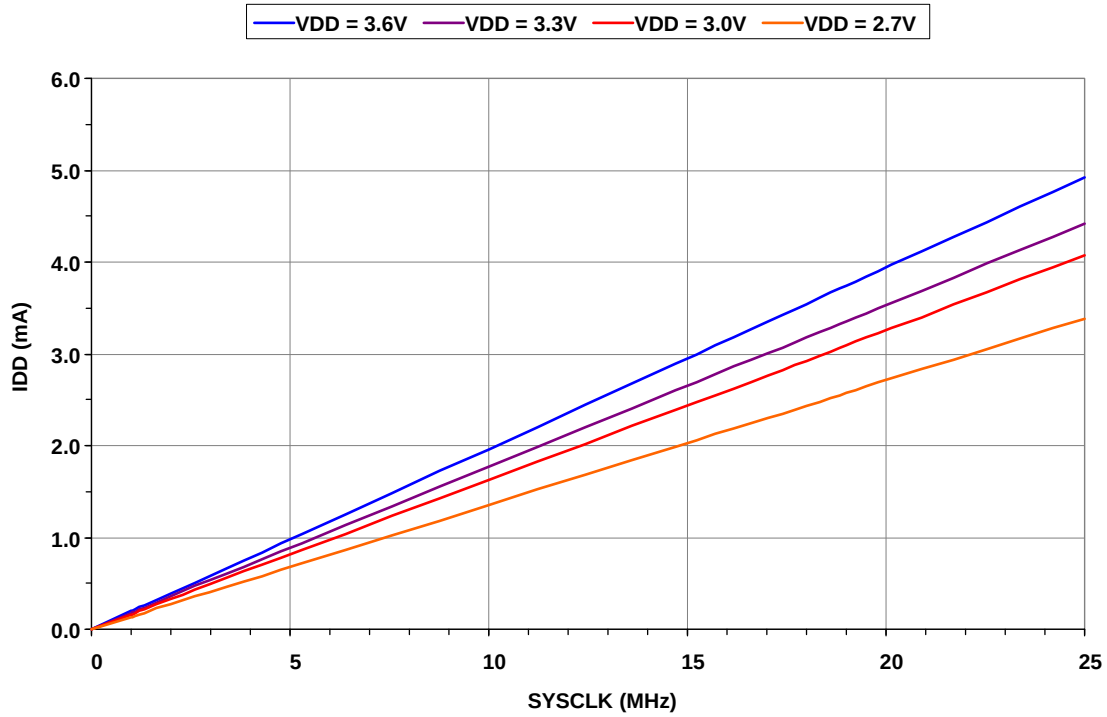



**Figure 6.2. Idle Mode Digital Supply Current vs. Frequency**

## 7. 10-Bit ADC (ADC0, C8051F336/8 only)

The ADC0 on the C8051F336/8 is a 200 ksps, 10-bit successive-approximation-register (SAR) ADC with integrated track-and-hold and programmable window detector. The ADC is fully configurable under software control via Special Function Registers. The ADC0 operates in both Single-ended and Differential modes, and may be configured to measure various different signals using the analog multiplexer described in **Section "7.4. ADC0 Analog Multiplexer (C8051F336/8 only)" on page 48**. The voltage reference for the ADC is selected as described in **Section "8. Temperature Sensor (C8051F336/8 only)" on page 51**. The ADC0 subsystem is enabled only when the AD0EN bit in the ADC0 Control register (ADC0CN) is set to logic 1. The ADC0 subsystem is in low power shutdown when this bit is logic 0.

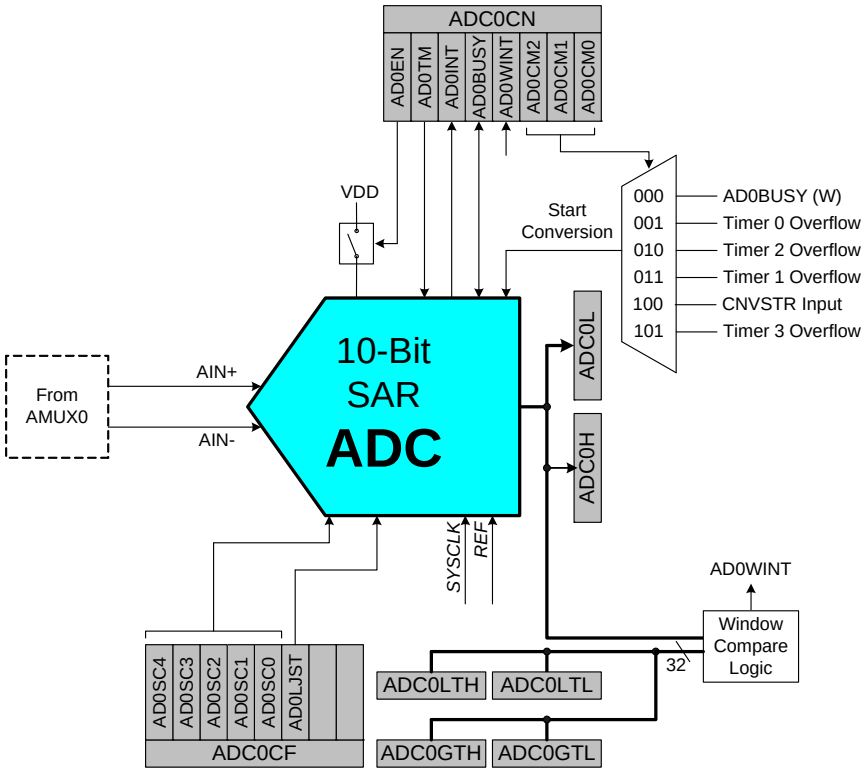



**Figure 7.1. ADC0 Functional Block Diagram**

## 7.1. Output Code Formatting

The ADC is in Single-ended mode when the negative input is connected to GND. The ADC will be in Differential mode when the negative input is connected to any other option. The output code format differs between Single-ended and Differential modes. The registers ADC0H and ADC0L contain the high and low bytes of the output conversion code from the ADC at the completion of each conversion. Data can be right-justified or left-justified, depending on the setting of the AD0LJST. When in Single-ended Mode, conversion codes are represented as 10-bit unsigned integers. Inputs are measured from '0' to VREF x 1023/1024. Example codes are shown below for both right-justified and left-justified data. Unused bits in the ADC0H and ADC0L registers are set to '0'.

| Input Voltage | Right-Justified ADC0H:ADC0L (AD0LJST = 0) | Left-Justified ADC0H:ADC0L (AD0LJST = 1) |
|---|---|---|
| VREF x 1023/1024 | 0x03FF | 0xFFC0 |
| VREF x 512/1024 | 0x0200 | 0x8000 |
| VREF x 256/1024 | 0x0100 | 0x4000 |
| 0 | 0x0000 | 0x0000 |

When in Differential Mode, conversion codes are represented as 10-bit signed 2's complement numbers. Inputs are measured from –VREF to VREF x 511/512. Example codes are shown below for both right-justified and left-justified data. For right-justified data, the unused MSBs of ADC0H are a sign-extension of the data word. For left-justified data, the unused LSBs in the ADC0L register are set to '0'.

| Input Voltage | Right-Justified ADC0H:ADC0L (AD0LJST = 0) | Left-Justified ADC0H:ADC0L (AD0LJST = 1) |
|---|---|---|
| VREF x 511/512 | 0x01FF | 0x7FC0 |
| VREF x 256/512 | 0x0100 | 0x4000 |
| 0 | 0x0000 | 0x0000 |
| –VREF x 256/512 | 0xFF00 | 0xC000 |
| –VREF | 0xFE00 | 0x8000 |

## 7.2. Modes of Operation

ADC0 has a maximum conversion speed of 200 ksps. The ADC0 conversion clock is a divided version of the system clock, determined by the AD0SC bits in the ADC0CF register.

### 7.2.1. Starting a Conversion

A conversion can be initiated in one of six ways, depending on the programmed states of the ADC0 Start of Conversion Mode bits (AD0CM2–0) in register ADC0CN. Conversions may be initiated by one of the following:

1. Writing a '1' to the AD0BUSY bit of register ADC0CN
2. A Timer 0 overflow (i.e., timed continuous conversions)
3. A Timer 2 overflow
4. A Timer 1 overflow
5. A rising edge on the CNVSTR input signal (pin P0.6)
6. A Timer 3 overflow

Writing a '1' to AD0BUSY provides software control of ADC0 whereby conversions are performed "on-demand". During conversion, the AD0BUSY bit is set to logic 1 and reset to logic 0 when the conversion is

complete. The falling edge of AD0BUSY triggers an interrupt (when enabled) and sets the ADC0 interrupt flag (AD0INT). Note: When polling for ADC conversion completions, the ADC0 interrupt flag (AD0INT) should be used. Converted data is available in the ADC0 data registers, ADC0H:ADC0L, when bit AD0INT is logic 1. Note that when Timer 2 or Timer 3 overflows are used as the conversion source, Low Byte overflows are used if Timer 2/3 is in 8-bit mode; High byte overflows are used if Timer 2/3 is in 16-bit mode. See **Section "24. Timers" on page 180** for timer configuration.

**Important Note About Using CNVSTR:** The CNVSTR input pin also functions as Port pin P0.6. When the CNVSTR input is used as the ADC0 conversion source, Port pin P0.6 should be skipped by the Digital Crossbar. To configure the Crossbar to skip P0.6, set to '1' Bit6 in register P0SKIP. See **Section "20. Port Input/Output" on page 119** for details on Port I/O configuration.

### 7.2.2. Tracking Modes

Each ADC0 conversion must be preceded by a minimum tracking time in order for the converted result to be accurate. Refer to **Section "6. Electrical Characteristics" on page 27** for minimum tracking time specifications. The AD0TM bit in register ADC0CN controls the ADC0 track-and-hold mode. In its default state, the ADC0 input is continuously tracked, except when a conversion is in progress. When the AD0TM bit is logic 1, ADC0 operates in low-power track-and-hold mode. In this mode, each conversion is preceded by a tracking period of 3 SAR clocks (after the start-of-conversion signal). When the CNVSTR signal is used to initiate conversions in low-power tracking mode, ADC0 tracks only when CNVSTR is low; conversion begins on the rising edge of CNVSTR (see Figure 7.2). Tracking can also be disabled (shutdown) when the device is in low power standby or sleep modes. Low-power track-and-hold mode is also useful when AMUX settings are frequently changed, due to the settling time requirements described in **Section "7.2.3. Settling Time Requirements" on page 40**.

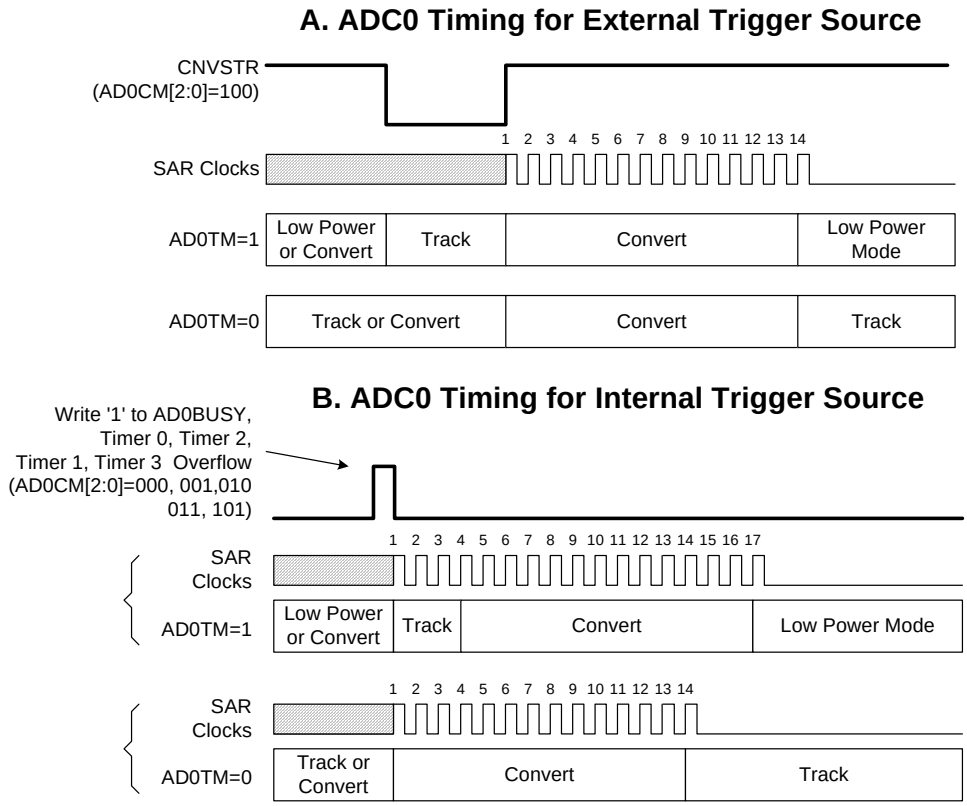



**Figure 7.2. 10-Bit ADC Track and Conversion Example Timing**

### 7.2.3. Settling Time Requirements

A minimum tracking time is required before each conversion to ensure that an accurate conversion is performed. This tracking time is determined by any series impedance, including the AMUX0 resistance, the the ADC0 sampling capacitance, and the accuracy required for the conversion. Note that in low-power tracking mode, three SAR clocks are used for tracking at the start of every conversion. For many applications, these three SAR clocks will meet the minimum tracking time requirements.

Figure 7.3 shows the equivalent ADC0 input circuits for both Differential and Single-ended modes. Notice that the equivalent time constant for both input circuits is the same. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 7.1. When measuring the Temperature Sensor output or $V_{DD}$ with respect to GND, $R_{TOTAL}$ reduces to $R_{MUX}$. See **Section "6. Electrical Characteristics" on page 27** for ADC0 minimum settling time requirements as well as the mux impedance and sampling capacitor values.

$$t = \ln\left(\frac{2^n}{SA}\right) \times R_{TOTAL} C_{SAMPLE}$$

**Equation 7.1. ADC0 Settling Time Requirements**

Where:
*SA* is the settling accuracy, given as a fraction of an LSB (for example, 0.25 to settle within 1/4 LSB)
*t* is the required settling time in seconds
$R_{TOTAL}$ is the sum of the AMUX0 resistance and any external source resistance.
*n* is the ADC resolution in bits (10).

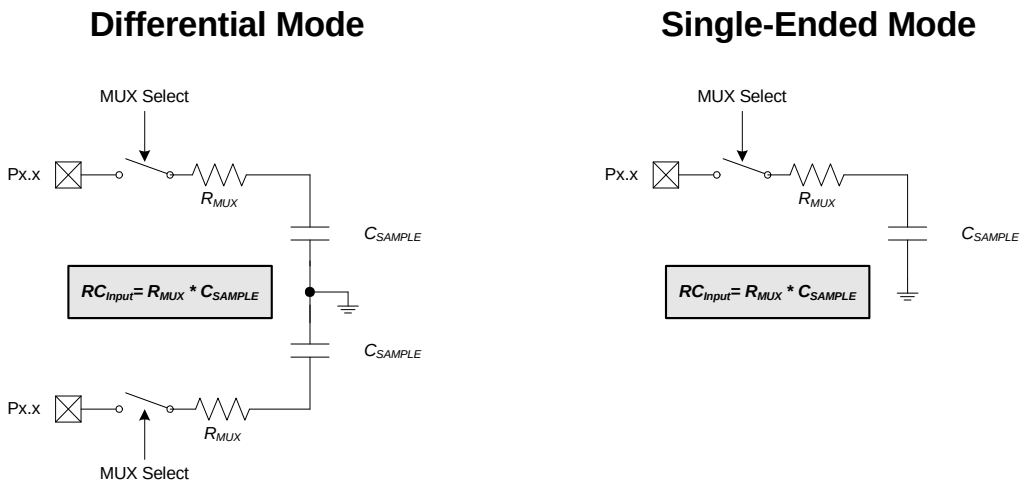



**Figure 7.3. ADC0 Equivalent Input Circuits**

## SFR Definition 7.1. ADC0CF: ADC0 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AD0SC[4:0] | | | | | AD0LJST | | |
| Type | R/W | | | | | R/W | R | R |
| Reset | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

SFR Address = 0xBC

| Bit | Name | Function |
|---|---|---|
| 7:3 | AD0SC[4:0] | **ADC0 SAR Conversion Clock Period Bits.**<br>SAR Conversion clock is derived from system clock by the following equation, where *AD0SC* refers to the 5-bit value held in bits AD0SC4–0. SAR Conversion clock requirements are given in the ADC specification table.<br>$AD0SC = \frac{SYSCLK}{CLK_{SAR}} - 1$ |
| 2 | AD0LJST | **ADC0 Left Justify Select.**<br>0: Data in ADC0H:ADC0L registers are right-justified.<br>1: Data in ADC0H:ADC0L registers are left-justified. |
| 1:0 | UNUSED | Unused. Read = 00b; Write = don't care. |

## SFR Definition 7.2. ADC0H: ADC0 Data Word MSB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0H[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xBE

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0H[7:0] | **ADC0 Data Word High-Order Bits.**<br>For AD0LJST = 0: Bits 7–2 are the sign extension of Bit1. Bits 1–0 are the upper 2 bits of the 10-bit ADC0 Data Word.<br>For AD0LJST = 1: Bits 7–0 are the most-significant bits of the 10-bit ADC0 Data Word. |

## SFR Definition 7.3. ADC0L: ADC0 Data Word LSB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0L[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xBD

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0L[7:0] | **ADC0 Data Word Low-Order Bits.**<br>For AD0LJST = 0: Bits 7–0 are the lower 8 bits of the 10-bit Data Word.<br>For AD0LJST = 1: Bits 7–6 are the lower 2 bits of the 10-bit Data Word. Bits 5–0 will always read ‘0’. |

## SFR Definition 7.4. ADC0CN: ADC0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AD0EN | AD0TM | AD0INT | AD0BUSY | AD0WINT | AD0CM[2:0] | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xE8; Bit-Addressable

| Bit | Name | Function | | |
|---|---|---|---|---|
| 7 | AD0EN | **ADC0 Enable Bit.**<br>0: ADC0 Disabled. ADC0 is in low-power shutdown.<br>1: ADC0 Enabled. ADC0 is active and ready for data conversions. | | |
| 6 | AD0TM | **ADC0 Track Mode Bit.**<br>0: Normal Track Mode: When ADC0 is enabled, tracking is continuous unless a conversion is in progress. Conversion begins immediately on start-of-conversion event, as defined by AD0CM[2:0].<br>1: Low-power Track Mode: For AD0CM[2:0] = 100, ADC is tracking when CNVSTR is low, and conversion begins immediately on rising edge of CNVSTR.<br>For all other values of AD0CM[2:0], tracking is initiated on start-of-conversion event, and lasts 3 SAR Clock cycles. The conversion immediately follows this tracking phase. | | |
| 5 | AD0INT | **ADC0 Conversion Complete Interrupt Flag.**<br>0: ADC0 has not completed a data conversion since AD0INT was last cleared.<br>1: ADC0 has completed a data conversion. | | |
| 4 | AD0BUSY | **ADC0 Busy Bit.** | **Read:**<br>0: ADC0 conversion is not in progress.<br>1: ADC0 conversion is in progress. | **Write:**<br>0: No Effect.<br>1: Initiates ADC0 Conversion if AD0CM[2:0] = 000b |
| 3 | AD0WINT | **ADC0 Window Compare Interrupt Flag.**<br>0: ADC0 Window Comparison Data match has not occurred since this flag was last cleared.<br>1: ADC0 Window Comparison Data match has occurred. | | |
| 2:0 | AD0CM[2:0] | **ADC0 Start of Conversion Mode Select.**<br>000: ADC0 start-of-conversion source is write of ‘1’ to AD0BUSY.<br>001: ADC0 start-of-conversion source is overflow of Timer 0.<br>010: ADC0 start-of-conversion source is overflow of Timer 2.<br>011: ADC0 start-of-conversion source is overflow of Timer 1.<br>100: ADC0 start-of-conversion source is rising edge of external CNVSTR.<br>101: ADC0 start-of-conversion source is overflow of Timer 3.<br>11x: Reserved. | | |

## 7.3. Programmable Window Detector

The ADC Programmable Window Detector continuously compares the ADC0 output registers to user-programmed limits, and notifies the system when a desired condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (AD0WINT in register ADC0CN) can also be used in polled mode. The ADC0 Greater-Than (ADC0GTH, ADC0GTL) and Less-Than (ADC0LTH, ADC0LTL) registers hold the comparison values. The window detector flag can be programmed to indicate when measured data is inside or outside of the user-programmed limits, depending on the contents of the ADC0 Less-Than and ADC0 Greater-Than registers.

### SFR Definition 7.5. ADC0GTH: ADC0 Greater-Than Data High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0GTH[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xC4

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0GTH[7:0] | **ADC0 Greater-Than Data Word High-Order Bits.** |

### SFR Definition 7.6. ADC0GTL: ADC0 Greater-Than Data Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0GTL[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xC3

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0GTL[7:0] | **ADC0 Greater-Than Data Word Low-Order Bits.** |

## SFR Definition 7.7. ADC0LTH: ADC0 Less-Than Data High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0LTH[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC6

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0LTH[7:0] | **ADC0 Less-Than Data Word High-Order Bits.** |

## SFR Definition 7.8. ADC0LTL: ADC0 Less-Than Data Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0LTL[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC5

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0LTL[7:0] | **ADC0 Less-Than Data Word Low-Order Bits.** |

### 7.3.1. Window Detector In Single-Ended Mode

Figure 7.4 shows two example window comparisons for right-justified, single-ended data, with ADC0LTH:ADC0LTL = 0x0080 (128d) and ADC0GTH:ADC0GTL = 0x0040 (64d). In single-ended mode, the input voltage can range from '0' to VREF x (1023/1024) with respect to GND, and is represented by a 10-bit unsigned integer value. In the left example, an AD0WINT interrupt will be generated if the ADC0 conversion word (ADC0H:ADC0L) is within the range defined by ADC0GTH:ADC0GTL and ADC0LTH:ADC0LTL (if 0x0040 < ADC0H:ADC0L < 0x0080). In the right example, and AD0WINT interrupt will be generated if the ADC0 conversion word is outside of the range defined by the ADC0GT and ADC0LT registers (if ADC0H:ADC0L < 0x0040 or ADC0H:ADC0L > 0x0080). Figure 7.5 shows an example using left-justified data with the same comparison values.

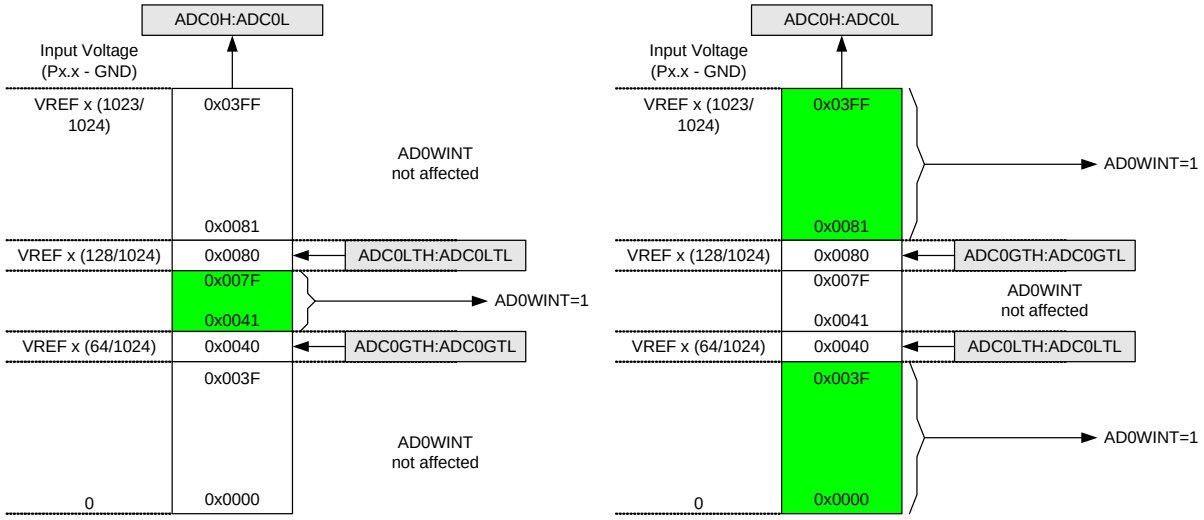



**Figure 7.4. ADC Window Compare Example: Right-Justified Single-Ended Data**

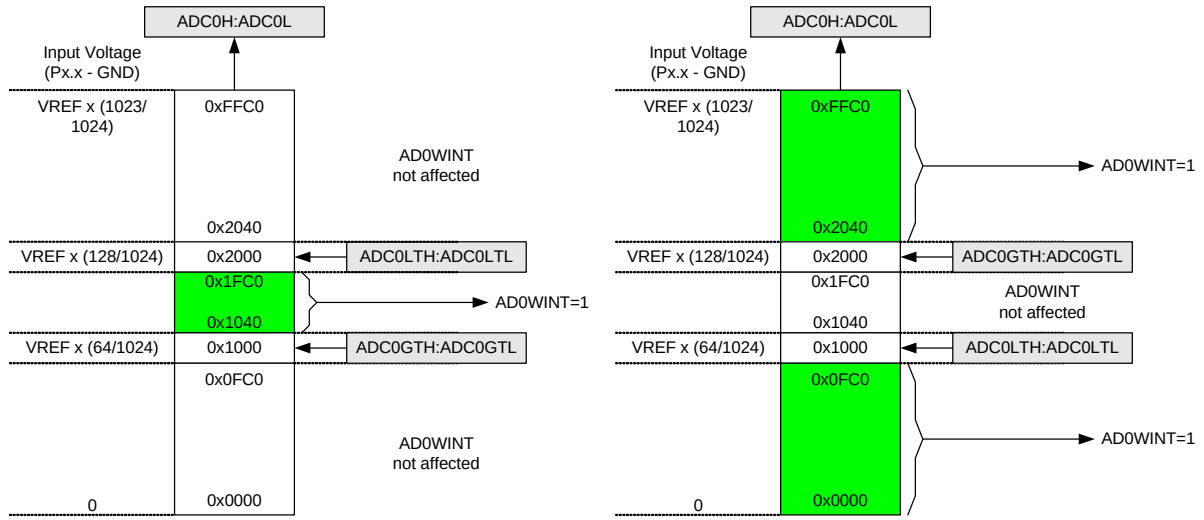



**Figure 7.5. ADC Window Compare Example: Left-Justified Single-Ended Data**

### 7.3.2. Window Detector In Differential Mode

Figure 7.6 shows two example window comparisons for right-justified, differential data, with ADC0LTH:ADC0LTL = 0x0040 (+64d) and ADC0GTH:ADC0GTH = 0xFFFF (–1d). In differential mode, the measurable voltage between the input pins is between –VREF and VREF x (511/512). Output codes are represented as 10-bit 2s complement signed integers. In the left example, an AD0WINT interrupt will be generated if the ADC0 conversion word (ADC0H:ADC0L) is within the range defined by ADC0GTH:ADC0GTL and ADC0LTH:ADC0LTL (if 0xFFFF (–1d) < ADC0H:ADC0L < 0x0040 (64d)). In the right example, an AD0WINT interrupt will be generated if the ADC0 conversion word is outside of the range defined by the ADC0GT and ADC0LT registers (if ADC0H:ADC0L < 0xFFFF (–1d) or ADC0H:ADC0L > 0x0040 (+64d)). Figure 7.7 shows an example using left-justified data with the same comparison values.

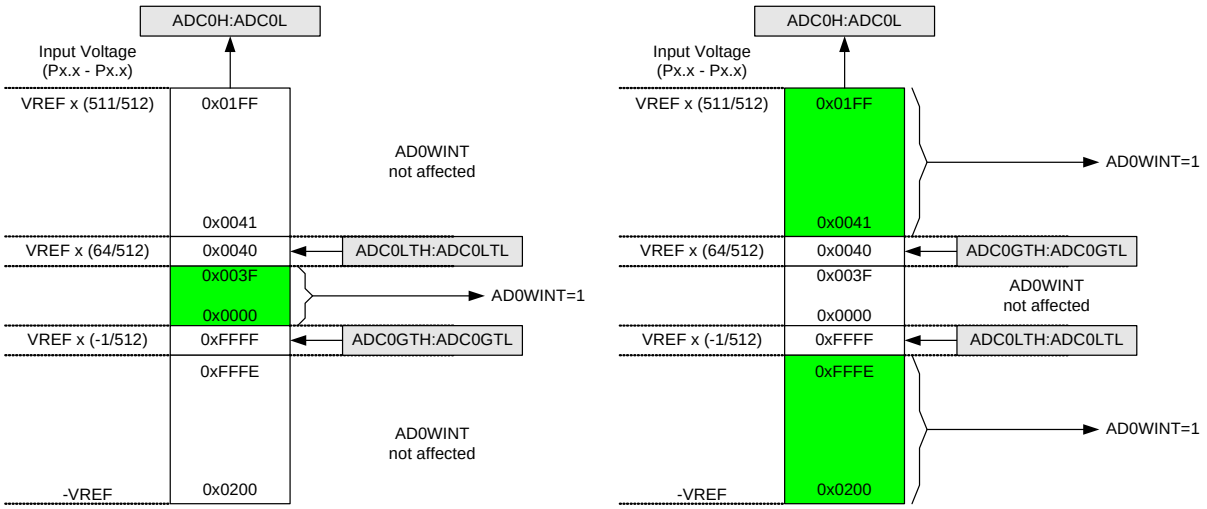



**Figure 7.6. ADC Window Compare Example: Right-Justified Differential Data**

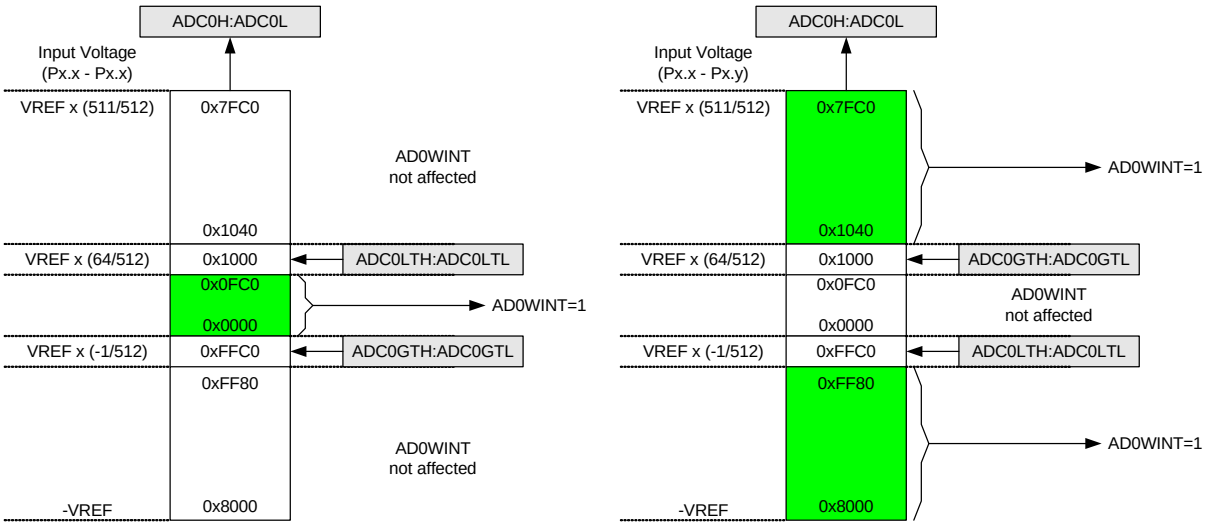



**Figure 7.7. ADC Window Compare Example: Left-Justified Differential Data**

## 7.4. ADC0 Analog Multiplexer (C8051F336/8 only)

ADC0 on the C8051F336/8 has two analog multiplexers, referred to collectively as AMUX0.

AMUX0 selects the positive and negative inputs to the ADC. Any of the following may be selected as the positive input: Port I/O pins, the on-chip temperature sensor, or the positive power supply ($V_{DD}$). Any of the following may be selected as the negative input: Port I/O pins, $V_{REF}$, or GND. **When GND is selected as the negative input, ADC0 operates in Single-ended Mode; all other times, ADC0 operates in Differential Mode.** The ADC0 input channels are selected in the AMX0P and AMX0N registers as described in SFR Definition 7.9 and SFR Definition 7.10.

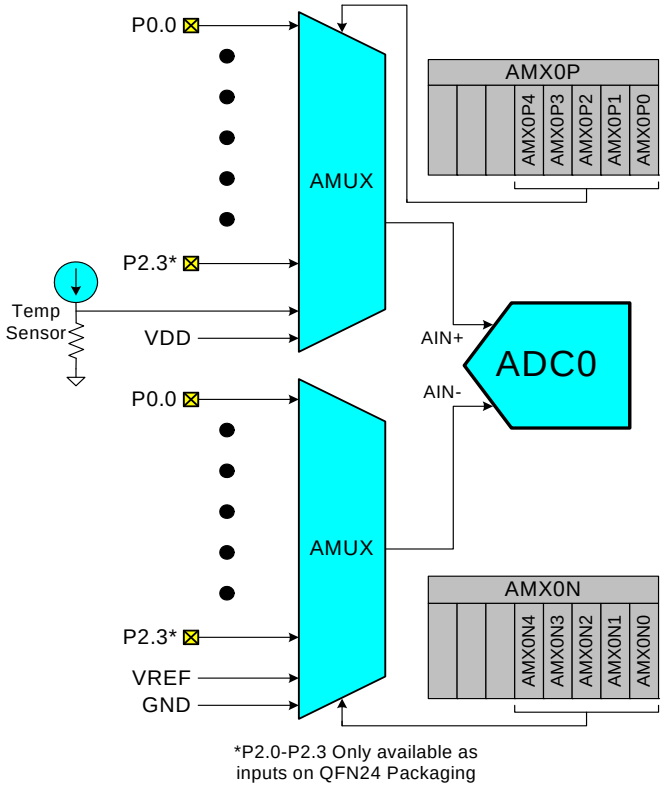



**Figure 7.8. ADC0 Multiplexer Block Diagram**

**Important Note About ADC0 Input Configuration:** Port pins selected as ADC0 inputs should be configured as analog inputs, and should be skipped by the Digital Crossbar. To configure a Port pin for analog input, set to '0' the corresponding bit in register PnMDIN. To force the Crossbar to skip a Port pin, set to '1' the corresponding bit in register PnSKIP. See **Section "20. Port Input/Output" on page 119** for more Port I/O configuration details.

## SFR Definition 7.9. AMX0P: AMUX0 Positive Channel Select

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | AMX0P[4:0] | | | | |
| Type | R | R | R | R/W | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xBB

| Bit | Name | Function |
|---|---|---|
| 7:5 | UNUSED | Unused. Read = 000b; Write = Don't Care. |
| 4:0 | AMX0P[4:0] | **AMUX0 Positive Input Selection.**<br>00000: P0.0<br>00001: P0.1<br>00010: P0.2<br>00011: P0.3<br>00100: P0.4<br>00101: P0.5<br>00110: P0.6<br>00111: P0.7<br>01000: P1.0<br>01001: P1.1<br>01010: P1.2<br>01011: P1.3<br>01100: P1.4<br>01101: P1.5<br>01110: P1.6<br>01111: P1.7<br>10000: Temp Sensor<br>10001: $V_{DD}$<br>10010: P2.0 (C8051F338/9 Only)<br>10011: P2.1 (C8051F338/9 Only)<br>10100: P2.2 (C8051F338/9 Only)<br>10101: P2.3 (C8051F338/9 Only)<br>10110 – 11111: no input selected |

## SFR Definition 7.10. AMX0N: AMUX0 Negative Channel Select

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | AMX0N[4:0] | | | | |
| Type | R | R | R | R/W | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xBA

| Bit | Name | Function |
|---|---|---|
| 7:5 | UNUSED | Unused. Read = 000b; Write = Don't Care. |
| 4:0 | AMX0N[4:0] | **AMUX0 Negative Input Selection.**<br>00000: P0.0<br>00001: P0.1<br>00010: P0.2<br>00011: P0.3<br>00100: P0.4<br>00101: P0.5<br>00110: P0.6<br>00111: P0.7<br>01000: P1.0<br>01001: P1.1<br>01010: P1.2<br>01011: P1.3<br>01100: P1.4<br>01101: P1.5<br>01110: P1.6<br>01111: P1.7<br>10000: $V_{REF}$<br>10001: GND (ADC in Single-Ended Mode)<br>10010: P2.0 (C8051F338/9 Only)<br>10011: P2.1 (C8051F338/9 Only)<br>10100: P2.2 (C8051F338/9 Only)<br>10101: P2.3 (C8051F338/9 Only)<br>10110 – 11111: no input selected |

## 8. Temperature Sensor (C8051F336/8 only)

An on-chip temperature sensor is included on the C8051F336/8 which can be directly accessed via the ADC multiplexer in single-ended configuration. To use the ADC to measure the temperature sensor, the positive ADC mux channel should be configured to connect to the temperature sensor and the negative ADC mux channel should be configured to connect to GND. The temperature sensor transfer function is shown in Figure 8.1. The output voltage ($V_{TEMP}$) is the positive ADC input when the ADC multiplexer is set correctly. The TEMPE bit in register REF0CN enables/disables the temperature sensor, as described in SFR Definition 10.1. While disabled, the temperature sensor defaults to a high impedance state and any ADC measurements performed on the sensor will result in meaningless data. Refer to **Section "6. Electrical Characteristics" on page 27** for the slope and offset parameters of the temperature sensor.

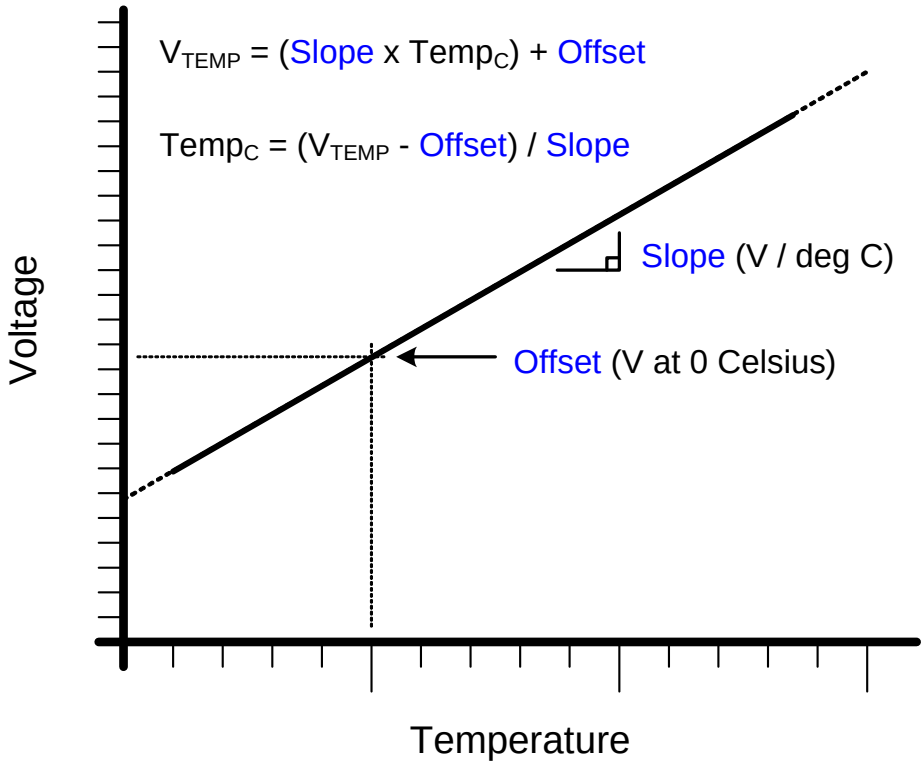



**Figure 8.1. Temperature Sensor Transfer Function**

# 9. 10-Bit Current Mode DAC (IDA0, C8051F336/8 only)

The C8051F336/8 device includes a 10-bit current-mode Digital-to-Analog Converter (IDAC). The maximum current output of the IDAC can be adjusted for three different current settings; 0.5 mA, 1 mA, and 2 mA. The IDAC is enabled or disabled with the IDA0EN bit in the IDA0 Control Register (see SFR Definition 9.1). When IDA0EN is set to 0, the IDAC port pin (P0.1) behaves as a normal GPIO pin. When IDA0EN is set to 1, the digital output drivers and weak pullup for the IDAC pin are automatically disabled, and the pin is connected to the IDAC output. An internal bandgap bias generator is used to generate a reference current for the IDAC whenever it is enabled. When using the IDAC, bit 1 in the P0SKIP register should be set to 1, to force the Crossbar to skip the IDAC pin.

## 9.1. IDA0 Output Scheduling

IDA0 features a flexible output update mechanism which allows for seamless full-scale changes and supports jitter-free updates for waveform generation. Three update modes are provided, allowing IDAC output updates on a write to IDA0H, on a Timer overflow, or on an external pin edge.

### 9.1.1. Update Output On-Demand

In its default mode (IDA0CN.[6:4] = 111) the IDA0 output is updated “on-demand” on a write to the high-byte of the IDA0 data register (IDA0H). It is important to note that writes to IDA0L are held in this mode, and have no effect on the IDA0 output until a write to IDA0H takes place. If writing a full 10-bit word to the IDAC data registers, the 10-bit data word is written to the low byte (IDA0L) and high byte (IDA0H) data registers. Data is latched into IDA0 after a write to the IDA0H register, **so the write sequence should be IDA0L followed by IDA0H** if the full 10-bit resolution is required. The IDAC can be used in 8-bit mode by initializing IDA0L to the desired value (typically 0x00), and writing data to only IDA0H (see Section 9.2 for information on the format of the 10-bit IDAC data word within the 16-bit SFR space).

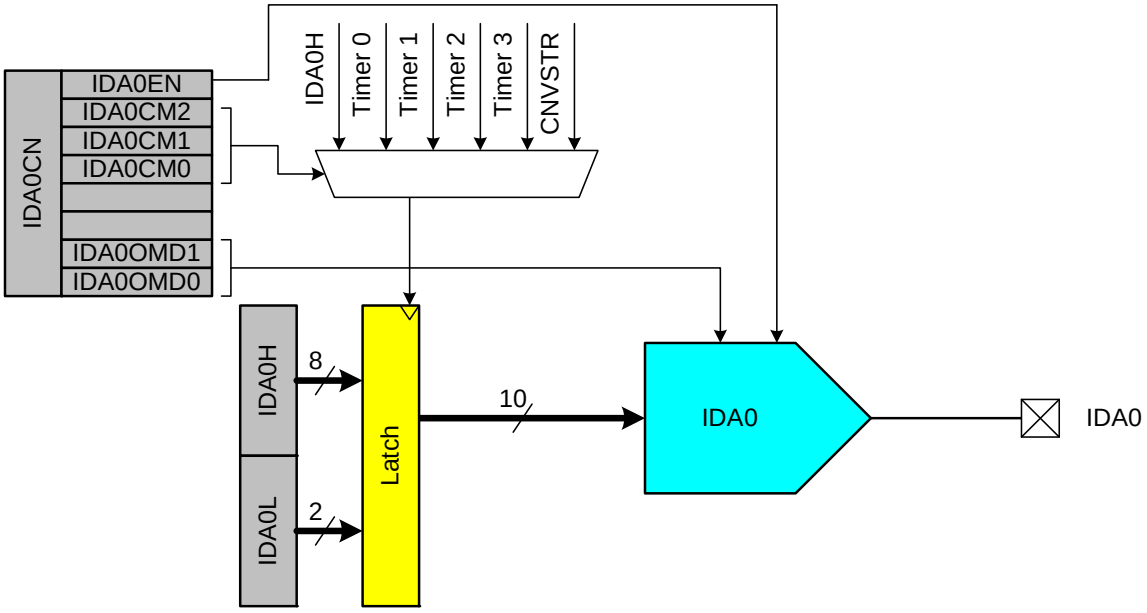



**Figure 9.1. IDA0 Functional Block Diagram**

### 9.1.2. Update Output Based on Timer Overflow

Similar to the ADC operation, in which an ADC conversion can be initiated by a timer overflow independently of the processor, the IDAC outputs can use a Timer overflow to schedule an output update event. This feature is useful in systems where the IDAC is used to generate a waveform of a defined sampling rate by eliminating the effects of variable interrupt latency and instruction execution on the timing of the IDAC output. When the IDA0CM bits (IDA0CN.[6:4]) are set to 000, 001, 010 or 011, writes to both IDAC data registers (IDA0L and IDA0H) are held until an associated Timer overflow event (Timer 0, Timer 1, Timer 2 or Timer 3, respectively) occurs, at which time the IDA0H:IDA0L contents are copied to the IDAC input latches, allowing the IDAC output to change to the new value.

### 9.1.3. Update Output Based on CNVSTR Edge

The IDAC output can also be configured to update on a rising edge, falling edge, or both edges of the external CNVSTR signal. When the IDA0CM bits (IDA0CN.[6:4]) are set to 100, 101, or 110, writes to both IDAC data registers (IDA0L and IDA0H) are held until an edge occurs on the CNVSTR input pin. The particular setting of the IDA0CM bits determines whether IDAC outputs are updated on rising, falling, or both edges of CNVSTR. When a corresponding edge occurs, the IDA0H:IDA0L contents are copied to the IDAC input latches, allowing the IDAC output to change to the new value.

## 9.2. IDAC Output Mapping

The IDAC data registers (IDA0H and IDA0L) are left-justified, meaning that the eight MSBs of the IDAC output word are mapped to bits 7–0 of the IDA0H register, and the two LSBs of the IDAC output word are mapped to bits 7 and 6 of the IDA0L register. The data word mapping for the IDAC is shown in Figure 9.2.

| IDA0H | | | | | | | | IDA0L | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B9 | B8 | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 | | | | | | |

| Input Data Word (IDA09–IDA00) | Output Current IDA0OMD[1:0] = 1x | Output Current IDA0OMD[1:0] = 01 | Output Current IDA0OMD[1:0] = 00 |
|---|---|---|---|
| 0x000 | 0 mA | 0 mA | 0 mA |
| 0x001 | 1/1024 x 2 mA | 1/1024 x 1 mA | 1/1024 x 0.5 mA |
| 0x200 | 512/1024 x 2 mA | 512/1024 x 1 mA | 512/1024 x 0.5 mA |
| 0x3FF | 1023/1024 x 2 mA | 1023/1024 x 1 mA | 1023/1024 x 0.5 mA |

**Figure 9.2. IDA0 Data Word Mapping**

The full-scale output current of the IDAC is selected using the IDA0OMD bits (IDA0CN[1:0]). By default, the IDAC is set to a full-scale output current of 2 mA. The IDA0OMD bits can also be configured to provide full-scale output currents of 1 mA or 0.5 mA, as shown in SFR Definition 9.1.

## SFR Definition 9.1. IDA0CN: IDA0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IDA0EN | IDA0CM[2:0] | | | | | IDA0OMD[1:0] | |
| Type | R/W | R/W | | | R | R | R/W | |
| Reset | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

SFR Address = 0xB9

| Bit | Name | Function |
|---|---|---|
| 7 | IDA0EN | **IDA0 Enable.**<br>0: IDA0 Disabled.<br>1: IDA0 Enabled. |
| 6:4 | IDA0CM[2:0] | **IDA0 Update Source Select bits.**<br>000: DAC output updates on Timer 0 overflow.<br>001: DAC output updates on Timer 1 overflow.<br>010: DAC output updates on Timer 2 overflow.<br>011: DAC output updates on Timer 3 overflow.<br>100: DAC output updates on rising edge of CNVSTR.<br>101: DAC output updates on falling edge of CNVSTR.<br>110: DAC output updates on any edge of CNVSTR.<br>111: DAC output updates on write to IDA0H. |
| 3:2 | Unused | Unused. Read = 00b. Write = Don't care. |
| 1:0 | IDA0OMD[1:0] | **IDA0 Output Mode Select bits.**<br>00: 0.5 mA full-scale output current.<br>01: 1.0 mA full-scale output current.<br>1x: 2.0 mA full-scale output current. |

## SFR Definition 9.2. IDA0H: IDA0 Data Word MSB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IDA0[9:2] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x97

| Bit | Name | Function |
|---|---|---|
| 7:0 | IDA0[9:2] | **IDA0 Data Word High-Order Bits.**<br>Upper 8 bits of the 10-bit IDA0 Data Word. |

## SFR Definition 9.3. IDA0L: IDA0 Data Word LSB

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IDA0[1:0] | | | | | | | |
| Type | R/W | | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x96

| Bit | Name | Function |
|---|---|---|
| 7:6 | IDA0[1:0] | **IDA0 Data Word Low-Order Bits.**<br>Lower 2 bits of the 10-bit IDA0 Data Word. |
| 5:0 | Unused | Unused. Read = 000000b. Write = Don’t care. |

## 10. Voltage Reference (C8051F336/8 only)

The Voltage reference multiplexer for the ADC is configurable to use an externally connected voltage reference, the on-chip reference voltage generator routed to the VREF pin, or the $V_{DD}$ power supply voltage (see Figure 10.1). The REFSL bit in the Reference Control register (REF0CN, SFR Definition 10.1) selects the reference source for the ADC. For an external source or the on-chip reference, REFSL should be set to '0' to select the VREF pin. To use $V_{DD}$ as the reference source, REFSL should be set to '1'.

The BIASE bit enables the internal voltage bias generator, which is used by many of the analog peripherals on the device. This bias is automatically enabled when any peripheral which requires it is enabled, and it does not need to be enabled manually. The bias generator may be enabled manually by writing a '1' to the BIASE bit in register REF0CN. The electrical specifications for the voltage reference circuit are given in **Section "6. Electrical Characteristics" on page 27**.

The on-chip voltage reference circuit consists of a 1.2 V, temperature stable bandgap voltage reference generator and a gain-of-two output buffer amplifier. The on-chip voltage reference can be driven on the VREF pin by setting the REFBE bit in register REF0CN to a '1'. The maximum load seen by the VREF pin must be less than 200 µA to GND. Bypass capacitors of 0.1 µF and 4.7 µF are recommended from the VREF pin to GND. If the on-chip reference is not used, the REFBE bit should be cleared to '0'.

**Important Note about the VREF Pin:** When using either an external voltage reference or the on-chip reference circuitry, the VREF pin should be configured as an analog pin and skipped by the Digital Crossbar. Refer to **Section "20. Port Input/Output" on page 119** for the location of the VREF pin, as well as details of how to configure the pin in analog mode and to be skipped by the crossbar.

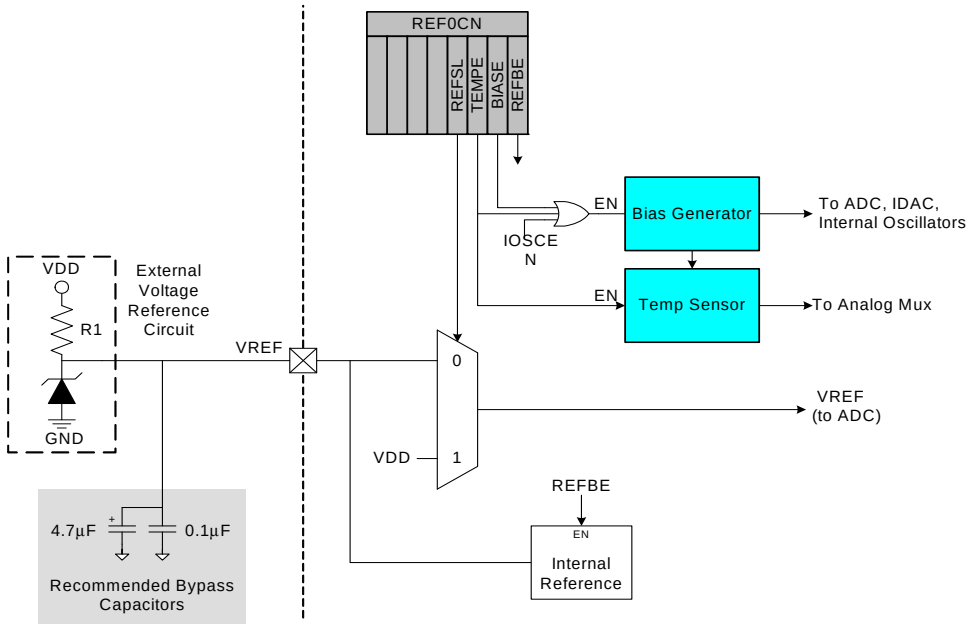



**Figure 10.1. Voltage Reference Functional Block Diagram**

## SFR Definition 10.1. REF0CN: Reference Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | REFSL | TEMPE | BIASE | REFBE |
| Type | R | R | R | R | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD1

| Bit | Name | Function |
|---|---|---|
| 7:4 | UNUSED | Unused. Read = 0000b; Write = don't care. |
| 3 | REFSL | **Voltage Reference Select.**<br>This bit selects the ADCs voltage reference.<br>0: $V_{REF}$ pin used as voltage reference.<br>1: $V_{DD}$ used as voltage reference. |
| 2 | TEMPE | **Temperature Sensor Enable Bit.**<br>0: Internal Temperature Sensor off.<br>1: Internal Temperature Sensor on. |
| 1 | BIASE | **Internal Analog Bias Generator Enable Bit.**<br>0: Internal Bias Generator off.<br>1: Internal Bias Generator on. |
| 0 | REFBE | **On-chip Reference Buffer Enable Bit.**<br>0: On-chip Reference Buffer off.<br>1: On-chip Reference Buffer on. Internal voltage reference driven on the $V_{REF}$ pin. |

# 11. Comparator0

C8051F336/7/8/9 devices include an on-chip programmable voltage comparator, Comparator0, shown in Figure 11.1.

The Comparator offers programmable response time and hysteresis, an analog input multiplexer, and two outputs that are optionally available at the Port pins: a synchronous “latched” output (CP0), or an asynchronous “raw” output (CP0A). The asynchronous CP0A signal is available even when the system clock is not active. This allows the Comparator to operate and generate an output with the device in STOP mode. When assigned to a Port pin, the Comparator output may be configured as open drain or push-pull (see Section “20.4. Port I/O Initialization” on page 126). Comparator0 may also be used as a reset source (see Section “17.5. Comparator0 Reset” on page 104).

The Comparator0 inputs are selected by the comparator input multiplexer, as detailed in Section “11.1. Comparator Multiplexer” on page 63.

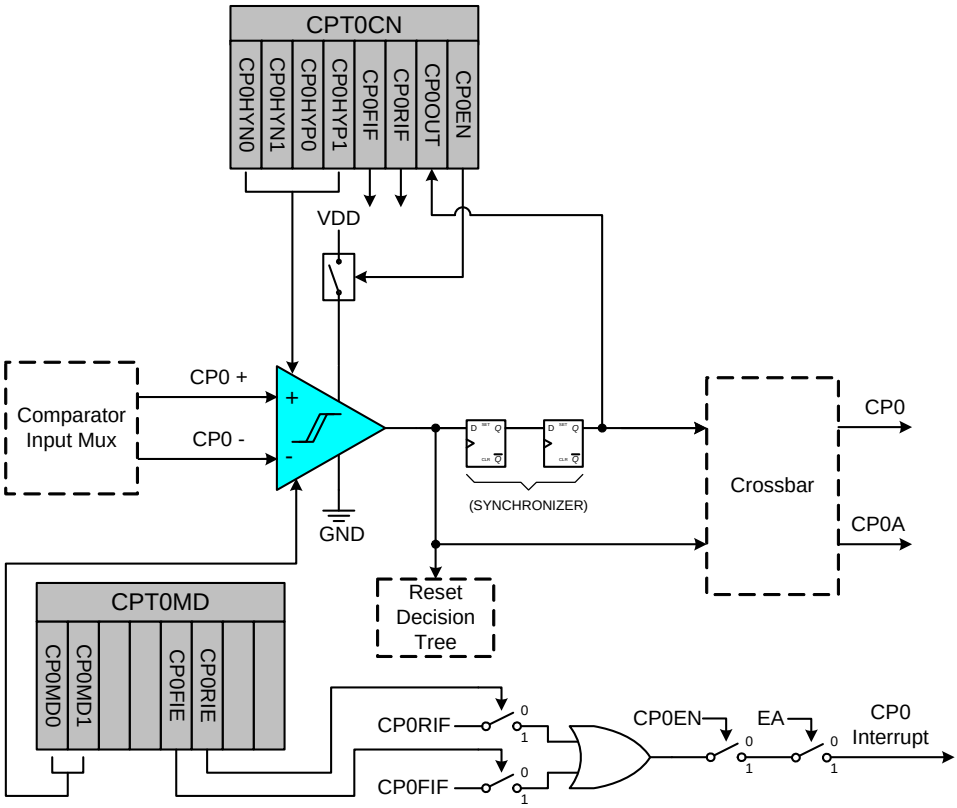



**Figure 11.1. Comparator0 Functional Block Diagram**

The Comparator output can be polled in software, used as an interrupt source, and/or routed to a Port pin. When routed to a Port pin, the Comparator output is available asynchronous or synchronous to the system clock; the asynchronous output is available even in STOP mode (with no system clock active). When disabled, the Comparator output (if assigned to a Port I/O pin via the Crossbar) defaults to the logic low state, and the power supply to the comparator is turned off. See Section “20.3. Priority Crossbar Decoder” on page 124 for details on configuring Comparator outputs via the digital Crossbar. Comparator inputs can be

externally driven from –0.25 V to ($V_{DD}$) + 0.25 V without damage or upset. The complete Comparator electrical specifications are given in Section “6. Electrical Characteristics” on page 27.

The Comparator response time may be configured in software via the CPT0MD register (see SFR Definition 11.2). Selecting a longer response time reduces the Comparator supply current.

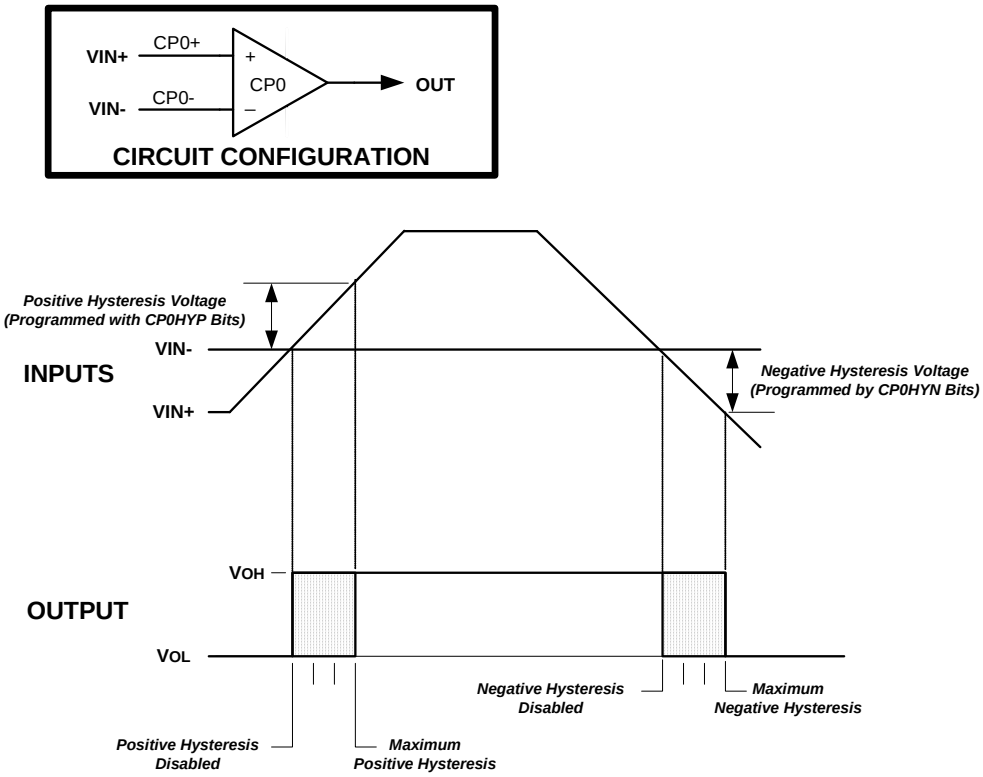



**Figure 11.2. Comparator Hysteresis Plot**

The Comparator hysteresis is software-programmable via its Comparator Control register CPT0CN. The user can program both the amount of hysteresis voltage (referred to the input voltage) and the positive and negative-going symmetry of this hysteresis around the threshold voltage.

The Comparator hysteresis is programmed using Bits3–0 in the Comparator Control Register CPT0CN (shown in SFR Definition 11.1). The amount of negative hysteresis voltage is determined by the settings of the CP0HYN bits. As shown in Figure 11.2, settings of 20, 10 or 5 mV of negative hysteresis can be programmed, or negative hysteresis can be disabled. In a similar way, the amount of positive hysteresis is determined by the setting the CP0HYP bits.

Comparator interrupts can be generated on both rising-edge and falling-edge output transitions. (For Interrupt enable and priority control, see Section “15.1. MCU Interrupt Sources and Vectors” on page 83). The CP0FIF flag is set to logic 1 upon a Comparator falling-edge occurrence, and the CP0RIF flag is set to logic 1 upon the Comparator rising-edge occurrence. Once set, these bits remain set until cleared by software. The Comparator rising-edge interrupt mask is enabled by setting CP0RIE to a logic 1. The Comparator0 falling-edge interrupt mask is enabled by setting CP0FIE to a logic 1.

The output state of the Comparator can be obtained at any time by reading the CP0OUT bit. The Comparator is enabled by setting the CP0EN bit to logic 1, and is disabled by clearing this bit to logic 0.

Note that false rising edges and falling edges can be detected when the comparator is first powered on or if changes are made to the hysteresis or response time control bits. Therefore, it is recommended that the rising-edge and falling-edge flags be explicitly cleared to logic 0 a short time after the comparator is enabled or its mode bits have been changed.

## SFR Definition 11.1. CPT0CN: Comparator0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CP0EN | CP0OUT | CP0RIF | CP0FIF | CP0HYP[1:0] | | CP0HYN[1:0] | |
| Type | R/W | R | R/W | R/W | R/W | | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x9B

| Bit | Name | Function |
|---|---|---|
| 7 | CP0EN | **Comparator0 Enable Bit.**<br>0: Comparator0 Disabled.<br>1: Comparator0 Enabled. |
| 6 | CP0OUT | **Comparator0 Output State Flag.**<br>0: Voltage on CP0+ < CP0–.<br>1: Voltage on CP0+ > CP0–. |
| 5 | CP0RIF | **Comparator0 Rising-Edge Flag. Must be cleared by software.**<br>0: No Comparator0 Rising Edge has occurred since this flag was last cleared.<br>1: Comparator0 Rising Edge has occurred. |
| 4 | CP0FIF | **Comparator0 Falling-Edge Flag. Must be cleared by software.**<br>0: No Comparator0 Falling-Edge has occurred since this flag was last cleared.<br>1: Comparator0 Falling-Edge has occurred. |
| 3:2 | CP0HYP[1:0] | **Comparator0 Positive Hysteresis Control Bits.**<br>00: Positive Hysteresis Disabled.<br>01: Positive Hysteresis = 5 mV.<br>10: Positive Hysteresis = 10 mV.<br>11: Positive Hysteresis = 20 mV. |
| 1:0 | CP0HYN[1:0] | **Comparator0 Negative Hysteresis Control Bits.**<br>00: Negative Hysteresis Disabled.<br>01: Negative Hysteresis = 5 mV.<br>10: Negative Hysteresis = 10 mV.<br>11: Negative Hysteresis = 20 mV. |

## SFR Definition 11.2. CPT0MD: Comparator0 Mode Selection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | CP0RIE | CP0FIE | | | CP0MD[1:0] | |
| Type | R | R | R/W | R/W | R | R | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

SFR Address = 0x9D

| Bit | Name | Function |
|---|---|---|
| 7:6 | Unused | Unused. Read = 00b, Write = Don't Care. |
| 5 | CP0RIE | **Comparator0 Rising-Edge Interrupt Enable.**<br>0: Comparator0 Rising-edge interrupt disabled.<br>1: Comparator0 Rising-edge interrupt enabled. |
| 4 | CP0FIE | **Comparator0 Falling-Edge Interrupt Enable.**<br>0: Comparator0 Falling-edge interrupt disabled.<br>1: Comparator0 Falling-edge interrupt enabled. |
| 3:2 | Unused | Unused. Read = 00b, Write = don't care. |
| 1:0 | CP0MD[1:0] | **Comparator0 Mode Select.**<br>These bits affect the response time and power consumption for Comparator0.<br>00: Mode 0 (Fastest Response Time, Highest Power Consumption)<br>01: Mode 1<br>10: Mode 2<br>11: Mode 3 (Slowest Response Time, Lowest Power Consumption) |

## 11.1. Comparator Multiplexer

C8051F336/7/8/9 devices include an analog input multiplexer to connect Port I/O pins to the comparator inputs. The Comparator0 inputs are selected in the CPT0MX register (SFR Definition 11.3). The CMX0P1–CMX0P0 bits select the Comparator0 positive input; the CMX0N1–CMX0N0 bits select the Comparator0 negative input. **Important Note About Comparator Inputs:** The Port pins selected as comparator inputs should be configured as analog inputs in their associated Port configuration register, and configured to be skipped by the Crossbar (for details on Port configuration, see **Section "20.6. Special Function Registers for Accessing and Configuring Port I/O" on page 131**).

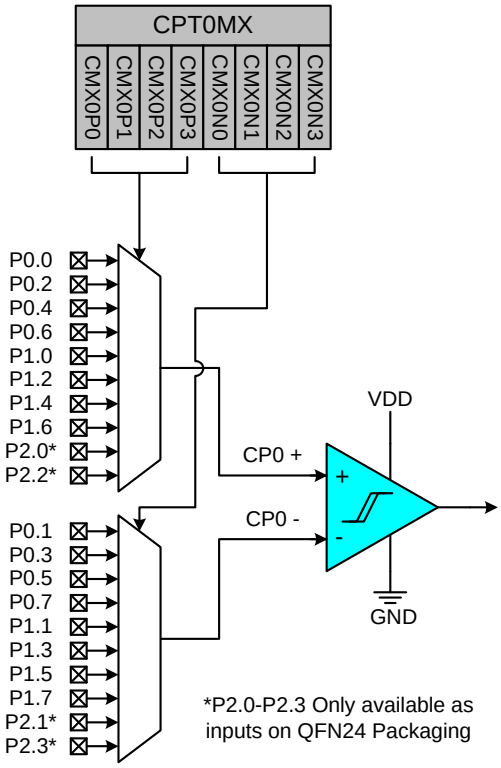



**Figure 11.3. Comparator Input Multiplexer Block Diagram**

## SFR Definition 11.3. CPT0MX: Comparator0 MUX Selection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CMX0N[3:0] | | | | CMX0P[3:0] | | | |
| Type | R/W | | | | R/W | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0x9F

| Bit | Name | Function |
|---|---|---|
| 7:4 | CMX0N[3:0] | **Comparator0 Negative Input MUX Selection.**<br>0000: P0.1<br>0001: P0.3<br>0010: P0.5<br>0011: P0.7<br>0100: P1.1<br>0101: P1.3<br>0110: P1.5<br>0111: P1.7<br>1000: P2.1 (C8051F338/9 Only)<br>1001: P2.3 (C8051F338/9 Only)<br>1010-1111: None |
| 3:0 | CMX0P[3:0] | **Comparator0 Positive Input MUX Selection.**<br>0000: P0.0<br>0001: P0.2<br>0010: P0.4<br>0011: P0.6<br>0100: P1.0<br>0101: P1.2<br>0110: P1.4<br>0111: P1.6<br>1000: P2.0 (C8051F338/9 Only)<br>1001: P2.2 (C8051F338/9 Only)<br>1010-1111: None |

# 12. CIP-51 Microcontroller

The MCU system controller core is the CIP-51 microcontroller. The CIP-51 is fully compatible with the MCS-51™ instruction set; standard 803x/805x assemblers and compilers can be used to develop software. The MCU family has a superset of all the peripherals included with a standard 8051. The CIP-51 also includes on-chip debug hardware (see description in Section 26), and interfaces directly with the analog and digital subsystems providing a complete data acquisition or control-system solution in a single integrated circuit.

The CIP-51 Microcontroller core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability (see Figure 12.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 25 MIPS Peak Throughput with 25 MHz Clock
- 0 to 25 MHz Clock Frequency
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- On-chip Debug Logic
- Program and Data Memory Security

**Performance**

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute, and usually have a maximum system clock of 12 MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with no instructions taking more than eight system clock cycles.

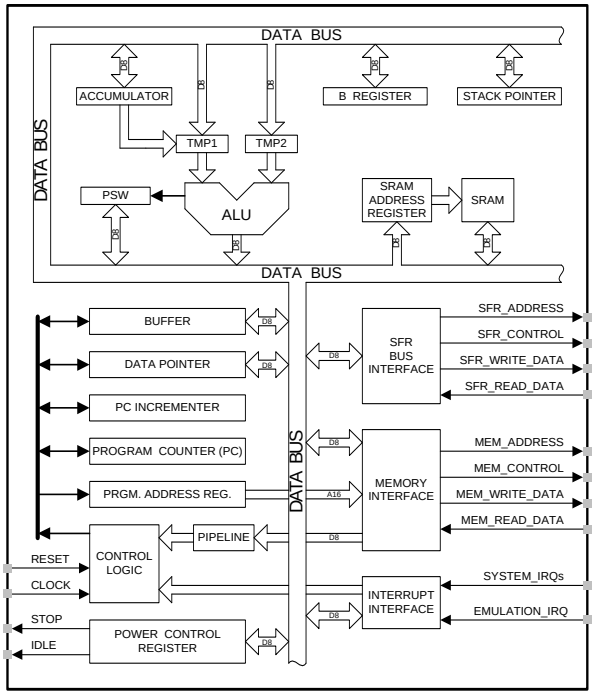



**Figure 12.1. CIP-51 Block Diagram**

With the CIP-51's maximum system clock at 25 MHz, it has a peak throughput of 25 MIPS. The CIP-51 has a total of 109 instructions. The table below shows the total number of instructions that require each execution time.

| Clocks to Execute | 1 | 2 | 2/3 | 3 | 3/4 | 4 | 4/5 | 5 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Number of Instructions | 26 | 50 | 5 | 14 | 7 | 3 | 1 | 2 | 1 |

## 12.1. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

### 12.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 12.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

## Table 12.1. CIP-51 Instruction Set Summary

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| **Arithmetic Operations** | | | |
| ADD A, Rn | Add register to A | 1 | 1 |
| ADD A, direct | Add direct byte to A | 2 | 2 |
| ADD A, @Ri | Add indirect RAM to A | 1 | 2 |
| ADD A, #data | Add immediate to A | 2 | 2 |
| ADDC A, Rn | Add register to A with carry | 1 | 1 |
| ADDC A, direct | Add direct byte to A with carry | 2 | 2 |
| ADDC A, @Ri | Add indirect RAM to A with carry | 1 | 2 |
| ADDC A, #data | Add immediate to A with carry | 2 | 2 |
| SUBB A, Rn | Subtract register from A with borrow | 1 | 1 |
| SUBB A, direct | Subtract direct byte from A with borrow | 2 | 2 |
| SUBB A, @Ri | Subtract indirect RAM from A with borrow | 1 | 2 |
| SUBB A, #data | Subtract immediate from A with borrow | 2 | 2 |
| INC A | Increment A | 1 | 1 |
| INC Rn | Increment register | 1 | 1 |
| INC direct | Increment direct byte | 2 | 2 |
| INC @Ri | Increment indirect RAM | 1 | 2 |
| DEC A | Decrement A | 1 | 1 |
| DEC Rn | Decrement register | 1 | 1 |
| DEC direct | Decrement direct byte | 2 | 2 |
| DEC @Ri | Decrement indirect RAM | 1 | 2 |
| INC DPTR | Increment Data Pointer | 1 | 1 |
| MUL AB | Multiply A and B | 1 | 4 |
| DIV AB | Divide A by B | 1 | 8 |
| DA A | Decimal adjust A | 1 | 1 |
| **Logical Operations** | | | |
| ANL A, Rn | AND Register to A | 1 | 1 |
| ANL A, direct | AND direct byte to A | 2 | 2 |
| ANL A, @Ri | AND indirect RAM to A | 1 | 2 |
| ANL A, #data | AND immediate to A | 2 | 2 |
| ANL direct, A | AND A to direct byte | 2 | 2 |
| ANL direct, #data | AND immediate to direct byte | 3 | 3 |
| ORL A, Rn | OR Register to A | 1 | 1 |
| ORL A, direct | OR direct byte to A | 2 | 2 |
| ORL A, @Ri | OR indirect RAM to A | 1 | 2 |
| ORL A, #data | OR immediate to A | 2 | 2 |
| ORL direct, A | OR A to direct byte | 2 | 2 |
| ORL direct, #data | OR immediate to direct byte | 3 | 3 |
| XRL A, Rn | Exclusive-OR Register to A | 1 | 1 |
| XRL A, direct | Exclusive-OR direct byte to A | 2 | 2 |
| XRL A, @Ri | Exclusive-OR indirect RAM to A | 1 | 2 |
| XRL A, #data | Exclusive-OR immediate to A | 2 | 2 |
| XRL direct, A | Exclusive-OR A to direct byte | 2 | 2 |

**Table 12.1. CIP-51 Instruction Set Summary (Continued)**

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| XRL direct, #data | Exclusive-OR immediate to direct byte | 3 | 3 |
| CLR A | Clear A | 1 | 1 |
| CPL A | Complement A | 1 | 1 |
| RL A | Rotate A left | 1 | 1 |
| RLC A | Rotate A left through Carry | 1 | 1 |
| RR A | Rotate A right | 1 | 1 |
| RRC A | Rotate A right through Carry | 1 | 1 |
| SWAP A | Swap nibbles of A | 1 | 1 |
| **Data Transfer** | | | |
| MOV A, Rn | Move Register to A | 1 | 1 |
| MOV A, direct | Move direct byte to A | 2 | 2 |
| MOV A, @Ri | Move indirect RAM to A | 1 | 2 |
| MOV A, #data | Move immediate to A | 2 | 2 |
| MOV Rn, A | Move A to Register | 1 | 1 |
| MOV Rn, direct | Move direct byte to Register | 2 | 2 |
| MOV Rn, #data | Move immediate to Register | 2 | 2 |
| MOV direct, A | Move A to direct byte | 2 | 2 |
| MOV direct, Rn | Move Register to direct byte | 2 | 2 |
| MOV direct, direct | Move direct byte to direct byte | 3 | 3 |
| MOV direct, @Ri | Move indirect RAM to direct byte | 2 | 2 |
| MOV direct, #data | Move immediate to direct byte | 3 | 3 |
| MOV @Ri, A | Move A to indirect RAM | 1 | 2 |
| MOV @Ri, direct | Move direct byte to indirect RAM | 2 | 2 |
| MOV @Ri, #data | Move immediate to indirect RAM | 2 | 2 |
| MOV DPTR, #data16 | Load DPTR with 16-bit constant | 3 | 3 |
| MOVC A, @A+DPTR | Move code byte relative DPTR to A | 1 | 3 |
| MOVC A, @A+PC | Move code byte relative PC to A | 1 | 3 |
| MOVX A, @Ri | Move external data (8-bit address) to A | 1 | 3 |
| MOVX @Ri, A | Move A to external data (8-bit address) | 1 | 3 |
| MOVX A, @DPTR | Move external data (16-bit address) to A | 1 | 3 |
| MOVX @DPTR, A | Move A to external data (16-bit address) | 1 | 3 |
| PUSH direct | Push direct byte onto stack | 2 | 2 |
| POP direct | Pop direct byte from stack | 2 | 2 |
| XCH A, Rn | Exchange Register with A | 1 | 1 |
| XCH A, direct | Exchange direct byte with A | 2 | 2 |
| XCH A, @Ri | Exchange indirect RAM with A | 1 | 2 |
| XCHD A, @Ri | Exchange low nibble of indirect RAM with A | 1 | 2 |
| **Boolean Manipulation** | | | |
| CLR C | Clear Carry | 1 | 1 |
| CLR bit | Clear direct bit | 2 | 2 |
| SETB C | Set Carry | 1 | 1 |
| SETB bit | Set direct bit | 2 | 2 |
| CPL C | Complement Carry | 1 | 1 |
| CPL bit | Complement direct bit | 2 | 2 |

**Table 12.1. CIP-51 Instruction Set Summary (Continued)**

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| ANL C, bit | AND direct bit to Carry | 2 | 2 |
| ANL C, /bit | AND complement of direct bit to Carry | 2 | 2 |
| ORL C, bit | OR direct bit to carry | 2 | 2 |
| ORL C, /bit | OR complement of direct bit to Carry | 2 | 2 |
| MOV C, bit | Move direct bit to Carry | 2 | 2 |
| MOV bit, C | Move Carry to direct bit | 2 | 2 |
| JC rel | Jump if Carry is set | 2 | 2/3 |
| JNC rel | Jump if Carry is not set | 2 | 2/3 |
| JB bit, rel | Jump if direct bit is set | 3 | 3/4 |
| JNB bit, rel | Jump if direct bit is not set | 3 | 3/4 |
| JBC bit, rel | Jump if direct bit is set and clear bit | 3 | 3/4 |
| **Program Branching** | | | |
| ACALL addr11 | Absolute subroutine call | 2 | 3 |
| LCALL addr16 | Long subroutine call | 3 | 4 |
| RET | Return from subroutine | 1 | 5 |
| RETI | Return from interrupt | 1 | 5 |
| AJMP addr11 | Absolute jump | 2 | 3 |
| LJMP addr16 | Long jump | 3 | 4 |
| SJMP rel | Short jump (relative address) | 2 | 3 |
| JMP @A+DPTR | Jump indirect relative to DPTR | 1 | 3 |
| JZ rel | Jump if A equals zero | 2 | 2/3 |
| JNZ rel | Jump if A does not equal zero | 2 | 2/3 |
| CJNE A, direct, rel | Compare direct byte to A and jump if not equal | 3 | 3/4 |
| CJNE A, #data, rel | Compare immediate to A and jump if not equal | 3 | 3/4 |
| CJNE Rn, #data, rel | Compare immediate to Register and jump if not equal | 3 | 3/4 |
| CJNE @Ri, #data, rel | Compare immediate to indirect and jump if not equal | 3 | 4/5 |
| DJNZ Rn, rel | Decrement Register and jump if not zero | 2 | 2/3 |
| DJNZ direct, rel | Decrement direct byte and jump if not zero | 3 | 3/4 |
| NOP | No operation | 1 | 1 |

| Notes on Registers, Operands and Addressing Modes: |
|---|
| **Rn** - Register R0–R7 of the currently selected register bank.<br><br>**@Ri** - Data RAM location addressed indirectly through R0 or R1.<br><br>**rel** - 8-bit, signed (twos complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.<br><br>**direct** - 8-bit internal data location's address. This could be a direct-access Data RAM location (0x00–0x7F) or an SFR (0x80–0xFF).<br><br>**#data** - 8-bit constant<br><br>**#data16** - 16-bit constant<br><br>**bit** - Direct-accessed bit in Data RAM or SFR<br><br>**addr11** - 11-bit destination address used by ACALL and AJMP. The destination must be within the same 2 kB page of program memory as the first byte of the following instruction.<br><br>**addr16** - 16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8 kB program memory space.<br><br>There is one unused opcode (0xA5) that performs the same function as NOP.<br>All mnemonics copyrighted © Intel Corporation 1980. |

## 12.2. CIP-51 Register Descriptions

Following are descriptions of SFRs related to the operation of the CIP-51 System Controller. Reserved bits should always be written to the value indicated in the SFR description. Future product versions may use these bits to implement new features in which case the reset value of the bit will be the indicated value, selecting the feature's default state. Detailed descriptions of the remaining SFRs are included in the sections of the data sheet associated with their corresponding system function.

### SFR Definition 12.1. DPL: Data Pointer Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DPL[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x82

| Bit | Name | Function |
|---|---|---|
| 7:0 | DPL[7:0] | **Data Pointer Low.**<br>The DPL register is the low byte of the 16-bit DPTR. |

### SFR Definition 12.2. DPH: Data Pointer High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DPH[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x83

| Bit | Name | Function |
|---|---|---|
| 7:0 | DPH[7:0] | **Data Pointer High.**<br>The DPH register is the high byte of the 16-bit DPTR. |

## SFR Definition 12.3. SP: Stack Pointer

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SP[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

SFR Address = 0x81

| Bit | Name | Function |
|---|---|---|
| 7:0 | SP[7:0] | **Stack Pointer.**<br>The Stack Pointer holds the location of the top of the stack. The stack pointer is incremented before every PUSH operation. The SP register defaults to 0x07 after reset. |

## SFR Definition 12.4. ACC: Accumulator

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ACC[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xE0; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7:0 | ACC[7:0] | **Accumulator.**<br>This register is the accumulator for arithmetic operations. |

## SFR Definition 12.5. B: B Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xF0; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7:0 | B[7:0] | **B Register.**<br>This register serves as a second accumulator for certain arithmetic operations. |

## SFR Definition 12.6. PSW: Program Status Word

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CY | AC | F0 | RS[1:0] | | OV | F1 | PARITY |
| Type | R/W | R/W | R/W | R/W | | R/W | R/W | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD0; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | CY | **Carry Flag.**<br>This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations. |
| 6 | AC | **Auxiliary Carry Flag.**<br>This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations. |
| 5 | F0 | **User Flag 0.**<br>This is a bit-addressable, general purpose flag for use under software control. |
| 4:3 | RS[1:0] | **Register Bank Select.**<br>These bits select which register bank is used during register accesses.<br>00: Bank 0, Addresses 0x00-0x07<br>01: Bank 1, Addresses 0x08-0x0F<br>10: Bank 2, Addresses 0x10-0x17<br>11: Bank 3, Addresses 0x18-0x1F |
| 2 | OV | **Overflow Flag.**<br>This bit is set to 1 under the following circumstances:<br>• An ADD, ADDC, or SUBB instruction causes a sign-change overflow.<br>• A MUL instruction results in an overflow (result is greater than 255).<br>• A DIV instruction causes a divide-by-zero condition.<br>The OV bit is cleared to 0 by the ADD, ADDC, SUBB, MUL, and DIV instructions in all other cases. |
| 1 | F1 | **User Flag 1.**<br>This is a bit-addressable, general purpose flag for use under software control. |
| 0 | PARITY | **Parity Flag.**<br>This bit is set to logic 1 if the sum of the eight bits in the accumulator is odd and cleared if the sum is even. |

## 13. Memory Organization

The memory organization of the CIP-51 System Controller is similar to that of a standard 8051. There are two separate memory spaces: program memory and data memory. Program and data memory share the same address space but are accessed via different instruction types. The memory organization of the C8051F336/7/8/9 device family is shown in Figure 13.1

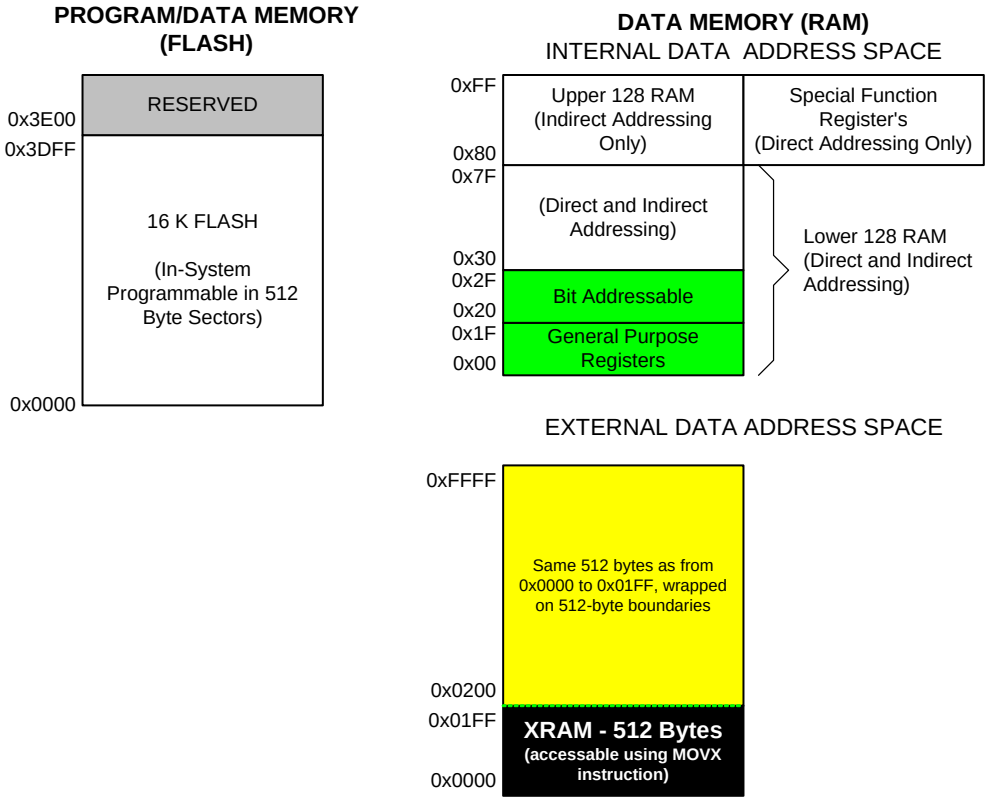



**Figure 13.1. C8051F336/7/8/9 Memory Map**

## 13.1. Program Memory

The CIP-51 core has a 64 kB program memory space. The C8051F336/7/8/9 implements 16 kB of this program memory space as in-system, re-programmable Flash memory, organized in a contiguous block from addresses 0x0000 to 0x3DFF. The address 0x3DFF serves as the security lock byte for the device, and addresses above 0x3DFF are reserved.

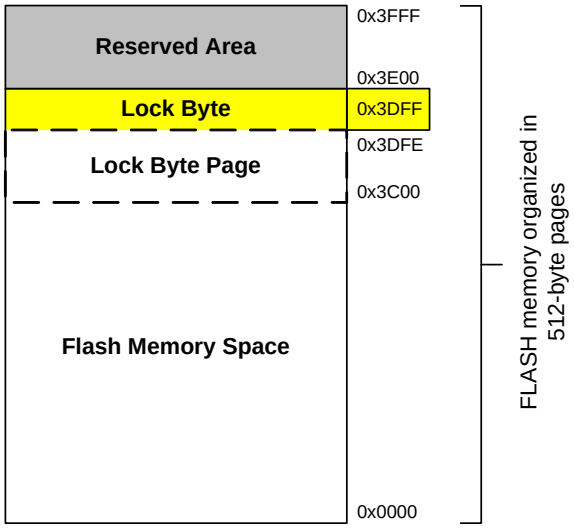



**Figure 13.2. Flash Program Memory Map**

### 13.1.1. MOVX Instruction and Program Memory

The MOVX instruction in an 8051 device is typically used to access external data memory. On the C8051F336/7/8/9 devices, the MOVX instruction is normally used to read and write on-chip XRAM, but can be re-configured to write and erase on-chip Flash memory space. MOVC instructions are always used to read Flash memory, while MOVX write instructions are used to erase and write Flash. This Flash access feature provides a mechanism for the C8051F336/7/8/9 to update program code and use the program memory space for non-volatile data storage. Refer to **Section “16. Flash Memory” on page 91** for further details.

## 13.2. Data Memory

The C8051F336/7/8/9 device family includes 768 bytes of RAM data memory. 256 bytes of this memory is mapped into the internal RAM space of the 8051. 512 bytes of this memory is on-chip “external” memory. The data memory map is shown in Figure 13.1 for reference.

### 13.2.1. Internal RAM

There are 256 bytes of internal RAM mapped into the data memory space from 0x00 through 0xFF. The lower 128 bytes of data memory are used for general purpose registers and scratch pad memory. Either direct or indirect addressing may be used to access the lower 128 bytes of data memory. Locations 0x00 through 0x1F are addressable as four banks of general purpose registers, each bank consisting of eight byte-wide registers. The next 16 bytes, locations 0x20 through 0x2F, may either be addressed as bytes or as 128 bit locations accessible with the direct addressing mode.

The upper 128 bytes of data memory are accessible only by indirect addressing. This region occupies the same address space as the Special Function Registers (SFR) but is physically separate from the SFR

space. The addressing mode used by an instruction when accessing locations above 0x7F determines whether the CPU accesses the upper 128 bytes of data memory space or the SFRs. Instructions that use direct addressing will access the SFR space. Instructions using indirect addressing above 0x7F access the upper 128 bytes of data memory. Figure 13.1 illustrates the data memory organization of the C8051F336/7/8/9.

#### 13.2.1.1. General Purpose Registers

The lower 32 bytes of data memory, locations 0x00 through 0x1F, may be addressed as four banks of general-purpose registers. Each bank consists of eight byte-wide registers designated R0 through R7. Only one of these banks may be enabled at a time. Two bits in the program status word, RS0 (PSW.3) and RS1 (PSW.4), select the active register bank (see description of the PSW in SFR Definition 12.6). This allows fast context switching when entering subroutines and interrupt service routines. Indirect addressing modes use registers R0 and R1 as index registers.

#### 13.2.1.2. Bit Addressable Locations

In addition to direct access to data memory organized as bytes, the sixteen data memory locations at 0x20 through 0x2F are also accessible as 128 individually addressable bits. Each bit has a bit address from 0x00 to 0x7F. Bit 0 of the byte at 0x20 has bit address 0x00 while bit7 of the byte at 0x20 has bit address 0x07. Bit 7 of the byte at 0x2F has bit address 0x7F. A bit access is distinguished from a full byte access by the type of instruction used (bit source or destination operands as opposed to a byte source or destination).

The MCS-51™ assembly language allows an alternate notation for bit addressing of the form XX.B where XX is the byte address and B is the bit position within the byte. For example, the instruction:

```
MOV    C, 22.3h
```

moves the Boolean value at 0x13 (bit 3 of the byte at location 0x22) into the Carry flag.

#### 13.2.1.3. Stack

A programmer's stack can be located anywhere in the 256-byte data memory. The stack area is designated using the Stack Pointer (SP) SFR. The SP will point to the last location used. The next value pushed on the stack is placed at SP+1 and then SP is incremented. A reset initializes the stack pointer to location 0x07. Therefore, the first value pushed on the stack is placed at location 0x08, which is also the first register (R0) of register bank 1. Thus, if more than one register bank is to be used, the SP should be initialized to a location in the data memory not being used for data storage. The stack depth can extend up to 256 bytes.

### 13.2.2. External RAM

There are 512 bytes of on-chip RAM mapped into the external data memory space. All of these address locations may be accessed using the external move instruction (MOVX) and the data pointer (DPTR), or using MOVX indirect addressing mode. If the MOVX instruction is used with an 8-bit address operand (such as @R1), then the high byte of the 16-bit address is provided by the External Memory Interface Control Register (EMI0CN as shown in SFR Definition 13.1). Note: the MOVX instruction is also used for writes to the Flash memory. See **Section “16. Flash Memory” on page 91** for details. The MOVX instruction accesses XRAM by default.

For a 16-bit MOVX operation (@DPTR), the upper 7 bits of the 16-bit external data memory address word are "don't cares". As a result, the 512-byte RAM is mapped modulo style over the entire 64 k external data memory address range. For example, the XRAM byte at address 0x0000 is shadowed at addresses 0x0200, 0x0400, 0x0600, 0x0800, etc. This is a useful feature when performing a linear memory fill, as the address pointer doesn't have to be reset when reaching the RAM block boundary.

## SFR Definition 13.1. EMI0CN: External Memory Interface Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | | PGSEL |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xAA

| Bit | Name | Function |
|---|---|---|
| 7:1 | UNUSED | Unused. Read = 0000000b; Write = Don't Care |
| 0 | PGSEL | **XRAM Page Select.**<br>The EMI0CN register provides the high byte of the 16-bit external data memory address when using an 8-bit MOVX command, effectively selecting a 256-byte page of RAM. Since the upper (unused) bits of the register are always zero, the PGSEL determines which page of XRAM is accessed.<br>For Example: If EMI0CN = 0x01, addresses 0x0100 through 0x01FF will be accessed. |

## 14. Special Function Registers

The direct-access data memory locations from 0x80 to 0xFF constitute the special function registers (SFRs). The SFRs provide control and data exchange with the C8051F336/7/8/9's resources and peripherals. The CIP-51 controller core duplicates the SFRs found in a typical 8051 implementation as well as implementing additional SFRs used to configure and access the sub-systems unique to the C8051F336/7/8/9. This allows the addition of new functionality while retaining compatibility with the MCS-51™ instruction set. Table 14.1 lists the SFRs implemented in the C8051F336/7/8/9 device family.

The SFR registers are accessed anytime the direct addressing mode is used to access memory locations from 0x80 to 0xFF. SFRs with addresses ending in 0x0 or 0x8 (e.g. P0, TCON, SCON0, IE, etc.) are bit-addressable as well as byte-addressable. All other SFRs are byte-addressable only. Unoccupied addresses in the SFR space are reserved for future use. Accessing these areas will have an indeterminate effect and should be avoided. Refer to the corresponding pages of the data sheet, as indicated in Table 14.2, for a detailed description of each register.

**Table 14.1. Special Function Register (SFR) Memory Map**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| F8 | SPI0CN | PCA0L | PCA0H | PCA0CPL0 | PCA0CPH0 | P0MAT | P0MASK | VDM0CN |
| F0 | B | P0MDIN | P1MDIN | P2MDIN | | | EIP1 | PCA0PWM |
| E8 | ADC0CN | PCA0CPL1 | PCA0CPH1 | PCA0CPL2 | PCA0CPH2 | P1MAT | P1MASK | RSTSRC |
| E0 | ACC | XBR0 | XBR1 | OSCLCN | IT01CF | | EIE1 | SMB0ADM |
| D8 | PCA0CN | PCA0MD | PCA0CPM0 | PCA0CPM1 | PCA0CPM2 | | | |
| D0 | PSW | REF0CN | | | P0SKIP | P1SKIP | P2SKIP | SMB0ADR |
| C8 | TMR2CN | | TMR2RLL | TMR2RLH | TMR2L | TMR2H | | |
| C0 | SMB0CN | SMB0CF | SMB0DAT | ADC0GTL | ADC0GTH | ADC0LTL | ADC0LTH | |
| B8 | IP | IDA0CN | AMX0N | AMX0P | ADC0CF | ADC0L | ADC0H | |
| B0 | | OSCXCN | OSCICN | OSCICL | | | FLSCL | FLKEY |
| A8 | IE | CLKSEL | EMI0CN | | | | | |
| A0 | P2 | SPI0CFG | SPI0CKR | SPI0DAT | P0MDOUT | P1MDOUT | P2MDOUT | |
| 98 | SCON0 | SBUF0 | | CPT0CN | | CPT0MD | | CPT0MX |
| 90 | P1 | TMR3CN | TMR3RLL | TMR3RLH | TMR3L | TMR3H | IDA0L | IDA0H |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | CKCON | PSCTL |
| 80 | P0 | SP | DPL | DPH | | | | PCON |
| | 0(8) | 1(9) | 2(A) | 3(B) | 4(C) | 5(D) | 6(E) | 7(F) |

**Note:** SFR Addresses ending in 0x0 or 0x8 are bit-addressable locations and can be used with bitwise instructions.

## Table 14.2. Special Function Registers

SFRs are listed in alphabetical order. All undefined SFR locations are reserved

| Register | Address | Description | Page |
|---|---|---|---|
| **ACC** | 0xE0 | Accumulator | 72 |
| **ADC0CF** | 0xBC | ADC0 Configuration | 41 |
| **ADC0CN** | 0xE8 | ADC0 Control | 43 |
| **ADC0GTH** | 0xC4 | ADC0 Greater-Than Compare High | 44 |
| **ADC0GTL** | 0xC3 | ADC0 Greater-Than Compare Low | 44 |
| **ADC0H** | 0xBE | ADC0 High | 42 |
| **ADC0L** | 0xBD | ADC0 Low | 42 |
| **ADC0LTH** | 0xC6 | ADC0 Less-Than Compare Word High | 45 |
| **ADC0LTL** | 0xC5 | ADC0 Less-Than Compare Word Low | 45 |
| **AMX0N** | 0xBA | AMUX0 Negative Channel Select | 50 |
| **AMX0P** | 0xBB | AMUX0 Positive Channel Select | 49 |
| **B** | 0xF0 | B Register | 72 |
| **CKCON** | 0x8E | Clock Control | 181 |
| **CLKSEL** | 0xA9 | Clock Select | 110 |
| **CPT0CN** | 0x9B | Comparator0 Control | 61 |
| **CPT0MD** | 0x9D | Comparator0 Mode Selection | 62 |
| **CPT0MX** | 0x9F | Comparator0 MUX Selection | 64 |
| **DPH** | 0x83 | Data Pointer High | 71 |
| **DPL** | 0x82 | Data Pointer Low | 71 |
| **EIE1** | 0xE6 | Extended Interrupt Enable 1 | 87 |
| **EIP1** | 0xF6 | Extended Interrupt Priority 1 | 88 |
| **EMI0CN** | 0xAA | External Memory Interface Control | 77 |
| **FLKEY** | 0xB7 | Flash Lock and Key | 98 |
| **FLSCL** | 0xB6 | Flash Scale | 99 |
| **IDA0CN** | 0xB9 | Current Mode DAC0 Control | 54 |
| **IDA0H** | 0x97 | Current Mode DAC0 High | 55 |
| **IDA0L** | 0x96 | Current Mode DAC0 Low | 55 |
| **IE** | 0xA8 | Interrupt Enable | 85 |
| **IP** | 0xB8 | Interrupt Priority | 86 |
| **IT01CF** | 0xE4 | INT0/INT1 Configuration | 90 |
| **OSCICL** | 0xB3 | Internal Oscillator Calibration | 111 |
| **OSCICN** | 0xB2 | Internal Oscillator Control | 112 |
| **OSCLCN** | 0xE3 | Low-Frequency Oscillator Control | 113 |

## Table 14.2. Special Function Registers (Continued)

SFRs are listed in alphabetical order. All undefined SFR locations are reserved

| Register | Address | Description | Page |
|---|---|---|---|
| **OSCXCN** | 0xB1 | External Oscillator Control | 115 |
| **P0** | 0x80 | Port 0 Latch | 132 |
| **P0MASK** | 0xFE | Port 0 Mask Configuration | 129 |
| **P0MAT** | 0xFD | Port 0 Match Configuration | 130 |
| **P0MDIN** | 0xF1 | Port 0 Input Mode Configuration | 132 |
| **P0MDOUT** | 0xA4 | Port 0 Output Mode Configuration | 133 |
| **P0SKIP** | 0xD4 | Port 0 Skip | 133 |
| **P1** | 0x90 | Port 1 Latch | 134 |
| **P1MASK** | 0xEE | Port 1Mask Configuration | 130 |
| **P1MAT** | 0xED | Port 1 Match Configuration | 131 |
| **P1MDIN** | 0xF2 | Port 1 Input Mode Configuration | 134 |
| **P1MDOUT** | 0xA5 | Port 1 Output Mode Configuration | 135 |
| **P1SKIP** | 0xD5 | Port 1 Skip | 135 |
| **P2** | 0xA0 | Port 2 Latch | 136 |
| **P2MDIN** | 0xF3 | Port 2 Input Mode Configuration | 136 |
| **P2MDOUT** | 0xA6 | Port 2 Output Mode Configuration | 137 |
| **P2SKIP** | 0xD6 | Port 2 Skip | 137 |
| **PCA0CN** | 0xD8 | PCA Control | 215 |
| **PCA0CPH0** | 0xFC | PCA Capture 0 High | 220 |
| **PCA0CPH1** | 0xEA | PCA Capture 1 High | 220 |
| **PCA0CPH2** | 0xEC | PCA Capture 2 High | 220 |
| **PCA0CPL0** | 0xFB | PCA Capture 0 Low | 220 |
| **PCA0CPL1** | 0xE9 | PCA Capture 1 Low | 220 |
| **PCA0CPL2** | 0xEB | PCA Capture 2 Low | 220 |
| **PCA0CPM0** | 0xDA | PCA Module 0 Mode Register | 218 |
| **PCA0CPM1** | 0xDB | PCA Module 1 Mode Register | 218 |
| **PCA0CPM2** | 0xDC | PCA Module 2 Mode Register | 218 |
| **PCA0H** | 0xFA | PCA Counter High | 219 |
| **PCA0L** | 0xF9 | PCA Counter Low | 219 |
| **PCA0MD** | 0xD9 | PCA Mode | 216 |
| **PCA0PWM** | 0xF7 | PCA PWM Configuration | 217 |
| **PCON** | 0x87 | Power Control | 108 |
| **PSCTL** | 0x8F | Program Store R/W Control | 97 |
| **PSW** | 0xD0 | Program Status Word | 73 |

## Table 14.2. Special Function Registers (Continued)

SFRs are listed in alphabetical order. All undefined SFR locations are reserved

| Register | Address | Description | Page |
|---|---|---|---|
| **REF0CN** | 0xD1 | Voltage Reference Control | 57 |
| **RSTSRC** | 0xEF | Reset Source Configuration/Status | 105 |
| **SBUF0** | 0x99 | UART0 Data Buffer | 165 |
| **SCON0** | 0x98 | UART0 Control | 164 |
| **SMB0ADM** | 0xE7 | SMBus Slave Address Mask | 149 |
| **SMB0ADR** | 0xD7 | SMBus Slave Address | 148 |
| **SMB0CF** | 0xC1 | SMBus Configuration | 144 |
| **SMB0CN** | 0xC0 | SMBus Control | 146 |
| **SMB0DAT** | 0xC2 | SMBus Data | 150 |
| **SP** | 0x81 | Stack Pointer | 72 |
| **SPI0CFG** | 0xA1 | SPI Configuration | 174 |
| **SPI0CKR** | 0xA2 | SPI Clock Rate Control | 176 |
| **SPI0CN** | 0xF8 | SPI Control | 175 |
| **SPI0DAT** | 0xA3 | SPI Data | 176 |
| **TCON** | 0x88 | Timer/Counter Control | 186 |
| **TH0** | 0x8C | Timer/Counter 0 High | 189 |
| **TH1** | 0x8D | Timer/Counter 1 High | 189 |
| **TL0** | 0x8A | Timer/Counter 0 Low | 188 |
| **TL1** | 0x8B | Timer/Counter 1 Low | 188 |
| **TMOD** | 0x89 | Timer/Counter Mode | 187 |
| **TMR2CN** | 0xC8 | Timer/Counter 2 Control | 193 |
| **TMR2H** | 0xCD | Timer/Counter 2 High | 195 |
| **TMR2L** | 0xCC | Timer/Counter 2 Low | 194 |
| **TMR2RLH** | 0xCB | Timer/Counter 2 Reload High | 194 |
| **TMR2RLL** | 0xCA | Timer/Counter 2 Reload Low | 194 |
| **TMR3CN** | 0x91 | Timer/Counter 3Control | 199 |
| **TMR3H** | 0x95 | Timer/Counter 3 High | 201 |
| **TMR3L** | 0x94 | Timer/Counter 3Low | 200 |
| **TMR3RLH** | 0x93 | Timer/Counter 3 Reload High | 200 |
| **TMR3RLL** | 0x92 | Timer/Counter 3 Reload Low | 200 |
| **VDM0CN** | 0xFF | $V_{DD}$ Monitor Control | 103 |
| **XBR0** | 0xE1 | Port I/O Crossbar Control 0 | 127 |
| **XBR1** | 0xE2 | Port I/O Crossbar Control 1 | 128 |

# 15. Interrupts

The C8051F336/7/8/9 includes an extended interrupt system supporting a total of 14 interrupt sources with two priority levels. The allocation of interrupt sources between on-chip peripherals and external input pins varies according to the specific version of the device. Each interrupt source has one or more associated interrupt-pending flag(s) located in an SFR. When a peripheral or external source meets a valid interrupt condition, the associated interrupt-pending flag is set to logic 1.

If interrupts are enabled for the source, an interrupt request is generated when the interrupt-pending flag is set. As soon as execution of the current instruction is complete, the CPU generates an LCALL to a predetermined address to begin execution of an interrupt service routine (ISR). Each ISR must end with an RETI instruction, which returns program execution to the next instruction that would have been executed if the interrupt request had not occurred. If interrupts are not enabled, the interrupt-pending flag is ignored by the hardware and program execution continues as normal. (The interrupt-pending flag is set to logic 1 regardless of the interrupt's enable/disable state.)

Each interrupt source can be individually enabled or disabled through the use of an associated interrupt enable bit in an SFR (IE–EIE1). However, interrupts must first be globally enabled by setting the EA bit (IE.7) to logic 1 before the individual interrupt enables are recognized. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings.

**Note:** Any instruction that clears a bit to disable an interrupt should be immediately followed by an instruction that has two or more opcode bytes. Using EA (global interrupt enable) as an example:

```
// in 'C':
EA = 0; // clear EA bit.
EA = 0; // this is a dummy instruction with two-byte opcode.

; in assembly:
CLR EA ; clear EA bit.
CLR EA ; this is a dummy instruction with two-byte opcode.
```

For example, if an interrupt is posted during the execution phase of a "CLR EA" opcode (or any instruction which clears a bit to disable an interrupt source), and the instruction is followed by a single-cycle instruction, the interrupt may be taken. However, a read of the enable bit will return a '0' inside the interrupt service routine. When the bit-clearing opcode is followed by a multi-cycle instruction, the interrupt will not be taken.

Some interrupt-pending flags are automatically cleared by the hardware when the CPU vectors to the ISR. However, most are not cleared by the hardware and must be cleared by software before returning from the ISR. If an interrupt-pending flag remains set after the CPU completes the return-from-interrupt (RETI) instruction, a new interrupt request will be generated immediately and the CPU will re-enter the ISR after the completion of the next instruction.

## 15.1. MCU Interrupt Sources and Vectors

The C8051F336/7/8/9 MCUs support 14 interrupt sources. Software can simulate an interrupt by setting any interrupt-pending flag to logic 1. If interrupts are enabled for the flag, an interrupt request will be generated and the CPU will vector to the ISR address associated with the interrupt-pending flag. MCU interrupt sources, associated vector addresses, priority order and control bits are summarized in Table 15.1. Refer to the datasheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

### 15.1.1. Interrupt Priorities

Each interrupt source can be individually programmed to one of two priority levels: low or high. A low priority interrupt service routine can be preempted by a high priority interrupt. A high priority interrupt cannot be preempted. Each interrupt has an associated interrupt priority bit in an SFR (IP or EIP1) used to configure its priority level. Low priority is the default. If two interrupts are recognized simultaneously, the interrupt with the higher priority is serviced first. If both interrupts have the same priority level, a fixed priority order is used to arbitrate, given in Table 15.1.

### 15.1.2. Interrupt Latency

Interrupt response time depends on the state of the CPU when the interrupt occurs. Pending interrupts are sampled and priority decoded each system clock cycle. Therefore, the fastest possible response time is 5 system clock cycles: 1 clock cycle to detect the interrupt and 4 clock cycles to complete the LCALL to the ISR. If an interrupt is pending when a RETI is executed, a single instruction is executed before an LCALL is made to service the pending interrupt. Therefore, the maximum response time for an interrupt (when no other interrupt is currently being serviced or the new interrupt is of greater priority) occurs when the CPU is performing an RETI instruction followed by a DIV as the next instruction. In this case, the response time is 18 system clock cycles: 1 clock cycle to detect the interrupt, 5 clock cycles to execute the RETI, 8 clock cycles to complete the DIV instruction and 4 clock cycles to execute the LCALL to the ISR. If the CPU is executing an ISR for an interrupt with equal or higher priority, the new interrupt will not be serviced until the current ISR completes, including the RETI and following instruction.

**Table 15.1. Interrupt Summary**

| Interrupt Source | Interrupt Vector | Priority Order | Pending Flag | Bit addressable? | Cleared by HW? | Enable Flag | Priority Control |
|---|---|---|---|---|---|---|---|
| Reset | 0x0000 | Top | None | N/A | N/A | Always Enabled | Always Highest |
| External Interrupt 0 (/INT0) | 0x0003 | 0 | IE0 (TCON.1) | Y | Y | EX0 (IE.0) | PX0 (IP.0) |
| Timer 0 Overflow | 0x000B | 1 | TF0 (TCON.5) | Y | Y | ET0 (IE.1) | PT0 (IP.1) |
| External Interrupt 1 (/INT1) | 0x0013 | 2 | IE1 (TCON.3) | Y | Y | EX1 (IE.2) | PX1 (IP.2) |
| Timer 1 Overflow | 0x001B | 3 | TF1 (TCON.7) | Y | Y | ET1 (IE.3) | PT1 (IP.3) |
| UART0 | 0x0023 | 4 | RI0 (SCON0.0)<br>TI0 (SCON0.1) | Y | N | ES0 (IE.4) | PS0 (IP.4) |
| Timer 2 Overflow | 0x002B | 5 | TF2H (TMR2CN.7)<br>TF2L (TMR2CN.6) | Y | N | ET2 (IE.5) | PT2 (IP.5) |
| SPI0 | 0x0033 | 6 | SPIF (SPI0CN.7)<br>WCOL (SPI0CN.6)<br>MODF (SPI0CN.5)<br>RXOVRN (SPI0CN.4) | Y | N | ESPI0 (IE.6) | PSPI0 (IP.6) |
| SMB0 | 0x003B | 7 | SI (SMB0CN.0) | Y | N | ESMB0 (EIE1.0) | PSMB0 (EIP1.0) |
| Port Match | 0x0043 | 8 | None | N/A | N/A | EMAT (EIE1.1) | PMAT (EIP1.1) |
| ADC0 Window Compare | 0x004B | 9 | AD0WINT (ADC0CN.3) | Y | N | EWADC0 (EIE1.2) | PWADC0 (EIP1.2) |
| ADC0 Conversion Complete | 0x0053 | 10 | AD0INT (ADC0CN.5) | Y | N | EADC0 (EIE1.3) | PADC0 (EIP1.3) |
| Programmable Counter Array | 0x005B | 11 | CF (PCA0CN.7)<br>CCFn (PCA0CN.n)<br>COVF (PCA0PWM.6) | Y | N | EPCA0 (EIE1.4) | PPCA0 (EIP1.4) |
| Comparator0 | 0x0063 | 12 | CP0FIF (CPT0CN.4)<br>CP0RIF (CPT0CN.5) | N | N | ECP0 (EIE1.5) | PCP0 (EIP1.5) |
| RESERVED | 0x006B | 13 | N/A | N/A | N/A | N/A | N/A |
| Timer 3 Overflow | 0x0073 | 14 | TF3H (TMR3CN.7)<br>TF3L (TMR3CN.6) | N | N | ET3 (EIE1.7) | PT3 (EIP1.7) |

## 15.2. Interrupt Register Descriptions

The SFRs used to enable the interrupt sources and set their priority level are described in this section. Refer to the data sheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

## SFR Definition 15.1. IE: Interrupt Enable

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | EA | ESPI0 | ET2 | ES0 | ET1 | EX1 | ET0 | EX0 |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA8; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | EA | **Enable All Interrupts.**<br>Globally enables/disables all interrupts. It overrides individual interrupt mask settings.<br>0: Disable all interrupt sources.<br>1: Enable each interrupt according to its individual mask setting. |
| 6 | ESPI0 | **Enable Serial Peripheral Interface (SPI0) Interrupt.**<br>This bit sets the masking of the SPI0 interrupts.<br>0: Disable all SPI0 interrupts.<br>1: Enable interrupt requests generated by SPI0. |
| 5 | ET2 | **Enable Timer 2 Interrupt.**<br>This bit sets the masking of the Timer 2 interrupt.<br>0: Disable Timer 2 interrupt.<br>1: Enable interrupt requests generated by the TF2L or TF2H flags. |
| 4 | ES0 | **Enable UART0 Interrupt.**<br>This bit sets the masking of the UART0 interrupt.<br>0: Disable UART0 interrupt.<br>1: Enable UART0 interrupt. |
| 3 | ET1 | **Enable Timer 1 Interrupt.**<br>This bit sets the masking of the Timer 1 interrupt.<br>0: Disable all Timer 1 interrupt.<br>1: Enable interrupt requests generated by the TF1 flag. |
| 2 | EX1 | **Enable External Interrupt 1.**<br>This bit sets the masking of External Interrupt 1.<br>0: Disable external interrupt 1.<br>1: Enable interrupt requests generated by the /INT1 input. |
| 1 | ET0 | **Enable Timer 0 Interrupt.**<br>This bit sets the masking of the Timer 0 interrupt.<br>0: Disable all Timer 0 interrupt.<br>1: Enable interrupt requests generated by the TF0 flag. |
| 0 | EX0 | **Enable External Interrupt 0.**<br>This bit sets the masking of External Interrupt 0.<br>0: Disable external interrupt 0.<br>1: Enable interrupt requests generated by the /INT0 input. |

## SFR Definition 15.2. IP: Interrupt Priority

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | PSPI0 | PT2 | PS0 | PT1 | PX1 | PT0 | PX0 |
| Type | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xB8; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | UNUSED | Unused. Read = 1, Write = Don't Care. |
| 6 | PSPI0 | **Serial Peripheral Interface (SPI0) Interrupt Priority Control.**<br>This bit sets the priority of the SPI0 interrupt.<br>0: SPI0 interrupt set to low priority level.<br>1: SPI0 interrupt set to high priority level. |
| 5 | PT2 | **Timer 2 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 2 interrupt.<br>0: Timer 2 interrupt set to low priority level.<br>1: Timer 2 interrupt set to high priority level. |
| 4 | PS0 | **UART0 Interrupt Priority Control.**<br>This bit sets the priority of the UART0 interrupt.<br>0: UART0 interrupt set to low priority level.<br>1: UART0 interrupt set to high priority level. |
| 3 | PT1 | **Timer 1 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 1 interrupt.<br>0: Timer 1 interrupt set to low priority level.<br>1: Timer 1 interrupt set to high priority level. |
| 2 | PX1 | **External Interrupt 1 Priority Control.**<br>This bit sets the priority of the External Interrupt 1 interrupt.<br>0: External Interrupt 1 set to low priority level.<br>1: External Interrupt 1 set to high priority level. |
| 1 | PT0 | **Timer 0 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 0 interrupt.<br>0: Timer 0 interrupt set to low priority level.<br>1: Timer 0 interrupt set to high priority level. |
| 0 | PX0 | **External Interrupt 0 Priority Control.**<br>This bit sets the priority of the External Interrupt 0 interrupt.<br>0: External Interrupt 0 set to low priority level.<br>1: External Interrupt 0 set to high priority level. |

## SFR Definition 15.3. EIE1: Extended Interrupt Enable 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ET3 | Reserved | ECP0 | EPCA0 | EADC0 | EWADC0 | EMAT | ESMB0 |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xE6

| Bit | Name | Function |
|---|---|---|
| 7 | ET3 | **Enable Timer 3 Interrupt.**<br>This bit sets the masking of the Timer 3 interrupt.<br>0: Disable Timer 3 interrupts.<br>1: Enable interrupt requests generated by the TF3L or TF3H flags. |
| 6 | Reserved | Reserved. Must Write 0. |
| 5 | ECP0 | **Enable Comparator0 (CP0) Interrupt.**<br>This bit sets the masking of the CP0 interrupt.<br>0: Disable CP0 interrupts.<br>1: Enable interrupt requests generated by the CP0RIF or CP0FIF flags. |
| 4 | EPCA0 | **Enable Programmable Counter Array (PCA0) Interrupt.**<br>This bit sets the masking of the PCA0 interrupts.<br>0: Disable all PCA0 interrupts.<br>1: Enable interrupt requests generated by PCA0. |
| 3 | EADC0 | **Enable ADC0 Conversion Complete Interrupt.**<br>This bit sets the masking of the ADC0 Conversion Complete interrupt.<br>0: Disable ADC0 Conversion Complete interrupt.<br>1: Enable interrupt requests generated by the AD0INT flag. |
| 2 | EWADC0 | **Enable Window Comparison ADC0 Interrupt.**<br>This bit sets the masking of ADC0 Window Comparison interrupt.<br>0: Disable ADC0 Window Comparison interrupt.<br>1: Enable interrupt requests generated by ADC0 Window Compare flag (AD0WINT). |
| 1 | EMAT | **Enable Port Match Interrupts.**<br>This bit sets the masking of the Port Match Event interrupt.<br>0: Disable all Port Match interrupts.<br>1: Enable interrupt requests generated by a Port Match. |
| 0 | ESMB0 | **Enable SMBus (SMB0) Interrupt.**<br>This bit sets the masking of the SMB0 interrupt.<br>0: Disable all SMB0 interrupts.<br>1: Enable interrupt requests generated by SMB0. |

## SFR Definition 15.4. EIP1: Extended Interrupt Priority 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PT3 | Reserved | PCP0 | PPCA0 | PADC0 | PWADC0 | PMAT | PSMB0 |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xF6

| Bit | Name | Function |
|---|---|---|
| 7 | PT3 | **Timer 3 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 3 interrupt.<br>0: Timer 3 interrupts set to low priority level.<br>1: Timer 3 interrupts set to high priority level. |
| 6 | Reserved | Reserved. Must Write 0. |
| 5 | PCP0 | **Comparator0 (CP0) Interrupt Priority Control.**<br>This bit sets the priority of the CP0 interrupt.<br>0: CP0 interrupt set to low priority level.<br>1: CP0 interrupt set to high priority level. |
| 4 | PPCA0 | **Programmable Counter Array (PCA0) Interrupt Priority Control.**<br>This bit sets the priority of the PCA0 interrupt.<br>0: PCA0 interrupt set to low priority level.<br>1: PCA0 interrupt set to high priority level. |
| 3 | PADC0 | **ADC0 Conversion Complete Interrupt Priority Control.**<br>This bit sets the priority of the ADC0 Conversion Complete interrupt.<br>0: ADC0 Conversion Complete interrupt set to low priority level.<br>1: ADC0 Conversion Complete interrupt set to high priority level. |
| 2 | PWADC0 | **ADC0 Window Comparator Interrupt Priority Control.**<br>This bit sets the priority of the ADC0 Window interrupt.<br>0: ADC0 Window interrupt set to low priority level.<br>1: ADC0 Window interrupt set to high priority level. |
| 1 | PMAT | **Port Match Interrupt Priority Control.**<br>This bit sets the priority of the Port Match Event interrupt.<br>0: Port Match interrupt set to low priority level.<br>1: Port Match interrupt set to high priority level. |
| 0 | PSMB0 | **SMBus (SMB0) Interrupt Priority Control.**<br>This bit sets the priority of the SMB0 interrupt.<br>0: SMB0 interrupt set to low priority level.<br>1: SMB0 interrupt set to high priority level. |

## 15.3. External Interrupts /INT0 and /INT1

The /INT0 and /INT1 external interrupt sources are configurable as active high or low, edge or level sensitive. The IN0PL (/INT0 Polarity) and IN1PL (/INT1 Polarity) bits in the IT01CF register select active high or active low; the IT0 and IT1 bits in TCON (**Section "24.1. Timer 0 and Timer 1" on page 182**) select level or edge sensitive. The table below lists the possible configurations.

| IT0 | IN0PL | /INT0 Interrupt |
|---|---|---|
| 1 | 0 | Active low, edge sensitive |
| 1 | 1 | Active high, edge sensitive |
| 0 | 0 | Active low, level sensitive |
| 0 | 1 | Active high, level sensitive |

| IT1 | IN1PL | /INT1 Interrupt |
|---|---|---|
| 1 | 0 | Active low, edge sensitive |
| 1 | 1 | Active high, edge sensitive |
| 0 | 0 | Active low, level sensitive |
| 0 | 1 | Active high, level sensitive |

/INT0 and /INT1 are assigned to Port pins as defined in the IT01CF register (see SFR Definition 15.5). Note that /INT0 and /INT0 Port pin assignments are independent of any Crossbar assignments. /INT0 and /INT1 will monitor their assigned Port pins without disturbing the peripheral that was assigned the Port pin via the Crossbar. To assign a Port pin only to /INT0 and/or /INT1, configure the Crossbar to skip the selected pin(s). This is accomplished by setting the associated bit in register XBR0 (see **Section "20.3. Priority Crossbar Decoder" on page 124** for complete details on configuring the Crossbar).

IE0 (TCON.1) and IE1 (TCON.3) serve as the interrupt-pending flags for the /INT0 and /INT1 external interrupts, respectively. If an /INT0 or /INT1 external interrupt is configured as edge-sensitive, the corresponding interrupt-pending flag is automatically cleared by the hardware when the CPU vectors to the ISR. When configured as level sensitive, the interrupt-pending flag remains logic 1 while the input is active as defined by the corresponding polarity bit (IN0PL or IN1PL); the flag remains logic 0 while the input is inactive. The external interrupt source must hold the input active until the interrupt request is recognized. It must then deactivate the interrupt request before execution of the ISR completes or another interrupt request will be generated.

## SFR Definition 15.5. IT01CF: INT0/INT1 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IN1PL | IN1SL[2:0] | | | IN0PL | IN0SL[2:0] | | |
| Type | R/W | R/W | | | R/W | R/W | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

SFR Address = 0xE4

| Bit | Name | Function |
|---|---|---|
| 7 | IN1PL | **/INT1 Polarity.**<br>0: /INT1 input is active low.<br>1: /INT1 input is active high. |
| 6:4 | IN1SL[2:0] | **/INT1 Port Pin Selection Bits.**<br>These bits select which Port pin is assigned to /INT1. Note that this pin assignment is independent of the Crossbar; /INT1 will monitor the assigned Port pin without disturbing the peripheral that has been assigned the Port pin via the Crossbar. The Crossbar will not assign the Port pin to a peripheral if it is configured to skip the selected pin.<br>000: Select P0.0<br>001: Select P0.1<br>010: Select P0.2<br>011: Select P0.3<br>100: Select P0.4<br>101: Select P0.5<br>110: Select P0.6<br>111: Select P0.7 |
| 3 | IN0PL | **/INT0 Polarity.**<br>0: /INT0 input is active low.<br>1: /INT0 input is active high. |
| 2:0 | IN0SL[2:0] | **/INT0 Port Pin Selection Bits.**<br>These bits select which Port pin is assigned to /INT0. Note that this pin assignment is independent of the Crossbar; /INT0 will monitor the assigned Port pin without disturbing the peripheral that has been assigned the Port pin via the Crossbar. The Crossbar will not assign the Port pin to a peripheral if it is configured to skip the selected pin.<br>000: Select P0.0<br>001: Select P0.1<br>010: Select P0.2<br>011: Select P0.3<br>100: Select P0.4<br>101: Select P0.5<br>110: Select P0.6<br>111: Select P0.7 |

# 16. Flash Memory

On-chip, re-programmable Flash memory is included for program code and non-volatile data storage. The Flash memory can be programmed in-system, a single byte at a time, through the C2 interface or by software using the MOVX instruction. Once cleared to logic 0, a Flash bit must be erased to set it back to logic 1. Flash bytes would typically be erased (set to 0xFF) before being reprogrammed. The write and erase operations are automatically timed by hardware for proper execution; data polling to determine the end of the write/erase operation is not required. Code execution is stalled during a Flash write/erase operation. Refer to **Section "6. Electrical Characteristics" on page 27** for complete Flash memory electrical characteristics.

## 16.1. Programming The Flash Memory

The simplest means of programming the Flash memory is through the C2 interface using programming tools provided by Silicon Labs or a third party vendor. This is the only means for programming a non-initialized device. For details on the C2 commands to program Flash memory, see **Section "26. C2 Interface" on page 221**.

**To ensure the integrity of Flash contents, it is strongly recommended that the on-chip $V_{DD}$ Monitor be enabled in any system that includes code that writes and/or erases Flash memory from software. See Section 16.4 for more details.**

### 16.1.1. Flash Lock and Key Functions

Flash writes and erases by user software are protected with a lock and key function. The Flash Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before Flash operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, Flash writes and erases will be disabled until the next system reset. Flash writes and erases will also be disabled if a Flash write or erase is attempted before the key codes have been written properly. The Flash lock resets after each write or erase; the key codes must be written again before a following Flash operation can be performed. The FLKEY register is detailed in SFR Definition 16.2.

### 16.1.2. Flash Erase Procedure

The Flash memory can be programmed by software using the MOVX write instruction with the address and data byte to be programmed provided as normal operands. Before writing to Flash memory using MOVX, Flash write operations must be enabled by: (1) setting the PSWE Program Store Write Enable bit (PSCTL.0) to logic 1 (this directs the MOVX writes to target Flash memory); and (2) Writing the Flash key codes in sequence to the Flash Lock register (FLKEY). The PSWE bit remains set until cleared by software.

A write to Flash memory can clear bits to logic 0 but cannot set them; only an erase operation can set bits to logic 1 in Flash. **A byte location to be programmed should be erased before a new value is written.** The Flash memory is organized in 512-byte pages. The erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire 512-byte page, perform the following steps:

1. Disable interrupts (recommended).
2. Set thePSEE bit (register PSCTL).
3. Set the PSWE bit (register PSCTL).
4. Write the first key code to FLKEY: 0xA5.
5. Write the second key code to FLKEY: 0xF1.
6. Using the MOVX instruction, write a data byte to any location within the 512-byte page to be erased.
7. Clear the PSWE and PSEE bits.

### 16.1.3. Flash Write Procedure

Flash bytes are programmed by software with the following sequence:

1. Disable interrupts (recommended).
2. Erase the 512-byte Flash page containing the target location, as described in **Section 16.1.2**.
3. Set the PSWE bit (register PSCTL).
4. Clear the PSEE bit (register PSCTL).
5. Write the first key code to FLKEY: 0xA5.
6. Write the second key code to FLKEY: 0xF1.
7. Using the MOVX instruction, write a single data byte to the desired location within the 512-byte sector.
8. Clear the PSWE bit.

Steps 5–7 must be repeated for each byte to be written. After Flash writes are complete, PSWE should be cleared so that MOVX instructions do not target program memory.

## 16.2. Non-volatile Data Storage

The Flash memory can be used for non-volatile data storage as well as program code. This allows data such as calibration coefficients to be calculated and stored at run time. Data is written using the MOVX write instruction and read using the MOVC instruction. Note: MOVX read instructions always target XRAM.

## 16.3. Security Options

The CIP-51 provides security options to protect the Flash memory from inadvertent modification by software as well as to prevent the viewing of proprietary program code and constants. The Program Store Write Enable (bit PSWE in register PSCTL) and the Program Store Erase Enable (bit PSEE in register PSCTL) bits protect the Flash memory from accidental modification by software. PSWE must be explicitly set to '1' before software can modify the Flash memory; both PSWE and PSEE must be set to '1' before software can erase Flash memory. Additional security features prevent proprietary program code and data constants from being read or altered across the C2 interface.

A Security Lock Byte located in Flash user space offers protection of the Flash program memory from access (reads, writes, or erases) by unprotected code or the C2 interface. See **Section "13. Memory Organization" on page 74** for the location of the security byte. The Flash security mechanism allows the user to lock *n* 512-byte Flash pages, starting at page 0 (addresses 0x0000 to 0x01FF), where *n* is the 1's complement number represented by the Security Lock Byte. **Note that the page containing the Flash Security Lock Byte is unlocked when no other Flash pages are locked (all bits of the Lock Byte are '1') and locked when any other Flash pages are locked (any bit of the Lock Byte is '0').** An example is shown in Figure 16.1.

| | |
|---|---|
| Security Lock Byte: | 11111101b |
| 1s Complement: | 00000010b |
| Flash pages locked: | 3 (First two Flash pages + Lock Byte Page) |

**Figure 16.1. Security Byte Decoding**

The level of Flash security depends on the Flash access method. The three Flash access methods that can be restricted are reads, writes, and erases from the C2 debug interface, user firmware executing on unlocked pages, and user firmware executing on locked pages. Table 16.1 summarizes the Flash security features of the C8051F336/7/8/9 devices.

**Table 16.1. Flash Security Summary**

| Action | C2 Debug Interface | User Firmware executing from: | |
|---|---|---|---|
| | | an unlocked page | a locked page |
| Read, Write or Erase unlocked pages (except page with Lock Byte) | Permitted | Permitted | Permitted |
| Read, Write or Erase locked pages (except page with Lock Byte) | Not Permitted | Flash Error Reset | Permitted |
| Read or Write page containing Lock Byte (if no pages are locked) | Permitted | Permitted | Permitted |
| Read or Write page containing Lock Byte (if any page is locked) | Not Permitted | Flash Error Reset | Permitted |
| Read contents of Lock Byte (if no pages are locked) | Permitted | Permitted | Permitted |
| Read contents of Lock Byte (if any page is locked) | Not Permitted | Flash Error Reset | Permitted |
| Erase page containing Lock Byte (if no pages are locked) | Permitted | Flash Error Reset | Flash Error Reset |
| Erase page containing Lock Byte—Unlock all pages (if any page is locked) | C2 Device Erase Only | Flash Error Reset | Flash Error Reset |
| Lock additional pages (change '1's to '0's in the Lock Byte) | Not Permitted | Flash Error Reset | Flash Error Reset |
| Unlock individual pages (change '0's to '1's in the Lock Byte) | Not Permitted | Flash Error Reset | Flash Error Reset |
| Read, Write or Erase Reserved Area | Not Permitted | Flash Error Reset | Flash Error Reset |

C2 Device Erase - Erases all Flash pages including the page containing the Lock Byte.

Flash Error Reset - Not permitted; Causes Flash Error Device Reset (FERROR bit in RSTSRC is '1' after reset).

- All prohibited operations that are performed via the C2 interface are ignored (do not cause device reset).
- Locking any Flash page also locks the page containing the Lock Byte.
- Once written to, the Lock Byte cannot be modified except by performing a C2 Device Erase.
- If user code writes to the Lock Byte, the Lock does not take effect until the next device reset.

## 16.4. Flash Write and Erase Guidelines

Any system which contains routines which write or erase Flash memory from software involves some risk that the write or erase routines will execute unintentionally if the CPU is operating outside its specified operating range of $V_{DD}$, system clock frequency, or temperature. This accidental execution of Flash modifying code can result in alteration of Flash memory contents causing a system failure that is only recoverable by re-Flashing the code in the device.

The following guidelines are recommended for any system which contains routines which write or erase Flash from code.

### 16.4.1. $V_{DD}$ Maintenance and the $V_{DD}$ monitor

1. If the system power supply is subject to voltage or current "spikes," add sufficient transient protection devices to the power supply to ensure that the supply voltages listed in the Absolute Maximum Ratings table are not exceeded.
2. Make certain that the minimum $V_{DD}$ rise time specification of 1 ms is met. If the system cannot meet this rise time specification, then add an external $V_{DD}$ brownout circuit to the $\overline{RST}$ pin of the device that holds the device in reset until $V_{DD}$ reaches 2.7 V and re-asserts $\overline{RST}$ if $V_{DD}$ drops below 2.7 V.
3. Enable the on-chip $V_{DD}$ monitor and enable the $V_{DD}$ monitor as a reset source as early in code as possible. This should be the first set of instructions executed after the Reset Vector. For 'C'-based systems, this will involve modifying the startup code added by the 'C' compiler. See your compiler documentation for more details. Make certain that there are no delays in software between enabling the $V_{DD}$ monitor and enabling the $V_{DD}$ monitor as a reset source. Code examples showing this can be found in “AN201: Writing to Flash from Firmware", available from the Silicon Laboratories web site.
4. As an added precaution, explicitly enable the $V_{DD}$ monitor and enable the $V_{DD}$ monitor as a reset source inside the functions that write and erase Flash memory. The $V_{DD}$ monitor enable instructions should be placed just after the instruction to set PSWE to a '1', but before the Flash write or erase operation instruction.
5. Make certain that all writes to the RSTSRC (Reset Sources) register use direct assignment operators and explicitly DO NOT use the bit-wise operators (such as AND or OR). For example, "RSTSRC = 0x02" is correct. "RSTSRC |= 0x02" is incorrect.
6. Make certain that all writes to the RSTSRC register explicitly set the PORSF bit to a '1'. Areas to check are initialization code which enables other reset sources, such as the Missing Clock Detector or Comparator, for example, and instructions which force a Software Reset. A global search on "RSTSRC" can quickly verify this.

### 16.4.2. PSWE Maintenance

7. Reduce the number of places in code where the PSWE bit (b0 in PSCTL) is set to a '1'. There should be exactly one routine in code that sets PSWE to a '1' to write Flash bytes and one routine in code that sets PSWE and PSEE both to a '1' to erase Flash pages.
8. Minimize the number of variable accesses while PSWE is set to a '1'. Handle pointer address updates and loop variable maintenance outside the "PSWE = 1;... PSWE = 0;" area. Code examples showing this can be found in *AN201, "Writing to Flash from Firmware"*, available from the Silicon Laboratories web site.
9. Disable interrupts prior to setting PSWE to a '1' and leave them disabled until after PSWE has been reset to '0'. Any interrupts posted during the Flash write or erase operation will be serviced in priority order after the Flash operation has been completed and interrupts have been re-enabled by software.
10. Make certain that the Flash write and erase pointer variables are not located in XRAM. See your compiler documentation for instructions regarding how to explicitly locate variables in different memory

areas.

11. Add address bounds checking to the routines that write or erase Flash memory to ensure that a routine called with an illegal address does not result in modification of the Flash.

### 16.4.3. System Clock

12. If operating from an external crystal, be advised that crystal performance is susceptible to electrical interference and is sensitive to layout and to changes in temperature. If the system is operating in an electrically noisy environment, use the internal oscillator or use an external CMOS clock.
13. If operating from the external oscillator, switch to the internal oscillator during Flash write or erase operations. The external oscillator can continue to run, and the CPU can switch back to the external oscillator after the Flash operation has completed.

Additional Flash recommendations and example code can be found in *AN201, "Writing to Flash from Firmware"*, available from the Silicon Laboratories web site.

## SFR Definition 16.1. PSCTL: Program Store R/W Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | PSEE | PSWE |
| Type | R | R | R | R | R | R | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8F

| Bit | Name | Function |
|---|---|---|
| 7:2 | UNUSED | Unused. Read = 000000b, Write = don't care. |
| 1 | PSEE | **Program Store Erase Enable**<br>Setting this bit (in combination with PSWE) allows an entire page of Flash program memory to be erased. If this bit is logic 1 and Flash writes are enabled (PSWE is logic 1), a write to Flash memory using the MOVX instruction will erase the entire page that contains the location addressed by the MOVX instruction. The value of the data byte written does not matter.<br>0: Flash program memory erasure disabled.<br>1: Flash program memory erasure enabled. |
| 0 | PSWE | **Program Store Write Enable**<br>Setting this bit allows writing a byte of data to the Flash program memory using the MOVX write instruction. The Flash location should be erased before writing data.<br>0: Writes to Flash program memory disabled.<br>1: Writes to Flash program memory enabled; the MOVX write instruction targets Flash memory. |

## SFR Definition 16.2. FLKEY: Flash Lock and Key

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FLKEY[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xB7

| Bit | Name | Function |
|---|---|---|
| 7:0 | FLKEY[7:0] | **Flash Lock and Key Register.**<br>Write:<br>This register provides a lock and key function for Flash erasures and writes. Flash writes and erases are enabled by writing 0xA5 followed by 0xF1 to the FLKEY register. Flash writes and erases are automatically disabled after the next write or erase is complete. If any writes to FLKEY are performed incorrectly, or if a Flash write or erase operation is attempted while these operations are disabled, the Flash will be permanently locked from writes or erasures until the next device reset. If an application never writes to Flash, it can intentionally lock the Flash by writing a non-0xA5 value to FLKEY from software.<br>Read:<br>When read, bits 1–0 indicate the current Flash lock state.<br>00: Flash is write/erase locked.<br>01: The first key code has been written (0xA5).<br>10: Flash is unlocked (writes/erases allowed).<br>11: Flash writes/erases disabled until the next reset. |

## SFR Definition 16.3. FLSCL: Flash Scale

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FOSE | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved | Reserved |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xB6

| Bit | Name | Function |
|---|---|---|
| 7 | FOSE | **Flash One-shot Enable**<br>This bit enables the Flash read one-shot (recommended). If the Flash one-shot is disabled, the Flash sense amps are enabled for a full clock cycle during Flash reads, increasing the device power consumption.<br>0: Flash one-shot disabled.<br>1: Flash one-shot enabled. |
| 6:0 | Reserved | Reserved. Must Write 0000000b. |

## 17. Reset Sources

Reset circuitry allows the controller to be easily placed in a predefined default condition. Upon entering this reset state, the following events occur:

- CIP-51 halts program execution
- Special Function Registers (SFRs) are initialized to their defined reset values
- External Port pins are forced to a known state
- Interrupts and timers are disabled.

All SFRs are reset to the predefined values noted in the SFR detailed descriptions. The contents of internal data memory are unaffected during a reset; any previously stored data is preserved. However, since the stack pointer SFR is reset, the stack is effectively lost, even though the data on the stack is not altered.

The Port I/O latches are reset to 0xFF (all logic ones) in open-drain mode. Weak pullups are enabled during and after the reset. For $V_{DD}$ Monitor and power-on resets, the $\overline{RST}$ pin is driven low until the device exits the reset state.

On exit from the reset state, the program counter (PC) is reset, and the system clock defaults to the internal oscillator. The Watchdog Timer is enabled with the system clock divided by 12 as its clock source. Program execution begins at location 0x0000.

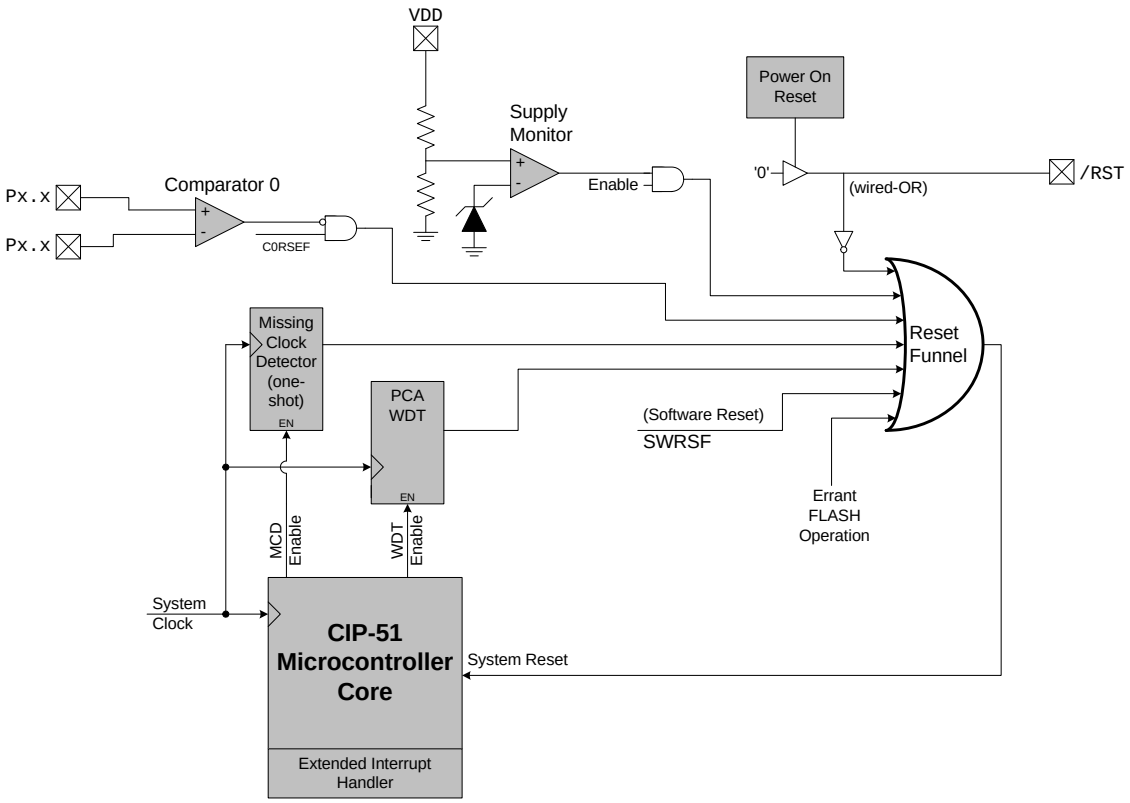



**Figure 17.1. Reset Sources**

## 17.1. Power-On Reset

During power-up, the device is held in a reset state and the $\overline{\text{RST}}$ pin is driven low until $V_{DD}$ settles above $V_{RST}$. A delay occurs before the device is released from reset; the delay decreases as the $V_{DD}$ ramp time increases ($V_{DD}$ ramp time is defined as how fast $V_{DD}$ ramps from 0 V to $V_{RST}$). Figure 17.2. plots the power-on and $V_{DD}$ monitor reset timing. The maximum $V_{DD}$ ramp time is 1 ms; slower ramp times may cause the device to be released from reset before $V_{DD}$ reaches the $V_{RST}$ level. For ramp times less than 1 ms, the power-on reset delay ($T_{PORDelay}$) is typically less than 0.3 ms.

On exit from a power-on reset, the PORSF flag (RSTSRC.1) is set by hardware to logic 1. When PORSF is set, all of the other reset flags in the RSTSRC Register are indeterminate (PORSF is cleared by all other resets). Since all resets cause program execution to begin at the same location (0x0000) software can read the PORSF flag to determine if a power-up was the cause of reset. The content of internal data memory should be assumed to be undefined after a power-on reset. The $V_{DD}$ monitor is enabled following a power-on reset.

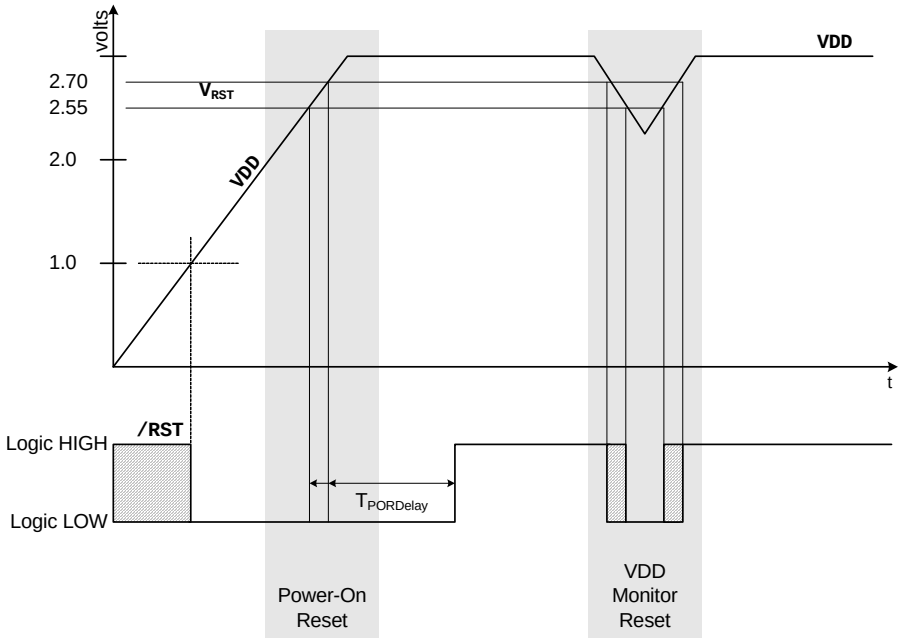



**Figure 17.2. Power-On and $V_{DD}$ Monitor Reset Timing**

## 17.2. Power-Fail Reset / $V_{DD}$ Monitor

When a power-down transition or power irregularity causes $V_{DD}$ to drop below $V_{RST}$, the power supply monitor will drive the $\overline{RST}$ pin low and hold the CIP-51 in a reset state (see Figure 17.2). When $V_{DD}$ returns to a level above $V_{RST}$, the CIP-51 will be released from the reset state. Note that even though internal data memory contents are not altered by the power-fail reset, it is impossible to determine if $V_{DD}$ dropped below the level required for data retention. If the PORSF flag reads '1', the data may no longer be valid. The $V_{DD}$ monitor is enabled after power-on resets. Its defined state (enabled/disabled) is not altered by any other reset source. For example, if the $V_{DD}$ monitor is disabled by code and a software reset is performed, the $V_{DD}$ monitor will still be disabled after the reset.

**Important Note:** If the $V_{DD}$ monitor is being turned on from a disabled state, it should be enabled before it is selected as a reset source. Selecting the $V_{DD}$ monitor as a reset source before it is enabled and stabilized may cause a system reset. In some applications, this reset may be undesirable. If this is not desirable in the application, a delay should be introduced between enabling the monitor and selecting it as a reset source. The procedure for enabling the $V_{DD}$ monitor and configuring it as a reset source from a disabled state is shown below:

1. Enable the $V_{DD}$ monitor (VDMEN bit in VDM0CN = '1').
2. If necessary, wait for the $V_{DD}$ monitor to stabilize.
3. Select the $V_{DD}$ monitor as a reset source (PORSF bit in RSTSRC = '1').

See Figure 17.2 for $V_{DD}$ monitor timing; note that the power-on-reset delay is not incurred after a $V_{DD}$ monitor reset. See **Section "6. Electrical Characteristics" on page 27** for complete electrical characteristics of the $V_{DD}$ monitor.

## SFR Definition 17.1. VDM0CN: $V_{DD}$ Monitor Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | VDMEN | VDDSTAT | | | | | | |
| Type | R/W | R | R | R | R | R | R | R |
| Reset | Varies | Varies | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xFF

| Bit | Name | Function |
|---|---|---|
| 7 | VDMEN | **$V_{DD}$ Monitor Enable.**<br>This bit turns the $V_{DD}$ monitor circuit on/off. The $V_{DD}$ Monitor cannot generate system resets until it is also selected as a reset source in register RSTSRC (SFR Definition 17.2). Selecting the $V_{DD}$ monitor as a reset source before it has stabilized may generate a system reset. In systems where this reset would be undesirable, a delay should be introduced between enabling the $V_{DD}$ Monitor and selecting it as a reset source.<br>0: $V_{DD}$ Monitor Disabled.<br>1: $V_{DD}$ Monitor Enabled. |
| 6 | VDDSTAT | **$V_{DD}$ Status.**<br>This bit indicates the current power supply status ($V_{DD}$ Monitor output).<br>0: $V_{DD}$ is at or below the $V_{DD}$ monitor threshold.<br>1: $V_{DD}$ is above the $V_{DD}$ monitor threshold. |
| 5:0 | UNUSED | Unused. Read = 000000b; Write = Don't care. |

## 17.3. External Reset

The external $\overline{\text{RST}}$ pin provides a means for external circuitry to force the device into a reset state. Asserting an active-low signal on the $\overline{\text{RST}}$ pin generates a reset; an external pullup and/or decoupling of the $\overline{\text{RST}}$ pin may be necessary to avoid erroneous noise-induced resets. See **Section "6. Electrical Characteristics" on page 27** for complete $\overline{\text{RST}}$ pin specifications. The PINRSF flag (RSTSRC.0) is set on exit from an external reset.

## 17.4. Missing Clock Detector Reset

The Missing Clock Detector (MCD) is a one-shot circuit that is triggered by the system clock. If the system clock remains high or low for more than 100 µs, the one-shot will time out and generate a reset. After a MCD reset, the MCDRSF flag (RSTSRC.2) will read '1', signifying the MCD as the reset source; otherwise, this bit reads '0'. Writing a '1' to the MCDRSF bit enables the Missing Clock Detector; writing a '0' disables it. The state of the $\overline{\text{RST}}$ pin is unaffected by this reset.

## 17.5. Comparator0 Reset

Comparator0 can be configured as a reset source by writing a ‘1’ to the C0RSEF flag (RSTSRC.5). Comparator0 should be enabled and allowed to settle prior to writing to C0RSEF to prevent any turn-on chatter on the output from generating an unwanted reset. The Comparator0 reset is active-low: if the non-inverting input voltage (on CP0+) is less than the inverting input voltage (on CP0-), the device is put into the reset state. After a Comparator0 reset, the C0RSEF flag (RSTSRC.5) will read ‘1’ signifying Comparator0 as the reset source; otherwise, this bit reads ‘0’. The state of the RST pin is unaffected by this reset.

## 17.6. PCA Watchdog Timer Reset

The programmable Watchdog Timer (WDT) function of the Programmable Counter Array (PCA) can be used to prevent software from running out of control during a system malfunction. The PCA WDT function can be enabled or disabled by software as described in **Section “25.4. Watchdog Timer Mode” on page 213**; the WDT is enabled and clocked by SYSCLK / 12 following any reset. If a system malfunction prevents user software from updating the WDT, a reset is generated and the WDTRSF bit (RSTSRC.5) is set to ‘1’. The state of the RST pin is unaffected by this reset.

## 17.7. Flash Error Reset

If a Flash read/write/erase or program read targets an illegal address, a system reset is generated. This may occur due to any of the following:

- A Flash write or erase is attempted above user code space. This occurs when PSWE is set to ‘1’ and a MOVX write operation targets an address above address 0x3DFF.
- A Flash read is attempted above user code space. This occurs when a MOVC operation targets an address above address 0x3DFF.
- A Program read is attempted above user code space. This occurs when user code attempts to branch to an address above 0x3DFF.
- A Flash read, write or erase attempt is restricted due to a Flash security setting (see **Section “16.3. Security Options” on page 93**).

The FERROR bit (RSTSRC.6) is set following a Flash error reset. The state of the RST pin is unaffected by this reset.

## 17.8. Software Reset

Software may force a reset by writing a ‘1’ to the SWRSF bit (RSTSRC.4). The SWRSF bit will read ‘1’ following a software forced reset. The state of the RST pin is unaffected by this reset.

## SFR Definition 17.2. RSTSRC: Reset Source

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | FERROR | C0RSEF | SWRSF | WDTRSF | MCDRSF | PORSF | PINRSF |
| Type | R | R | R/W | R/W | R | R/W | R/W | R |
| Reset | 0 | Varies | Varies | Varies | Varies | Varies | Varies | Varies |

SFR Address = 0xEF

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7 | UNUSED | **Unused.** | Don't care. | 0 |
| 6 | FERROR | **Flash Error Reset Flag.** | N/A | Set to '1' if Flash read/write/erase error caused the last reset. |
| 5 | C0RSEF | **Comparator0 Reset Enable and Flag.** | Writing a '1' enables Comparator0 as a reset source (active-low). | Set to '1' if Comparator0 caused the last reset. |
| 4 | SWRSF | **Software Reset Force and Flag.** | Writing a '1' forces a system reset. | Set to '1' if last reset was caused by a write to SWRSF. |
| 3 | WDTRSF | **Watchdog Timer Reset Flag.** | N/A | Set to '1' if Watchdog Timer overflow caused the last reset. |
| 2 | MCDRSF | **Missing Clock Detector Enable and Flag.** | Writing a '1' enables the Missing Clock Detector. The MCD triggers a reset if a missing clock condition is detected. | Set to '1' if Missing Clock Detector timeout caused the last reset. |
| 1 | PORSF | **Power-On / $V_{DD}$ Monitor Reset Flag, and $V_{DD}$ monitor Reset Enable.** | Writing a '1' enables the $V_{DD}$ monitor as a reset source. **Writing '1' to this bit before the $V_{DD}$ monitor is enabled and stabilized may cause a system reset.** | Set to '1' anytime a power-on or $V_{DD}$ monitor reset occurs. **When set to '1' all other RSTSRC flags are indeterminate.** |
| 0 | PINRSF | **HW Pin Reset Flag.** | N/A | Set to '1' if $\overline{\text{RST}}$ pin caused the last reset. |

**Note:** Do not use read-modify-write operations on this register

# 18. Power Management Modes

The C8051F336/7/8/9 devices have three software programmable power management modes: Idle, Stop, and Suspend. Idle mode and Stop mode are part of the standard 8051 architecture, while Suspend mode is an enhanced power-saving mode implemented by the high-speed oscillator peripheral.

Idle mode halts the CPU while leaving the peripherals and clocks active. In Stop mode, the CPU is halted, all interrupts and timers (except the Missing Clock Detector) are inactive, and the internal oscillator is stopped (analog peripherals remain in their selected states; the external oscillator is not affected). Suspend mode is similar to Stop mode in that the internal oscillator and CPU are halted, but the device can wake on events such as a Port Mismatch, Comparator low output, or a Timer 3 overflow. Since clocks are running in Idle mode, power consumption is dependent upon the system clock frequency and the number of peripherals left in active mode before entering Idle. Stop mode and Suspend mode consume the least power because the majority of the device is shut down with no clocks active. SFR Definition 18.1 describes the Power Control Register (PCON) used to control the C8051F336/7/8/9's Stop and Idle power management modes. Suspend mode is controlled by the SUSPEND bit in the OSCICN register (SFR Definition 19.3).

Although the C8051F336/7/8/9 has Idle, Stop, and Suspend modes available, more control over the device power can be achieved by enabling/disabling individual peripherals as needed. Each analog peripheral can be disabled when not in use and placed in low power mode. Digital peripherals, such as timers or serial buses, draw little power when they are not in use. Turning off oscillators lowers power consumption considerably, at the expense of reduced functionality.

## 18.1. Idle Mode

Setting the Idle Mode Select bit (PCON.0) causes the hardware to halt the CPU and enter Idle mode as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data. All analog and digital peripherals can remain active during Idle mode.

Idle mode is terminated when an enabled interrupt is asserted or a reset occurs. The assertion of an enabled interrupt will cause the Idle Mode Selection bit (PCON.0) to be cleared and the CPU to resume operation. The pending interrupt will be serviced and the next instruction to be executed after the return from interrupt (RETI) will be the instruction immediately following the one that set the Idle Mode Select bit. If Idle mode is terminated by an internal or external reset, the CIP-51 performs a normal reset sequence and begins program execution at address 0x0000.

Note: If the instruction following the write of the IDLE bit is a single-byte instruction and an interrupt occurs during the execution phase of the instruction that sets the IDLE bit, the CPU may not wake from Idle mode when a future interrupt occurs. Therefore, instructions that set the IDLE bit should be followed by an instruction that has two or more opcode bytes, for example:

```
// in ‘C’:
PCON |= 0x01;                  // set IDLE bit
PCON = PCON;                   // ... followed by a 3-cycle dummy instruction

; in assembly:
ORL PCON, #01h                 ; set IDLE bit
MOV PCON, PCON                 ; ... followed by a 3-cycle dummy instruction
```

If enabled, the Watchdog Timer (WDT) will eventually cause an internal watchdog reset and thereby terminate the Idle mode. This feature protects the system from an unintended permanent shutdown in the event of an inadvertent write to the PCON register. If this behavior is not desired, the WDT may be disabled by

software prior to entering the Idle mode if the WDT was initially configured to allow this operation. This provides the opportunity for additional power savings, allowing the system to remain in the Idle mode indefinitely, waiting for an external stimulus to wake up the system. Refer to **Section "17.6. PCA Watchdog Timer Reset" on page 104** for more information on the use and configuration of the WDT.

## 18.2. Stop Mode

Setting the Stop Mode Select bit (PCON.1) causes the controller core to enter Stop mode as soon as the instruction that sets the bit completes execution. In Stop mode the internal oscillator, CPU, and all digital peripherals are stopped; the state of the external oscillator circuit is not affected. Each analog peripheral (including the external oscillator circuit) may be shut down individually prior to entering Stop Mode. Stop mode can only be terminated by an internal or external reset. On reset, the device performs the normal reset sequence and begins program execution at address 0x0000.

If enabled, the Missing Clock Detector will cause an internal reset and thereby terminate the Stop mode. The Missing Clock Detector should be disabled if the CPU is to be put to in STOP mode for longer than the MCD timeout of 100 µs.

## 18.3. Suspend Mode

Setting the SUSPEND bit (OSCICN.5) causes the hardware to halt the CPU and the high-frequency internal oscillator, and go into Suspend mode as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data. Most digital peripherals are not active in Suspend mode. The exception to this is the Port Match feature and Timer 3, when it is run from an external oscillator source or the internal low-frequency oscillator.

Suspend mode can be terminated by four types of events, a port match (described in **Section "20.5. Port Match" on page 129**), a Timer 3 overflow (described in **Section "24.3. Timer 3" on page 196**), a Comparator low output (if enabled), or a device reset event. Note that in order to run Timer 3 in Suspend mode, the timer must be configured to clock from either the external clock source or the internal low-frequency oscillator source. When Suspend mode is terminated, the device will continue execution on the instruction following the one that set the SUSPEND bit. If the wake event (port match or Timer 3 overflow) was configured to generate an interrupt, the interrupt will be serviced upon waking the device. If Suspend mode is terminated by an internal or external reset, the CIP-51 performs a normal reset sequence and begins program execution at address 0x0000.

## SFR Definition 18.1. PCON: Power Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | GF[5:0] | | | | | | STOP | IDLE |
| Type | R/W | | | | | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x87

| Bit | Name | Function |
|---|---|---|
| 7:2 | GF[5:0] | **General Purpose Flags 5–0.**<br>These are general purpose flags for use under software control. |
| 1 | STOP | **Stop Mode Select.**<br>Setting this bit will place the CIP-51 in Stop mode. This bit will always be read as 0.<br>1: CPU goes into Stop mode (internal oscillator stopped). |
| 0 | IDLE | **IDLE: Idle Mode Select.**<br>Setting this bit will place the CIP-51 in Idle mode. This bit will always be read as 0.<br>1: CPU goes into Idle mode. (Shuts off clock to CPU, but clock to Timers, Interrupts, Serial Ports, and Analog Peripherals are still active.) |

# 19. Oscillators and Clock Selection

C8051F336/7/8/9 devices include a programmable internal high-frequency oscillator, a programmable internal low-frequency oscillator, and an external oscillator drive circuit. The internal high-frequency oscillator can be enabled/disabled and calibrated using the OSCICN and OSCICL registers, as shown in Figure 19.1. The internal low-frequency oscillator can be enabled/disabled and calibrated using the OSCLCN register. The system clock can be sourced by the external oscillator circuit or either internal oscillator. Both internal oscillators offer a selectable post-scaling feature.

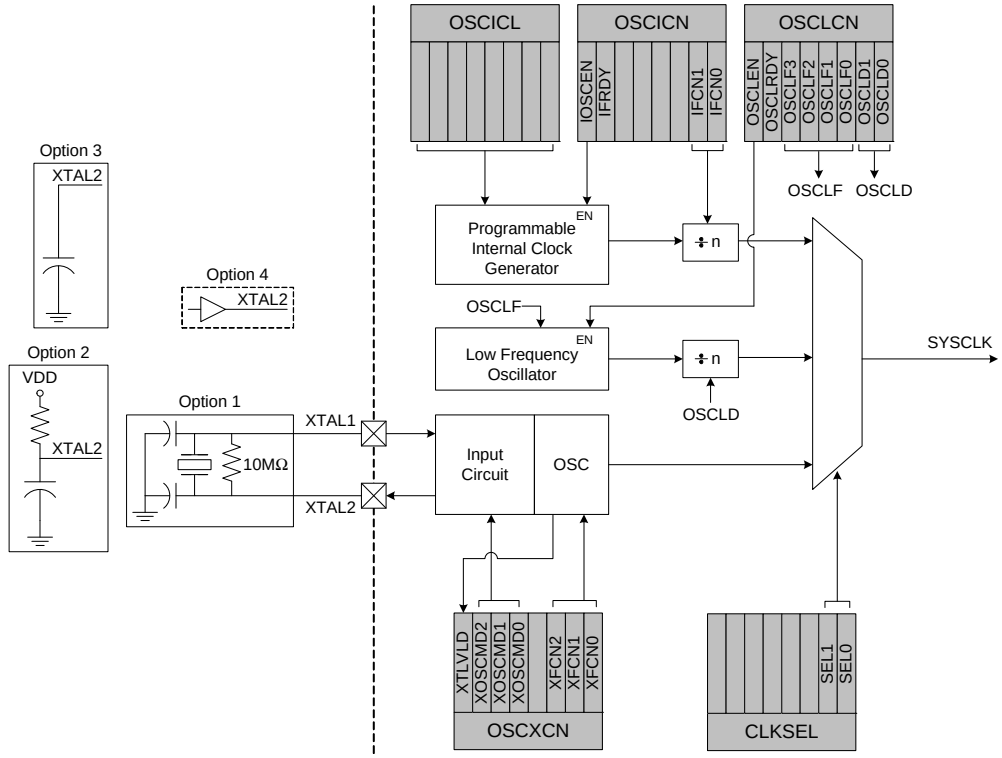



**Figure 19.1. Oscillator Options**

## 19.1. System Clock Selection

The CLKSL[1:0] bits in register CLKSEL select which oscillator source is used as the system clock. CLKSL[1:0] must be set to 01b for the system clock to run from the external oscillator; however the external oscillator may still clock certain peripherals (timers, PCA) when the internal oscillator is selected as the system clock. The system clock may be switched on-the-fly between the internal oscillator, external oscillator, and Clock Multiplier so long as the selected clock source is enabled and has settled.

The internal high-frequency and low-frequency oscillators require little start-up time and may be selected as the system clock immediately following the register write which enables the oscillator. The external RC and C modes also typically require no startup time.

External crystals and ceramic resonators however, typically require a start-up time before they are settled and ready for use. The Crystal Valid Flag (XTLVLD in register OSCXCN) is set to '1' by hardware when the external crystal or ceramic resonator is settled. **In crystal mode, to avoid reading a false XTLVLD, software should delay at least 1 ms between enabling the external oscillator and checking XTLVLD.**

## SFR Definition 19.1. CLKSEL: Clock Select

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | CLKSL[1:0] | |
| Type | R | R | R | R | R | R | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA9

| Bit | Name | Function |
|---|---|---|
| 7:2 | UNUSED | Unused. Read = 000000b; Write = Don't Care |
| 1:0 | CLKSL[1:0] | **System Clock Source Select Bits.**<br>00: SYSCLK derived from the Internal High-Frequency Oscillator and scaled per the IFCN bits in register OSCICN.<br>01: SYSCLK derived from the External Oscillator circuit.<br>10: SYSCLK derived from the Internal Low-Frequency Oscillator and scaled per the OSCLD bits in register OSCLCN.<br>11: reserved. |

## 19.2. Programmable Internal High-Frequency (H-F) Oscillator

All C8051F336/7/8/9 devices include a programmable internal high-frequency oscillator that defaults as the system clock after a system reset. The internal oscillator period caPara1n be adjusted via the OSCICL register as defined by SFR Definition 19.2.

On C8051F336/7/8/9 devices, OSCICL is factory calibrated to obtain a 24.5 MHz base frequency.

The system clock may be derived from the programmed internal oscillator divided by 1, 2, 4, or 8, as defined by the IFCN bits in register OSCICN. The divide value defaults to 8 following a reset.

### 19.2.1. Internal Oscillator Suspend Mode

When software writes a logic 1 to SUSPEND (OSCICN.5), the internal oscillator is suspended. If the system clock is derived from the internal oscillator, the input clock to the peripheral or CIP-51 will be stopped until one of the following events occur:

- Port 0 Match Event.
- Port 1 Match Event.
- Comparator 0 enabled and output is logic 0.
- Timer3 Overflow Event.

When one of the oscillator awakening events occur, the internal oscillator, CIP-51, and affected peripherals resume normal operation, regardless of whether the event also causes an interrupt. The CPU resumes execution at the instruction following the write to SUSPEND.

## SFR Definition 19.2. OSCICL: Internal H-F Oscillator Calibration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | OSCICL[6:0] | | | | | | |
| Type | R | R/W | | | | | | |
| Reset | 0 | Varies | Varies | Varies | Varies | Varies | Varies | Varies |

SFR Address = 0xB3

| Bit | Name | Function |
|---|---|---|
| 7 | Unused | Unused. Read = 0; Write = Don't Care |
| 6:0 | OSCICL[6:0] | **Internal Oscillator Calibration Bits.**<br>These bits determine the internal oscillator period. When set to 0000000b, the H-F oscillator operates at its fastest setting. When set to 1111111b, the H-F oscillator operates at its slowest setting. The reset value is factory calibrated to generate an internal oscillator frequency of 24.5 MHz. |

## SFR Definition 19.3. OSCICN: Internal H-F Oscillator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IOSCEN | IFRDY | SUSPEND | STSYNC | | | IFCN[1:0] | |
| Type | R/W | R | R/W | R | R | R | R/W | |
| Reset | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xB2

| Bit | Name | Function |
|---|---|---|
| 7 | IOSCEN | **Internal H-F Oscillator Enable Bit.**<br>0: Internal H-F Oscillator Disabled.<br>1: Internal H-F Oscillator Enabled. |
| 6 | IFRDY | **Internal H-F Oscillator Frequency Ready Flag.**<br>0: Internal H-F Oscillator is not running at programmed frequency.<br>1: Internal H-F Oscillator is running at programmed frequency. |
| 5 | SUSPEND | **Internal Oscillator Suspend Enable Bit.**<br>Setting this bit to logic 1 places the internal oscillator in SUSPEND mode. The internal oscillator resumes operation when one of the SUSPEND mode awakening events occurs. |
| 4 | STSYNC | **Suspend Timer Synchronization Bit.**<br>This bit is used to indicate when it is safe to read and write the registers associated with the suspend wake-up timer. If a suspend wake-up source other than the timer has brought the oscillator out of suspend mode, it may take up to three timer clocks before the timer can be read or written. When STSYNC reads '1', reads and writes of the timer register should not be performed. When STSYNC reads '0', it is safe to read and write the timer registers. |
| 3:2 | Unused | Unused. Read = 00b; Write = Don't Care |
| 1:0 | IFCN[1:0] | **Internal H-F Oscillator Frequency Divider Control Bits.**<br>00: SYSCLK derived from Internal H-F Oscillator divided by 8.<br>01: SYSCLK derived from Internal H-F Oscillator divided by 4.<br>10: SYSCLK derived from Internal H-F Oscillator divided by 2.<br>11: SYSCLK derived from Internal H-F Oscillator divided by 1. |

## 19.3. Programmable Internal Low-Frequency (L-F) Oscillator

All C8051F336/7/8/9 devices include a programmable low-frequency internal oscillator, which is calibrated to a nominal frequency of 80 kHz. The low-frequency oscillator circuit includes a divider that can be changed to divide the clock by 1, 2, 4, or 8, using the OSCLD bits in the OSCLCN register (see SFR Definition 19.4). Additionally, the OSCLF[3:0] bits can be used to adjust the oscillator's output frequency.

### 19.3.1. Calibrating the Internal L-F Oscillator

Timers 2 and 3 include capture functions that can be used to capture the oscillator frequency, when running from a known time base. When either Timer 2 or Timer 3 is configured for L-F Oscillator Capture Mode, a falling edge (Timer 2) or rising edge (Timer 3) of the low-frequency oscillator's output will cause a capture event on the corresponding timer. As a capture event occurs, the current timer value (TMRnH:TMRnL) is copied into the timer reload registers (TMRnRLH:TMRnRLL). By recording the difference between two successive timer capture values, the low-frequency oscillator's period can be calculated. The OSCLF bits can then be adjusted to produce the desired oscillator frequency.

## SFR Definition 19.4. OSCLCN: Internal L-F Oscillator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OSCLEN | OSCLRDY | OSCLF[3:0] | | | | OSCLD[1:0] | |
| Type | R/W | R | R.W | | | | R/W | |
| Reset | 0 | 0 | Varies | Varies | Varies | Varies | 0 | 0 |

SFR Address = 0xE3

| Bit | Name | Function |
|---|---|---|
| 7 | OSCLEN | **Internal L-F Oscillator Enable.**<br>0: Internal L-F Oscillator Disabled.<br>1: Internal L-F Oscillator Enabled. |
| 6 | OSCLRDY | **Internal L-F Oscillator Ready.**<br>0: Internal L-F Oscillator frequency not stabilized.<br>1: Internal L-F Oscillator frequency stabilized.<br>**Note:** OSCLRDY is only set back to 0 in the event of a device reset or a change to the OSCLD[1:0] bits. |
| 5:2 | OSCLF[3:0] | **Internal L-F Oscillator Frequency Control Bits.**<br>Fine-tune control bits for the Internal L-F oscillator frequency. When set to 0000b, the L-F oscillator operates at its fastest setting. When set to 1111b, the L-F oscillator operates at its slowest setting. |
| 1:0 | OSCLD[1:0] | **Internal L-F Oscillator Divider Select.**<br>00: Divide by 8 selected.<br>01: Divide by 4 selected.<br>10: Divide by 2 selected.<br>11: Divide by 1 selected. |

## 19.4. External Oscillator Drive Circuit

The external oscillator circuit may drive an external crystal, ceramic resonator, capacitor, or RC network. A CMOS clock may also provide a clock input. For a crystal or ceramic resonator configuration, the crystal/resonator must be wired across the XTAL1 and XTAL2 pins as shown in Option 1 of Figure 19.1. A 10 MΩ resistor also must be wired across the XTAL2 and XTAL1 pins for the crystal/resonator configuration. In RC, capacitor, or CMOS clock configuration, the clock source should be wired to the XTAL2 pin as shown in Option 2, 3, or 4 of Figure 19.1. The type of external oscillator must be selected in the OSCXCN register, and the frequency control bits (XFCN) must be selected appropriately (see SFR Definition 19.5).

**Important Note on External Oscillator Usage:** Port pins must be configured when using the external oscillator circuit. When the external oscillator drive circuit is enabled in crystal/resonator mode, Port pins P0.2 and P0.3 are used as XTAL1 and XTAL2 respectively. When the external oscillator drive circuit is enabled in capacitor, RC, or CMOS clock mode, Port pin P0.3 is used as XTAL2. The Port I/O Crossbar should be configured to skip the Port pins used by the oscillator circuit; see **Section "20.3. Priority Crossbar Decoder" on page 124** for Crossbar configuration. Additionally, when using the external oscillator circuit in crystal/resonator, capacitor, or RC mode, the associated Port pins should be configured as **analog inputs**. In CMOS clock mode, the associated pin should be configured as a **digital input**. See **Section "20.4. Port I/O Initialization" on page 126** for details on Port input mode selection.

## SFR Definition 19.5. OSCXCN: External Oscillator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | XTLVLD | XOSCMD[2:0] | | | | XFCN[2:0] | | |
| Type | R | R/W | | | R | R/W | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xB1

| Bit | Name | Function |
|---|---|---|
| 7 | XTLVLD | **Crystal Oscillator Valid Flag.**<br>(Read only when XOSCMD = 11x.)<br>0: Crystal Oscillator is unused or not yet stable.<br>1: Crystal Oscillator is running and stable. |
| 6:4 | XOSCMD[2:0] | **External Oscillator Mode Select.**<br>00x: External Oscillator circuit off.<br>010: External CMOS Clock Mode.<br>011: External CMOS Clock Mode with divide by 2 stage.<br>100: RC Oscillator Mode.<br>101: Capacitor Oscillator Mode.<br>110: Crystal Oscillator Mode.<br>111: Crystal Oscillator Mode with divide by 2 stage. |
| 3 | UNUSED | Read = 0; Write = Don't Care |
| 2:0 | XFCN[2:0] | **External Oscillator Frequency Control Bits.**<br>Set according to the desired frequency for Crystal or RC mode.<br>Set according to the desired K Factor for C mode. |

| XFCN | Crystal Mode | RC Mode | C Mode |
|---|---|---|---|
| 000 | f ≤ 32 kHz | f ≤ 25 kHz | K Factor = 0.87 |
| 001 | 32 kHz < f ≤ 84 kHz | 25 kHz < f ≤ 50 kHz | K Factor = 2.6 |
| 010 | 84 kHz < f ≤ 225 kHz | 50 kHz < f ≤ 100 kHz | K Factor = 7.7 |
| 011 | 225 kHz < f ≤ 590 kHz | 100 kHz < f ≤ 200 kHz | K Factor = 22 |
| 100 | 590 kHz < f ≤ 1.5 MHz | 200 kHz < f ≤ 400 kHz | K Factor = 65 |
| 101 | 1.5 MHz < f ≤ 4 MHz | 400 kHz < f ≤ 800 kHz | K Factor = 180 |
| 110 | 4 MHz < f ≤ 10 MHz | 800 kHz < f ≤ 1.6 MHz | K Factor = 664 |
| 111 | 10 MHz < f ≤ 30 MHz | 1.6 MHz < f ≤ 3.2 MHz | K Factor = 1590 |

### 19.4.1. External Crystal Example

If a crystal or ceramic resonator is used as an external oscillator source for the MCU, the circuit should be configured as shown in Figure 19.1, Option 1. The External Oscillator Frequency Control value (XFCN) should be chosen from the Crystal column of the table in SFR Definition 19.5 (OSCXCN register). For example, an 11.0592 MHz crystal requires an XFCN setting of 111b and a 32.768 kHz Watch Crystal requires an XFCN setting of 001b. After an external 32.768 kHz oscillator is stabilized, the XFCN setting can be switched to 000 to save power. It is recommended to enable the missing clock detector before switching the system clock to any external oscillator source.

When the crystal oscillator is first enabled, the oscillator amplitude detection circuit requires a settling time to achieve proper bias. Introducing a delay of 1 ms between enabling the oscillator and checking the XTLVLD bit will prevent a premature switch to the external oscillator as the system clock. Switching to the external oscillator before the crystal oscillator has stabilized can result in unpredictable behavior. The recommended procedure is:

1. Force XTAL1 and XTAL2 to a low state. This involves enabling the Crossbar and writing ‘0’ to the port pins associated with XTAL1 and XTAL2.
2. Configure XTAL1 and XTAL2 as analog inputs using.
3. Enable the external oscillator.
4. Wait at least 1 ms.
5. Poll for XTLVLD => ‘1’.
6. Enable the Missing Clock Detector.
7. Switch the system clock to the external oscillator.

**Important Note on External Crystals:** Crystal oscillator circuits are quite sensitive to PCB layout. The crystal should be placed as close as possible to the XTAL pins on the device. The traces should be as short as possible and shielded with ground plane from any other traces which could introduce noise or interference.

The capacitors shown in the external crystal configuration provide the load capacitance required by the crystal for correct oscillation. These capacitors are "in series" as seen by the crystal and "in parallel" with the stray capacitance of the XTAL1 and XTAL2 pins.

**Note:** The desired load capacitance depends upon the crystal and the manufacturer. Please refer to the crystal data sheet when completing these calculations.

For example, a tuning-fork crystal of 32.768 kHz with a recommended load capacitance of 12.5 pF should use the configuration shown in Figure 19.1, Option 1. The total value of the capacitors and the stray capacitance of the XTAL pins should equal 25 pF. With a stray capacitance of 3 pF per pin, the 22 pF capacitors yield an equivalent capacitance of 12.5 pF across the crystal, as shown in Figure 19.2.

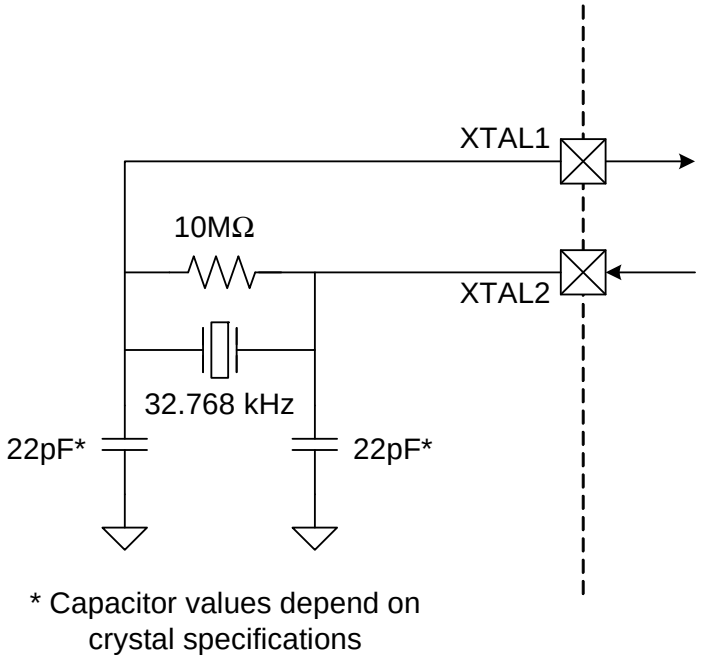



**Figure 19.2. External 32.768 kHz Quartz Crystal Oscillator Connection Diagram**

### 19.4.2. External RC Example

If an RC network is used as an external oscillator source for the MCU, the circuit should be configured as shown in Figure 19.1, Option 2. The capacitor should be no greater than 100 pF; however for very small capacitors, the total capacitance may be dominated by parasitic capacitance in the PCB layout. To determine the required External Oscillator Frequency Control value (XFCN) in the OSCXCN Register, first select the RC network value to produce the desired frequency of oscillation, according to Equation 19.1, where f = the frequency of oscillation in MHz, C = the capacitor value in pF, and R = the pull-up resistor value in kΩ.

**Equation 19.1. RC Mode Oscillator Frequency**

$$f = 1.23 \times 10^3 / (R \times C)$$

For example: If the frequency desired is 100 kHz, let R = 246 kΩ and C = 50 pF:

f = 1.23( $10^3$ ) / RC = 1.23 ( $10^3$ ) / [ 246 x 50 ] = 0.1 MHz = 100 kHz

Referring to the table in SFR Definition 19.5, the required XFCN setting is 010b.

### 19.4.3. External Capacitor Example

If a capacitor is used as an external oscillator for the MCU, the circuit should be configured as shown in Figure 19.1, Option 3. The capacitor should be no greater than 100 pF; however for very small capacitors, the total capacitance may be dominated by parasitic capacitance in the PCB layout. To determine the required External Oscillator Frequency Control value (XFCN) in the OSCXCN Register, select the capacitor to be used and find the frequency of oscillation according to Equation 19.2, where f = the frequency of oscillation in MHz, C = the capacitor value in pF, and $V_{DD}$ = the MCU power supply in Volts.

**Equation 19.2. C Mode Oscillator Frequency**

$$f = (KF) / (R \times V_{DD})$$

For example: Assume $V_{DD}$ = 3.0 V and f = 150 kHz:

f = KF / (C x VDD)
0.150 MHz = KF / (C x 3.0)

Since the frequency of roughly 150 kHz is desired, select the K Factor from the table in SFR Definition 19.5 (OSCXCN) as KF = 22:

0.150 MHz = 22 / (C x 3.0)
C x 3.0 = 22 / 0.150 MHz
C = 146.6 / 3.0 pF = 48.8 pF

Therefore, the XFCN value to use in this example is 011b and C = 50 pF.

## 20. Port Input/Output

Digital and analog resources are available through 17 (C8051F336/7) or 21 (C8051F338/9) I/O pins. Port pins P0.0-P2.3 can be defined as general-purpose I/O (GPIO), assigned to one of the internal digital resources, or assigned to an analog function as shown in Figure 20.4. Port pin P2.4 on the C8051F338/9 and P2.0 on the C8051F336/7 can be used as GPIO and are shared with the C2 Interface Data signal (C2D). The designer has complete control over which functions are assigned, limited only by the number of physical I/O pins. This resource assignment flexibility is achieved through the use of a Priority Crossbar Decoder. Note that the state of a Port I/O pin can always be read in the corresponding Port latch, regardless of the Crossbar settings.

The Crossbar assigns the selected internal digital resources to the I/O pins based on the Priority Decoder (Figure 20.4 and Figure 20.5). The registers XBR0 and XBR1, defined in SFR Definition 20.1 and SFR Definition 20.2, are used to select internal digital functions.

All Port I/Os are 5 V tolerant (refer to Figure 20.2 for the Port cell circuit). The Port I/O cells are configured as either push-pull or open-drain in the Port Output Mode registers (PnMDOUT, where n = 0,1). Complete Electrical Specifications for Port I/O are given in **Section "6. Electrical Characteristics" on page 27**.

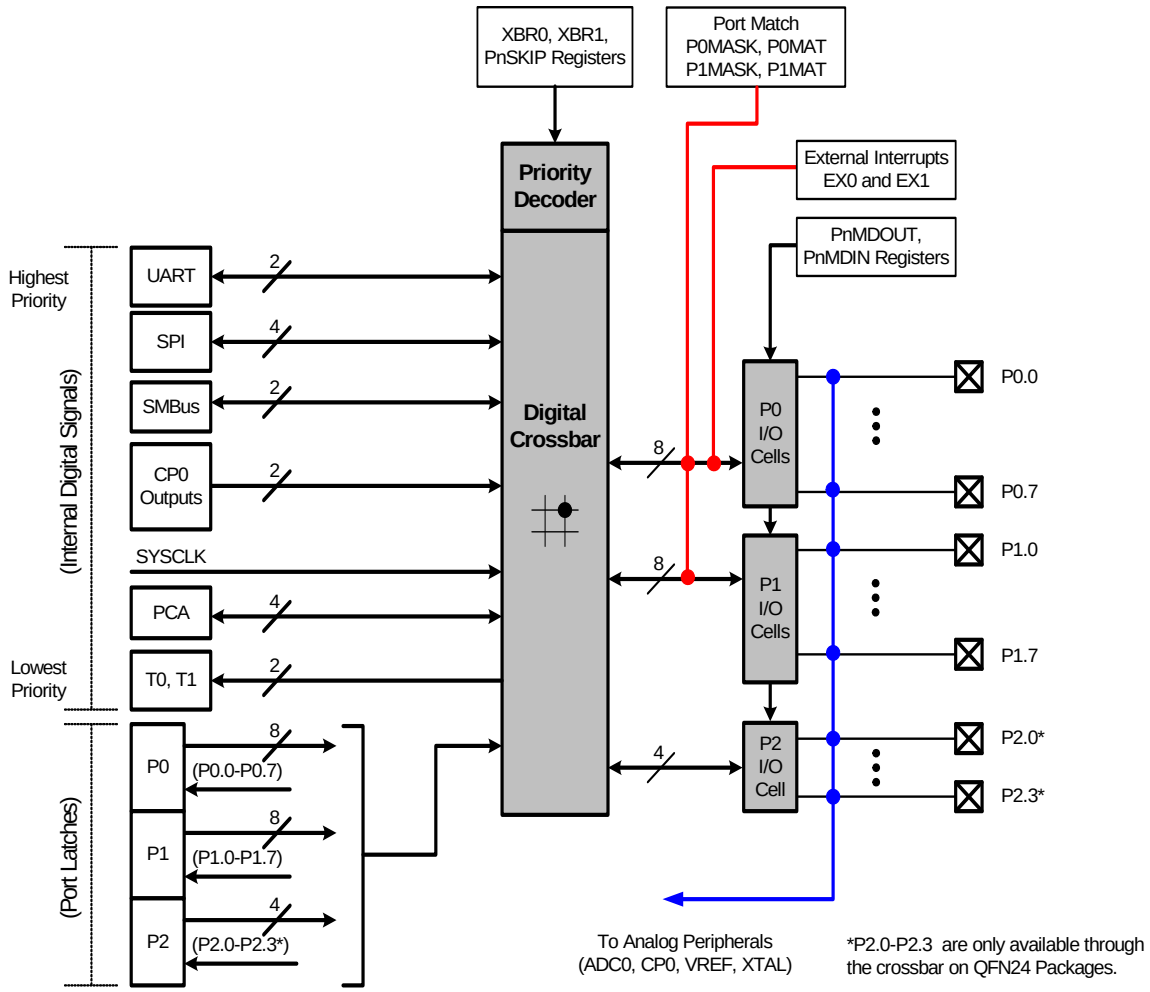



**Figure 20.1. Port I/O Functional Block Diagram**

## 20.1. Port I/O Modes of Operation

Port pins P0.0 - P2.3 use the Port I/O cell shown in Figure 20.2. Each Port I/O cell can be configured by software for analog I/O or digital I/O using the PnMDIN registers. On reset, all Port I/O cells default to a high impedance state with weak pull-ups enabled. Until the crossbar is enabled (XBARE = ‘1’), both the high and low port I/O drive circuits are explicitly disabled on all crossbar pins.

### 20.1.1. Port Pins Configured for Analog I/O

Any pins to be used as Comparator or ADC input, external oscillator input/output, VREF, or IDAC output should be configured for analog I/O (PnMDIN.n = ‘1’). When a pin is configured for analog I/O, its weak pullup, digital driver, and digital receiver are disabled. Port pins configured for analog I/O will always read back a value of ‘0’.

Configuring pins as analog I/O saves power and isolates the Port pin from digital interference. Port pins configured as digital I/O may still be used by analog peripherals; however, this practice is not recommended and may result in measurement errors.

### 20.1.2. Port Pins Configured For Digital I/O

Any pins to be used by digital peripherals (UART, SPI, SMBus, etc.), external event trigger functions, or as GPIO should be configured as digital I/O (PnMDIN.n = ‘1’). For digital I/O pins, one of two output modes (push-pull or open-drain) must be selected using the PnMDOUT registers.

Push-pull outputs (PnMDOUT.n = ‘1’) drive the Port pad to the VDD or GND supply rails based on the output logic value of the Port pin. Open-drain outputs have the high side driver disabled; therefore, they only drive the Port pad to GND when the output logic value is ‘0’ and become high impedance inputs (both high low drivers turned off) when the output logic value is ‘1’.

When a digital I/O cell is placed in the high impedance state, a weak pull-up transistor pulls the Port pad to the VDD supply voltage to ensure the digital input is at a defined logic state. Weak pull-ups are disabled when the I/O cell is driven to GND to minimize power consumption, and they may be globally disabled by setting WEAKPUD to ‘1’. The user should ensure that digital I/O are always internally or externally pulled or driven to a valid logic state to minimize power consumption. Port pins configured for digital I/O always read back the logic state of the Port pad, regardless of the output logic value of the Port pin.

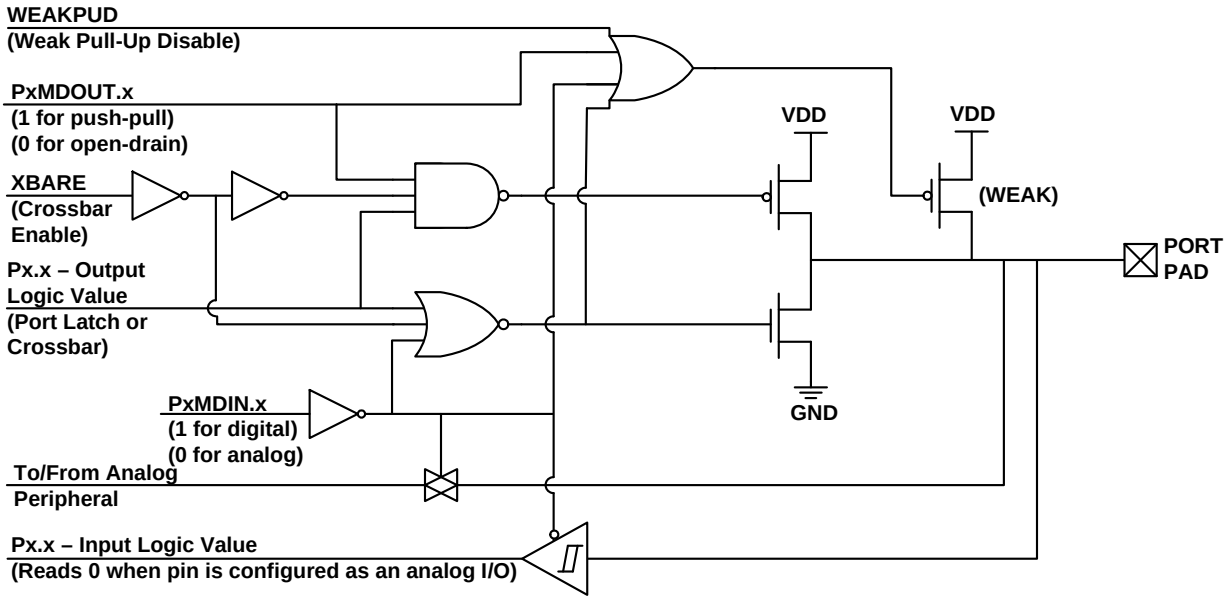
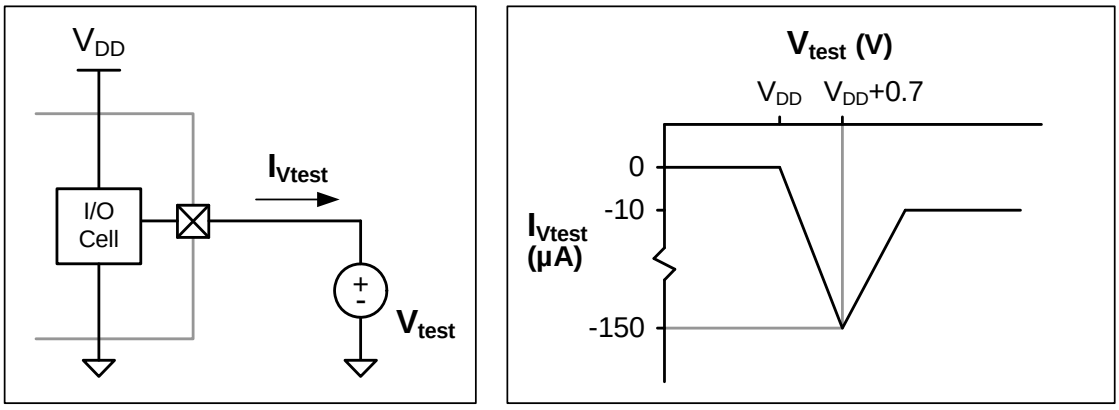



**Figure 20.2. Port I/O Cell Block Diagram**

### 20.1.3. Interfacing Port I/O to 5V Logic

All Port I/O configured for digital, open-drain operation are capable of interfacing to digital logic operating at a supply voltage higher than VDD and less than 5.25V. An external pull-up resistor to the higher supply voltage is typically required for most systems.

**Important Note:** In a multi-voltage interface, the external pull-up resistor should be sized to allow a current of at least 150uA to flow into the Port pin when the supply voltage is between (VDD + 0.6V) and (VDD + 1.0V). Once the Port pin voltage increases beyond this range, the current flowing into the Port pin is minimal. Figure 20.3 shows the input current characteristics of port pins driven above VDD. The port pin requires 150 µA peak overdrive current when its voltage reaches approximately (VDD + 0.7 V).




**Port I/O Overdrive Test Circuit** **Port I/O Overdrive Current vs. Voltage**

**Figure 20.3. Port I/O Overdrive Current**

## 20.2. Assigning Port I/O Pins to Analog and Digital Functions

Port I/O pins P0.0 - P2.3 can be assigned to various analog, digital, and external interrupt functions. The Port pins assigned to analog functions should be configured for analog I/O, and Port pins assigned to digital or external interrupt functions should be configured for digital I/O.

### 20.2.1. Assigning Port I/O Pins to Analog Functions

Table 20.1 shows all available analog functions that require Port I/O assignments. **Port pins selected for these analog functions should have their corresponding bit in PnSKIP set to '1'.** This reserves the pin for use by the analog function and does not allow it to be claimed by the Crossbar. Table 20.1 shows the potential mapping of Port I/O to each analog function.

**Table 20.1. Port I/O Assignment for Analog Functions**

| Analog Function | Potentially Assignable Port Pins | SFR(s) used for Assignment |
|---|---|---|
| ADC Input | P0.0 - P2.3 | AMX0P, AMX0N, PnSKIP, PnMDIN |
| Comparator0 Input | P0.0 - P2.3 | CPT0MX, PnSKIP, PnMDIN |
| Voltage Reference (VREF0) | P0.0 | REF0CN, PnSKIP, PnMDIN |
| Current DAC Output (IDA0) | P0.1 | IDA0CN, PnSKIP, PnMDIN |
| External Oscillator in Crystal Mode (XTAL1) | P0.2 | OSCXCN, PnSKIP, PnMDIN |
| External Oscillator in RC, C, or Crystal Mode (XTAL2) | P0.3 | OSCXCN, PnSKIP, PnMDIN |

### 20.2.2. Assigning Port I/O Pins to Digital Functions

Any Port pins not assigned to analog functions may be assigned to digital functions or used as GPIO. Most digital functions rely on the Crossbar for pin assignment; however, some digital functions bypass the Crossbar in a manner similar to the analog functions listed above. **Port pins used by these digital functions and any Port pins selected for use as GPIO should have their corresponding bit in PnSKIP set to '1'.** Table 20.2 shows all available digital functions and the potential mapping of Port I/O to each digital function.

**Table 20.2. Port I/O Assignment for Digital Functions**

| Digital Function | Potentially Assignable Port Pins | SFR(s) used for Assignment |
|---|---|---|
| UART0, SPI0, SMBus, CP0, CP0A, SYSCLK, PCA0 (CEX0-2 and ECI), T0 or T1. | Any Port pin available for assignment by the Crossbar. This includes P0.0 - P2.3 pins which have their PnSKIP bit set to '0'.<br>**Note:** The Crossbar will always assign UART0 pins to P0.4 and P0.5. | XBR0, XBR1 |
| Any pin used for GPIO | P0.0 - P2.4 | P0SKIP, P1SKIP, P2SKIP |

### 20.2.3. Assigning Port I/O Pins to External Event Trigger Functions

External event trigger functions can be used to trigger an interrupt or wake the device from a low power mode when a transition occurs on a digital I/O pin. The event trigger functions do not require dedicated pins and will function on both GPIO pins (PnSKIP = ’1’) and pins in use by the Crossbar (PnSKIP = ‘0’). External event trigger functions cannot be used on pins configured for analog I/O. Table 20.3 shows all available external event trigger functions.

**Table 20.3. Port I/O Assignment for External Event Trigger Functions**

| Event Trigger Function | Potentially Assignable Port Pins | SFR(s) used for Assignment |
|---|---|---|
| External Interrupt 0 | P0.0 - P0.7 | IT01CF |
| External Interrupt 1 | P0.0 - P0.7 | IT01CF |
| Port Match | P0.0 - P1.7 | P0MASK, P0MAT<br>P1MASK, P1MAT |

## 20.3. Priority Crossbar Decoder

The Priority Crossbar Decoder (Figure 20.4) assigns a priority to each I/O function, starting at the top with UART0. When a digital resource is selected, the least-significant unassigned Port pin is assigned to that resource (excluding UART0, which is always at pins 4 and 5). If a Port pin is assigned, the Crossbar skips that pin when assigning the next selected resource. Additionally, the Crossbar will skip Port pins whose associated bits in the PnSKIP registers are set. The PnSKIP registers allow software to skip Port pins that are to be used for analog input, dedicated functions, or GPIO.

**Important Note on Crossbar Configuration:** If a Port pin is claimed by a peripheral without use of the Crossbar, its corresponding PnSKIP bit should be set. This applies to P0.0 if VREF is used, P0.3 and/or P0.2 if the external oscillator circuit is enabled, P0.6 if the ADC or IDAC is configured to use the external conversion start signal (CNVSTR), and any selected ADC or Comparator inputs. The Crossbar skips selected pins as if they were already assigned, and moves to the next unassigned pin.

Figure 20.4 shows all of the potential peripheral-to-pin assignments available to the crossbar. Note that this does not mean any peripheral can always be assigned to the highlighted pins. The actual pin assignments are determined by the priority of the enabled peripherals.

| | P0 | | | | | | | | P1 | | | | | | | | P2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SF Signals | VREF | IDA | x1 | x2 | | | | CNVSTR | | | | | | | | | | | | | |
| PIN I/O | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1[2] | 2[2] | 3[2] | 4[2] |
| TX0 | | | | | ■ | | | | | | | | | | | | | | | | Pin not available for crossbar peripherals. |
| RX0 | | | | | | ■ | | | | | | | | | | | | | | | |
| SCK | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | | |
| MISO | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | |
| MOSI | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| NSS[1] | | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| SDA | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | | |
| SCL | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| CP0 | ■ | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| CP0A | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| SYSCLK | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| CEX0 | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| CEX1 | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| CEX2 | | | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| ECI | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| T0 | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |
| T1 | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | ■ | |

■ Port pin potentially available to peripheral

SF Signals — Special Function Signals are not assigned by the crossbar. When these signals are enabled, the Crossbar must be manually configured to skip their corresponding port pins.

Notes:
1. NSS is only pinned out in 4-wire SPI Mode
2. Pins P2.1-P2.4 only on QFN24 Package
3. Pin 2.0 unavailable on crossbar in QFN20 Package

**Figure 20.4. Crossbar Priority Decoder - Possible Pin Assignments**

Registers XBR0 and XBR1 are used to assign the digital I/O resources to the physical I/O Port pins. Note that when the SMBus is selected, the Crossbar assigns both pins associated with the SMBus (SDA and SCL); when the UART is selected, the Crossbar assigns both pins associated with the UART (TX and RX). UART0 pin assignments are fixed for bootloading purposes: UART TX0 is always assigned to P0.4; UART RX0 is always assigned to P0.5. Standard Port I/Os appear contiguously after the prioritized functions have been assigned.

Figure 20.5 shows an example of the resulting pin assignments of the device with UART0, SMBus, and CEX0 enabled, the XTAL1 (P0.2) and XTAL2 (P0.3) pins skipped (P0SKIP = 0x0C). UART0 is the highest priority and it will be assigned first. The UART can only appear on P0.4 and P0.5, so that is where it is assigned. The next-highest enabled peripheral is the SMBus. P0.0 and P0.1 are free, so the SMBus takes these two pins. The last peripheral enabled is the PCA's CEX0 pin. P0.0, P0.1, P0.4 and P0.5 are already occupied by higher-priority peripherals. Additionally, P0.2 and P0.3 are set to be skipped by the crossbar. The CEX0 signal ends up getting routed to P0.6, as it is the next available pin. The other pins on the device are available for use as general-purpose digital I/O or analog functions.

| | P0 | | | | | | | | P1 | | | | | | | | P2 | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| SF Signals | VREF | IDA | x1 | x2 | | | | CNVSTR | | | | | | | | | | | | | |
| PIN I/O | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1[2] | 2[2] | 3[2] | 4[2] |
| TX0 | | | | | ■ | | | | | | | | | | | | | | | | |
| RX0 | | | | | | ■ | | | | | | | | | | | | | | | |
| SCK | | | | | | | | | | | | | | | | | | | | | |
| MISO | | | | | | | | | | | | | | | | | | | | | |
| MOSI | | | | | | | | | | | | | | | | | | | | | |
| NSS[1] | | | | | | | | | | | | | | | | | | | | | |
| SDA | ■ | | | | | | | | | | | | | | | | | | | | |
| SCL | | ■ | | | | | | | | | | | | | | | | | | | |
| CP0 | | | | | | | | | | | | | | | | | | | | | |
| CP0A | | | | | | | | | | | | | | | | | | | | | |
| SYSCLK | | | | | | | | | | | | | | | | | | | | | |
| CEX0 | | | | | | | ■ | | | | | | | | | | | | | | |
| CEX1 | | | | | | | | | | | | | | | | | | | | | |
| CEX2 | | | | | | | | | | | | | | | | | | | | | |
| ECI | | | | | | | | | | | | | | | | | | | | | |
| T0 | | | | | | | | | | | | | | | | | | | | | |
| T1 | | | | | | | | | | | | | | | | | | | | | |
| | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | P0SKIP[0:7] | | | | | | | | P1SKIP[0:7] | | | | | | | | P2SKIP[0:3] | | | | |

Pin not available for crossbar peripherals.

■ Port pin potentially available to peripheral

SF Signals — Special Function Signals are not assigned by the crossbar. When these signals are enabled, the CrossBar must be manually configured to skip their corresponding port pins.

Notes:
1. NSS is only pinned out in 4-wire SPI Mode
2. Pins P2.1-P2.4 only on QFN24 Package
3. Pin 2.0 unavailable on crossbar in QFN20 Package

**Figure 20.5. Crossbar Priority Decoder Example**

**Important Note:** The SPI can be operated in either 3-wire or 4-wire modes, pending the state of the NSSMD1–NSSMD0 bits in register SPI0CN. According to the SPI mode, the NSS signal may or may not be routed to a Port pin.

## 20.4. Port I/O Initialization

Port I/O initialization consists of the following steps:

1. Select the input mode (analog or digital) for all Port pins, using the Port Input Mode register (PnMDIN).
2. Select the output mode (open-drain or push-pull) for all Port pins, using the Port Output Mode register (PnMDOUT).
3. Select any pins to be skipped by the I/O Crossbar using the Port Skip registers (PnSKIP).
4. Assign Port pins to desired peripherals.
5. Enable the Crossbar (XBARE = ‘1’).

All Port pins must be configured as either analog or digital inputs. Any pins to be used as Comparator or ADC inputs should be configured as an analog inputs. When a pin is configured as an analog input, its weak pullup, digital driver, and digital receiver are disabled. This process saves power and reduces noise on the analog input. Pins configured as digital inputs may still be used by analog peripherals; however this practice is not recommended.

Additionally, all analog input pins should be configured to be skipped by the Crossbar (accomplished by setting the associated bits in PnSKIP). Port input mode is set in the PnMDIN register, where a ‘1’ indicates a digital input, and a ‘0’ indicates an analog input. All pins default to digital inputs on reset. See SFR Definition 20.8 for the PnMDIN register details.

The output driver characteristics of the I/O pins are defined using the Port Output Mode registers (PnMDOUT). Each Port Output driver can be configured as either open drain or push-pull. This selection is required even for the digital resources selected in the XBRn registers, and is not automatic. The only exception to this is the SMBus (SDA, SCL) pins, which are configured as open-drain regardless of the PnMDOUT settings. When the WEAKPUD bit in XBR1 is ‘0’, a weak pullup is enabled for all Port I/O configured as open-drain. WEAKPUD does not affect the push-pull Port I/O. Furthermore, the weak pullup is turned off on an output that is driving a ‘0’ to avoid unnecessary power dissipation.

Registers XBR0 and XBR1 must be loaded with the appropriate values to select the digital I/O functions required by the design. Setting the XBARE bit in XBR1 to ‘1’ enables the Crossbar. Until the Crossbar is enabled, the external pins remain as standard Port I/O (in input mode), regardless of the XBRn Register settings. For given XBRn Register settings, one can determine the I/O pin-out using the Priority Decode Table; as an alternative, the Configuration Wizard utility of the Silicon Labs IDE software will determine the Port I/O pin-assignments based on the XBRn Register settings.

The Crossbar must be enabled to use Port pins as standard Port I/O in output mode. Port output drivers are disabled while the Crossbar is disabled.

## SFR Definition 20.1. XBR0: Port I/O Crossbar Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | CP0AE | CP0E | SYSCKE | SMB0E | SPI0E | URT0E |
| Type | R | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xE1

| Bit | Name | Function |
|---|---|---|
| 7:6 | UNUSED | Unused. Read = 00b; Write = Don't Care. |
| 5 | CP0AE | **Comparator0 Asynchronous Output Enable.**<br>0: Asynchronous CP0 unavailable at Port pin.<br>1: Asynchronous CP0 routed to Port pin. |
| 4 | CP0E | **Comparator0 Output Enable.**<br>0: CP0 unavailable at Port pin.<br>1: CP0 routed to Port pin. |
| 3 | SYSCKE | **/SYSCLK Output Enable.**<br>0: /SYSCLK unavailable at Port pin.<br>1: /SYSCLK output routed to Port pin. |
| 2 | SMB0E | **SMBus I/O Enable.**<br>0: SMBus I/O unavailable at Port pins.<br>1: SMBus I/O routed to Port pins. |
| 1 | SPI0E | **SPI I/O Enable.**<br>0: SPI I/O unavailable at Port pins.<br>1: SPI I/O routed to Port pins. Note that the SPI can be assigned either 3 or 4 GPIO pins. |
| 0 | URT0E | **UART I/O Output Enable.**<br>0: UART I/O unavailable at Port pin.<br>1: UART TX0, RX0 routed to Port pins P0.4 and P0.5. |

## SFR Definition 20.2. XBR1: Port I/O Crossbar Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | WEAKPUD | XBARE | T1E | T0E | ECIE | | PCA0ME[1:0] | |
| Type | R/W | R/W | R/W | R/W | R/W | R | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xE2

| Bit | Name | Function |
|---|---|---|
| 7 | WEAKPUD | **Port I/O Weak Pullup Disable.**<br>0: Weak Pullups enabled (except for Ports whose I/O are configured for analog mode).<br>1: Weak Pullups disabled. |
| 6 | XBARE | **Crossbar Enable.**<br>0: Crossbar disabled.<br>1: Crossbar enabled. |
| 5 | T1E | **T1 Enable.**<br>0: T1 unavailable at Port pin.<br>1: T1 routed to Port pin. |
| 4 | T0E | **T0 Enable.**<br>0: T0 unavailable at Port pin.<br>1: T0 routed to Port pin. |
| 3 | ECIE | **PCA0 External Counter Input Enable.**<br>0: ECI unavailable at Port pin.<br>1: ECI routed to Port pin. |
| 2 | UNUSED | Unused. Read = 0b; Write = Don't Care. |
| 1:0 | PCA0ME[1:0] | **PCA Module I/O Enable Bits.**<br>00: All PCA I/O unavailable at Port pins.<br>01: CEX0 routed to Port pin.<br>10: CEX0, CEX1 routed to Port pins.<br>11: CEX0, CEX1, CEX2 routed to Port pins. |

## 20.5. Port Match

Port match functionality allows system events to be triggered by a logic value change on P0 or P1. A software controlled value stored in the PnMATCH registers specifies the expected or normal logic values of P0 and P1. A Port mismatch event occurs if the logic levels of the Port's input pins no longer match the software controlled value. This allows Software to be notified if a certain change or pattern occurs on P0 or P1 input pins regardless of the XBRn settings.

The PnMASK registers can be used to individually select which P0 and P1 pins should be compared against the PnMATCH registers. A Port mismatch event is generated if (P0 & P0MASK) does not equal (P0MATCH & P0MASK) or if (P1 & P1MASK) does not equal (P1MATCH & P1MASK).

A Port mismatch event may be used to generate an interrupt or wake the device from a low power mode, such as IDLE or SUSPEND. See the Interrupts and Power Options chapters for more details on interrupt and wake-up sources.

### SFR Definition 20.3. P0MASK: Port 0 Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0MASK[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xFE

| Bit | Name | Function |
|---|---|---|
| 7:0 | P0MASK[7:0] | **Port 0 Mask Value.**<br>Selects P0 pins to be compared to the corresponding bits in P0MAT.<br>0: P0.n pin logic value is ignored and cannot cause a Port Mismatch event.<br>1: P0.n pin logic value is compared to P0MAT.n. |

## SFR Definition 20.4. P0MAT: Port 0 Match Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0MAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xFD

| Bit | Name | Function |
|---|---|---|
| 7:0 | P0MAT[7:0] | **Port 0 Match Value.**<br>Match comparison value used on Port 0 for bits in P0MASK which are set to '1'.<br>0: P0.n pin logic value is compared with logic LOW.<br>1: P0.n pin logic value is compared with logic HIGH. |

## SFR Definition 20.5. P1MASK: Port 1 Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P1MASK[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xEE

| Bit | Name | Function |
|---|---|---|
| 7:0 | P1MASK[7:0] | **Port 1 Mask Value.**<br>Selects P1 pins to be compared to the corresponding bits in P1MAT.<br>0: P1.n pin logic value is ignored and cannot cause a Port Mismatch event.<br>1: P1.n pin logic value is compared to P1MAT.n. |

## SFR Definition 20.6. P1MAT: Port 1 Match Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P1MAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xED

| Bit | Name | Function |
|---|---|---|
| 7:0 | P1MAT[7:0] | **Port 1 Match Value.**<br>Match comparison value used on Port 1 for bits in P1MASK which are set to ‘1’.<br>0: P1.n pin logic value is compared with logic LOW.<br>1: P1.n pin logic value is compared with logic HIGH. |

## 20.6. Special Function Registers for Accessing and Configuring Port I/O

All Port I/O are accessed through corresponding special function registers (SFRs) that are both byte addressable and bit addressable. When writing to a Port, the value written to the SFR is latched to maintain the output data value at each pin. When reading, the logic levels of the Port's input pins are returned regardless of the XBRn settings (i.e., even when the pin is assigned to another signal by the Crossbar, the Port register can always read its corresponding Port I/O pin). The exception to this is the execution of the read-modify-write instructions that target a Port Latch register as the destination. The read-modify-write instructions when operating on a Port SFR are the following: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ and MOV, CLR or SETB, when the destination is an individual bit in a Port SFR. For these instructions, the value of the latch register (not the pin) is read, modified, and written back to the SFR.

Each Port has a corresponding PnSKIP register which allows its individual Port pins to be assigned to digital functions or skipped by the Crossbar. All Port pins used for analog functions, GPIO, or dedicated digital functions such as the EMIF should have their PnSKIP bit set to ‘1’.

The Port input mode of the I/O pins is defined using the Port Input Mode registers (PnMDIN). Each Port cell can be configured for analog or digital I/O. This selection is required even for the digital resources selected in the XBRn registers, and is not automatic. The only exception to this is P2.4, which can only be used for digital I/O.

The output driver characteristics of the I/O pins are defined using the Port Output Mode registers (PnMDOUT). Each Port Output driver can be configured as either open drain or push-pull. This selection is required even for the digital resources selected in the XBRn registers, and is not automatic. The only exception to this is the SMBus (SDA, SCL) pins, which are configured as open-drain regardless of the PnMDOUT settings.

## SFR Definition 20.7. P0: Port 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0x80; Bit Addressable

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:0 | P0[7:0] | **Port 0 Data.**<br>Sets the Port latch logic value or reads the Port pin logic state in Port cells configured for digital I/O. | 0: Set output latch to logic LOW.<br>1: Set output latch to logic HIGH. | 0: P0.n Port pin is logic LOW.<br>1: P0.n Port pin is logic HIGH. |

## SFR Definition 20.8. P0MDIN: Port 0 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0MDIN[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xF1

| Bit | Name | Function |
|---|---|---|
| 7:0 | P0MDIN[7:0] | **Analog Configuration Bits for P0.7–P0.0 (respectively).**<br>Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled.<br>0: Corresponding P0.n pin is configured for analog mode.<br>1: Corresponding P0.n pin is not configured for analog mode. |

## SFR Definition 20.9. P0MDOUT: Port 0 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0MDOUT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA4

| Bit | Name | Function |
|---|---|---|
| 7:0 | P0MDOUT[7:0] | **Output Configuration Bits for P0.7–P0.0 (respectively).**<br>These bits are ignored if the corresponding bit in register P0MDIN is logic 0.<br>0: Corresponding P0.n Output is open-drain.<br>1: Corresponding P0.n Output is push-pull. |

## SFR Definition 20.10. P0SKIP: Port 0 Skip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0SKIP[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD4

| Bit | Name | Function |
|---|---|---|
| 7:0 | P0SKIP[7:0] | **Port 0 Crossbar Skip Enable Bits.**<br>These bits select Port 0 pins to be skipped by the Crossbar Decoder. Port pins used for analog, special functions or GPIO should be skipped by the Crossbar.<br>0: Corresponding P0.n pin is not skipped by the Crossbar.<br>1: Corresponding P0.n pin is skipped by the Crossbar. |

## SFR Definition 20.11. P1: Port 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P1[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0x90; Bit Addressable

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:0 | P1[7:0] | **Port 1 Data.**<br>Sets the Port latch logic value or reads the Port pin logic state in Port cells configured for digital I/O. | 0: Set output latch to logic LOW.<br>1: Set output latch to logic HIGH. | 0: P1.n Port pin is logic LOW.<br>1: P1.n Port pin is logic HIGH. |

## SFR Definition 20.12. P1MDIN: Port 1 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P1MDIN[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xF2

| Bit | Name | Function |
|---|---|---|
| 7:0 | P1MDIN[7:0] | **Analog Configuration Bits for P1.7–P1.0 (respectively).**<br>Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled.<br>0: Corresponding P1.n pin is configured for analog mode.<br>1: Corresponding P1.n pin is not configured for analog mode. |

## SFR Definition 20.13. P1MDOUT: Port 1 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P1MDOUT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA5

| Bit | Name | Function |
|---|---|---|
| 7:0 | P1MDOUT[7:0] | **Output Configuration Bits for P1.7–P1.0 (respectively).**<br>These bits are ignored if the corresponding bit in register P1MDIN is logic 0.<br>0: Corresponding P1.n Output is open-drain.<br>1: Corresponding P1.n Output is push-pull. |

## SFR Definition 20.14. P1SKIP: Port 1 Skip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P1SKIP[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD5

| Bit | Name | Function |
|---|---|---|
| 7:0 | P1SKIP[7:0] | **Port 1 Crossbar Skip Enable Bits.**<br>These bits select Port 1 pins to be skipped by the Crossbar Decoder. Port pins used for analog, special functions or GPIO should be skipped by the Crossbar.<br>0: Corresponding P1.n pin is not skipped by the Crossbar.<br>1: Corresponding P1.n pin is skipped by the Crossbar. |

## SFR Definition 20.15. P2: Port 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | P2[4:0] | | | | |
| Type | R | R | R | R/W | | | | |
| Reset | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xA0; Bit Addressable

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:5 | UNUSED | **Unused.** | Don't Care | 000b |
| 4:0 | P2[4:0] | **Port 2 Data.**<br>Sets the Port latch logic value or reads the Port pin logic state in Port cells configured for digital I/O. | 0: Set output latch to logic LOW.<br>1: Set output latch to logic HIGH. | 0: P2.n Port pin is logic LOW.<br>1: P2.n Port pin is logic HIGH. |
| **Note:** Pins P2.1-P2.4 are only available in QFN24-packaged devices. | | | | |

## SFR Definition 20.16. P2MDIN: Port 2 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | P2MDIN[7:0] | | | |
| Type | R | R | R | R | R/W | | | |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |

SFR Address = 0xF3

| Bit | Name | Function |
|---|---|---|
| 7:4 | UNUSED | Unused. Read = 0000b; Write = Don't Care |
| 3:0 | P2MDIN[3:0] | **Analog Configuration Bits for P2.3–P2.0 (respectively).**<br>Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled.<br>0: Corresponding P2.n pin is configured for analog mode.<br>1: Corresponding P2.n pin is not configured for analog mode. |
| **Note:** Pins P2.1-P2.4 are only available in QFN24-packaged devices. | | |

## SFR Definition 20.17. P2MDOUT: Port 2 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | P2MDOUT[4:0] | | | | |
| Type | R | R | R | R/W | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA6

| Bit | Name | Function |
|---|---|---|
| 7:5 | UNUSED | Unused. Read = 000b; Write = Don't Care |
| 4:0 | P2MDOUT[4:0] | **Output Configuration Bits for P2.4–P2.0 (respectively).**<br>These bits are ignored if the corresponding bit in register P2MDIN is logic 0.<br>0: Corresponding P2.n Output is open-drain.<br>1: Corresponding P2.n Output is push-pull. |

**Note:** P2.0 is not available for analog input in the QFN20-packaged devices, and P2.1-P2.4 are only available in the QFN24-packaged devices.

## SFR Definition 20.18. P2SKIP: Port 2 Skip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | P2SKIP[7:0] | | | |
| Type | R | R | R | R | R/W | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD6

| Bit | Name | Function |
|---|---|---|
| 7:4 | UNUSED | Unused. Read = 0000b; Write = Don't Care |
| 3:0 | P2SKIP[3:0] | **Port 2 Crossbar Skip Enable Bits.**<br>These bits select Port 2 pins to be skipped by the Crossbar Decoder. Port pins used for analog, special functions or GPIO should be skipped by the Crossbar.<br>0: Corresponding P2.n pin is not skipped by the Crossbar.<br>1: Corresponding P2.n pin is skipped by the Crossbar. |

**Note:** P2.0 is not available for crossbar peripherals in the QFN20-packaged devices, and P2.1-P2.4 are only available in the QFN24-packaged devices.

## 21. SMBus

The SMBus I/O interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the $I^2C$ serial bus. Reads and writes to the interface by the system controller are byte oriented with the SMBus interface autonomously controlling the serial transfer of the data. Data can be transferred at up to 1/20th of the system clock as a master or slave (this can be faster than allowed by the SMBus specification, depending on the system clock used). A method of extending the clock-low duration is available to accommodate devices with different speed capabilities on the same bus.

The SMBus interface may operate as a master and/or slave, and may function on a bus with multiple masters. The SMBus provides control of SDA (serial data), SCL (serial clock) generation and synchronization, arbitration logic, and START/STOP control and generation. The SMBus peripheral can be fully driven by software (i.e., software accepts/rejects slave addresses, and generates ACKs), or hardware slave address recognition and automatic ACK generation can be enabled to minimize software overhead. A block diagram of the SMBus peripheral and the associated SFRs is shown in Figure 21.1.

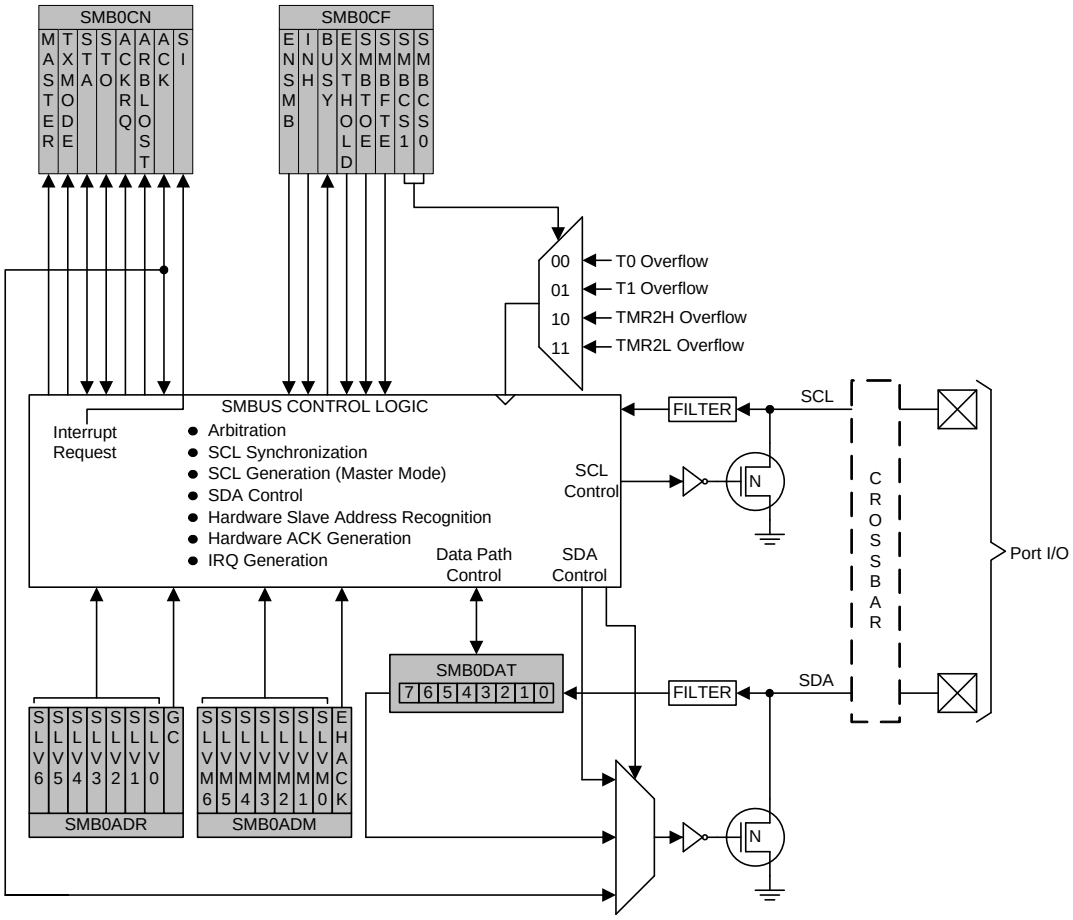



**Figure 21.1. SMBus Block Diagram**

## 21.1. Supporting Documents

It is assumed the reader is familiar with or has access to the following supporting documents:

1. The I2C-Bus and How to Use It (including specifications), Philips Semiconductor.
2. The I2C-Bus Specification—Version 2.0, Philips Semiconductor.
3. System Management Bus Specification—Version 1.1, SBS Implementers Forum.

## 21.2. SMBus Configuration

Figure 21.2 shows a typical SMBus configuration. The SMBus specification allows any recessive voltage between 3.0 V and 5.0 V; different devices on the bus may operate at different voltage levels. The bi-directional SCL (serial clock) and SDA (serial data) lines must be connected to a positive power supply voltage through a pullup resistor or similar circuit. Every device connected to the bus must have an open-drain or open-collector output for both the SCL and SDA lines, so that both are pulled high (recessive state) when the bus is free. The maximum number of devices on the bus is limited only by the requirement that the rise and fall times on the bus not exceed 300 ns and 1000 ns, respectively.

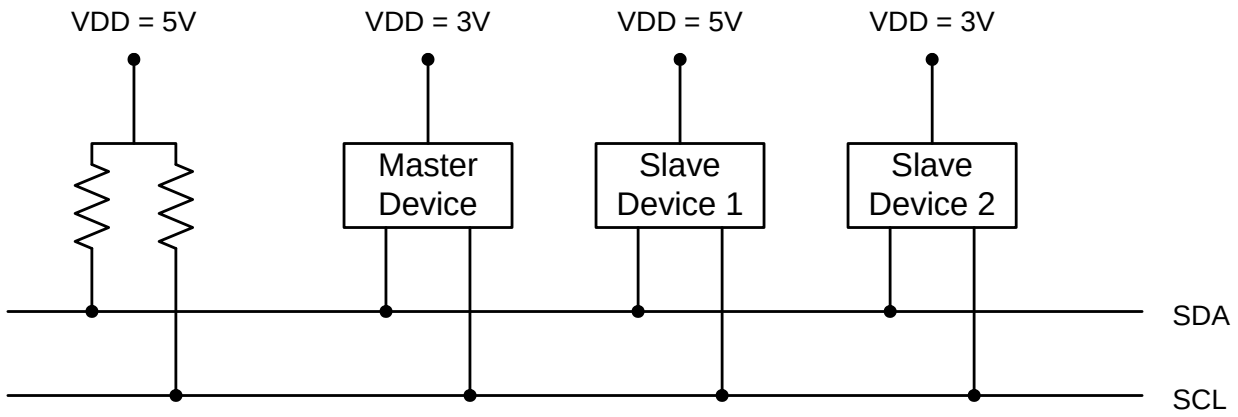



**Figure 21.2. Typical SMBus Configuration**

## 21.3. SMBus Operation

Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver (WRITE), and data transfers from an addressed slave transmitter to a master receiver (READ). The master device initiates both types of data transfers and provides the serial clock pulses on SCL. The SMBus interface may operate as a master or a slave, and multiple master devices on the same bus are supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration. Note that it is not necessary to specify one device as the Master in a system; any device who transmits a START and a slave address becomes the master for the duration of that transfer.

A typical SMBus transaction consists of a START condition followed by an address byte (Bits7–1: 7-bit slave address; Bit0: R/W direction bit), one or more bytes of data, and a STOP condition. Bytes that are received (by a master or slave) are acknowledged (ACK) with a low SDA during a high SCL (see Figure 21.3). If the receiving device does not ACK, the transmitting device will read a NACK (not acknowledge), which is a high SDA during a high SCL.

The direction bit (R/W) occupies the least-significant bit position of the address byte. The direction bit is set to logic 1 to indicate a "READ" operation and cleared to logic 0 to indicate a "WRITE" operation.

All transactions are initiated by a master, with one or more addressed slave devices as the target. The master generates the START condition and then transmits the slave address and direction bit. If the transaction is a WRITE operation from the master to the slave, the master transmits the data a byte at a time waiting for an ACK from the slave at the end of each byte. For READ operations, the slave transmits the data waiting for an ACK from the master at the end of each byte. At the end of the data transfer, the master generates a STOP condition to terminate the transaction and free the bus. Figure 21.3 illustrates a typical SMBus transaction.

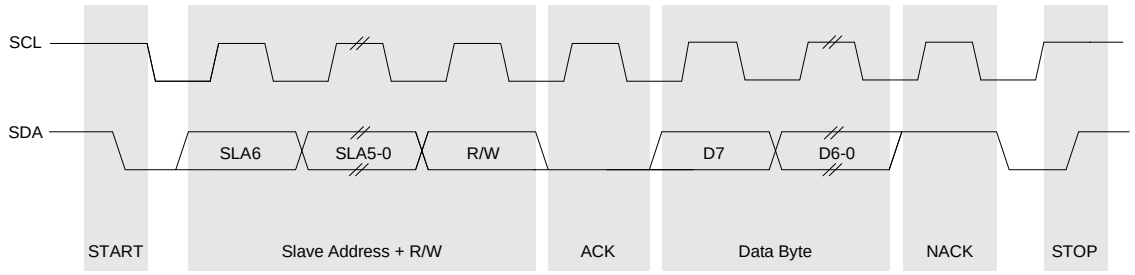



**Figure 21.3. SMBus Transaction**

### 21.3.1. Transmitter Vs. Receiver

On the SMBus communications interface, a device is the “transmitter” when it is sending an address or data byte to another device on the bus. A device is a “receiver” when an address or data byte is being sent to it from another device on the bus. The transmitter controls the SDA line during the address or data byte. After each byte of address or data information is sent by the transmitter, the receiver sends an ACK or NACK bit during the ACK phase of the transfer, during which time the receiver controls the SDA line.

### 21.3.2. Arbitration

A master may start a transfer only if the bus is free. The bus is free after a STOP condition or after the SCL and SDA lines remain high for a specified time (see Section “21.3.5. SCL High (SMBus Free) Timeout” on page 141). In the event that two or more devices attempt to begin a transfer at the same time, an arbitration scheme is employed to force one master to give up the bus. The master devices continue transmitting until one attempts a HIGH while the other transmits a LOW. Since the bus is open-drain, the bus will be pulled LOW. The master attempting the HIGH will detect a LOW SDA and lose the arbitration. The winning master continues its transmission without interruption; the losing master becomes a slave and receives the rest of the transfer if addressed. This arbitration scheme is non-destructive: one device always wins, and no data is lost.

### 21.3.3. Clock Low Extension

SMBus provides a clock synchronization mechanism, similar to I2C, which allows devices with different speed capabilities to coexist on the bus. A clock-low extension is used during a transfer in order to allow slower slave devices to communicate with faster masters. The slave may temporarily hold the SCL line LOW to extend the clock low period, effectively decreasing the serial clock frequency.

### 21.3.4. SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than 25 ms as a “timeout” condition. Devices that have detected the timeout condition must reset the communication no later than 10 ms after detecting the timeout condition.

When the SMBTOE bit in SMB0CF is set, Timer 3 is used to detect SCL low timeouts. Timer 3 is forced to reload when SCL is high, and allowed to count when SCL is low. With Timer 3 enabled and configured to overflow after 25 ms (and SMBTOE set), the Timer 3 interrupt service routine can be used to reset (disable and re-enable) the SMBus in the event of an SCL low timeout.

#### 21.3.5. SCL High (SMBus Free) Timeout

The SMBus specification stipulates that if the SCL and SDA lines remain high for more that 50 µs, the bus is designated as free. When the SMBFTE bit in SMB0CF is set, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods (as defined by the timer configured for the SMBus clock source). If the SMBus is waiting to generate a Master START, the START will be generated following this timeout. A clock source is required for free timeout detection, even in a slave-only implementation.

## 21.4. Using the SMBus

The SMBus can operate in both Master and Slave modes. The interface provides timing and shifting control for serial transfers; higher level protocol is determined by user software. The SMBus interface provides the following application-independent features:

- Byte-wise serial data transfers
- Clock signal generation on SCL (Master Mode only) and SDA data synchronization
- Timeout/bus error recognition, as defined by the SMB0CF configuration register
- START/STOP timing, detection, and generation
- Bus arbitration
- Interrupt generation
- Status information
- Optional hardware recognition of slave address and automatic acknowledgement of address/data

SMBus interrupts are generated for each data byte or slave address that is transferred. When hardware acknowledgement is disabled, the point at which the interrupt is generated depends on whether the hardware is acting as a data transmitter or receiver. When a transmitter (i.e., sending address/data, receiving an ACK), this interrupt is generated after the ACK cycle so that software may read the received ACK value; when receiving data (i.e., receiving address/data, sending an ACK), this interrupt is generated before the ACK cycle so that software may define the outgoing ACK value. If hardware acknowledgement is enabled, these interrupts are always generated after the ACK cycle. See Section 21.5 for more details on transmission sequences.

Interrupts are also generated to indicate the beginning of a transfer when a master (START generated), or the end of a transfer when a slave (STOP detected). Software should read the SMB0CN (SMBus Control register) to find the cause of the SMBus interrupt. The SMB0CN register is described in Section 21.4.2; Table 21.5 provides a quick SMB0CN decoding reference.

#### 21.4.1. SMBus Configuration Register

The SMBus Configuration register (SMB0CF) is used to enable the SMBus Master and/or Slave modes, select the SMBus clock source, and select the SMBus timing and timeout options. When the ENSMB bit is set, the SMBus is enabled for all master and slave events. Slave events may be disabled by setting the INH bit. With slave events inhibited, the SMBus interface will still monitor the SCL and SDA pins; however, the interface will NACK all received addresses and will not generate any slave interrupts. When the INH bit is set, all slave events will be inhibited following the next START (interrupts will continue for the duration of the current transfer).

**Table 21.1. SMBus Clock Source Selection**

| SMBCS1 | SMBCS0 | SMBus Clock Source |
|---|---|---|
| 0 | 0 | Timer 0 Overflow |
| 0 | 1 | Timer 1 Overflow |
| 1 | 0 | Timer 2 High Byte Overflow |
| 1 | 1 | Timer 2 Low Byte Overflow |

The SMBCS1–0 bits select the SMBus clock source, which is used only when operating as a master or when the Free Timeout detection is enabled. When operating as a master, overflows from the selected source determine the absolute minimum SCL low and high times as defined in Equation 21.1. Note that the selected clock source may be shared by other peripherals so long as the timer is left running at all times. For example, Timer 1 overflows may generate the SMBus and UART baud rates simultaneously. Timer configuration is covered in Section “24. Timers” on page 180.

$$T_{HighMin} = T_{LowMin} = \frac{1}{f_{ClockSourceOverflow}}$$

**Equation 21.1. Minimum SCL High and Low Times**

The selected clock source should be configured to establish the minimum SCL High and Low times as per Equation 21.1. When the interface is operating as a master (and SCL is not driven or extended by any other devices on the bus), the typical SMBus bit rate is approximated by Equation 21.2.

$$BitRate = \frac{f_{ClockSourceOverflow}}{3}$$

**Equation 21.2. Typical SMBus Bit Rate**

Figure 21.4 shows the typical SCL generation described by Equation 21.2. Notice that $T_{HIGH}$ is typically twice as large as $T_{LOW}$. The actual SCL output may vary due to other devices on the bus (SCL may be extended low by slower slave devices, or driven low by contending master devices). The bit rate when operating as a master will never exceed the limits defined by equation Equation 21.1.

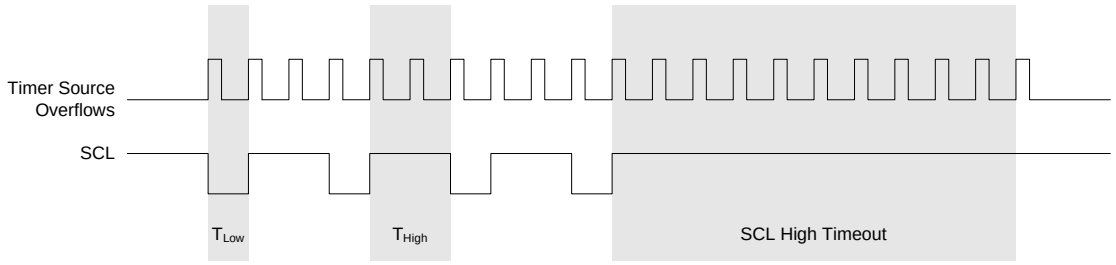



**Figure 21.4. Typical SMBus SCL Generation**

Setting the EXTHOLD bit extends the minimum setup and hold times for the SDA line. The minimum SDA setup time defines the absolute minimum time that SDA is stable before SCL transitions from low-to-high. The minimum SDA hold time defines the absolute minimum time that the current SDA value remains stable

after SCL transitions from high-to-low. EXTHOLD should be set so that the minimum setup and hold times meet the SMBus Specification requirements of 250 ns and 300 ns, respectively. Table 21.2 shows the minimum setup and hold times for the two EXTHOLD settings. Setup and hold time extensions are typically necessary when SYSCLK is above 10 MHz.

**Table 21.2. Minimum SDA Setup and Hold Times**

| EXTHOLD | Minimum SDA Setup Time | Minimum SDA Hold Time |
|---|---|---|
| 0 | $T_{low}$ – 4 system clocks<br>or<br>1 system clock + s/w delay* | 3 system clocks |
| 1 | 11 system clocks | 12 system clocks |
| **Note:** Setup Time for ACK bit transmissions and the MSB of all data transfers. When using software acknowledgement, the s/w delay occurs between the time SMB0DAT or ACK is written and when SI is cleared. Note that if SI is cleared in the same write that defines the outgoing ACK value, s/w delay is zero. | | |

With the SMBTOE bit set, Timer 3 should be configured to overflow after 25 ms in order to detect SCL low timeouts (see Section “21.3.4. SCL Low Timeout” on page 140). The SMBus interface will force Timer 3 to reload while SCL is high, and allow Timer 3 to count when SCL is low. The Timer 3 interrupt service routine should be used to reset SMBus communication by disabling and re-enabling the SMBus.

SMBus Free Timeout detection can be enabled by setting the SMBFTE bit. When this bit is set, the bus will be considered free if SDA and SCL remain high for more than 10 SMBus clock source periods (see Figure 21.4).

## SFR Definition 21.1. SMB0CF: SMBus Clock/Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ENSMB | INH | BUSY | EXTHOLD | SMBTOE | SMBFTE | SMBCS[1:0] | |
| Type | R/W | R/W | R | R/W | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC1

| Bit | Name | Function |
|---|---|---|
| 7 | ENSMB | **SMBus Enable.**<br>This bit enables the SMBus interface when set to 1. When enabled, the interface constantly monitors the SDA and SCL pins. |
| 6 | INH | **SMBus Slave Inhibit.**<br>When this bit is set to logic 1, the SMBus does not generate an interrupt when slave events occur. This effectively removes the SMBus slave from the bus. Master Mode interrupts are not affected. |
| 5 | BUSY | **SMBus Busy Indicator.**<br>This bit is set to logic 1 by hardware when a transfer is in progress. It is cleared to logic 0 when a STOP or free-timeout is sensed. |
| 4 | EXTHOLD | **SMBus Setup and Hold Time Extension Enable.**<br>This bit controls the SDA setup and hold times according to Table 21.2.<br>0: SDA Extended Setup and Hold Times disabled.<br>1: SDA Extended Setup and Hold Times enabled. |
| 3 | SMBTOE | **SMBus SCL Timeout Detection Enable.**<br>This bit enables SCL low timeout detection. If set to logic 1, the SMBus forces Timer 3 to reload while SCL is high and allows Timer 3 to count when SCL goes low. If Timer 3 is configured to Split Mode, only the High Byte of the timer is held in reload while SCL is high. Timer 3 should be programmed to generate interrupts at 25 ms, and the Timer 3 interrupt service routine should reset SMBus communication. |
| 2 | SMBFTE | **SMBus Free Timeout Detection Enable.**<br>When this bit is set to logic 1, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods. |
| 1:0 | SMBCS[1:0] | **SMBus Clock Source Selection.**<br>These two bits select the SMBus clock source, which is used to generate the SMBus bit rate. The selected device should be configured according to Equation 21.1.<br>00: Timer 0 Overflow<br>01: Timer 1 Overflow<br>10: Timer 2 High Byte Overflow<br>11: Timer 2 Low Byte Overflow |

### 21.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see SFR Definition 21.2). The higher four bits of SMB0CN (MASTER, TXMODE, STA, and STO) form a status vector that can be used to jump to service routines. MASTER indicates whether a device is the master or slave during the current transfer. TXMODE indicates whether the device is transmitting or receiving data for the current byte.

STA and STO indicate that a START and/or STOP has been detected or generated since the last SMBus interrupt. STA and STO are also used to generate START and STOP conditions when operating as a master. Writing a 1 to STA will cause the SMBus interface to enter Master Mode and generate a START when the bus becomes free (STA is not cleared by hardware after the START is generated). Writing a 1 to STO while in Master Mode will cause the interface to generate a STOP and end the current transfer after the next ACK cycle. If STO and STA are both set (while in Master Mode), a STOP followed by a START will be generated.

The ARBLOST bit indicates that the interface has lost an arbitration. This may occur anytime the interface is transmitting (master or slave). A lost arbitration while operating as a slave indicates a bus error condition. ARBLOST is cleared by hardware each time SI is cleared.

The SI bit (SMBus Interrupt Flag) is set at the beginning and end of each transfer, after each byte frame, or when an arbitration is lost; see Table 21.3 for more details.

**Important Note About the SI Bit:** The SMBus interface is stalled while SI is set; thus SCL is held low, and the bus is stalled until software clears SI.

#### 21.4.2.1. Software ACK Generation

When the EHACK bit in register SMB0ADM is cleared to 0, the firmware on the device must detect incoming slave addresses and ACK or NACK the slave address and incoming data bytes. As a receiver, writing the ACK bit defines the outgoing ACK value; as a transmitter, reading the ACK bit indicates the value received during the last ACK cycle. ACKRQ is set each time a byte is received, indicating that an outgoing ACK value is needed. When ACKRQ is set, software should write the desired outgoing value to the ACK bit before clearing SI. A NACK will be generated if software does not write the ACK bit before clearing SI. SDA will reflect the defined ACK value immediately following a write to the ACK bit; however SCL will remain low until SI is cleared. If a received slave address is not acknowledged, further slave events will be ignored until the next START is detected.

#### 21.4.2.2. Hardware ACK Generation

When the EHACK bit in register SMB0ADM is set to 1, automatic slave address recognition and ACK generation is enabled. More detail about automatic slave address recognition can be found in Section 21.4.3. As a receiver, the value currently specified by the ACK bit will be automatically sent on the bus during the ACK cycle of an incoming data byte. As a transmitter, reading the ACK bit indicates the value received on the last ACK cycle. The ACKRQ bit is not used when hardware ACK generation is enabled. If a received slave address is NACKed by hardware, further slave events will be ignored until the next START is detected, and no interrupt will be generated.

Table 21.3 lists all sources for hardware changes to the SMB0CN bits. Refer to Table 21.5 for SMBus status decoding using the SMB0CN register.

## SFR Definition 21.2. SMB0CN: SMBus Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MASTER | TXMODE | STA | STO | ACKRQ | ARBLOST | ACK | SI |
| Type | R | R | R/W | R/W | R | R | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC0; Bit-Addressable

| Bit | Name | Description | Read | Write |
|---|---|---|---|---|
| 7 | MASTER | **SMBus Master/Slave Indicator.** This read-only bit indicates when the SMBus is operating as a master. | 0: SMBus operating in slave mode.<br>1: SMBus operating in master mode. | N/A |
| 6 | TXMODE | **SMBus Transmit Mode Indicator.** This read-only bit indicates when the SMBus is operating as a transmitter. | 0: SMBus in Receiver Mode.<br>1: SMBus in Transmitter Mode. | N/A |
| 5 | STA | **SMBus Start Flag.** | 0: No Start or repeated Start detected.<br>1: Start or repeated Start detected. | 0: No Start generated.<br>1: When Configured as a Master, initiates a START or repeated START. |
| 4 | STO | **SMBus Stop Flag.** | 0: No Stop condition detected.<br>1: Stop condition detected (if in Slave Mode) or pending (if in Master Mode). | 0: No STOP condition is transmitted.<br>1: When configured as a Master, causes a STOP condition to be transmitted after the next ACK cycle.<br>Cleared by Hardware. |
| 3 | ACKRQ | **SMBus Acknowledge Request.** | 0: No Ack requested<br>1: ACK requested | N/A |
| 2 | ARBLOST | **SMBus Arbitration Lost Indicator.** | 0: No arbitration error.<br>1: Arbitration Lost | N/A |
| 1 | ACK | **SMBus Acknowledge.** | 0: NACK received.<br>1: ACK received. | 0: Send NACK<br>1: Send ACK |
| 0 | SI | **SMBus Interrupt Flag.** This bit is set by hardware under the conditions listed in Table 15.3. SI must be cleared by software. While SI is set, SCL is held low and the SMBus is stalled. | 0: No interrupt pending<br>1: Interrupt Pending | 0: Clear interrupt, and initiate next state machine event.<br>1: Force interrupt. |

**Table 21.3. Sources for Hardware Changes to SMB0CN**

| Bit | Set by Hardware When: | Cleared by Hardware When: |
|---|---|---|
| MASTER | ■ A START is generated. | ■ A STOP is generated.<br>■ Arbitration is lost. |
| TXMODE | ■ START is generated.<br>■ SMB0DAT is written before the start of an SMBus frame. | ■ A START is detected.<br>■ Arbitration is lost.<br>■ SMB0DAT is not written before the start of an SMBus frame. |
| STA | ■ A START followed by an address byte is received. | ■ Must be cleared by software. |
| STO | ■ A STOP is detected while addressed as a slave.<br>■ Arbitration is lost due to a detected STOP. | ■ A pending STOP is generated. |
| ACKRQ | ■ A byte has been received and an ACK response value is needed (only when hardware ACK is not enabled). | ■ After each ACK cycle. |
| ARBLOST | ■ A repeated START is detected as a MASTER when STA is low (unwanted repeated START).<br>■ SCL is sensed low while attempting to generate a STOP or repeated START condition.<br>■ SDA is sensed low while transmitting a 1 (excluding ACK bits). | ■ Each time SI is cleared. |
| ACK | ■ The incoming ACK value is low (ACKNOWLEDGE). | ■ The incoming ACK value is high (NOT ACKNOWLEDGE). |
| SI | ■ A START has been generated.<br>■ Lost arbitration.<br>■ A byte has been transmitted and an ACK/NACK received.<br>■ A byte has been received.<br>■ A START or repeated START followed by a slave address + R/W has been received.<br>■ A STOP has been received. | ■ Must be cleared by software. |

### 21.4.3. Hardware Slave Address Recognition

The SMBus hardware has the capability to automatically recognize incoming slave addresses and send an ACK without software intervention. Automatic slave address recognition is enabled by setting the EHACK bit in register SMB0ADM to 1. This will enable both automatic slave address recognition and automatic hardware ACK generation for received bytes (as a master or slave). More detail on automatic hardware ACK generation can be found in Section 21.4.2.2.

The registers used to define which address(es) are recognized by the hardware are the SMBus Slave Address register (SFR Definition 21.3) and the SMBus Slave Address Mask register (SFR Definition 21.4). A single address or range of addresses (including the General Call Address 0x00) can be specified using these two registers. The most-significant seven bits of the two registers are used to define which addresses will be ACKed. A 1 in bit positions of the slave address mask SLVM[6:0] enable a comparison between the received slave address and the hardware's slave address SLV[6:0] for those bits. A 0 in a bit

of the slave address mask means that bit will be treated as a “don’t care” for comparison purposes. In this case, either a 1 or a 0 value are acceptable on the incoming slave address. Additionally, if the GC bit in register SMB0ADR is set to 1, hardware will recognize the General Call Address (0x00). Table 21.4 shows some example parameter settings and the slave addresses that will be recognized by hardware under those conditions.

**Table 21.4. Hardware Address Recognition Examples (EHACK = 1)**

| Hardware Slave Address SLV[6:0] | Slave Address Mask SLVM[6:0] | GC bit | Slave Addresses Recognized by Hardware |
|---|---|---|---|
| 0x34 | 0x7F | 0 | 0x34 |
| 0x34 | 0x7F | 1 | 0x34, 0x00 (General Call) |
| 0x34 | 0x7E | 0 | 0x34, 0x35 |
| 0x34 | 0x7E | 1 | 0x34, 0x35, 0x00 (General Call) |
| 0x70 | 0x73 | 0 | 0x70, 0x74, 0x78, 0x7C |

## SFR Definition 21.3. SMB0ADR: SMBus Slave Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SLV[6:0] | | | | | | | GC |
| Type | R/W | | | | | | | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD7

| Bit | Name | Function |
|---|---|---|
| 7:1 | SLV[6:0] | **SMBus Hardware Slave Address.**<br>Defines the SMBus Slave Address(es) for automatic hardware acknowledgement. Only address bits which have a 1 in the corresponding bit position in SLVM[6:0] are checked against the incoming address. This allows multiple addresses to be recognized. |
| 0 | GC | **General Call Address Enable.**<br>When hardware address recognition is enabled (EHACK = 1), this bit will determine whether the General Call Address (0x00) is also recognized by hardware.<br>0: General Call Address is ignored.<br>1: General Call Address is recognized. |

## SFR Definition 21.4. SMB0ADM: SMBus Slave Address Mask

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SLVM[6:0] | | | | | | | EHACK |
| Type | R/W | | | | | | | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |

SFR Address = 0xE7

| Bit | Name | Function |
|---|---|---|
| 7:1 | SLVM[6:0] | **SMBus Slave Address Mask.**<br>Defines which bits of register SMB0ADR are compared with an incoming address byte, and which bits are ignored. Any bit set to 1 in SLVM[6:0] enables comparisons with the corresponding bit in SLV[6:0]. Bits set to 0 are ignored (can be either 0 or 1 in the incoming address). |
| 0 | EHACK | **Hardware Acknowledge Enable.**<br>Enables hardware acknowledgement of slave address and received data bytes.<br>0: Firmware must manually acknowledge all incoming address and data bytes.<br>1: Automatic Slave Address Recognition and Hardware Acknowledge is Enabled. |

### 21.4.4. Data Register

The SMBus Data register SMB0DAT holds a byte of serial data to be transmitted or one that has just been received. Software may safely read or write to the data register when the SI flag is set. Software should not attempt to access the SMB0DAT register when the SMBus is enabled and the SI flag is cleared to logic 0, as the interface may be in the process of shifting a byte of data into or out of the register.

Data in SMB0DAT is always shifted out MSB first. After a byte has been received, the first bit of received data is located at the MSB of SMB0DAT. While data is being shifted out, data on the bus is simultaneously being shifted in. SMB0DAT always contains the last data byte present on the bus. In the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data or address in SMB0DAT.

## SFR Definition 21.5. SMB0DAT: SMBus Data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SMB0DAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC2

| Bit | Name | Function |
|---|---|---|
| 7:0 | SMB0DAT[7:0] | **SMBus Data.**<br>The SMB0DAT register contains a byte of data to be transmitted on the SMBus serial interface or a byte that has just been received on the SMBus serial interface. The CPU can read from or write to this register whenever the SI serial interrupt flag (SMB0CN.0) is set to logic 1. The serial data in the register remains stable as long as the SI flag is set. When the SI flag is not set, the system may be in the process of shifting data in/out and the CPU should not attempt to access this register. |

## 21.5. SMBus Transfer Modes

The SMBus interface may be configured to operate as master and/or slave. At any particular time, it will be operating in one of the following four modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. The SMBus interface enters Master Mode any time a START is generated, and remains in Master Mode until it loses an arbitration or generates a STOP. An SMBus interrupt is generated at the end of all SMBus byte frames. Note that the position of the ACK interrupt when operating as a receiver depends on whether hardware ACK generation is enabled. As a receiver, the interrupt for an ACK occurs **before** the ACK with hardware ACK generation disabled, and **after** the ACK when hardware ACK generation is enabled. As a transmitter, interrupts occur **after** the ACK, regardless of whether hardware ACK generation is enabled or not.

### 21.5.1. Write Sequence (Master)

During a write sequence, an SMBus master writes data to a slave device. The master in this transfer will be a transmitter during the address byte, and a transmitter during all data bytes. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 0 (WRITE). The master then transmits one or more bytes of serial data. After each byte is transmitted, an acknowledge bit is generated by the slave. The transfer is ended when the STO bit is set and a STOP is generated. Note that the interface will switch to Master Receiver Mode if SMB0DAT is not written following a Master Transmitter interrupt. Figure 21.5 shows a typical master write sequence. Two transmit data bytes are shown, though any number of bytes may be transmitted. Notice that all of the “data byte transferred” interrupts occur **after** the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

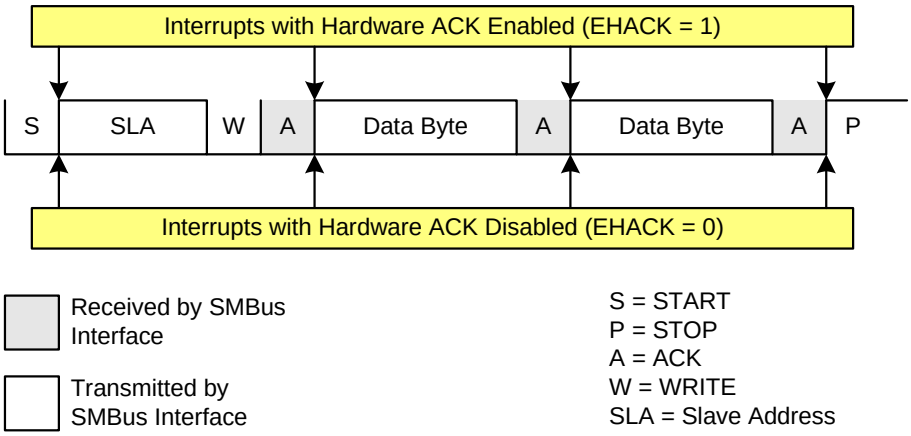



**Figure 21.5. Typical Master Write Sequence**

### 21.5.2. Read Sequence (Master)

During a read sequence, an SMBus master reads data from a slave device. The master in this transfer will be a transmitter during the address byte, and a receiver during all data bytes. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 1 (READ). Serial data is then received from the slave on SDA while the SMBus outputs the serial clock. The slave transmits one or more bytes of serial data.

If hardware ACK generation is disabled, the ACKRQ is set to 1 and an interrupt is generated after each received byte. Software must write the ACK bit at that time to ACK or NACK the received byte.

With hardware ACK generation enabled, the SMBus hardware will automatically generate the ACK/NACK, and then post the interrupt. **It is important to note that the appropriate ACK or NACK value should be set up by the software prior to receiving the byte when hardware ACK generation is enabled.**

Writing a 1 to the ACK bit generates an ACK; writing a 0 generates a NACK. Software should write a 0 to the ACK bit for the last data transfer, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. The interface will switch to Master Transmitter Mode if SMB0DAT is written while an active Master Receiver. Figure 21.6 shows a typical master read sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the ‘data byte transferred’ interrupts occur at different places in the sequence, depending on whether hardware ACK generation is enabled. The interrupt occurs **before** the ACK with hardware ACK generation disabled, and **after** the ACK when hardware ACK generation is enabled.

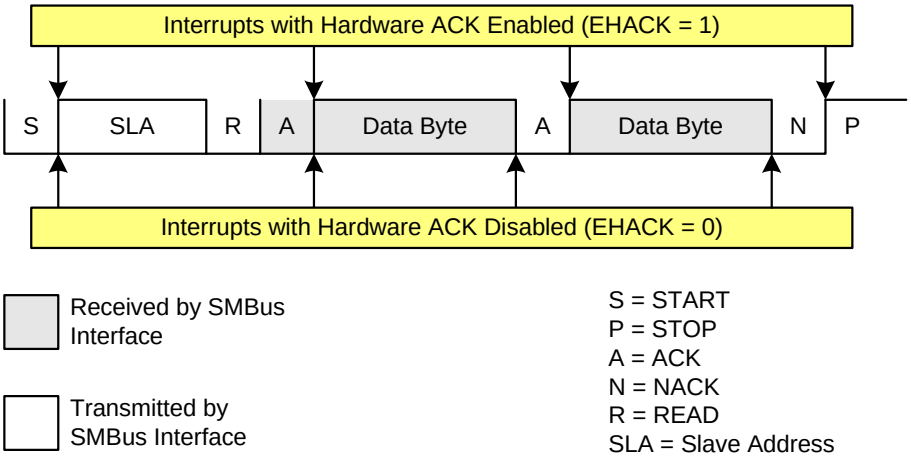



**Figure 21.6. Typical Master Read Sequence**

### 21.5.3. Write Sequence (Slave)

During a write sequence, an SMBus master writes data to a slave device. The slave in this transfer will be a receiver during the address byte, and a receiver during all data bytes. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode when a START followed by a slave address and direction bit (WRITE in this case) is received. If hardware ACK generation is disabled, upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. The software must respond to the received slave address with an ACK, or ignore the received slave address with a NACK. If hardware ACK generation is enabled, the hardware will apply the ACK for a slave address which matches the criteria set up by SMB0ADR and SMB0ADM. The interrupt will occur after the ACK cycle.

If the received slave address is ignored (by software or hardware), slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are received.

If hardware ACK generation is disabled, the ACKRQ is set to 1 and an interrupt is generated after each received byte. Software must write the ACK bit at that time to ACK or NACK the received byte.

With hardware ACK generation enabled, the SMBus hardware will automatically generate the ACK/NACK, and then post the interrupt. **It is important to note that the appropriate ACK or NACK value should be set up by the software prior to receiving the byte when hardware ACK generation is enabled.**

The interface exits Slave Receiver Mode after receiving a STOP. Note that the interface will switch to Slave Transmitter Mode if SMB0DAT is written while an active Slave Receiver. Figure 21.7 shows a typical slave write sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the ‘data byte transferred’ interrupts occur at different places in the sequence, depending on whether hardware ACK generation is enabled. The interrupt occurs **before** the ACK with hardware ACK generation disabled, and **after** the ACK when hardware ACK generation is enabled.

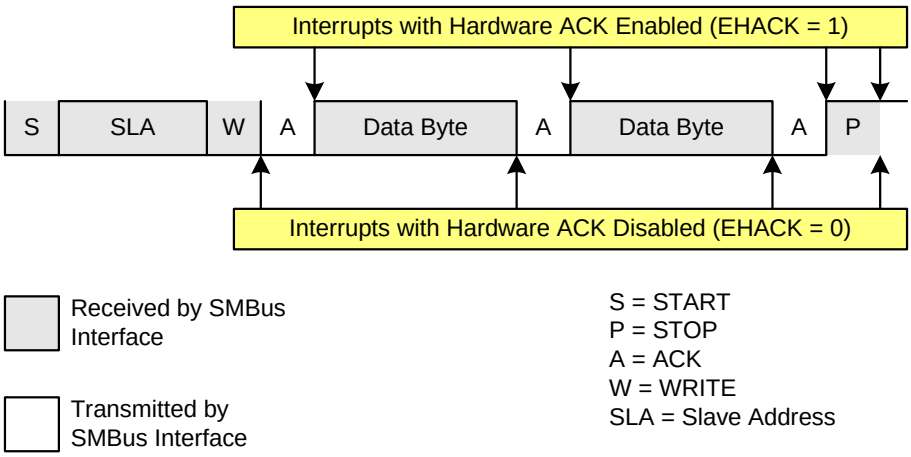



**Figure 21.7. Typical Slave Write Sequence**

### 21.5.4. Read Sequence (Slave)

During a read sequence, an SMBus master reads data from a slave device. The slave in this transfer will be a receiver during the address byte, and a transmitter during all data bytes. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode (to receive the slave address) when a START followed by a slave address and direction bit (READ in this case) is received. If hardware ACK generation is disabled, upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. The software must respond to the received slave address with an ACK, or ignore the received slave address with a NACK. If hardware ACK generation is enabled, the hardware will apply the ACK for a slave address which matches the criteria set up by SMB0ADR and SMB0ADM. The interrupt will occur after the ACK cycle.

If the received slave address is ignored (by software or hardware), slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are transmitted. If the received slave address is acknowledged, data should be written to SMB0DAT to be transmitted. The interface enters slave transmitter mode, and transmits one or more bytes of data. After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (an error condition may be generated if SMB0DAT is written following a received NACK while in slave transmitter mode). The interface exits slave transmitter mode after receiving a STOP. Note that the interface will switch to slave receiver mode if SMB0DAT is not written following a Slave Transmitter interrupt. Figure 21.8 shows a typical slave read sequence. Two transmitted data bytes are shown, though any number of bytes may be transmitted. Notice that all of the “data byte transferred” interrupts occur **after** the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

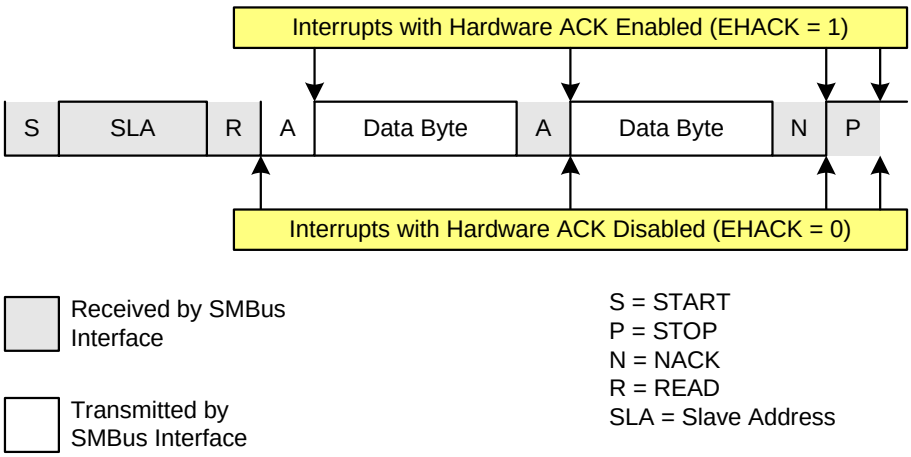



**Figure 21.8. Typical Slave Read Sequence**

## 21.6. SMBus Status Decoding

The current SMBus status can be easily decoded using the SMB0CN register. The appropriate actions to take in response to an SMBus event depend on whether hardware slave address recognition and ACK generation is enabled or disabled. Table 21.5 describes the typical actions when hardware slave address recognition and ACK generation is disabled. Table 21.6 describes the typical actions when hardware slave address recognition and ACK generation is enabled. In the tables, STATUS VECTOR refers to the four upper bits of SMB0CN: MASTER, TXMODE, STA, and STO. The shown response options are only the typical responses; application-specific procedures are allowed as long as they conform to the SMBus specification. Highlighted responses are allowed by hardware but do not conform to the SMBus specification.

**Table 21.5. SMBus Status Decoding With Hardware ACK Generation Disabled (EHACK = 0)**

| Mode | Values Read | | | | Current SMbus State | Typical Response Options | Values to Write | | | Next Status Vector Expected |
|---|---|---|---|---|---|---|---|---|---|---|
| | Status Vector | ACKRQ | ARBLOST | ACK | | | STA | STO | ACK | |
| Master Transmitter | 1110 | 0 | 0 | X | A master START was generated. | Load slave address + R/W into SMB0DAT. | 0 | 0 | X | 1100 |
| | 1100 | 0 | 0 | 0 | A master data or address byte was transmitted; NACK received. | Set STA to restart transfer. | 1 | 0 | X | 1110 |
| | | | | | | Abort transfer. | 0 | 1 | X | — |
| | | 0 | 0 | 1 | A master data or address byte was transmitted; ACK received. | Load next data byte into SMB0DAT. | 0 | 0 | X | 1100 |
| | | | | | | End transfer with STOP. | 0 | 1 | X | — |
| | | | | | | End transfer with STOP and start another transfer. | 1 | 1 | X | — |
| | | | | | | Send repeated START. | 1 | 0 | X | 1110 |
| | | | | | | Switch to Master Receiver Mode (clear SI without writing new data to SMB0DAT). | 0 | 0 | X | 1000 |
| Master Receiver | 1000 | 1 | 0 | X | A master data byte was received; ACK requested. | Acknowledge received byte; Read SMB0DAT. | 0 | 0 | 1 | 1000 |
| | | | | | | Send NACK to indicate last byte, and send STOP. | 0 | 1 | 0 | — |
| | | | | | | Send NACK to indicate last byte, and send STOP followed by START. | 1 | 1 | 0 | 1110 |
| | | | | | | Send ACK followed by repeated START. | 1 | 0 | 1 | 1110 |
| | | | | | | Send NACK to indicate last byte, and send repeated START. | 1 | 0 | 0 | 1110 |
| | | | | | | Send ACK and switch to Master Transmitter Mode (write to SMB0DAT before clearing SI). | 0 | 0 | 1 | 1100 |
| | | | | | | Send NACK and switch to Master Transmitter Mode (write to SMB0DAT before clearing SI). | 0 | 0 | 0 | 1100 |

**Table 21.5. SMBus Status Decoding With Hardware ACK Generation Disabled (EHACK = 0) (Continued)**

| Mode | Values Read | | | | Current SMbus State | Typical Response Options | Values to Write | | | Next Status Vector Expected |
|---|---|---|---|---|---|---|---|---|---|---|
| | Status Vector | ACKRQ | ARBLOST | ACK | | | STA | STO | ACK | |
| Slave Transmitter | 0100 | 0 | 0 | 0 | A slave byte was transmitted; NACK received. | No action required (expecting STOP condition). | 0 | 0 | X | 0001 |
| | | 0 | 0 | 1 | A slave byte was transmitted; ACK received. | Load SMB0DAT with next data byte to transmit. | 0 | 0 | X | 0100 |
| | | 0 | 1 | X | A Slave byte was transmitted; error detected. | No action required (expecting Master to end transfer). | 0 | 0 | X | 0001 |
| | 0101 | 0 | X | X | An illegal STOP or bus error was detected while a Slave Transmission was in progress. | Clear STO. | 0 | 0 | X | — |
| Slave Receiver | 0010 | 1 | 0 | X | A slave address + R/W was received; ACK requested. | If Write, Acknowledge received address | 0 | 0 | 1 | 0000 |
| | | | | | | If Read, Load SMB0DAT with data byte; ACK received address | 0 | 0 | 1 | 0100 |
| | | | | | | NACK received address. | 0 | 0 | 0 | — |
| | | 1 | 1 | X | Lost arbitration as master; slave address + R/W received; ACK requested. | If Write, Acknowledge received address | 0 | 0 | 1 | 0000 |
| | | | | | | If Read, Load SMB0DAT with data byte; ACK received address | 0 | 0 | 1 | 0100 |
| | | | | | | NACK received address. | 0 | 0 | 0 | — |
| | | | | | | Reschedule failed transfer; NACK received address. | 1 | 0 | 0 | 1110 |
| | 0001 | 0 | 0 | X | A STOP was detected while addressed as a Slave Transmitter or Slave Receiver. | Clear STO. | 0 | 0 | X | — |
| | | 1 | 1 | X | Lost arbitration while attempting a STOP. | No action required (transfer complete/aborted). | 0 | 0 | 0 | — |
| | 0000 | 1 | 0 | X | A slave byte was received; ACK requested. | Acknowledge received byte; Read SMB0DAT. | 0 | 0 | 1 | 0000 |
| | | | | | | NACK received byte. | 0 | 0 | 0 | — |
| Bus Error Condition | 0010 | 0 | 1 | X | Lost arbitration while attempting a repeated START. | Abort failed transfer. | 0 | 0 | X | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X | 1110 |
| | 0001 | 0 | 1 | X | Lost arbitration due to a detected STOP. | Abort failed transfer. | 0 | 0 | X | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X | 1110 |
| | 0000 | 1 | 1 | X | Lost arbitration while transmitting a data byte as master. | Abort failed transfer. | 0 | 0 | 0 | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | 0 | 1110 |

**Table 21.6. SMBus Status Decoding With Hardware ACK Generation Enabled (EHACK = 1)**

| Mode | Values Read | | | | Current SMbus State | Typical Response Options | Values to Write | | | Next Status Vector Expected |
|---|---|---|---|---|---|---|---|---|---|---|
| | Status Vector | ACKRQ | ARBLOST | ACK | | | STA | STO | ACK | |
| Master Transmitter | 1110 | 0 | 0 | X | A master START was generated. | Load slave address + R/W into SMB0DAT. | 0 | 0 | X | 1100 |
| | 1100 | 0 | 0 | 0 | A master data or address byte was transmitted; NACK received. | Set STA to restart transfer. | 1 | 0 | X | 1110 |
| | | | | | | Abort transfer. | 0 | 1 | X | — |
| | | 0 | 0 | 1 | A master data or address byte was transmitted; ACK received. | Load next data byte into SMB0DAT. | 0 | 0 | X | 1100 |
| | | | | | | End transfer with STOP. | 0 | 1 | X | — |
| | | | | | | End transfer with STOP and start another transfer. | 1 | 1 | X | — |
| | | | | | | Send repeated START. | 1 | 0 | X | 1110 |
| | | | | | | Switch to Master Receiver Mode (clear SI without writing new data to SMB0DAT). Set ACK for initial data byte. | 0 | 0 | 1 | 1000 |
| Master Receiver | 1000 | 0 | 0 | 1 | A master data byte was received; ACK sent. | Set ACK for next data byte; Read SMB0DAT. | 0 | 0 | 1 | 1000 |
| | | | | | | Set NACK to indicate next data byte as the last data byte; Read SMB0DAT. | 0 | 0 | 0 | 1000 |
| | | | | | | Initiate repeated START. | 1 | 0 | 0 | 1110 |
| | | | | | | Switch to Master Transmitter Mode (write to SMB0DAT before clearing SI). | 0 | 0 | X | 1100 |
| | | 0 | 0 | 0 | A master data byte was received; NACK sent (last byte). | Read SMB0DAT; send STOP. | 0 | 1 | 0 | — |
| | | | | | | Read SMB0DAT; Send STOP followed by START. | 1 | 1 | 0 | 1110 |
| | | | | | | Initiate repeated START. | 1 | 0 | 0 | 1110 |
| | | | | | | Switch to Master Transmitter Mode (write to SMB0DAT before clearing SI). | 0 | 0 | X | 1100 |

**Table 21.6. SMBus Status Decoding With Hardware ACK Generation Enabled (EHACK = 1) (Continued)**

| Mode | Values Read | | | | Current SMbus State | Typical Response Options | Values to Write | | | Next Status Vector Expected |
|---|---|---|---|---|---|---|---|---|---|---|
| | Status Vector | ACKRQ | ARBLOST | ACK | | | STA | STO | ACK | |
| Slave Transmitter | 0100 | 0 | 0 | 0 | A slave byte was transmitted; NACK received. | No action required (expecting STOP condition). | 0 | 0 | X | 0001 |
| | | 0 | 0 | 1 | A slave byte was transmitted; ACK received. | Load SMB0DAT with next data byte to transmit. | 0 | 0 | X | 0100 |
| | | 0 | 1 | X | A Slave byte was transmitted; error detected. | No action required (expecting Master to end transfer). | 0 | 0 | X | 0001 |
| | 0101 | 0 | X | X | An illegal STOP or bus error was detected while a Slave Transmission was in progress. | Clear STO. | 0 | 0 | X | — |
| Slave Receiver | 0010 | 0 | 0 | X | A slave address + R/W was received; ACK sent. | If Write, Set ACK for first data byte. | 0 | 0 | 1 | 0000 |
| | | | | | | If Read, Load SMB0DAT with data byte | 0 | 0 | X | 0100 |
| | | 0 | 1 | X | Lost arbitration as master; slave address + R/W received; ACK sent. | If Write, Set ACK for first data byte. | 0 | 0 | 1 | 0000 |
| | | | | | | If Read, Load SMB0DAT with data byte | 0 | 0 | X | 0100 |
| | | | | | | Reschedule failed transfer | 1 | 0 | X | 1110 |
| | 0001 | 0 | 0 | X | A STOP was detected while addressed as a Slave Transmitter or Slave Receiver. | Clear STO. | 0 | 0 | X | — |
| | | 0 | 1 | X | Lost arbitration while attempting a STOP. | No action required (transfer complete/aborted). | 0 | 0 | 0 | — |
| | 0000 | 0 | 0 | X | A slave byte was received. | Set ACK for next data byte; Read SMB0DAT. | 0 | 0 | 1 | 0000 |
| | | | | | | Set NACK for next data byte; Read SMB0DAT. | 0 | 0 | 0 | 0000 |
| Bus Error Condition | 0010 | 0 | 1 | X | Lost arbitration while attempting a repeated START. | Abort failed transfer. | 0 | 0 | X | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X | 1110 |
| | 0001 | 0 | 1 | X | Lost arbitration due to a detected STOP. | Abort failed transfer. | 0 | 0 | X | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X | 1110 |
| | 0000 | 0 | 1 | X | Lost arbitration while transmitting a data byte as master. | Abort failed transfer. | 0 | 0 | X | — |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X | 1110 |

## 22. UART0

UART0 is an asynchronous, full duplex serial port offering modes 1 and 3 of the standard 8051 UART. Enhanced baud rate support allows a wide range of clock sources to generate standard baud rates (details in Section “22.1. Enhanced Baud Rate Generation” on page 160). Received data buffering allows UART0 to start reception of a second incoming data byte before software has finished reading the previous data byte.

UART0 has two associated SFRs: Serial Control Register 0 (SCON0) and Serial Data Buffer 0 (SBUF0). The single SBUF0 location provides access to both transmit and receive registers. **Writes to SBUF0 always access the Transmit register. Reads of SBUF0 always access the buffered Receive register; it is not possible to read data from the Transmit register.**

With UART0 interrupts enabled, an interrupt is generated each time a transmit is completed (TI0 is set in SCON0), or a data byte has been received (RI0 is set in SCON0). The UART0 interrupt flags are not cleared by hardware when the CPU vectors to the interrupt service routine. They must be cleared manually by software, allowing software to determine the cause of the UART0 interrupt (transmit complete or receive complete).

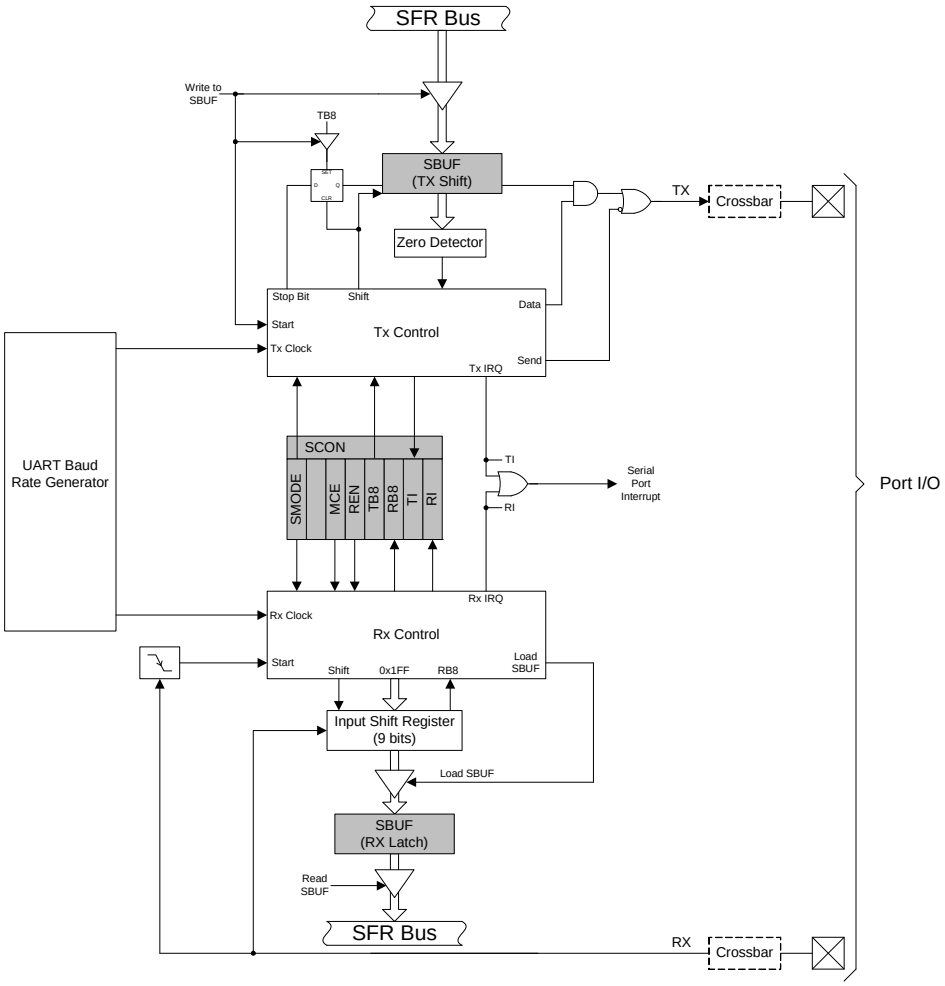



**Figure 22.1. UART0 Block Diagram**

## 22.1. Enhanced Baud Rate Generation

The UART0 baud rate is generated by Timer 1 in 8-bit auto-reload mode. The TX clock is generated by TL1; the RX clock is generated by a copy of TL1 (shown as RX Timer in Figure 22.2), which is not user-accessible. Both TX and RX Timer overflows are divided by two to generate the TX and RX baud rates. The RX Timer runs when Timer 1 is enabled, and uses the same reload value (TH1). However, an RX Timer reload is forced when a START condition is detected on the RX pin. This allows a receive to begin any time a START is detected, independent of the TX Timer state.

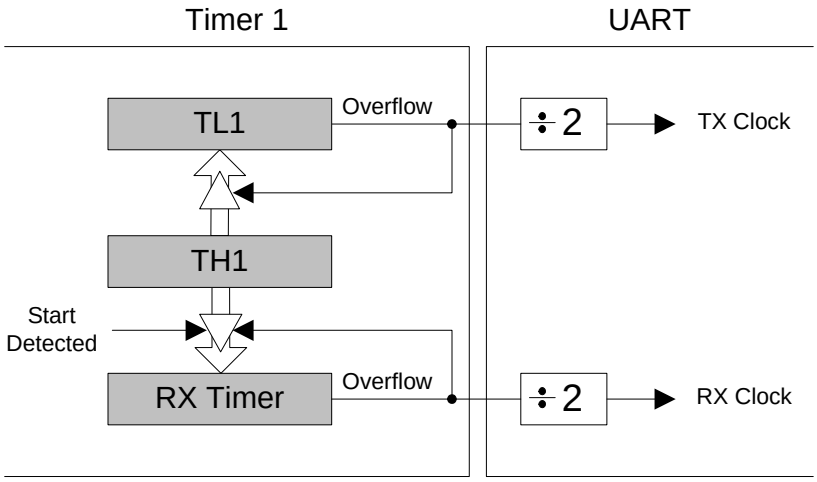



**Figure 22.2. UART0 Baud Rate Logic**

Timer 1 should be configured for Mode 2, 8-bit auto-reload (see Section “24.1.3. Mode 2: 8-bit Counter/Timer with Auto-Reload” on page 184). The Timer 1 reload value should be set so that overflows will occur at two times the desired UART baud rate frequency. Note that Timer 1 may be clocked by one of six sources: SYSCLK, SYSCLK/4, SYSCLK/12, SYSCLK/48, the external oscillator clock/8, or an external input T1. For any given Timer 1 clock source, the UART0 baud rate is determined by Equation 22.1-A and Equation 22.1-B.

A) $$\text{UartBaudRate} = \frac{1}{2} \times \text{T1_Overflow_Rate}$$

B) $$\text{T1_Overflow_Rate} = \frac{\text{T1}_{\text{CLK}}}{256 - \text{TH1}}$$

**Equation 22.1. UART0 Baud Rate**

Where $T1_{CLK}$ is the frequency of the clock supplied to Timer 1, and *T1H* is the high byte of Timer 1 (reload value). Timer 1 clock frequency is selected as described in Section “24. Timers” on page 180. A quick reference for typical baud rates and system clock frequencies is given in Table 22.1 through Table 22.2. The internal oscillator may still generate the system clock when the external oscillator is driving Timer 1.

## 22.2. Operational Modes

UART0 provides standard asynchronous, full duplex communication. The UART mode (8-bit or 9-bit) is selected by the S0MODE bit (SCON0.7). Typical UART connection options are shown in Figure 22.3.

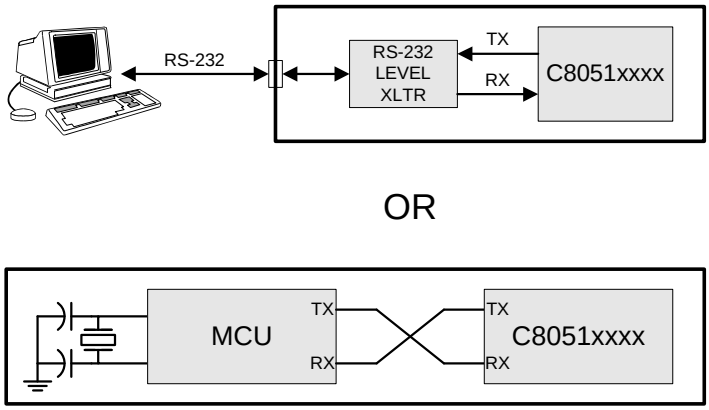



**Figure 22.3. UART Interconnect Diagram**

### 22.2.1. 8-Bit UART

8-Bit UART mode uses a total of 10 bits per data byte: one start bit, eight data bits (LSB first), and one stop bit. Data are transmitted LSB first from the TX0 pin and received at the RX0 pin. On receive, the eight data bits are stored in SBUF0 and the stop bit goes into RB80 (SCON0.2).

Data transmission begins when software writes a data byte to the SBUF0 register. The TI0 Transmit Interrupt Flag (SCON0.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN0 Receive Enable bit (SCON0.4) is set to logic 1. After the stop bit is received, the data byte will be loaded into the SBUF0 receive register if the following conditions are met: RI0 must be logic 0, and if MCE0 is logic 1, the stop bit must be logic 1. In the event of a receive data overrun, the first received 8 bits are latched into the SBUF0 receive register and the following overrun data bits are lost.

If these conditions are met, the eight bits of data is stored in SBUF0, the stop bit is stored in RB80 and the RI0 flag is set. If these conditions are not met, SBUF0 and RB80 will not be loaded and the RI0 flag will not be set. An interrupt will occur if enabled when either TI0 or RI0 is set.

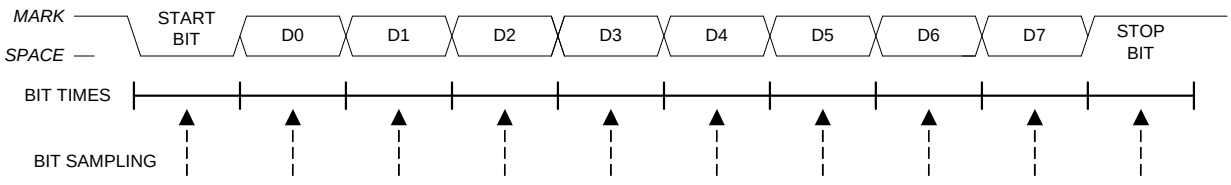



**Figure 22.4. 8-Bit UART Timing Diagram**

### 22.2.2. 9-Bit UART

9-bit UART mode uses a total of eleven bits per data byte: a start bit, 8 data bits (LSB first), a programmable ninth data bit, and a stop bit. The state of the ninth transmit data bit is determined by the value in TB80 (SCON0.3), which is assigned by user software. It can be assigned the value of the parity flag (bit P in register PSW) for error detection, or used in multiprocessor communications. On receive, the ninth data bit goes into RB80 (SCON0.2) and the stop bit is ignored.

Data transmission begins when an instruction writes a data byte to the SBUF0 register. The TI0 Transmit Interrupt Flag (SCON0.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN0 Receive Enable bit (SCON0.4) is set to 1. After the stop bit is received, the data byte will be loaded into the SBUF0 receive register if the following conditions are met: (1) RI0 must be logic 0, and (2) if MCE0 is logic 1, the 9th bit must be logic 1 (when MCE0 is logic 0, the state of the ninth data bit is unimportant). If these conditions are met, the eight bits of data are stored in SBUF0, the ninth bit is stored in RB80, and the RI0 flag is set to 1. If the above conditions are not met, SBUF0 and RB80 will not be loaded and the RI0 flag will not be set to 1. A UART0 interrupt will occur if enabled when either TI0 or RI0 is set to 1.

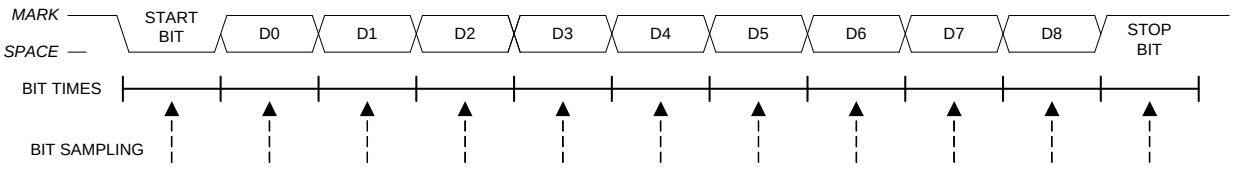



**Figure 22.5. 9-Bit UART Timing Diagram**

## 22.3. Multiprocessor Communications

9-Bit UART mode supports multiprocessor communication between a master processor and one or more slave processors by special use of the ninth data bit. When a master processor wants to transmit to one or more slaves, it first sends an address byte to select the target(s). An address byte differs from a data byte in that its ninth bit is logic 1; in a data byte, the ninth bit is always set to logic 0.

Setting the MCE0 bit (SCON0.5) of a slave processor configures its UART such that when a stop bit is received, the UART will generate an interrupt only if the ninth bit is logic 1 (RB80 = 1) signifying an address byte has been received. In the UART interrupt handler, software will compare the received address with the slave's own assigned 8-bit address. If the addresses match, the slave will clear its MCE0 bit to enable interrupts on the reception of the following data byte(s). Slaves that weren't addressed leave their MCE0 bits set and do not generate interrupts on the reception of the following data bytes, thereby ignoring the data. Once the entire message is received, the addressed slave resets its MCE0 bit to ignore all transmissions until it receives the next address byte.

Multiple addresses can be assigned to a single slave and/or a single address can be assigned to multiple slaves, thereby enabling "broadcast" transmissions to more than one slave simultaneously. The master processor can be configured to receive all transmissions or a protocol can be implemented such that the master/slave role is temporarily reversed to enable half-duplex transmission between the original master and slave(s).

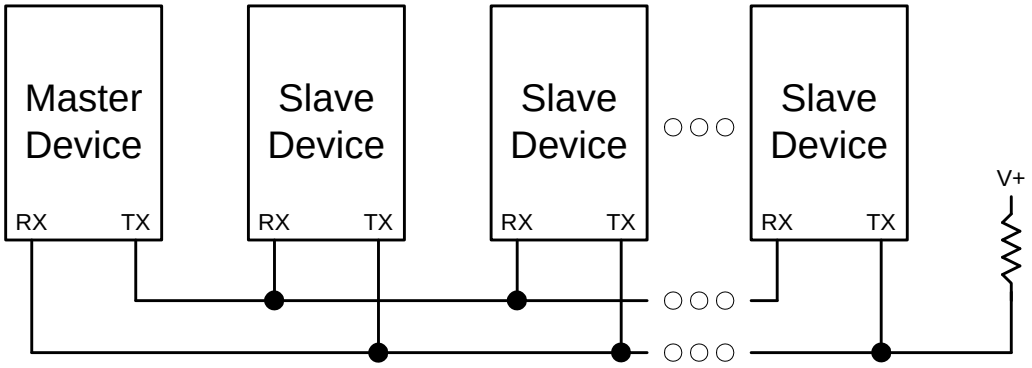



**Figure 22.6. UART Multi-Processor Mode Interconnect Diagram**

## SFR Definition 22.1. SCON0: Serial Port 0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | S0MODE | | MCE0 | REN0 | TB80 | RB80 | TI0 | RI0 |
| Type | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x98; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | S0MODE | **Serial Port 0 Operation Mode.**<br>Selects the UART0 Operation Mode.<br>0: 8-bit UART with Variable Baud Rate.<br>1: 9-bit UART with Variable Baud Rate. |
| 6 | Unused | Unused. Read = 1b, Write = Don't Care. |
| 5 | MCE0 | **Multiprocessor Communication Enable.**<br>The function of this bit is dependent on the Serial Port 0 Operation Mode:<br>**Mode 0: Checks for valid stop bit.**<br>0: Logic level of stop bit is ignored.<br>1: RI0 will only be activated if stop bit is logic level 1.<br>**Mode 1: Multiprocessor Communications Enable.**<br>0: Logic level of ninth bit is ignored.<br>1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1. |
| 4 | REN0 | **Receive Enable.**<br>0: UART0 reception disabled.<br>1: UART0 reception enabled. |
| 3 | TB80 | **Ninth Transmission Bit.**<br>The logic level of this bit will be sent as the ninth transmission bit in 9-bit UART Mode (Mode 1). Unused in 8-bit mode (Mode 0). |
| 2 | RB80 | **Ninth Receive Bit.**<br>RB80 is assigned the value of the STOP bit in Mode 0; it is assigned the value of the 9th data bit in Mode 1. |
| 1 | TI0 | **Transmit Interrupt Flag.**<br>Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in 8-bit UART Mode, or at the beginning of the STOP bit in 9-bit UART Mode). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software. |
| 0 | RI0 | **Receive Interrupt Flag.**<br>Set to 1 by hardware when a byte of data has been received by UART0 (set at the STOP bit sampling time). When the UART0 interrupt is enabled, setting this bit to 1 causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software. |

## SFR Definition 22.2. SBUF0: Serial (UART0) Port Data Buffer

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SBUF0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x99

| Bit | Name | Function |
|---|---|---|
| 7:0 | SBUF0[7:0] | **Serial Data Buffer Bits 7–0 (MSB–LSB).**<br>This SFR accesses two registers; a transmit shift register and a receive latch register. When data is written to SBUF0, it goes to the transmit shift register and is held for serial transmission. Writing a byte to SBUF0 initiates the transmission. A read of SBUF0 returns the contents of the receive latch. |

**Table 22.1. Timer Settings for Standard Baud Rates Using The Internal 24.5 MHz Oscillator**

| | Frequency: 24.5 MHz | | | | | | |
|---|---|---|---|---|---|---|---|
| | Target Baud Rate (bps) | Baud Rate % Error | Oscillator Divide Factor | Timer Clock Source | SCA1–SCA0 (pre-scale select)[1] | T1M[1] | Timer 1 Reload Value (hex) |
| SYSCLK from Internal Osc. | 230400 | –0.32% | 106 | SYSCLK | XX[2] | 1 | 0xCB |
| | 115200 | –0.32% | 212 | SYSCLK | XX | 1 | 0x96 |
| | 57600 | 0.15% | 426 | SYSCLK | XX | 1 | 0x2B |
| | 28800 | –0.32% | 848 | SYSCLK/4 | 01 | 0 | 0x96 |
| | 14400 | 0.15% | 1704 | SYSCLK/12 | 00 | 0 | 0xB9 |
| | 9600 | –0.32% | 2544 | SYSCLK/12 | 00 | 0 | 0x96 |
| | 2400 | –0.32% | 10176 | SYSCLK/48 | 10 | 0 | 0x96 |
| | 1200 | 0.15% | 20448 | SYSCLK/48 | 10 | 0 | 0x2B |

**Notes:**
1. SCA1–SCA0 and T1M bit definitions can be found in Section 24.1.
2. X = Don't care.

**Table 22.2. Timer Settings for Standard Baud Rates Using an External 22.1184 MHz Oscillator**

| | Frequency: 22.1184 MHz | | | | | | |
|---|---|---|---|---|---|---|---|
| | Target Baud Rate (bps) | Baud Rate % Error | Oscillator Divide Factor | Timer Clock Source | SCA1–SCA0 (pre-scale select)[1] | T1M[1] | Timer 1 Reload Value (hex) |
| SYSCLK from External Osc. | 230400 | 0.00% | 96 | SYSCLK | XX[2] | 1 | 0xD0 |
| | 115200 | 0.00% | 192 | SYSCLK | XX | 1 | 0xA0 |
| | 57600 | 0.00% | 384 | SYSCLK | XX | 1 | 0x40 |
| | 28800 | 0.00% | 768 | SYSCLK / 12 | 00 | 0 | 0xE0 |
| | 14400 | 0.00% | 1536 | SYSCLK / 12 | 00 | 0 | 0xC0 |
| | 9600 | 0.00% | 2304 | SYSCLK / 12 | 00 | 0 | 0xA0 |
| | 2400 | 0.00% | 9216 | SYSCLK / 48 | 10 | 0 | 0xA0 |
| | 1200 | 0.00% | 18432 | SYSCLK / 48 | 10 | 0 | 0x40 |
| SYSCLK from Internal Osc. | 230400 | 0.00% | 96 | EXTCLK / 8 | 11 | 0 | 0xFA |
| | 115200 | 0.00% | 192 | EXTCLK / 8 | 11 | 0 | 0xF4 |
| | 57600 | 0.00% | 384 | EXTCLK / 8 | 11 | 0 | 0xE8 |
| | 28800 | 0.00% | 768 | EXTCLK / 8 | 11 | 0 | 0xD0 |
| | 14400 | 0.00% | 1536 | EXTCLK / 8 | 11 | 0 | 0xA0 |
| | 9600 | 0.00% | 2304 | EXTCLK / 8 | 11 | 0 | 0x70 |

**Notes:**
1. SCA1–SCA0 and T1M bit definitions can be found in Section 24.1.
2. X = Don't care.

# 23. Enhanced Serial Peripheral Interface (SPI0)

The Enhanced Serial Peripheral Interface (SPI0) provides access to a flexible, full-duplex synchronous serial bus. SPI0 can operate as a master or slave device in both 3-wire or 4-wire modes, and supports multiple masters and slaves on a single SPI bus. The slave-select (NSS) signal can be configured as an input to select SPI0 in slave mode, or to disable Master Mode operation in a multi-master environment, avoiding contention on the SPI bus when more than one master attempts simultaneous data transfers. NSS can also be configured as a chip-select output in master mode, or disabled for 3-wire operation. Additional general purpose port I/O pins can be used to select multiple slave devices in master mode.

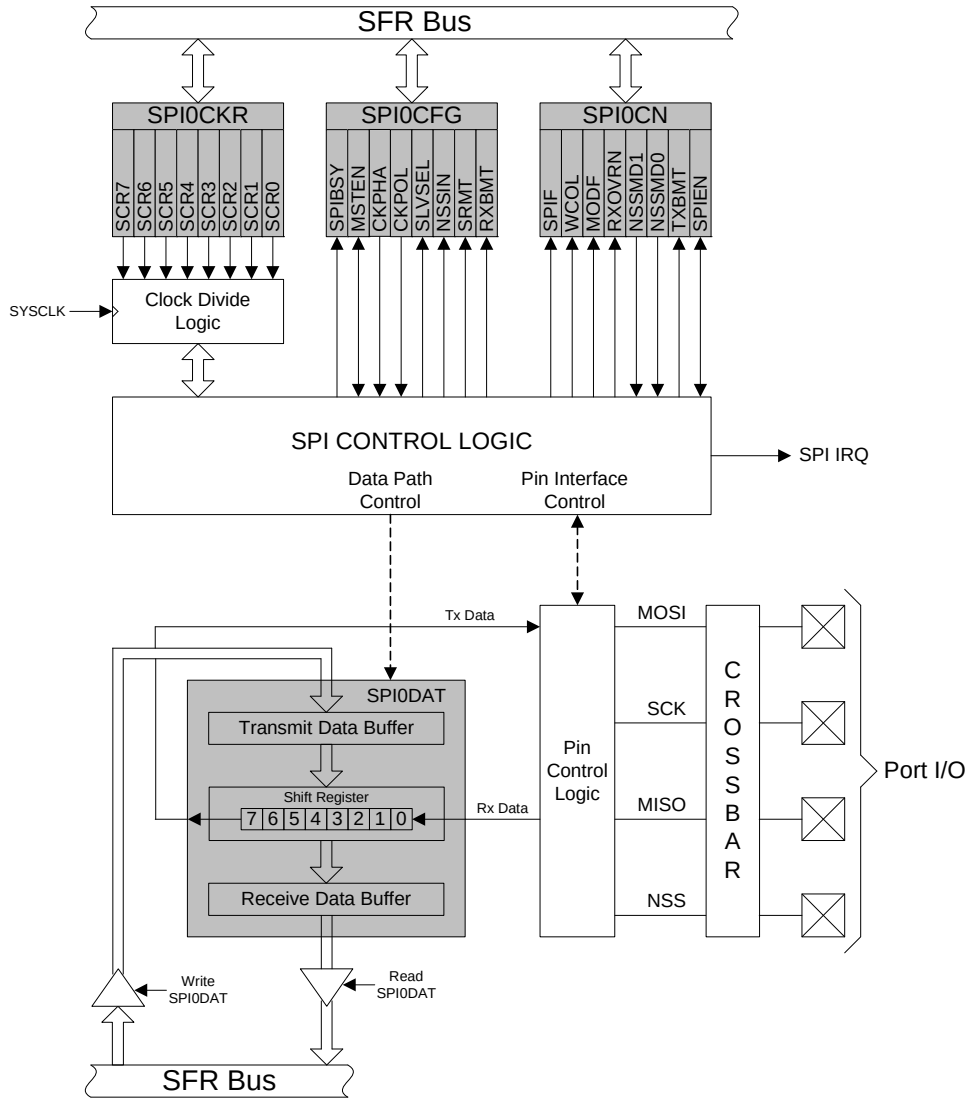



**Figure 23.1. SPI Block Diagram**

## 23.1. Signal Descriptions

The four signals used by SPI0 (MOSI, MISO, SCK, NSS) are described below.

### 23.1.1. Master Out, Slave In (MOSI)

The master-out, slave-in (MOSI) signal is an output from a master device and an input to slave devices. It is used to serially transfer data from the master to the slave. This signal is an output when SPI0 is operating as a master and an input when SPI0 is operating as a slave. Data is transferred most-significant bit first. When configured as a master, MOSI is driven by the MSB of the shift register in both 3- and 4-wire mode.

### 23.1.2. Master In, Slave Out (MISO)

The master-in, slave-out (MISO) signal is an output from a slave device and an input to the master device. It is used to serially transfer data from the slave to the master. This signal is an input when SPI0 is operating as a master and an output when SPI0 is operating as a slave. Data is transferred most-significant bit first. The MISO pin is placed in a high-impedance state when the SPI module is disabled and when the SPI operates in 4-wire mode as a slave that is not selected. When acting as a slave in 3-wire mode, MISO is always driven by the MSB of the shift register.

### 23.1.3. Serial Clock (SCK)

The serial clock (SCK) signal is an output from the master device and an input to slave devices. It is used to synchronize the transfer of data between the master and slave on the MOSI and MISO lines. SPI0 generates this signal when operating as a master. The SCK signal is ignored by a SPI slave when the slave is not selected (NSS = 1) in 4-wire slave mode.

### 23.1.4. Slave Select (NSS)

The function of the slave-select (NSS) signal is dependent on the setting of the NSSMD1 and NSSMD0 bits in the SPI0CN register. There are three possible modes that can be selected with these bits:

1. NSSMD[1:0] = 00: 3-Wire Master or 3-Wire Slave Mode: SPI0 operates in 3-wire mode, and NSS is disabled. When operating as a slave device, SPI0 is always selected in 3-wire mode. Since no select signal is present, SPI0 must be the only slave on the bus in 3-wire mode. This is intended for point-to-point communication between a master and one slave.
2. NSSMD[1:0] = 01: 4-Wire Slave or Multi-Master Mode: SPI0 operates in 4-wire mode, and NSS is enabled as an input. When operating as a slave, NSS selects the SPI0 device. When operating as a master, a 1-to-0 transition of the NSS signal disables the master function of SPI0 so that multiple master devices can be used on the same SPI bus.
3. NSSMD[1:0] = 1x: 4-Wire Master Mode: SPI0 operates in 4-wire mode, and NSS is enabled as an output. The setting of NSSMD0 determines what logic level the NSS pin will output. This configuration should only be used when operating SPI0 as a master device.

See Figure 23.2, Figure 23.3, and Figure 23.4 for typical connection diagrams of the various operational modes. **Note that the setting of NSSMD bits affects the pinout of the device.** When in 3-wire master or 3-wire slave mode, the NSS pin will not be mapped by the crossbar. In all other modes, the NSS signal will be mapped to a pin on the device. See Section “20. Port Input/Output” on page 119 for general purpose port I/O and crossbar information.

## 23.2. SPI0 Master Mode Operation

A SPI master device initiates all data transfers on a SPI bus. SPI0 is placed in master mode by setting the Master Enable flag (MSTEN, SPI0CN.6). Writing a byte of data to the SPI0 data register (SPI0DAT) when in master mode writes to the transmit buffer. If the SPI shift register is empty, the byte in the transmit buffer is moved to the shift register, and a data transfer begins. The SPI0 master immediately shifts out the data serially on the MOSI line while providing the serial clock on SCK. The SPIF (SPI0CN.7) flag is set to logic 1 at the end of the transfer. If interrupts are enabled, an interrupt request is generated when the SPIF flag is set. While the SPI0 master transfers data to a slave on the MOSI line, the addressed SPI slave device simultaneously transfers the contents of its shift register to the SPI master on the MISO line in a full-duplex operation. Therefore, the SPIF flag serves as both a transmit-complete and receive-data-ready flag. The data byte received from the slave is transferred MSB-first into the master's shift register. When a byte is fully shifted into the register, it is moved to the receive buffer where it can be read by the processor by reading SPI0DAT.

When configured as a master, SPI0 can operate in one of three different modes: multi-master mode, 3-wire single-master mode, and 4-wire single-master mode. The default, multi-master mode is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 1. In this mode, NSS is an input to the device, and is used to disable the master SPI0 when another master is accessing the bus. When NSS is pulled low in this mode, MSTEN (SPI0CN.6) and SPIEN (SPI0CN.0) are set to 0 to disable the SPI master device, and a Mode Fault is generated (MODF, SPI0CN.5 = 1). Mode Fault will generate an interrupt if enabled. SPI0 must be manually re-enabled in software under these circumstances. In multi-master systems, devices will typically default to being slave devices while they are not acting as the system master device. In multi-master mode, slave devices can be addressed individually (if needed) using general-purpose I/O pins. Figure 23.2 shows a connection diagram between two master devices in multiple-master mode.

3-wire single-master mode is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 0. In this mode, NSS is not used, and is not mapped to an external port pin through the crossbar. Any slave devices that must be addressed in this mode should be selected using general-purpose I/O pins. Figure 23.3 shows a connection diagram between a master device in 3-wire master mode and a slave device.

4-wire single-master mode is active when NSSMD1 (SPI0CN.3) = 1. In this mode, NSS is configured as an output pin, and can be used as a slave-select signal for a single SPI device. In this mode, the output value of NSS is controlled (in software) with the bit NSSMD0 (SPI0CN.2). Additional slave devices can be addressed using general-purpose I/O pins. Figure 23.4 shows a connection diagram for a master device in 4-wire master mode and two slave devices.

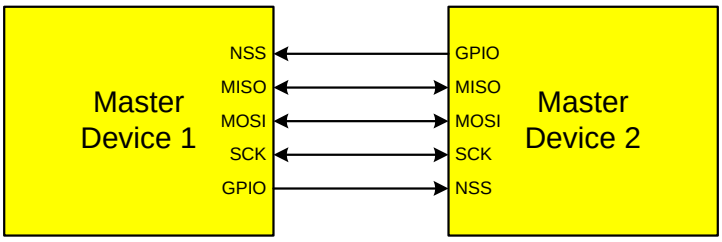



**Figure 23.2. Multiple-Master Mode Connection Diagram**

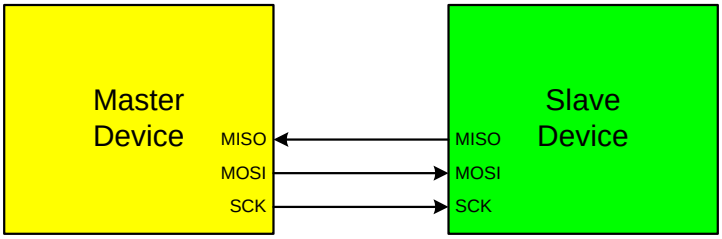



**Figure 23.3. 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram**

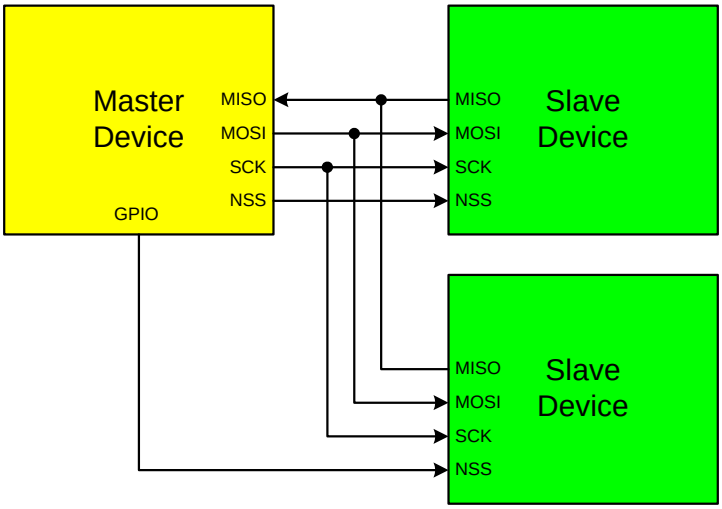



**Figure 23.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram**

## 23.3. SPI0 Slave Mode Operation

When SPI0 is enabled and not configured as a master, it will operate as a SPI slave. As a slave, bytes are shifted in through the MOSI pin and out through the MISO pin by a master device controlling the SCK signal. A bit counter in the SPI0 logic counts SCK edges. When 8 bits have been shifted through the shift register, the SPIF flag is set to logic 1, and the byte is copied into the receive buffer. Data is read from the receive buffer by reading SPI0DAT. A slave device cannot initiate transfers. Data to be transferred to the master device is pre-loaded into the shift register by writing to SPI0DAT. Writes to SPI0DAT are double-buffered, and are placed in the transmit buffer first. If the shift register is empty, the contents of the transmit buffer will immediately be transferred into the shift register. When the shift register already contains data, the SPI will load the shift register with the transmit buffer's contents after the last SCK edge of the next (or current) SPI transfer.

When configured as a slave, SPI0 can be configured for 4-wire or 3-wire operation. The default, 4-wire slave mode, is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 1. In 4-wire mode, the NSS signal is routed to a port pin and configured as a digital input. SPI0 is enabled when NSS is logic 0, and disabled when NSS is logic 1. The bit counter is reset on a falling edge of NSS. Note that the NSS signal must be driven low at least 2 system clocks before the first active edge of SCK for each byte transfer. Figure 23.4 shows a connection diagram between two slave devices in 4-wire slave mode and a master device.

3-wire slave mode is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 0. NSS is not used in this mode, and is not mapped to an external port pin through the crossbar. Since there is no way of uniquely addressing the device in 3-wire slave mode, SPI0 must be the only slave device present on the bus. It is important to note that in 3-wire slave mode there is no external means of resetting the bit counter that determines when a full byte has been received. The bit counter can only be reset by disabling and re-enabling SPI0 with the SPIEN bit. Figure 23.3 shows a connection diagram between a slave device in 3-wire slave mode and a master device.

## 23.4. SPI0 Interrupt Sources

When SPI0 interrupts are enabled, the following four flags will generate an interrupt when they are set to logic 1:

All of the following bits must be cleared by software.

- The SPI Interrupt Flag, SPIF (SPI0CN.7) is set to logic 1 at the end of each byte transfer. This flag can occur in all SPI0 modes.
- The Write Collision Flag, WCOL (SPI0CN.6) is set to logic 1 if a write to SPI0DAT is attempted when the transmit buffer has not been emptied to the SPI shift register. When this occurs, the write to SPI0DAT will be ignored, and the transmit buffer will not be written.This flag can occur in all SPI0 modes.
- The Mode Fault Flag MODF (SPI0CN.5) is set to logic 1 when SPI0 is configured as a master, and for multi-master mode and the NSS pin is pulled low. When a Mode Fault occurs, the MSTEN and SPIEN bits in SPI0CN are set to logic 0 to disable SPI0 and allow another master device to access the bus.
- The Receive Overrun Flag RXOVRN (SPI0CN.4) is set to logic 1 when configured as a slave, and a transfer is completed and the receive buffer still holds an unread byte from a previous transfer. The new byte is not transferred to the receive buffer, allowing the previously received data byte to be read. The data byte which caused the overrun is lost.

## 23.5. Serial Clock Phase and Polarity

Four combinations of serial clock phase and polarity can be selected using the clock control bits in the SPI0 Configuration Register (SPI0CFG). The CKPHA bit (SPI0CFG.5) selects one of two clock phases (edge used to latch the data). The CKPOL bit (SPI0CFG.4) selects between an active-high or active-low clock. Both master and slave devices must be configured to use the same clock phase and polarity. SPI0 should be disabled (by clearing the SPIEN bit, SPI0CN.0) when changing the clock phase or polarity. The clock and data line relationships for master mode are shown in Figure 23.5. For slave mode, the clock and data relationships are shown in Figure 23.6 and Figure 23.7. Note that CKPHA should be set to 0 on both the master and slave SPI when communicating between two Silicon Labs C8051 devices.

The SPI0 Clock Rate Register (SPI0CKR) as shown in SFR Definition 23.3 controls the master mode serial clock frequency. This register is ignored when operating in slave mode. When the SPI is configured as a master, the maximum data transfer rate (bits/sec) is one-half the system clock frequency or 12.5 MHz, whichever is slower. When the SPI is configured as a slave, the maximum data transfer rate (bits/sec) for full-duplex operation is 1/10 the system clock frequency, provided that the master issues SCK, NSS (in 4-wire slave mode), and the serial input data synchronously with the slave's system clock. If the master issues SCK, NSS, and the serial input data asynchronously, the maximum data transfer rate (bits/sec) must be less than 1/10 the system clock frequency. In the special case where the master only wants to transmit data to the slave and does not need to receive data from the slave (i.e. half-duplex operation), the SPI slave can receive data at a maximum data transfer rate (bits/sec) of 1/4 the system clock frequency. This is provided that the master issues SCK, NSS, and the serial input data synchronously with the slave's system clock.

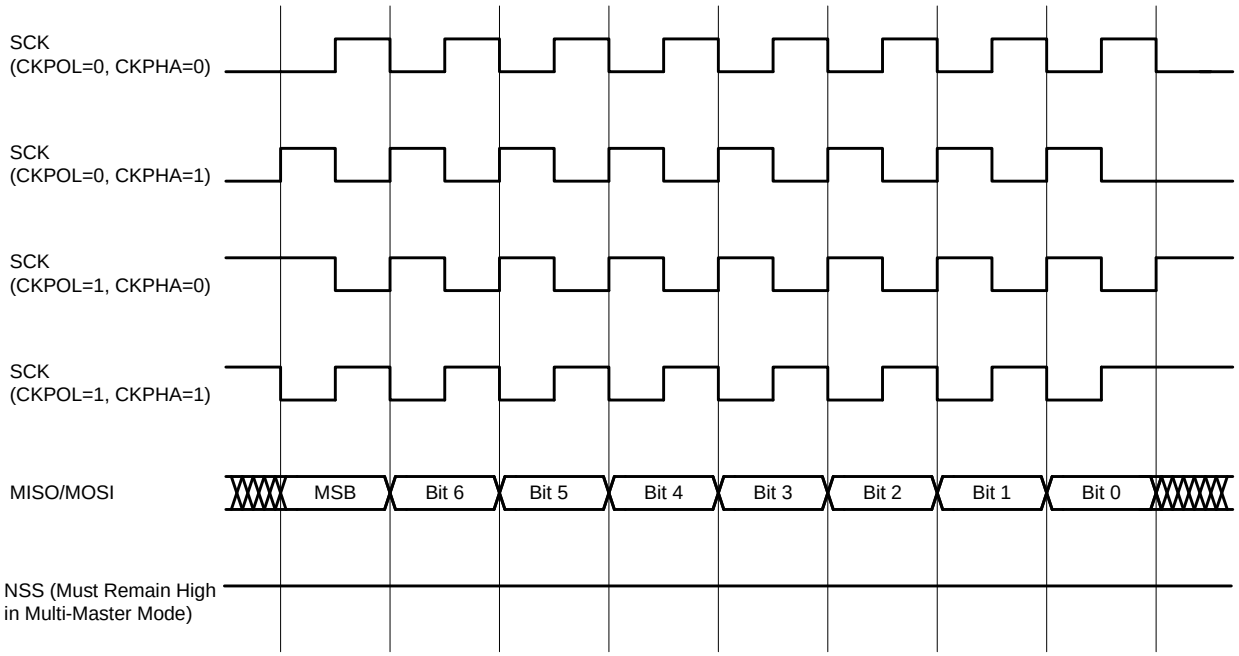



**Figure 23.5. Master Mode Data/Clock Timing**

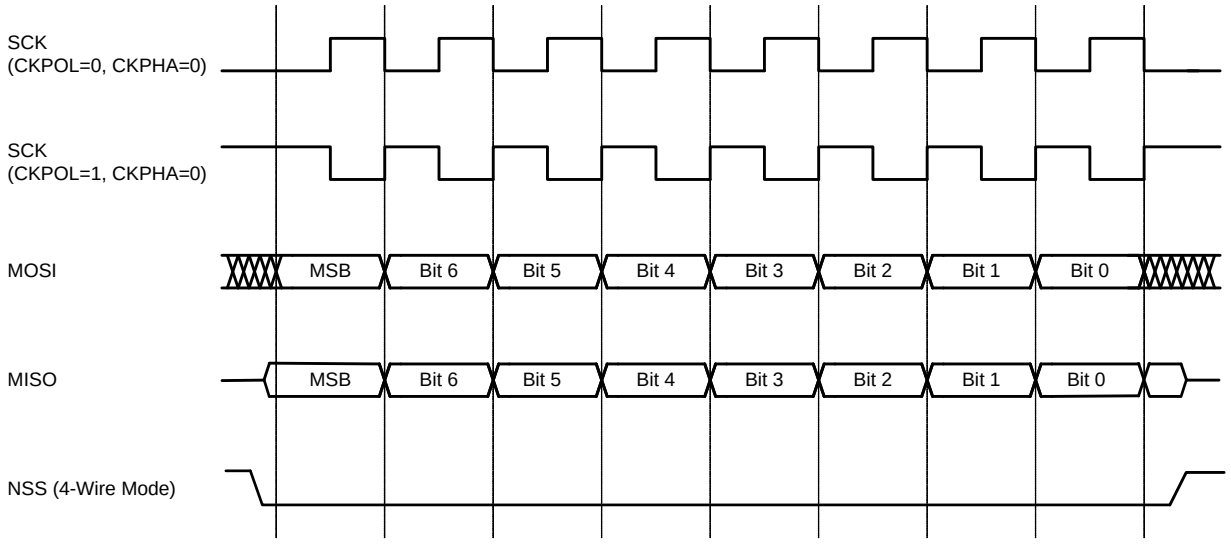



**Figure 23.6. Slave Mode Data/Clock Timing (CKPHA = 0)**

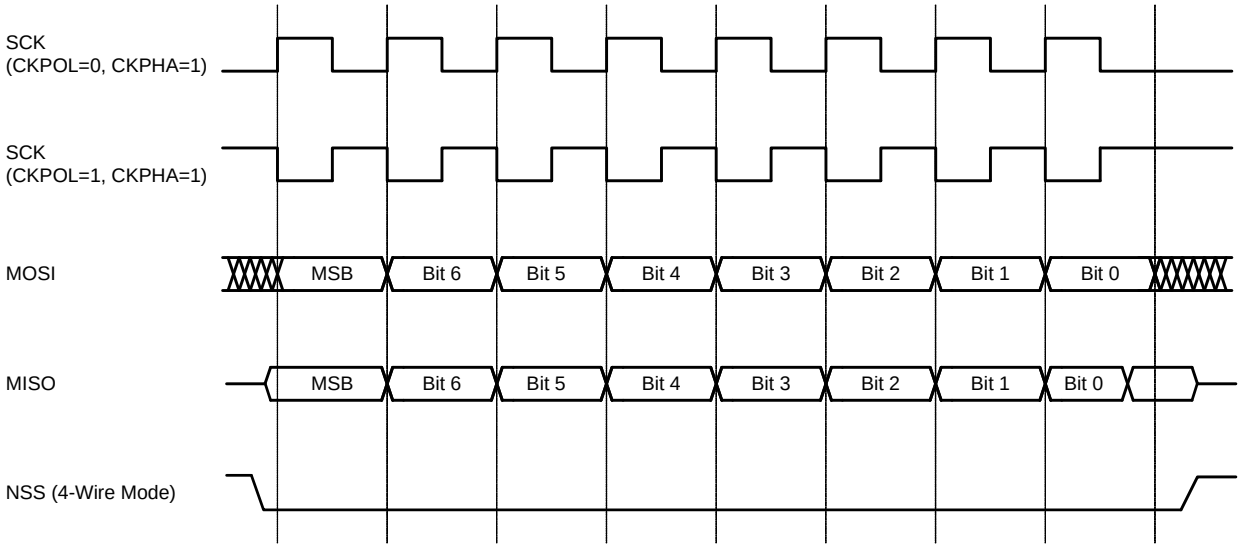



**Figure 23.7. Slave Mode Data/Clock Timing (CKPHA = 1)**

## 23.6. SPI Special Function Registers

SPI0 is accessed and controlled through four special function registers in the system controller: SPI0CN Control Register, SPI0DAT Data Register, SPI0CFG Configuration Register, and SPI0CKR Clock Rate Register. The four special function registers related to the operation of the SPI0 Bus are described in the following figures.

## SFR Definition 23.1. SPI0CFG: SPI0 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPIBSY | MSTEN | CKPHA | CKPOL | SLVSEL | NSSIN | SRMT | RXBMT |
| Type | R | R/W | R/W | R/W | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

SFR Address = 0xA1

| Bit | Name | Function |
|---|---|---|
| 7 | SPIBSY | **SPI Busy.**<br>This bit is set to logic 1 when a SPI transfer is in progress (master or slave mode). |
| 6 | MSTEN | **Master Mode Enable.**<br>0: Disable master mode. Operate in slave mode.<br>1: Enable master mode. Operate as a master. |
| 5 | CKPHA | **SPI0 Clock Phase.**<br>0: Data centered on first edge of SCK period.*<br>1: Data centered on second edge of SCK period.* |
| 4 | CKPOL | **SPI0 Clock Polarity.**<br>0: SCK line low in idle state.<br>1: SCK line high in idle state. |
| 3 | SLVSEL | **Slave Selected Flag.**<br>This bit is set to logic 1 whenever the NSS pin is low indicating SPI0 is the selected slave. It is cleared to logic 0 when NSS is high (slave not selected). This bit does not indicate the instantaneous value at the NSS pin, but rather a de-glitched version of the pin input. |
| 2 | NSSIN | **NSS Instantaneous Pin Input.**<br>This bit mimics the instantaneous value that is present on the NSS port pin at the time that the register is read. This input is not de-glitched. |
| 1 | SRMT | **Shift Register Empty (valid in slave mode only).**<br>This bit will be set to logic 1 when all data has been transferred in/out of the shift register, and there is no new information available to read from the transmit buffer or write to the receive buffer. It returns to logic 0 when a data byte is transferred to the shift register from the transmit buffer or by a transition on SCK. SRMT = 1 when in Master Mode. |
| 0 | RXBMT | **Receive Buffer Empty (valid in slave mode only).**<br>This bit will be set to logic 1 when the receive buffer has been read and contains no new information. If there is new information available in the receive buffer that has not been read, this bit will return to logic 0. RXBMT = 1 when in Master Mode. |

**Note:** In slave mode, data on MOSI is sampled in the center of each data bit. In master mode, data on MISO is sampled one SYSCLK before the end of each data bit, to provide maximum settling time for the slave device. See Table 23.1 for timing parameters.

## SFR Definition 23.2. SPI0CN: SPI0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPIF | WCOL | MODF | RXOVRN | NSSMD[1:0] | | TXBMT | SPIEN |
| Type | R/W | R/W | R/W | R/W | R/W | | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

SFR Address = 0xF8; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | SPIF | **SPI0 Interrupt Flag.**<br>This bit is set to logic 1 by hardware at the end of a data transfer. If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by software. |
| 6 | WCOL | **Write Collision Flag.**<br>This bit is set to logic 1 if a write to SPI0DAT is attempted when TXBMT is 0. When this occurs, the write to SPI0DAT will be ignored, and the transmit buffer will not be written. If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by software. |
| 5 | MODF | **Mode Fault Flag.**<br>This bit is set to logic 1 by hardware when a master mode collision is detected (NSS is low, MSTEN = 1, and NSSMD[1:0] = 01). If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by software. |
| 4 | RXOVRN | **Receive Overrun Flag (valid in slave mode only).**<br>This bit is set to logic 1 by hardware when the receive buffer still holds unread data from a previous transfer and the last bit of the current transfer is shifted into the SPI0 shift register. If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by software. |
| 3:2 | NSSMD[1:0] | **Slave Select Mode.**<br>Selects between the following NSS operation modes:<br>(See Section 23.2 and Section 23.3).<br>00: 3-Wire Slave or 3-Wire Master Mode. NSS signal is not routed to a port pin.<br>01: 4-Wire Slave or Multi-Master Mode (Default). NSS is an input to the device.<br>1x: 4-Wire Single-Master Mode. NSS signal is mapped as an output from the device and will assume the value of NSSMD0. |
| 1 | TXBMT | **Transmit Buffer Empty.**<br>This bit will be set to logic 0 when new data has been written to the transmit buffer. When data in the transmit buffer is transferred to the SPI shift register, this bit will be set to logic 1, indicating that it is safe to write a new byte to the transmit buffer. |
| 0 | SPIEN | **SPI0 Enable.**<br>0: SPI disabled.<br>1: SPI enabled. |

## SFR Definition 23.3. SPI0CKR: SPI0 Clock Rate

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SCR[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA2

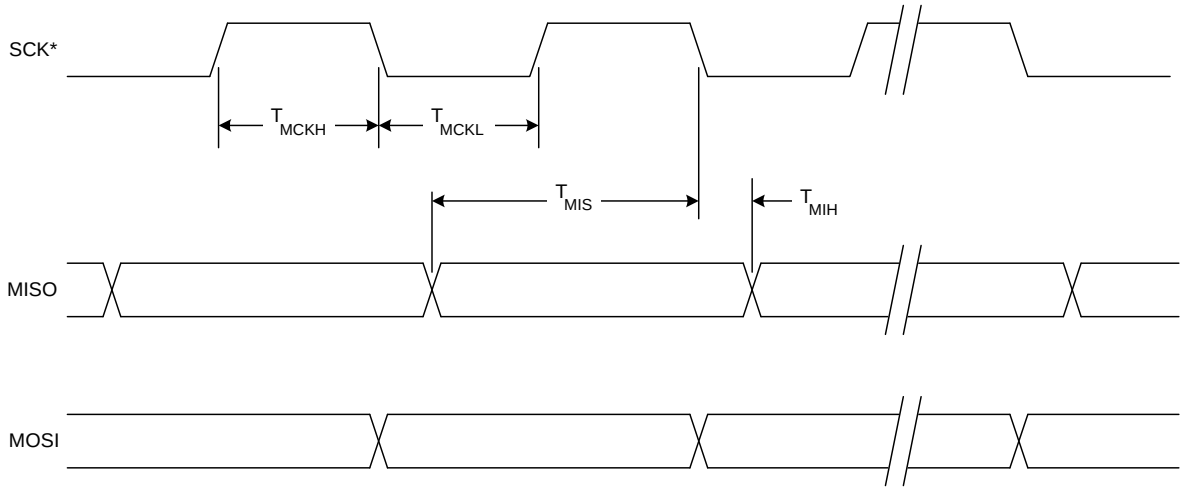
| Bit | Name | Function |
|---|---|---|
| 7:0 | SCR[7:0] | **SPI0 Clock Rate.**<br>These bits determine the frequency of the SCK output when the SPI0 module is configured for master mode operation. The SCK clock frequency is a divided version of the system clock, and is given in the following equation, where *SYSCLK* is the system clock frequency and *SPI0CKR* is the 8-bit value held in the SPI0CKR register.<br>$f_{SCK} = \frac{SYSCLK}{2 \times (SPI0CKR[7:0] + 1)}$<br>for 0 <= SPI0CKR <= 255<br><br>Example: If SYSCLK = 2 MHz and SPI0CKR = 0x04,<br>$f_{SCK} = \frac{2000000}{2 \times (4 + 1)}$<br>$f_{SCK} = 200kHz$ |

## SFR Definition 23.4. SPI0DAT: SPI0 Data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPI0DAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

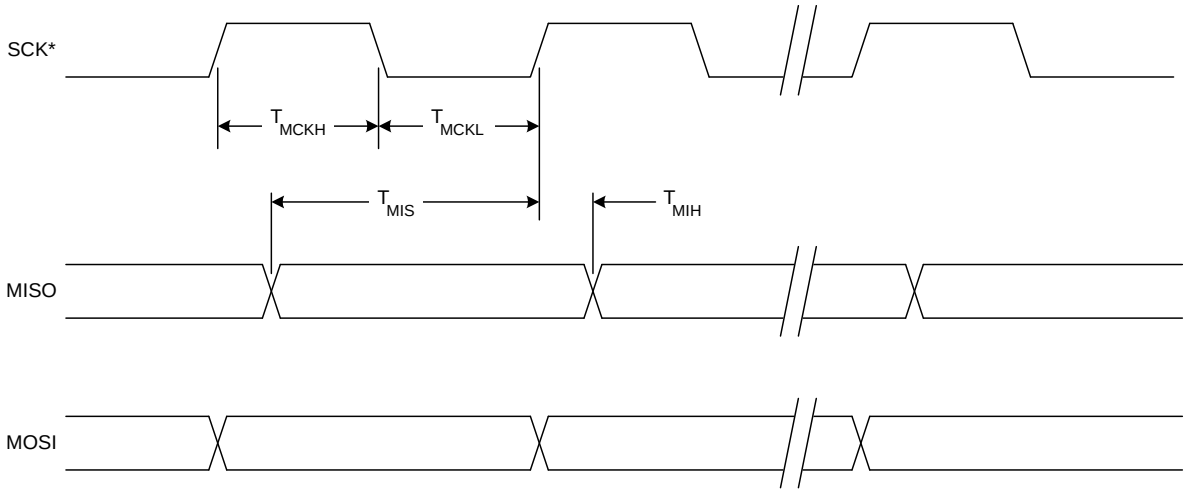
SFR Address = 0xA3

| Bit | Name | Function |
|---|---|---|
| 7:0 | SPI0DAT[7:0] | **SPI0 Transmit and Receive Data.**<br>The SPI0DAT register is used to transmit and receive SPI0 data. Writing data to SPI0DAT places the data into the transmit buffer and initiates a transfer when in Master Mode. A read of SPI0DAT returns the contents of the receive buffer. |



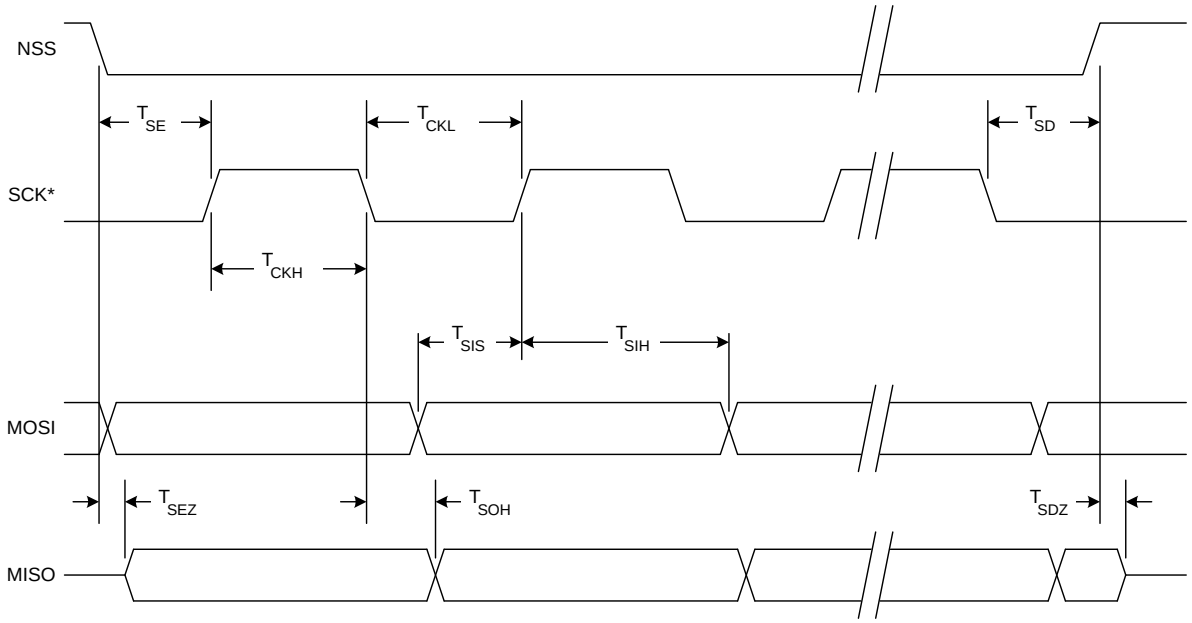

* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

**Figure 23.8. SPI Master Timing (CKPHA = 0)**



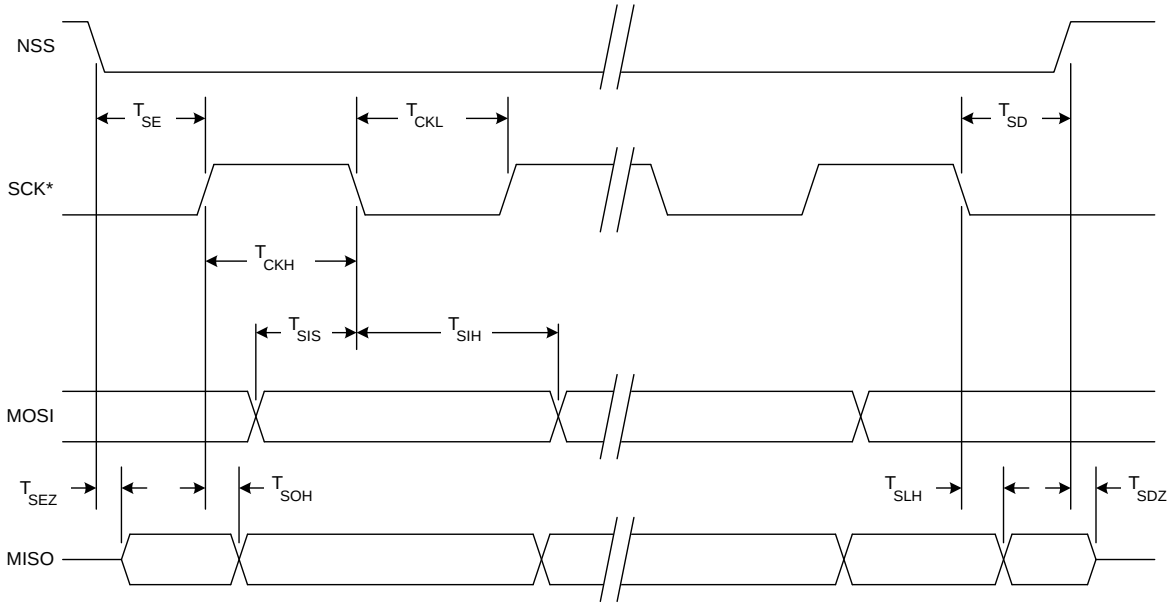

* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

**Figure 23.9. SPI Master Timing (CKPHA = 1)**




* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

**Figure 23.10. SPI Slave Timing (CKPHA = 0)**




* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

**Figure 23.11. SPI Slave Timing (CKPHA = 1)**

**Table 23.1. SPI Slave Timing Parameters**

| Parameter | Description | Min | Max | Units |
|---|---|---|---|---|
| **Master Mode Timing** (See Figure 23.8 and Figure 23.9) | | | | |
| $T_{MCKH}$ | SCK High Time | 1 x $T_{SYSCLK}$ | — | ns |
| $T_{MCKL}$ | SCK Low Time | 1 x $T_{SYSCLK}$ | — | ns |
| $T_{MIS}$ | MISO Valid to SCK Shift Edge | 1 x $T_{SYSCLK}$ + 20 | — | ns |
| $T_{MIH}$ | SCK Shift Edge to MISO Change | 0 | — | ns |
| **Slave Mode Timing** (See Figure 23.10 and Figure 23.11) | | | | |
| $T_{SE}$ | NSS Falling to First SCK Edge | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SD}$ | Last SCK Edge to NSS Rising | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SEZ}$ | NSS Falling to MISO Valid | — | 4 x $T_{SYSCLK}$ | ns |
| $T_{SDZ}$ | NSS Rising to MISO High-Z | — | 4 x $T_{SYSCLK}$ | ns |
| $T_{CKH}$ | SCK High Time | 5 x $T_{SYSCLK}$ | — | ns |
| $T_{CKL}$ | SCK Low Time | 5 x $T_{SYSCLK}$ | — | ns |
| $T_{SIS}$ | MOSI Valid to SCK Sample Edge | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SIH}$ | SCK Sample Edge to MOSI Change | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SOH}$ | SCK Shift Edge to MISO Change | — | 4 x $T_{SYSCLK}$ | ns |
| $T_{SLH}$ | Last SCK Edge to MISO Change (CKPHA = 1 ONLY) | 6 x $T_{SYSCLK}$ | 8 x $T_{SYSCLK}$ | ns |
| **Note:** $T_{SYSCLK}$ is equal to one period of the device system clock (SYSCLK). | | | | |

## 24. Timers

Each MCU includes four counter/timers: two are 16-bit counter/timers compatible with those found in the standard 8051, and two are 16-bit auto-reload timer for use with the ADC, SMBus, or for general purpose use. These timers can be used to measure time intervals, count external events and generate periodic interrupt requests. Timer 0 and Timer 1 are nearly identical and have four primary modes of operation. Timer 2 and Timer 3 offer 16-bit and split 8-bit timer functionality with auto-reload. Additionally, Timer 3 offers the ability to be clocked from the external oscillator while the device is in Suspend mode, and can be used as a wake-up source. This allows for implementation of a very low-power system, including RTC capability.

<table>
<tr><th>Timer 0 and Timer 1 Modes:</th><th>Timer 2 Modes:</th><th>Timer 3 Modes:</th></tr>
<tr><td>13-bit counter/timer</td><td rowspan="2">16-bit timer with auto-reload</td><td rowspan="2">16-bit timer with auto-reload</td></tr>
<tr><td>16-bit counter/timer</td></tr>
<tr><td>8-bit counter/timer with auto-reload</td><td rowspan="2">Two 8-bit timers with auto-reload</td><td rowspan="2">Two 8-bit timers with auto-reload</td></tr>
<tr><td>Two 8-bit counter/timers (Timer 0 only)</td></tr>
</table>

Timers 0 and 1 may be clocked by one of five sources, determined by the Timer Mode Select bits (T1M–T0M) and the Clock Scale bits (SCA1–SCA0). The Clock Scale bits define a pre-scaled clock from which Timer 0 and/or Timer 1 may be clocked (See SFR Definition 24.1 for pre-scaled clock selection).

Timer 0/1 may then be configured to use this pre-scaled clock signal or the system clock. Timer 2 and Timer 3 may be clocked by the system clock, the system clock divided by 12, or the external oscillator clock source divided by 8.

Timer 0 and Timer 1 may also be operated as counters. When functioning as a counter, a counter/timer register is incremented on each high-to-low transition at the selected input pin (T0 or T1). Events with a frequency of up to one-fourth the system clock frequency can be counted. The input signal need not be periodic, but it should be held at a given level for at least two full system clock cycles to ensure the level is properly sampled.

## SFR Definition 24.1. CKCON: Clock Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T3MH | T3ML | T2MH | T2ML | T1M | T0M | SCA[1:0] | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8E

| Bit | Name | Function |
|---|---|---|
| 7 | T3MH | **Timer 3 High Byte Clock Select.**<br>Selects the clock supplied to the Timer 3 high byte (split 8-bit timer mode only).<br>0: Timer 3 high byte uses the clock defined by the T3XCLK bit in TMR3CN.<br>1: Timer 3 high byte uses the system clock. |
| 6 | T3ML | **Timer 3 Low Byte Clock Select.**<br>Selects the clock supplied to Timer 3. Selects the clock supplied to the lower 8-bit timer in split 8-bit timer mode.<br>0: Timer 3 low byte uses the clock defined by the T3XCLK bit in TMR3CN.<br>1: Timer 3 low byte uses the system clock. |
| 5 | T2MH | **Timer 2 High Byte Clock Select.**<br>Selects the clock supplied to the Timer 2 high byte (split 8-bit timer mode only).<br>0: Timer 2 high byte uses the clock defined by the T2XCLK bit in TMR2CN.<br>1: Timer 2 high byte uses the system clock. |
| 4 | T2ML | **Timer 2 Low Byte Clock Select.**<br>Selects the clock supplied to Timer 2. If Timer 2 is configured in split 8-bit timer mode, this bit selects the clock supplied to the lower 8-bit timer.<br>0: Timer 2 low byte uses the clock defined by the T2XCLK bit in TMR2CN.<br>1: Timer 2 low byte uses the system clock. |
| 3 | T1 | **Timer 1 Clock Select.**<br>Selects the clock source supplied to Timer 1. Ignored when C/T1 is set to ’1’.<br>0: Timer 1 uses the clock defined by the prescale bits SCA[1:0].<br>1: Timer 1 uses the system clock. |
| 2 | T0 | **Timer 0 Clock Select.**<br>Selects the clock source supplied to Timer 0. Ignored when C/T0 is set to ’1’.<br>0: Counter/Timer 0 uses the clock defined by the prescale bits SCA[1:0].<br>1: Counter/Timer 0 uses the system clock. |
| 1:0 | SCA[1:0] | **Timer 0/1 Prescale Bits.**<br>These bits control the Timer 0/1 Clock Prescaler:<br>00: System clock divided by 12<br>01: System clock divided by 4<br>10: System clock divided by 48<br>11: External clock divided by 8 (synchronized with the system clock) |

## 24.1. Timer 0 and Timer 1

Each timer is implemented as a 16-bit register accessed as two separate bytes: a low byte (TL0 or TL1) and a high byte (TH0 or TH1). The Counter/Timer Control register (TCON) is used to enable Timer 0 and Timer 1 as well as indicate status. Timer 0 interrupts can be enabled by setting the ET0 bit in the IE register (Section “15.2. Interrupt Register Descriptions” on page 84); Timer 1 interrupts can be enabled by setting the ET1 bit in the IE register (Section “15.2. Interrupt Register Descriptions” on page 84). Both counter/timers operate in one of four primary modes selected by setting the Mode Select bits T1M1–T0M0 in the Counter/Timer Mode register (TMOD). Each timer can be configured independently. Each operating mode is described below.

### 24.1.1. Mode 0: 13-bit Counter/Timer

Timer 0 and Timer 1 operate as 13-bit counter/timers in Mode 0. The following describes the configuration and operation of Timer 0. However, both timers operate identically, and Timer 1 is configured in the same manner as described for Timer 0.

The TH0 register holds the eight MSBs of the 13-bit counter/timer. TL0 holds the five LSBs in bit positions TL0.4–TL0.0. The three upper bits of TL0 (TL0.7–TL0.5) are indeterminate and should be masked out or ignored when reading. As the 13-bit timer register increments and overflows from 0x1FFF (all ones) to 0x0000, the timer overflow flag TF0 in TCON is set and an interrupt will occur if Timer 0 interrupts are enabled.

The C/T0 bit in the TMOD register selects the counter/timer's clock source. When C/T0 is set to logic 1, high-to-low transitions at the selected Timer 0 input pin (T0) increment the timer register (Refer to Section “20.3. Priority Crossbar Decoder” on page 124 for information on selecting and configuring external I/O pins). Clearing C/T selects the clock defined by the T0M bit in register CKCON. When T0M is set, Timer 0 is clocked by the system clock. When T0M is cleared, Timer 0 is clocked by the source selected by the Clock Scale bits in CKCON (see SFR Definition 24.1).

Setting the TR0 bit (TCON.4) enables the timer when either GATE0 in the TMOD register is logic 0 or the input signal INT0 is active as defined by bit IN0PL in register IT01CF (see SFR Definition 15.5). Setting GATE0 to 1 allows the timer to be controlled by the external input signal INT0 (see Section “15.2. Interrupt Register Descriptions” on page 84), facilitating pulse width measurements

| TR0 | GATE0 | INT0 | Counter/Timer |
|---|---|---|---|
| 0 | X | X | Disabled |
| 1 | 0 | X | Enabled |
| 1 | 1 | 0 | Disabled |
| 1 | 1 | 1 | Enabled |
| **Note:** X = Don't Care | | | |

Setting TR0 does not force the timer to reset. The timer registers should be loaded with the desired initial value before the timer is enabled.

TL1 and TH1 form the 13-bit register for Timer 1 in the same manner as described above for TL0 and TH0. Timer 1 is configured and controlled using the relevant TCON and TMOD bits just as with Timer 0. The input signal INT0 is used with Timer 1; the /INT1 polarity is defined by bit IN1PL in register IT01CF (see SFR Definition 15.5).

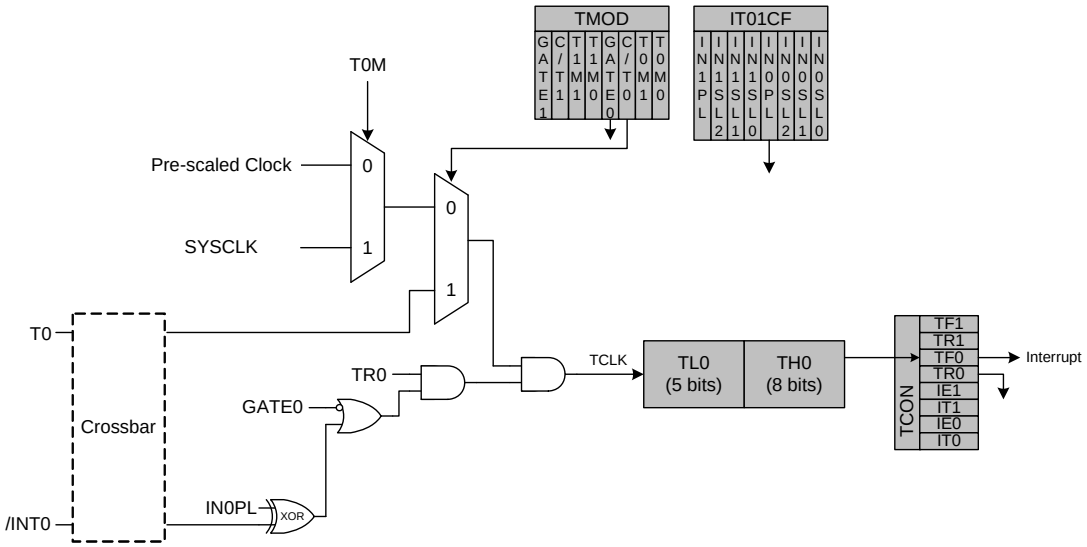



**Figure 24.1. T0 Mode 0 Block Diagram**

### 24.1.2. Mode 1: 16-bit Counter/Timer

Mode 1 operation is the same as Mode 0, except that the counter/timer registers use all 16 bits. The counter/timers are enabled and configured in Mode 1 in the same manner as for Mode 0.

### 24.1.3. Mode 2: 8-bit Counter/Timer with Auto-Reload

Mode 2 configures Timer 0 and Timer 1 to operate as 8-bit counter/timers with automatic reload of the start value. TL0 holds the count and TH0 holds the reload value. When the counter in TL0 overflows from all ones to 0x00, the timer overflow flag TF0 in the TCON register is set and the counter in TL0 is reloaded from TH0. If Timer 0 interrupts are enabled, an interrupt will occur when the TF0 flag is set. The reload value in TH0 is not changed. TL0 must be initialized to the desired value before enabling the timer for the first count to be correct. When in Mode 2, Timer 1 operates identically to Timer 0.

Both counter/timers are enabled and configured in Mode 2 in the same manner as Mode 0. Setting the TR0 bit (TCON.4) enables the timer when either GATE0 in the TMOD register is logic 0 or when the input signal INT0 is active as defined by bit IN0PL in register IT01CF (see Section “15.3. External Interrupts /INT0 and /INT1” on page 89 for details on the external input signals INT0 and INT1).

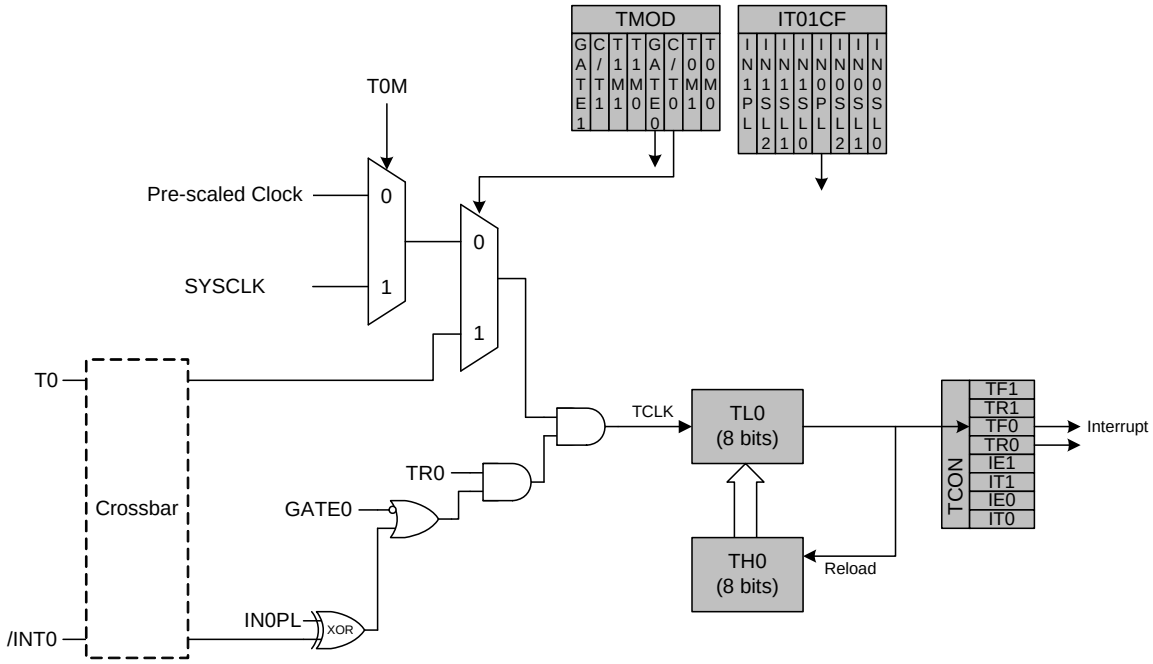



**Figure 24.2. T0 Mode 2 Block Diagram**

### 24.1.4. Mode 3: Two 8-bit Counter/Timers (Timer 0 Only)

In Mode 3, Timer 0 is configured as two separate 8-bit counter/timers held in TL0 and TH0. The counter/timer in TL0 is controlled using the Timer 0 control/status bits in TCON and TMOD: TR0, C/T0, GATE0 and TF0. TL0 can use either the system clock or an external input signal as its timebase. The TH0 register is restricted to a timer function sourced by the system clock or prescaled clock. TH0 is enabled using the Timer 1 run control bit TR1. TH0 sets the Timer 1 overflow flag TF1 on overflow and thus controls the Timer 1 interrupt.

Timer 1 is inactive in Mode 3. When Timer 0 is operating in Mode 3, Timer 1 can be operated in Modes 0, 1 or 2, but cannot be clocked by external signals nor set the TF1 flag and generate an interrupt. However, the Timer 1 overflow can be used to generate baud rates for the SMBus and/or UART, and/or initiate ADC conversions. While Timer 0 is operating in Mode 3, Timer 1 run control is handled through its mode settings. To run Timer 1 while Timer 0 is in Mode 3, set the Timer 1 Mode as 0, 1, or 2. To disable Timer 1, configure it for Mode 3.

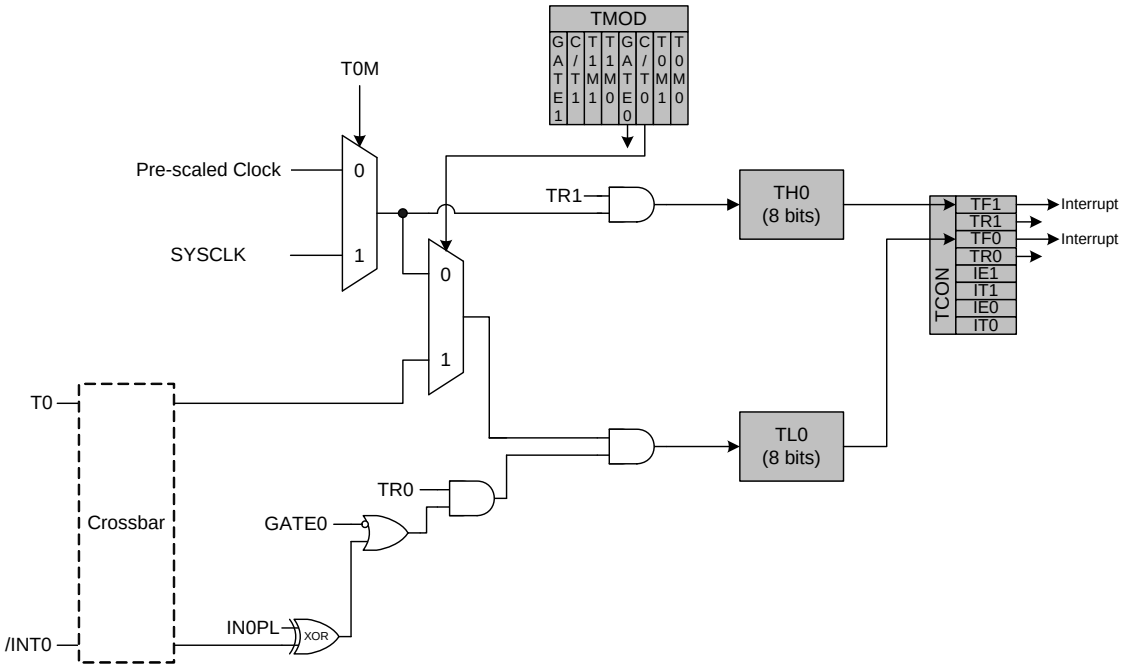



**Figure 24.3. T0 Mode 3 Block Diagram**

## SFR Definition 24.2. TCON: Timer Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x88; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | TF1 | **Timer 1 Overflow Flag.**<br>Set to 1 by hardware when Timer 1 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine. |
| 6 | TR1 | **Timer 1 Run Control.**<br>Timer 1 is enabled by setting this bit to 1. |
| 5 | TF0 | **Timer 0 Overflow Flag.**<br>Set to 1 by hardware when Timer 0 overflows. This flag can be cleared by software but is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine. |
| 4 | TR0 | **Timer 0 Run Control.**<br>Timer 0 is enabled by setting this bit to 1. |
| 3 | IE1 | **External Interrupt 1.**<br>This flag is set by hardware when an edge/level of type defined by IT1 is detected. It can be cleared by software but is automatically cleared when the CPU vectors to the External Interrupt 1 service routine in edge-triggered mode. |
| 2 | IT1 | **Interrupt 1 Type Select.**<br>This bit selects whether the configured /INT1 interrupt will be edge or level sensitive. /INT1 is configured active low or high by the IN1PL bit in the IT01CF register (see SFR Definition 15.5).<br>0: /INT1 is level triggered.<br>1: /INT1 is edge triggered. |
| 1 | IE0 | **External Interrupt 0.**<br>This flag is set by hardware when an edge/level of type defined by IT1 is detected. It can be cleared by software but is automatically cleared when the CPU vectors to the External Interrupt 0 service routine in edge-triggered mode. |
| 0 | IT0 | **Interrupt 0 Type Select.**<br>This bit selects whether the configured $\overline{\text{INT0}}$ interrupt will be edge or level sensitive. $\overline{\text{INT0}}$ is configured active low or high by the IN0PL bit in register IT01CF (see SFR Definition 15.5).<br>0: $\overline{\text{INT0}}$ is level triggered.<br>1: $\overline{\text{INT0}}$ is edge triggered. |

## SFR Definition 24.3. TMOD: Timer Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | GATE1 | C/T1 | T1M[1:0] | | GATE0 | C/T0 | T0M[1:0] | |
| Type | R/W | R/W | R/W | | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x89

| Bit | Name | Function |
|---|---|---|
| 7 | GATE1 | **Timer 1 Gate Control.**<br>0: Timer 1 enabled when TR1 = 1 irrespective of $\overline{INT1}$ logic level.<br>1: Timer 1 enabled only when TR1 = 1 AND $\overline{INT1}$ is active as defined by bit IN1PL in register IT01CF (see SFR Definition 15.5). |
| 6 | C/T1 | **Counter/Timer 1 Select.**<br>0: Timer: Timer 1 incremented by clock defined by T1M bit in register CKCON.<br>1: Counter: Timer 1 incremented by high-to-low transitions on external pin (T1). |
| 5:4 | T1M[1:0] | **Timer 1 Mode Select.**<br>These bits select the Timer 1 operation mode.<br>00: Mode 0, 13-bit Counter/Timer<br>01: Mode 1, 16-bit Counter/Timer<br>10: Mode 2, 8-bit Counter/Timer with Auto-Reload<br>11: Mode 3, Timer 1 Inactive |
| 3 | GATE0 | **Timer 0 Gate Control.**<br>0: Timer 0 enabled when TR0 = 1 irrespective of $\overline{INT0}$ logic level.<br>1: Timer 0 enabled only when TR0 = 1 AND $\overline{INT0}$ is active as defined by bit IN0PL in register IT01CF (see SFR Definition 15.5). |
| 2 | C/T0 | **Counter/Timer 0 Select.**<br>0: Timer: Timer 0 incremented by clock defined by T0M bit in register CKCON.<br>1: Counter: Timer 0 incremented by high-to-low transitions on external pin (T0). |
| 1:0 | T0M[1:0] | **Timer 0 Mode Select.**<br>These bits select the Timer 0 operation mode.<br>00: Mode 0, 13-bit Counter/Timer<br>01: Mode 1, 16-bit Counter/Timer<br>10: Mode 2, 8-bit Counter/Timer with Auto-Reload<br>11: Mode 3, Two 8-bit Counter/Timers |

## SFR Definition 24.4. TL0: Timer 0 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TL0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8A

| Bit | Name | Function |
|---|---|---|
| 7:0 | TL0[7:0] | **Timer 0 Low Byte.**<br>The TL0 register is the low byte of the 16-bit Timer 0. |

## SFR Definition 24.5. TL1: Timer 1 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TL1[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8B

| Bit | Name | Function |
|---|---|---|
| 7:0 | TL1[7:0] | **Timer 1 Low Byte.**<br>The TL1 register is the low byte of the 16-bit Timer 1. |

## SFR Definition 24.6. TH0: Timer 0 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TH0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8C

| Bit | Name | Function |
|---|---|---|
| 7:0 | TH0[7:0] | **Timer 0 High Byte.**<br>The TH0 register is the high byte of the 16-bit Timer 0. |

## SFR Definition 24.7. TH1: Timer 1 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TH1[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8D

| Bit | Name | Function |
|---|---|---|
| 7:0 | TH1[7:0] | **Timer 1 High Byte.**<br>The TH1 register is the high byte of the 16-bit Timer 1. |

## 24.2. Timer 2

Timer 2 is a 16-bit timer formed by two 8-bit SFRs: TMR2L (low byte) and TMR2H (high byte). Timer 2 may operate in 16-bit auto-reload mode or (split) 8-bit auto-reload mode. The T2SPLIT bit (TMR2CN.3) defines the Timer 2 operation mode.

Timer 2 may be clocked by the system clock, the system clock divided by 12, or the external oscillator source divided by 8. The external clock mode is ideal for real-time clock (RTC) functionality, where the internal oscillator drives the system clock while Timer 2 (and/or the PCA) is clocked by an external precision oscillator. Note that the external oscillator source divided by 8 is synchronized with the system clock.

### 24.2.1. 16-bit Timer with Auto-Reload

When T2SPLIT (TMR2CN.3) is zero, Timer 2 operates as a 16-bit timer with auto-reload. Timer 2 can be clocked by SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the Timer 2 reload registers (TMR2RLH and TMR2RLL) is loaded into the Timer 2 register as shown in Figure 24.4, and the Timer 2 High Byte Overflow Flag (TMR2CN.7) is set. If Timer 2 interrupts are enabled (if IE.5 is set), an interrupt will be generated on each Timer 2 overflow. Additionally, if Timer 2 interrupts are enabled and the TF2LEN bit is set (TMR2CN.5), an interrupt will be generated each time the lower 8 bits (TMR2L) overflow from 0xFF to 0x00.

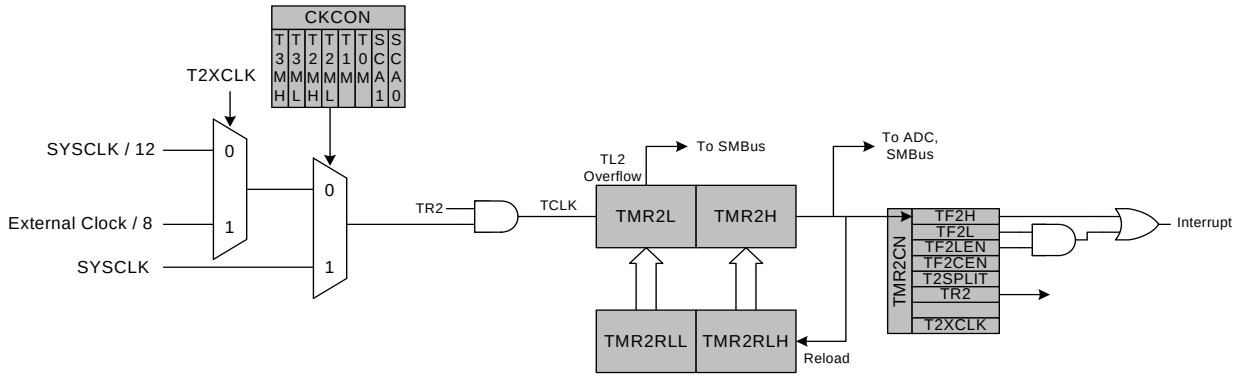



**Figure 24.4. Timer 2 16-Bit Mode Block Diagram**

### 24.2.2. 8-bit Timers with Auto-Reload

When T2SPLIT is set, Timer 2 operates as two 8-bit timers (TMR2H and TMR2L). Both 8-bit timers operate in auto-reload mode as shown in Figure 24.5. TMR2RLL holds the reload value for TMR2L; TMR2RLH holds the reload value for TMR2H. The TR2 bit in TMR2CN handles the run control for TMR2H. TMR2L is always running when configured for 8-bit Mode.

Each 8-bit timer may be configured to use SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. The Timer 2 Clock Select bits (T2MH and T2ML in CKCON) select either SYSCLK or the clock defined by the Timer 2 External Clock Select bit (T2XCLK in TMR2CN), as follows:

| T2MH | T2XCLK | TMR2H Clock Source |
|---|---|---|
| 0 | 0 | SYSCLK / 12 |
| 0 | 1 | External Clock / 8 |
| 1 | X | SYSCLK |

| T2ML | T2XCLK | TMR2L Clock Source |
|---|---|---|
| 0 | 0 | SYSCLK / 12 |
| 0 | 1 | External Clock / 8 |
| 1 | X | SYSCLK |

The TF2H bit is set when TMR2H overflows from 0xFF to 0x00; the TF2L bit is set when TMR2L overflows from 0xFF to 0x00. When Timer 2 interrupts are enabled (IE.5), an interrupt is generated each time TMR2H overflows. If Timer 2 interrupts are enabled and TF2LEN (TMR2CN.5) is set, an interrupt is generated each time either TMR2L or TMR2H overflows. When TF2LEN is enabled, software must check the TF2H and TF2L flags to determine the source of the Timer 2 interrupt. The TF2H and TF2L interrupt flags are not cleared by hardware and must be manually cleared by software.

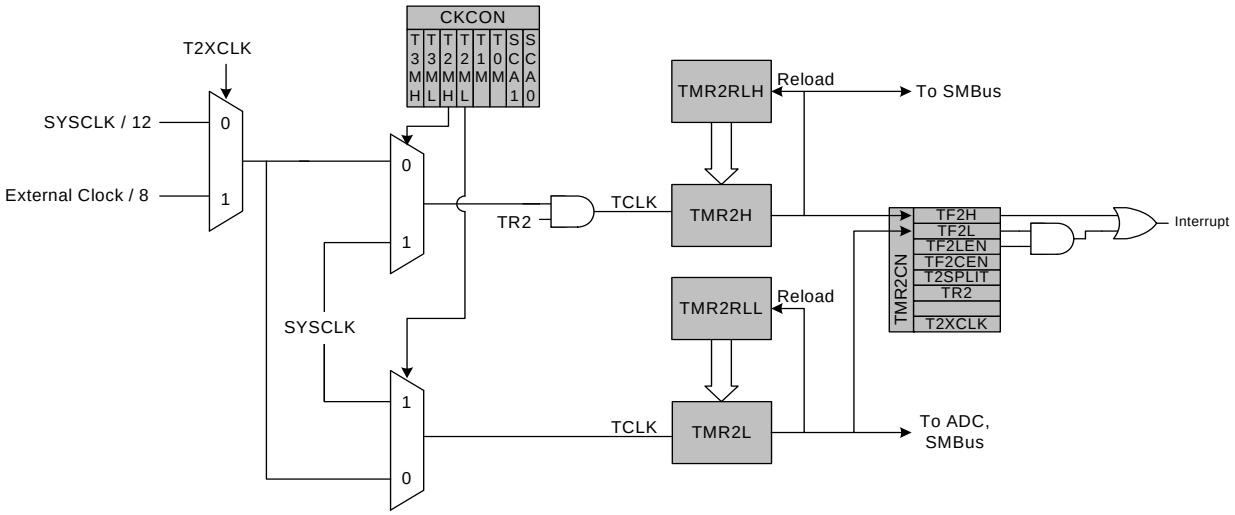



**Figure 24.5. Timer 2 8-Bit Mode Block Diagram**

### 24.2.3. Low-Frequency Oscillator (LFO) Capture Mode

The Low-Frequency Oscillator Capture Mode allows the LFO clock to be measured against the system clock or an external oscillator source. Timer 2 can be clocked from the system clock, the system clock divided by 12, or the external oscillator divided by 8, depending on the T2ML (CKCON.4), and T2XCLK settings.

Setting TF2CEN to 1 enables the LFO Capture Mode for Timer 2. In this mode, T2SPLIT should be set to 0, as the full 16-bit timer is used. Upon a falling edge of the low-frequency oscillator, the contents of Timer 2 (TMR2H:TMR2L) are loaded into the Timer 2 reload registers (TMR2RLH:TMR2RLL) and the TF2H flag is set. By recording the difference between two successive timer capture values, the LFO clock frequency can be determined with respect to the Timer 2 clock. The Timer 2 clock should be much faster than the LFO to achieve an accurate reading.

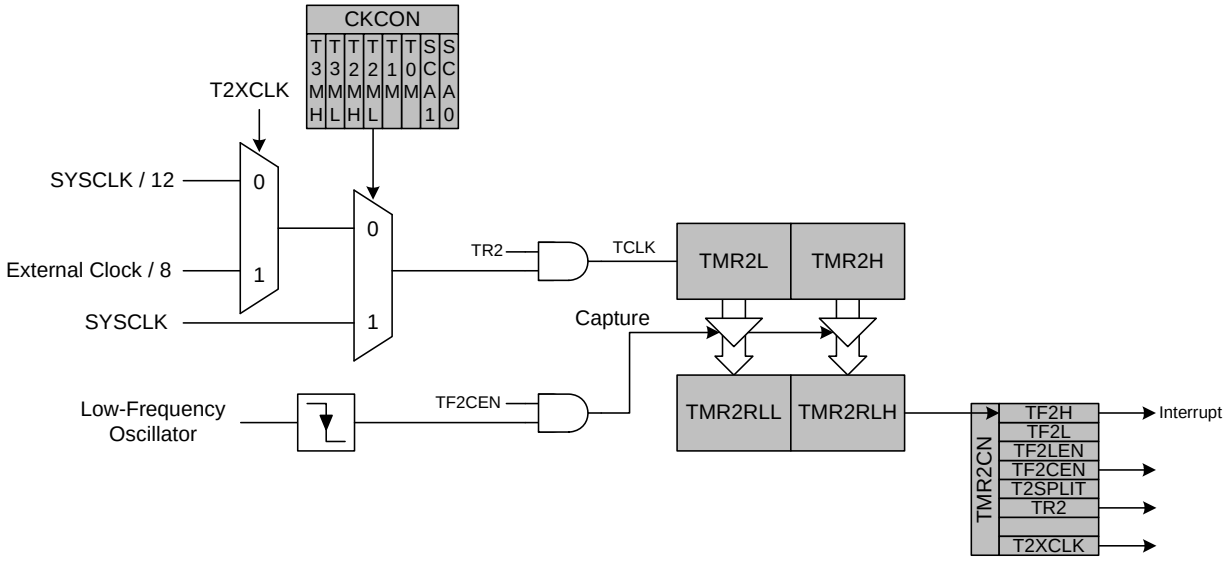



**Figure 24.6. Timer 2 Low-Frequency Oscillation Capture Mode Block Diagram**

## SFR Definition 24.8. TMR2CN: Timer 2 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF2H | TF2L | TF2LEN | TF2CEN | T2SPLIT | TR2 | | T2XCLK |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC8; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | TF2H | **Timer 2 High Byte Overflow Flag.**<br>Set by hardware when the Timer 2 high byte overflows from 0xFF to 0x00. In 16 bit mode, this will occur when Timer 2 overflows from 0xFFFF to 0x0000. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 interrupt service routine. This bit is not automatically cleared by hardware. |
| 6 | TF2L | **Timer 2 Low Byte Overflow Flag.**<br>Set by hardware when the Timer 2 low byte overflows from 0xFF to 0x00. TF2L will be set when the low byte overflows regardless of the Timer 2 mode. This bit is not automatically cleared by hardware. |
| 5 | TF2LEN | **Timer 2 Low Byte Interrupt Enable.**<br>When set to 1, this bit enables Timer 2 Low Byte interrupts. If Timer 2 interrupts are also enabled, an interrupt will be generated when the low byte of Timer 2 overflows. |
| 4 | TF2CEN | **Timer 2 Low-Frequency Oscillator Capture Enable.**<br>When set to 1, this bit enables Timer 2 Low-Frequency Oscillator Capture Mode. If TF2CEN is set and Timer 2 interrupts are enabled, an interrupt will be generated on a falling edge of the low-frequency oscillator output, and the current 16-bit timer value in TMR2H:TMR2L will be copied to TMR2RLH:TMR2RLL. |
| 3 | T2SPLIT | **Timer 2 Split Mode Enable.**<br>When this bit is set, Timer 2 operates as two 8-bit timers with auto-reload.<br>0: Timer 2 operates in 16-bit auto-reload mode.<br>1: Timer 2 operates as two 8-bit auto-reload timers. |
| 2 | TR2 | **Timer 2 Run Control.**<br>Timer 2 is enabled by setting this bit to 1. In 8-bit mode, this bit enables/disables TMR2H only; TMR2L is always enabled in split mode. |
| 1 | Unused | Unused. Read = 0b; Write = Don't Care |
| 0 | T2XCLK | **Timer 2 External Clock Select.**<br>This bit selects the external clock source for Timer 2. If Timer 2 is in 8-bit mode, this bit selects the external oscillator clock source for both timer bytes. However, the Timer 2 Clock Select bits (T2MH and T2ML in register CKCON) may still be used to select between the external clock and the system clock for either timer.<br>0: Timer 2 clock is the system clock divided by 12.<br>1: Timer 2 clock is the external clock divided by 8 (synchronized with SYSCLK). |

## SFR Definition 24.9. TMR2RLL: Timer 2 Reload Register Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2RLL[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xCA

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR2RLL[7:0] | **Timer 2 Reload Register Low Byte.**<br>TMR2RLL holds the low byte of the reload value for Timer 2. |

## SFR Definition 24.10. TMR2RLH: Timer 2 Reload Register High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2RLH[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xCB

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR2RLH[7:0] | **Timer 2 Reload Register High Byte.**<br>TMR2RLH holds the high byte of the reload value for Timer 2. |

## SFR Definition 24.11. TMR2L: Timer 2 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2L[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xCC

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR2L[7:0] | **Timer 2 Low Byte.**<br>In 16-bit mode, the TMR2L register contains the low byte of the 16-bit Timer 2. In 8-bit mode, TMR2L contains the 8-bit low byte timer value. |

## SFR Definition 24.12. TMR2H Timer 2 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2H[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xCD

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR2H[7:0] | **Timer 2 Low Byte.**<br>In 16-bit mode, the TMR2H register contains the high byte of the 16-bit Timer 2. In 8-bit mode, TMR2H contains the 8-bit high byte timer value. |

## 24.3. Timer 3

Timer 3 is a 16-bit timer formed by two 8-bit SFRs: TMR3L (low byte) and TMR3H (high byte). Timer 3 may operate in 16-bit auto-reload mode or (split) 8-bit auto-reload mode. The T3SPLIT bit (TMR3CN.3) defines the Timer 3 operation mode.

Timer 3 may be clocked by the system clock, the system clock divided by 12, the external oscillator source divided by 8, or the internal low-frequency oscillator divided by 8. The external clock mode is ideal for real-time clock (RTC) functionality, where the internal high-frequency oscillator drives the system clock while Timer 3 is clocked by an external oscillator source. Note that the external oscillator source divided by 8 and the LFO source divided by 8 are synchronized with the system clock when in all operating modes except suspend. When the internal oscillator is placed in suspend mode, The external clock/8 signal or the LFO/8 output can directly drive the timer. This allows the use of an external clock or the LFO to wake up the device from suspend mode. The timer will continue to run in suspend mode and count up. When the timer overflow occurs, the device will wake from suspend mode, and begin executing code again. The timer value may be set prior to entering suspend, to overflow in the desired amount of time (number of clocks) to wake the device. If a wake-up source other than the timer wakes the device from suspend mode, it may take up to three timer clocks before the timer registers can be read or written. During this time, the STSYNC bit in register OSCICN will be set to 1, to indicate that it is not safe to read or write the timer registers.

**Important Note: In internal LFO/8 mode, the divider for the internal LFO must be set to 1 for proper functionality. The timer will not operate if the LFO divider is not set to 1.**

### 24.3.1. 16-bit Timer with Auto-Reload

When T3SPLIT (TMR3CN.3) is zero, Timer 3 operates as a 16-bit timer with auto-reload. Timer 3 can be clocked by SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the Timer 3 reload registers (TMR3RLH and TMR3RLL) is loaded into the Timer 3 register as shown in Figure 24.7, and the Timer 3 High Byte Overflow Flag (TMR3CN.7) is set. If Timer 3 interrupts are enabled (if EIE1.7 is set), an interrupt will be generated on each Timer 3 overflow. Additionally, if Timer 3 interrupts are enabled and the TF3LEN bit is set (TMR3CN.5), an interrupt will be generated each time the lower 8 bits (TMR3L) overflow from 0xFF to 0x00.

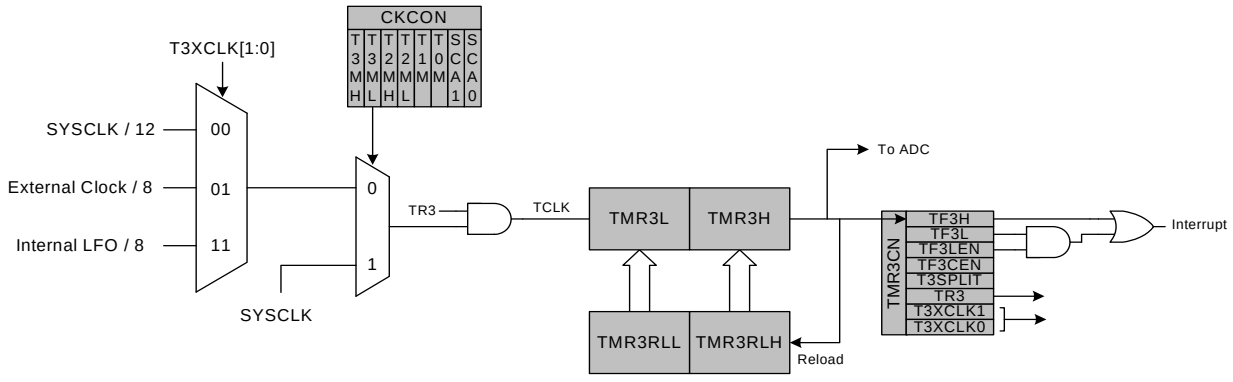



**Figure 24.7. Timer 3 16-Bit Mode Block Diagram**

### 24.3.2. 8-bit Timers with Auto-Reload

When T3SPLIT is set, Timer 3 operates as two 8-bit timers (TMR3H and TMR3L). Both 8-bit timers operate in auto-reload mode as shown in Figure 24.8. TMR3RLL holds the reload value for TMR3L; TMR3RLH holds the reload value for TMR3H. The TR3 bit in TMR3CN handles the run control for TMR3H. TMR3L is always running when configured for 8-bit Mode.

Each 8-bit timer may be configured to use SYSCLK, SYSCLK divided by 12, the external oscillator clock source divided by 8, or the internal Low-frequency Oscillator. The Timer 3 Clock Select bits (T3MH and T3ML in CKCON) select either SYSCLK or the clock defined by the Timer 3 External Clock Select bits (T3XCLK[1:0] in TMR3CN), as follows:

| T3MH | T3XCLK[1:0] | TMR3H Clock Source |
|---|---|---|
| 0 | 00 | SYSCLK / 12 |
| 0 | 01 | External Clock / 8 |
| 0 | 10 | Reserved |
| 0 | 11 | Internal LFO |
| 1 | X | SYSCLK |

| T3ML | T3XCLK[1:0] | TMR3L Clock Source |
|---|---|---|
| 0 | 00 | SYSCLK / 12 |
| 0 | 01 | External Clock / 8 |
| 0 | 10 | Reserved |
| 0 | 11 | Internal LFO |
| 1 | X | SYSCLK |

The TF3H bit is set when TMR3H overflows from 0xFF to 0x00; the TF3L bit is set when TMR3L overflows from 0xFF to 0x00. When Timer 3 interrupts are enabled, an interrupt is generated each time TMR3H overflows. If Timer 3 interrupts are enabled and TF3LEN (TMR3CN.5) is set, an interrupt is generated each time either TMR3L or TMR3H overflows. When TF3LEN is enabled, software must check the TF3H and TF3L flags to determine the source of the Timer 3 interrupt. The TF3H and TF3L interrupt flags are not cleared by hardware and must be manually cleared by software.

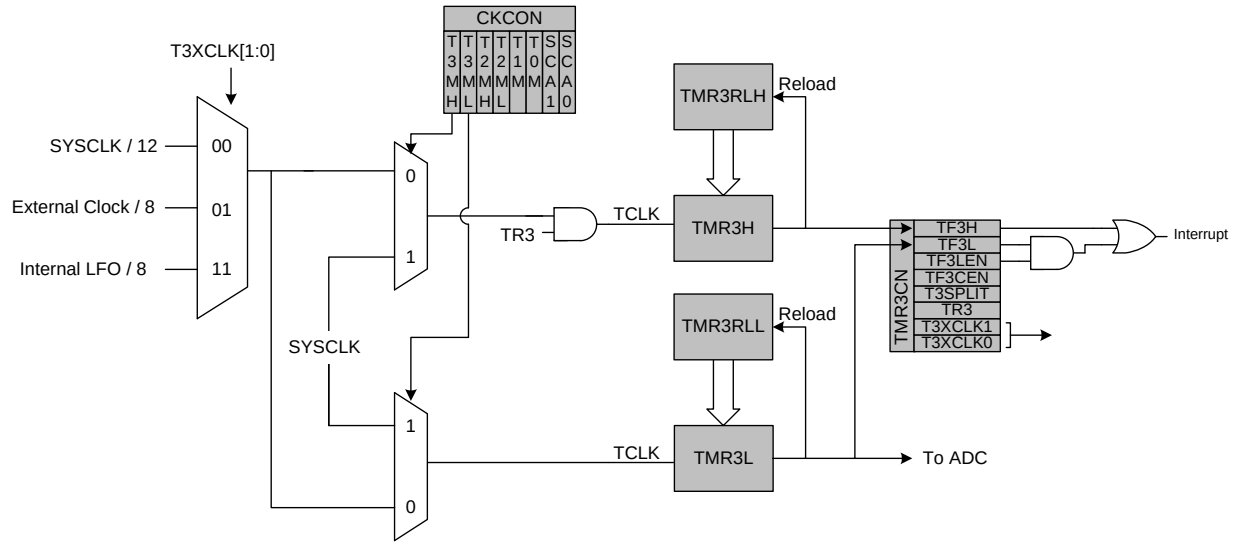



**Figure 24.8. Timer 3 8-Bit Mode Block Diagram**

### 24.3.3. Low-Frequency Oscillator (LFO) Capture Mode

The Low-Frequency Oscillator Capture Mode allows the LFO clock to be measured against the system clock or an external oscillator source. Timer 3 can be clocked from the system clock, the system clock divided by 12, or the external oscillator divided by 8, depending on the T3ML (CKCON.6), and T3XCLK[1:0] settings.

Setting TF3CEN to 1 enables the LFO Capture Mode for Timer 3. In this mode, T3SPLIT should be set to 0, as the full 16-bit timer is used. Upon a falling edge of the low-frequency oscillator, the contents of Timer 3 (TMR3H:TMR3L) are loaded into the Timer 3 reload registers (TMR3RLH:TMR3RLL) and the TF3H flag is set. By recording the difference between two successive timer capture values, the LFO clock frequency can be determined with respect to the Timer 3 clock. The Timer 3 clock should be much faster than the LFO to achieve an accurate reading. This means that the LFO/8 should not be selected as the timer clock source in this mode.

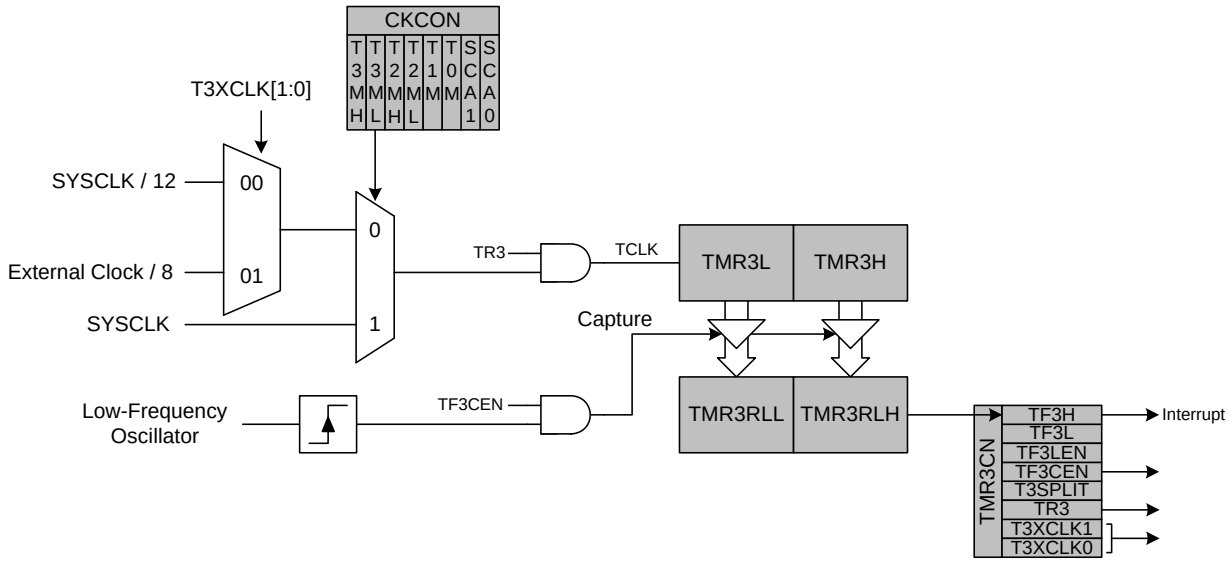



**Figure 24.9. Timer 3 Low-Frequency Oscillation Capture Mode Block Diagram**

## SFR Definition 24.13. TMR3CN: Timer 3 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF3H | TF3L | TF3LEN | TF3CEN | T3SPLIT | TR3 | T3XCLK[1:0] | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x91

| Bit | Name | Function |
|---|---|---|
| 7 | TF3H | **Timer 3 High Byte Overflow Flag.**<br>Set by hardware when the Timer 3 high byte overflows from 0xFF to 0x00. In 16 bit mode, this will occur when Timer 3 overflows from 0xFFFF to 0x0000. When the Timer 3 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 3 interrupt service routine. This bit is not automatically cleared by hardware. |
| 6 | TF3L | **Timer 3 Low Byte Overflow Flag.**<br>Set by hardware when the Timer 3 low byte overflows from 0xFF to 0x00. TF3L will be set when the low byte overflows regardless of the Timer 3 mode. This bit is not automatically cleared by hardware. |
| 5 | TF3LEN | **Timer 3 Low Byte Interrupt Enable.**<br>When set to 1, this bit enables Timer 3 Low Byte interrupts. If Timer 3 interrupts are also enabled, an interrupt will be generated when the low byte of Timer 3 overflows. |
| 4 | TF3CEN | **Timer 3 Low-Frequency Oscillator Capture Enable.**<br>When set to 1, this bit enables Timer 3 Low-Frequency Oscillator Capture Mode. If TF3CEN is set and Timer 3 interrupts are enabled, an interrupt will be generated on a falling edge of the low-frequency oscillator output, and the current 16-bit timer value in TMR3H:TMR3L will be copied to TMR3RLH:TMR3RLL. |
| 3 | T3SPLIT | **Timer 3 Split Mode Enable.**<br>When this bit is set, Timer 3 operates as two 8-bit timers with auto-reload.<br>0: Timer 3 operates in 16-bit auto-reload mode.<br>1: Timer 3 operates as two 8-bit auto-reload timers. |
| 2 | TR3 | **Timer 3 Run Control.**<br>Timer 3 is enabled by setting this bit to 1. In 8-bit mode, this bit enables/disables TMR3H only; TMR3L is always enabled in split mode. |
| 1:0 | T3XCLK[1:0] | **Timer 3 External Clock Select.**<br>This bit selects the “external” clock source for Timer 3. If Timer 3 is in 8-bit mode, this bit selects the external oscillator clock source for both timer bytes. However, the Timer 3 Clock Select bits (T3MH and T3ML in register CKCON) may still be used to select between the external clock and the system clock for either timer.<br>00: System clock divided by 12.<br>01: External clock divided by 8 (synchronized with SYSCLK when not in suspend).<br>10: Reserved.<br>11: Internal LFO/8 (synchronized with SYSCLK when not in suspend). |

## SFR Definition 24.14. TMR3RLL: Timer 3 Reload Register Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3RLL[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x92

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR3RLL[7:0] | **Timer 3 Reload Register Low Byte.**<br>TMR3RLL holds the low byte of the reload value for Timer 3. |

## SFR Definition 24.15. TMR3RLH: Timer 3 Reload Register High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3RLH[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x93

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR3RLH[7:0] | **Timer 3 Reload Register High Byte.**<br>TMR3RLH holds the high byte of the reload value for Timer 3. |

## SFR Definition 24.16. TMR3L: Timer 3 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3L[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x94

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR3L[7:0] | **Timer 3 Low Byte.**<br>In 16-bit mode, the TMR3L register contains the low byte of the 16-bit Timer 3. In 8-bit mode, TMR3L contains the 8-bit low byte timer value. |

## SFR Definition 24.17. TMR3H Timer 3 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3H[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x95

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR3H[7:0] | **Timer 3 High Byte.**<br>In 16-bit mode, the TMR3H register contains the high byte of the 16-bit Timer 3. In 8-bit mode, TMR3H contains the 8-bit high byte timer value. |

## 25. Programmable Counter Array

The Programmable Counter Array (PCA0) provides enhanced timer functionality while requiring less CPU intervention than the standard 8051 counter/timers. The PCA consists of a dedicated 16-bit counter/timer and three 16-bit capture/compare modules. Each capture/compare module has its own associated I/O line (CEXn) which is routed through the Crossbar to Port I/O when enabled. The counter/timer is driven by a programmable timebase that can select between six sources: system clock, system clock divided by four, system clock divided by twelve, the external oscillator clock source divided by 8, Timer 0 overflows, or an external clock signal on the ECI input pin. Each capture/compare module may be configured to operate independently in one of six modes: Edge-Triggered Capture, Software Timer, High-Speed Output, Frequency Output, 8 to 11-Bit PWM, or 16-Bit PWM (each mode is described in Section "25.3. Capture/Compare Modules" on page 205). The external oscillator clock option is ideal for real-time clock (RTC) functionality, allowing the PCA to be clocked by a precision external oscillator while the internal oscillator drives the system clock. The PCA is configured and controlled through the system controller's Special Function Registers. The PCA block diagram is shown in Figure 25.1

**Important Note:** The PCA Module 2 may be used as a watchdog timer (WDT), and is enabled in this mode following a system reset. **Access to certain PCA registers is restricted while WDT mode is enabled**. See Section 25.4 for details.

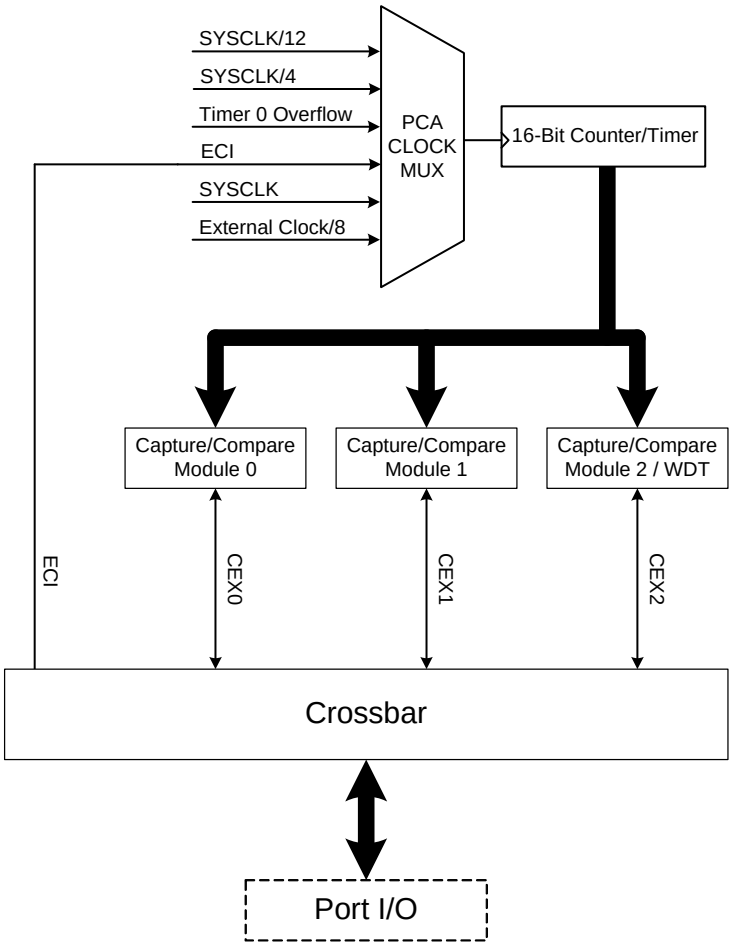



**Figure 25.1. PCA Block Diagram**

## 25.1. PCA Counter/Timer

The 16-bit PCA counter/timer consists of two 8-bit SFRs: PCA0L and PCA0H. PCA0H is the high byte (MSB) of the 16-bit counter/timer and PCA0L is the low byte (LSB). Reading PCA0L automatically latches the value of PCA0H into a “snapshot” register; the following PCA0H read accesses this “snapshot” register. **Reading the PCA0L Register first guarantees an accurate reading of the entire 16-bit PCA0 counter.** Reading PCA0H or PCA0L does not disturb the counter operation. The CPS2–CPS0 bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 25.1.

When the counter/timer overflows from 0xFFFF to 0x0000, the Counter Overflow Flag (CF) in PCA0MD is set to logic 1 and an interrupt request is generated if CF interrupts are enabled. Setting the ECF bit in PCA0MD to logic 1 enables the CF flag to generate an interrupt request. The CF bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Clearing the CIDL bit in the PCA0MD register allows the PCA to continue normal operation while the CPU is in Idle mode.

**Table 25.1. PCA Timebase Input Options**

| CPS2 | CPS1 | CPS0 | Timebase |
|---|---|---|---|
| 0 | 0 | 0 | System clock divided by 12 |
| 0 | 0 | 1 | System clock divided by 4 |
| 0 | 1 | 0 | Timer 0 overflow |
| 0 | 1 | 1 | High-to-low transitions on ECI (max rate = system clock divided by 4) |
| 1 | 0 | 0 | System clock |
| 1 | 0 | 1 | External oscillator source divided by 8* |
| 1 | 1 | x | Reserved |
| **Note:** External oscillator source divided by 8 is synchronized with the system clock. | | | |

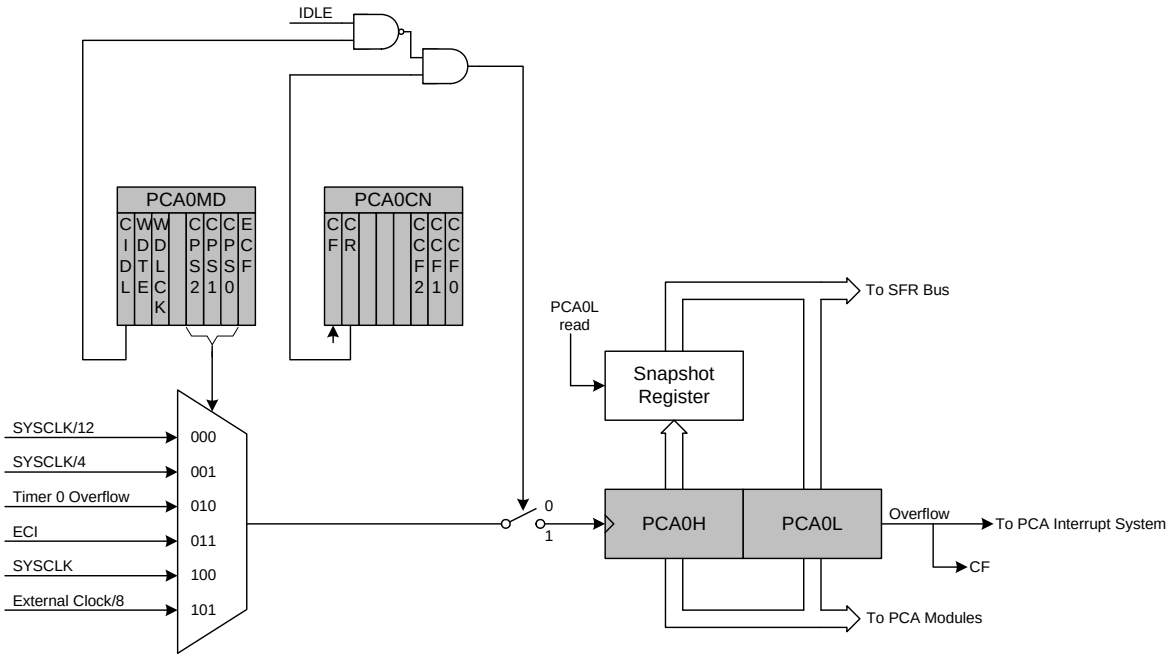



**Figure 25.2. PCA Counter/Timer Block Diagram**

## 25.2. PCA0 Interrupt Sources

Figure 25.3 shows a diagram of the PCA interrupt tree. There are five independent event flags that can be used to generate a PCA0 interrupt. They are: the main PCA counter overflow flag (CF), which is set upon a 16-bit overflow of the PCA0 counter, an intermediate overflow flag (COVF), which can be set on an overflow from the 8th, 9th, 10th, or 11th bit of the PCA0 counter, and the individual flags for each PCA channel (CCF0, CCF1, and CCF2), which are set according to the operation mode of that module. These event flags are always set when the trigger condition occurs. Each of these flags can be individually selected to generate a PCA0 interrupt, using the corresponding interrupt enable flag (ECF for CF, ECOV for COVF, and ECCFn for each CCFn). PCA0 interrupts must be globally enabled before any individual interrupt sources are recognized by the processor. PCA0 interrupts are globally enabled by setting the EA bit and the EPCA0 bit to logic 1.

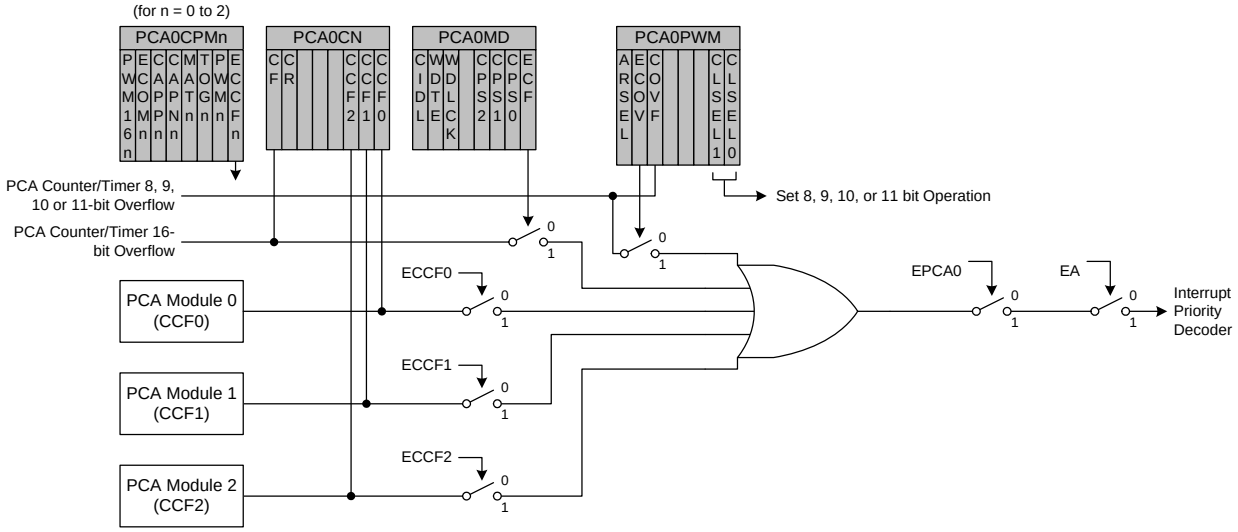



**Figure 25.3. PCA Interrupt Block Diagram**

## 25.3. Capture/Compare Modules

Each module can be configured to operate independently in one of six operation modes: edge-triggered capture, software timer, high-speed output, frequency output, 8 to 11-bit pulse width modulator, or 16-bit pulse width modulator. Each module has Special Function Registers (SFRs) associated with it in the CIP-51 system controller. These registers are used to exchange data with a module and configure the module's mode of operation. Table 25.2 summarizes the bit settings in the PCA0CPMn and PCA0PWM registers used to select the PCA capture/compare module's operating mode. Note that all modules set to use 8, 9, 10, or 11-bit PWM mode must use the same cycle length (8–11 bits). Setting the ECCFn bit in a PCA0CPMn register enables the module's CCFn interrupt.

**Table 25.2. PCA0CPM and PCA0PWM Bit Settings for PCA Capture/Compare Modules**

| Operational Mode | PCA0CPMn | | | | | | | | PCA0PWM | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit Number | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4–2 | 1–0 |
| Capture triggered by positive edge on CEXn | X | X | 1 | 0 | 0 | 0 | 0 | A | 0 | X | B | XXX | XX |
| Capture triggered by negative edge on CEXn | X | X | 0 | 1 | 0 | 0 | 0 | A | 0 | X | B | XXX | XX |
| Capture triggered by any transition on CEXn | X | X | 1 | 1 | 0 | 0 | 0 | A | 0 | X | B | XXX | XX |
| Software Timer | X | C | 0 | 0 | 1 | 0 | 0 | A | 0 | X | B | XXX | XX |
| High Speed Output | X | C | 0 | 0 | 1 | 1 | 0 | A | 0 | X | B | XXX | XX |
| Frequency Output | X | C | 0 | 0 | 0 | 1 | 1 | A | 0 | X | B | XXX | XX |
| 8-Bit Pulse Width Modulator (Note 7) | 0 | C | 0 | 0 | E | 0 | 1 | A | 0 | X | B | XXX | 00 |
| 9-Bit Pulse Width Modulator (Note 7) | 0 | C | 0 | 0 | E | 0 | 1 | A | D | X | B | XXX | 01 |
| 10-Bit Pulse Width Modulator (Note 7) | 0 | C | 0 | 0 | E | 0 | 1 | A | D | X | B | XXX | 10 |
| 11-Bit Pulse Width Modulator (Note 7) | 0 | C | 0 | 0 | E | 0 | 1 | A | D | X | B | XXX | 11 |
| 16-Bit Pulse Width Modulator | 1 | C | 0 | 0 | E | 0 | 1 | A | 0 | X | B | XXX | XX |

**Notes:**

1. X = Don't Care (no functional difference for individual module if 1 or 0).
2. A = Enable interrupts for this module (PCA interrupt triggered on CCFn set to 1).
3. B = Enable 8th, 9th, 10th or 11th bit overflow interrupt (Depends on setting of CLSEL[1:0]).
4. C = When set to 0, the digital comparator is off. For high speed and frequency output modes, the associated pin will not toggle. In any of the PWM modes, this generates a 0% duty cycle (output = 0).
5. D = Selects whether the Capture/Compare register (0) or the Auto-Reload register (1) for the associated channel is accessed via addresses PCA0CPHn and PCA0CPLn.
6. E = When set, a match event will cause the CCFn flag for the associated channel to be set.
7. All modules set to 8, 9, 10 or 11-bit PWM mode use the same cycle length setting.

### 25.3.1. Edge-triggered Capture Mode

In this mode, a valid transition on the CEXn pin causes the PCA to capture the value of the PCA counter/timer and load it into the corresponding module's 16-bit capture/compare register (PCA0CPLn and PCA0CPHn). The CAPPn and CAPNn bits in the PCA0CPMn register are used to select the type of transition that triggers the capture: low-to-high transition (positive edge), high-to-low transition (negative edge), or either transition (positive or negative edge). When a capture occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1. An interrupt request is generated if the CCFn interrupt for that module is enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. If both CAPPn and CAPNn bits are set to logic 1, then the state of the Port pin associated with CEXn can be read directly to determine whether a rising-edge or falling-edge caused the capture.

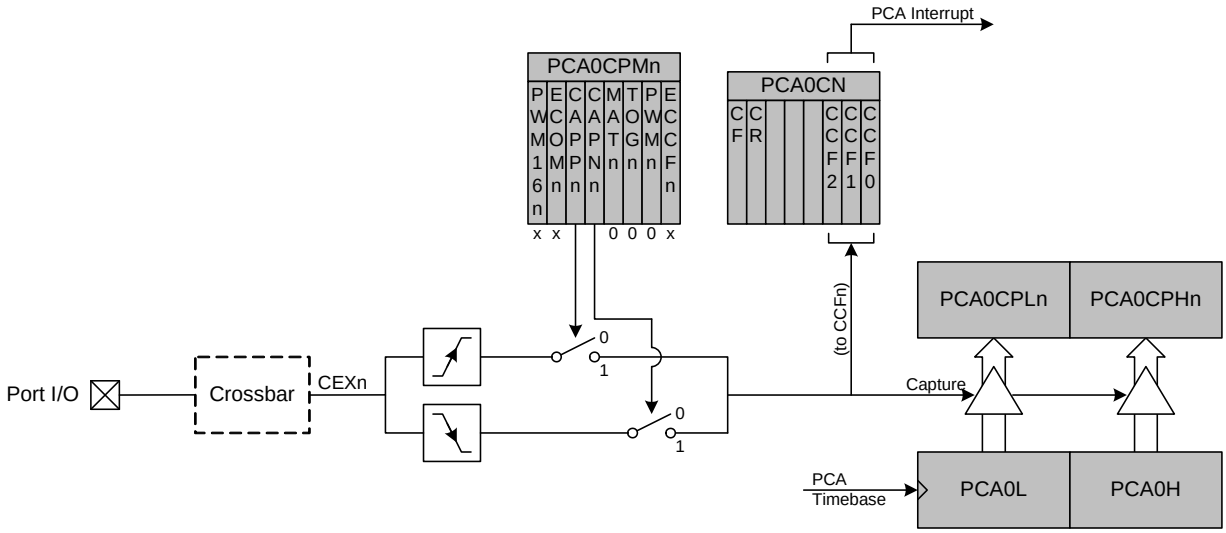



**Figure 25.4. PCA Capture Mode Diagram**

**Note:** The CEXn input signal must remain high or low for at least 2 system clock cycles to be recognized by the hardware.

### 25.3.2. Software Timer (Compare) Mode

In Software Timer mode, the PCA counter/timer value is compared to the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1. An interrupt request is generated if the CCFn interrupt for that module is enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Setting the ECOMn and MATn bits in the PCA0CPMn register enables Software Timer mode.

**Important Note About Capture/Compare Registers**: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to 0; writing to PCA0CPHn sets ECOMn to 1.

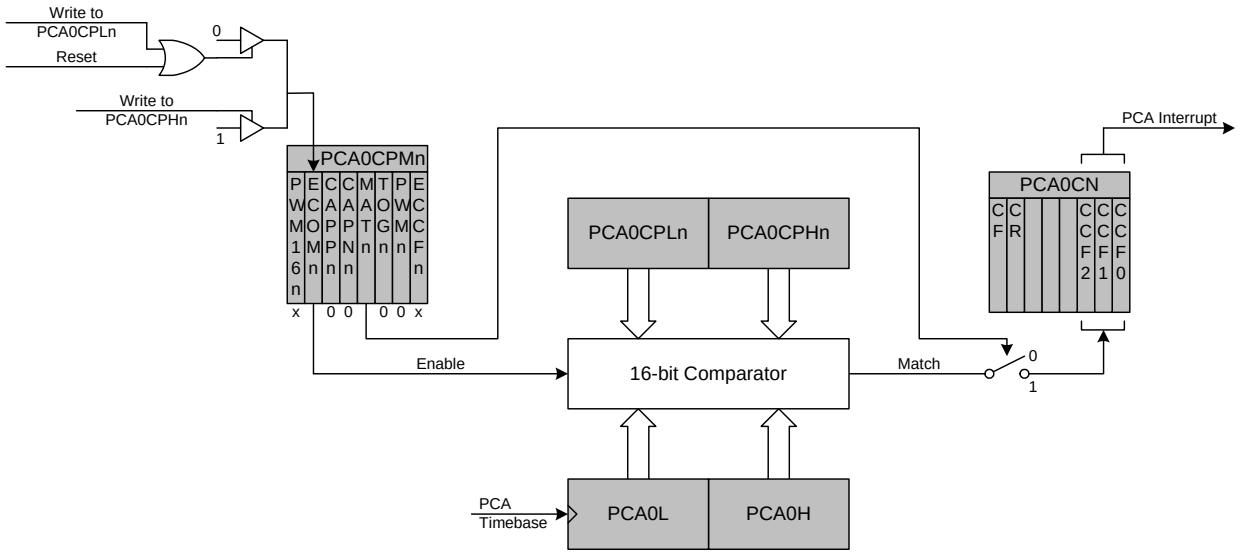



**Figure 25.5. PCA Software Timer Mode Diagram**

### 25.3.3. High-Speed Output Mode

In High-Speed Output mode, a module's associated CEXn pin is toggled each time a match occurs between the PCA Counter and the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1. An interrupt request is generated if the CCFn interrupt for that module is enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Setting the TOGn, MATn, and ECOMn bits in the PCA0CPMn register enables the High-Speed Output mode. If ECOMn is cleared, the associated pin will retain its state, and not toggle on the next match event.

**Important Note About Capture/Compare Registers**: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to 0; writing to PCA0CPHn sets ECOMn to 1.

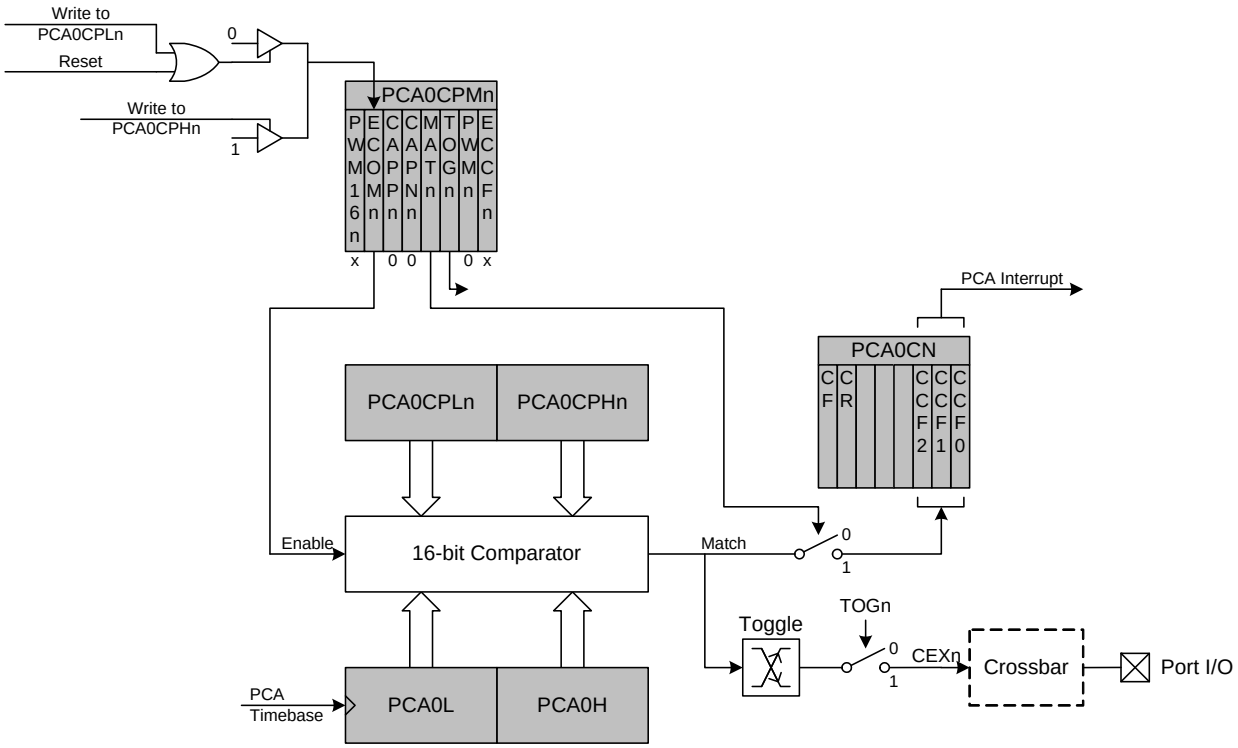



**Figure 25.6. PCA High-Speed Output Mode Diagram**

### 25.3.4. Frequency Output Mode

Frequency Output Mode produces a programmable-frequency square wave on the module's associated CEXn pin. The capture/compare module high byte holds the number of PCA clocks to count before the output is toggled. The frequency of the square wave is then defined by Equation 25.1.

$$F_{CEXn} = \frac{F_{PCA}}{2 \times PCA0CPHn}$$

Note: A value of 0x00 in the PCA0CPHn register is equal to 256 for this equation.

**Equation 25.1. Square Wave Frequency Output**

Where $F_{PCA}$ is the frequency of the clock selected by the CPS2–0 bits in the PCA mode register, PCA0MD. The lower byte of the capture/compare module is compared to the PCA counter low byte; on a match, CEXn is toggled and the offset held in the high byte is added to the matched value in PCA0CPLn. Frequency Output Mode is enabled by setting the ECOMn, TOGn, and PWMn bits in the PCA0CPMn register. Note that the MATn bit should normally be set to 0 in this mode. If the MATn bit is set to 1, the CCFn flag for the channel will be set when the 16-bit PCA0 counter and the 16-bit capture/compare register for the channel are equal.

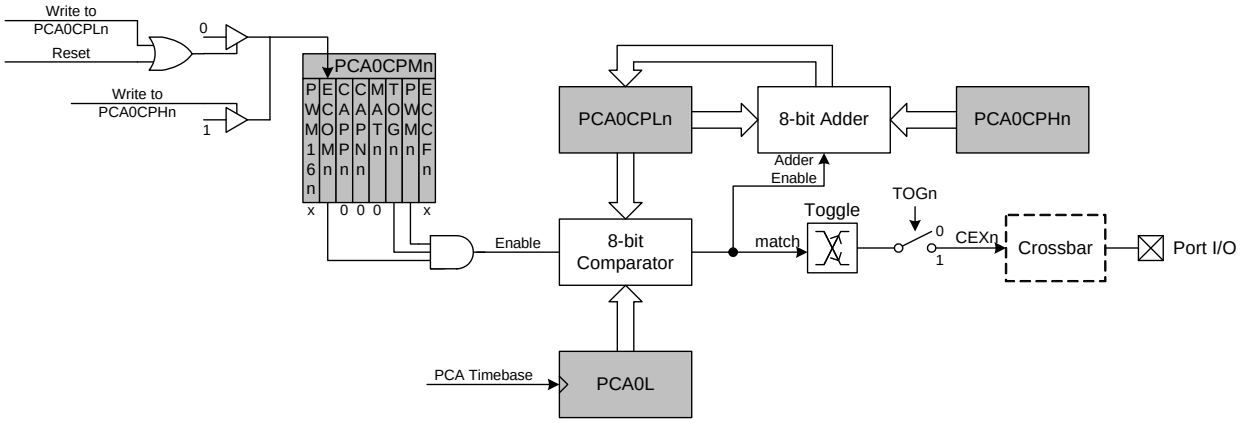



**Figure 25.7. PCA Frequency Output Mode**

### 25.3.5. 8-bit, 9-bit, 10-bit and 11-bit Pulse Width Modulator Modes

Each module can be used independently to generate a pulse width modulated (PWM) output on its associated CEXn pin. The frequency of the output is dependent on the timebase for the PCA counter/timer, and the setting of the PWM cycle length (8, 9, 10 or 11-bits). For backwards-compatibility with the 8-bit PWM mode available on other devices, the 8-bit PWM mode operates slightly different than 9, 10 and 11-bit PWM modes. **It is important to note that all channels configured for 8/9/10/11-bit PWM mode will use the same cycle length.** It is not possible to configure one channel for 8-bit PWM mode and another for 11-bit mode (for example). However, other PCA channels can be configured to Pin Capture, High-Speed Output, Software Timer, Frequency Output, or 16-bit PWM mode independently.

#### 25.3.5.1. 8-bit Pulse Width Modulator Mode

The duty cycle of the PWM output signal in 8-bit PWM mode is varied using the module's PCA0CPLn capture/compare register. When the value in the low byte of the PCA counter/timer (PCA0L) is equal to the value in PCA0CPLn, the output on the CEXn pin will be set. When the count value in PCA0L overflows, the CEXn output will be reset (see Figure 25.8). Also, when the counter/timer low byte (PCA0L) overflows from 0xFF to 0x00, PCA0CPLn is reloaded automatically with the value stored in the module's capture/compare high byte (PCA0CPHn) without software intervention. Setting the ECOMn and PWMn bits in the PCA0CPMn register, and setting the CLSEL bits in register PCA0PWM to 00b enables 8-Bit Pulse Width Modulator mode. If the MATn bit is set to 1, the CCFn flag for the module will be set each time an 8-bit comparator match (rising edge) occurs. The COVF flag in PCA0PWM can be used to detect the overflow (falling edge), which will occur every 256 PCA clock cycles. The duty cycle for 8-Bit PWM Mode is given in Equation 25.2.

**Important Note About Capture/Compare Registers**: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to 0; writing to PCA0CPHn sets ECOMn to 1.

$$\text{Duty Cycle} = \frac{(256 - PCA0CPHn)}{256}$$

**Equation 25.2. 8-Bit PWM Duty Cycle**

Using Equation 25.2, the largest duty cycle is 100% (PCA0CPHn = 0), and the smallest duty cycle is 0.39% (PCA0CPHn = 0xFF). A 0% duty cycle may be generated by clearing the ECOMn bit to 0.

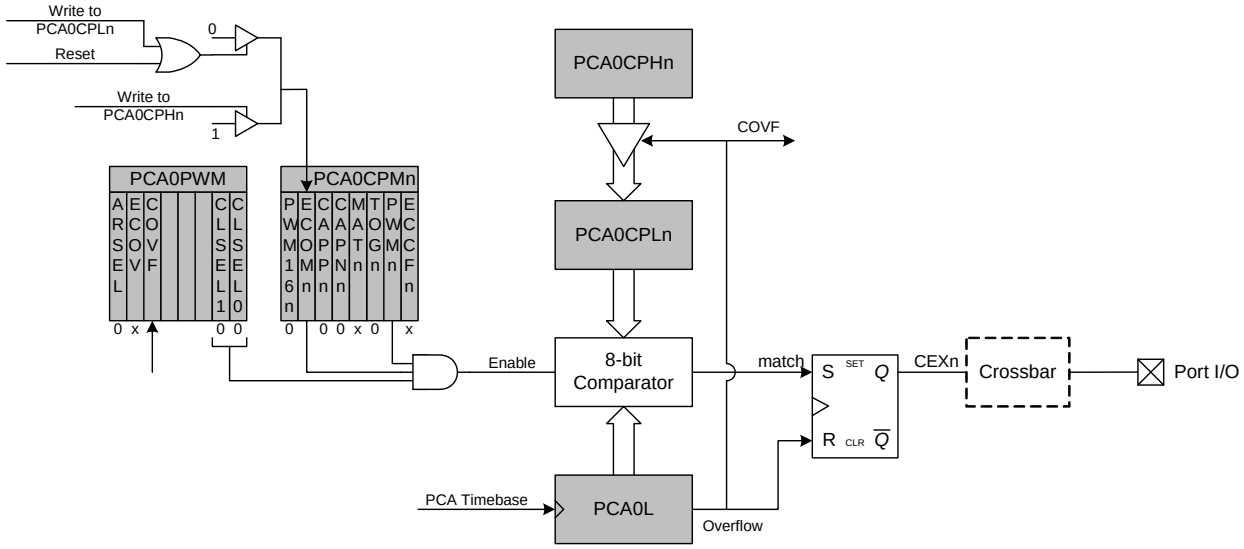



**Figure 25.8. PCA 8-Bit PWM Mode Diagram**

### 25.3.5.2. 9/10/11-bit Pulse Width Modulator Mode

The duty cycle of the PWM output signal in 9/10/11-bit PWM mode should be varied by writing to an "Auto-Reload" Register, which is dual-mapped into the PCA0CPHn and PCA0CPLn register locations. The data written to define the duty cycle should be right-justified in the registers. The auto-reload registers are accessed (read or written) when the bit ARSEL in PCA0PWM is set to 1. The capture/compare registers are accessed when ARSEL is set to 0.

When the least-significant N bits of the PCA0 counter match the value in the associated module's capture/compare register (PCA0CPn), the output on CEXn is asserted high. When the counter overflows from the Nth bit, CEXn is asserted low (see Figure 25.9). Upon an overflow from the Nth bit, the COVF flag is set, and the value stored in the module's auto-reload register is loaded into the capture/compare register. The value of N is determined by the CLSEL bits in register PCA0PWM.

The 9, 10 or 11-bit PWM mode is selected by setting the ECOMn and PWMn bits in the PCA0CPMn register, and setting the CLSEL bits in register PCA0PWM to the desired cycle length (other than 8-bits). If the MATn bit is set to 1, the CCFn flag for the module will be set each time a comparator match (rising edge) occurs. The COVF flag in PCA0PWM can be used to detect the overflow (falling edge), which will occur every 512 (9-bit), 1024 (10-bit) or 2048 (11-bit) PCA clock cycles. The duty cycle for 9/10/11-Bit PWM Mode is given in Equation 25.2, where N is the number of bits in the PWM cycle.

**Important Note About PCA0CPHn and PCA0CPLn Registers**: When writing a 16-bit value to the PCA0CPn registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to 0; writing to PCA0CPHn sets ECOMn to 1.

$$\text{Duty Cycle} = \frac{(2^N - PCA0CPn)}{2^N}$$

**Equation 25.3. 9, 10, and 11-Bit PWM Duty Cycle**

A 0% duty cycle may be generated by clearing the ECOMn bit to 0.

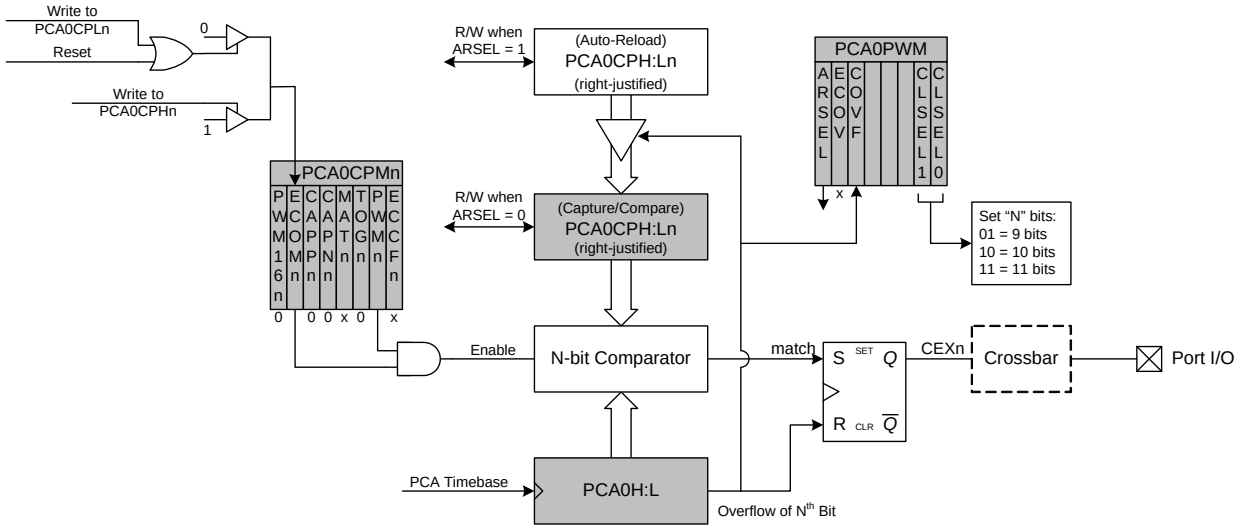



**Figure 25.9. PCA 9, 10 and 11-Bit PWM Mode Diagram**

### 25.3.6. 16-Bit Pulse Width Modulator Mode

A PCA module may also be operated in 16-Bit PWM mode. 16-bit PWM mode is independent of the other (8/9/10/11-bit) PWM modes. In this mode, the 16-bit capture/compare module defines the number of PCA clocks for the low time of the PWM signal. When the PCA counter matches the module contents, the output on CEXn is asserted high; when the 16-bit counter overflows, CEXn is asserted low. To output a varying duty cycle, new value writes should be synchronized with PCA CCFn match interrupts. 16-Bit PWM Mode is enabled by setting the ECOMn, PWMn, and PWM16n bits in the PCA0CPMn register. For a varying duty cycle, match interrupts should be enabled (ECCFn = 1 AND MATn = 1) to help synchronize the capture/compare register writes. If the MATn bit is set to 1, the CCFn flag for the module will be set each time a 16-bit comparator match (rising edge) occurs. The CF flag in PCA0CN can be used to detect the overflow (falling edge). The duty cycle for 16-Bit PWM Mode is given by Equation 25.4.

**Important Note About Capture/Compare Registers**: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to 0; writing to PCA0CPHn sets ECOMn to 1.

$$\text{Duty Cycle} = \frac{(65536 - PCA0CPn)}{65536}$$

**Equation 25.4. 16-Bit PWM Duty Cycle**

Using Equation 25.4, the largest duty cycle is 100% (PCA0CPn = 0), and the smallest duty cycle is 0.0015% (PCA0CPn = 0xFFFF). A 0% duty cycle may be generated by clearing the ECOMn bit to 0.

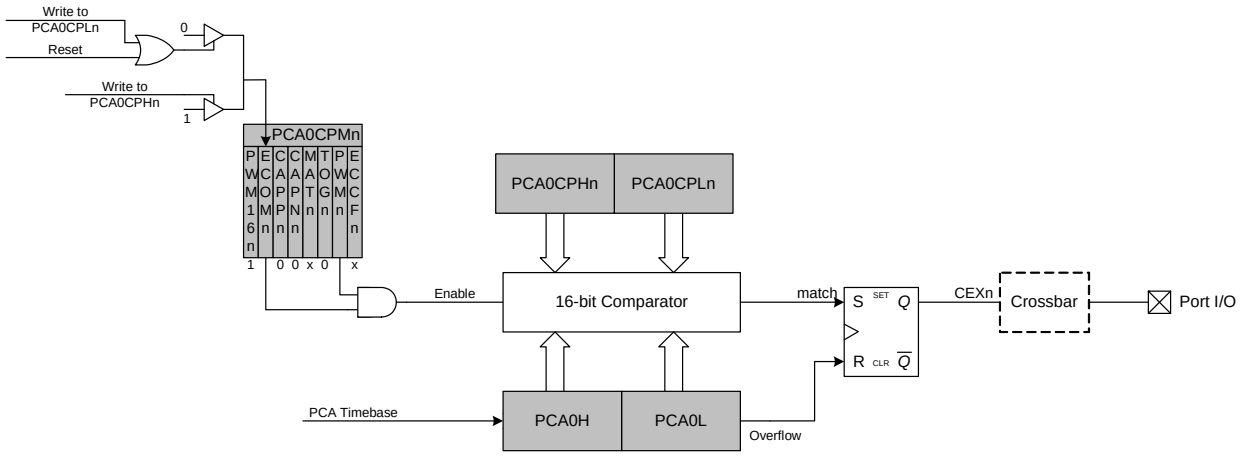



**Figure 25.10. PCA 16-Bit PWM Mode**

## 25.4. Watchdog Timer Mode

A programmable watchdog timer (WDT) function is available through the PCA Module 2. The WDT is used to generate a reset if the time between writes to the WDT update register (PCA0CPH2) exceed a specified limit. The WDT can be configured and enabled/disabled as needed by software.

With the WDTE bit set in the PCA0MD register, Module 2 operates as a watchdog timer (WDT). The Module 2 high byte is compared to the PCA counter high byte; the Module 2 low byte holds the offset to be used when WDT updates are performed. **The Watchdog Timer is enabled on reset. Writes to some PCA registers are restricted while the Watchdog Timer is enabled.** The WDT will generate a reset shortly after code begins execution. To avoid this reset, the WDT should be explicitly disabled (and optionally re-configured and re-enabled if it is used in the system).

### 25.4.1. Watchdog Timer Operation

While the WDT is enabled:

- PCA counter is forced on.
- Writes to PCA0L and PCA0H are not allowed.
- PCA clock source bits (CPS2–CPS0) are frozen.
- PCA Idle control bit (CIDL) is frozen.
- Module 2 is forced into software timer mode.
- Writes to the Module 2 mode register (PCA0CPM2) are disabled.

While the WDT is enabled, writes to the CR bit will not change the PCA counter state; the counter will run until the WDT is disabled. The PCA counter run control bit (CR) will read zero if the WDT is enabled but user software has not enabled the PCA counter. If a match occurs between PCA0CPH2 and PCA0H while the WDT is enabled, a reset will be generated. To prevent a WDT reset, the WDT may be updated with a write of any value to PCA0CPH2. Upon a PCA0CPH2 write, PCA0H plus the offset held in PCA0CPL2 is loaded into PCA0CPH2 (See Figure 25.11).

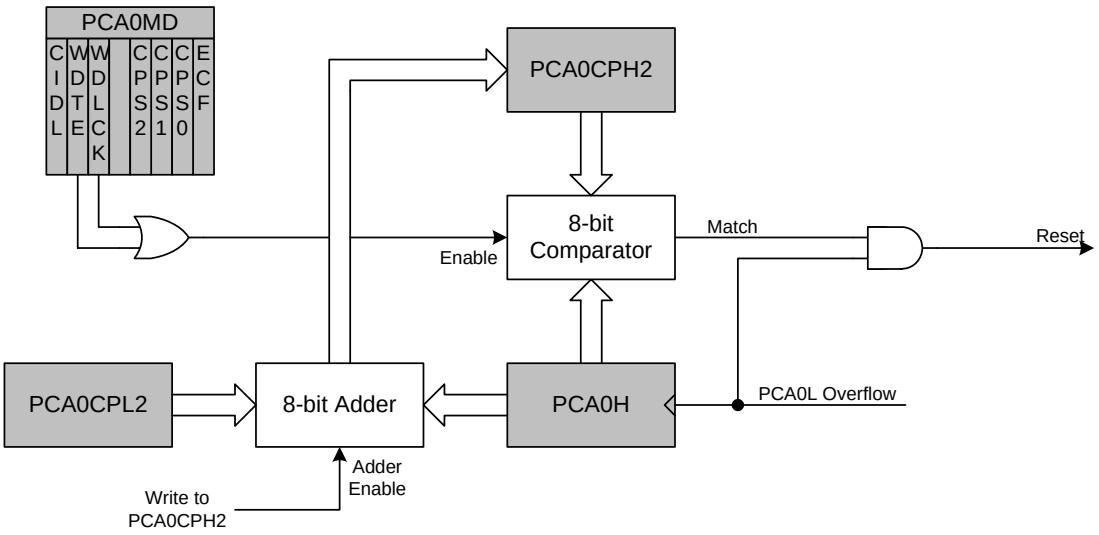



**Figure 25.11. PCA Module 2 with Watchdog Timer Enabled**

The 8-bit offset held in PCA0CPH2 is compared to the upper byte of the 16-bit PCA counter. This offset value is the number of PCA0L overflows before a reset. Up to 256 PCA clocks may pass before the first PCA0L overflow occurs, depending on the value of the PCA0L when the update is performed. The total offset is then given (in PCA clocks) by Equation 25.5, where PCA0L is the value of the PCA0L register at the time of the update.

$$Offset = (256 \times PCA0CPL2) + (256 - PCA0L)$$

**Equation 25.5. Watchdog Timer Offset in PCA Clocks**

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH2 and PCA0H. Software may force a WDT reset by writing a 1 to the CCF2 flag (PCA0CN.2) while the WDT is enabled.

### 25.4.2. Watchdog Timer Usage

To configure the WDT, perform the following tasks:

1. Disable the WDT by writing a 0 to the WDTE bit.
2. Select the desired PCA clock source (with the CPS2–CPS0 bits).
3. Load PCA0CPL2 with the desired WDT update offset value.
4. Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
5. Enable the WDT by setting the WDTE bit to 1.
6. Reset the WDT timer by writing to PCA0CPH2.

The PCA clock source and Idle mode select cannot be changed while the WDT is enabled. The watchdog timer is enabled by setting the WDTE or WDLCK bits in the PCA0MD register. When WDLCK is set, the WDT cannot be disabled until the next system reset. If WDLCK is not set, the WDT is disabled by clearing the WDTE bit.

The WDT is enabled following any reset. The PCA0 counter clock defaults to the system clock divided by 12, PCA0L defaults to 0x00, and PCA0CPL2 defaults to 0x00. Using Equation 25.5, this results in a WDT timeout interval of 256 PCA clock cycles, or 3072 system clock cycles. Table 25.3 lists some example timeout intervals for typical system clocks.

**Table 25.3. Watchdog Timer Timeout Intervals[1]**

| System Clock (Hz) | PCA0CPL2 | Timeout Interval (ms) |
|---|---|---|
| 24,500,000 | 255 | 32.1 |
| 24,500,000 | 128 | 16.2 |
| 24,500,000 | 32 | 4.1 |
| 3,062,500[2] | 255 | 257 |
| 3,062,500[2] | 128 | 129.5 |
| 3,062,500[2] | 32 | 33.1 |
| 32,000 | 255 | 24576 |
| 32,000 | 128 | 12384 |
| 32,000 | 32 | 3168 |

**Notes:**
1. Assumes SYSCLK/12 as the PCA clock source, and a PCA0L value of 0x00 at the update time.
2. Internal SYSCLK reset frequency = Internal Oscillator divided by 8.

## 25.5. Register Descriptions for PCA0

Following are detailed descriptions of the special function registers related to the operation of the PCA.

### SFR Definition 25.1. PCA0CN: PCA Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CF | CR | | | | CCF2 | CCF1 | CCF0 |
| Type | R/W | R/W | R | R | R | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD8; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | CF | **PCA Counter/Timer Overflow Flag.**<br>Set by hardware when the PCA Counter/Timer overflows from 0xFFFF to 0x0000. When the Counter/Timer Overflow (CF) interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |
| 6 | CR | **PCA Counter/Timer Run Control.**<br>This bit enables/disables the PCA Counter/Timer.<br>0: PCA Counter/Timer disabled.<br>1: PCA Counter/Timer enabled. |
| 5:3 | Unused | Unused. Read = 000b, Write = Don't care. |
| 2 | CCF2 | **PCA Module 2 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF2 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |
| 1 | CCF1 | **PCA Module 1 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF1 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |
| 0 | CCF0 | **PCA Module 0 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF0 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |

## SFR Definition 25.2. PCA0MD: PCA Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CIDL | WDTE | WDLCK | | CPS2 | CPS1 | CPS0 | ECF |
| Type | R/W | R/W | R/W | R | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD9

| Bit | Name | Function |
|---|---|---|
| 7 | CIDL | **PCA Counter/Timer Idle Control.**<br>Specifies PCA behavior when CPU is in Idle Mode.<br>0: PCA continues to function normally while the system controller is in Idle Mode.<br>1: PCA operation is suspended while the system controller is in Idle Mode. |
| 6 | WDTE | **Watchdog Timer Enable.**<br>If this bit is set, PCA Module 2 is used as the watchdog timer.<br>0: Watchdog Timer disabled.<br>1: PCA Module 2 enabled as Watchdog Timer. |
| 5 | WDLCK | **Watchdog Timer Lock.**<br>This bit locks/unlocks the Watchdog Timer Enable. When WDLCK is set, the Watchdog Timer may not be disabled until the next system reset.<br>0: Watchdog Timer Enable unlocked.<br>1: Watchdog Timer Enable locked. |
| 4 | Unused | Unused. Read = 0b, Write = Don't care. |
| 3:1 | CPS[2:0] | **PCA Counter/Timer Pulse Select.**<br>These bits select the timebase source for the PCA counter<br>000: System clock divided by 12<br>001: System clock divided by 4<br>010: Timer 0 overflow<br>011: High-to-low transitions on ECI (max rate = system clock divided by 4)<br>100: System clock<br>101: External clock divided by 8 (synchronized with the system clock)<br>11x: Reserved |
| 0 | ECF | **PCA Counter/Timer Overflow Interrupt Enable.**<br>This bit sets the masking of the PCA Counter/Timer Overflow (CF) interrupt.<br>0: Disable the CF interrupt.<br>1: Enable a PCA Counter/Timer Overflow interrupt request when CF (PCA0CN.7) is set. |
| **Note:** When the WDTE bit is set to 1, the other bits in the PCA0MD register cannot be modified. To change the contents of the PCA0MD register, the Watchdog Timer must first be disabled. | | |

## SFR Definition 25.3. PCA0PWM: PCA PWM Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ARSEL | ECOV | COVF | | | | CLSEL[1:0] | |
| Type | R/W | R/W | R/W | R | R | R | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xF7

| Bit | Name | Function |
|---|---|---|
| 7 | ARSEL | **Auto-Reload Register Select.**<br>This bit selects whether to read and write the normal PCA capture/compare registers (PCA0CPn), or the Auto-Reload registers at the same SFR addresses. This function is used to define the reload value for 9, 10, and 11-bit PWM modes. In all other modes, the Auto-Reload registers have no function.<br>0: Read/Write Capture/Compare Registers at PCA0CPHn and PCA0CPLn.<br>1: Read/Write Auto-Reload Registers at PCA0CPHn and PCA0CPLn. |
| 6 | ECOV | **Cycle Overflow Interrupt Enable.**<br>This bit sets the masking of the Cycle Overflow Flag (COVF) interrupt.<br>0: COVF will not generate PCA interrupts.<br>1: A PCA interrupt will be generated when COVF is set. |
| 5 | COVF | **Cycle Overflow Flag.**<br>This bit indicates an overflow of the 8th, 9th, 10th, or 11th bit of the main PCA counter (PCA0). The specific bit used for this flag depends on the setting of the Cycle Length Select bits. The bit can be set by hardware or software, but must be cleared by software.<br>0: No overflow has occurred since the last time this bit was cleared.<br>1: An overflow has occurred since the last time this bit was cleared. |
| 4:2 | Unused | Unused. Read = 000b; Write = Don't care. |
| 1:0 | CLSEL[1:0] | **Cycle Length Select.**<br>When 16-bit PWM mode is not selected, these bits select the length of the PWM cycle, between 8, 9, 10, or 11 bits. This affects all channels configured for PWM which are not using 16-bit PWM mode. These bits are ignored for individual channels configured to16-bit PWM mode.<br>00: 8 bits.<br>01: 9 bits.<br>10: 10 bits.<br>11: 11 bits. |

## SFR Definition 25.4. PCA0CPMn: PCA Capture/Compare Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PWM16n | ECOMn | CAPPn | CAPNn | MATn | TOGn | PWMn | ECCFn |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Addresses: PCA0CPM0 = 0xDA, PCA0CPM1 = 0xDB, PCA0CPM2 = 0xDC

| Bit | Name | Function |
|---|---|---|
| 7 | PWM16n | **16-bit Pulse Width Modulation Enable.**<br>This bit enables 16-bit mode when Pulse Width Modulation mode is enabled.<br>0: 8 to 11-bit PWM selected.<br>1: 16-bit PWM selected. |
| 6 | ECOMn | **Comparator Function Enable.**<br>This bit enables the comparator function for PCA module n when set to 1. |
| 5 | CAPPn | **Capture Positive Function Enable.**<br>This bit enables the positive edge capture for PCA module n when set to 1. |
| 4 | CAPNn | **Capture Negative Function Enable.**<br>This bit enables the negative edge capture for PCA module n when set to 1. |
| 3 | MATn | **Match Function Enable.**<br>This bit enables the match function for PCA module n when set to 1. When enabled, matches of the PCA counter with a module's capture/compare register cause the CCFn bit in PCA0MD register to be set to logic 1. |
| 2 | TOGn | **Toggle Function Enable.**<br>This bit enables the toggle function for PCA module n when set to 1. When enabled, matches of the PCA counter with a module's capture/compare register cause the logic level on the CEXn pin to toggle. If the PWMn bit is also set to logic 1, the module operates in Frequency Output Mode. |
| 1 | PWMn | **Pulse Width Modulation Mode Enable.**<br>This bit enables the PWM function for PCA module n when set to 1. When enabled, a pulse width modulated signal is output on the CEXn pin. 8 to 11-bit PWM is used if PWM16n is cleared; 16-bit mode is used if PWM16n is set to logic 1. If the TOGn bit is also set, the module operates in Frequency Output Mode. |
| 0 | ECCFn | **Capture/Compare Flag Interrupt Enable.**<br>This bit sets the masking of the Capture/Compare Flag (CCFn) interrupt.<br>0: Disable CCFn interrupts.<br>1: Enable a Capture/Compare Flag interrupt request when CCFn is set. |
| **Note:** When the WDTE bit is set to 1, the PCA0CPM2 register cannot be modified, and module 2 acts as the watchdog timer. To change the contents of the PCA0CPM2 register or the function of module 2, the Watchdog Timer must be disabled. | | |

## SFR Definition 25.5. PCA0L: PCA Counter/Timer Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0[7:0] | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xF9

| Bit | Name | Function |
|---|---|---|
| 7:0 | PCA0[7:0] | **PCA Counter/Timer Low Byte.**<br>The PCA0L register holds the low byte (LSB) of the 16-bit PCA Counter/Timer. |

**Note:** When the WDTE bit is set to 1, the PCA0L register cannot be modified by software. To change the contents of the PCA0L register, the Watchdog Timer must first be disabled.

## SFR Definition 25.6. PCA0H: PCA Counter/Timer High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0[15:8] | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xFA

| Bit | Name | Function |
|---|---|---|
| 7:0 | PCA0[15:8] | **PCA Counter/Timer High Byte.**<br>The PCA0H register holds the high byte (MSB) of the 16-bit PCA Counter/Timer. Reads of this register will read the contents of a “snapshot” register, whose contents are updated only when the contents of PCA0L are read (see Section 25.1). |

**Note:** When the WDTE bit is set to 1, the PCA0H register cannot be modified by software. To change the contents of the PCA0H register, the Watchdog Timer must first be disabled.

## SFR Definition 25.7. PCA0CPLn: PCA Capture Module Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPn[7:0] | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Addresses: PCA0CPL0 = 0xFB, PCA0CPL1 = 0xE9, PCA0CPL2 = 0xEB

| Bit | Name | Function |
|---|---|---|
| 7:0 | PCA0CPn[7:0] | **PCA Capture Module Low Byte.**<br>The PCA0CPLn register holds the low byte (LSB) of the 16-bit capture module n. This register address also allows access to the low byte of the corresponding PCA channel's auto-reload value for 9, 10, or 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| **Note:** A write to this register will clear the module's ECOMn bit to a 0. | | |

## SFR Definition 25.8. PCA0CPHn: PCA Capture Module High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPn[15:8] | | | | | | | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Addresses: PCA0CPH0 = 0xFC, PCA0CPH1 = 0xEA, PCA0CPH2 = 0xEC

| Bit | Name | Function |
|---|---|---|
| 7:0 | PCA0CPn[15:8] | **PCA Capture Module High Byte.**<br>The PCA0CPHn register holds the high byte (MSB) of the 16-bit capture module n. This register address also allows access to the high byte of the corresponding PCA channel's auto-reload value for 9, 10, or 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| **Note:** A write to this register will set the module's ECOMn bit to a 1. | | |

# 26. C2 Interface

C8051F336/7/8/9 devices include an on-chip Silicon Labs 2-Wire (C2) debug interface to allow Flash programming and in-system debugging with the production part installed in the end application. The C2 interface uses a clock signal (C2CK) and a bi-directional C2 data signal (C2D) to transfer information between the device and a host system. See the C2 Interface Specification for details on the C2 protocol.

## 26.1. C2 Interface Registers

The following describes the C2 registers necessary to perform Flash programming through the C2 interface. All C2 registers are accessed through the C2 interface as described in the C2 Interface Specification.

### C2 Register Definition 26.1. C2ADD: C2 Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | C2ADD[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Name | Function | |
|---|---|---|---|
| 7:0 | C2ADD[7:0] | **C2 Address.**<br>The C2ADD register is accessed via the C2 interface to select the target Data register for C2 Data Read and Data Write commands. | |
| | | **Address** | **Description** |
| | | 0x00 | Selects the Device ID register for Data Read instructions |
| | | 0x01 | Selects the Revision ID register for Data Read instructions |
| | | 0x02 | Selects the C2 Flash Programming Control register for Data Read/Write instructions |
| | | 0xB4 | Selects the C2 Flash Programming Data register for Data Read/Write instructions |

## C2 Register Definition 26.2. DEVICEID: C2 Device ID

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DEVICEID[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |

C2 Address: 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | DEVICEID[7:0] | **Device ID.**<br>This read-only register returns the 8-bit device ID: 0x14 (C8051F336/7/8/9). |

## C2 Register Definition 26.3. REVID: C2 Revision ID

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | REVID[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | Varies | Varies | Varies | Varies | Varies | Varies | Varies | Varies |

C2 Address: 0x01

| Bit | Name | Function |
|---|---|---|
| 7:0 | REVID[7:0] | **Revision ID.**<br>This read-only register returns the 8-bit revision ID. For example: 0x00 = Revision A. |

## C2 Register Definition 26.4. FPCTL: C2 Flash Programming Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FPCTL[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

C2 Address: 0x02

| Bit | Name | Function |
|---|---|---|
| 7:0 | FPCTL[7:0] | **Flash Programming Control Register.**<br>This register is used to enable Flash programming via the C2 interface. To enable C2 Flash programming, the following codes must be written in order: 0x02, 0x01. Note that once C2 Flash programming is enabled, a system reset must be issued to resume normal operation. |

## C2 Register Definition 26.5. FPDAT: C2 Flash Programming Data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FPDAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

C2 Address: 0xB4

| Bit | Name | Function |
|---|---|---|
| 7:0 | FPDAT[7:0] | **C2 Flash Programming Data Register.**<br>This register is used to pass Flash commands, addresses, and data during C2 Flash accesses. Valid commands are listed below. |

| Code | Command |
|---|---|
| 0x06 | Flash Block Read |
| 0x07 | Flash Block Write |
| 0x08 | Flash Page Erase |
| 0x03 | Device Erase |

## 26.2. C2 Pin Sharing

The C2 protocol allows the C2 pins to be shared with user functions so that in-system debugging and Flash programming may be performed. This is possible because C2 communication is typically performed when the device is in the halt state, where all on-chip peripherals and user software are stalled. In this halted state, the C2 interface can safely ‘borrow’ the C2CK (RST) and C2D pins. In most applications, external resistors are required to isolate C2 interface traffic from the user application. A typical isolation configuration is shown in Figure 26.1.

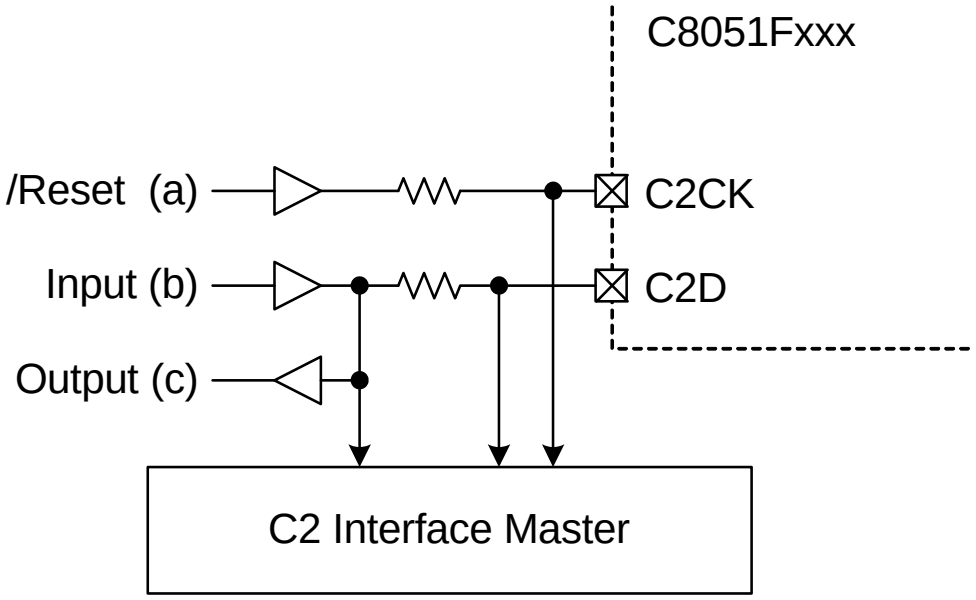



**Figure 26.1. Typical C2 Pin Sharing**

The configuration in Figure 26.1 assumes the following:

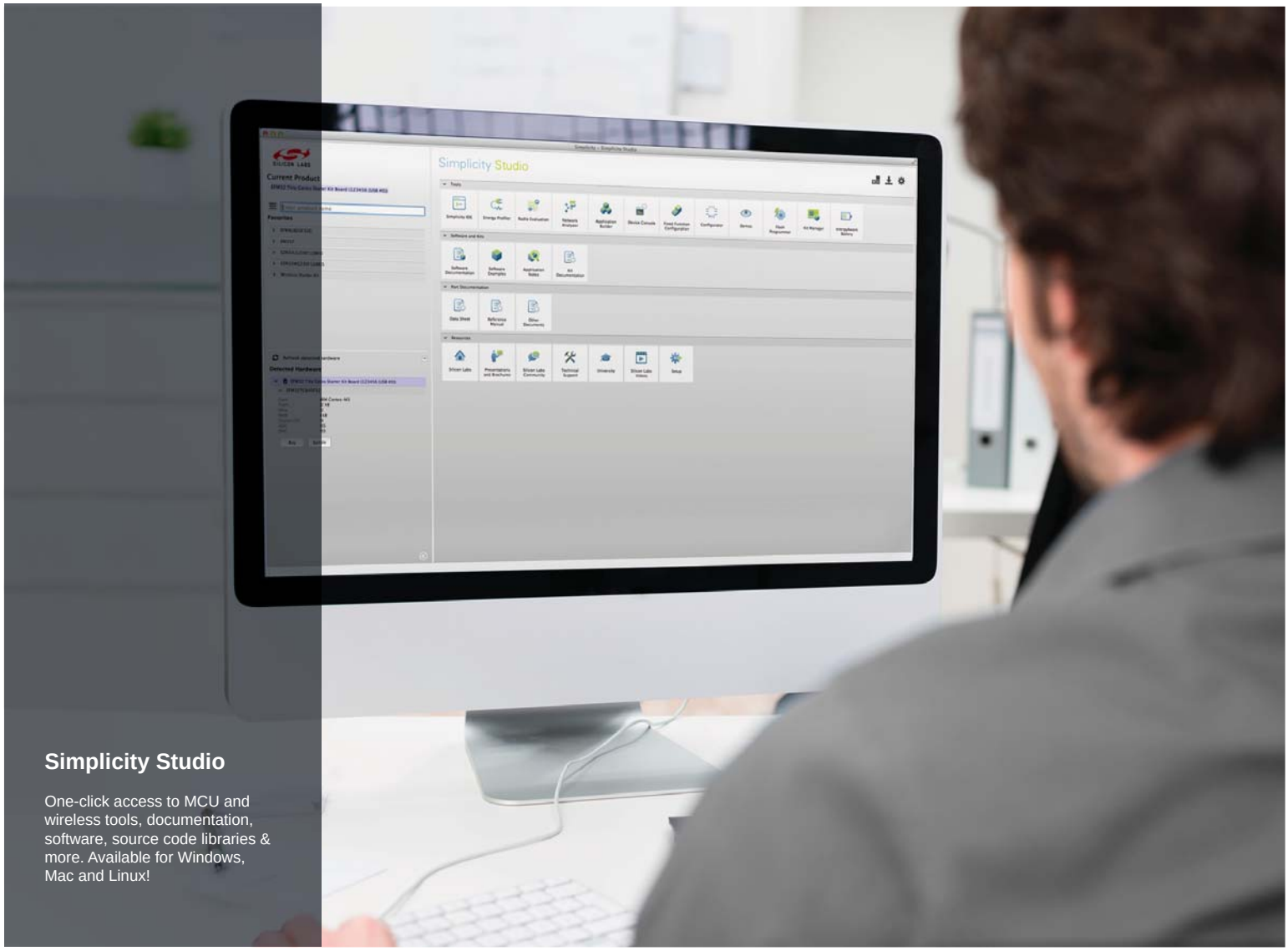
1. The user input (b) cannot change state while the target device is halted.
2. The RST pin on the target device is used as an input only.

Additional resistors may be necessary depending on the specific application.

# DOCUMENT CHANGE LIST

## Revision 0.2 to Revision 1.0

- Updated electrical specification tables based on test, characterization, and qualification data.
- Corrected minor errors in figures and text.
- Updated package information with JEDEC standard drawings for package and land pattern.




## Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

**IoT Portfolio**
*www.silabs.com/IoT*

**SW/HW**
*www.silabs.com/simplicity*

**Quality**
*www.silabs.com/quality*

**Support and Community**
*community.silabs.com*

**Disclaimer**
Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

**Trademark Information**
Silicon Laboratories Inc., Silicon Laboratories, Silicon Labs, SiLabs and the Silicon Labs logo, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem ®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

**Silicon Laboratories Inc.**
**400 West Cesar Chavez**
**Austin, TX 78701**
**USA**

**http://www.silabs.com**