# C8051F54x

## Mixed Signal ISP Flash MCU Family

### Analog Peripherals
- **12-Bit ADC**
  - Up to 200 ksps
  - Up to 25 external single-ended inputs
  - VREF from on-chip VREF, external pin or $V_{DD}$
  - Internal or external start of conversion source
  - Built-in temperature sensor
- **Two Comparators**
  - Programmable hysteresis and response time
  - Configurable as interrupt or reset source
  - Low current

### 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 1.8 to 5.25 V
- Typical operating current: 19 mA at 50 MHz; Typical stop mode current: 1 µA

### High-Speed 8051 µC Core
- Pipelined instruction architecture; executes 70% of instructions in 1 or 2 system clocks
- Up to 50 MIPS throughput with 50 MHz clock
- Expanded interrupt handler

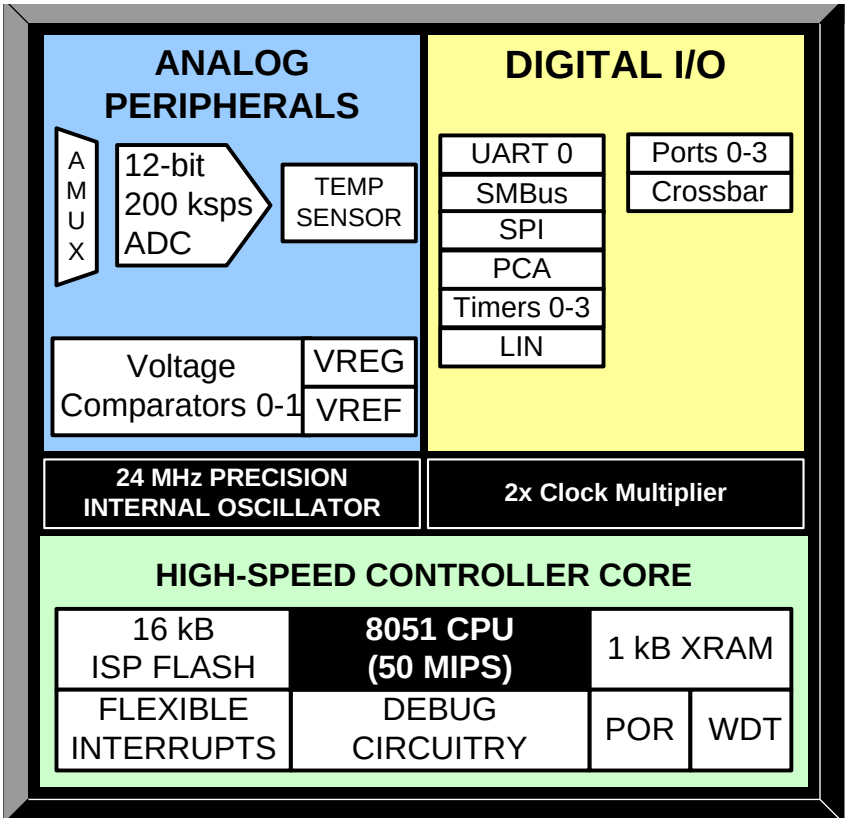
### Memory
- 1280 bytes internal data RAM (256 + 1024 XRAM)
- 16 or 8 kB Flash; In-system programmable in 512-byte Sectors

### Digital Peripherals
- 25 or 18 Port I/O; All 5 V tolerant
- LIN 2.1 Controller (Master and Slave capable); no crystal required
- Hardware enhanced UART, SMBus™, and enhanced SPI™ serial ports
- Four general purpose 16-bit counter/timers
- 16-Bit programmable counter array (PCA) with six capture/compare modules and enhanced PWM functionality

### Clock Sources
- Internal 24 MHz with ±0.5% accuracy master LIN operation
- 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

### Packages
- 32-Pin QFP/QFN (C8051F540/1/4/5)
- 24-Pin QFN (C8051F542/3/6/7)

### Automotive Qualified
- Temperature Range: –40 to +125 °C
- Compliant to AEC-Q100





Copyright © 2011 by Silicon Laboratories


Downloaded from Arrow.com.

Downloaded from Arrow.com.

## Table of Contents

**1. System Overview ..... 13**
**2. Ordering Information ..... 16**
**3. Pin Definitions ..... 18**
**4. Package Specifications ..... 23**
4.1. QFP-32 Package Specifications ..... 23
4.2. QFN-32 Package Specifications ..... 25
4.3. QFN-24 Package Specifications ..... 27
**5. 12-Bit ADC (ADC0) ..... 29**
5.1. Modes of Operation ..... 30
5.2. Output Code Formatting ..... 34
5.3. Selectable Gain ..... 35
5.4. Programmable Window Detector ..... 43
**6. Electrical Characteristics ..... 47**
6.1. Absolute Maximum Specifications ..... 47
6.2. Electrical Characteristics ..... 48
6.1. ADC0 Analog Multiplexer ..... 58
6.2. Temperature Sensor ..... 60
**7. Voltage Reference ..... 61**
**8. Comparators ..... 63**
8.1. Comparator Multiplexer ..... 69
**9. Voltage Regulator (REG0) ..... 72**
**10. CIP-51 Microcontroller ..... 74**
10.1. Performance ..... 74
10.2. Instruction Set ..... 76
10.3. CIP-51 Register Descriptions ..... 80
10.4. Serial Number Special Function Registers (SFRs) ..... 84
**11. Memory Organization ..... 85**
11.1. Program Memory ..... 85
11.2. Data Memory ..... 86
11.3. External RAM ..... 87
**12. Special Function Registers ..... 89**
12.1. SFR Paging ..... 89
12.2. Interrupts and SFR Paging ..... 89
12.3. SFR Page Stack Example ..... 91
**13. Interrupts ..... 105**
13.1. MCU Interrupt Sources and Vectors ..... 105
13.2. Interrupt Register Descriptions ..... 108
13.3. External Interrupts INT0 and INT1 ..... 115
**14. Flash Memory ..... 117**
14.1. Programming the Flash Memory ..... 117
14.2. Non-volatile Data Storage ..... 119
14.3. Security Options ..... 119
14.4. Flash Write and Erase Guidelines ..... 121



Downloaded from Arrow.com.

**15. Power Management Modes ..... 126**
15.1. Idle Mode ..... 126
15.2. Stop Mode ..... 127
15.3. Suspend Mode ..... 127
**16. Reset Sources ..... 129**
16.1. Power-On Reset ..... 130
16.2. Power-Fail Reset/VDD Monitor ..... 130
16.3. External Reset ..... 132
16.4. Missing Clock Detector Reset ..... 132
16.5. Comparator0 Reset ..... 133
16.6. PCA Watchdog Timer Reset ..... 133
16.7. Flash Error Reset ..... 133
16.8. Software Reset ..... 133
**17. Oscillators and Clock Selection ..... 135**
17.1. System Clock Selection ..... 135
17.2. Programmable Internal Oscillator ..... 137
17.3. Clock Multiplier ..... 140
17.4. External Oscillator Drive Circuit ..... 142
**18. Port Input/Output ..... 147**
18.1. Port I/O Modes of Operation ..... 148
18.2. Assigning Port I/O Pins to Analog and Digital Functions ..... 149
18.3. Priority Crossbar Decoder ..... 150
18.4. Port I/O Initialization ..... 152
18.5. Port Match ..... 157
18.6. Special Function Registers for Accessing and Configuring Port I/O ..... 161
**19. Local Interconnect Network (LIN) ..... 170**
19.1. Software Interface with the LIN Controller ..... 171
19.2. LIN Interface Setup and Operation ..... 171
19.3. LIN Master Mode Operation ..... 174
19.4. LIN Slave Mode Operation ..... 175
19.5. Sleep Mode and Wake-Up ..... 176
19.6. Error Detection and Handling ..... 176
19.7. LIN Registers ..... 177
**20. SMBus ..... 187**
20.1. Supporting Documents ..... 188
20.2. SMBus Configuration ..... 188
20.3. SMBus Operation ..... 188
20.4. Using the SMBus ..... 190
20.5. SMBus Transfer Modes ..... 197
20.6. SMBus Status Decoding ..... 201
**21. UART0 ..... 205**
21.1. Baud Rate Generator ..... 205
21.2. Data Format ..... 207
21.3. Configuration and Operation ..... 208
**22. Enhanced Serial Peripheral Interface (SPI0) ..... 214**



Downloaded from Arrow.com.

22.1. Signal Descriptions ..... 215
22.2. SPI0 Master Mode Operation ..... 216
22.3. SPI0 Slave Mode Operation ..... 218
22.4. SPI0 Interrupt Sources ..... 218
22.5. Serial Clock Phase and Polarity ..... 219
22.6. SPI Special Function Registers ..... 220
**23. Timers ..... 227**
23.1. Timer 0 and Timer 1 ..... 229
23.2. Timer 2 ..... 237
23.3. Timer 3 ..... 243
**24. Programmable Counter Array ..... 249**
24.1. PCA Counter/Timer ..... 250
24.2. PCA0 Interrupt Sources ..... 251
24.3. Capture/Compare Modules ..... 252
24.4. Watchdog Timer Mode ..... 260
24.5. Register Descriptions for PCA0 ..... 263
**25. C2 Interface ..... 269**
25.1. C2 Interface Registers ..... 269
25.2. C2 Pin Sharing ..... 272



Downloaded from Arrow.com.

## List of Figures

Figure 1.1. C8051F540/1/4/5 Block Diagram ..... 14
Figure 1.2. C8051F542/3/6/7 Block Diagram ..... 15
Figure 3.1. QFP-32 Pinout Diagram (Top View) ..... 20
Figure 3.2. QFN-32 Pinout Diagram (Top View) ..... 21
Figure 3.3. QFN-24 Pinout Diagram (Top View) ..... 22
Figure 4.1. QFP-32 Package Drawing ..... 23
Figure 4.2. QFP-32 Landing Diagram ..... 24
Figure 4.3. QFN-32 Package Drawing ..... 25
Figure 4.4. QFN-32 Landing Diagram ..... 26
Figure 4.5. QFN-24 Package Drawing ..... 27
Figure 4.6. QFN-24 Landing Diagram ..... 28
Figure 5.1. ADC0 Functional Block Diagram ..... 29
Figure 5.2. ADC0 Tracking Modes ..... 31
Figure 5.3. 12-Bit ADC Tracking Mode Example ..... 32
Figure 5.4. 12-Bit ADC Burst Mode Example With Repeat Count Set to 4 ..... 33
Figure 5.5. ADC0 Equivalent Input Circuit ..... 35
Figure 5.6. ADC Window Compare Example: Right-Justified Data ..... 46
Figure 5.7. ADC Window Compare Example: Left-Justified Data ..... 46
Figure 6.1. Minimum VDD Monitor Threshold vs. System Clock Frequency ..... 50
Figure 6.2. ADC0 Multiplexer Block Diagram ..... 58
Figure 6.3. Temperature Sensor Transfer Function ..... 60
Figure 7.1. Voltage Reference Functional Block Diagram ..... 61
Figure 8.1. Comparator Functional Block Diagram ..... 63
Figure 8.2. Comparator Hysteresis Plot ..... 64
Figure 8.3. Comparator Input Multiplexer Block Diagram ..... 69
Figure 9.1. External Capacitors for Voltage Regulator Input/Output—Regulator Enabled ..... 72
Figure 9.2. External Capacitors for Voltage Regulator Input/Output—Regulator Disabled ..... 73
Figure 10.1. CIP-51 Block Diagram ..... 75
Figure 11.1. C8051F54x Memory Map ..... 85
Figure 11.2. Flash Program Memory Map ..... 86
Figure 12.1. SFR Page Stack ..... 90
Figure 12.2. SFR Page Stack While Using SFR Page 0x0 To Access SMB0ADR ..... 91
Figure 12.3. SFR Page Stack After SPI0 Interrupt Occurs ..... 92
Figure 12.4. SFR Page Stack Upon PCA Interrupt Occurring During a SPI0 ISR ..... 93
Figure 12.5. SFR Page Stack Upon Return From PCA Interrupt ..... 94
Figure 12.6. SFR Page Stack Upon Return From SPI0 Interrupt ..... 95
Figure 14.1. Flash Program Memory Map ..... 119
Figure 16.1. Reset Sources ..... 129
Figure 16.2. Power-On and VDD Monitor Reset Timing ..... 130
Figure 17.1. Oscillator Options ..... 135
Figure 17.2. Example Clock Multiplier Output ..... 140



Downloaded from Arrow.com.

Figure 17.3. External 32.768 kHz Quartz Crystal Oscillator Connection Diagram 145
Figure 18.1. Port I/O Functional Block Diagram ..... 147
Figure 18.2. Port I/O Cell Block Diagram ..... 148
Figure 18.3. Peripheral Availability on Port I/O Pins ..... 151
Figure 18.4. Crossbar Priority Decoder in Example Configuration ..... 152
Figure 19.1. LIN Block Diagram ..... 170
Figure 20.1. SMBus Block Diagram ..... 187
Figure 20.2. Typical SMBus Configuration ..... 188
Figure 20.3. SMBus Transaction ..... 189
Figure 20.4. Typical SMBus SCL Generation ..... 191
Figure 20.5. Typical Master Write Sequence ..... 198
Figure 20.6. Typical Master Read Sequence ..... 199
Figure 20.7. Typical Slave Write Sequence ..... 200
Figure 20.8. Typical Slave Read Sequence ..... 201
Figure 21.1. UART0 Block Diagram ..... 205
Figure 21.2. UART0 Timing Without Parity or Extra Bit ..... 207
Figure 21.3. UART0 Timing With Parity ..... 207
Figure 21.4. UART0 Timing With Extra Bit ..... 207
Figure 21.5. Typical UART Interconnect Diagram ..... 208
Figure 21.6. UART Multi-Processor Mode Interconnect Diagram ..... 209
Figure 22.1. SPI Block Diagram ..... 214
Figure 22.2. Multiple-Master Mode Connection Diagram ..... 217
Figure 22.3. 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram 217
Figure 22.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram 217
Figure 22.5. Master Mode Data/Clock Timing ..... 219
Figure 22.6. Slave Mode Data/Clock Timing (CKPHA = 0) ..... 220
Figure 22.7. Slave Mode Data/Clock Timing (CKPHA = 1) ..... 220
Figure 22.8. SPI Master Timing (CKPHA = 0) ..... 224
Figure 22.9. SPI Master Timing (CKPHA = 1) ..... 224
Figure 22.10. SPI Slave Timing (CKPHA = 0) ..... 225
Figure 22.11. SPI Slave Timing (CKPHA = 1) ..... 225
Figure 23.1. T0 Mode 0 Block Diagram ..... 230
Figure 23.2. T0 Mode 2 Block Diagram ..... 231
Figure 23.3. T0 Mode 3 Block Diagram ..... 232
Figure 23.4. Timer 2 16-Bit Mode Block Diagram ..... 237
Figure 23.5. Timer 2 8-Bit Mode Block Diagram ..... 238
Figure 23.6. Timer 2 External Oscillator Capture Mode Block Diagram ..... 239
Figure 23.7. Timer 3 16-Bit Mode Block Diagram ..... 243
Figure 23.8. Timer 3 8-Bit Mode Block Diagram ..... 244
Figure 23.9. Timer 3 External Oscillator Capture Mode Block Diagram ..... 245
Figure 24.1. PCA Block Diagram ..... 249
Figure 24.2. PCA Counter/Timer Block Diagram ..... 251



Downloaded from Arrow.com.

Figure 24.3. PCA Interrupt Block Diagram ..... 252
Figure 24.4. PCA Capture Mode Diagram ..... 254
Figure 24.5. PCA Software Timer Mode Diagram ..... 255
Figure 24.6. PCA High-Speed Output Mode Diagram ..... 256
Figure 24.7. PCA Frequency Output Mode ..... 257
Figure 24.8. PCA 8-Bit PWM Mode Diagram ..... 258
Figure 24.9. PCA 9, 10 and 11-Bit PWM Mode Diagram ..... 259
Figure 24.10. PCA 16-Bit PWM Mode ..... 260
Figure 24.11. PCA Module 2 with Watchdog Timer Enabled ..... 261
Figure 25.1. Typical C2 Pin Sharing ..... 272



Downloaded from Arrow.com.

## List of Tables

Table 2.1. Product Selection Guide ..... 17
Table 3.1. Pin Definitions for the C8051F54x ..... 18
Table 4.1. QFP-32 Package Dimensions ..... 23
Table 4.2. QFP-32 Landing Diagram Dimensions ..... 24
Table 4.3. QFN-32 Package Dimensions ..... 25
Table 4.4. QFN-32 Landing Diagram Dimensions ..... 26
Table 4.5. QFN-24 Package Dimensions ..... 27
Table 4.6. QFN-24 Landing Diagram Dimensions ..... 28
Table 6.1. Absolute Maximum Ratings ..... 47
Table 6.2. Global Electrical Characteristics ..... 48
Table 6.3. Port I/O DC Electrical Characteristics ..... 51
Table 6.4. Reset Electrical Characteristics ..... 52
Table 6.5. Flash Electrical Characteristics ..... 52
Table 6.6. Internal High-Frequency Oscillator Electrical Characteristics ..... 53
Table 6.7. Clock Multiplier Electrical Specifications ..... 54
Table 6.8. Voltage Regulator Electrical Characteristics ..... 54
Table 6.9. ADC0 Electrical Characteristics ..... 55
Table 6.10. Temperature Sensor Electrical Characteristics ..... 56
Table 6.11. Voltage Reference Electrical Characteristics ..... 56
Table 6.12. Comparator 0 and Comparator 1 Electrical Characteristics ..... 57
Table 10.1. CIP-51 Instruction Set Summary ..... 77
Table 12.1. Special Function Register (SFR) Memory Map for Pages 0x0 and 0xF ..... 100
Table 12.2. Special Function Registers ..... 101
Table 13.1. Interrupt Summary ..... 107
Table 14.1. Flash Security Summary ..... 120
Table 18.1. Port I/O Assignment for Analog Functions ..... 149
Table 18.2. Port I/O Assignment for Digital Functions ..... 149
Table 18.3. Port I/O Assignment for External Digital Event Capture Functions ..... 150
Table 19.1. Baud Rate Calculation Variable Ranges ..... 171
Table 19.2. Manual Baud Rate Parameters Examples ..... 173
Table 19.3. Autobaud Parameters Examples ..... 174
Table 19.4. LIN Registers* (Indirectly Addressable) ..... 179
Table 20.1. SMBus Clock Source Selection ..... 191
Table 20.2. Minimum SDA Setup and Hold Times ..... 192
Table 20.3. Sources for Hardware Changes to SMB0CN ..... 196
Table 20.4. SMBus Status Decoding ..... 202
Table 21.1. Baud Rate Generator Settings for Standard Baud Rates ..... 206
Table 22.1. SPI Slave Timing Parameters ..... 226
Table 24.1. PCA Timebase Input Options ..... 250
Table 24.2. PCA0CPM and PCA0PWM Bit Settings for PCA Capture/Compare Modules ..... 253
Table 24.3. Watchdog Timer Timeout Intervals1 ..... 262



Downloaded from Arrow.com.

## List of Registers

SFR Definition 5.4. ADC0CF: ADC0 Configuration ..... 40
SFR Definition 5.5. ADC0H: ADC0 Data Word MSB ..... 41
SFR Definition 5.6. ADC0L: ADC0 Data Word LSB ..... 41
SFR Definition 5.7. ADC0CN: ADC0 Control ..... 42
SFR Definition 5.8. ADC0TK: ADC0 Tracking Mode Select ..... 43
SFR Definition 5.9. ADC0GTH: ADC0 Greater-Than Data High Byte ..... 44
SFR Definition 5.10. ADC0GTL: ADC0 Greater-Than Data Low Byte ..... 44
SFR Definition 5.11. ADC0LTH: ADC0 Less-Than Data High Byte ..... 45
SFR Definition 5.12. ADC0LTL: ADC0 Less-Than Data Low Byte ..... 45
SFR Definition 6.3. ADC0MX: ADC0 Channel Select ..... 59
SFR Definition 7.1. REF0CN: Reference Control ..... 62
SFR Definition 8.1. CPT0CN: Comparator0 Control ..... 65
SFR Definition 8.2. CPT0MD: Comparator0 Mode Selection ..... 66
SFR Definition 8.3. CPT1CN: Comparator1 Control ..... 67
SFR Definition 8.4. CPT1MD: Comparator1 Mode Selection ..... 68
SFR Definition 8.5. CPT0MX: Comparator0 MUX Selection ..... 70
SFR Definition 8.6. CPT1MX: Comparator1 MUX Selection ..... 71
SFR Definition 9.1. REG0CN: Regulator Control ..... 73
SFR Definition 10.1. DPL: Data Pointer Low Byte ..... 81
SFR Definition 10.2. DPH: Data Pointer High Byte ..... 81
SFR Definition 10.3. SP: Stack Pointer ..... 82
SFR Definition 10.4. ACC: Accumulator ..... 82
SFR Definition 10.5. B: B Register ..... 82
SFR Definition 10.6. PSW: Program Status Word ..... 83
SFR Definition 10.7. SNn: Serial Number n ..... 84
SFR Definition 11.1. EMI0CN: External Memory Interface Control ..... 88
SFR Definition 12.1. SFR0CN: SFR Page Control ..... 96
SFR Definition 12.2. SFRPAGE: SFR Page ..... 97
SFR Definition 12.3. SFRNEXT: SFR Next ..... 98
SFR Definition 12.4. SFRLAST: SFR Last ..... 99
SFR Definition 13.1. IE: Interrupt Enable ..... 109
SFR Definition 13.2. IP: Interrupt Priority ..... 110
SFR Definition 13.3. EIE1: Extended Interrupt Enable 1 ..... 111
SFR Definition 13.4. EIP1: Extended Interrupt Priority 1 ..... 112
SFR Definition 13.5. EIE2: Extended Interrupt Enable 2 ..... 113
SFR Definition 13.6. EIP2: Extended Interrupt Priority Enabled 2 ..... 114
SFR Definition 13.7. IT01CF: INT0/INT1 Configuration ..... 116
SFR Definition 14.1. PSCTL: Program Store R/W Control ..... 122
SFR Definition 14.2. FLKEY: Flash Lock and Key ..... 123
SFR Definition 14.3. FLSCL: Flash Scale ..... 124
SFR Definition 14.4. CCH0CN: Cache Control ..... 125
SFR Definition 14.5. ONESHOT: Flash Oneshot Period ..... 125
SFR Definition 15.1. PCON: Power Control ..... 128



Downloaded from Arrow.com.

SFR Definition 16.1. VDM0CN: VDD Monitor Control ..... 132
SFR Definition 16.2. RSTSRC: Reset Source ..... 134
SFR Definition 17.1. CLKSEL: Clock Select ..... 136
SFR Definition 17.2. OSCICN: Internal Oscillator Control ..... 138
SFR Definition 17.3. OSCICRS: Internal Oscillator Coarse Calibration ..... 139
SFR Definition 17.4. OSCIFIN: Internal Oscillator Fine Calibration ..... 139
SFR Definition 17.5. CLKMUL: Clock Multiplier ..... 141
SFR Definition 17.6. OSCXCN: External Oscillator Control ..... 143
SFR Definition 18.1. XBR0: Port I/O Crossbar Register 0 ..... 154
SFR Definition 18.2. XBR1: Port I/O Crossbar Register 1 ..... 155
SFR Definition 18.3. XBR2: Port I/O Crossbar Register 1 ..... 156
SFR Definition 18.4. P0MASK: Port 0 Mask Register ..... 157
SFR Definition 18.5. P0MAT: Port 0 Match Register ..... 157
SFR Definition 18.6. P1MASK: Port 1 Mask Register ..... 158
SFR Definition 18.7. P1MAT: Port 1 Match Register ..... 158
SFR Definition 18.8. P2MASK: Port 2 Mask Register ..... 159
SFR Definition 18.9. P2MAT: Port 2 Match Register ..... 159
SFR Definition 18.10. P3MASK: Port 3 Mask Register ..... 160
SFR Definition 18.11. P3MAT: Port 3 Match Register ..... 160
SFR Definition 18.12. P0: Port 0 ..... 161
SFR Definition 18.13. P0MDIN: Port 0 Input Mode ..... 162
SFR Definition 18.14. P0MDOUT: Port 0 Output Mode ..... 162
SFR Definition 18.15. P0SKIP: Port 0 Skip ..... 163
SFR Definition 18.16. P1: Port 1 ..... 163
SFR Definition 18.17. P1MDIN: Port 1 Input Mode ..... 164
SFR Definition 18.18. P1MDOUT: Port 1 Output Mode ..... 164
SFR Definition 18.19. P1SKIP: Port 1 Skip ..... 165
SFR Definition 18.20. P2: Port 2 ..... 165
SFR Definition 18.21. P2MDIN: Port 2 Input Mode ..... 166
SFR Definition 18.22. P2MDOUT: Port 2 Output Mode ..... 166
SFR Definition 18.23. P2SKIP: Port 2 Skip ..... 167
SFR Definition 18.24. P3: Port 3 ..... 167
SFR Definition 18.25. P3MDIN: Port 3 Input Mode ..... 168
SFR Definition 18.26. P3MDOUT: Port 3 Output Mode ..... 168
SFR Definition 18.27. P3SKIP: Port 3Skip ..... 169
SFR Definition 19.1. LIN0ADR: LIN0 Indirect Address Register ..... 177
SFR Definition 19.2. LIN0DAT: LIN0 Indirect Data Register ..... 177
SFR Definition 19.3. LIN0CF: LIN0 Control Mode Register ..... 178
SFR Definition 20.1. SMB0CF: SMBus Clock/Configuration ..... 193
SFR Definition 20.2. SMB0CN: SMBus Control ..... 195
SFR Definition 20.3. SMB0DAT: SMBus Data ..... 197
SFR Definition 21.1. SCON0: Serial Port 0 Control ..... 210
SFR Definition 21.2. SMOD0: Serial Port 0 Control ..... 211
SFR Definition 21.3. SBUF0: Serial (UART0) Port Data Buffer ..... 212
SFR Definition 21.4. SBCON0: UART0 Baud Rate Generator Control ..... 212



Downloaded from Arrow.com.

SFR Definition 21.6. SBRLL0: UART0 Baud Rate Generator Reload Low Byte ..... 213
SFR Definition 21.5. SBRLH0: UART0 Baud Rate Generator Reload High Byte ..... 213
SFR Definition 22.1. SPI0CFG: SPI0 Configuration ..... 221
SFR Definition 22.2. SPI0CN: SPI0 Control ..... 222
SFR Definition 22.3. SPI0CKR: SPI0 Clock Rate ..... 223
SFR Definition 22.4. SPI0DAT: SPI0 Data ..... 223
SFR Definition 23.1. CKCON: Clock Control ..... 228
SFR Definition 23.2. TCON: Timer Control ..... 233
SFR Definition 23.3. TMOD: Timer Mode ..... 234
SFR Definition 23.4. TL0: Timer 0 Low Byte ..... 235
SFR Definition 23.5. TL1: Timer 1 Low Byte ..... 235
SFR Definition 23.6. TH0: Timer 0 High Byte ..... 236
SFR Definition 23.7. TH1: Timer 1 High Byte ..... 236
SFR Definition 23.8. TMR2CN: Timer 2 Control ..... 240
SFR Definition 23.9. TMR2RLL: Timer 2 Reload Register Low Byte ..... 241
SFR Definition 23.10. TMR2RLH: Timer 2 Reload Register High Byte ..... 241
SFR Definition 23.11. TMR2L: Timer 2 Low Byte ..... 242
SFR Definition 23.12. TMR2H Timer 2 High Byte ..... 242
SFR Definition 23.13. TMR3CN: Timer 3 Control ..... 246
SFR Definition 23.14. TMR3RLL: Timer 3 Reload Register Low Byte ..... 247
SFR Definition 23.15. TMR3RLH: Timer 3 Reload Register High Byte ..... 247
SFR Definition 23.16. TMR3L: Timer 3 Low Byte ..... 248
SFR Definition 23.17. TMR3H Timer 3 High Byte ..... 248
SFR Definition 24.1. PCA0CN: PCA Control ..... 263
SFR Definition 24.2. PCA0MD: PCA Mode ..... 264
SFR Definition 24.3. PCA0PWM: PCA PWM Configuration ..... 265
SFR Definition 24.4. PCA0CPMn: PCA Capture/Compare Mode ..... 266
SFR Definition 24.5. PCA0L: PCA Counter/Timer Low Byte ..... 267
SFR Definition 24.6. PCA0H: PCA Counter/Timer High Byte ..... 267
SFR Definition 24.7. PCA0CPLn: PCA Capture Module Low Byte ..... 268
SFR Definition 24.8. PCA0CPHn: PCA Capture Module High Byte ..... 268



Downloaded from Arrow.com.

# 1. System Overview

C8051F54x devices are fully integrated mixed-signal System-on-a-Chip MCUs. Highlighted features are listed below. Refer to Table 2.1 for specific product feature selection and part ordering numbers.

- High-speed pipelined 8051-compatible microcontroller core (up to 50 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- LIN 2.1 peripheral (fully backwards compatible, master and slave modes) (C8051F540/2/4/6)
- True 12-bit 200 ksps 32-channel single-ended ADC with analog multiplexer
- Precision programmable 24 MHz internal oscillator that is within ±0.5% across the temperature range and for VDD voltages greater than or equal to the on-chip voltage regulator minimum output at the low setting. The oscillator is within ±1.0% for VDD voltages below this minimum output setting.
- On-chip Clock Multiplier to reach up to 50 MHz
- 16 kB (C8051F540/1/2/3) or 8 kB (C8051F544/5/6/7) of on-chip Flash memory
- 1280 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 six capture/compare modules and Watchdog Timer function
- On-chip Voltage Regulator
- On-chip Power-On Reset, $V_{DD}$ Monitor, and Temperature Sensor
- On-chip Voltage Comparator
- 25 or 18 Port I/O (5 V push-pull)

With on-chip Voltage Regulator, Power-On Reset, $V_{DD}$ monitor, Watchdog Timer, and clock oscillator, the C8051F54x 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.

The devices are specified for 1.8 V to 5.25 V operation over the automotive temperature range (–40 to +125 °C). The C8051F540/1/4/5 devices are available in 32-pin QFP and QFN packages and the C8051F542/3/6/7 devices are available in 32-pin QFN packages. All package options are lead-free and RoHS compliant. See Table 2.1 for ordering information. Block diagrams are included in Figure 1.1 and Figure 1.2.



Downloaded from Arrow.com.

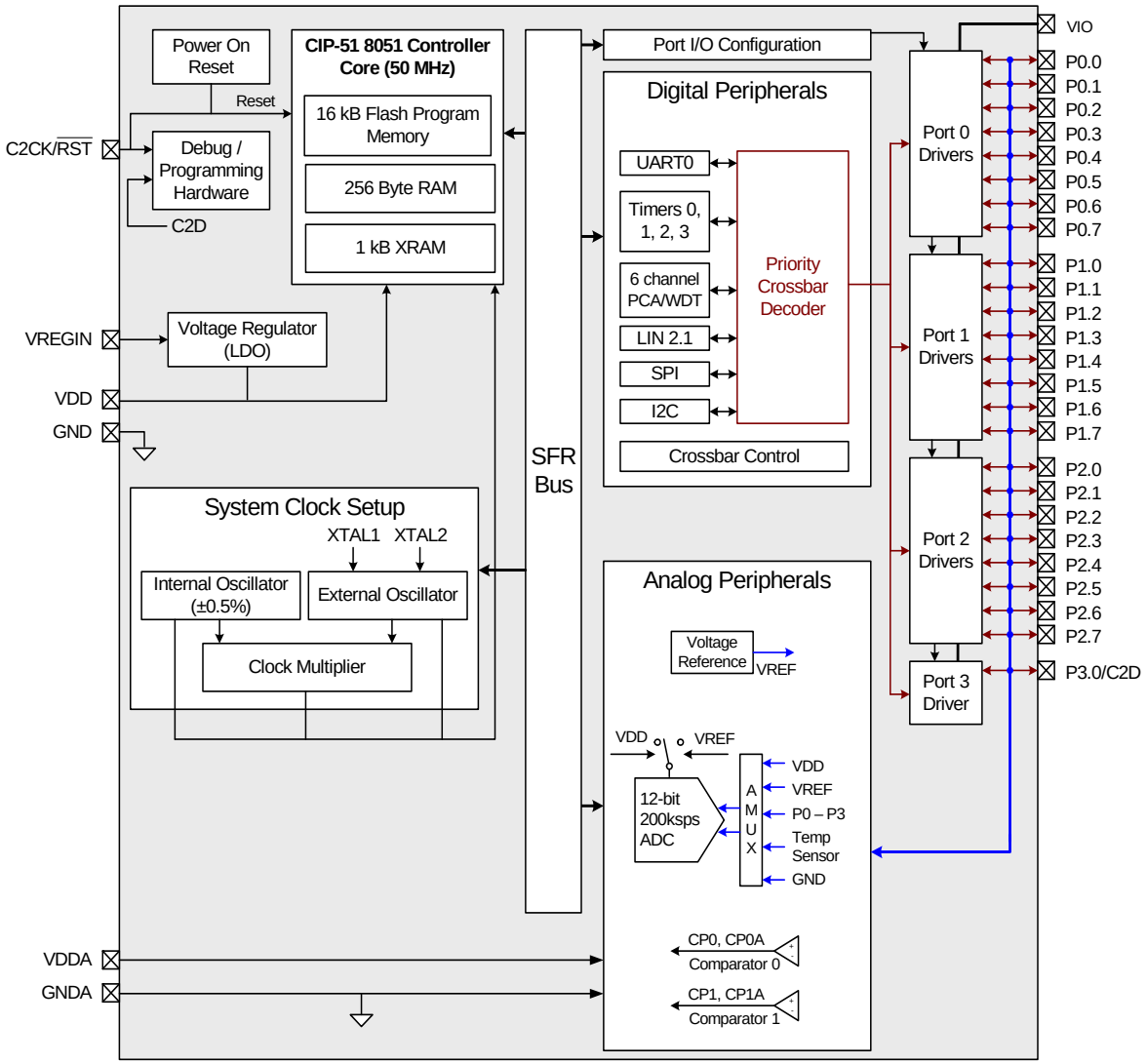



**Figure 1.1. C8051F540/1/4/5 Block Diagram**



Downloaded from Arrow.com.

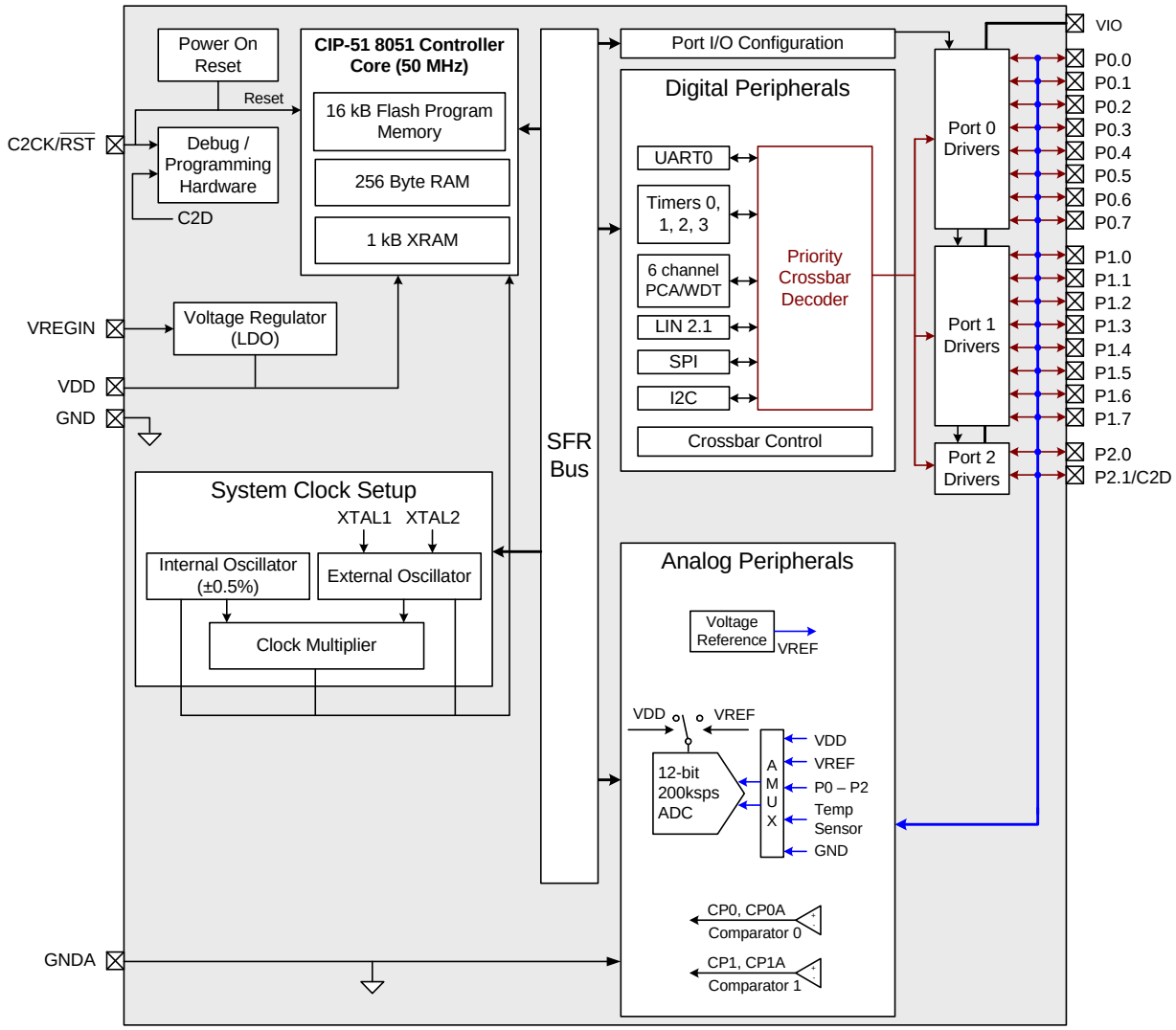



**Figure 1.2. C8051F542/3/6/7 Block Diagram**



Downloaded from Arrow.com.

## 2. Ordering Information

The following features are common to all devices in this family:

- 50 MHz system clock and 50 MIPS throughput (peak)
- 1280 bytes of RAM (256 internal bytes and 1024 XRAM bytes)
- Internal 24 MHz oscillator
- SMBus / I2C, Enhanced SPI, Enhanced UART
- Four Timers
- Six Programmable Counter Array channels
- Internal Voltage Regulator
- 12-bit, 200 ksps ADC, Internal Voltage Reference and Temperature Sensor
- Two Analog Comparators

Table 2.1 shows the features that differentiate the devices in this family.



Downloaded from Arrow.com.

**Table 2.1. Product Selection Guide**

| Ordering Part Number | Flash Memory (kB) | LIN2.1 | Digital Port I/Os | Package |
|---|---|---|---|---|
| C8051F540-IQ | 16 | ✓ | 25 | QFP32 |
| C8051F540-IM | 16 | ✓ | 25 | QFN32 |
| C8051F541-IQ | 16 | — | 25 | QFP32 |
| C8051F541-IM | 16 | — | 25 | QFN32 |
| C8051F542-IM | 16 | ✓ | 18 | QFN24 |
| C8051F543-IM | 16 | — | 18 | QFN24 |
| C8051F544-IQ | 8 | ✓ | 25 | QFP32 |
| C8051F544-IM | 8 | ✓ | 25 | QFN32 |
| C8051F545-IQ | 8 | — | 25 | QFP32 |
| C8051F545-IM | 8 | — | 25 | QFN32 |
| C8051F546-IM | 8 | ✓ | 18 | QFN24 |
| C8051F547-IM | 8 | — | 18 | QFN24 |

**Note:** The suffix of the part number indicates the device rating and the package. All devices are RoHS compliant.

All of these devices are also available in an automotive version. For the automotive version, the -I in the ordering part number is replaced with -A. For example, the automotive version of the C8051F540-IM is the C8051F540-AM.

The -AM and -AQ devices receive full automotive quality production status, including AEC-Q100 qualification, registration with International Material Data System (IMDS) and Part Production Approval Process (PPAP) documentation. PPAP documentation is available at www.silabs.com with a registered and NDA approved user account. The -AM and -AQ devices enable high volume automotive OEM applications with their enhanced testing and processing. Please contact Silicon Labs sales for more information regarding –AM and -AQ devices for your automotive project.



Downloaded from Arrow.com.

# 3. Pin Definitions

**Table 3.1. Pin Definitions for the C8051F54x**

| Name | Pin ‘F540/1/4/5 (32-pin) | Pin ‘F542/3/6/7 (24-pin) | Type | Description |
|---|---|---|---|---|
| VDD | 4 | 3 | | Digital Supply Voltage. Must be connected. |
| GND | 6 | 4 | | Digital Ground. Must be connected. |
| VDDA | 5 | — | | Analog Supply Voltage. Must be connected. Connected internally to VDD on the 24-pin packages. |
| GNDA | 7 | 5 | | Analog Ground. Must be connected. |
| VREGIN | 3 | 2 | | Voltage Regulator Input |
| VIO | 2 | 1 | | Port I/O Supply Voltage. Must be connected. |
| $\overline{\text{RST}}$/<br><br>C2CK | 10 | 8 | D I/O<br><br>D I/O | Device Reset. Open-drain output of internal POR or $V_{DD}$ Monitor.<br><br>Clock signal for the C2 Debug Interface. |
| P2.1/<br><br>C2D | — | 7 | D I/O or A In<br><br>D I/O | Port 2.1. See SFR Definition 18.20 for a description.<br><br>Bi-directional data signal for the C2 Debug Interface. |
| P3.0/<br><br>C2D | 9 | — | D I/O or A In<br><br>D I/O | Port 3.0. See SFR Definition 18.24 for a description.<br><br>Bi-directional data signal for the C2 Debug Interface. |
| P0.0 | 8 | 6 | D I/O or A In | Port 0.0. See SFR Definition 18.12 for a description. |
| P0.1 | 1 | 24 | D I/O or A In | Port 0.1 |
| P0.2 | 32 | 23 | D I/O or A In | Port 0.2 |
| P0.3 | 31 | 22 | D I/O or A In | Port 0.3 |
| P0.4 | 30 | 21 | D I/O or A In | Port 0.4 |
| P0.5 | 29 | 20 | D I/O or A In | Port 0.5 |
| P0.6 | 28 | 19 | D I/O or A In | Port 0.6 |
| P0.7 | 27 | 18 | D I/O or A In | Port 0.7 |
| P1.0 | 26 | 17 | D I/O or A In | Port 1.0. See SFR Definition 18.16 for a description. |
| P1.1 | 25 | 16 | D I/O or A In | Port 1.1. |
| P1.2 | 24 | 15 | D I/O or A In | Port 1.2. |



Downloaded from Arrow.com.

**Table 3.1. Pin Definitions for the C8051F54x (Continued)**

| Name | Pin ‘F540/1/4/5 (32-pin) | Pin ‘F542/3/6/7 (24-pin) | Type | Description |
|---|---|---|---|---|
| P1.3 | 23 | 14 | D I/O or A In | Port 1.3. |
| P1.4 | 22 | 13 | D I/O or A In | Port 1.4. |
| P1.5 | 21 | 12 | D I/O or A In | Port 1.5. |
| P1.6 | 20 | 11 | D I/O or A In | Port 1.6. |
| P1.7 | 19 | 10 | D I/O or A In | Port 1.7. |
| P2.0 | 18 | 9 | D I/O or A In | Port 2.0. See SFR Definition 18.20 for a description. |
| P2.1 | 17 | — | D I/O or A In | Port 2.1. |
| P2.2 | 16 | — | D I/O or A In | Port 2.2. |
| P2.3 | 15 | — | D I/O or A In | Port 2.3. |
| P2.4 | 14 | — | D I/O or A In | Port 2.4. |
| P2.5 | 13 | — | D I/O or A In | Port 2.5. |
| P2.6 | 12 | — | D I/O or A In | Port 2.6. |
| P2.7 | 11 | — | D I/O or A In | Port 2.7. |



Downloaded from Arrow.com.

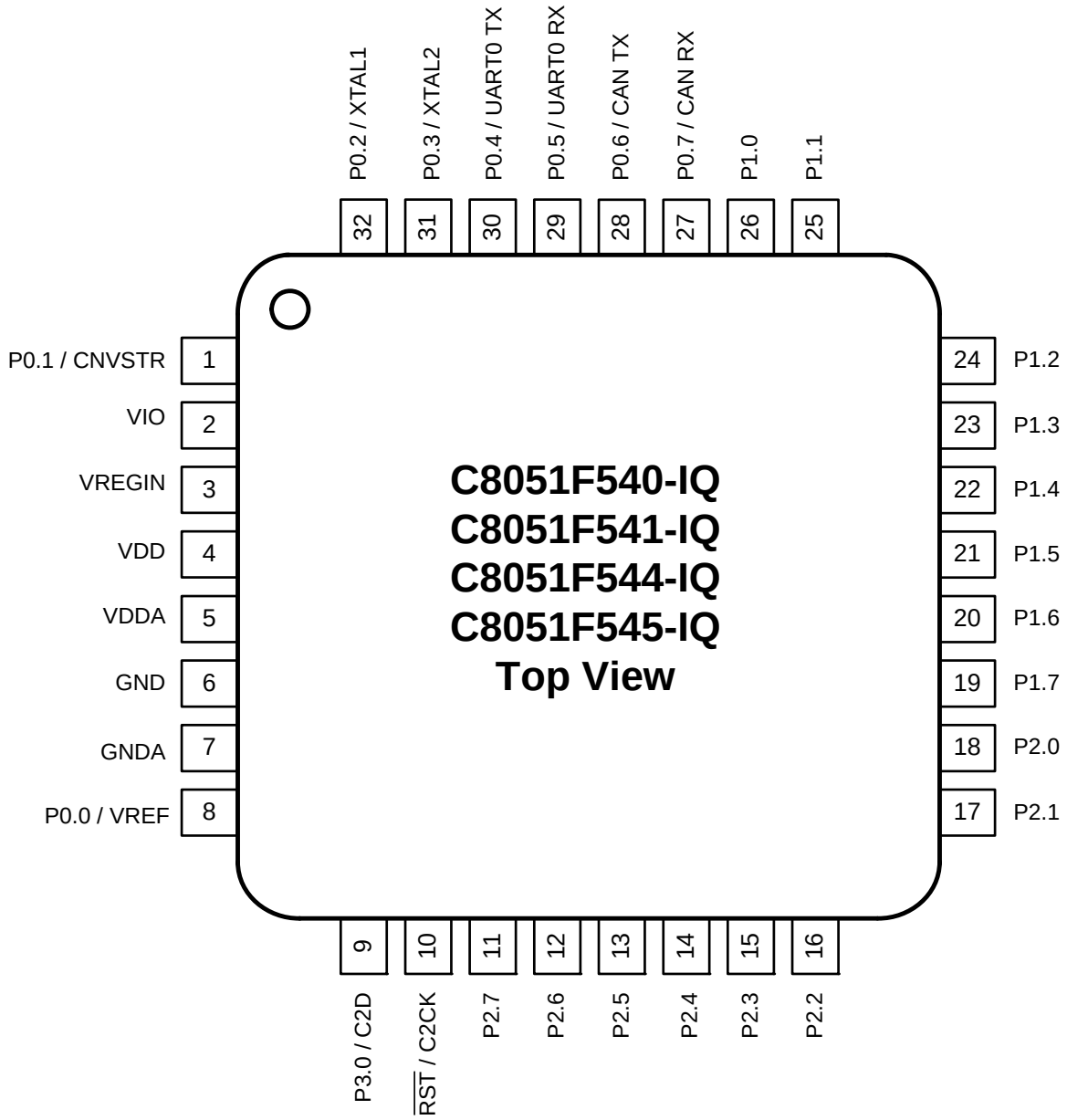



**Figure 3.1. QFP-32 Pinout Diagram (Top View)**



Downloaded from Arrow.com.

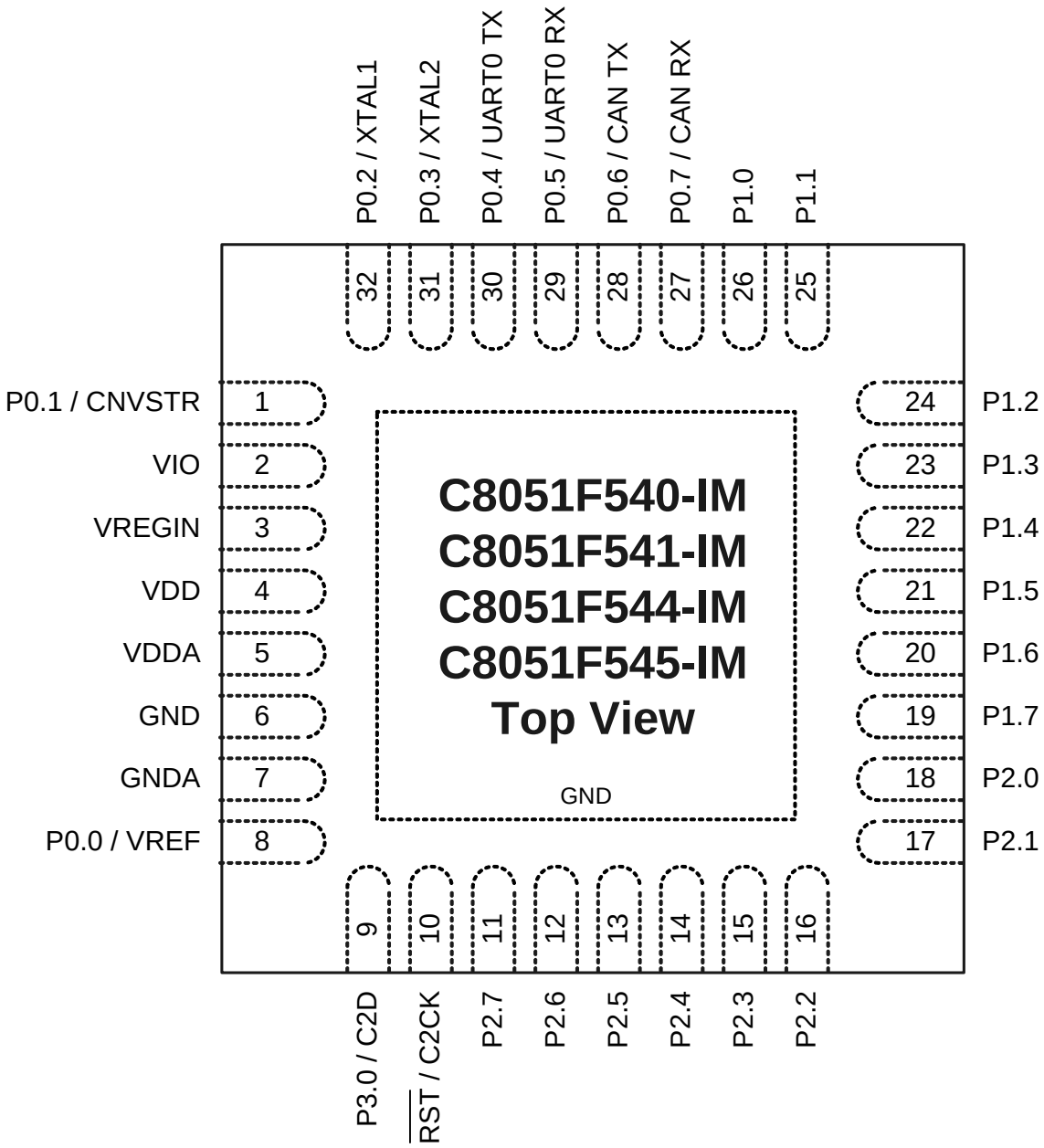



**Figure 3.2. QFN-32 Pinout Diagram (Top View)**



Downloaded from Arrow.com.

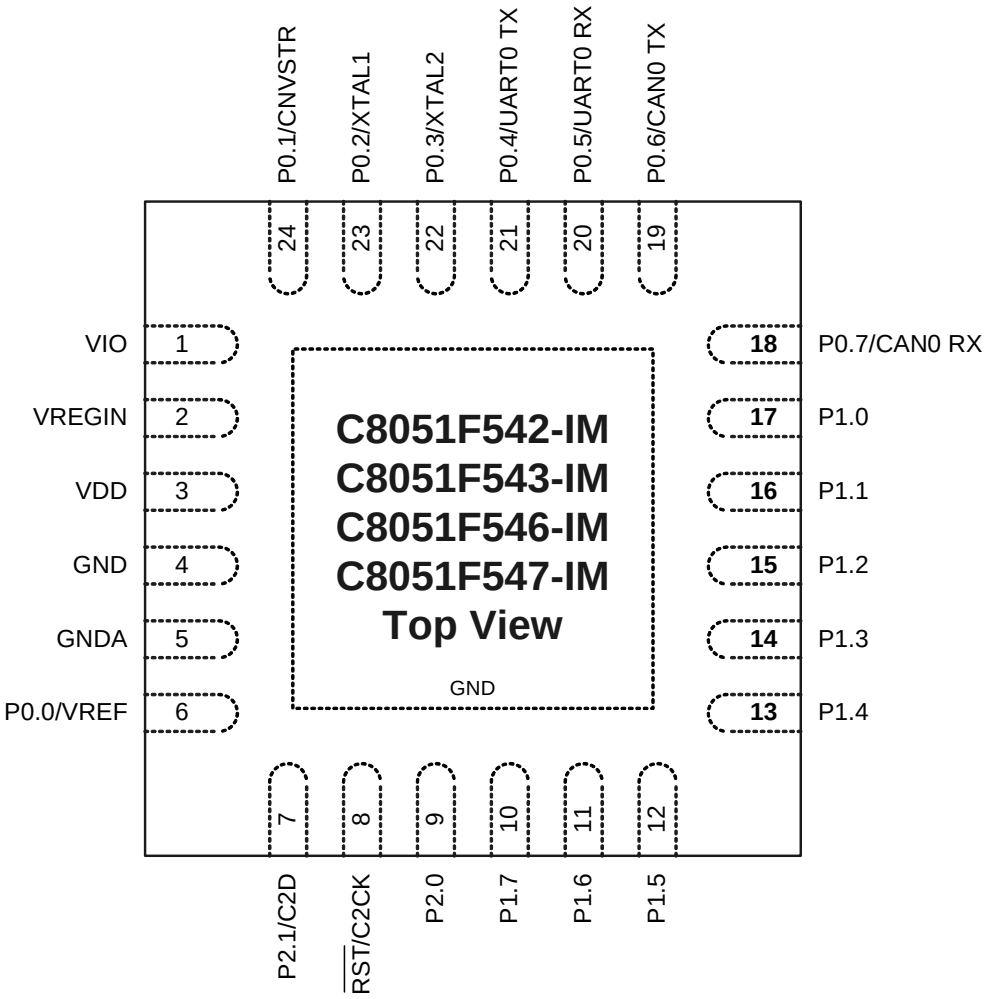



**Figure 3.3. QFN-24 Pinout Diagram (Top View)**



Downloaded from Arrow.com.

# 4. Package Specifications

## 4.1. QFP-32 Package Specifications

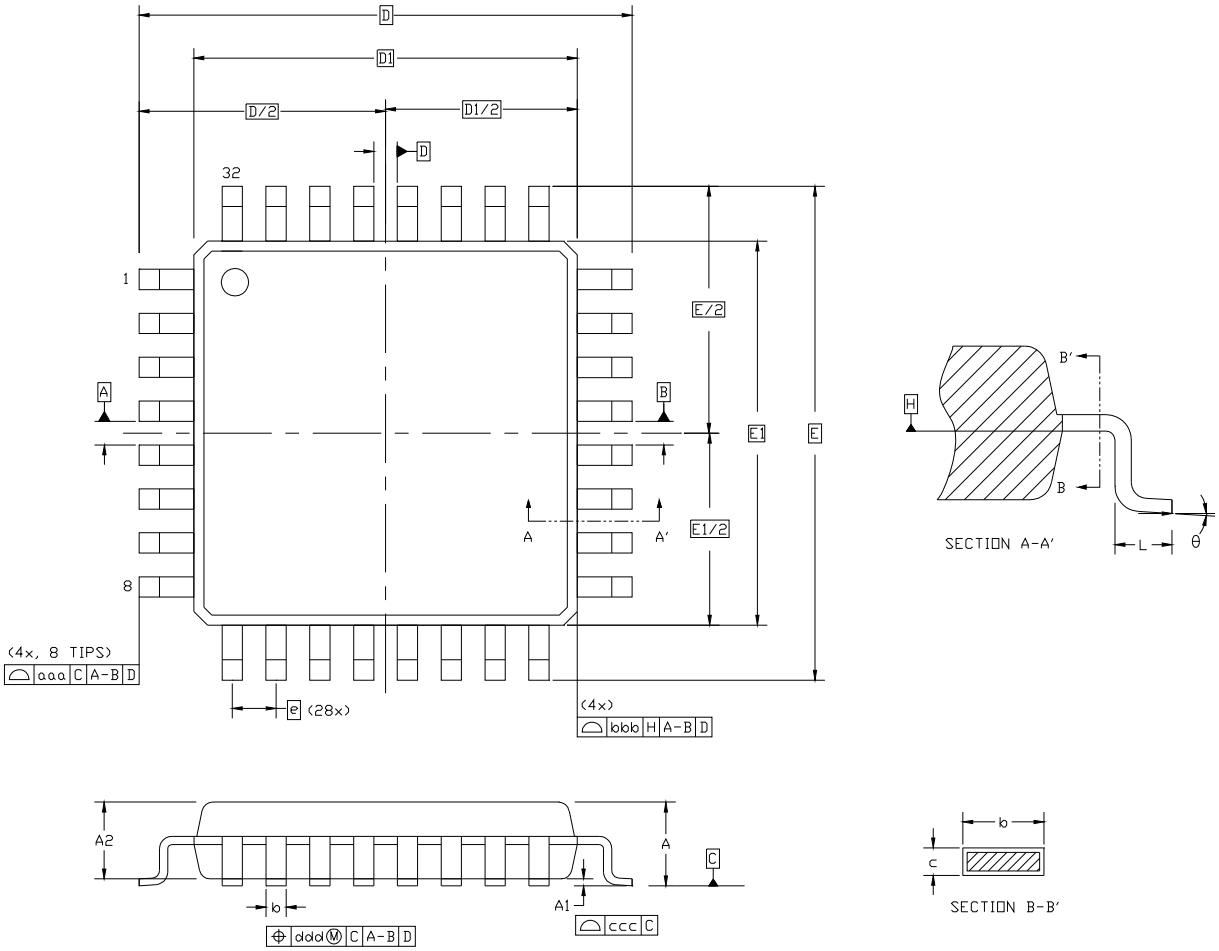



**Figure 4.1. QFP-32 Package Drawing**

**Table 4.1. QFP-32 Package Dimensions**

| Dimension | Min | Typ | Max |
|---|---|---|---|
| A | — | — | 1.60 |
| A1 | 0.05 | — | 0.15 |
| A2 | 1.35 | 1.40 | 1.45 |
| b | 0.30 | 0.37 | 0.45 |
| c | 0.09 | — | 0.20 |
| D | 9.00 BSC. | | |
| D1 | 7.00 BSC. | | |
| e | 0.80 BSC. | | |

| Dimension | Min | Typ | Max |
|---|---|---|---|
| E | 9.00 BSC. | | |
| E1 | 7.00 BSC. | | |
| L | 0.45 | 0.60 | 0.75 |
| aaa | 0.20 | | |
| bbb | 0.20 | | |
| ccc | 0.10 | | |
| ddd | 0.20 | | |
| θ | 0° | 3.5° | 7° |

**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 outline MS-026, variation BBA.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.



Downloaded from Arrow.com.

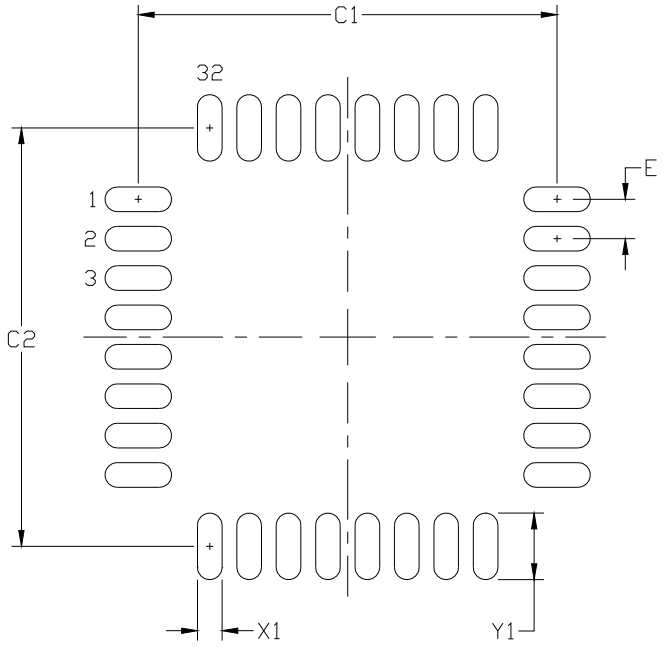



**Figure 4.2. QFP-32 Landing Diagram**

**Table 4.2. QFP-32 Landing Diagram Dimensions**

| Dimension | Min | Max | Dimension | Min | Max |
|---|---|---|---|---|---|
| C1 | 8.40 | 8.50 | X1 | 0.40 | 0.50 |
| C2 | 8.40 | 8.50 | Y1 | 1.25 | 1.35 |
| E | 0.80 BSC | | | | |

**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 60 µm 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 0.125 mm (5 mils).
6. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.

**Card Assembly**

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



Downloaded from Arrow.com.

## 4.2. QFN-32 Package Specifications

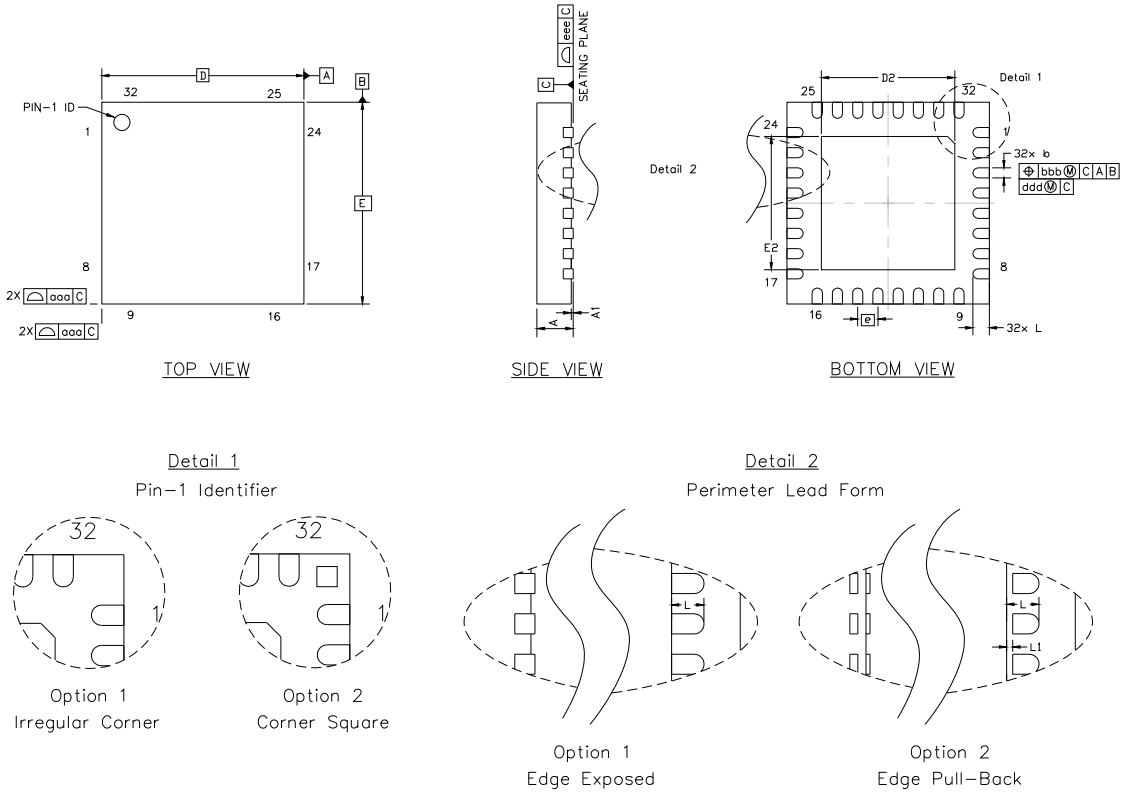



**Figure 4.3. QFN-32 Package Drawing**

**Table 4.3. QFN-32 Package Dimensions**

| Dimension | Min | Typ | Max |
|---|---|---|---|
| A | 0.80 | 0.9 | 1.00 |
| A1 | 0.00 | 0.02 | 0.05 |
| b | 0.18 | 0.25 | 0.30 |
| D | | 5.00 BSC. | |
| D2 | 3.20 | 3.30 | 3.40 |
| e | | 0.50 BSC. | |
| E | | 5.00 BSC. | |

| Dimension | Min | Typ | Max |
|---|---|---|---|
| E2 | 3.20 | 3.30 | 3.40 |
| L | 0.30 | 0.40 | 0.50 |
| L1 | 0.00 | — | 0.15 |
| aaa | — | — | 0.15 |
| bbb | — | — | 0.15 |
| ddd | — | — | 0.05 |
| eee | — | — | 0.08 |

**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 VHHD except for custom features D2, E2, and L which are toleranced per supplier designation.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.



Downloaded from Arrow.com.

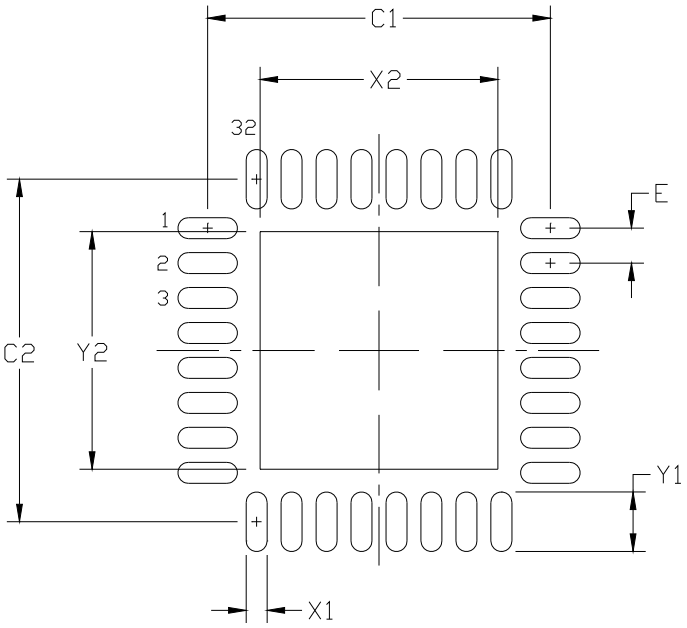



**Figure 4.4. QFN-32 Landing Diagram**

**Table 4.4. QFN-32 Landing Diagram Dimensions**

| Dimension | Min | Max | | Dimension | Min | Max |
|---|---|---|---|---|---|---|
| C1 | 4.80 | 4.90 | | X2 | 3.20 | 3.40 |
| C2 | 4.80 | 4.90 | | Y1 | 0.75 | 0.85 |
| e | 0.50 BSC | | | Y2 | 3.20 | 3.40 |
| X1 | 0.20 | 0.30 | | | | |

**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 60 µm 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 0.125 mm (5 mils).
6. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.
7. A 3x3 array of 1.0 mm openings on a 1.20 mm pitch should be used for the center ground 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-020 specification for Small Body Components.



Downloaded from Arrow.com.

## 4.3. QFN-24 Package Specifications

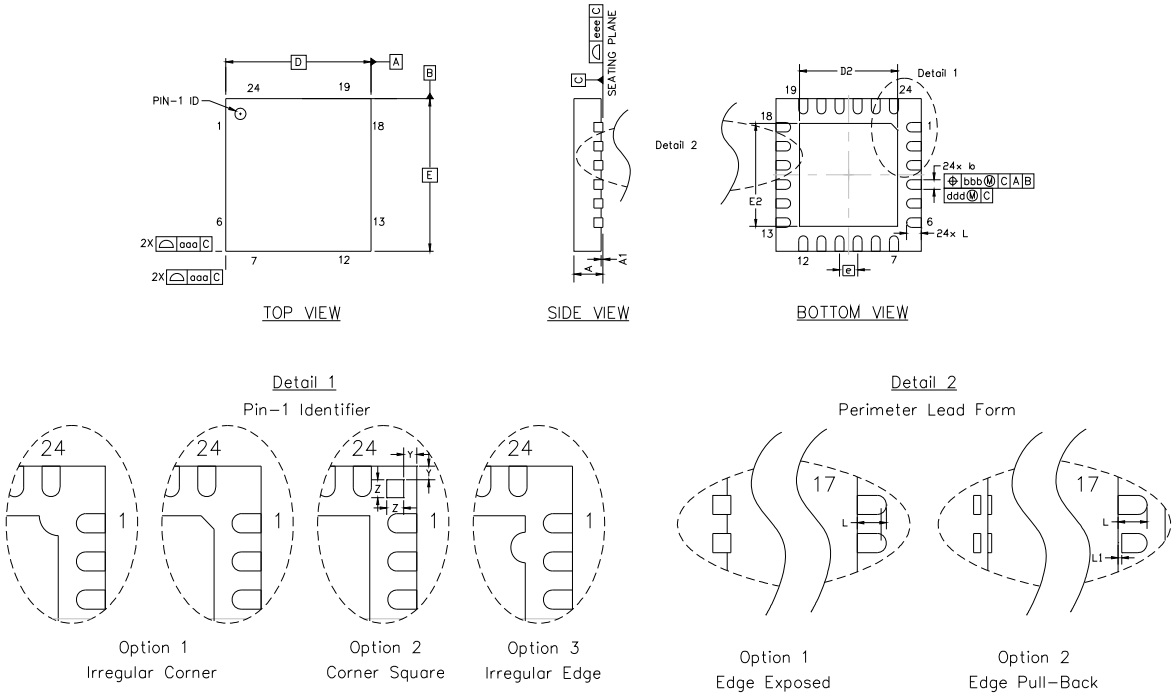



**Figure 4.5. QFN-24 Package Drawing**

**Table 4.5. 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-020 specification for Small Body Components.



Downloaded from Arrow.com.

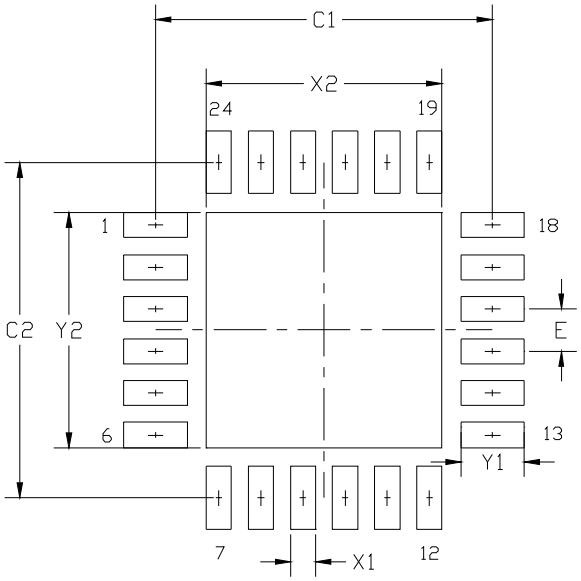



**Figure 4.6. QFN-24 Landing Diagram**

**Table 4.6. QFN-24 Landing Diagram Dimensions**

| Dimension | Min | Max | Dimension | Min | Max |
|---|---|---|---|---|---|
| C1 | 3.90 | 4.00 | X2 | 2.70 | 2.80 |
| C2 | 3.90 | 4.00 | Y1 | 0.65 | 0.75 |
| E | 0.50 BSC | | Y2 | 2.70 | 2.80 |
| X1 | 0.20 | 0.30 | | | |

**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 60 µm 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 0.125 mm (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 1.10 mm x 1.10 mm openings on a 1.30 mm pitch should be used for the center ground 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-020 specification for Small Body Components.



Downloaded from Arrow.com.

## 5. 12-Bit ADC (ADC0)

The ADC0 on the C8051F54x consists of an analog multiplexer (AMUX0) with 25/18 total input selections and a 200 ksps, 12-bit successive-approximation-register (SAR) ADC with integrated track-and-hold, programmable window detector, programmable attenuation (1:2), and hardware accumulator. The ADC0 subsystem has a special Burst Mode which can automatically enable ADC0, capture and accumulate samples, then place ADC0 in a low power shutdown mode without CPU intervention. The AMUX0, data conversion modes, and window detector are all configurable under software control via the Special Function Registers shows in Figure 5.1. ADC0 inputs are single-ended and may be configured to measure P0.0-P3.7, the Temperature Sensor output, $V_{DD}$, or GND with respect to GND. The voltage reference for ADC0 is selected as described in Section “6.2. Temperature Sensor” on page 60. ADC0 is enabled when the AD0EN bit in the ADC0 Control register (ADC0CN) is set to logic 1, or when performing conversions in Burst Mode. ADC0 is in low power shutdown when AD0EN is logic 0 and no Burst Mode conversions are taking place.

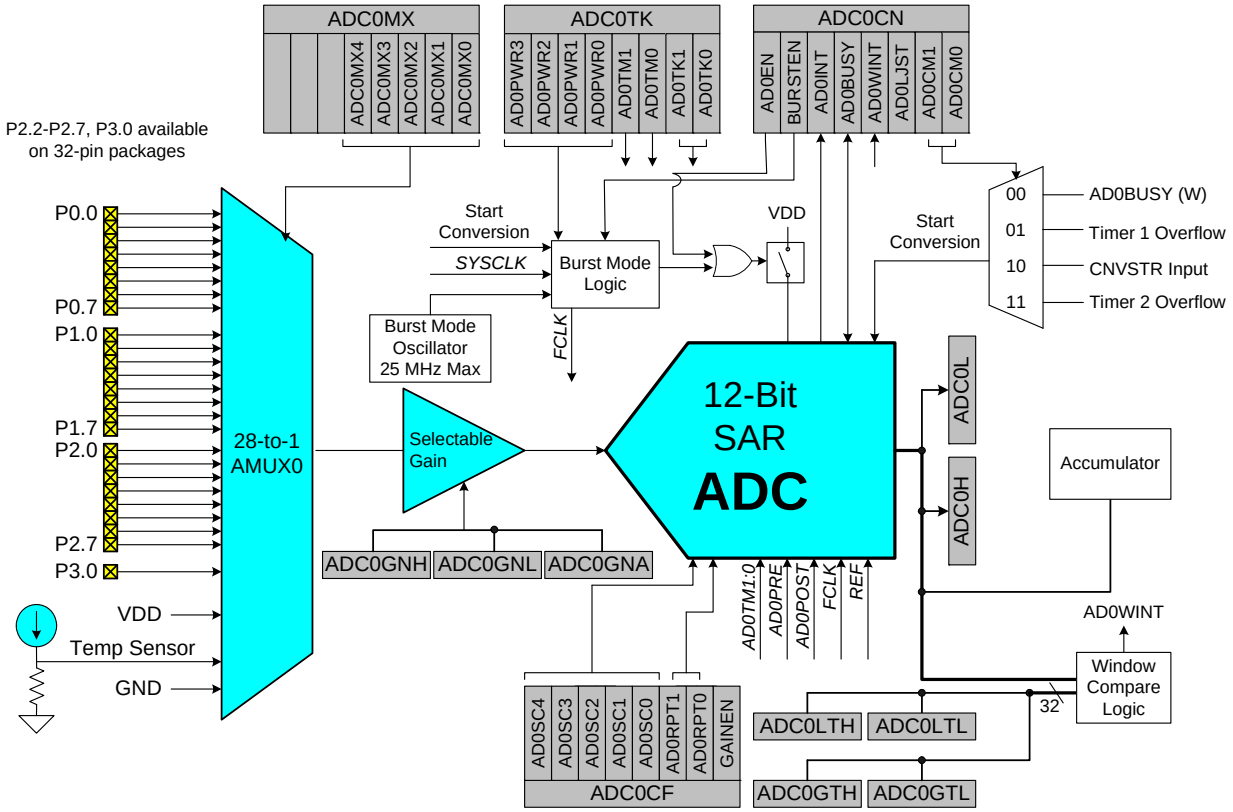



**Figure 5.1. ADC0 Functional Block Diagram**



Downloaded from Arrow.com.

## 5.1. Modes of Operation

In a typical system, ADC0 is configured using the following steps:

1. If a gain adjustment is required, refer to Section “5.3. Selectable Gain” on page 35.
2. Choose the start of conversion source.
3. Choose Normal Mode or Burst Mode operation.
4. If Burst Mode, choose the ADC0 Idle Power State and set the Power-Up Time.
5. Choose the tracking mode. Note that Pre-Tracking Mode can only be used with Normal Mode.
6. Calculate the required settling time and set the post convert-start tracking time using the AD0TK bits.
7. Choose the repeat count.
8. Choose the output word justification (Right-Justified or Left-Justified).
9. Enable or disable the End of Conversion and Window Comparator Interrupts.

### 5.1.1. Starting a Conversion

A conversion can be initiated in one of four ways, depending on the programmed states of the ADC0 Start of Conversion Mode bits (AD0CM1–0) in register ADC0CN. Conversions may be initiated by one of the following:

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

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 overflows are used as the conversion source, Low Byte overflows are used if Timer2 is in 8-bit mode; High byte overflows are used if Timer 2 is in 16-bit mode. See Section “23. Timers” on page 227 for timer configuration.

**Important Note About Using CNVSTR:** The CNVSTR input pin also functions as Port pin P0.1. When the CNVSTR input is used as the ADC0 conversion source, Port pin P0.1 should be skipped by the Digital Crossbar. To configure the Crossbar to skip P0.1, set to 1 Bit1 in register P0SKIP. See Section “18. Port Input/Output” on page 147 for details on Port I/O configuration.

### 5.1.2. Tracking Modes

Each ADC0 conversion must be preceded by a minimum tracking time for the converted result to be accurate. ADC0 has three tracking modes: Pre-Tracking, Post-Tracking, and Dual-Tracking. Pre-Tracking Mode provides the minimum delay between the convert start signal and end of conversion by tracking continuously before the convert start signal. This mode requires software management in order to meet minimum tracking requirements. In Post-Tracking Mode, a programmable tracking time starts after the convert start signal and is managed by hardware. Dual-Tracking Mode maximizes tracking time by tracking before and after the convert start signal. Figure 5.2 shows examples of the three tracking modes.

Pre-Tracking Mode is selected when AD0TM is set to 10b. Conversions are started immediately following the convert start signal. ADC0 is tracking continuously when not performing a conversion. Software must allow at least the minimum tracking time between each end of conversion and the next convert start signal. The minimum tracking time must also be met prior to the first convert start signal after ADC0 is enabled.



Downloaded from Arrow.com.

Post-Tracking Mode is selected when AD0TM is set to 01b. A programmable tracking time based on AD0TK is started immediately following the convert start signal. Conversions are started after the programmed tracking time ends. After a conversion is complete, ADC0 does not track the input. Rather, the sampling capacitor remains disconnected from the input making the input pin high-impedance until the next convert start signal.

Dual-Tracking Mode is selected when AD0TM is set to 11b. A programmable tracking time based on AD0TK is started immediately following the convert start signal. Conversions are started after the programmed tracking time ends. After a conversion is complete, ADC0 tracks continuously until the next conversion is started.

Depending on the output connected to the ADC input, additional tracking time, more than is specified in Table 6.9, may be required after changing MUX settings. See the settling time requirements described in Section “5.2.1. Settling Time Requirements” on page 34.

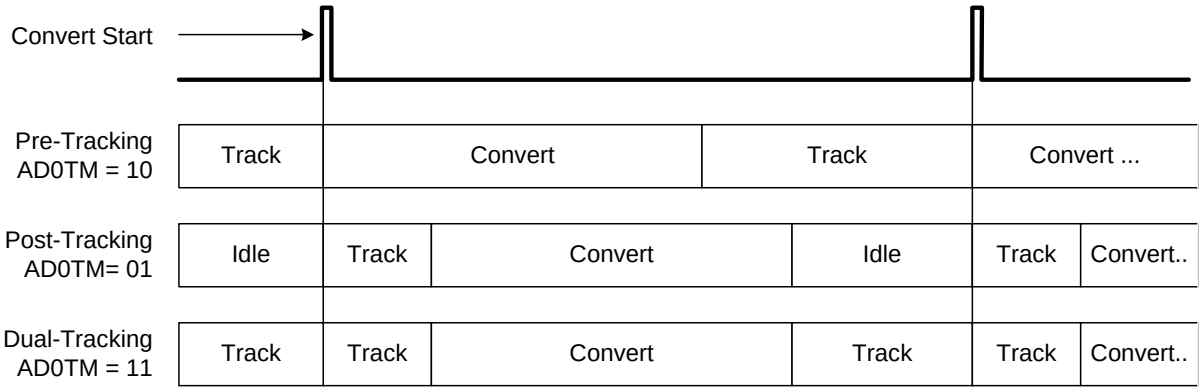



**Figure 5.2. ADC0 Tracking Modes**

### 5.1.3. Timing

ADC0 has a maximum conversion speed specified in Table 6.9. ADC0 is clocked from the ADC0 Subsystem Clock (FCLK). The source of FCLK is selected based on the BURSTEN bit. When BURSTEN is logic 0, FCLK is derived from the current system clock. When BURSTEN is logic 1, FCLK is derived from the Burst Mode Oscillator, an independent clock source with a maximum frequency of 25 MHz.

When ADC0 is performing a conversion, it requires a clock source that is typically slower than FCLK. The ADC0 SAR conversion clock (SAR clock) is a divided version of FCLK. The divide ratio can be configured using the AD0SC bits in the ADC0CF register. The maximum SAR clock frequency is listed in Table 6.9.

ADC0 can be in one of three states at any given time: tracking, converting, or idle. Tracking time depends on the tracking mode selected. For Pre-Tracking Mode, tracking is managed by software and ADC0 starts conversions immediately following the convert start signal. For Post-Tracking and Dual-Tracking Modes, the tracking time after the convert start signal is equal to the value determined by the AD0TK bits plus 2 FCLK cycles. Tracking is immediately followed by a conversion. The ADC0 conversion time is always 13 SAR clock cycles plus an additional 2 FCLK cycles to start and complete a conversion. Figure 5.3 shows timing diagrams for a conversion in Pre-Tracking Mode and tracking plus conversion in Post-Tracking or Dual-Tracking Mode. In this example, repeat count is set to one.



Downloaded from Arrow.com.

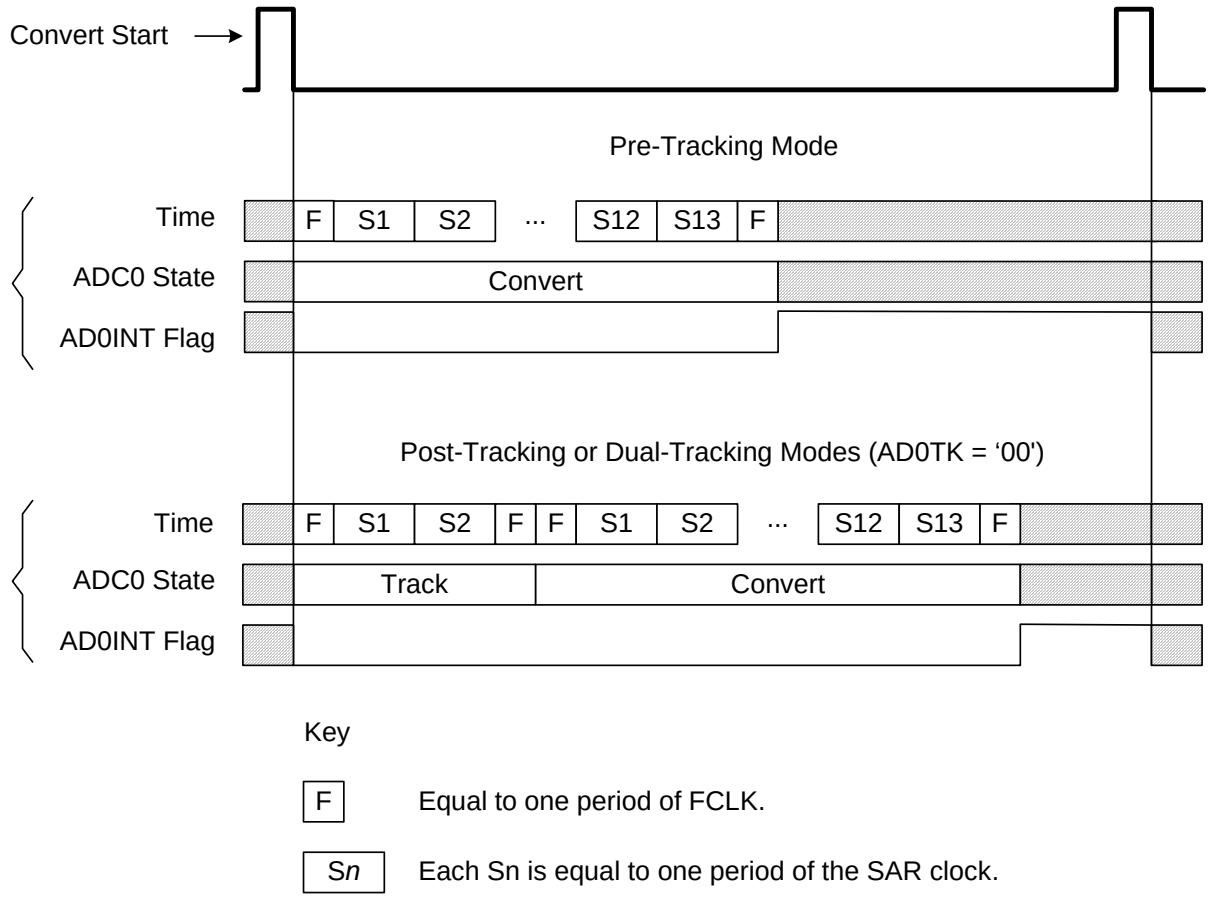



**Figure 5.3. 12-Bit ADC Tracking Mode Example**

### 5.1.4. Burst Mode

Burst Mode is a power saving feature that allows ADC0 to remain in a very low power state between conversions. When Burst Mode is enabled, ADC0 wakes from a very low power state, accumulates 1, 4, 8, or 16 samples using an internal Burst Mode clock (approximately 25 MHz), then re-enters a very low power state. Since the Burst Mode clock is independent of the system clock, ADC0 can perform multiple conversions then enter a very low power state within a single system clock cycle, even if the system clock is slow (e.g., 32.768 kHz), or suspended.

Burst Mode is enabled by setting BURSTEN to logic 1. When in Burst Mode, AD0EN controls the ADC0 idle power state (i.e. the state ADC0 enters when not tracking or performing conversions). If AD0EN is set to logic 0, ADC0 is powered down after each burst. If AD0EN is set to logic 1, ADC0 remains enabled after each burst. On each convert start signal, ADC0 is awakened from its Idle Power State. If ADC0 is powered down, it will automatically power up and wait the programmable Power-Up Time controlled by the AD0PWR bits. Otherwise, ADC0 will start tracking and converting immediately. Figure 5.4 shows an example of Burst Mode Operation with a slow system clock and a repeat count of 4.

**Important Note:** When Burst Mode is enabled, only Post-Tracking and Dual-Tracking modes can be used.

When Burst Mode is enabled, a single convert start will initiate a number of conversions equal to the repeat count. When Burst Mode is disabled, a convert start is required to initiate each conversion. In both modes, the ADC0 End of Conversion Interrupt Flag (AD0INT) will be set after “repeat count” conversions have



Downloaded from Arrow.com.

been accumulated. Similarly, the Window Comparator will not compare the result to the greater-than and less-than registers until “repeat count” conversions have been accumulated.

**Note:** When using Burst Mode, care must be taken to issue a convert start signal no faster than once every four SYSCLK periods. This includes external convert start signals.

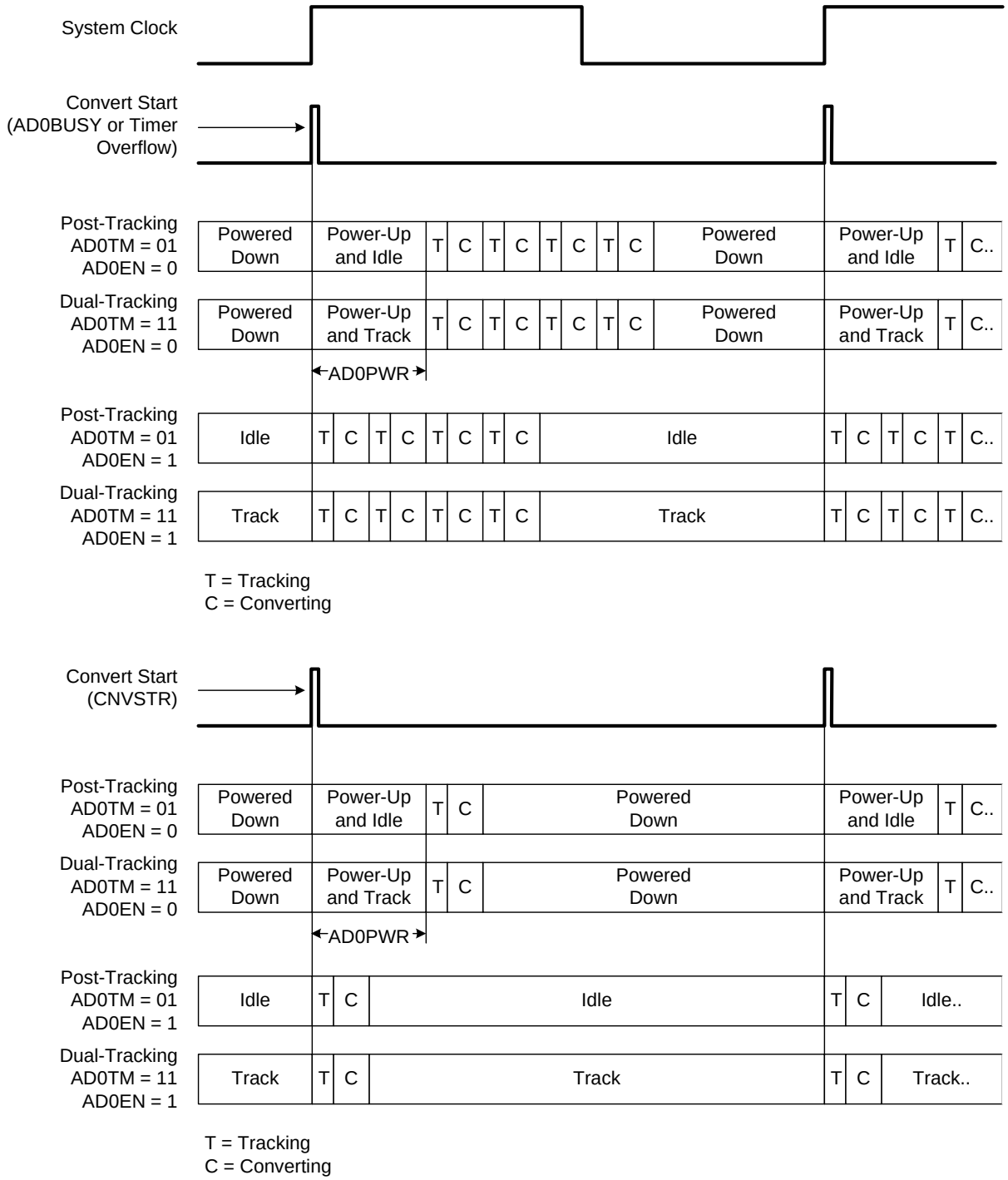



**Figure 5.4. 12-Bit ADC Burst Mode Example With Repeat Count Set to 4**



Downloaded from Arrow.com.

## 5.2. Output Code Formatting

The registers ADC0H and ADC0L contain the high and low bytes of the output conversion code. When the repeat count is set to 1, conversion codes are represented in 12-bit unsigned integer format and the output conversion code is updated after each conversion. Inputs are measured from 0 to $V_{REF}$ x 4095/4096. Data can be right-justified or left-justified, depending on the setting of the AD0LJST bit (ADC0CN.2). Unused bits in the ADC0H and ADC0L registers are set to 0. Example codes are shown below for both right-justified and left-justified data.

| Input Voltage | Right-Justified ADC0H:ADC0L (AD0LJST = 0) | Left-Justified ADC0H:ADC0L (AD0LJST = 1) |
|---|---|---|
| VREF x 4095/4096 | 0x0FFF | 0xFFF0 |
| VREF x 2048/4096 | 0x0800 | 0x8000 |
| VREF x 2047/4096 | 0x07FF | 0x7FF0 |
| 0 | 0x0000 | 0x0000 |

When the ADC0 Repeat Count is greater than 1, the output conversion code represents the accumulated result of the conversions performed and is updated after the last conversion in the series is finished. Sets of 4, 8, or 16 consecutive samples can be accumulated and represented in unsigned integer format. The repeat count can be selected using the AD0RPT bits in the ADC0CF register. The value must be right-justified (AD0LJST = 0), and unused bits in the ADC0H and ADC0L registers are set to 0. The following example shows right-justified codes for repeat counts greater than 1. Notice that accumulating $2^n$ samples is equivalent to left-shifting by *n* bit positions when all samples returned from the ADC have the same value.

| Input Voltage | Repeat Count = 4 | Repeat Count = 8 | Repeat Count = 16 |
|---|---|---|---|
| $V_{REF}$ x 4095/4096 | 0x3FFC | 0x7FF8 | 0xFFF0 |
| $V_{REF}$ x 2048/4096 | 0x2000 | 0x4000 | 0x8000 |
| $V_{REF}$ x 2047/4096 | 0x1FFC | 0x3FF8 | 0x7FF0 |
| 0 | 0x0000 | 0x0000 | 0x0000 |

### 5.2.1. Settling Time Requirements

A minimum tracking time is required before an accurate conversion is performed. This tracking time is determined by any series impedance, including the AMUX0 resistance, the ADC0 sampling capacitance, and the accuracy required for the conversion.

Figure 5.5 shows the equivalent ADC0 input circuit. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 5.1. When measuring the Temperature Sensor output, use the settling time specified in Table 6.10. When measuring $V_{DD}$ with respect to GND, $R_{TOTAL}$ reduces to $R_{MUX}$. See Table 6.9 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 5.1. 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).



Downloaded from Arrow.com.

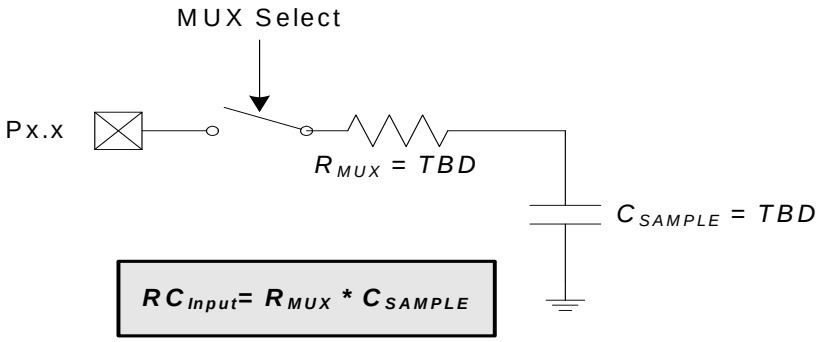



**Figure 5.5. ADC0 Equivalent Input Circuit**

## 5.3. Selectable Gain

ADC0 on the C8051F54x family of devices implements a selectable gain adjustment option. By writing a value to the gain adjust address range, the user can select gain values between 0 and 1.016.

For example, three analog sources to be measured have full-scale outputs of 5.0 V, 4.0 V, and 3.0 V, respectively. Each ADC measurement would ideally use the full dynamic range of the ADC with an internal voltage reference of 1.5 V or 2.2 V (set to 2.2 V for this example). When selecting the first source (5.0 V full-scale), a gain value of 0.44 (5 V full scale x 0.44 = 2.2 V full scale) provides a full-scale signal of 2.2 V when the input signal is 5.0 V. Likewise, a gain value of 0.55 (4 V full scale x 0.55 = 2.2 V full scale) for the second source and 0.73 (3 V full scale x 0.73 = 2.2 V full scale) for the third source provide full-scale ADC0 measurements when the input signal is full-scale.

Additionally, some sensors or other input sources have small part-to-part variations that must be accounted for to achieve accurate results. In this case, the programmable gain value could be used as a calibration value to eliminate these part-to-part variations.

### 5.3.1. Calculating the Gain Value

The ADC0 selectable gain feature is controlled by 13 bits in three registers. ADC0GNH contains the 8 upper bits of the gain value and ADC0GNL contains the 4 lower bits of the gain value. The final GAINADD bit (ADC0GNA.0) controls an optional extra 1/64 (0.016) of gain that can be added in addition to the ADC0GNH and ADC0GNL gain. The ADC0GNA.0 bit is set to 1 after a power-on reset.

The equivalent gain for the ADC0GNH, ADC0GNL and ADC0GNA registers is as follows:

$$gain = \left(\frac{GAIN}{4096}\right) + GAINADD \times \left(\frac{1}{64}\right)$$

**Equation 5.2. Equivalent Gain from the ADC0GNH and ADC0GNL Registers**

Where:
*GAIN* is the 12-bit word of ADC0GNH[7:0] and ADC0GNL[7:4]
*GAINADD* is the value of the GAINADD bit (ADC0GNA.0)

*gain* is the equivalent gain value from 0 to 1.016



Downloaded from Arrow.com.

For example, if ADC0GNH = 0xFC, ADC0GNL = 0x00, and GAINADD = 1, GAIN = 0xFC0 = 4032, and the resulting equation is as follows:

$$GAIN = \left(\frac{4032}{4096}\right) + 1 \times \left(\frac{1}{64}\right) = 0.984 + 0.016 = 1.0$$

The table below equates values in the ADC0GNH, ADC0GNL, and ADC0GNA registers to the equivalent gain using this equation.

| ADC0GNH Value | ADC0GNL Value | GAINADD Value | GAIN Value | Equivalent Gain |
|---|---|---|---|---|
| 0xFC (default) | 0x00 (default) | 1 (default) | 4032 + 64 | 1.0 (default) |
| 0x7C | 0x00 | 1 | 1984 + 64 | 0.5 |
| 0xBC | 0x00 | 1 | 3008 + 64 | 0.75 |
| 0x3C | 0x00 | 1 | 960 + 64 | 0.25 |
| 0xFF | 0xF0 | 0 | 4095 + 0 | ~1.0 |
| 0xFF | 0xF0 | 1 | 4096 + 64 | 1.016 |

For any desired gain value, the GAIN registers can be calculated by the following:

$$GAIN = \left(gain - GAINADD \times \left(\frac{1}{64}\right)\right) \times 4096$$

**Equation 5.3. Calculating the ADC0GNH and ADC0GNL Values from the Desired Gain**

Where:
*GAIN* is the 12-bit word of ADC0GNH[7:0] and ADC0GNL[7:4]
*GAINADD* is the value of the GAINADD bit (ADC0GNA.0)

*gain* is the equivalent gain value from 0 to 1.016

When calculating the value of GAIN to load into the ADC0GNH and ADC0GNL registers, the GAINADD bit can be turned on or off to reach a value closer to the desired gain value.

For example, the initial example in this section requires a gain of 0.44 to convert 5 V full scale to 2.2 V full scale. Using Equation 5.3:

$$GAIN = \left(0.44 - GAINADD \times \left(\frac{1}{64}\right)\right) \times 4096$$

If GAINADD is set to 1, this makes the equation:

$$GAIN = \left(0.44 - 1 \times \left(\frac{1}{64}\right)\right) \times 4096 = 0.424 \times 4096 = 1738 = 0x06CA$$

The actual gain from setting GAINADD to 1 and ADC0GNH and ADC0GNL to 0x6CA is 0.4399. A similar gain can be achieved if GAINADD is set to 0 with a different value for ADC0GNH and ADC0GNL.



Downloaded from Arrow.com.

### 5.3.2. Setting the Gain Value

The three programmable gain registers are accessed indirectly using the ADC0H and ADC0L registers when the GAINEN bit (ADC0CF.0) bit is set. ADC0H acts as the address register, and ADC0L is the data register. The programmable gain registers can only be written to and cannot be read. See Gain Register Definition 5.1, Gain Register Definition 5.2, and Gain Register Definition 5.3 for more information.

The gain is programmed using the following steps:

1. Set the GAINEN bit (ADC0CF.0)
2. Load the ADC0H with the ADC0GNH, ADC0GNL, or ADC0GNA address.
3. Load ADC0L with the desired value for the selected gain register.
4. Reset the GAINEN bit (ADC0CF.0)

**Notes:**

1. An ADC conversion should not be performed while the GAINEN bit is set.
2. Even with gain enabled, the maximum input voltage must be less than $V_{REGIN}$ and the maximum voltage of the signal after gain must be less than or equal to $V_{REF}$.

In code, changing the value to 0.44 gain from the previous example looks like:

```
// in 'C':
ADC0CF |= 0x01;          // GAINEN = 1
ADC0H = 0x04;            // Load the ADC0GNH address
ADC0L = 0x6C;            // Load the upper byte of 0x6CA to ADC0GNH
ADC0H = 0x07;            // Load the ADC0GNL address
ADC0L = 0xA0;            // Load the lower nibble of 0x6CA to ADC0GNL
ADC0H = 0x08;            // Load the ADC0GNA address
ADC0L = 0x01;            // Set the GAINADD bit
ADC0CF &= ~0x01;         // GAINEN = 0

; in assembly
ORL ADC0CF,#01H          ; GAINEN = 1
MOV ADC0H,#04H           ; Load the ADC0GNH address
MOV ADC0L,#06CH          ; Load the upper byte of 0x6CA to ADC0GNH
MOV ADC0H,#07H           ; Load the ADC0GNL address
MOV ADC0L,#0A0H          ; Load the lower nibble of 0x6CA to ADC0GNL
MOV ADC0H,#08H           ; Load the ADC0GNA address
MOV ADC0L,#01H           ; Set the GAINADD bit
ANL ADC0CF,#0FEH         ; GAINEN = 0
```



Downloaded from Arrow.com.

## Gain Register Definition 5.1. ADC0GNH: ADC0 Selectable Gain High Byte

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

Indirect Address = 0x04;

| Bit | Name | Function |
|---|---|---|
| 7:0 | GAINH[7:0] | **ADC0 Gain High Byte.**<br>See Section 5.3.1 for details on calculating the value for this register. |
| **Note:** This register is accessed indirectly; See Section 5.3.2 for details for writing this register. | | |

## Gain Register Definition 5.2. ADC0GNL: ADC0 Selectable Gain Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | GAINL[3:0] | | | | Reserved | Reserved | Reserved | Reserved |
| Type | W | | | | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Indirect Address = 0x07;

| Bit | Name | Function |
|---|---|---|
| 7:4 | GAINL[3:0] | **ADC0 Gain Lower 4 Bits.**<br>See Figure 5.3.1 for details for setting this register.<br><br>This register is only accessed indirectly through the ADC0H and ADC0L register. |
| 3:0 | Reserved | Must Write 0000b |
| **Note:** This register is accessed indirectly; See Section 5.3.2 for details for writing this register. | | |



Downloaded from Arrow.com.

## Gain Register Definition 5.3. ADC0GNA: ADC0 Additional Selectable Gain

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

Indirect Address = 0x08;

| Bit | Name | Function |
|---|---|---|
| 7:1 | Reserved | Must Write 0000000b. |
| 0 | GAINADD | **ADC0 Additional Gain Bit.**<br>Setting this bit add 1/64 (0.016) gain to the gain value in the ADC0GNH and ADC0GNL registers. |
| **Note:** This register is accessed indirectly; See Section 5.3.2 for details for writing this register. | | |



Downloaded from Arrow.com.

## SFR Definition 5.4. ADC0CF: ADC0 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AD0SC[4:0] | | | | | AD0RPT[1:0] | | GAINEN |
| Type | R/W | | | | | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

SFR Address = 0xBC; SFR Page = 0x00

| 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>BURSTEN = 0: FCLK is the current system clock<br>BURSTEN = 1: FCLK is a maximum of 30 MHz, independent of the current system clock..<br>$AD0SC = \frac{FCLK}{CLK_{SAR}} - 1$<br>**Note:** Round up the result of the calculation for AD0SC |
| 2:1 | A0RPT[1:0] | **ADC0 Repeat Count**<br>Controls the number of conversions taken and accumulated between ADC0 End of Conversion (ADCINT) and ADC0 Window Comparator (ADCWINT) interrupts. A convert start is required for each conversion unless Burst Mode is enabled. In Burst Mode, a single convert start can initiate multiple self-timed conversions. Results in both modes are accumulated in the ADC0H:ADC0L register. **When AD0RPT1–0 are set to a value other than '00', the AD0LJST bit in the ADC0CN register must be set to '0' (right justified).**<br>00: 1 conversion is performed.<br>01: 4 conversions are performed and accumulated.<br>10: 8 conversions are performed and accumulated.<br>11: 16 conversions are performed and accumulated. |
| 0 | GAINEN | **Gain Enable Bit.**<br>Controls the gain programming. Refer to Section “5.3. Selectable Gain” on page 35 for information about using this bit. |



Downloaded from Arrow.com.

## SFR Definition 5.5. 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; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0H[7:0] | **ADC0 Data Word High-Order Bits.**<br>For AD0LJST = 0 and AD0RPT as follows:<br>00: Bits 3–0 are the upper 4 bits of the 12-bit result. Bits 7–4 are 0000b.<br>01: Bits 4–0 are the upper 5 bits of the 14-bit result. Bits 7–5 are 000b.<br>10: Bits 5–0 are the upper 6 bits of the 15-bit result. Bits 7–6 are 00b.<br>11: Bits 7–0 are the upper 8 bits of the 16-bit result.<br>For AD0LJST = 1 (AD0RPT must be 00): Bits 7–0 are the most-significant bits of the ADC0 12-bit result. |

## SFR Definition 5.6. 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; SFR Page = 0x00

| 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 ADC0 Accumulated Result.<br>For AD0LJST = 1 (AD0RPT must be '00'): Bits 7–4 are the lower 4 bits of the 12-bit result. Bits 3–0 are 0000b. |



Downloaded from Arrow.com.

## SFR Definition 5.7. ADC0CN: ADC0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AD0EN | BURSTEN | AD0INT | AD0BUSY | AD0WINT | AD0LJST | AD0CM[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 = 0xE8; SFR Page = 0x00; 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 | BURSTEN | **ADC0 Burst Mode Enable Bit.**<br>0: Burst Mode Disabled.<br>1: Burst Mode Enabled. | | |
| 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[1:0] = 00b |
| 3 | AD0WINT | **ADC0 Window Compare Interrupt Flag.**<br>This bit must be cleared by software<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 | AD0LJST | **ADC0 Left Justify Select Bit.**<br>0: Data in ADC0H:ADC0L registers is right-justified<br>1: Data in ADC0H:ADC0L registers is left-justified. This option should not be used with a repeat count greater than 1 (when AD0RPT[1:0] is 01b, 10b, or 11b). | | |
| 1:0 | AD0CM[1:0] | **ADC0 Start of Conversion Mode Select.**<br>00: ADC0 start-of-conversion source is write of 1 to AD0BUSY.<br>01: ADC0 start-of-conversion source is overflow of Timer 1.<br>10: ADC0 start-of-conversion source is rising edge of external CNVSTR.<br>11: ADC0 start-of-conversion source is overflow of Timer 2. | | |



Downloaded from Arrow.com.

## SFR Definition 5.8. ADC0TK: ADC0 Tracking Mode Select

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AD0PWR[3:0] | | | | AD0TM[1:0] | | AD0TK[1:0] | |
| Type | R/W | | | | R/W | | R/W | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xBA; SFR Page = 0x00;

| Bit | Name | Function |
|---|---|---|
| 7:4 | AD0PWR[3:0] | **ADC0 Burst Power-Up Time.**<br>For BURSTEN = 0: ADC0 Power state controlled by AD0EN<br>For BURSTEN = 1, AD0EN = 1: ADC0 remains enabled and does not enter the very low power state<br>For BURSTEN = 1, AD0EN = 0: ADC0 enters the very low power state and is enabled after each convert start signal. The Power-Up time is programmed according the following equation:<br>$AD0PWR = \frac{Tstartup}{200ns} - 1$ or $Tstartup = (AD0PWR + 1)200ns$ |
| 3:2 | AD0TM[1:0] | **ADC0 Tracking Mode Enable Select Bits.**<br>00: Reserved.<br>01: ADC0 is configured to Post-Tracking Mode.<br>10: ADC0 is configured to Pre-Tracking Mode.<br>11: ADC0 is configured to Dual Tracking Mode. |
| 1:0 | AD0TK[1:0] | **ADC0 Post-Track Time.**<br>00: Post-Tracking time is equal to 2 SAR clock cycles + 2 FCLK cycles.<br>01: Post-Tracking time is equal to 4 SAR clock cycles + 2 FCLK cycles.<br>10: Post-Tracking time is equal to 8 SAR clock cycles + 2 FCLK cycles.<br>11: Post-Tracking time is equal to 16 SAR clock cycles + 2 FCLK cycles. |

## 5.4. 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.



Downloaded from Arrow.com.

## SFR Definition 5.9. 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; SFR Page = 0x00

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

## SFR Definition 5.10. 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; SFR Page = 0x00

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



Downloaded from Arrow.com.

## SFR Definition 5.11. 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; SFR Page = 0x00

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

## SFR Definition 5.12. 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; SFR Page = 0x00

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

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

Figure 5.6 shows two example window comparisons for right-justified data with ADC0LTH:ADC0LTL = 0x0200 (512d) and ADC0GTH:ADC0GTL = 0x0100 (256d). The input voltage can range from 0 to $V_{REF}$ x (4095/4096) with respect to GND, and is represented by a 12-bit unsigned integer value. The repeat count is set to one. 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 0x0100 < ADC0H:ADC0L < 0x0200). 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 < 0x0100 or ADC0H:ADC0L > 0x0200). Figure 5.7 shows an example using left-justified data with the same comparison values.



Downloaded from Arrow.com.

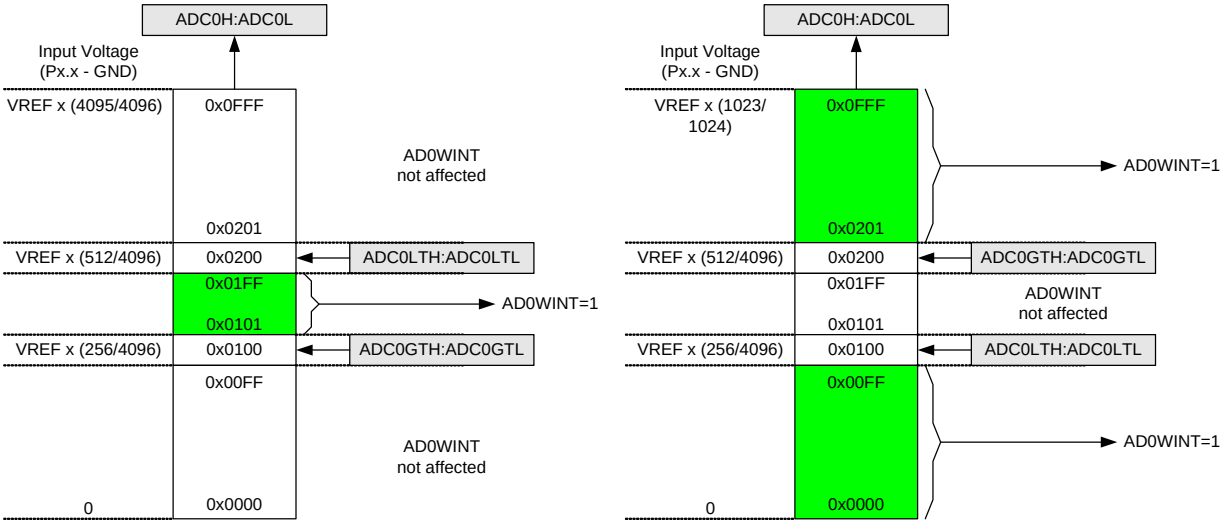



**Figure 5.6. ADC Window Compare Example: Right-Justified Data**

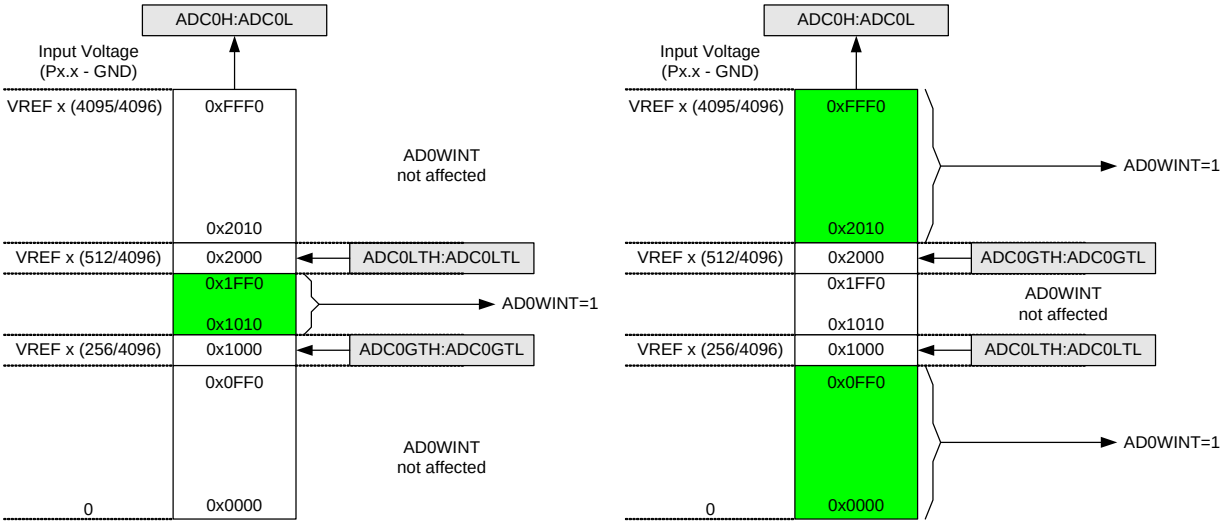



**Figure 5.7. ADC Window Compare Example: Left-Justified Data**



Downloaded from Arrow.com.

# 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 | — | 135 | °C |
| Storage Temperature | | –65 | — | 150 | °C |
| Voltage on $V_{REGIN}$ with Respect to GND | | –0.3 | — | 5.5 | V |
| Voltage on $V_{DD}$ with Respect to GND | | –0.3 | — | 2.8 | V |
| Voltage on VDDA with Respect to GND | | –0.3 | — | 2.8 | V |
| Voltage on $V_{IO}$ with Respect to GND | | –0.3 | — | 5.5 | V |
| Voltage on any Port I/O Pin or $\overline{RST}$ with Respect to GND | | –0.3 | — | $V_{IO}$ + 0.3 | V |
| Maximum Total Current through $V_{REGIN}$ or GND | | — | — | 500 | mA |
| Maximum Output Current Sunk by $\overline{RST}$ or any Port Pin | | — | — | 100 | mA |
| Maximum Output Current Sourced by any Port Pin | | — | — | 100 | mA |

**Note:** Stresses outside of the range of the "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 outside of those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.



Downloaded from Arrow.com.

## 6.2. Electrical Characteristics

### Table 6.2. Global Electrical Characteristics

–40 to +125 °C, 24 MHz system clock unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Supply Input Voltage ($V_{REGIN}$) | | 1.8 | — | 5.25 | V |
| Digital Supply Voltage ($V_{DD}$) | System Clock ≤ 25 MHz | $V_{RST}$[1] | — | 2.75 | V |
| | System Clock > 25 MHz | 2 | — | 2.75 | |
| Analog Supply Voltage (VDDA) | System Clock ≤ 25 MHz | $V_{RST}$[1] | — | 2.75 | V |
| (Must be connected to $V_{DD}$) | System Clock > 25 MHz | 2 | | 2.75 | |
| Port I/O Supply Voltage ($V_{IO}$) | Normal Operation | 1.8[2] | — | 5.25 | V |
| Digital Supply RAM Data Retention Voltage | | — | 1.5 | — | V |
| SYSCLK (System Clock)[3] | | 0 | — | 50 | MHz |
| $T_{SYSH}$ (SYSCLK High Time) | | 9 | — | — | ns |
| $T_{SYSL}$ (SYSCLK Low Time) | | 9 | — | — | ns |
| Specified Operating Temperature Range | | –40 | — | +125 | °C |
| **Digital Supply Current—CPU Active (Normal Mode, fetching instructions from Flash)** | | | | | |
| $I_{DD}$[4] | $V_{DD}$ = 2.1 V, F = 200 kHz | — | 85 | — | µA |
| | $V_{DD}$ = 2.1 V, F = 1.5 MHz | — | 600 | — | µA |
| | $V_{DD}$ = 2.1 V, F = 25 MHz | — | 9.2 | 11 | mA |
| | $V_{DD}$ = 2.1 V, F = 50 MHz | — | 17 | 21 | mA |
| $I_{DD}$[4] | $V_{DD}$ = 2.6 V, F = 200 kHz | — | 120 | — | µA |
| | $V_{DD}$ = 2.6 V, F = 1.5 MHz | — | 920 | — | µA |
| | $V_{DD}$ = 2.6 V, F = 25 MHz | — | 13 | 21 | mA |
| | $V_{DD}$ = 2.6 V, F = 50 MHz | — | 22 | 33 | mA |
| $I_{DD}$ Supply Sensitivity[4] | F = 25 MHz | — | 68 | — | %/V |
| | F = 1 MHz | — | 77 | — | %/V |
| $I_{DD}$ Frequency Sensitivity [4,5] | $V_{DD}$ = 2.1 V, F ≤ 12.5 MHz, T = 25 °C | — | 0.43 | — | mA/MHz |
| | $V_{DD}$ = 2.1 V, F > 12.5 MHz, T = 25 °C | — | 0.33 | — | mA/MHz |
| | $V_{DD}$ = 2.6 V, F ≤ 12.5 MHz, T = 25 °C | — | 0.60 | — | mA/MHz |
| | $V_{DD}$ = 2.6 V, F > 12.5 MHz, T = 25 °C | — | 0.42 | — | mA/MHz |

**Notes:**

1. Given in Table 6.4 on page 52.
2. $V_{IO}$ should not be lower than the $V_{DD}$ voltage.
3. SYSCLK must be at least 32 kHz to enable debugging.
4. Guaranteed by characterization. Does not include oscillator supply current.
5. IDD estimation for different frequencies.
6. Idle IDD estimation for different frequencies.



Downloaded from Arrow.com.

**Table 6.2. Global Electrical Characteristics (Continued)**
–40 to +125 °C, 24 MHz system clock unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| **Digital Supply Current—CPU Inactive (Idle Mode, not fetching instructions from Flash)** | | | | | |
| $I_{DD}$[4] | $V_{DD}$ = 2.1 V, F = 200 kHz | — | 50 | — | µA |
| | $V_{DD}$ = 2.1 V, F = 1.5 MHz | — | 410 | — | µA |
| | $V_{DD}$ = 2.1 V, F = 25 MHz | — | 6.5 | 8.0 | mA |
| | $V_{DD}$ = 2.1 V, F = 50 MHz | — | 13 | 16 | mA |
| $I_{DD}$[4] | $V_{DD}$ = 2.6 V, F = 200 kHz | — | 67 | — | µA |
| | $V_{DD}$ = 2.6 V, F = 1.5 MHz | — | 530 | — | µA |
| | $V_{DD}$ = 2.6 V, F = 25 MHz | — | 8.0 | 15 | mA |
| | $V_{DD}$ = 2.6 V, F = 50 MHz | — | 16 | 25 | mA |
| $I_{DD}$ Supply Sensitivity[4] | F = 25 MHz | — | 55 | — | %/V |
| | F = 1 MHz | — | 58 | — | |
| $I_{DD}$ Frequency Sensitivity [4.6] | $V_{DD}$ = 2.1V, F ≤ 12.5 MHz, T = 25 °C | — | 0.26 | — | mA/MHz |
| | $V_{DD}$ = 2.1V, F > 12.5 MHz, T = 25 °C | — | 0.26 | — | |
| | $V_{DD}$ = 2.6V, F ≤ 12.5 MHz, T = 25 °C | — | 0.34 | — | |
| | $V_{DD}$ = 2.6V, F > 12.5 MHz, T = 25 °C | — | 0.34 | — | |
| Digital Supply Current[4] (Stop or Suspend Mode) | Oscillator not running, $V_{DD}$ Monitor Disabled | | | | µA |
| | Temp = 25 °C | — | 1 | — | |
| | Temp = 60 °C | — | 6 | — | |
| | Temp= 125 °C | — | 70 | — | |

**Notes:**
1. Given in Table 6.4 on page 52.
2. $V_{IO}$ should not be lower than the $V_{DD}$ voltage.
3. SYSCLK must be at least 32 kHz to enable debugging.
4. Guaranteed by characterization. Does not include oscillator supply current.
5. IDD estimation for different frequencies.
6. Idle IDD estimation for different frequencies.



Downloaded from Arrow.com.

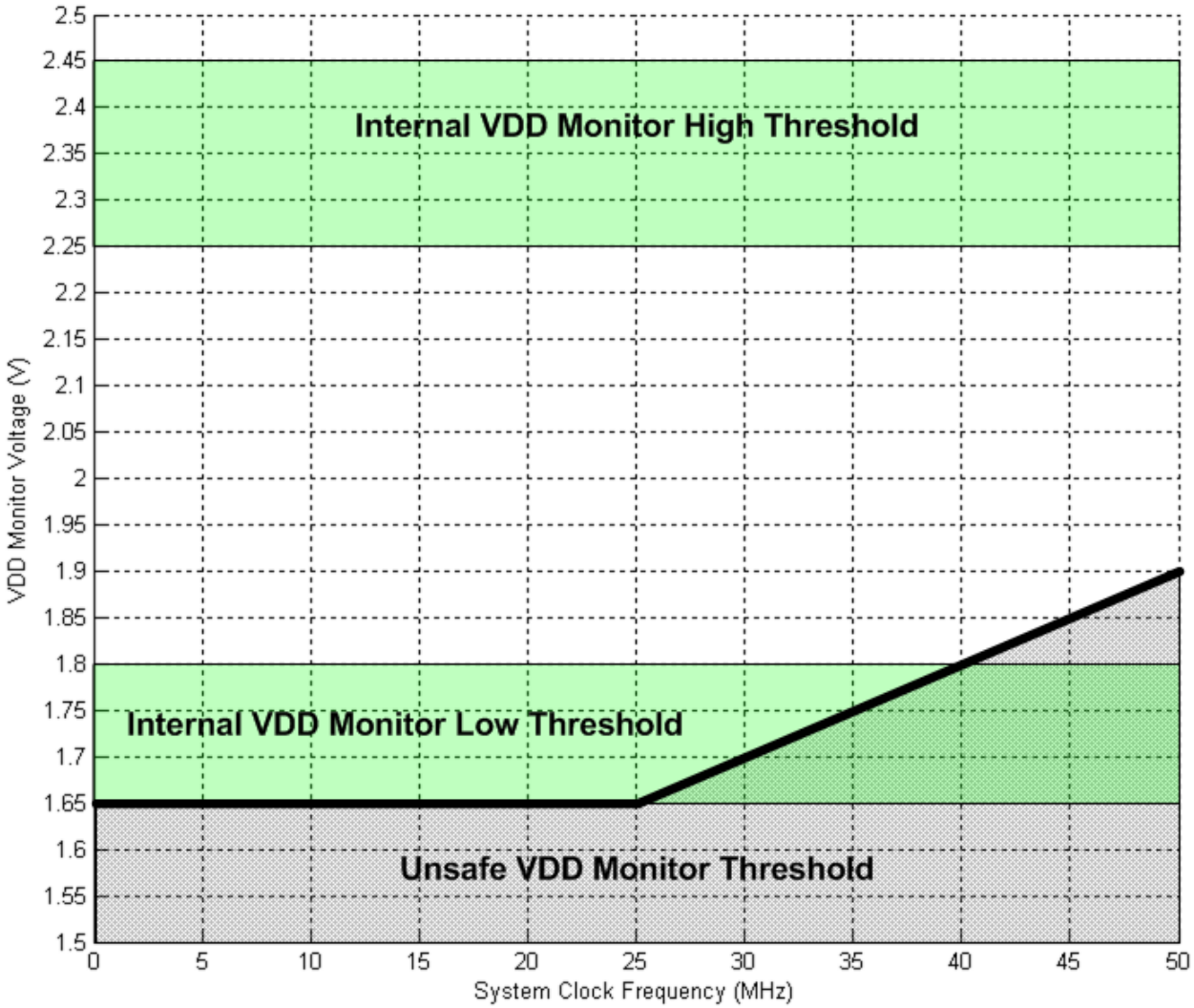



**Figure 6.1. Minimum VDD Monitor Threshold vs. System Clock Frequency**

**Note:** With system clock frequencies greater than 25 MHz, the $V_{DD}$ monitor level should be set to the high threshold (VDMLVL = 1b in SFR VDM0CN) to prevent undefined CPU operation. The high threshold should only be used with an external regulator powering $V_{DD}$ directly. See Figure 9.2 on page 73 for the recommended power supply connections.



Downloaded from Arrow.com.

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

$V_{DD}$ = 1.8 to 2.75 V, –40 to +125 °C unless otherwise specified.

| Parameters | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Output High Voltage | $I_{OH}$ = –3 mA, Port I/O push-pull | $V_{IO}$ – 0.4 | — | — | V |
| | $I_{OH}$ = –10 µA, Port I/O push-pull | $V_{IO}$ – 0.02 | — | — | |
| | $I_{OH}$ = –10 mA, Port I/O push-pull | — | $V_{IO}$ – 0.7 | — | |
| Output Low Voltage | **$V_{IO}$ = 1.8 V:** | | | | |
| | $I_{OL}$ = 70 µA | — | — | 50 | |
| | $I_{OL}$ = 8.5 mA | — | — | 750 | |
| | **$V_{IO}$ = 2.7 V:** | | | | |
| | $I_{OL}$ = 70 µA | — | — | 45 | mV |
| | $I_{OL}$ = 8.5 mA | — | — | 550 | |
| | **$V_{IO}$ = 5.25 V:** | | | | |
| | $I_{OL}$ = 70 µA | — | — | 40 | |
| | $I_{OL}$ = 8.5 mA | — | — | 400 | |
| Input High Voltage | $V_{REGIN}$ = 5.25 V | 0.7 x VIO | — | | V |
| Input Low Voltage | $V_{REGIN}$ = 2.7 V | — | — | 0.3 x VIO | V |
| Input Leakage Current | Weak Pullup Off | — | — | ±2 | µA |
| | Weak Pullup On, $V_{IO}$ = 2.1 V, $V_{IN}$ = 0 V, $V_{DD}$ = 1.8 V | — | 7 | 9 | |
| | Weak Pullup On, $V_{IO}$ = 2.6 V, $V_{IN}$ = 0 V, $V_{DD}$ = 2.6 V | — | 17 | 22 | |
| | Weak Pullup On, $V_{IO}$ = 5.0 V, $V_{IN}$ = 0 V, $V_{DD}$ = 2.6 V | — | 49 | 115 | |



Downloaded from Arrow.com.

### Table 6.4. Reset Electrical Characteristics

–40 to +125 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| $\overline{\text{RST}}$ Output Low Voltage | VIO = 5 V; IOL = 70 µA | — | — | 40 | mV |
| $\overline{\text{RST}}$ Input High Voltage | | 0.7 x $V_{IO}$ | — | — | |
| $\overline{\text{RST}}$ Input Low Voltage | | — | — | 0.3 x $V_{IO}$ | |
| $\overline{\text{RST}}$ Input Pullup Current | $\overline{\text{RST}}$ = 0.0 V, VIO = 5 V | — | 49 | 115 | µA |
| $V_{DD}$ RST Threshold ($V_{RST-LOW}$) | | 1.65 | 1.75 | 1.80 | V |
| $V_{DD}$ RST Threshold ($V_{RST-HIGH}$) | | 2.25 | 2.30 | 2.45 | V |
| Missing Clock Detector Timeout | Time from last system clock rising edge to reset initiation<br>$V_{DD}$ = 2.1 V<br>$V_{DD}$ = 2.5 V | <br><br>200<br>200 | <br><br>340<br>250 | <br><br>600<br>600 | µs |
| Reset Time Delay | Delay between release of any reset source and code execution at location 0x0000 | — | 155 | 175 | µs |
| Minimum $\overline{\text{RST}}$ Low Time to Generate a System Reset | | 6 | — | — | µs |
| $V_{DD}$ Monitor Turn-on Time | | — | 60 | 100 | µs |
| $V_{DD}$ Monitor Supply Current | | — | 1 | 2 | µA |

### Table 6.5. Flash Electrical Characteristics

$V_{DD}$ = 1.8 to 2.75 V, –40 to +125 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Flash Size | C8051F540/1/2/3 | 16384[1] | | | Bytes |
| | C8051F544/5/6/7 | 8192 | | | |
| Endurance | | 20 k | 150 k | — | Erase/Write |
| Retention | 125 °C | 10 | — | — | Years |
| Erase Cycle Time | 25 MHz System Clock | 28 | 30 | 45 | ms |
| Write Cycle Time | 25 MHz System Clock | 79 | 84 | 125 | µs |
| $V_{DD}$ | Write/Erase Operations | $V_{RST-HIGH}$[2] | — | — | V |

1. On the 16 kB Flash devices, 1024 bytes at addresses 0x3C00 to 0x3FFF are reserved.
2. See Table 6.4 for the $V_{RST-HIGH}$ specification.



Downloaded from Arrow.com.

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

$V_{DD}$ = 1.8 to 2.75 V, –40 to +125 °C unless otherwise specified; Using factory-calibrated settings.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Oscillator Frequency | IFCN = 111b;<br>VDD ≥ VREGMIN[1]<br><br>IFCN = 111b;<br>VDD < VREGMIN[1] | 24 – 0.5%<br><br>24 – 1.0% | 24[2]<br><br>24[2] | 24 + 0.5%<br><br>24 + 1.0% | MHz |
| Oscillator Supply Current (from $V_{DD}$) | Internal Oscillator On<br>OSCICN[7:6] = 11b | — | 880 | 1300 | µA |
| Internal Oscillator Suspend<br>OSCICN[7:6] = 00b<br>ZTCEN = 1 | Temp = 25 °C<br>Temp = 85 °C<br>Temp = 125 °C | — | 67<br>90<br>130 | — | |
| Wake-up Time From Suspend | OSCICN[7:6] = 00b | — | 1 | — | µs |
| Power Supply Sensitivity | Constant Temperature | — | 0.11 | — | %/V |
| Temperature Sensitivity[3] | Constant Supply<br>$TC_1$<br>$TC_2$ | <br>—<br>— | <br>5.0<br>–0.65 | <br>—<br>— | <br>ppm/°C<br>ppm/°C$^2$ |

1. VREGMIN is the minimum output of the voltage regulator for its low setting (REG0CN: REG0MD = 0b). See Table 6.8, “Voltage Regulator Electrical Characteristics,” on page 54.
2. This is the average frequency across the operating temperature range
3. Use temperature coefficients $TC_1$ and $TC_2$ to calculate the new internal oscillator frequency using the following equation:

$$f(T) = f0 \times (1 + TC_1 \times (T - T0) + TC_2 \times (T - T0)^2)$$

where f0 is the internal oscillator frequency at 25 °C and T0 is 25 °C.



Downloaded from Arrow.com.

### Table 6.7. Clock Multiplier Electrical Specifications

$V_{DD}$ = 1.8 to 2.75 V, –40 to +125 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Input Frequency ($Fcm_{in}$) | | 2 | — | — | MHz |
| Output Frequency | | — | — | 50 | MHz |
| Power Supply Current | | — | 0.9 | 1.9 | mA |

### Table 6.8. Voltage Regulator Electrical Characteristics

$V_{DD}$ = 1.8 to 2.75 V, –40 to +125 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Input Voltage Range ($V_{REGIN}$)* | | 1.8* | — | 5.25 | V |
| Dropout Voltage ($V_{DO}$) | Maximum Current = 50 mA | — | 10 | — | mV/mA |
| Output Voltage ($V_{DD}$) | 2.1 V operation (REG0MD = 0)<br>2.6 V operation (REG0MD = 1) | 2.0<br>2.5 | 2.1<br>2.6 | 2.25<br>2.75 | V |
| Bias Current | | — | 1 | 9 | µA |
| Dropout Indicator Detection Threshold | With respect to VDD | –0.21 | — | –0.02 | V |
| Output Voltage Temperature Coefficient | | — | 0.29 | — | mV/°C |
| VREG Settling Time | 50 mA load with $V_{REGIN}$ = 2.4 V and $V_{DD}$ load capacitor of 4.8 µF | — | 450 | — | µs |
| ***Note:** The minimum input voltage is 1.8 V or $V_{DD}$ + $V_{DO}$(max load), whichever is greater | | | | | |



Downloaded from Arrow.com.

### Table 6.9. ADC0 Electrical Characteristics

VDDA = 1.8 to 2.75 V, –40 to +125 °C, VREF = 1.5 V (REFSL=0) unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| **DC Accuracy** | | | | | |
| Resolution | | | 12 | | bits |
| Integral Nonlinearity | | — | ±0.5 | ±3 | LSB |
| Differential Nonlinearity | Guaranteed Monotonic | — | ±0.5 | ±1 | LSB |
| Offset Error[1] | | –10 | 3.0 | 10 | LSB |
| Full Scale Error | | –20 | 5.7 | 20 | LSB |
| Offset Temperature Coefficient | | — | 7.7 | — | ppm/°C |
| **Dynamic performance (10 kHz sine-wave single-ended input, 1 dB below Full Scale, 200 ksps)** | | | | | |
| Signal-to-Noise Plus Distortion | | 63 | 65 | — | dB |
| Total Harmonic Distortion | Up to the 5th harmonic; | — | 80 | — | dB |
| Spurious-Free Dynamic Range | | — | -82 | — | dB |
| **Conversion Rate** | | | | | |
| SAR Conversion Clock | | — | — | 3.6 | MHz |
| Conversion Time in SAR Clocks[2] | | 13 | — | — | clocks |
| Track/Hold Acquisition Time[3] | VDDA ≥ 2.0 V<br>VDDA < 2.0 V | 1.5<br>3.5 | —<br>— | —<br>— | µs |
| Throughput Rate[4] | VDDA ≥ 2.0 V | — | — | 200 | ksps |
| **Analog Inputs** | | | | | |
| ADC Input Voltage Range[5] | gain = 1.0 (default)<br>gain = n | 0<br>0 | — | VREF<br>VREF / n | V |
| Absolute Pin Voltage with respect to GND | | 0 | — | $V_{IO}$ | V |
| Sampling Capacitance | | — | 31 | — | pF |
| Input Multiplexer Impedance | | — | 3 | — | kΩ |
| **Power Specifications** | | | | | |
| Power Supply Current (VDDA supplied to ADC0) | Operating Mode, 200 ksps | — | 1100 | 1500 | µA |
| Burst Mode (Idle) | | — | 1100 | 1500 | µA |
| Power-On Time | | 5 | — | — | µs |
| Power Supply Rejection | | — | –60 | — | mV/V |

**Notes:**

1. Represents one standard deviation from the mean. Offset and full-scale error can be removed through calibration.
2. An additional 2 FCLK cycles are required to start and complete a conversion
3. Additional tracking time may be required depending on the output impedance connected to the ADC input. See Section "5.2.1. Settling Time Requirements" on page 34.
4. An increase in tracking time will decrease the ADC throughput.
5. See Section "5.3. Selectable Gain" on page 35 for more information about the setting the gain.



Downloaded from Arrow.com.

### Table 6.10. Temperature Sensor Electrical Characteristics

VDDA = 1.8 to 2.75 V, –40 to +125 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Linearity | | — | ±0.1 | — | °C |
| Slope | | — | 3.33 | — | mV/°C |
| Slope Error* | | — | 88 | — | µV/°C |
| Offset | Temp = 0 °C | — | 856 | — | mV |
| Offset Error* | Temp = 0 °C | — | ±14 | — | mV |
| Power Supply Current | | — | 18 | — | µA |
| Tracking Time | | 12 | — | — | µs |
| ***Note:** Represents one standard deviation from the mean. | | | | | |

### Table 6.11. Voltage Reference Electrical Characteristics

VDDA = 1.8 to 2.75 V, –40 to +125 °C unless otherwise specified.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| **Internal Reference (REFBE = 1)** | | | | | |
| Output Voltage | 25 °C ambient (REFLV = 0) | 1.45 | 1.50 | 1.55 | V |
| | 25 °C ambient (REFLV = 1), $V_{DD}$ = 2.6 V | 2.15 | 2.20 | 2.25 | |
| VREF Short-Circuit Current | | — | 5 | 10 | mA |
| VREF Temperature Coefficient | | — | 38 | — | ppm/°C |
| Power Consumption | Internal | — | 30 | 50 | µA |
| Load Regulation | Load = 0 to 200 µA to AGND | — | 3 | — | µV/µA |
| VREF Turn-on Time 1 | 4.7 µF tantalum and 0.1 µF bypass | — | 1.5 | — | ms |
| VREF Turn-on Time 2 | 0.1 µF bypass | — | 46 | — | µs |
| Power Supply Rejection | | — | 1.2 | — | mV/V |
| **External Reference (REFBE = 0)** | | | | | |
| Input Voltage Range | | 1.5 | — | $V_{DDA}$ | V |
| Input Current | Sample Rate = 200 ksps; VREF = 1.5 V | — | 2.1 | — | µA |
| **Power Specifications** | | | | | |
| Reference Bias Generator | REFBE = 1 or TEMPE = 1 | — | 21 | 40 | µA |



Downloaded from Arrow.com.

**Table 6.12. Comparator 0 and Comparator 1 Electrical Characteristics**

VIO = 1.8 to 5.25 V, –40 to +125 °C unless otherwise noted.

| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Response Time: Mode 0, Vcm* = 1.5 V | CPn+ – CPn– = 100 mV | — | 330 | — | ns |
| | CPn+ – CPn– = –100 mV | — | 390 | — | ns |
| Response Time: Mode 1, Vcm* = 1.5 V | CPn+ – CPn– = 100 mV | — | 490 | — | ns |
| | CPn+ – CPn– = –100 mV | — | 610 | — | ns |
| Response Time: Mode 2, Vcm* = 1.5 V | CPn+ – CPn– = 100 mV | — | 590 | — | ns |
| | CP0+ – CP0– = –100 mV | — | 750 | — | ns |
| Response Time: Mode 3, Vcm* = 1.5 V | CPn+ – CPn– = 100 mV | — | 2300 | — | ns |
| | CPn+ – CPn– = –100 mV | — | 3100 | — | ns |
| Common-Mode Rejection Ratio | | — | 2.1 | 13 | mV/V |
| Positive Hysteresis 1 | CPnHYP1–0 = 00 | –2 | 0 | 2 | mV |
| Positive Hysteresis 2 | CPnHYP1–0 = 01 | 2 | 6 | 10 | mV |
| Positive Hysteresis 3 | CPnHYP1–0 = 10 | 5 | 11 | 20 | mV |
| Positive Hysteresis 4 | CPnHYP1–0 = 11 | 13 | 21 | 40 | mV |
| Negative Hysteresis 1 | CPnHYN1–0 = 00 | –2 | 0 | 2 | mV |
| Negative Hysteresis 2 | CPnHYN1–0 = 01 | 2 | 5 | 10 | mV |
| Negative Hysteresis 3 | CPnHYN1–0 = 10 | 5 | 11 | 20 | mV |
| Negative Hysteresis 4 | CPnHYN1–0 = 11 | 13 | 21 | 40 | mV |
| Inverting or Non-Inverting Input Voltage Range | | –0.25 | — | $V_{IO}$ + 0.25 | V |
| Input Capacitance | | — | 8 | — | pF |
| Input Offset Voltage | | –10 | — | +10 | mV |
| **Power Supply** | | | | | |
| Power Supply Rejection | | — | 0.18 | — | mV/V |
| Power-Up Time | | — | 3 | — | µs |
| Supply Current at DC | Mode 0 | — | 6.3 | 20 | µA |
| | Mode 1 | — | 3.4 | 10 | µA |
| | Mode 2 | — | 2.6 | 7.5 | µA |
| | Mode 3 | — | 0.6 | 3 | µA |

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



Downloaded from Arrow.com.

Downloaded from Arrow.com.

## 6.1. ADC0 Analog Multiplexer

ADC0 includes an analog multiplexer to enable multiple analog input sources. Any of the following may be selected as an input: P0.0–P3.0, the on-chip temperature sensor, the core power supply ($V_{DD}$), or ground (GND). **ADC0 is single-ended and all signals measured are with respect to GND.** The ADC0 input channels are selected using the ADC0MX register as described in SFR Definition 6.3.

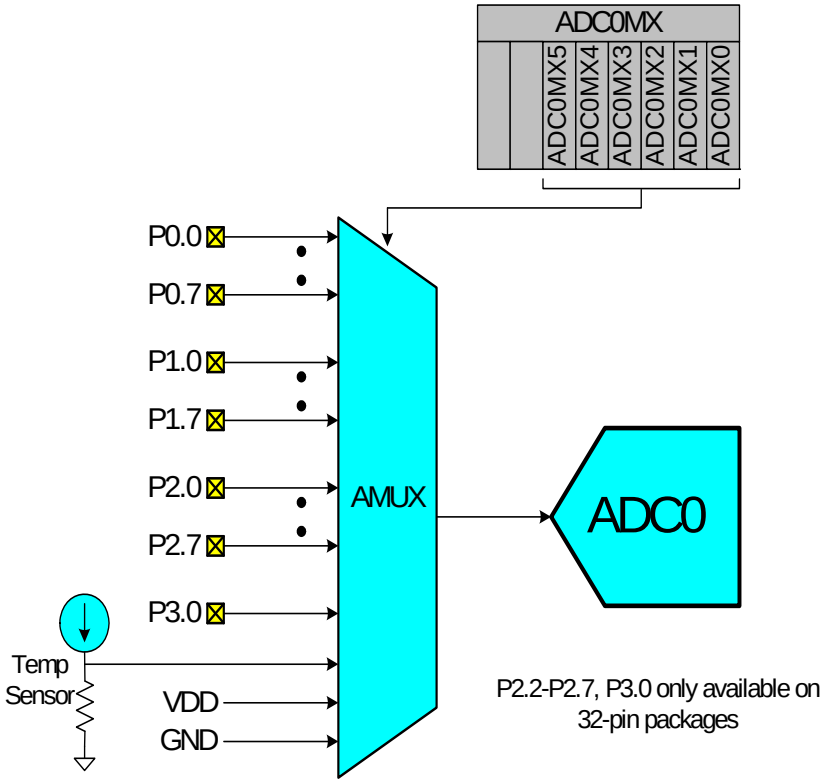



**Figure 6.2. ADC0 Multiplexer Block Diagram**

**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 “18. Port Input/Output” on page 147 for more Port I/O configuration details.



Downloaded from Arrow.com.

## SFR Definition 6.3. ADC0MX: ADC0 Channel Select

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | ADC0MX[5:0] | | | | | |
| Type | R | R | R/W | | | | | |
| Reset | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xBB; SFR Page = 0x00;

| Bit | Name | Function |
|---|---|---|
| 7:6 | Unused | Read = 00b; Write = Don't Care. |
| 5:0 | AMX0P[5:0] | **AMUX0 Positive Input Selection.**<br>000000: P0.0<br>000001: P0.1<br>000010: P0.2<br>000011: P0.3<br>000100: P0.4<br>000101: P0.5<br>000110: P0.6<br>000111: P0.7<br>001000: P1.0<br>001001: P1.1<br>001010: P1.2<br>001011: P1.3<br>001100: P1.4<br>001101: P1.5<br>001110: P1.6<br>001111: P1.7<br>010000: P2.0<br>010001: P2.1<br>010010: P2.2 (Only available on 32-pin package devices)<br>010011: P2.3 (Only available on 32-pin package devices)<br>010100: P2.4 (Only available on 32-pin package devices)<br>010101: P2.5 (Only available on 32-pin package devices)<br>010110: P2.6 (Only available on 32-pin package devices)<br>010111: P2.7 (Only available on 32-pin package devices)<br>011000: P3.0 (Only available on 32-pin package devices)<br>011001–101111: Reserved<br>110000: Temp Sensor<br>110001: $V_{DD}$<br>110010–111111: GND |



Downloaded from Arrow.com.

## 6.2. Temperature Sensor

An on-chip temperature sensor is included on the C8051F54x devices which can be directly accessed via the ADC multiplexer in single-ended configuration. To use the ADC to measure the temperature sensor, the ADC multiplexer channel should be configured to connect to the temperature sensor. The temperature sensor transfer function is shown in Figure 6.3. The output voltage ($V_{TEMP}$) is the positive ADC input is selected by bits AD0MX[4:0] in register ADC0MX. The TEMPE bit in register REF0CN enables/disables the temperature sensor, as described in SFR Definition 7.1. 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 Table 6.10 for the slope and offset parameters of the temperature sensor.

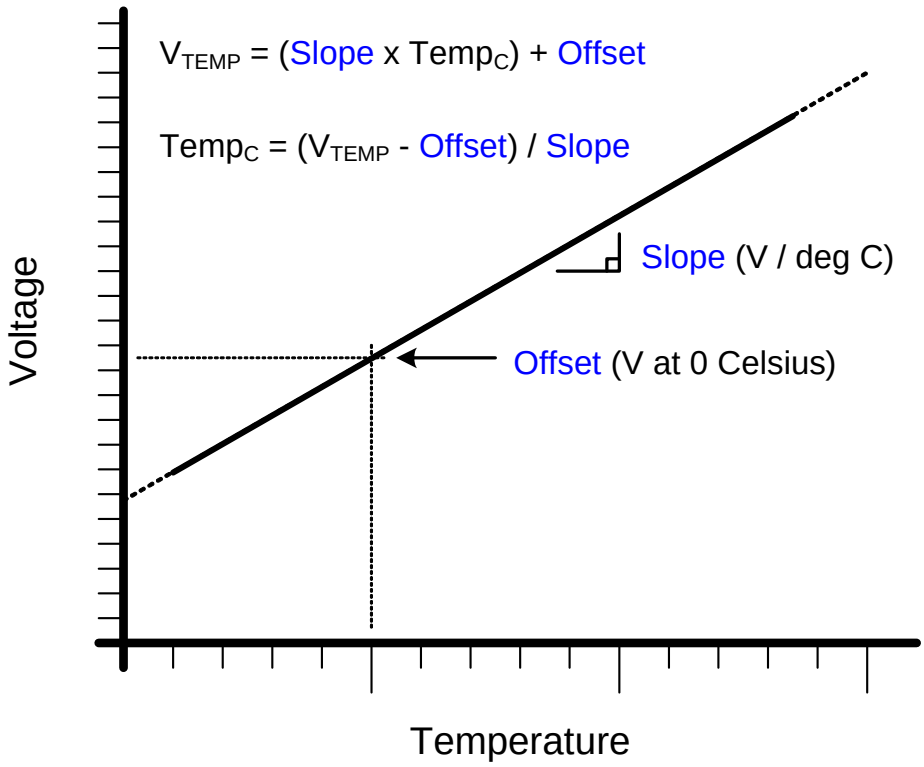



**Figure 6.3. Temperature Sensor Transfer Function**



Downloaded from Arrow.com.

# 7. Voltage Reference

The Voltage reference multiplexer on the C8051F54x devices 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 7.1). The REFSL bit in the Reference Control register (REF0CN, SFR Definition 7.1) 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 the ADC, Temperature Sensor, and internal oscillator. 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 Table 6.11.

The on-chip voltage reference circuit consists of a temperature stable bandgap voltage reference generator and a gain-of-two output buffer amplifier. The output voltage is selectable between 1.5 V and 2.25 V. 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. Electrical specifications for the on-chip voltage reference are given in Table 6.11.

**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 “18. Port Input/Output” on page 147 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.

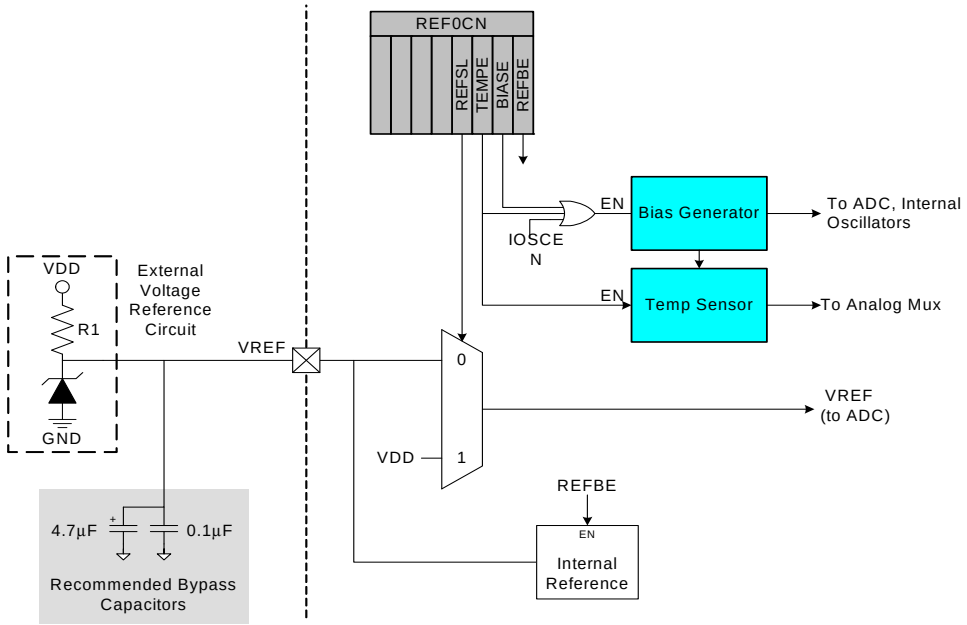



**Figure 7.1. Voltage Reference Functional Block Diagram**



Downloaded from Arrow.com.

## SFR Definition 7.1. REF0CN: Reference Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | ZTCEN | REFLV | 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; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:6 | Unused | Read = 00b; Write = don't care. |
| 5 | ZTCEN | **Zero Temperature Coefficient Bias Enable Bit.**<br>This bit must be set to 1b before entering oscillator suspend mode.<br>0: ZeroTC Bias Generator automatically enabled when required.<br>1: ZeroTC Bias Generator forced on. |
| 4 | REFLV | **Voltage Reference Output Level Select.**<br>This bit selects the output voltage level for the internal voltage reference<br>0: Internal voltage reference set to 1.5 V.<br>1: Internal voltage reference set to 2.20 V. |
| 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. |



Downloaded from Arrow.com.

# 8. Comparators

The C8051F54x devices include two on-chip programmable voltage Comparators. A block diagram of the comparators is shown in Figure 8.1, where “n” is the comparator number (0 or 1). The two Comparators operate identically except that Comparator0 can also be used a reset source. For input selection details, refer to SFR Definition 8.5 and SFR Definition 8.6.

Each 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, CP1), or an asynchronous “raw” output (CP0A, CP1A). The asynchronous signal is available even when the system clock is not active. This allows the Comparators to operate and generate an output with the device in STOP mode. When assigned to a Port pin, the Comparator outputs may be configured as open drain or push-pull (see Section “18.4. Port I/O Initialization” on page 152). Comparator0 may also be used as a reset source (see Section “16.5. Comparator0 Reset” on page 133).

The Comparator0 inputs are selected in the CPT0MX register (SFR Definition 8.5). The CMX0P1-CMX0P0 bits select the Comparator0 positive input; the CMX0N1-CMX0N0 bits select the Comparator0 negative input. The Comparator1 inputs are selected in the CPT1MX register (SFR Definition 8.6). The CMX1P1-CMX1P0 bits select the Comparator1 positive input; the CMX1N1-CMX1N0 bits select the Comparator1 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 “18.1. Port I/O Modes of Operation” on page 148).

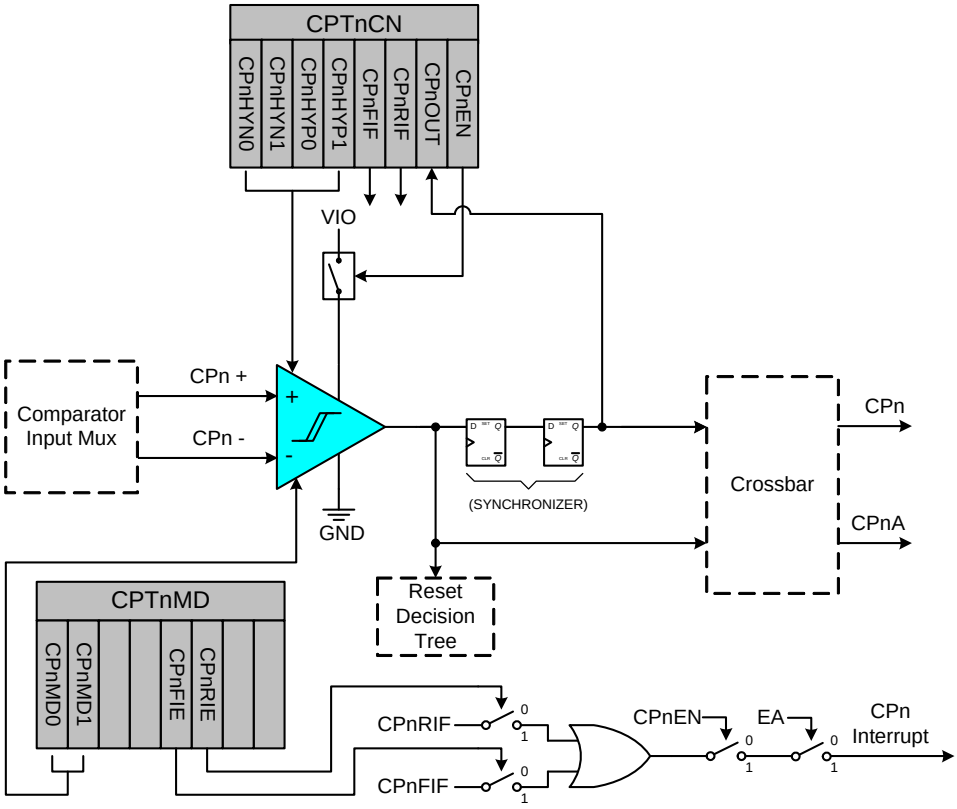



**Figure 8.1. Comparator Functional Block Diagram**



Downloaded from Arrow.com.

Comparator outputs can be polled in software, used as an interrupt source, and/or routed to a Port pin. When routed to a Port pin, Comparator outputs are 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 “18.3. Priority Crossbar Decoder” on page 150 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 Table 6.12.

The Comparator response time may be configured in software via the CPTnMD registers (see SFR Definition 8.2). Selecting a longer response time reduces the Comparator supply current. See Table 6.12 for complete timing and supply current requirements.

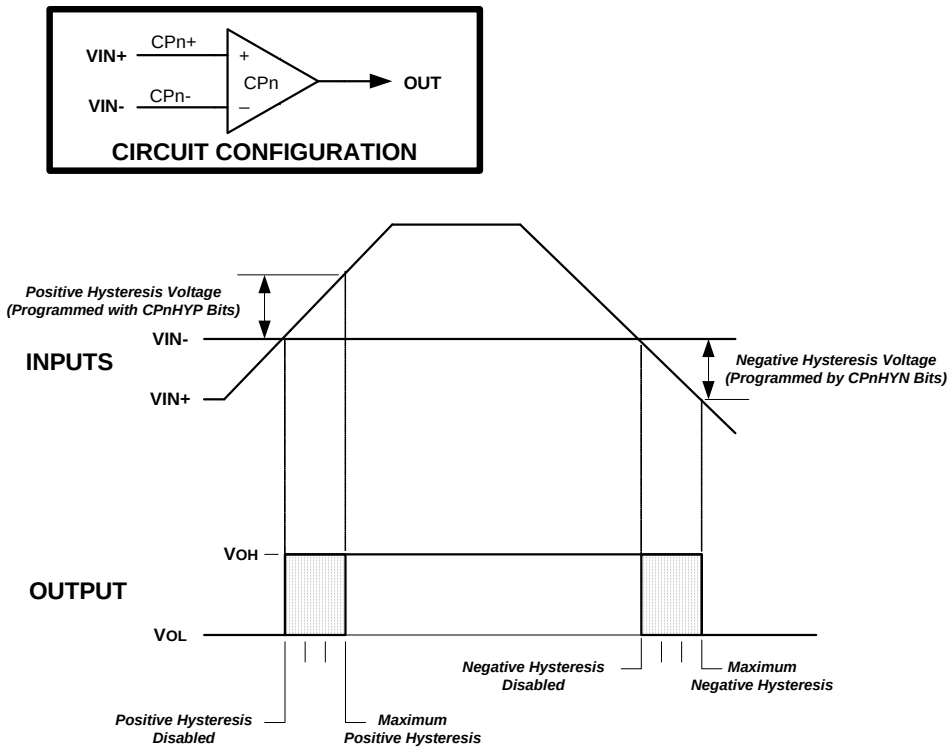



**Figure 8.2. Comparator Hysteresis Plot**

Comparator hysteresis is software-programmable via its Comparator Control register CPTnCN.

The amount of negative hysteresis voltage is determined by the settings of the CPnHYN bits. As shown in Figure 8.2, various levels 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 CPnHYP bits.

Comparator interrupts can be generated on both rising-edge and falling-edge output transitions. (For Interrupt enable and priority control, see “13. Interrupts” .) The CPnFIF flag is set to 1 upon a Comparator falling-edge, and the CPnRIF flag is set to 1 upon the Comparator rising-edge. Once set, these bits remain set until cleared by software. The output state of the Comparator can be obtained at any time by reading the CPnOUT bit. The Comparator is enabled by setting the CPnEN bit to 1, and is disabled by clearing this bit to 0.



Downloaded from Arrow.com.

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 8.1. 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 = 0x9A; SFR Page = 0x00

| 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. |



Downloaded from Arrow.com.

## SFR Definition 8.2. 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 = 0x9B; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:6 | 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 | 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) |



Downloaded from Arrow.com.

## SFR Definition 8.3. CPT1CN: Comparator1 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CP1EN | CP1OUT | CP1RIF | CP1FIF | CP1HYP[1:0] | | CP1HYN[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 = 0x9D; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7 | CP1EN | **Comparator1 Enable Bit.**<br>0: Comparator1 Disabled.<br>1: Comparator1 Enabled. |
| 6 | CP1OUT | **Comparator1 Output State Flag.**<br>0: Voltage on CP1+ < CP1–.<br>1: Voltage on CP1+ > CP1–. |
| 5 | CP1RIF | **Comparator1 Rising-Edge Flag. Must be cleared by software.**<br>0: No Comparator1 Rising Edge has occurred since this flag was last cleared.<br>1: Comparator1 Rising Edge has occurred. |
| 4 | CP1FIF | **Comparator1 Falling-Edge Flag. Must be cleared by software.**<br>0: No Comparator1 Falling-Edge has occurred since this flag was last cleared.<br>1: Comparator1 Falling-Edge has occurred. |
| 3:2 | CP1HYP[1:0] | **Comparator1 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 | CP1HYN[1:0] | **Comparator1 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. |



Downloaded from Arrow.com.

## SFR Definition 8.4. CPT1MD: Comparator1 Mode Selection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | CP1RIE | CP1FIE | | | CP1MD[1:0] | |
| Type | R | R | R/W | R/W | R | R | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

SFR Address = 0x9E; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:6 | Unused | Read = 00b, Write = Don't Care. |
| 5 | CP1RIE | **Comparator1 Rising-Edge Interrupt Enable.**<br>0: Comparator1 Rising-edge interrupt disabled.<br>1: Comparator1 Rising-edge interrupt enabled. |
| 4 | CP1FIE | **Comparator1 Falling-Edge Interrupt Enable.**<br>0: Comparator1 Falling-edge interrupt disabled.<br>1: Comparator1 Falling-edge interrupt enabled. |
| 3:2 | Unused | Read = 00b, Write = don't care. |
| 1:0 | CP1MD[1:0] | **Comparator1 Mode Select.**<br>These bits affect the response time and power consumption for Comparator1.<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) |



Downloaded from Arrow.com.

## 8.1. Comparator Multiplexer

C8051F54x devices include an analog input multiplexer for each of the comparators to connect Port I/O pins to the comparator inputs. The Comparator0 inputs are selected in the CPT0MX register (SFR Definition 8.5). The CMX0P3–CMX0P0 bits select the Comparator0 positive input; the CMX0N3–CMX0N0 bits select the Comparator0 negative input. Similarly, the Comparator1 inputs are selected in the CPT1MX register using the CMX1P3-CMX1P0 bits and CMX1N3-CMX1N0 bits. The same pins are available to both multiplexers at the same time and can be used by both comparators simultaneously.

**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 “18.6. Special Function Registers for Accessing and Configuring Port I/O” on page 161).

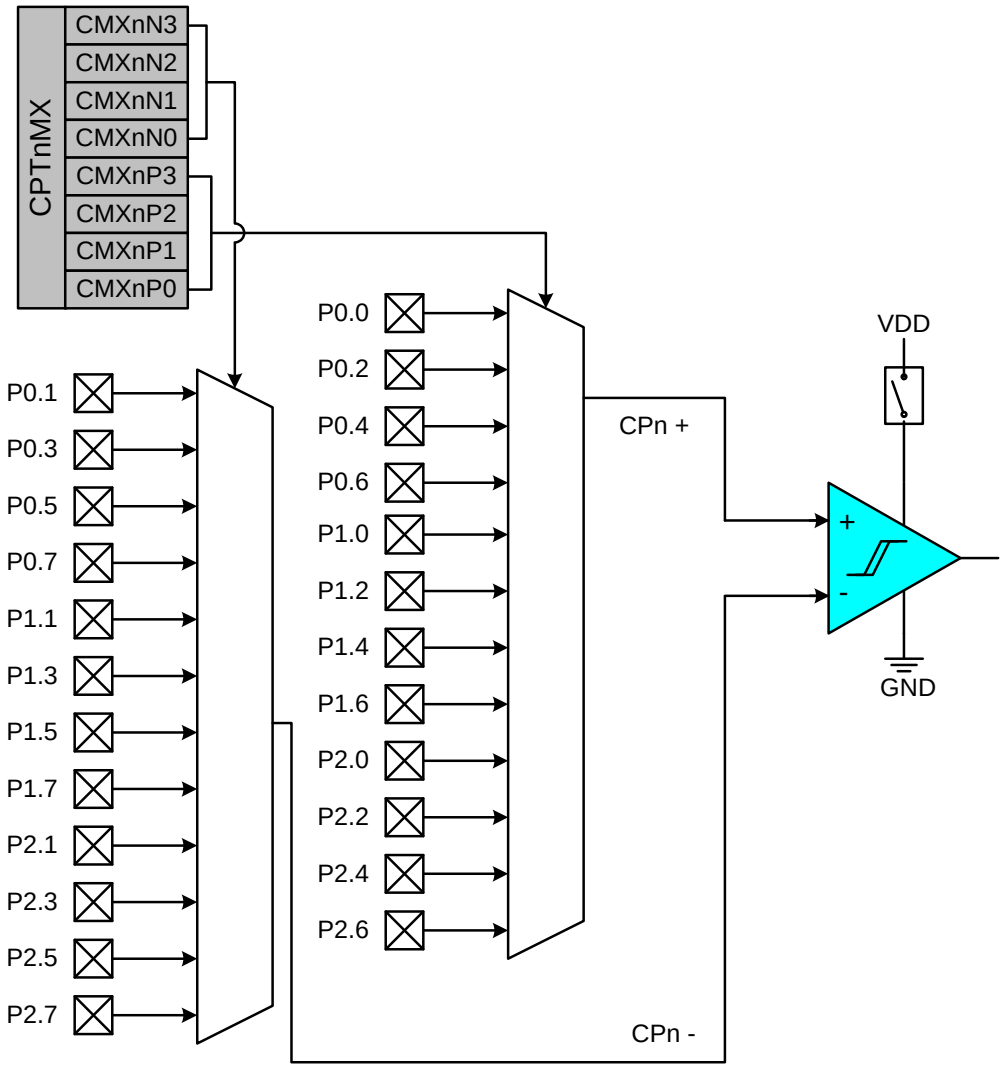



**Figure 8.3. Comparator Input Multiplexer Block Diagram**



Downloaded from Arrow.com.

## SFR Definition 8.5. 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 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

SFR Address = 0x9C; SFR Page = 0x00

| 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<br>1001: P2.3 (only available on 32-pin devices)<br>1010: P2.5 (only available on 32-pin devices)<br>1011: P2.7 (only available on 32-pin devices)<br>1100–1111: 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<br>1001: P2.2 (only available on 32-pin devices)<br>1010: P2.4 (only available on 32-pin devices)<br>1011: P2.6 (only available on 32-pin devices)<br>1100–1111: None |



Downloaded from Arrow.com.

## SFR Definition 8.6. CPT1MX: Comparator1 MUX Selection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CMX1N[3:0] | | | | CMX1P[3:0] | | | |
| Type | R/W | | | | R/W | | | |
| Reset | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |

SFR Address = 0x9F; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:4 | CMX1N[3:0] | **Comparator1 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<br>1001: P2.3 (only available on 32-pin devices)<br>1010: P2.5 (only available on 32-pin devices)<br>1011: P2.7 (only available on 32-pin devices)<br>1100–1111: None |
| 3:0 | CMX1P[3:0] | **Comparator1 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<br>1001: P2.2 (only available on 32-pin devices)<br>1010: P2.4 (only available on 32-pin devices)<br>1011: P2.6 (only available on 32-pin devices)<br>1100–1111: None |



Downloaded from Arrow.com.

# 9. Voltage Regulator (REG0)

C8051F54x devices include an on-chip low dropout voltage regulator (REG0). The input to REG0 at the $V_{REGIN}$ pin can be as high as 5.25 V. The output can be selected by software to 2.1 V or 2.6 V. When enabled, the output of REG0 appears on the $V_{DD}$ pin, powers the microcontroller core, and can be used to power external devices. On reset, REG0 is enabled and can be disabled by software.

The Voltage regulator can generate an interrupt (if enabled by EREG0, EIE2.0) that is triggered whenever the $V_{REGIN}$ input voltage drops below the dropout threshold voltage. This dropout interrupt has no pending flag and the recommended procedure to use it is as follows:

1. Wait enough time to ensure the $V_{REGIN}$ input voltage is stable
2. Enable the dropout interrupt (EREG0, EIE2.0) and select the proper priority (PREG0, EIP2.0)
3. If triggered, inside the interrupt disable it (clear EREG0, EIE2.0), execute all procedures necessary to protect your application (put it in a safe mode and leave the interrupt now disabled).
4. In the main application, now running in the safe mode, regularly check the DROPOUT bit (REG0CN.0). Once it is cleared by the regulator hardware, the application can enable the interrupt again (EREG0, EIE1.6) and return to the normal mode operation.

The input ($V_{REGIN}$) and output ($V_{DD}$) of the voltage regulator should both be bypassed with a large capacitor (4.7 µF + 0.1 µF) to ground as shown in Figure 9.1 below. This capacitor will eliminate power spikes and provide any immediate power required by the microcontroller. The settling time associated with the voltage regulator is shown in Table 6.8 on page 54.

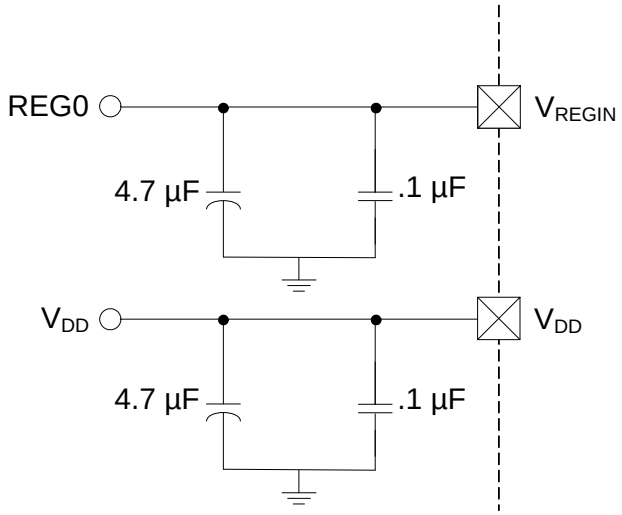



**Figure 9.1. External Capacitors for Voltage Regulator Input/Output—Regulator Enabled**

If the internal voltage regulator is not used, the $V_{REGIN}$ input should be tied to $V_{DD}$, as shown in Figure 9.2.



Downloaded from Arrow.com.

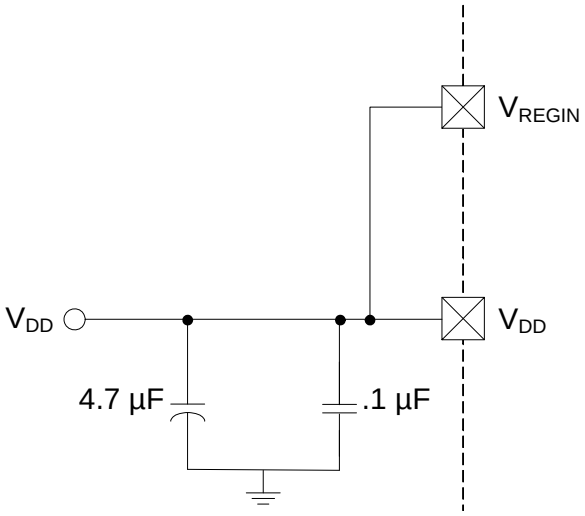



**Figure 9.2. External Capacitors for Voltage Regulator Input/Output—Regulator Disabled**

## SFR Definition 9.1. REG0CN: Regulator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | REGDIS | Reserved | | REG0MD | | | | DROPOUT |
| Type | R/W | R/W | R | R/W | R | R | R | R |
| Reset | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

SFR Address = 0xC9; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7 | REGDIS | **Voltage Regulator Disable Bit.**<br>0: Voltage Regulator Enabled<br>1: Voltage Regulator Disabled |
| 6 | Reserved | Read = 1b; Must Write 1b. |
| 5 | Unused | Read = 0b; Write = Don't Care. |
| 4 | REG0MD | **Voltage Regulator Mode Select Bit.**<br>0: Voltage Regulator Output is 2.1V.<br>1: Voltage Regulator Output is 2.6V. |
| 3:1 | Unused | Read = 000b. Write = Don't Care. |
| 0 | DROPOUT | **Voltage Regulator Dropout Indicator.**<br>0: Voltage Regulator is not in dropout<br>1: Voltage Regulator is in or near dropout. |



Downloaded from Arrow.com.

# 10. 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 25), 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 10.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 50 MIPS Peak Throughput with 50 MHz Clock
- 0 to 50 MHz Clock Frequency
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- On-chip Debug Logic
- Program and Data Memory Security

## 10.1. 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.



Downloaded from Arrow.com.

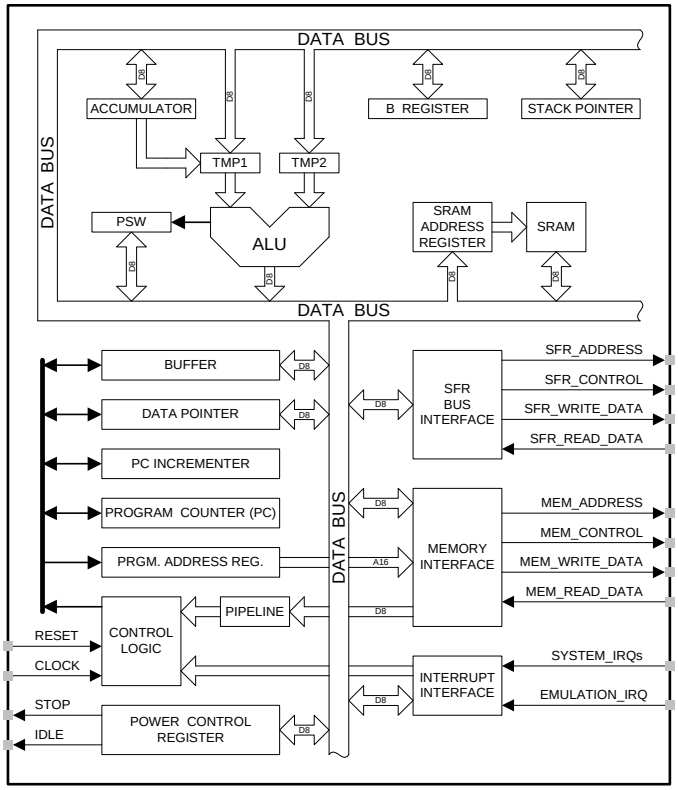



**Figure 10.1. CIP-51 Block Diagram**

With the CIP-51's maximum system clock at 50 MHz, it has a peak throughput of 50 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 |

**Programming and Debugging Support**

In-system programming of the Flash program memory and communication with on-chip debug support logic is accomplished via the Silicon Labs 2-Wire Development Interface (C2).

The on-chip debug support logic facilitates full speed in-circuit debugging, allowing the setting of hardware breakpoints, starting, stopping and single stepping through program execution (including interrupt service routines), examination of the program's call stack, and reading/writing the contents of registers and memory. This method of on-chip debugging is completely non-intrusive, requiring no RAM, Stack, timers, or other on-chip resources. C2 details can be found in Section “25. C2 Interface” on page 269.

The CIP-51 is supported by development tools from Silicon Labs and third party vendors. Silicon Labs provides an integrated development environment (IDE) including editor, debugger and programmer. The IDE's debugger and programmer interface to the CIP-51 via the C2 interface to provide fast and efficient in-system device programming and debugging. Third party macro assemblers and C compilers are also available.



Downloaded from Arrow.com.

## 10.2. 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.

### 10.2.1. 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 10.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.



Downloaded from Arrow.com.

### Table 10.1. 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 |

**Note:** Certain instructions take a variable number of clock cycles to execute depending on instruction alignment and the FLRT setting (SFR Definition 14.3).



Downloaded from Arrow.com.

**Table 10.1. CIP-51 Instruction Set Summary (Continued)**

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| XRL A, #data | Exclusive-OR immediate to A | 2 | 2 |
| XRL direct, A | Exclusive-OR A to direct byte | 2 | 2 |
| 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 |

**Note:** Certain instructions take a variable number of clock cycles to execute depending on instruction alignment and the FLRT setting (SFR Definition 14.3).



Downloaded from Arrow.com.

**Table 10.1. CIP-51 Instruction Set Summary (Continued)**

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| 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 |
| 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 | 4/5* |
| 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 |

**Note:** Certain instructions take a variable number of clock cycles to execute depending on instruction alignment and the FLRT setting (SFR Definition 14.3).



Downloaded from Arrow.com.

| 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 (two's 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 64 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. |

## 10.3. CIP-51 Register Descriptions

Following are descriptions of SFRs related to the operation of the CIP-51 System Controller. Reserved bits should not be set to logic l. Future product versions may use these bits to implement new features in which case the reset value of the bit will be logic 0, selecting the feature's default state. Detailed descriptions of the remaining SFRs are included in the sections of the datasheet associated with their corresponding system function.



Downloaded from Arrow.com.

## SFR Definition 10.1. 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; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | DPL[7:0] | **Data Pointer Low.**<br>The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed Flash memory or XRAM. |

## SFR Definition 10.2. 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; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | DPH[7:0] | **Data Pointer High.**<br>The DPH register is the high byte of the 16-bit DPTR. DPTR is used to access indirectly addressed Flash memory or XRAM. |



Downloaded from Arrow.com.

## SFR Definition 10.3. 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; SFR Page = All Pages

| 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 10.4. 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; SFR Page = All Pages; Bit-Addressable

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

## SFR Definition 10.5. 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; SFR Page = All Pages; Bit-Addressable

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



Downloaded from Arrow.com.

## SFR Definition 10.6. 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; SFR Page = All Pages; 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. |



Downloaded from Arrow.com.

## 10.4. Serial Number Special Function Registers (SFRs)

The C8051F54x devices include four SFRs, SN0 through SN3, that are pre-programmed during production with a unique, 32-bit serial number. The serial number provides a unique identification number for each device and can be read from the application firmware. If the serial number is not used in the application, these four registers can be used as general purpose SFRs.

### SFR Definition 10.7. SNn: Serial Number n

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SERNUMn[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | Varies—Unique 32-bit value | | | | | | | |

SFR Addresses: SN0 = 0xF9; SN1 = 0xFA; SN2 = 0xFB; SN3 = 0xFC; SFR Page = 0x0F;

| Bit | Name | Function |
|---|---|---|
| 7:0 | SERNUMn[7:0] | **Serial Number Bits.**<br>The four serial number registers form a 32-bit serial number, with SN3 as the most significant byte and SN0 as the least significant byte. |



Downloaded from Arrow.com.

# 11. 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 is shown in Figure 11.1

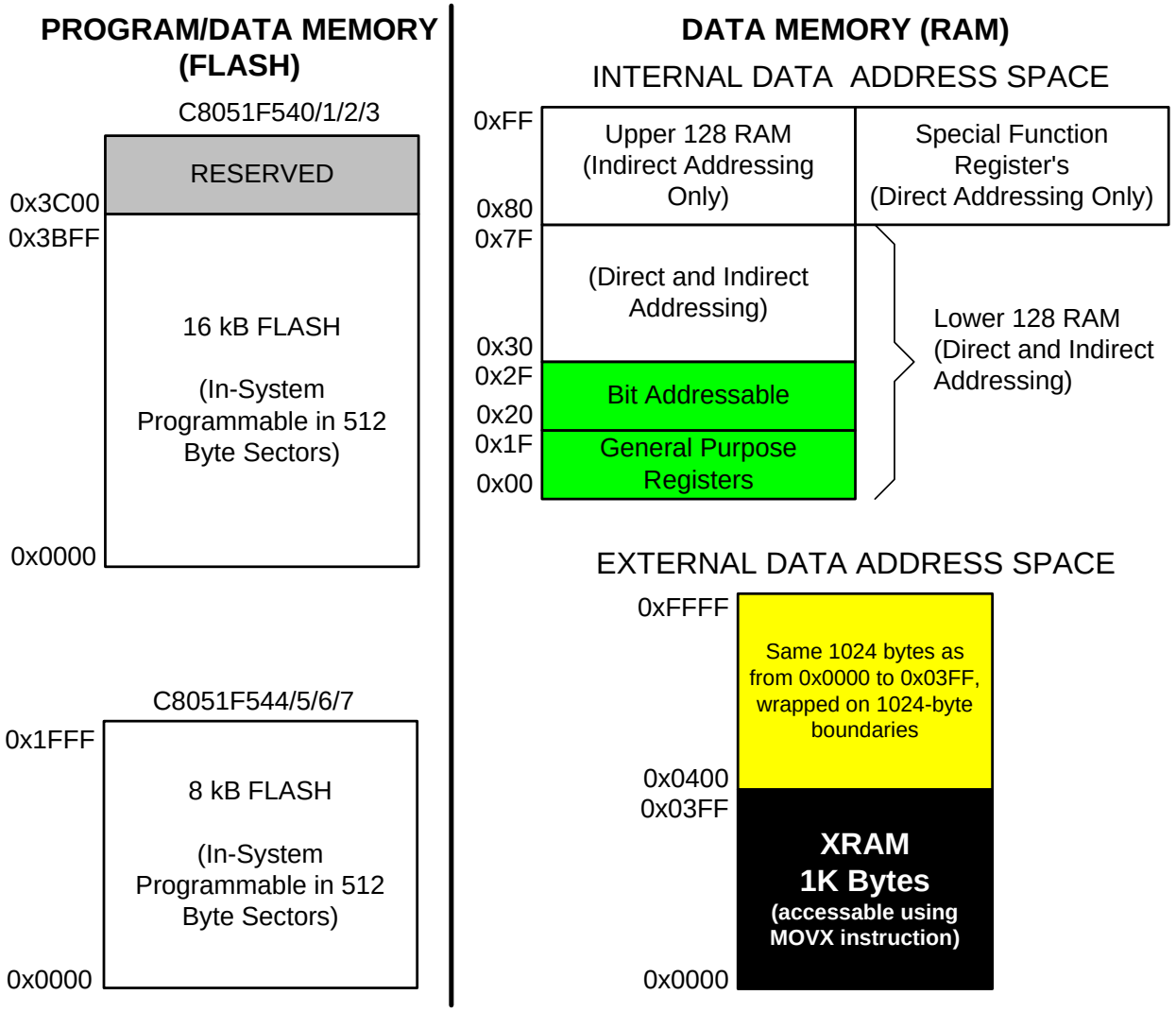



**Figure 11.1. C8051F54x Memory Map**

## 11.1. Program Memory

The CIP-51 core has a 64 kB program memory space. The C8051F54x devices implement 16 kB or 8 kB of this program memory space as in-system, re-programmable Flash memory, organized in a contiguous block from addresses 0x0000 to 0x3FFF in 16 kB devices and addresses 0x0000 to 0x1FFF in 8 kB devices. The address 0x3BFF in 16 kB devices and 0x1FFF in 8 kB devices serves as the security lock byte for the device. Addresses above 0x3BFF are reserved in the 16 kB devices.



Downloaded from Arrow.com.

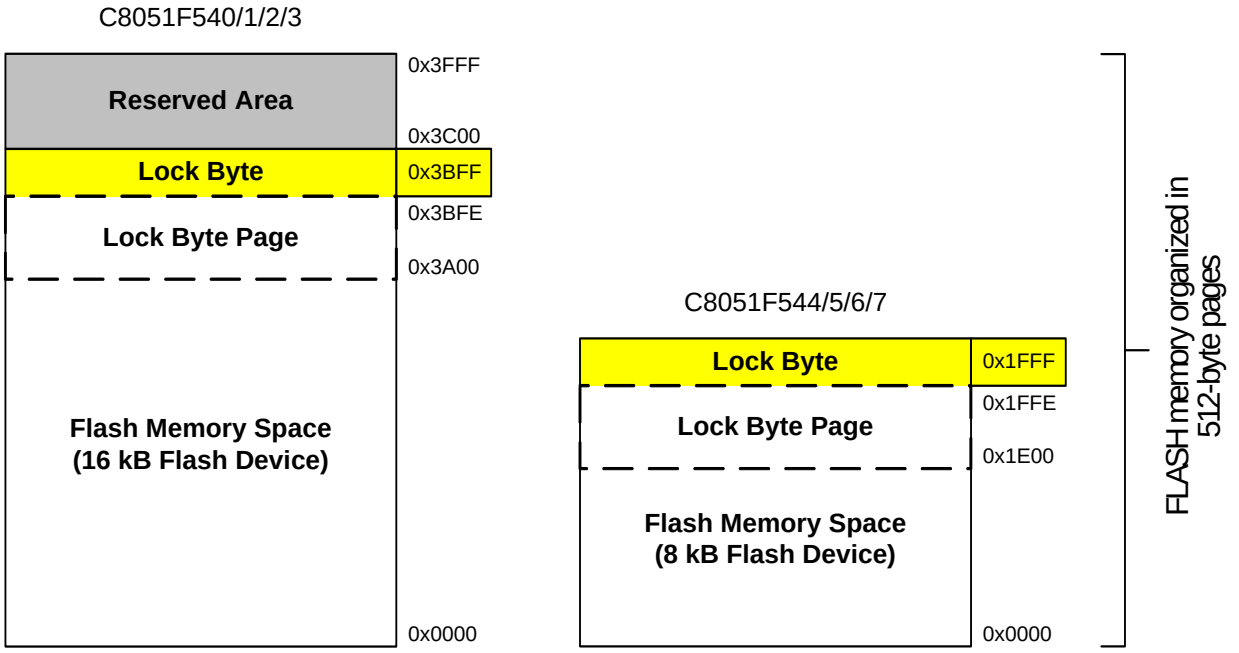



**Figure 11.2. Flash Program Memory Map**

### 11.1.1. MOVX Instruction and Program Memory

The MOVX instruction in an 8051 device is typically used to access external data memory. On the C8051F54x 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 C8051F54x to update program code and use the program memory space for non-volatile data storage. Refer to Section “14. Flash Memory” on page 117 for further details.

## 11.2. Data Memory

The C8051F54x devices include 1280 bytes of RAM data memory. 256 bytes of this memory is mapped into the internal RAM space of the 8051. The other 1024 bytes of this memory is on-chip “external” memory. The data memory map is shown in Figure 11.1 for reference.

### 11.2.1. 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 11.1 illustrates the data memory organization of the C8051F54x.



Downloaded from Arrow.com.

#### 11.2.1.1. 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 10.6). This allows fast context switching when entering subroutines and interrupt service routines. Indirect addressing modes use registers R0 and R1 as index registers.

#### 11.2.1.2. 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.

#### 11.2.1.3. 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.

## 11.3. External RAM

For C8051F54x devices, 1 kB of RAM are included on-chip and mapped into the external data memory space (XRAM). The external memory space may be accessed using the external move instruction (MOVX) and the data pointer (DPTR), or using the MOVX indirect addressing mode using R0 or R1. 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, shown in SFR Definition 11.1).

**Note:** The MOVX instruction can also be used for writing to the Flash memory. See Section "14. Flash Memory" on page 117 for details. The MOVX instruction accesses XRAM by default.

### 11.3.1. 16-Bit MOVX Example

The 16-bit form of the MOVX instruction accesses the memory location pointed to by the contents of the DPTR register. The following series of instructions reads the value of the byte at address 0x1234 into the accumulator A:

```
MOV    DPTR, #1234h        ; load DPTR with 16-bit address to read (0x1234)
MOVX   A, @DPTR            ; load contents of 0x1234 into accumulator A
```

The above example uses the 16-bit immediate MOV instruction to set the contents of DPTR. Alternately,



Downloaded from Arrow.com.

the DPTR can be accessed through the SFR registers DPH, which contains the upper 8-bits of DPTR, and DPL, which contains the lower 8-bits of DPTR.

### 11.3.2. 8-Bit MOVX Example

The 8-bit form of the MOVX instruction uses the contents of the EMI0CN SFR to determine the upper 8-bits of the effective address to be accessed and the contents of R0 or R1 to determine the lower 8-bits of the effective address to be accessed. The following series of instructions read the contents of the byte at address 0x1234 into the accumulator A.

```
MOV    EMI0CN, #12h        ; load high byte of address into EMI0CN
MOV    R0, #34h            ; load low byte of address into R0 (or R1)
MOVX   a, @R0              ; load contents of 0x1234 into accumulator A
```

## SFR Definition 11.1. EMI0CN: External Memory Interface Control

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

SFR Address = 0xAA; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | PGSEL[7:0] | **XRAM Page Select Bits.**<br>The XRAM Page Select Bits provide 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.<br>0x00: 0x0000 to 0x00FF<br>0x01: 0x0100 to 0x01FF<br>...<br>0xFE: 0xFE00 to 0xFEFF<br>0xFF: 0xFF00 to 0xFFFF |



Downloaded from Arrow.com.

# 12. 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 C8051F54x'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 C8051F54x. This allows the addition of new functionality while retaining compatibility with the MCS-51™ instruction set. Table 12.2 lists the SFRs implemented in the C8051F54x 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 unoccupied addresses in the SFR space will have an indeterminate effect and should be avoided. Refer to the corresponding pages of the data sheet, as indicated in Table 12.2, for a detailed description of each register.

## 12.1. SFR Paging

The CIP-51 features SFR paging, allowing the device to map many SFRs into the 0x80 to 0xFF memory address space. The SFR memory space has 256 *pages*. In this way, each memory location from 0x80 to 0xFF can access up to 256 SFRs. The C8051F54x family of devices utilizes two SFR pages: 0x00 and 0x0F. SFR pages are selected using the Special Function Register Page Selection register, SFRPAGE (see SFR Definition 11.3). The procedure for reading and writing an SFR is as follows:

1. Select the appropriate SFR page number using the SFRPAGE register.
2. Use direct accessing mode to read or write the special function register (MOV instruction).

## 12.2. Interrupts and SFR Paging

When an interrupt occurs, the SFR Page Register will automatically switch to the SFR page containing the flag bit that caused the interrupt. The automatic SFR Page switch function conveniently removes the burden of switching SFR pages from the interrupt service routine. Upon execution of the RETI instruction, the SFR page is automatically restored to the SFR Page in use prior to the interrupt. This is accomplished via a three-byte *SFR Page Stack*. The top byte of the stack is SFRPAGE, the current SFR Page. The second byte of the SFR Page Stack is SFRNEXT. The third, or bottom byte of the SFR Page Stack is SFRLAST. Upon an interrupt, the current SFRPAGE value is pushed to the SFRNEXT byte, and the value of SFRNEXT is pushed to SFRLAST. Hardware then loads SFRPAGE with the SFR Page containing the flag bit associated with the interrupt. On a return from interrupt, the SFR Page Stack is popped resulting in the value of SFRNEXT returning to the SFRPAGE register, thereby restoring the SFR page context without software intervention. The value in SFRLAST (0x00 if there is no SFR Page value in the bottom of the stack) of the stack is placed in SFRNEXT register. If desired, the values stored in SFRNEXT and SFRLAST may be modified during an interrupt, enabling the CPU to return to a different SFR Page upon execution of the RETI instruction (on interrupt exit). Modifying registers in the SFR Page Stack does not cause a push or pop of the stack. Only interrupt calls and returns will cause push/pop operations on the SFR Page Stack.

On the C8051F54x devices, vectoring to an interrupt will switch SFRPAGE to page 0x00.



Downloaded from Arrow.com.

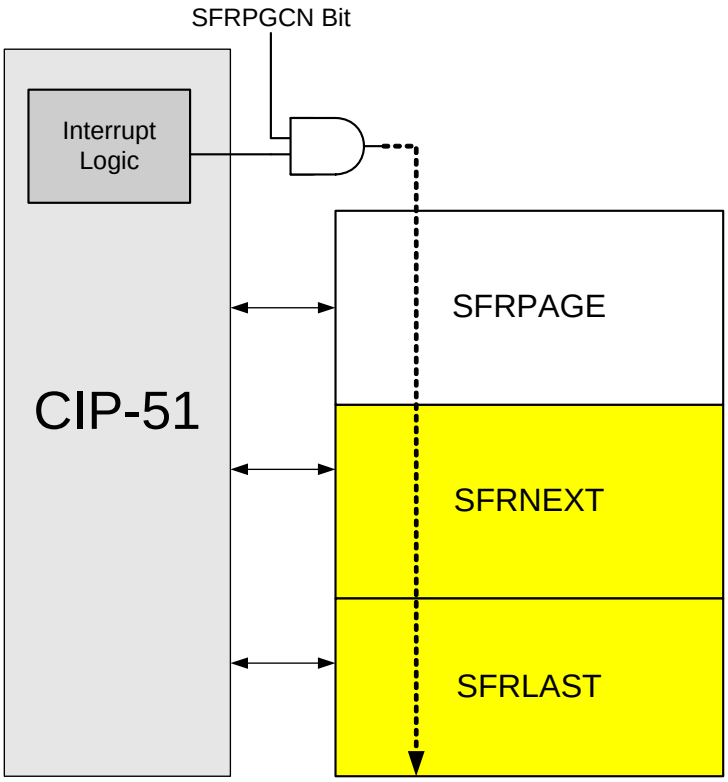



**Figure 12.1. SFR Page Stack**

Automatic hardware switching of the SFR Page on interrupts may be enabled or disabled as desired using the SFR Automatic Page Control Enable Bit located in the SFR Page Control Register (SFR0CN). This function defaults to “enabled” upon reset. In this way, the autoswitching function will be enabled unless disabled in software.

A summary of the SFR locations (address and SFR page) are provided in Table 12.2 in the form of an SFR memory map. Each memory location in the map has an SFR page row, denoting the page in which that SFR resides. Certain SFRs are accessible from ALL SFR pages, and are denoted by the “(ALL PAGES)” designation. For example, the Port I/O registers P0, P1, P2, and P3 all have the “(ALL PAGES)” designation, indicating these SFRs are accessible from all SFR pages regardless of the SFRPAGE register value.



Downloaded from Arrow.com.

## 12.3. SFR Page Stack Example

The following is an example that shows the operation of the SFR Page Stack during interrupts. In this example, the SFR Control register is left in the default enabled state (i.e., SFRPGEN = 1), and the CIP-51 is executing in-line code that is writing values to SMBus Address Register (SFR “SMB0ADR”, located at address 0xB9 on SFR Page 0x0F). The device is also using the SPI peripheral (SPI0) and the Programmable Counter Array (PCA0) peripheral to generate a PWM output. The PCA is timing a critical control function in its interrupt service routine, and so its associated ISR is set to high priority. At this point, the SFR page is set to access the SMB0ADR SFR (SFRPAGE = 0x0F). See Figure 12.2.

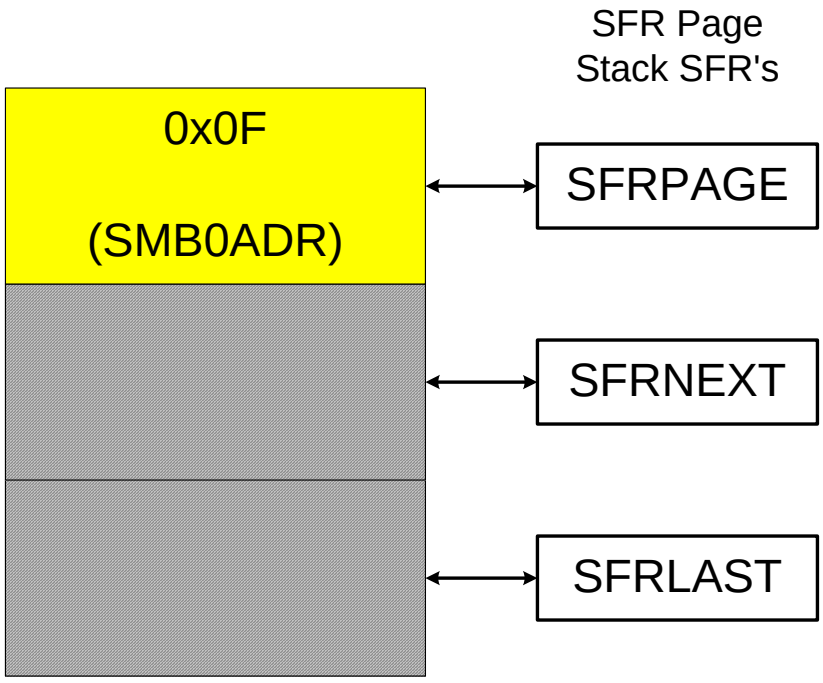



**Figure 12.2. SFR Page Stack While Using SFR Page 0x0 To Access SMB0ADR**

While CIP-51 executes in-line code (writing a value to SMB0ADR in this example), the SPI0 Interrupt occurs. The CIP-51 vectors to the SPI0 ISR and pushes the current SFR Page value (SFR Page 0x0F) into SFRNEXT in the SFR Page Stack. The SFR page needed to access SPI0’s SFRs is then automatically placed in the SFRPAGE register (SFR Page 0x00). SFRPAGE is considered the “top” of the SFR Page Stack. Software can now access the SPI0 SFRs. Software may switch to any SFR Page by writing a new value to the SFRPAGE register at any time during the SPI0 ISR to access SFRs that are not on SFR Page 0x00. See Figure 12.3.



Downloaded from Arrow.com.

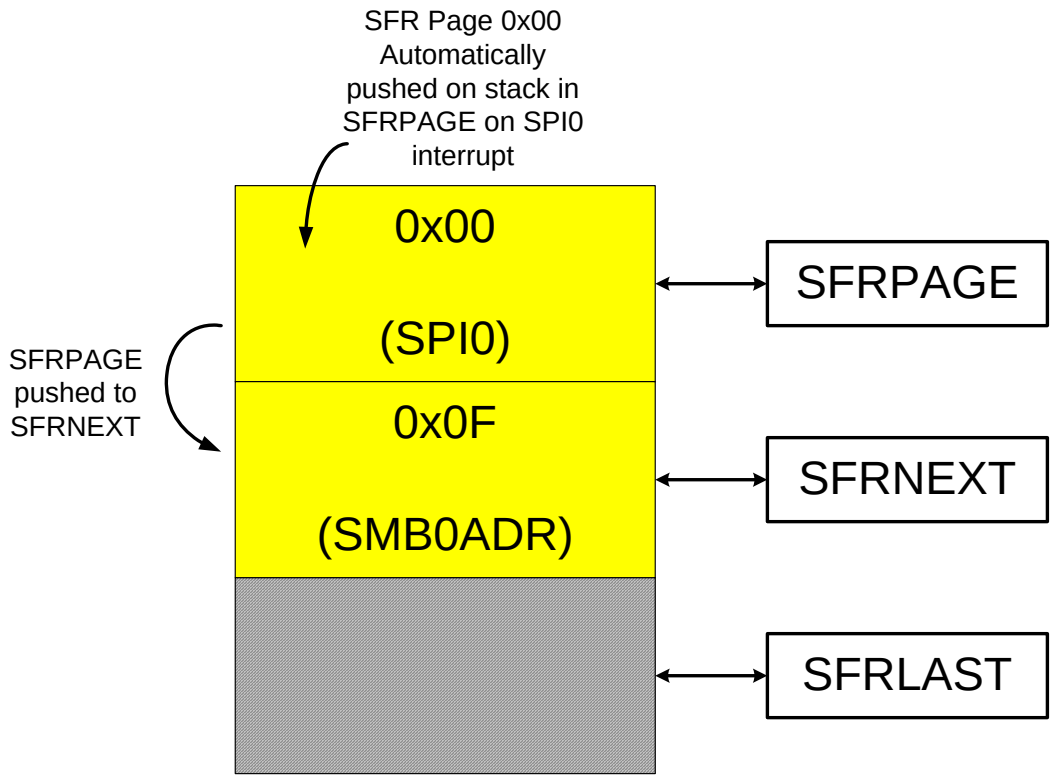



**Figure 12.3. SFR Page Stack After SPI0 Interrupt Occurs**

While in the SPI0 ISR, a PCA interrupt occurs. Recall the PCA interrupt is configured as a high priority interrupt, while the SPI0 interrupt is configured as a *low* priority interrupt. Thus, the CIP-51 will now vector to the high priority PCA ISR. Upon doing so, the CIP-51 will automatically place the SFR page needed to access the PCA's special function registers into the SFRPAGE register, SFR Page 0x00. The value that was in the SFRPAGE register before the PCA interrupt (SFR Page 0x00 for SPI00) is pushed down the stack into SFRNEXT. Likewise, the value that was in the SFRNEXT register before the PCA interrupt (in this case SFR Page 0x0F for SMB0ADR) is pushed down to the SFRLAST register, the "bottom" of the stack. Note that a value stored in SFRLAST (via a previous software write to the SFRLAST register) will be overwritten. See Figure 12.4.



Downloaded from Arrow.com.

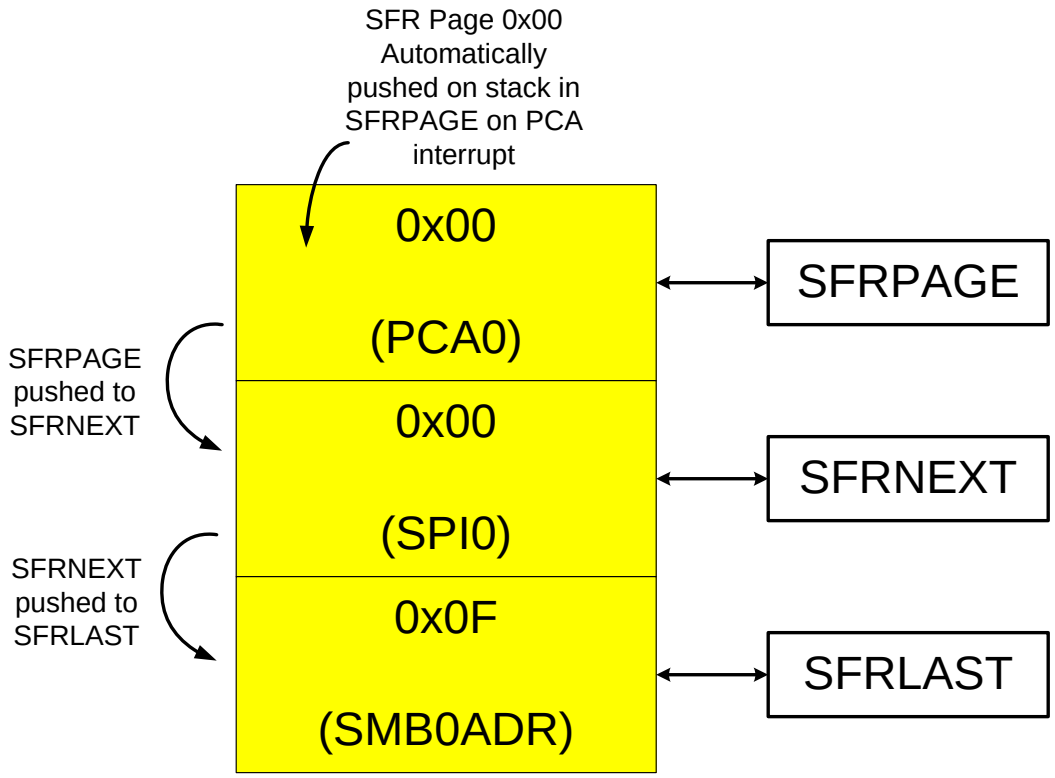



**Figure 12.4. SFR Page Stack Upon PCA Interrupt Occurring During a SPI0 ISR**

On exit from the PCA interrupt service routine, the CIP-51 will return to the SPI0 ISR. On execution of the RETI instruction, SFR Page 0x00 used to access the PCA registers will be automatically popped off of the SFR Page Stack, and the contents of the SFRNEXT register will be moved to the SFRPAGE register. Software in the SPI0 ISR can continue to access SFRs as it did prior to the PCA interrupt. Likewise, the contents of SFRLAST are moved to the SFRNEXT register. Recall this was the SFR Page value 0x0F being used to access SMB0ADR before the SPI0 interrupt occurred. See Figure 12.5.



Downloaded from Arrow.com.

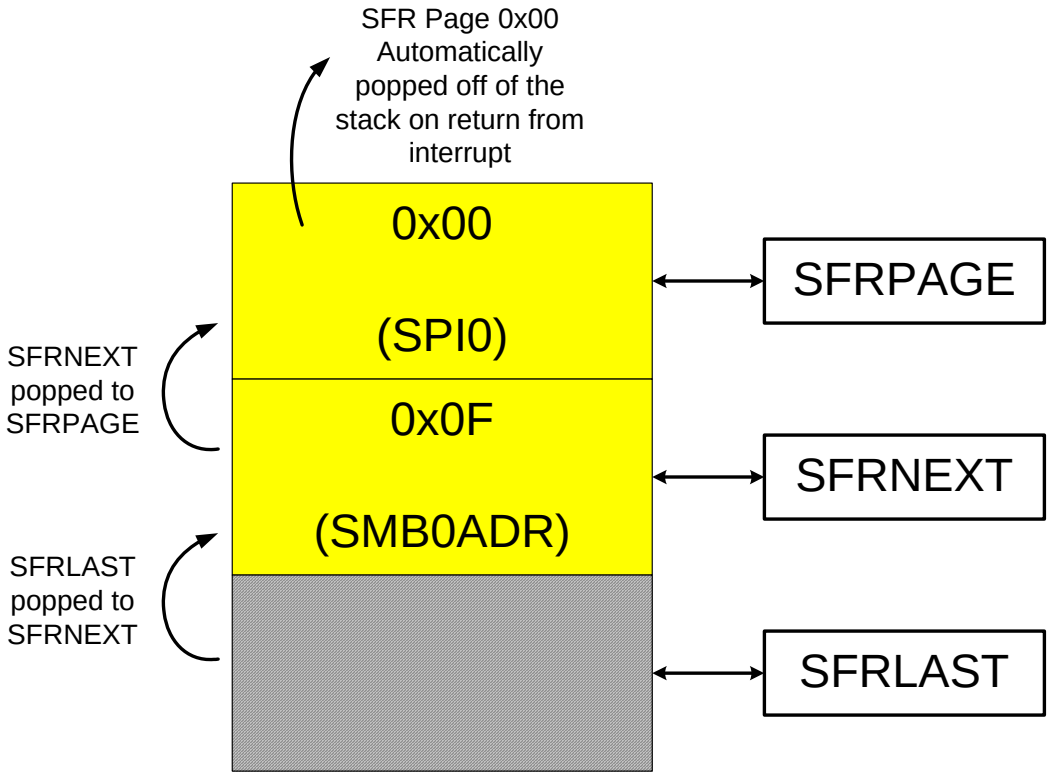



**Figure 12.5. SFR Page Stack Upon Return From PCA Interrupt**

On the execution of the RETI instruction in the SPI0 ISR, the value in SFRPAGE register is overwritten with the contents of SFRNEXT. The CIP-51 may now access the SMB0ADR register as it did prior to the interrupts occurring. See Figure 12.6.



Downloaded from Arrow.com.

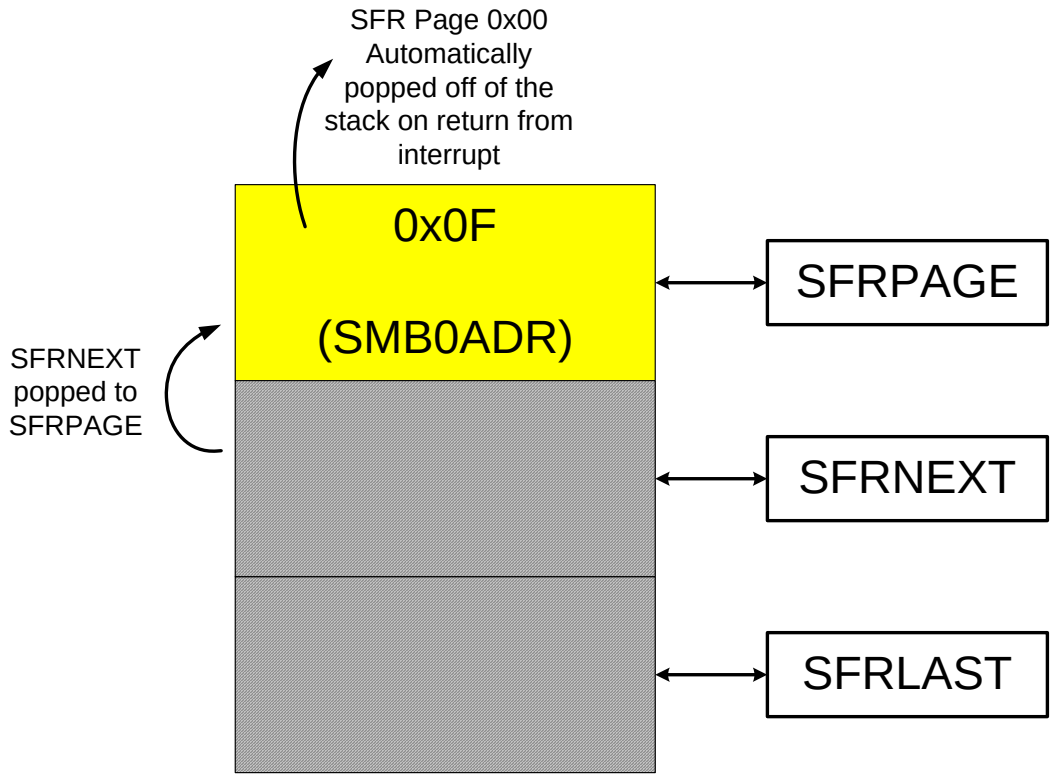



**Figure 12.6. SFR Page Stack Upon Return From SPI0 Interrupt**

In the example above, all three bytes in the SFR Page Stack are accessible via the SFRPAGE, SFRNEXT, and SFRLAST special function registers. If the stack is altered while servicing an interrupt, it is possible to return to a different SFR Page upon interrupt exit than selected prior to the interrupt call. Direct access to the SFR Page stack can be useful to enable real-time operating systems to control and manage context switching between multiple tasks.

Push operations on the SFR Page Stack only occur on interrupt service, and pop operations only occur on interrupt exit (execution on the RETI instruction). The automatic switching of the SFRPAGE and operation of the SFR Page Stack as described above can be disabled in software by clearing the SFR Automatic Page Enable Bit (SFRPGEN) in the SFR Page Control Register (SFR0CN). See SFR Definition 12.1.



Downloaded from Arrow.com.

## SFR Definition 12.1. SFR0CN: SFR Page Control

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

SFR Address = 0x84; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care |
| 0 | SFRPGEN | **SFR Automatic Page Control Enable.**<br>Upon interrupt, the C8051 Core will vector to the specified interrupt service routine and automatically switch the SFR page to the corresponding peripheral or function's SFR page. This bit is used to control this autopaging function.<br>0: SFR Automatic Paging disabled. The C8051 core will not automatically change to the appropriate SFR page (i.e., the SFR page that contains the SFRs for the peripheral/function that was the source of the interrupt).<br>1: SFR Automatic Paging enabled. Upon interrupt, the C8051 will switch the SFR page to the page that contains the SFRs for the peripheral or function that is the source of the interrupt. |



Downloaded from Arrow.com.

## SFR Definition 12.2. SFRPAGE: SFR Page

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

SFR Address = 0xA7; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | SFRPAGE[7:0] | **SFR Page Bits.**<br>Represents the SFR Page the C8051 core uses when reading or modifying SFRs.<br>Write: Sets the SFR Page.<br>Read: Byte is the SFR page the C8051 core is using.<br>When enabled in the SFR Page Control Register (SFR0CN), the C8051 core will automatically switch to the SFR Page that contains the SFRs of the corresponding peripheral/function that caused the interrupt, and return to the previous SFR page upon return from interrupt (unless SFR Stack was altered before a returning from the interrupt). SFRPAGE is the top byte of the SFR Page Stack, and push/pop events of this stack are caused by interrupts (and not by reading/writing to the SFRPAGE register) |



Downloaded from Arrow.com.

## SFR Definition 12.3. SFRNEXT: SFR Next

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

SFR Address = 0x85; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | SFRNEXT[7:0] | **SFR Page Bits.**<br>This is the value that will go to the SFR Page register upon a return from interrupt.<br>Write: Sets the SFR Page contained in the second byte of the SFR Stack. This will cause the SFRPAGE SFR to have this SFR page value upon a return from interrupt.<br>Read: Returns the value of the SFR page contained in the second byte of the SFR stack.<br>SFR page context is retained upon interrupts/return from interrupts in a 3 byte SFR Page Stack: SFRPAGE is the first entry, SFRNEXT is the second, and SFRLAST is the third entry. The SFR stack bytes may be used alter the context in the SFR Page Stack, and will not cause the stack to “push” or “pop”. Only interrupts and return from interrupts cause pushes and pops of the SFR Page Stack. |



Downloaded from Arrow.com.

## SFR Definition 12.4. SFRLAST: SFR Last

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

SFR Address = 0xA7; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | SFRLAST[7:0] | **SFR Page Stack Bits.**<br>This is the value that will go to the SFRNEXT register upon a return from interrupt.<br>Write: Sets the SFR Page in the last entry of the SFR Stack. This will cause the SFRNEXT SFR to have this SFR page value upon a return from interrupt.<br>Read: Returns the value of the SFR page contained in the last entry of the SFR stack.<br>SFR page context is retained upon interrupts/return from interrupts in a 3 byte SFR Page Stack: SFRPAGE is the first entry, SFRNEXT is the second, and SFRLAST is the third entry. The SFR stack bytes may be used alter the context in the SFR Page Stack, and will not cause the stack to “push” or “pop”. Only interrupts and return from interrupts cause pushes and pops of the SFR Page Stack. |



Downloaded from Arrow.com.

**Table 12.1. Special Function Register (SFR) Memory Map for Pages 0x0 and 0xF**

| Address | Page | 0(8) | 1(9) | 2(A) | 3(B) | 4(C) | 5(D) | 6(E) | 7(F) |
|---|---|---|---|---|---|---|---|---|---|
| F8 | 0F | SPI0CN | PCA0L<br>SN0 | PCA0H<br>SN1 | PCA0CPL0<br>SN2 | PCA0CPH0<br>SN3 | PCACPL4 | PCACPH4 | VDM0CN |
| F0 | 0F | B<br>(All Pages) | P0MAT<br>P0MDIN | P0MASK<br>P1MDIN | P1MAT<br>P2MDIN | P1MASK<br>P3MDIN | | EIP1<br>EIP1 | EIP2<br>EIP2 |
| E8 | 0F | ADC0CN | PCA0CPL1 | PCA0CPH1 | PCA0CPL2 | PCA0CPH2 | PCA0CPL3 | PCA0CPL3 | RSTSRC |
| E0 | 0F | ACC<br>(All Pages) | XBR0 | XBR1 | CCH0CN | IT01CF | | EIE1<br>(All Pages) | EIE2<br>(All Pages) |
| D8 | 0F | PCA0CN | PCA0MD<br>PCA0PWM | PCA0CPM0 | PCA0CPM1 | PCA0CPM2 | PCA0CPM3 | PCA0CPM4 | PCA0CPM5 |
| D0 | 0F | PSW<br>(All Pages) | REF0CN | LIN0DATA | LIN0ADDR | P0SKIP | P1SKIP | P2SKIP | P3SKIP |
| C8 | 0F | TMR2CN | REG0CN<br>LIN0CF | TMR2RLL | TMR2RLH | TMR2L | TMR2H | PCA0CPL5 | PCA0CPH5 |
| C0 | 0F | SMB0CN | SMB0CF | SMB0DAT | ADC0GTL | ADC0GTH | ADC0LTL | ADC0LTH | XBR2 |
| B8 | 0F | IP<br>(All Pages) | | ADC0TK | ADC0MX | ADC0CF | ADC0L | ADC0H | |
| B0 | 0F | P3<br>(All Pages) | P2MAT | P2MASK | | | | FLSCL<br>(All Pages) | FLKEY<br>(All Pages) |
| A8 | 0F | IE<br>(All Pages) | SMOD0 | EMI0CN | SBCON0 | SBRLL0 | SBRLH0 | P3MAT<br>P3MDOUT | P3MASK |
| A0 | 0F | P2<br>(All Pages) | SPI0CFG<br>OSCICN | SPI0CKR<br>OSCICRS | SPI0DAT | P0MDOUT | P1MDOUT | P2MDOUT | SFRPAGE<br>(All Pages) |
| 98 | 0F | SCON0 | SBUF0 | CPT0CN | CPT0MD | CPT0MX | CPT1CN | CPT1MD<br>OSCIFIN | CPT1MX<br>OSCXCN |
| 90 | 0F | P1<br>(All Pages) | TMR3CN | TMR3RLL | TMR3RLH | TMR3L | TMR3H | | CLKMUL |
| 88 | 0F | TCON<br>(All Pages) | TMOD<br>(All Pages) | TL0<br>(All Pages) | TL1<br>(All Pages) | TH0<br>(All Pages) | TH1<br>(All Pages) | CKCON<br>(All Pages) | PSCTL<br>CLKSEL |
| 80 | 0F | P0<br>(All Pages) | SP<br>(All Pages) | DPL<br>(All Pages) | DPH<br>(All Pages) | SFR0CN | SFRNEXT<br>(All Pages) | SFRLAST<br>(All Pages) | PCON<br>(All Pages) |
| | | 0(8) | 1(9) | 2(A) | 3(B) | 4(C) | 5(D) | 6(E) | 7(F) |

(bit addressable)



Downloaded from Arrow.com.

### Table 12.2. Special Function Registers

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

| Register | Address | Description | Page |
|---|---|---|---|
| **ACC** | 0xE0 | Accumulator | 82 |
| **ADC0CF** | 0xBC | ADC0 Configuration | 40 |
| **ADC0CN** | 0xE8 | ADC0 Control | 42 |
| **ADC0GTH** | 0xC4 | ADC0 Greater-Than Compare High | 44 |
| **ADC0GTL** | 0xC3 | ADC0 Greater-Than Compare Low | 44 |
| **ADC0H** | 0xBE | ADC0 High | 41 |
| **ADC0L** | 0xBD | ADC0 Low | 41 |
| **ADC0LTH** | 0xC6 | ADC0 Less-Than Compare Word High | 45 |
| **ADC0LTL** | 0xC5 | ADC0 Less-Than Compare Word Low | 45 |
| **ADC0MX** | 0xBB | ADC0 Mux Configuration | 59 |
| **ADC0TK** | 0xBA | ADC0 Tracking Mode Select | 43 |
| **B** | 0xF0 | B Register | 82 |
| **CCH0CN** | 0xE3 | Cache Control | 125 |
| **CKCON** | 0x8E | Clock Control | 228 |
| **CLKMUL** | 0x97 | Clock Multiplier | 141 |
| **CLKSEL** | 0x8F | Clock Select | 136 |
| **CPT0CN** | 0x9A | Comparator0 Control | 65 |
| **CPT0MD** | 0x9B | Comparator0 Mode Selection | 66 |
| **CPT0MX** | 0x9C | Comparator0 MUX Selection | 70 |
| **CPT1CN** | 0x9D | Comparator1 Control | 65 |
| **CPT1MD** | 0x9E | Comparator1 Mode Selection | 66 |
| **CPT1MX** | 0x9F | Comparator1 MUX Selection | 70 |
| **DPH** | 0x83 | Data Pointer High | 81 |
| **DPL** | 0x82 | Data Pointer Low | 81 |
| **EIE1** | 0xE6 | Extended Interrupt Enable 1 | 111 |
| **EIE2** | 0xE7 | Extended Interrupt Enable 2 | 111 |
| **EIP1** | 0xF6 | Extended Interrupt Priority 1 | 112 |
| **EIP2** | 0xF7 | Extended Interrupt Priority 2 | 113 |
| **EMI0CN** | 0xAA | External Memory Interface Control | 88 |
| **FLKEY** | 0xB7 | Flash Lock and Key | 123 |
| **FLSCL** | 0xB6 | Flash Scale | 124 |
| **IE** | 0xA8 | Interrupt Enable | 109 |
| **IP** | 0xB8 | Interrupt Priority | 110 |
| **IT01CF** | 0xE4 | INT0/INT1 Configuration | 116 |
| **LIN0ADR** | 0xD3 | LIN0 Address | 177 |



Downloaded from Arrow.com.

**Table 12.2. Special Function Registers (Continued)**

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

| Register | Address | Description | Page |
|---|---|---|---|
| **LIN0CF** | 0xC9 | LIN0 Configuration | 177 |
| **LIN0DAT** | 0xD2 | LIN0 Data | 178 |
| **OSCICN** | 0xA1 | Internal Oscillator Control | 138 |
| **OSCICRS** | 0xA2 | Internal Oscillator Coarse Control | 139 |
| **OSCIFIN** | 0x9E | Internal Oscillator Fine Calibration | 139 |
| **OSCXCN** | 0x9F | External Oscillator Control | 143 |
| **P0** | 0x80 | Port 0 Latch | 161 |
| **P0MASK** | 0xF2 | Port 0 Mask Configuration | 157 |
| **P0MAT** | 0xF1 | Port 0 Match Configuration | 157 |
| **P0MDIN** | 0xF1 | Port 0 Input Mode Configuration | 162 |
| **P0MDOUT** | 0xA4 | Port 0 Output Mode Configuration | 162 |
| **P0SKIP** | 0xD4 | Port 0 Skip | 163 |
| **P1** | 0x90 | Port 1 Latch | 163 |
| **P1MASK** | 0xF4 | Port 1 Mask Configuration | 158 |
| **P1MAT** | 0xF3 | Port 1 Match Configuration | 158 |
| **P1MDIN** | 0xF2 | Port 1 Input Mode Configuration | 164 |
| **P1MDOUT** | 0xA5 | Port 1 Output Mode Configuration | 164 |
| **P1SKIP** | 0xD5 | Port 1 Skip | 165 |
| **P2** | 0xA0 | Port 2 Latch | 165 |
| **P2MASK** | 0xB2 | Port 2 Mask Configuration | 159 |
| **P2MAT** | 0xB1 | Port 2 Match Configuration | 159 |
| **P2MDIN** | 0xF3 | Port 2 Input Mode Configuration | 166 |
| **P2MDOUT** | 0xA6 | Port 2 Output Mode Configuration | 166 |
| **P2SKIP** | 0xD6 | Port 2 Skip | 167 |
| **P3** | 0xB0 | Port 3 Latch | 167 |
| **P3MASK** | 0xAF | Port 3 Mask Configuration | 160 |
| **P3MAT** | 0xAE | Port 3 Match Configuration | 160 |
| **P3MDIN** | 0xF4 | Port 3 Input Mode Configuration | 168 |
| **P3MDOUT** | 0xAE | Port 3 Output Mode Configuration | 168 |
| **P3SKIP** | 0xD7 | Port 3 Skip | 169 |
| **PCA0CN** | 0xD8 | PCA Control | 263 |
| **PCA0CPH0** | 0xFC | PCA Capture 0 High | 268 |
| **PCA0CPH1** | 0xEA | PCA Capture 1 High | 268 |
| **PCA0CPH2** | 0xEC | PCA Capture 2 High | 268 |
| **PCA0CPH3** | 0xEE | PCA Capture 3 High | 268 |
| **PCA0CPH4** | 0xFE | PCA Capture 4 High | 268 |



Downloaded from Arrow.com.

**Table 12.2. Special Function Registers (Continued)**

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

| Register | Address | Description | Page |
|---|---|---|---|
| **PCA0CPH5** | 0xCF | PCA Capture 5 High | 268 |
| **PCA0CPL0** | 0xFB | PCA Capture 0 Low | 268 |
| **PCA0CPL1** | 0xE9 | PCA Capture 1 Low | 268 |
| **PCA0CPL2** | 0xEB | PCA Capture 2 Low | 268 |
| **PCA0CPL3** | 0xED | PCA Capture 3 Low | 268 |
| **PCA0CPL4** | 0xFD | PCA Capture 4 Low | 268 |
| **PCA0CPL5** | 0xCE | PCA Capture 5 Low | 268 |
| **PCA0CPM0** | 0xDA | PCA Module 0 Mode Register | 266 |
| **PCA0CPM1** | 0xDB | PCA Module 1 Mode Register | 266 |
| **PCA0CPM2** | 0xDC | PCA Module 2 Mode Register | 266 |
| **PCA0CPM3** | 0xDD | PCA Module 3 Mode Register | 266 |
| **PCA0CPM4** | 0xDE | PCA Module 4 Mode Register | 266 |
| **PCA0CPM5** | 0xDF | PCA Module 5 Mode Register | 266 |
| **PCA0H** | 0xFA | PCA Counter High | 267 |
| **PCA0L** | 0xF9 | PCA Counter Low | 267 |
| **PCA0MD** | 0xD9 | PCA Mode | 264 |
| **PCA0PWM** | 0xD9 | PCA PWM Configuration | 265 |
| **PCON** | 0x87 | Power Control | 128 |
| **PSCTL** | 0x8F | Program Store R/W Control | 122 |
| **PSW** | 0xD0 | Program Status Word | 83 |
| **REF0CN** | 0xD1 | Voltage Reference Control | 62 |
| **REG0CN** | 0xC9 | Voltage Regulator Control | 73 |
| **RSTSRC** | 0xEF | Reset Source Configuration/Status | 134 |
| **SBCON0** | 0xAB | UART0 Baud Rate Generator Control | 212 |
| **SBRLH0** | 0xAD | UART0 Baud Rate Reload High Byte | 213 |
| **SBRLL0** | 0xAC | UART0 Baud Rate Reload Low Byte | 213 |
| **SBUF0** | 0x99 | UART0 Data Buffer | 212 |
| **SCON0** | 0x98 | UART0 Control | 210 |
| **SFR0CN** | 0x84 | SFR Page Control | 96 |
| **SFRLAST** | 0x86 | SFR Stack Last Page | 99 |
| **SFRNEXT** | 0x85 | SFR Stack Next Page | 98 |
| **SFRPAGE** | 0xA7 | SFR Page Select | 97 |
| **SMB0CF** | 0xC1 | SMBus0 Configuration | 193 |
| **SMB0CN** | 0xC0 | SMBus0 Control | 195 |
| **SMB0DAT** | 0xC2 | SMBus0 Data | 197 |
| **SMOD0** | 0xA9 | UART0 Mode | 211 |



Downloaded from Arrow.com.

**Table 12.2. Special Function Registers (Continued)**

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

| Register | Address | Description | Page |
|---|---|---|---|
| **SN0** | 0xF9 | Serial Number 0 | 84 |
| **SN1** | 0xFA | Serial Number 1 | 84 |
| **SN2** | 0xFB | Serial Number 2 | 84 |
| **SN3** | 0xFC | Serial Number 3 | 84 |
| **SP** | 0x81 | Stack Pointer | 82 |
| **SPI0CFG** | 0xA1 | SPI0 Configuration | 221 |
| **SPI0CKR** | 0xA2 | SPI0 Clock Rate Control | 223 |
| **SPI0CN** | 0xF8 | SPI0 Control | 222 |
| **SPI0DAT** | 0xA3 | SPI0 Data | 223 |
| **TCON** | 0x88 | Timer/Counter Control | 233 |
| **TH0** | 0x8C | Timer/Counter 0 High | 236 |
| **TH1** | 0x8D | Timer/Counter 1 High | 236 |
| **TL0** | 0x8A | Timer/Counter 0 Low | 235 |
| **TL1** | 0x8B | Timer/Counter 1 Low | 235 |
| **TMOD** | 0x89 | Timer/Counter Mode | 234 |
| **TMR2CN** | 0xC8 | Timer/Counter 2 Control | 240 |
| **TMR2H** | 0xCD | Timer/Counter 2 High | 242 |
| **TMR2L** | 0xCC | Timer/Counter 2 Low | 242 |
| **TMR2RLH** | 0xCB | Timer/Counter 2 Reload High | 241 |
| **TMR2RLL** | 0xCA | Timer/Counter 2 Reload Low | 241 |
| **TMR3CN** | 0x91 | Timer/Counter 3 Control | 246 |
| **TMR3H** | 0x95 | Timer/Counter 3 High | 248 |
| **TMR3L** | 0x94 | Timer/Counter 3 Low | 248 |
| **TMR3RLH** | 0x93 | Timer/Counter 3 Reload High | 247 |
| **TMR3RLL** | 0x92 | Timer/Counter 3 Reload Low | 247 |
| **VDM0CN** | 0xFF | $V_{DD}$ Monitor Control | 132 |
| **XBR0** | 0xE1 | Port I/O Crossbar Control 0 | 154 |
| **XBR1** | 0xE2 | Port I/O Crossbar Control 1 | 155 |
| **XBR2** | 0xC7 | Port I/O Crossbar Control 2 | 156 |



Downloaded from Arrow.com.

# 13. Interrupts

The C8051F54x devices include 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 inputs 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, or EIE2). 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.

## 13.1. MCU Interrupt Sources and Vectors

The C8051F54x MCUs support 17 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 13.1. 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).



Downloaded from Arrow.com.

### 13.1.1. 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 (IE, EIP1, or EIP2) 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 13.1.

### 13.1.2. 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.



Downloaded from Arrow.com.

**Table 13.1. 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) |
| ADC0 Window Compare | 0x0043 | 8 | AD0WINT (ADC0CN.3) | Y | N | EWADC0 (EIE1.1) | PWADC0 (EIP1.1) |
| ADC0 Conversion Complete | 0x004B | 9 | AD0INT (ADC0CN.5) | Y | N | EADC0 (EIE1.2) | PADC0 (EIP1.2) |
| Programmable Counter Array | 0x0053 | 10 | CF (PCA0CN.7)<br>CCFn (PCA0CN.n)<br>COVF (PCA0PWM.6) | Y | N | EPCA0 (EIE1.3) | PPCA0 (EIP1.3) |
| Comparator0 | 0x005B | 11 | CP0FIF (CPT0CN.4)<br>CP0RIF (CPT0CN.5) | N | N | ECP0 (EIE1.4) | PCP0 (EIP1.4) |
| Comparator1 | 0x0063 | 12 | CP1FIF (CPT1CN.4)<br>CP1RIF (CPT1CN.5) | N | N | ECP1 (EIE1.5) | PCP1 (EIP1.5) |
| Timer 3 Overflow | 0x006B | 13 | TF3H (TMR3CN.7)<br>TF3L (TMR3CN.6) | N | N | ET3 (EIE1.6) | PT3 (EIP1.6) |
| LIN0 | 0x0073 | 14 | LIN0INT (LINST.3) | N | N* | ELIN0 (EIE1.7) | PLIN0 (EIP1.7) |
| Voltage Regulator Dropout | 0x007B | 15 | N/A | N/A | N/A | EREG0 (EIE2.0) | PREG0 (EIP2.0) |
| Port Match | 0x008B | 17 | None | N/A | N/A | EMAT (EIE2.2) | PMAT (EIP2.2) |

***Note:** The LIN0INT bit is cleared by setting RSTINT (LINCTRL.3)



Downloaded from Arrow.com.

## 13.2. 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).



Downloaded from Arrow.com.

## SFR Definition 13.1. 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; SFR Page = All Pages

| 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 $\overline{\text{INT1}}$ 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 $\overline{\text{INT0}}$ input. |



Downloaded from Arrow.com.

## SFR Definition 13.2. 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; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7 | Unused | Read = 1b, 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. |



Downloaded from Arrow.com.

## SFR Definition 13.3. EIE1: Extended Interrupt Enable 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ELIN0 | ET3 | ECP1 | ECP0 | EPCA0 | EADC0 | EWADC0 | 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; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7 | ELIN0 | **Enable LIN0 Interrupt.**<br>This bit sets the masking of the LIN0 interrupt.<br>0: Disable LIN0 interrupts.<br>1: Enable interrupt requests generated by the LIN0INT flag. |
| 6 | 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. |
| 5 | ECP1 | **Enable Comparator1 (CP1) Interrupt.**<br>This bit sets the masking of the CP1 interrupt.<br>0: Disable CP1 interrupts.<br>1: Enable interrupt requests generated by the CP1RIF or CP1FIF flags. |
| 4 | 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. |
| 3 | 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. |
| 2 | 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. |
| 1 | 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). |
| 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. |



Downloaded from Arrow.com.

## SFR Definition 13.4. EIP1: Extended Interrupt Priority 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PLIN0 | PT3 | PCP1 | PCP0 | PPCA0 | PADC0 | PWADC0 | 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; SFR Page = 0x00 and 0x0F

| Bit | Name | Function |
|---|---|---|
| 7 | PLIN0 | **LIN0 Interrupt Priority Control.**<br>This bit sets the priority of the LIN0 interrupt.<br>0: LIN0 interrupts set to low priority level.<br>1: LIN0 interrupts set to high priority level. |
| 6 | 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. |
| 5 | PCP1 | **Comparator0 (CP1) Interrupt Priority Control.**<br>This bit sets the priority of the CP1 interrupt.<br>0: CP1 interrupt set to low priority level.<br>1: CP1 interrupt set to high priority level. |
| 4 | 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. |
| 3 | 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. |
| 2 | 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. |
| 1 | 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. |
| 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. |



Downloaded from Arrow.com.

## SFR Definition 13.5. EIE2: Extended Interrupt Enable 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | EMAT | | EREG0 |
| Type | R | R | R | R | R | R/W | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xE7; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:3 | Unused | Read = 00000b; Write = Don't Care. |
| 2 | EMAT | **Enable Port Match Interrupt.**<br>This bit sets the masking of the Port Match interrupt.<br>0: Disable all Port Match interrupts.<br>1: Enable interrupt requests generated by a Port Match |
| 1 | Unused | Read = 0b; Write = Don't Care. |
| 0 | EREG0 | **Enable Voltage Regulator Dropout Interrupt.**<br>This bit sets the masking of the Voltage Regulator Dropout interrupt.<br>0: Disable the Voltage Regulator Dropout interrupt.<br>1: Enable the Voltage Regulator Dropout interrupt. |



Downloaded from Arrow.com.

## SFR Definition 13.6. EIP2: Extended Interrupt Priority Enabled 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | PMAT | | PREG0 |
| Type | R | R | R | R | R | R/W | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xF7; SFR Page = 0x00 and 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:3 | Unused | Read = 00000b; Write = Don't Care. |
| 2 | PMAT | **Port Match Interrupt Priority Control.**<br>This bit sets the priority of the Port Match interrupt.<br>0: Port Match interrupt set to low priority level.<br>1: Port Match interrupt set to high priority level. |
| 1 | Unused | Read = 0b; Write = Don't Care. |
| 0 | PREG0 | **Voltage Regulator Dropout Interrupt Priority Control.**<br>This bit sets the priority of the Voltage Regulator Dropout interrupt.<br>0: Voltage Regulator Dropout interrupt set to low priority level.<br>1: Voltage Regulator Dropout interrupt set to high priority level. |



Downloaded from Arrow.com.

## 13.3. External Interrupts $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$

The $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ external interrupt sources are configurable as active high or low, edge or level sensitive. The IN0PL ($\overline{\text{INT0}}$ Polarity) and IN1PL ($\overline{\text{INT1}}$ Polarity) bits in the IT01CF register select active high or active low; the IT0 and IT1 bits in TCON (Section “23.1. Timer 0 and Timer 1” on page 229) select level or edge sensitive. The table below lists the possible configurations.

| IT0 | IN0PL | $\overline{\text{INT0}}$ 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 | $\overline{\text{INT1}}$ 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 |

$\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are assigned to Port pins as defined in the IT01CF register (see SFR Definition 13.7). Note that $\overline{\text{INT0}}$ and $\overline{\text{INT0}}$ Port pin assignments are independent of any Crossbar assignments. $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ 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 $\overline{\text{INT0}}$ and/or $\overline{\text{INT1}}$, configure the Crossbar to skip the selected pin(s). This is accomplished by setting the associated bit in register XBR0 (see Section “18.3. Priority Crossbar Decoder” on page 150 for complete details on configuring the Crossbar).

IE0 (TCON.1) and IE1 (TCON.3) serve as the interrupt-pending flags for the $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ external interrupts, respectively. If an $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ 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.



Downloaded from Arrow.com.

## SFR Definition 13.7. 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 | 0 |

SFR Address = 0xE4; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7 | IN1PL | **$\overline{\text{INT1}}$ Polarity.**<br>0: $\overline{\text{INT1}}$ input is active low.<br>1: $\overline{\text{INT1}}$ input is active high. |
| 6:4 | IN1SL[2:0] | **$\overline{\text{INT1}}$ Port Pin Selection Bits.**<br>These bits select which Port pin is assigned to $\overline{\text{INT1}}$. Note that this pin assignment is independent of the Crossbar; $\overline{\text{INT1}}$ 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 P1.0<br>001: Select P1.1<br>010: Select P1.2<br>011: Select P1.3<br>100: Select P1.4<br>101: Select P1.5<br>110: Select P1.6<br>111: Select P1.7 |
| 3 | IN0PL | **$\overline{\text{INT0}}$ Polarity.**<br>0: $\overline{\text{INT0}}$ input is active low.<br>1: $\overline{\text{INT0}}$ input is active high. |
| 2:0 | IN0SL[2:0] | **$\overline{\text{INT0}}$ Port Pin Selection Bits.**<br>These bits select which Port pin is assigned to $\overline{\text{INT0}}$. Note that this pin assignment is independent of the Crossbar; $\overline{\text{INT0}}$ 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 P1.0<br>001: Select P1.1<br>010: Select P1.2<br>011: Select P1.3<br>100: Select P1.4<br>101: Select P1.5<br>110: Select P1.6<br>111: Select P1.7 |



Downloaded from Arrow.com.

# 14. 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 Table 6.5 for complete Flash memory electrical characteristics.

## 14.1. 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 “25. C2 Interface” on page 269.

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 14.4 for more details. Before performing any Flash write or erase procedure, set the FLEWT bit in Flash Scale register (FLSCL) to ‘1’. Also, note that 8-bit MOVX instructions cannot be used to erase or write to Flash memory at addresses higher than 0x00FF

### 14.1.1. 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 14.2.

### 14.1.2. 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 doing the following: (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 the PSEE 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.



Downloaded from Arrow.com.

### 14.1.3. 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 14.1.2.
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.

### 14.1.4. Flash Write Optimization

The Flash write procedure includes a block write option to optimize the time to perform consecutive byte writes. When block write is enabled by setting the CHBLKW bit (CCH0CN.0), writes to two consecutive bytes in Flash require the same amount of time as a single byte write. This is performed by caching the first byte that is written to Flash and then committing both bytes to Flash when the second byte is written. When block writes are enabled, if the second write does not occur, the first data byte written is not actually written to Flash. Flash bytes with block write enabled 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 14.1.2.
3. Set the CHBLKW bit (register CCH0CN).
4. Set the PSWE bit (register PSCTL).
5. Clear the PSEE bit (register PSCTL).
6. Write the first key code to FLKEY: 0xA5.
7. Write the second key code to FLKEY: 0xF1.
8. Using the MOVX instruction, write the first data byte to the desired location within the 512-byte sector.
9. Write the first key code to FLKEY: 0xA5.
10. Write the second key code to FLKEY: 0xF1.
11. Using the MOVX instruction, write the second data byte to the desired location within the 512-byte sector. The location of the second byte must be the next higher address from the first data byte.
12. Clear the PSWE bit.
13. Clear the CHBLKW bit.



Downloaded from Arrow.com.

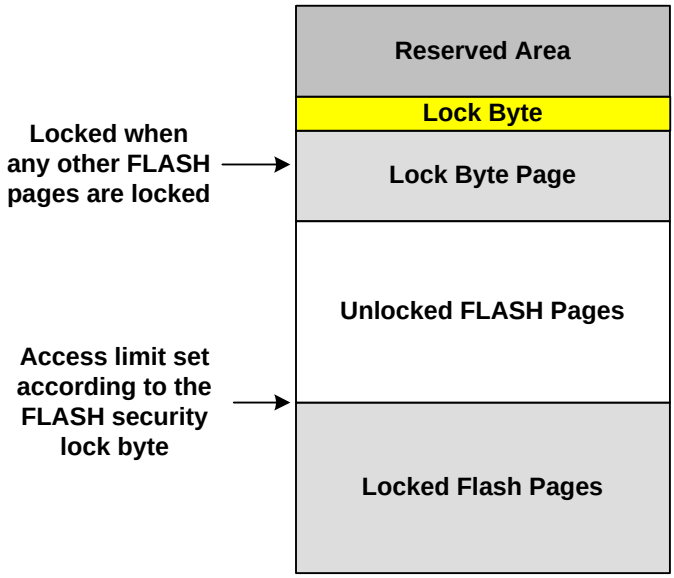
## 14.2. Non-volatile Data Storage

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.

## 14.3. 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 at the last byte of Flash user space offers protection of the Flash program memory from access (reads, writes, or erases) by unprotected code or the C2 interface. 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 ones 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).** See example in Figure 14.1.




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

**Figure 14.1. Flash Program Memory Map**



Downloaded from Arrow.com.

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 14.1 summarizes the Flash security features of the C8051F54x devices.

**Table 14.1. 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 1s to 0s in the Lock Byte) | Not Permitted | Flash Error Reset | Flash Error Reset |
| Unlock individual pages (change 0s to 1s in the Lock Byte) | Not Permitted | Flash Error Reset | Flash Error Reset |
| Read, Write or Erase Reserved Area | Not Permitted | See note | See note |

**Note:** Flash Reads will return indeterminate data. Flash Writes and Erases are ignored.

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.



Downloaded from Arrow.com.

## 14.4. 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.

### 14.4.1. $V_{DD}$ 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. 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.
3. 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.
4. 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.
5. 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.

### 14.4.2. PSWE Maintenance

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.



Downloaded from Arrow.com.

### 14.4.3. System Clock

1. 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.
2. 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 14.1. 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; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:2 | 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. |



Downloaded from Arrow.com.

## SFR Definition 14.2. 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; SFR Page = All Pages

| 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. |



Downloaded from Arrow.com.

## SFR Definition 14.3. FLSCL: Flash Scale

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | Reserved | Reserved | FLRT | Reserved | Reserved | FLEWT | Reserved |
| 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 = 0xB6; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:5 | Reserved | Must Write 000b. |
| 4 | FLRT | **Flash Read Time Control.**<br>This bit should be programmed to the smallest allowed value, according to the system clock speed.<br>0: SYSCLK ≤ 25 MHz (Flash read strobe is one system clock).<br>1: SYSCLK > 25 MHz (Flash read strobe is two system clocks). |
| 3:2 | Reserved | Must Write 00b. |
| 1 | FLEWT | **Flash Erase Write Time Control.**<br>This bit should be set to 1b before Writing or Erasing Flash.<br>0: Short Flash Erase / Write Timing.<br>1: Extended Flash Erase / Write Timing. |
| 0 | Reserved | Must Write 0b. |



Downloaded from Arrow.com.

## SFR Definition 14.4. CCH0CN: Cache Control

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

SFR Address = 0xE3; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:6 | Reserved | Must Write 00b |
| 5 | CHPFEN | **Cache Prefect Enable Bit.**<br>0: Prefetch engine is disabled.<br>1: Prefetch engine is enabled. |
| 4:1 | Reserved | Must Write 0000b. |
| 0 | CHBLKW | **Block Write Enable Bit.**<br>This bit allows block writes to Flash memory from firmware.<br>0: Each byte of a software Flash write is written individually.<br>1: Flash bytes are written in groups of two. |

## SFR Definition 14.5. ONESHOT: Flash Oneshot Period

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

SFR Address = 0xBE; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:4 | Unused | Read = 0000b. Write = don't care. |
| 3:0 | PERIOD[3:0] | **Oneshot Period Control Bits.**<br>These bits limit the internal Flash read strobe width as follows. When the Flash read strobe is de-asserted, the Flash memory enters a low-power state for the remainder of the system clock cycle.<br>$FLASH_{RDMAX} = 5ns + (PERIOD \times 5ns)$ |



Downloaded from Arrow.com.

# 15. Power Management Modes

The C8051F54x 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 Match or Comparator low output. 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 15.1 describes the Power Control Register (PCON) used to control the C8051F54x devices' Stop and Idle power management modes. Suspend mode is controlled by the SUSPEND bit in the OSCICN register (SFR Definition 17.2).

Although the C8051F54x 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.

## 15.1. 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 "16.6. PCA Watchdog Timer Reset" on page 133 for more information on the use and configuration of the WDT.



Downloaded from Arrow.com.

## 15.2. 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.

## 15.3. 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.

Suspend mode can be terminated by three types of events, a port match (described in Section “18.5. Port Match” on page 157), a Comparator low output (if enabled), or a device reset event. 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 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.

Note: When entering Suspend mode, firmware must set the ZTCEN bit in REF0CN (SFR Definition 7.1).



Downloaded from Arrow.com.

## SFR Definition 15.1. 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; SFR Page = All Pages

| 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 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.) |



Downloaded from Arrow.com.

# 16. Reset Sources

Reset circuitry allows the controller to be easily placed in a predefined default condition. On entry to this reset state, the following 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.

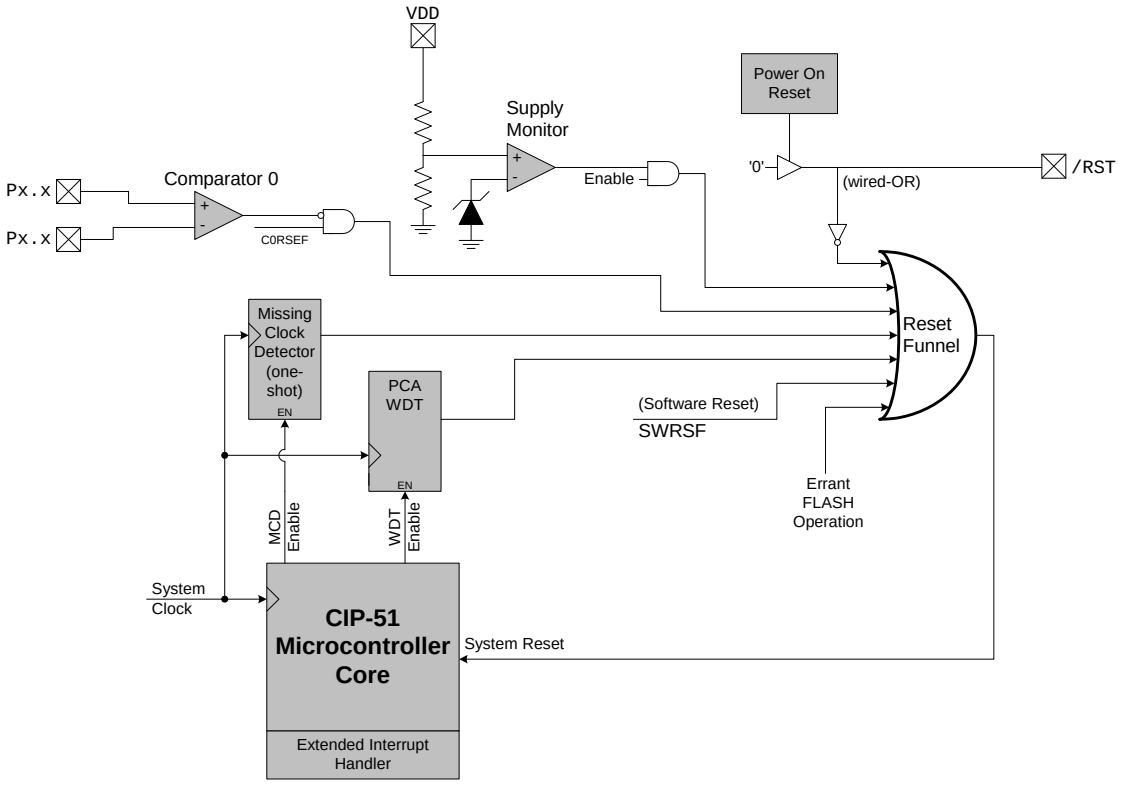



**Figure 16.1. Reset Sources**



Downloaded from Arrow.com.

## 16.1. 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 16.2. plots the power-on and $V_{DD}$ monitor reset timing.

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.

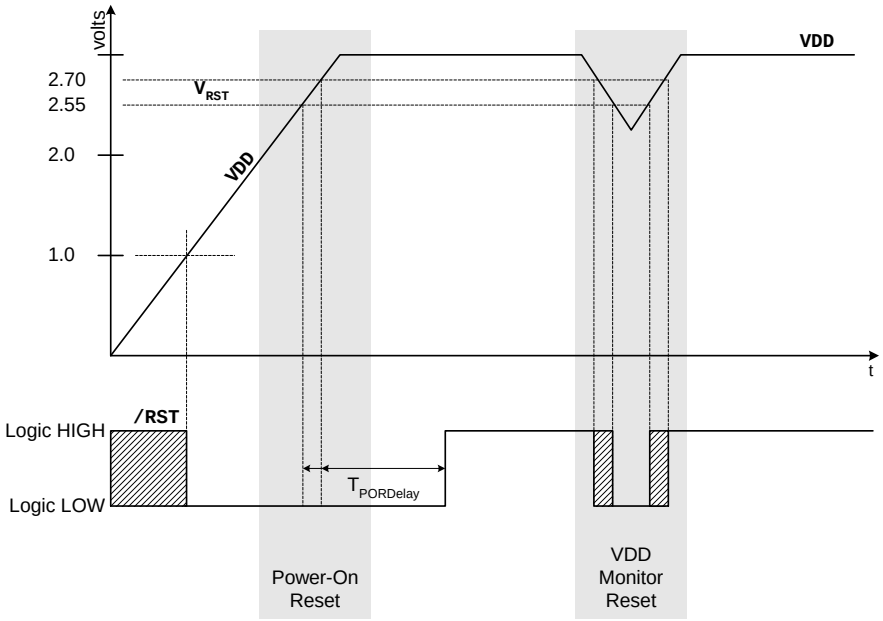



**Figure 16.2. Power-On and $V_{DD}$ Monitor Reset Timing**

## 16.2. 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 $\overline{\text{RST}}$ pin low and hold the CIP-51 in a reset state (see Figure 16.2). 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. **To protect the integrity of Flash contents, the $V_{DD}$ monitor must be enabled to the higher setting (VDMLVL = 1) and selected as a reset source if software contains routines which erase or write Flash memory. If the $V_{DD}$ monitor is not enabled and set to the high level, any erase or write performed on Flash memory will cause a Flash Error device reset.**



Downloaded from Arrow.com.

**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 as follows:

1. Enable the $V_{DD}$ monitor (VDMEN bit in VDM0CN = 1).
2. If necessary, wait for the $V_{DD}$ monitor to stabilize (see Table 6.4 for the $V_{DD}$ Monitor turn-on time). **Note: This delay should be omitted if software contains routines that erase or write Flash memory.**
3. Select the $V_{DD}$ monitor as a reset source (PORSF bit in RSTSRC = 1).

See Figure 16.2 for $V_{DD}$ monitor timing; note that the power-on-reset delay is not incurred after a $V_{DD}$ monitor reset. See Table 6.4 for complete electrical characteristics of the $V_{DD}$ monitor.

**Note:** The output of the internal voltage regulator is calibrated by the MCU immediately after any reset event. The output of the un-calibrated internal regulator could be below the high threshold setting of the VDD Monitor. If this is the case *and* the $V_{DD}$ Monitor is set to the high threshold setting *and* if the MCU receives a non-power on reset (POR), the MCU will remain in reset until a POR occurs (i.e., $V_{DD}$ Monitor will keep the device in reset). A POR will force the $V_{DD}$ Monitor to the low threshold setting which is guaranteed to be below the un-calibrated output of the internal regulator. The device will then exit reset and resume normal operation. It is for this reason Silicon Labs strongly recommends that the $V_{DD}$ Monitor is always left in the low threshold setting (i.e., default value upon POR).

When programming the Flash in-system, the $V_{DD}$ Monitor must be set to the high threshold setting. For the highest system reliability, the time the $V_{DD}$ Monitor is set to the high threshold setting should be minimized (e.g., setting the $V_{DD}$ Monitor to the high threshold setting just before the Flash write operation and then changing it back to the low threshold setting immediately after the Flash write operation).



Downloaded from Arrow.com.

### SFR Definition 16.1. VDM0CN: $V_{DD}$ Monitor Control

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

SFR Address = 0xFF; SFR Page = 0x00

| 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 16.2). 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. See Table 6.4 for the minimum $V_{DD}$ Monitor turn-on time.<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 | VDMLVL | **$V_{DD}$ Monitor Level Select.**<br>0: $V_{DD}$ Monitor Threshold is set to VRST-LOW<br>1: $V_{DD}$ Monitor Threshold is set to VRST-HIGH. This setting is required for any system includes code that writes to and/or erases Flash. |
| 4:0 | Unused | Read = 00000b; Write = Don't care. |

## 16.3. 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 Table 6.4 for complete $\overline{\text{RST}}$ pin specifications. The PINRSF flag (RSTSRC.0) is set on exit from an external reset.

## 16.4. 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 the time specified in Table 6.4, "Reset Electrical Characteristics," on page 52, 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.



Downloaded from Arrow.com.

## 16.5. 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.

## 16.6. 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 “24.4. Watchdog Timer Mode” on page 260; 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.

## 16.7. 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 in or above the reserved space.
- A Flash read is attempted above user code space. This occurs when a MOVC operation targets an address in or above the reserved space.
- A Program read is attempted above user code space. This occurs when user code attempts to branch to an address in or above the reserved space.
- A Flash read, write or erase attempt is restricted due to a Flash security setting (see Section “14.3. Security Options” on page 119).
- A Flash read, write, or erase is attempted when the VDD Monitor is not enabled to the high threshold and set as a reset source.

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

## 16.8. 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.



Downloaded from Arrow.com.

## SFR Definition 16.2. 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; SFR Page = 0x00

| 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



Downloaded from Arrow.com.

# 17. Oscillators and Clock Selection

C8051F54x devices include a programmable internal high-frequency oscillator, an external oscillator drive circuit, and a clock multiplier. The internal oscillator can be enabled/disabled and calibrated using the OSCICN, OSCICRS, and OSCIFIN registers, as shown in Figure 17.1. The system clock can be sourced by the external oscillator circuit or the internal oscillator. The clock multiplier can produce three possible base outputs which can be scaled by a programmable factor of 1, 2/3, 2/4 (or 1/2), 2/5, 2/6 (or 1/3), or 2/7: Internal Oscillator x 2, External Oscillator x 2, or External Oscillator x 4.

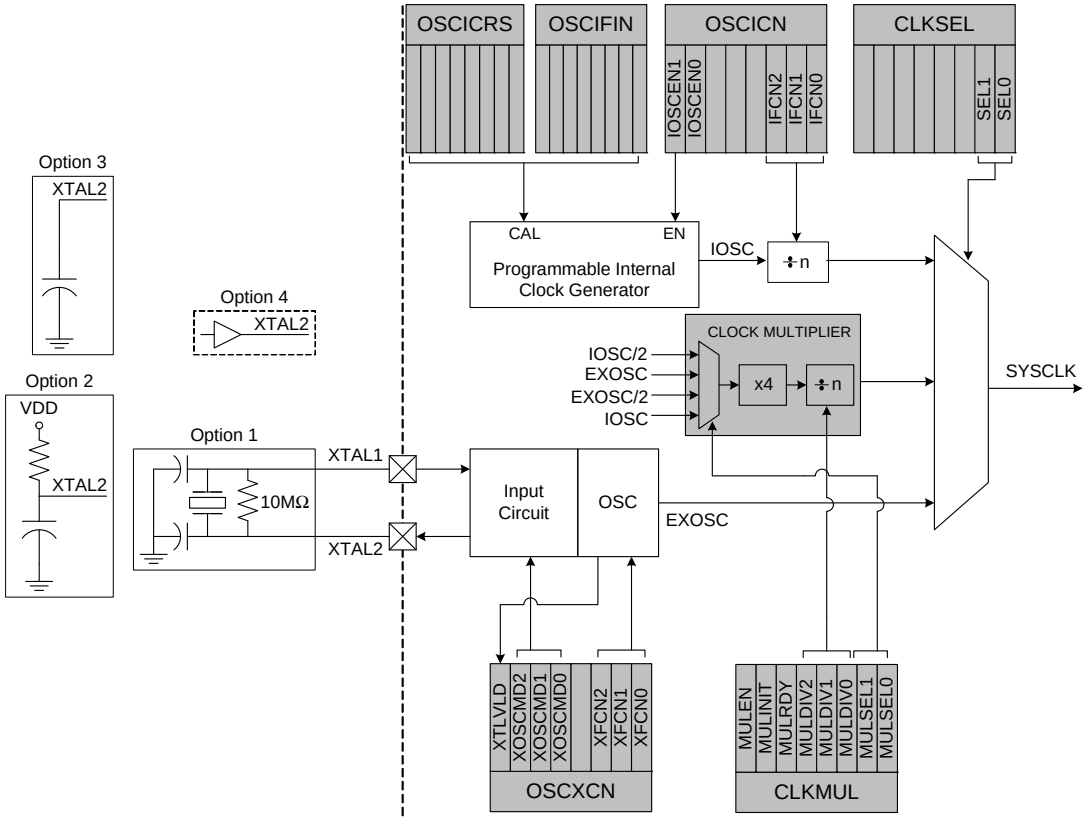



**Figure 17.1. Oscillator Options**

## 17.1. 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 oscillator requires 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.**



Downloaded from Arrow.com.

## SFR Definition 17.1. 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 = 0x8F; SFR Page = 0x0F;

| Bit | Name | Function |
|---|---|---|
| 7:2 | Unused | Read = 000000b; Write = Don't Care |
| 1:0 | CLKSL[1:0] | **System Clock Source Select Bits.**<br>00: SYSCLK derived from the Internal 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 Clock Multiplier.<br>11: reserved. |



Downloaded from Arrow.com.

## 17.2. Programmable Internal Oscillator

All C8051F54x devices include a programmable internal high-frequency oscillator that defaults as the system clock after a system reset. The internal oscillator period can be adjusted via the OSCICRS and OSCIFIN registers defined in SFR Definition 17.3 and SFR Definition 17.4. On C8051F54x devices, OSCICRS and OSCIFIN are factory calibrated to obtain a 24 MHz base frequency. Note that the system clock may be derived from the programmed internal oscillator divided by 1, 2, 4, 8, 16, 32, 64, or 128, as defined by the IFCN bits in register OSCICN. The divide value defaults to 128 following a reset.

### 17.2.1. 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.
- Port 2 Match Event.
- Port 3 Match Event.
- Comparator 0 enabled and output is logic 0.

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.

**Note:** When entering suspend mode, firmware must set the ZTCEN bit in REF0CN (SFR Definition 7.1).



Downloaded from Arrow.com.

## SFR Definition 17.2. OSCICN: Internal Oscillator Control

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

SFR Address = 0xA1; SFR Page = 0x0F;

| Bit | Name | Function |
|---|---|---|
| 7:6 | IOSCEN[1:0] | **Internal Oscillator Enable Bits.**<br>00: Oscillator Disabled.<br>01: Reserved.<br>10: Reserved.<br>11: Oscillator enabled in normal mode and disabled in suspend mode. |
| 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 | IFRDY | **Internal Oscillator Frequency Ready Flag.**<br>0: Internal oscillator is not running at programmed frequency.<br>1: Internal oscillator is running at programmed frequency. |
| 3 | Reserved | Read = 0b; Write = 0b. |
| 2:0 | IFCN[2:0] | **Internal Oscillator Frequency Divider Control Bits.**<br>000: SYSCLK derived from Internal Oscillator divided by 128.<br>001: SYSCLK derived from Internal Oscillator divided by 64.<br>010: SYSCLK derived from Internal Oscillator divided by 32.<br>011: SYSCLK derived from Internal Oscillator divided by 16.<br>100: SYSCLK derived from Internal Oscillator divided by 8.<br>101: SYSCLK derived from Internal Oscillator divided by 4.<br>110: SYSCLK derived from Internal Oscillator divided by 2.<br>111: SYSCLK derived from Internal Oscillator divided by 1. |



Downloaded from Arrow.com.

## SFR Definition 17.3. OSCICRS: Internal Oscillator Coarse Calibration

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

SFR Address = 0xA2; SFR Page = 0x0F;

| Bit | Name | Function |
|---|---|---|
| 7 | Unused | Read = 0; Write = Don't Care |
| 6:0 | OSCICRS[6:0] | **Internal Oscillator Coarse Calibration Bits.**<br>These bits determine the internal oscillator period. When set to 0000000b, the internal oscillator operates at its slowest setting. When set to 1111111b, the internal oscillator operates at its fastest setting. The reset value is factory calibrated to generate an internal oscillator frequency of 24 MHz. |

## SFR Definition 17.4. OSCIFIN: Internal Oscillator Fine Calibration

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

SFR Address = 0x9E; SFR Page = 0x0F;

| Bit | Name | Function |
|---|---|---|
| 7:6 | Unused | Read = 00b; Write = Don't Care |
| 5:0 | OSCIFIN[5:0] | **Internal Oscillator Fine Calibration Bits.**<br>These bits are fine adjustment for the internal oscillator period. The reset value is factory calibrated to generate an internal oscillator frequency of 24 MHz. |



Downloaded from Arrow.com.

Downloaded from Arrow.com.

## 17.3. Clock Multiplier

The Clock Multiplier generates an output clock which is 4 times the input clock frequency scaled by a programmable factor of 1, 2/3, 2/4 (or 1/2), 2/5, 2/6 (or 1/3), or 2/7. The Clock Multiplier's input can be selected from the external oscillator, or the internal or external oscillators divided by 2. This produces three possible base outputs which can be scaled by a programmable factor: Internal Oscillator x 2, External Oscillator x 2, or External Oscillator x 4. See Section 17.1 on page 135 for details on system clock selection.

The Clock Multiplier is configured via the CLKMUL register (SFR Definition 17.5). The procedure for configuring and enabling the Clock Multiplier is as follows:

1. Reset the Multiplier by writing 0x00 to register CLKMUL.
2. Select the Multiplier input source via the MULSEL bits.
3. Select the Multiplier output scaling factor via the MULDIV bits
4. Enable the Multiplier with the MULEN bit (CLKMUL | = 0x80).
5. Delay for >5 µs.
6. Initialize the Multiplier with the MULINIT bit (CLKMUL | = 0xC0).
7. Poll for MULRDY $\geq$ 1.

**Important Note**: When using an external oscillator as the input to the Clock Multiplier, the external source must be enabled and stable before the Multiplier is initialized. See "17.4. External Oscillator Drive Circuit" on page 142 for details on selecting an external oscillator source.

The Clock Multiplier allows faster operation of the CIP-51 core and is intended to generate an output frequency between 25 and 50 MHz. The clock multiplier can also be used with slow input clocks. However, if the clock is below the minimum Clock Multiplier input frequency ($FCM_{min}$), the generated clock will consist of four fast pulses followed by a long delay until the next input clock rising edge. The average frequency of the output is equal to 4x the input, but the instantaneous frequency may be faster. See Figure 17.2 below for more information.

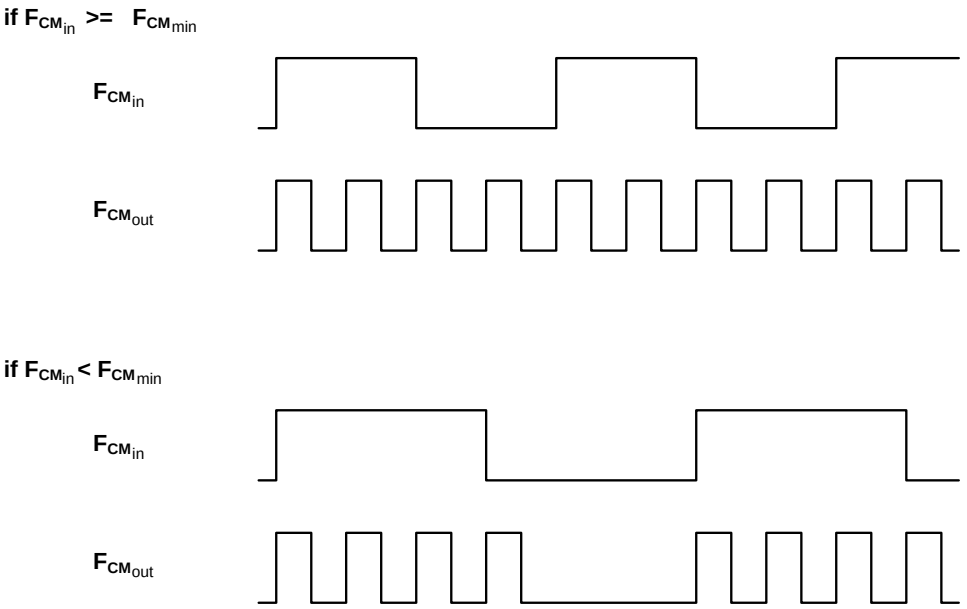



**Figure 17.2. Example Clock Multiplier Output**



Downloaded from Arrow.com.

## SFR Definition 17.5. CLKMUL: Clock Multiplier

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MULEN | MULINIT | MULRDY | MULDIV[2:0] | | | MULSEL[1:0] | |
| Type | R/W | R/W | R | R/W | | | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x97; SFR Page = 0x0F;

| Bit | Name | Function |
|---|---|---|
| 7 | MULEN | **Clock Multiplier Enable.**<br>0: Clock Multiplier disabled.<br>1: Clock Multiplier enabled. |
| 6 | MULINIT | **Clock Multiplier Initialize.**<br>This bit is 0 when the Clock Multiplier is enabled. Once enabled, writing a 1 to this bit will initialize the Clock Multiplier. The MULRDY bit reads 1 when the Clock Multiplier is stabilized. |
| 5 | MULRDY | **Clock Multiplier Ready.**<br>0: Clock Multiplier is not ready.<br>1: Clock Multiplier is ready (PLL is locked). |
| 4:2 | MULDIV[2:0] | **Clock Multiplier Output Scaling Factor.**<br>000: Clock Multiplier Output scaled by a factor of 1.<br>001: Clock Multiplier Output scaled by a factor of 1.<br>010: Clock Multiplier Output scaled by a factor of 1.<br>011: Clock Multiplier Output scaled by a factor of 2/3*.<br>100: Clock Multiplier Output scaled by a factor of 2/4 (1/2).<br>101: Clock Multiplier Output scaled by a factor of 2/5*.<br>110: Clock Multiplier Output scaled by a factor of 2/6 (1/3).<br>111: Clock Multiplier Output scaled by a factor of 2/7*.<br>***Note:** The Clock Multiplier output duty cycle is not 50% for these settings. |
| 1:0 | MULSEL[1:0] | **Clock Multiplier Input Select.**<br>These bits select the clock supplied to the Clock Multiplier |

| MULSEL[1:0] | Selected Input Clock | Clock Multiplier Output for MULDIV[2:0] = 000b |
|---|---|---|
| 00 | Internal Oscillator | Internal Oscillator x 2 |
| 01 | External Oscillator | External Oscillator x 2 |
| 10 | Internal Oscillator | Internal Oscillator x 4 |
| 11 | External Oscillator | External Oscillator x 4 |

**Notes:** The maximum system clock is 50 MHz, and so the Clock Multiplier output should be scaled accordingly.
If Internal Oscillator x 2 or External Oscillator x 2 is selected using the MULSEL bits, MULDIV[2:0] is ignored.



Downloaded from Arrow.com.

## 17.4. 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 17.1. 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 17.1. 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 17.6).

**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 “18.3. Priority Crossbar Decoder” on page 150 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 “18.4. Port I/O Initialization” on page 152 for details on Port input mode selection.



Downloaded from Arrow.com.

## SFR Definition 17.6. 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 = 0x9F; SFR Page = 0x0F;

| 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 = 0b; Write =0b |
| 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 |



Downloaded from Arrow.com.

### 17.4.1. 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 17.1, Option 1. The External Oscillator Frequency Control value (XFCN) should be chosen from the Crystal column of the table in SFR Definition 17.6 (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 high state. This involves enabling the Crossbar and writing 1 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. 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 17.1, 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 17.3.



Downloaded from Arrow.com.

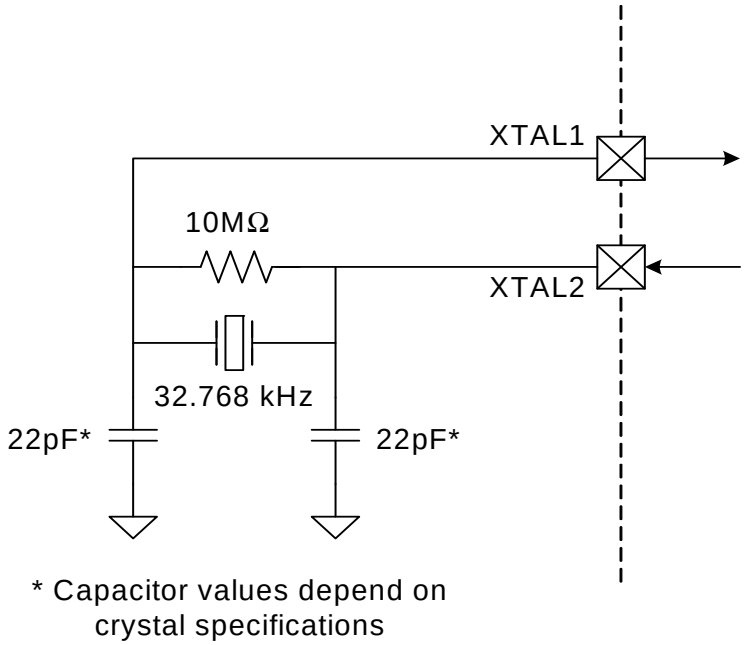



**Figure 17.3. External 32.768 kHz Quartz Crystal Oscillator Connection Diagram**

### 17.4.2. 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 17.1, 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 , where f = the frequency of oscillation in MHz, C = the capacitor value in pF, and R = the pull-up resistor value in kΩ.

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

**Equation 17.1. RC Mode Oscillator Frequency**

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 17.6, the required XFCN setting is 010b.

### 17.4.3. External Capacitor Example

If a capacitor is used as an external oscillator for the MCU, the circuit should be configured as shown in Figure 17.1, 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 17.2, where f = the frequency of oscillation in MHz, C = the capacitor value in pF, and $V_{DD}$ = the MCU power supply in Volts.



Downloaded from Arrow.com.

**Equation 17.2. C Mode Oscillator Frequency**

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

For example: Assume $V_{DD}$ = 2.1 V and f = 75 kHz:

f = KF / (C x VDD)

0.075 MHz = KF / (C x 2.1)

Since the frequency of roughly 75 kHz is desired, select the K Factor from the table in SFR Definition 17.6 (OSCXCN) as KF = 7.7:

0.075 MHz = 7.7 / (C x 2.1)

C x 2.1 = 7.7 / 0.075 MHz

C = 102.6 / 2.0 pF = 51.3 pF

Therefore, the XFCN value to use in this example is 010b.



Downloaded from Arrow.com.

# 18. Port Input/Output

Digital and analog resources are available through 25 (C8051F540/1/4/5) or 18 (C8051F542/3/6/7) I/O pins. Port pins P0.0-P3.0 on the C8051F540/1/4/5 and port pins P0.0-P2.1 on the C8051F542/3/6/7 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 18.3. Port pin P3.0 on the C8051F540/1/4/5 can be used as GPIO and is shared with the C2 Interface Data signal (C2D). Similarly, port pin P2.1 is shared with C2D on the C8051F542/3/6/7. 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. 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 18.3 and Figure 18.4). The registers XBR0, XBR1, XBR2 are defined in SFR Definition 18.1 and SFR Definition 18.2 and are used to select internal digital functions.

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 Table 6.3 on page 51.

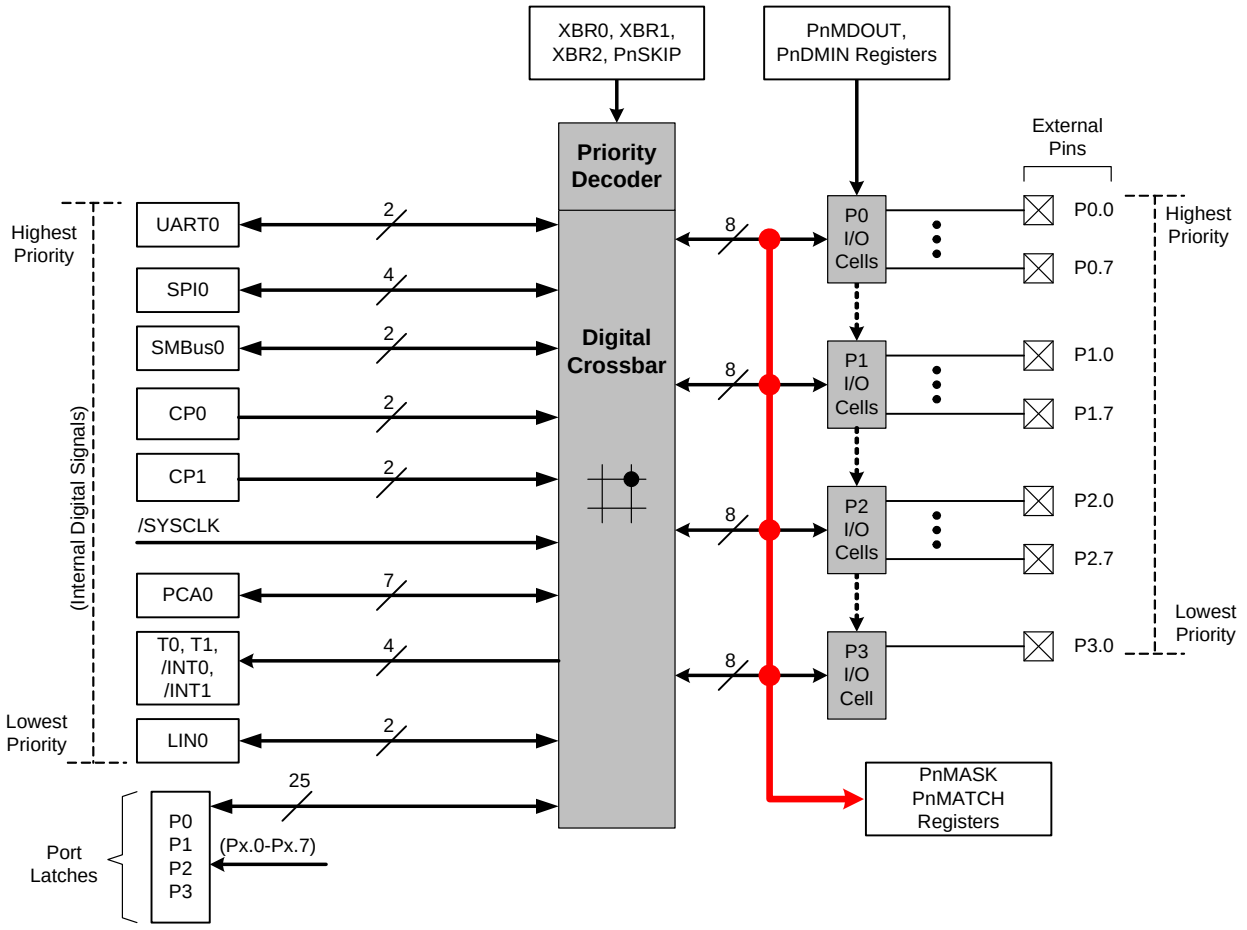



**Figure 18.1. Port I/O Functional Block Diagram**



Downloaded from Arrow.com.

## 18.1. Port I/O Modes of Operation

Port pins P0.0–P3.0 use the Port I/O cell shown in Figure 18.2. 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).

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

Any pins to be used as Comparator or ADC inputs, external oscillator inputs, or VREF should be configured for analog I/O (PnMDIN.n = 0). 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 inputs may still be used by analog peripherals; however, this practice is not recommended and may result in measurement errors.

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

Any pins to be used by digital peripherals (UART, SPI, SMBus, etc.), external digital event capture 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 VIO 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 VIO 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 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.

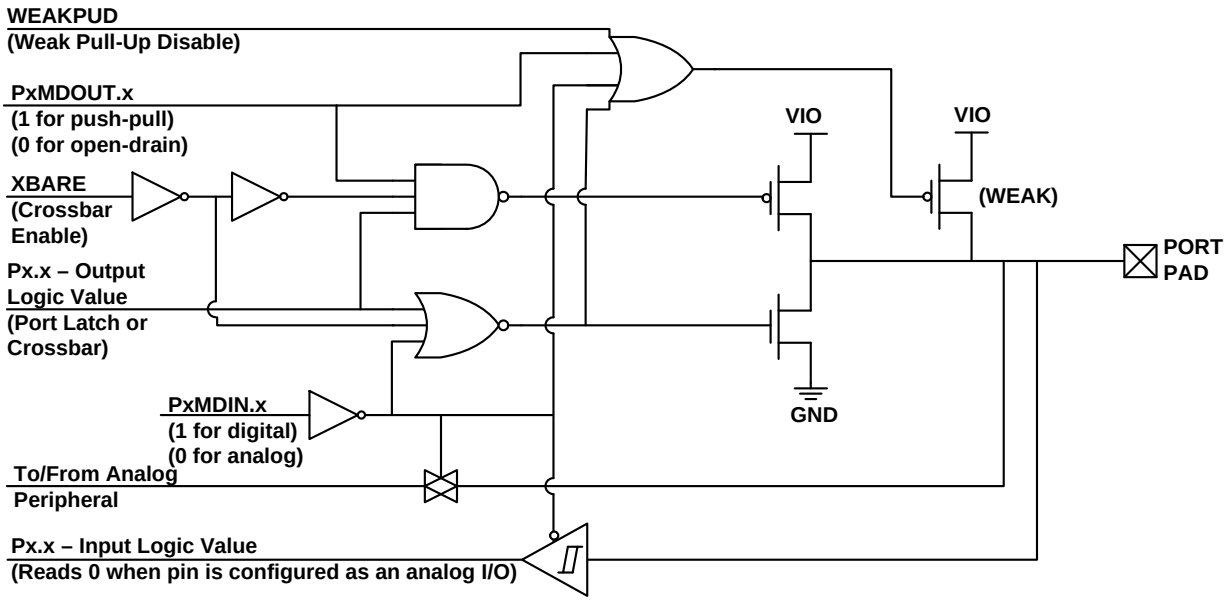



**Figure 18.2. Port I/O Cell Block Diagram**



Downloaded from Arrow.com.

### 18.1.3. Interfacing Port I/O in a Multi-Voltage System

All Port I/O are capable of interfacing to digital logic operating at a supply voltage higher than $V_{DD}$ and less than 5.25 V. Connect the $V_{IO}$ pin to the voltage source of the interface logic.

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

Port I/O pins P0.0–P3.0 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.

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

Table 18.1 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 18.1 shows the potential mapping of Port I/O to each analog function.

**Table 18.1. Port I/O Assignment for Analog Functions**

| Analog Function | Potentially Assignable Port Pins | SFR(s) used for Assignment |
|---|---|---|
| ADC Input | P0.0–P3.0* | ADC0MX, PnSKIP |
| Comparator0 or Compartor1 Input | P0.0–P2.7* | CPT0MX, CPT1MX, PnSKIP |
| Voltage Reference (VREF0) | P0.0 | REF0CN, PnSKIP |
| External Oscillator in Crystal Mode (XTAL1) | P0.2 | OSCXCN, PnSKIP |
| External Oscillator in RC, C, or Crystal Mode (XTAL2) | P0.3 | OSCXCN, PnSKIP |
| ***Note:** P2.2-P2.7, P3.0 are only available on the 32-pin packages | | |

### 18.2.2. 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 18.2 shows all available digital functions and the potential mapping of Port I/O to each digital function.

**Table 18.2. Port I/O Assignment for Digital Functions**

| Digital Function | Potentially Assignable Port Pins | SFR(s) used for Assignment |
|---|---|---|
| UART0, SPI0, SMBus, LIN0, CP0, CP0A, CP1, CP1A, SYSCLK, PCA0 (CEX0-5 and ECI), T0 or T1. | Any Port pin available for assignment by the Crossbar. This includes P0.0–P3.0* 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, XBR2 |
| ***Note:** P2.2-P2.7, P3.0 are only available on the 32-pin packages. | | |



Downloaded from Arrow.com.

**Table 18.2. Port I/O Assignment for Digital Functions**

| Digital Function | Potentially Assignable Port Pins | SFR(s) used for Assignment |
|---|---|---|
| Any pin used for GPIO | P0.0–P3.0* | P0SKIP, P1SKIP, P2SKIP, P3SKIP |
| ***Note:** P2.2-P2.7, P3.0 are only available on the 32-pin packages. | | |

### 18.2.3. Assigning Port I/O Pins to External Digital Event Capture Functions

External digital event capture 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 digital event capture 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 digital event capture functions cannot be used on pins configured for analog I/O. Table 18.3 shows all available external digital event capture functions.

**Table 18.3. Port I/O Assignment for External Digital Event Capture Functions**

| Digital Function | Potentially Assignable Port Pins | SFR(s) used for Assignment |
|---|---|---|
| External Interrupt 0 | P1.0–P1.7 | IT01CF |
| External Interrupt 1 | P1.0–P1.7 | IT01CF |
| Port Match | P0.0–P3.0* | P0MASK, P0MAT<br>P1MASK, P1MAT<br>P2MASK, P2MAT<br>P3MASK, P3MAT |
| ***Note:** P2.2-P2.7, P3.0 are only available on the 32-pin packages. | | |

## 18.3. Priority Crossbar Decoder

The Priority Crossbar Decoder (Figure 18.3) 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 assigned to pins P0.4 and P0. 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.

Because of the nature of Priority Crossbar Decoder, not all peripherals can be located on all port pins. Figure 18.3 maps peripherals to the potential port pins on which the peripheral I/O can appear.

**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, P0.1 if the ADC is configured to use the external conversion start signal (CNVSTR), P0.3 and/or P0.2 if the external oscillator circuit is enabled, 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.



Downloaded from Arrow.com.

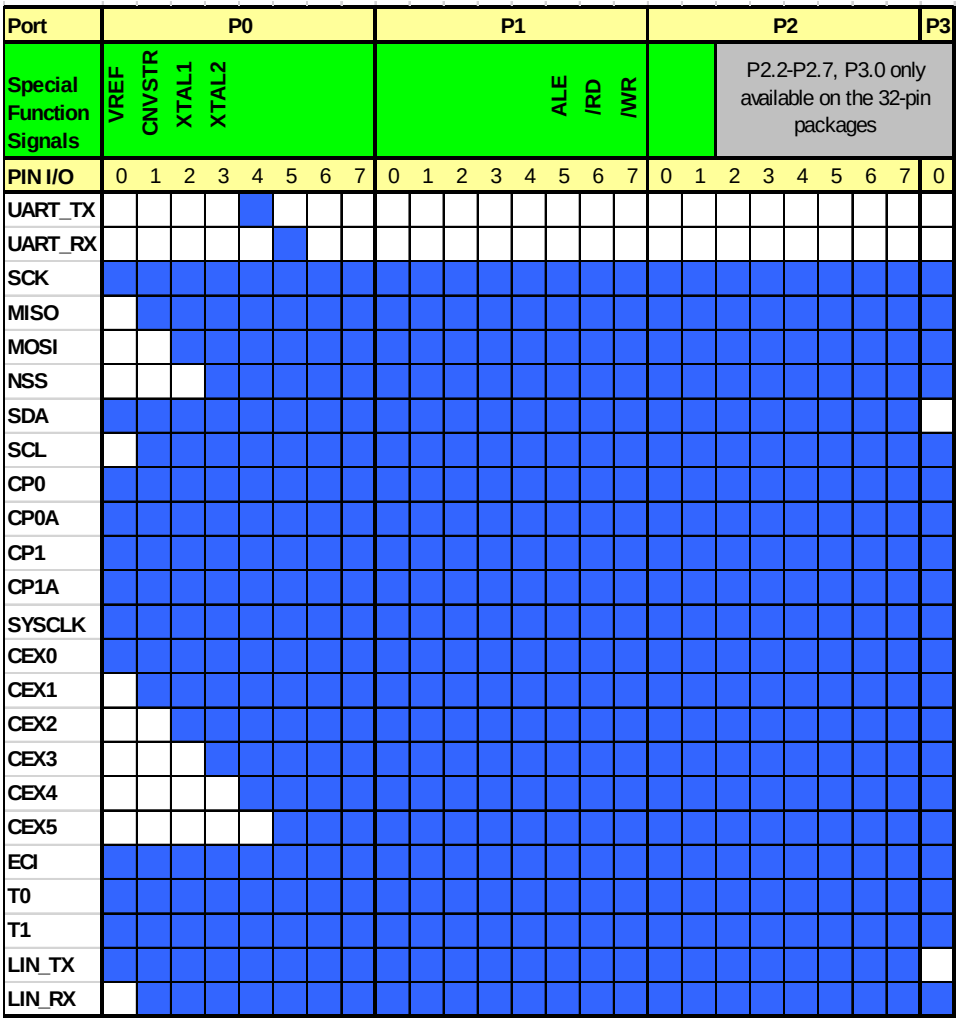



**Figure 18.3. Peripheral Availability on Port I/O Pins**

Registers XBR0, XBR1, and XBR2 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); and similarly when the UART or LIN are selected, the Crossbar assigns both pins associated with the peripheral (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.

**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.

As an example configuration, if SPI0 in 4-wire mode, and PCA0 Modules 0, 1, and 2 are enabled on the crossbar with P0.1, P0.2, and P0.5 skipped, the registers should be set as follows: XBR0 = 0x04 (SPI0 enabled), XBR1 = 0x0C (PCA0 modules 0, 1, and 2 enabled), XBR2 = 0x40 (Crossbar enabled), and P0SKIP = 0x26 (P0.1, P0.2, and P0.5 skipped). The resulting crossbar would look as shown in Figure 18.4.



Downloaded from Arrow.com.

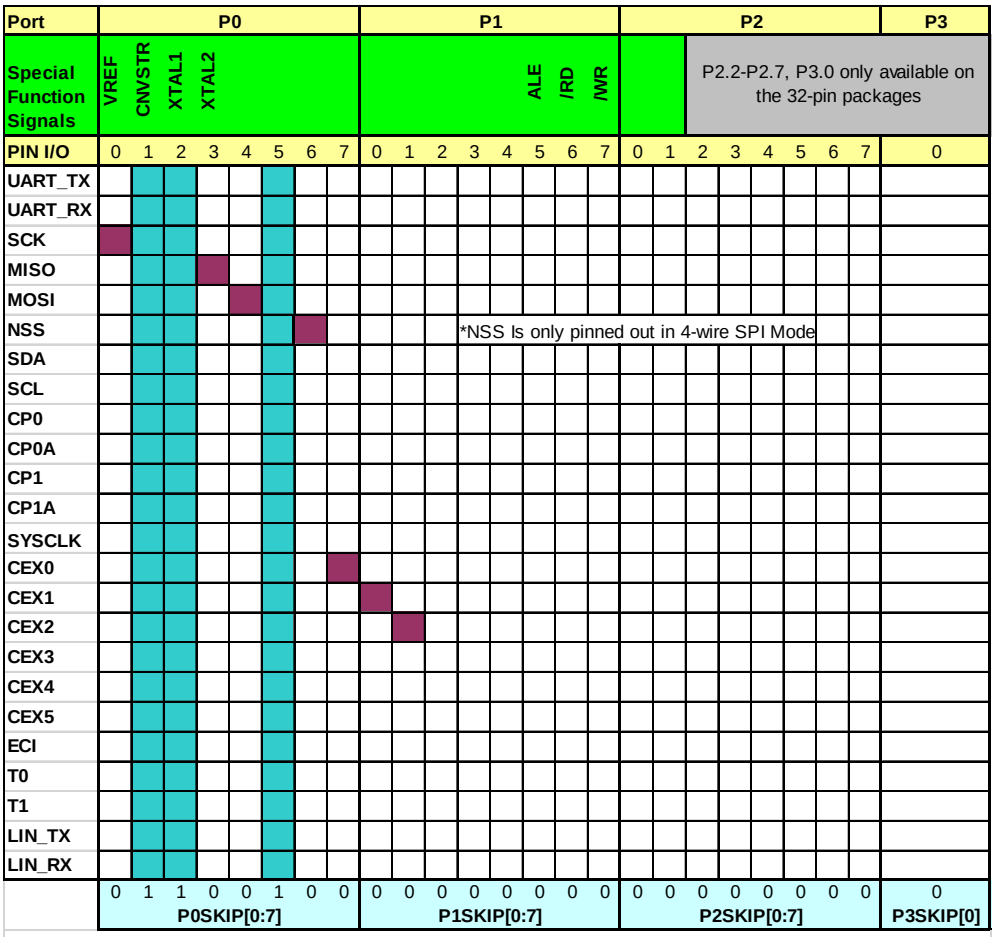



**Figure 18.4. Crossbar Priority Decoder in Example Configuration**

## 18.4. 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 18.13 for the PnMDIN register details.



Downloaded from Arrow.com.

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 XBR2 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, XBR1, and XBR2 must be loaded with the appropriate values to select the digital I/O functions required by the design. Setting the XBARE bit in XBR2 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.



Downloaded from Arrow.com.

## SFR Definition 18.1. XBR0: Port I/O Crossbar Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CP1AE | CP1E | CP0AE | CP0E | SMB0E | SPI0E | Reserved | URT0E |
| 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 = 0xE1; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7 | CP1AE | **Comparator1 Asynchronous Output Enable.**<br>0: Asynchronous CP1 unavailable at Port pin.<br>1: Asynchronous CP1 routed to Port pin. |
| 6 | CP1E | **Comparator1 Output Enable.**<br>0: CP1 unavailable at Port pin.<br>1: CP1 routed to Port pin. |
| 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 | SMB0E | **SMBus I/O Enable.**<br>0: SMBus I/O unavailable at Port pins.<br>1: SMBus I/O routed to Port pins. |
| 2 | 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. |
| 1 | Reserved | Always Write to 0. |
| 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. |



Downloaded from Arrow.com.

## SFR Definition 18.2. XBR1: Port I/O Crossbar Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T1E | T0E | ECIE | PCA0ME[2:0] | | | SYSCKE | Reserved |
| 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; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7 | T1E | **T1 Enable.**<br>0: T1 unavailable at Port pin.<br>1: T1 routed to Port pin. |
| 6 | T0E | **T0 Enable.**<br>0: T0 unavailable at Port pin.<br>1: T0 routed to Port pin. |
| 5 | ECIE | **PCA0 External Counter Input Enable.**<br>0: ECI unavailable at Port pin.<br>1: ECI routed to Port pin. |
| 4:2 | PCA0ME[2:0] | **PCA Module I/O Enable Bits.**<br>000: All PCA I/O unavailable at Port pins.<br>001: CEX0 routed to Port pin.<br>010: CEX0, CEX1 routed to Port pins.<br>011: CEX0, CEX1, CEX2 routed to Port pins.<br>100: CEX0, CEX1, CEX2, CEX3 routed to Port pins.<br>101: CEX0, CEX1, CEX2, CEX3, CEX4 routed to Port pins.<br>110: CEX0, CEX1, CEX2, CEX3, CEX4, CEX5 routed to Port pins.<br>111: RESERVED |
| 1 | SYSCKE | **$\overline{\text{SYSCLK}}$ Output Enable.**<br>0: $\overline{\text{SYSCLK}}$ unavailable at Port pin.<br>1: $\overline{\text{SYSCLK}}$ output routed to Port pin. |
| 0 | Reserved | Always Write to 0. |



Downloaded from Arrow.com.

## SFR Definition 18.3. XBR2: Port I/O Crossbar Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | WEAKPUD | XBARE | Reserved | | | | | LIN0E |
| 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 = 0xC7; SFR Page = 0x0F

| 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. |
| 5:1 | Reserved | Always Write to 00000b. |
| 0 | LIN0E | **LIN I/O Output Enable.**<br>0: LIN I/O unavailable at Port pin.<br>1: LIN_TX, LIN_RX routed to Port pins. |



Downloaded from Arrow.com.

## 18.5. Port Match

Port match functionality allows system events to be triggered by a logic value change on P0, P1, P2 or P3. A software controlled value stored in the PnMATCH registers specifies the expected or normal logic values of P0, P1, P2, and P3. 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, P1, P2, or P3 input pins regardless of the XBRn settings.

The PnMASK registers can be used to individually select which of the port pins should be compared against the PnMATCH registers. A Port mismatch event is generated if (Pn & PnMASK) does not equal (PnMATCH & PnMASK), where n is 0, 1, 2 or 3

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 18.4. 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 = 0xF2; SFR Page = 0x00

| 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 18.5. 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 = 0xF1; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | P0MAT[7:0] | **Port 0 Match Value.**<br>Match comparison value used on Port 0 for bits in P0MAT 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. |



Downloaded from Arrow.com.

## SFR Definition 18.6. 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 = 0xF4; SFR Page = 0x00

| 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 18.7. 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 = 0xF3; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | P1MAT[7:0] | **Port 1 Match Value.**<br>Match comparison value used on Port 1 for bits in P1MAT 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. |



Downloaded from Arrow.com.

## SFR Definition 18.8. P2MASK: Port 2 Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P2MASK[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xB2; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MASK[7:0] | **Port 2 Mask Value.**<br>Selects P2 pins to be compared to the corresponding bits in P2MAT.<br>0: P2.n pin logic value is ignored and cannot cause a Port Mismatch event.<br>1: P2.n pin logic value is compared to P2MAT.n. |
| **Note:** Ports 2.2-P2.7 only available on 32-pin packages. | | |

## SFR Definition 18.9. P2MAT: Port 2 Match Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P2MAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xB1; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MAT[7:0] | **Port 2 Match Value.**<br>Match comparison value used on Port 2 for bits in P2MAT which are set to 1.<br>0: P2.n pin logic value is compared with logic LOW.<br>1: P2.n pin logic value is compared with logic HIGH. |
| **Note:** Ports 2.2-P2.7 only available on 32-pin packages. | | |



Downloaded from Arrow.com.

## SFR Definition 18.10. P3MASK: Port 3 Mask Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P3MASK[0] |
| Type | R | R | R | R | R | R | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xAF; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care. |
| 0 | P3MASK[7:0] | **Port 3 Mask Value.**<br>Selects P3.n pins to be compared to the corresponding bits in P3MAT.<br>0: P3.n pin logic value is ignored and cannot cause a Port Mismatch event.<br>1: P3.n pin logic value is compared to P3MAT.n. |
| **Note:** P3.0 is only available on the 32-pin packages. | | |

## SFR Definition 18.11. P3MAT: Port 3 Match Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | 0 | 0 | 0 | 0 | 0 | 0 | 0 | P3MAT[0] |
| Type | R | R | R | R | R | R | R | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xAE; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care. |
| 0 | P3MAT[0] | **Port 3 Match Value.**<br>Match comparison value used on Port 3 for bits in P3MAT which are set to 1.<br>0: P3.n pin logic value is compared with logic LOW.<br>1: P3.n pin logic value is compared with logic HIGH. |
| **Note:** P3.0 is only available on the 32-pin packages. | | |



Downloaded from Arrow.com.

## 18.6. 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.

Ports 0–3 have 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 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 18.12. 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; SFR Page = All Pages; 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. |



Downloaded from Arrow.com.

## SFR Definition 18.13. 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; SFR Page = 0x0F

| 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 pull-up and digital receiver disabled. For analog mode, the pin also needs to be configured for open-drain mode in the P0MDOUT register.<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 18.14. 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; SFR Page = 0x0F

| 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. |



Downloaded from Arrow.com.

## SFR Definition 18.15. 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; SFR Page = 0x0F

| 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 18.16. 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; SFR Page = All Pages; 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. |



Downloaded from Arrow.com.

## SFR Definition 18.17. 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; SFR Page = 0x0F

| 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 pull-up and digital receiver disabled. For analog mode, the pin also needs to be configured for open-drain mode in the P1MDOUT register.<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 18.18. 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; SFR Page = 0x0F

| 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. |



Downloaded from Arrow.com.

## SFR Definition 18.19. 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; SFR Page = 0x0F

| 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 18.20. P2: Port 2

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

SFR Address = 0xA0; SFR Page = All Pages; Bit-Addressable

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:0 | P2[7:0] | **Port 2Data.**<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:** P2.2-P2.7 are only available on the 32-pin packages. | | | | |



Downloaded from Arrow.com.

## SFR Definition 18.21. P2MDIN: Port 2 Input Mode

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

SFR Address = 0xF3; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MDIN[7:0] | **Analog Configuration Bits for P2.7–P2.0 (respectively).**<br>Port pins configured for analog mode have their weak pull-up and digital receiver disabled. For analog mode, the pin also needs to be configured for open-drain mode in the P2MDOUT register.<br>0: Corresponding P2.n pin is configured for analog mode.<br>1: Corresponding P2.n pin is not configured for analog mode. |
| **Note:** P2.2-P2.7 are only available on the 32-pin packages. | | |

## SFR Definition 18.22. P2MDOUT: Port 2 Output Mode

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

SFR Address = 0xA6; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MDOUT[7:0] | **Output Configuration Bits for P2.7–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:** P2.2-P2.7 are only available on the 32-pin packages. | | |



Downloaded from Arrow.com.

## SFR Definition 18.23. P2SKIP: Port 2 Skip

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

SFR Address = 0xD6; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2SKIP[7:0] | **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:** P2.2-P2.7 are only available on the 32-pin packages. | | |

## SFR Definition 18.24. P3: Port 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | | P3 |
| Type | R | R | R | R | R | R | R | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xB0; SFR Page = All Pages; Bit-Addressable

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care. | | |
| 0 | P3[0] | **Port 3 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: P3.n Port pin is logic LOW.<br>1: P3.n Port pin is logic HIGH. |
| **Note:** Port P3.0 is only available on the 32-pin packages. | | | | |



Downloaded from Arrow.com.

## SFR Definition 18.25. P3MDIN: Port 3 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | | P3MDIN[0] |
| Type | R | R | R | R | R | R | R | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xF4; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care. |
| 0 | P3MDIN[0] | **Analog Configuration Bits for P3.0.**<br>Port pins configured for analog mode have their weak pull-up and digital receiver disabled. For analog mode, the pin also needs to be configured for open-drain mode in the P3MDOUT register.<br>0: Corresponding P3.n pin is configured for analog mode.<br>1: Corresponding P3.n pin is not configured for analog mode. |
| **Note:** Port P3.0 is only available on the 32-pin packages. | | |

## SFR Definition 18.26. P3MDOUT: Port 3 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | | P3MDOUT[0] |
| Type | R | R | R | R | R | R | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xAE; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care. |
| 7:0 | P3MDOUT[7:0] | **Output Configuration Bits for P3.0.**<br>These bits are ignored if the corresponding bit in register P3MDIN is logic 0.<br>0: Corresponding P3.n Output is open-drain.<br>1: Corresponding P3.n Output is push-pull. |
| **Note:** Port P3.0 is only available on the 32-pin packages. | | |



Downloaded from Arrow.com.

## SFR Definition 18.27. P3SKIP: Port 3Skip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | | P3SKIP[0] |
| Type | R | R | R | R | R | R | R | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD7; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care. |
| 0 | P3SKIP[0] | **Port 3 Crossbar Skip Enable Bits.**<br>These bits select Port 3 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 P3.n pin is not skipped by the Crossbar.<br>1: Corresponding P3.n pin is skipped by the Crossbar. |
| **Note:** Port P3.0 is only available on the 32-pin packages. | | |



Downloaded from Arrow.com.

# 19. Local Interconnect Network (LIN)

**Important Note:** This chapter assumes an understanding of the Local Interconnect Network (LIN) protocol. For more information about the LIN protocol, including specifications, please refer to the LIN consortium (http://www.lin-subbus.org).

LIN is an asynchronous, serial communications interface used primarily in automotive networks. The Silicon Laboratories LIN controller is compliant to the 2.1 Specification, implements a complete hardware LIN interface and includes the following features:

- Selectable Master and Slave modes.
- Automatic baud rate option in slave mode.
- The internal oscillator is accurate to within 0.5% of 24 MHz across the entire temperature range and for VDD voltages greater than or equal to the minimum output of the on-chip voltage regulator, so an external oscillator is not necessary for master mode operation for most systems.

**Note:** The minimum system clock (SYSCLK) required when using the LIN controller is 8 MHz.

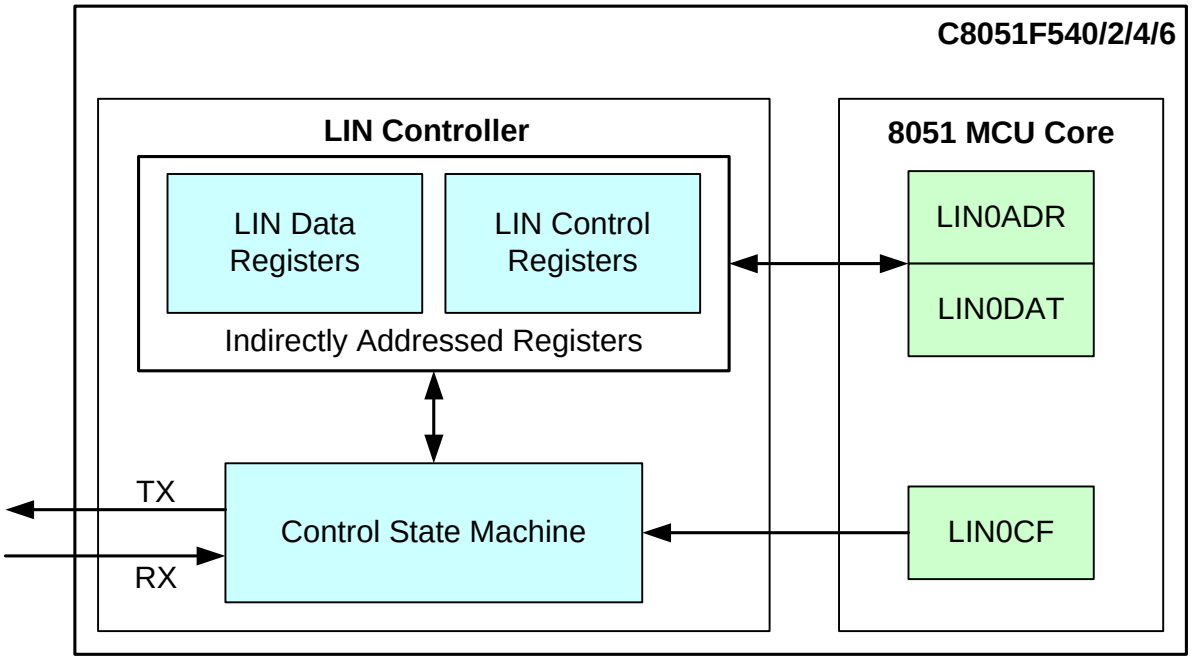



**Figure 19.1. LIN Block Diagram**

The LIN controller has four main components:

- LIN Access Registers—Provide the interface between the MCU core and the LIN controller.
- LIN Data Registers—Where transmitted and received message data bytes are stored.
- LIN Control Registers—Control the functionality of the LIN interface.
- Control State Machine and Bit Streaming Logic—Contains the hardware that serializes messages and controls the bus timing of the controller.



Downloaded from Arrow.com.

## 19.1. Software Interface with the LIN Controller

The selection of the mode (Master or Slave) and the automatic baud rate feature are done though the LIN0 Control Mode (LIN0CF) register. The other LIN registers are accessed indirectly through the two SFRs LIN0 Address (LIN0ADR) and LIN0 Data (LIN0DAT). The LIN0ADR register selects which LIN register is targeted by reads/writes of the LIN0DAT register. The full list of indirectly-accessible LIN registers is given in Table 19.4 on page 179.

## 19.2. LIN Interface Setup and Operation

The hardware based LIN controller allows for the implementation of both Master and Slave nodes with minimal firmware overhead and complete control of the interface status while allowing for interrupt and polled mode operation.

The first step to use the controller is to define the basic characteristics of the node:

Mode—Master or Slave

Baud Rate—Either defined manually or using the autobaud feature (slave mode only)

Checksum Type—Select between classic or enhanced checksum, both of which are implemented in hardware.

### 19.2.1. Mode Definition

Following the LIN specification, the controller implements in hardware both the Slave and Master operating modes. The mode is configured using the MODE bit (LIN0CF.6).

### 19.2.2. Baud Rate Options: Manual or Autobaud

The LIN controller can be selected to have its baud rate calculated manually or automatically. A master node must always have its baud rate set manually, but slave nodes can choose between a manual or automatic setup. The configuration is selected using the ABAUD bit (LIN0CF.5).

Both the manual and automatic baud rate configurations require additional setup. The following sections explain the different options available and their relation with the baud rate, along with the steps necessary to achieve the required baud rate.

### 19.2.3. Baud Rate Calculations: Manual Mode

The baud rate used by the LIN controller is a function of the System Clock (SYSCLK) and the LIN timing registers according to the following equation:

$$baud_rate = \frac{SYSCLK}{2^{(prescaler+1)} \times divider \times (multiplier+1)}$$

The prescaler, divider and multiplier factors are part of the LIN0DIV and LIN0MUL registers and can assume values in the following range:

**Table 19.1. Baud Rate Calculation Variable Ranges**

| Factor | Range |
|---|---|
| prescaler | 0…3 |
| multiplier | 0…31 |
| divider | 200…511 |

**Important**: The minimum system clock (SYSCLK) to operate the LIN controller is 8 MHz.



Downloaded from Arrow.com.

Use the following equations to calculate the values for the variables for the baud-rate equation:

$$\text{multiplier} = \frac{20000}{\text{baud_rate}} - 1$$

$$\text{prescaler} = \ln\left[\frac{\text{SYSCLK}}{(\text{multiplier}+1)\times\text{baud_rate}\times 200}\right]\times\frac{1}{\ln 2} - 1$$

$$\text{divider} = \frac{\text{SYSCLK}}{(2^{(\text{prescaler}+1)}\times(\text{multiplier}+1)\times\text{baud_rate})}$$

In all of these equations, the results must be rounded down to the nearest integer.

The following example shows the steps for calculating the baud rate values for a Master node running at 24 MHz and communicating at 19200 bits/sec. First, calculate the multiplier:

$$\text{multiplier} = \frac{20000}{19200} - 1 = 0.0417 \cong 0$$

Next, calculate the prescaler:

$$\text{prescaler} = \ln\frac{24000000}{(0+1)\times 19200\times 200}\times\frac{1}{\ln 2} - 1 = 1.644 \cong 1$$

Finally, calculate the divider:

$$\text{divider} = \frac{24000000}{2^{(1+1)}\times(0+1)\times 19200} = 312.5 \cong 312$$

These values lead to the following baud rate:

$$\text{baud_rate} = \frac{24000000}{2^{(1+1)}\times(0+1)\times 312} \cong 19230.77$$

The following code programs the interface in Master mode, using the Enhanced Checksum and enables the interface to operate at 19230 bits/sec using a 24 MHz system clock.

```
LIN0CF   = 0x80;                        // Activate the interface
LIN0CF  |= 0x40;                        // Set the node as a Master

LIN0ADR  = 0x0D;                        // Point to the LIN0MUL register
// Initialize the register (prescaler, multiplier and bit 8 of divider)
LIN0DAT  = ( 0x01 << 6 ) + ( 0x00 << 1 ) + ( ( 0x138 & 0x0100 ) >> 8 );
LIN0ADR   = 0x0C;                       // Point to the LIN0DIV register
LIN0DAT   = (unsigned char)_0x138;      // Initialize LIN0DIV

LIN0ADR   = 0x0B;                       // Point to the LIN0SIZE register
LIN0DAT  |= 0x80;                       // Initialize the checksum as Enhanced

LIN0ADR   = 0x08;                       // Point to LIN0CTRL register
LIN0DAT   = 0x0C;                       // Reset any error and the interrupt
```

Table 19.2 includes the configuration values required for the typical system clocks and baud rates:



Downloaded from Arrow.com.

**Table 19.2. Manual Baud Rate Parameters Examples**

| | Baud (bits/sec) | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 20 K | | | 19.2 K | | | 9.6 K | | | 4.8 K | | | 1 K | | |
| SYSCLK (MHz) | Mult. | Pres. | Div. | Mult. | Pres. | Div. | Mult. | Pres. | Div. | Mult. | Pres. | Div. | Mult. | Pres. | Div. |
| 25 | 0 | 1 | 312 | 0 | 1 | 325 | 1 | 1 | 325 | 3 | 1 | 325 | 19 | 1 | 312 |
| 24.5 | 0 | 1 | 306 | 0 | 1 | 319 | 1 | 1 | 319 | 3 | 1 | 319 | 19 | 1 | 306 |
| 24 | 0 | 1 | 300 | 0 | 1 | 312 | 1 | 1 | 312 | 3 | 1 | 312 | 19 | 1 | 300 |
| 22.1184 | 0 | 1 | 276 | 0 | 1 | 288 | 1 | 1 | 288 | 3 | 1 | 288 | 19 | 1 | 276 |
| 16 | 0 | 1 | 200 | 0 | 1 | 208 | 1 | 1 | 208 | 3 | 1 | 208 | 19 | 1 | 200 |
| 12.25 | 0 | 0 | 306 | 0 | 0 | 319 | 1 | 0 | 319 | 3 | 0 | 319 | 19 | 0 | 306 |
| 12 | 0 | 0 | 300 | 0 | 0 | 312 | 1 | 0 | 312 | 3 | 0 | 312 | 19 | 0 | 300 |
| 11.0592 | 0 | 0 | 276 | 0 | 0 | 288 | 1 | 0 | 288 | 3 | 0 | 288 | 19 | 0 | 276 |
| 8 | 0 | 0 | 200 | 0 | 0 | 208 | 1 | 0 | 208 | 3 | 0 | 208 | 19 | 0 | 200 |

### 19.2.4. Baud Rate Calculations—Automatic Mode

If the LIN controller is configured for slave mode, only the prescaler and divider need to be calculated:

$$\text{prescaler} = \ln\left[\frac{\text{SYSCLK}}{4000000}\right] \times \frac{1}{\ln 2} - 1$$

$$\text{divider} = \frac{\text{SYSCLK}}{2^{(\text{prescaler}+1)} \times 20000}$$

The following example calculates the values of these variables for a 24 MHz system clock:

$$\text{prescaler} = \ln\left[\frac{24000000}{4000000}\right] \times \frac{1}{\ln 2} - 1 = 1.585 \cong 1$$

$$\text{divider} = \frac{24000000}{2^{(1+1)} \times 20000} = 300$$

Table 19.3 presents some typical values of system clock and baud rate along with their factors.



Downloaded from Arrow.com.

**Table 19.3. Autobaud Parameters Examples**

| System Clock (MHz) | Prescaler | Divider |
|---|---|---|
| 25 | 1 | 312 |
| 24.5 | 1 | 306 |
| 24 | 1 | 300 |
| 22.1184 | 1 | 276 |
| 16 | 1 | 200 |
| 12.25 | 0 | 306 |
| 12 | 0 | 300 |
| 11.0592 | 0 | 276 |
| 8 | 0 | 200 |

## 19.3. LIN Master Mode Operation

The master node is responsible for the scheduling of messages and sends the header of each frame containing the SYNCH BREAK FIELD, SYNCH FIELD, and IDENTIFIER FIELD. The steps to schedule a message transmission or reception are listed below.

1. Load the 6-bit Identifier into the LIN0ID register.
2. Load the data length into the LIN0SIZE register. Set the value to the number of data bytes or "1111b" if the data length should be decoded from the identifier. Also, set the checksum type, classic or enhanced, in the same LIN0SIZE register.
3. Set the data direction by setting the TXRX bit (LIN0CTRL.5). Set the bit to 1 to perform a master transmit operation, or set the bit to 0 to perform a master receive operation.
4. If performing a master transmit operation, load the data bytes to transmit into the data buffer (LIN0DT1 to LIN0DT8).
5. Set the STREQ bit (LIN0CTRL.0) to start the message transfer. The LIN controller will schedule the message frame and request an interrupt if the message transfer is successfully completed or if an error has occurred.

This code segment shows the procedure to schedule a message in a transmission operation:

```
LIN0ADR   = 0x08;               // Point to LIN0CTRL
LIN0DAT  |= 0x20;               // Select to transmit data
LIN0ADR   = 0x0E;               // Point to LIN0ID
LIN0DAT   = 0x11;               // Load the ID, in this example 0x11
LIN0ADR   = 0x0B;               // Point to LIN0SIZE
LIN0DAT  = ( LIN0DAT & 0xF0 ) | 0x08;      // Load the size with 8

LIN0ADR  = 0x00;               // Point to Data buffer first byte
for (i=0; i<8; i++)
{
   LIN0DAT = i + 0x41;         // Load the buffer with ‘A’, ‘B’, ...
   LIN0ADR++;                  // Increment the address to the next buffer
}
LIN0ADR  = 0x08;               // Point to LIN0CTRL
LIN0DAT  = 0x01;               // Start Request
```

The application should perform the following steps when an interrupt is requested.



Downloaded from Arrow.com.

1. Check the DONE bit (LIN0ST.0) and the ERROR bit (LIN0ST.2).
2. If performing a master receive operation and the transfer was successful, read the received data from the data buffer.
3. If the transfer was not successful, check the error register to determine the kind of error. Further error handling has to be done by the application.
4. Set the RSTINT (LIN0CTRL.3) and RSTERR bits (LIN0CTRL.2) to reset the interrupt request and the error flags.

## 19.4. LIN Slave Mode Operation

When the device is configured for slave mode operation, it must wait for a command from a master node. Access from the firmware to the data buffer and ID registers of the LIN controller is only possible when a data request is pending (DTREQ bit (LIN0ST.4) is 1) and also when the LIN bus is not active (ACTIVE bit (LIN0ST.7) is set to 0).

The LIN controller in slave mode detects the header of the message frame sent by the LIN master. If slave synchronization is enabled (autobaud), the slave synchronizes its internal bit time to the master bit time.

The LIN controller configured for slave mode will generated an interrupt in one of three situations:

1. After the reception of the IDENTIFIER FIELD
2. When an error is detected
3. When the message transfer is completed.

The application should perform the following steps when an interrupt is detected:

1. Check the status of the DTREQ bit (LIN0ST.4). This bit is set when the IDENTIFIER FIELD has been received.
2. If DTREQ (LIN0ST.4) is set, read the identifier from LIN0ID and process it. If DTREQ (LIN0ST.4) is not set, continue to step 7.
3. Set the TXRX bit (LIN0CTRL.5) to 1 if the current frame is a transmit operation for the slave and set to 0 if the current frame is a receive operation for the slave.
4. Load the data length into LIN0SIZE.
5. For a slave transmit operation, load the data to transmit into the data buffer.
6. Set the DTACK bit (LIN0CTRL.4). Continue to step 10.
7. If DTREQ (LIN0ST.4) is not set, check the DONE bit (LIN0ST.0). The transmission was successful if the DONE bit is set.
8. If the transmission was successful and the current frame was a receive operation for the slave, load the received data bytes from the data buffer.
9. If the transmission was not successful, check LIN0ERR to determine the nature of the error. Further error handling has to be done by the application.
10. Set the RSTINT (LIN0CTRL.3) and RSTERR bits (LIN0CTRL.2) to reset the interrupt request and the error flags.

In addition to these steps, the application should be aware of the following:

1. If the current frame is a transmit operation for the slave, steps 1 through 5 must be completed during the IN-FRAME RESPONSE SPACE. If it is not completed in time, a timeout will be detected by the master.
2. If the current frame is a receive operation for the slave, steps 1 through 5 have to be finished until the reception of the first byte after the IDENTIFIER FIELD. Otherwise, the internal receive buffer of the LIN controller will be overwritten and a timeout error will be detected in the LIN controller.



Downloaded from Arrow.com.

3. The LIN controller does not directly support LIN Version 1.3 Extended Frames. If the application detects an unknown identifier (e.g. extended identifier), it has to write a 1 to the STOP bit (LIN0CTRL.7) instead of setting the DTACK (LIN0CTRL.4) bit. At that time, steps 2 through 5 can then be skipped. In this situation, the LIN controller stops the processing of LIN communication until the next SYNC BREAK is received.
4. Changing the configuration of the checksum during a transaction will cause the interface to reset and the transaction to be lost. To prevent this, the checksum should not be configured while a transaction is in progress. The same applies to changes in the LIN interface mode from slave mode to master mode and from master mode to slave mode.

## 19.5. Sleep Mode and Wake-Up

To reduce the system's power consumption, the LIN Protocol Specification defines a Sleep Mode. The message used to broadcast a Sleep Mode request must be transmitted by the LIN master application in the same way as a normal transmit message. The LIN slave application must decode the Sleep Mode Frame from the Identifier and data bytes. After that, it has to put the LIN slave node into the Sleep Mode by setting the SLEEP bit (LIN0CTRL.6).

If the SLEEP bit (LIN0CTRL.6) of the LIN slave application is not set and there is no bus activity for four seconds (specified bus idle timeout), the IDLTOUT bit (LIN0ST.6) is set and an interrupt request is generated. After that the application may assume that the LIN bus is in Sleep Mode and set the SLEEP bit (LIN0CTRL.6).

Sending a wake-up signal from the master or any slave node terminates the Sleep Mode of the LIN bus. To send a wake-up signal, the application has to set the WUPREQ bit (LIN0CTRL.1). After successful transmission of the wake-up signal, the DONE bit (LIN0ST.0) of the master node is set and an interrupt request is generated. The LIN slave does not generate an interrupt request after successful transmission of the wake-up signal but it generates an interrupt request if the master does not respond to the wake-up signal within 150 milliseconds. In that case, the ERROR bit (LIN0ST.2) and TOUT bit (LIN0ERR.2) are set. The application then has to decide whether or not to transmit another wake-up signal.

All LIN nodes that detect a wake-up signal will set the WAKEUP (LIN0ST.1) and DONE bits (LIN0ST.0) and generate an interrupt request. After that, the application has to clear the SLEEP bit (LIN0CTRL.6) in the LIN slave.

## 19.6. Error Detection and Handling

The LIN controller generates an interrupt request and stops the processing of the current frame if it detects an error. The application has to check the type of error by processing LIN0ERR. After that, it has to reset the error register and the ERROR bit (LIN0ST.2) by writing a 1 to the RSTERR bit (LIN0CTRL.2). Starting a new message with the LIN controller selected as master or sending a Wakeup signal with the LIN controller selected as a master or slave is possible only if the ERROR bit (LIN0ST.2) is set to 0.



Downloaded from Arrow.com.

## 19.7. LIN Registers

The following Special Function Registers (SFRs) and indirect registers are available for the LIN controller.

### 19.7.1. LIN Direct Access SFR Registers Definitions

### SFR Definition 19.1. LIN0ADR: LIN0 Indirect Address Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | LIN0ADR[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD3; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | LIN0ADR[7:0] | **LIN Indirect Address Register Bits.**<br>This register hold an 8-bit address used to indirectly access the LIN0 core registers. Table 19.4 lists the LIN0 core registers and their indirect addresses. Reads and writes to LIN0DAT will target the register indicated by the LIN0ADR bits. |

### SFR Definition 19.2. LIN0DAT: LIN0 Indirect Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | LIN0DAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD2; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | LIN0DAT[7:0] | **LIN Indirect Data Register Bits.**<br>When this register is read, it will read the contents of the LIN0 core register pointed to by LIN0ADR.<br>When this register is written, it will write the value to the LIN0 core register pointed to by LIN0ADR. |



Downloaded from Arrow.com.

## SFR Definition 19.3. LIN0CF: LIN0 Control Mode Register

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

SFR Address = 0xC9; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7 | LINEN | **LIN Interface Enable Bit.**<br>0: LIN0 is disabled.<br>1: LIN0 is enabled. |
| 6 | MODE | **LIN Mode Selection Bit.**<br>0: LIN0 operates in slave mode.<br>1: LIN0 operates in master mode. |
| 5 | ABAUD | **LIN Mode Automatic Baud Rate Selection.**<br>This bit only has an effect when the MODE bit is configured for slave mode.<br>0: Manual baud rate selection is enabled.<br>1: Automatic baud rate selection is enabled. |
| 4:0 | Unused | Read = 00000b; Write = Don't Care |



Downloaded from Arrow.com.

### 19.7.2. LIN Indirect Access SFR Registers Definitions

Table 19.4 lists the 15 indirect registers used to configured and communicate with the LIN controller.

**Table 19.4. LIN Registers* (Indirectly Addressable)**

| Name | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|
| LIN0DT1 | 0x00 | DATA1[7:0] | | | | | | | |
| LIN0DT2 | 0x01 | DATA2[7:0] | | | | | | | |
| LIN0DT3 | 0x02 | DATA3[7:0] | | | | | | | |
| LIN0DT4 | 0x03 | DATA4[7:0] | | | | | | | |
| LIN0DT5 | 0x04 | DATA5[7:0] | | | | | | | |
| LIN0DT6 | 0x05 | DATA67:0] | | | | | | | |
| LIN0DT7 | 0x06 | DATA7[7:0] | | | | | | | |
| LIN0DT8 | 0x07 | DATA8[7:0] | | | | | | | |
| LIN0CTRL | 0x08 | STOP(s) | SLEEP(s) | TXRX | DTACK(s) | RSTINT | RSTERR | WUPREQ | STREQ(m) |
| LIN0ST | 0x09 | ACTIVE | IDLTOUT | ABORT(s) | DTREQ(s) | LININT | ERROR | WAKEUP | DONE |
| LIN0ERR | 0x0A | | | | SYNCH(s) | PRTY(s) | TOUT | CHK | BITERR |
| LIN0SIZE | 0x0B | ENHCHK | | | | LINSIZE[3:0] | | | |
| LIN0DIV | 0x0C | DIVLSB[7:0] | | | | | | | |
| LIN0MUL | 0x0D | PRESCL[1:0] | | LINMUL[4:0] | | | | | DIV9 |
| LIN0ID | 0x0E | | | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 |

***Note:** These registers are used in both master and slave mode. The register bits marked with (m) are accessible only in Master mode while the register bits marked with (s) are accessible only in slave mode. All other registers are accessible in both modes.



Downloaded from Arrow.com.

## LIN Register Definition 19.4. LIN0DTn: LIN0 Data Byte n

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

Indirect Address: LIN0DT1 = 0x00, LIN0DT2 = 0x01, LIN0DT3 = 0x02, LIN0DT4 = 0x03, LIN0DT5 = 0x04, LIN0DT6 = 0x05, LIN0DT7 = 0x06, LIN0DT8 = 0x07

| Bit | Name | Function |
|---|---|---|
| 7:0 | DATAn[7:0] | **LIN Data Byte n.**<br>Serial Data Byte that is received or transmitted across the LIN interface. |



Downloaded from Arrow.com.

## LIN Register Definition 19.5. LIN0CTRL: LIN0 Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | STOP | SLEEP | TXRX | DTACK | RSTINT | RSTERR | WUPREQ | STREQ |
| Type | W | R/W | R/W | R/W | W | W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Indirect Address = 0x08

| Bit | Name | Function |
|---|---|---|
| 7 | STOP | **Stop Communication Processing Bit. (slave mode only)**<br>This bit always reads as 0.<br>0: No effect.<br>1: Block the processing of LIN communications until the next SYNC BREAK signal. |
| 6 | SLEEP | **Sleep Mode Bit. (slave mode only)**<br>0: Wake the device after receiving a Wakeup interrupt.<br>1: Put the device into sleep mode after receiving a Sleep Mode frame or a bus idle timeout. |
| 5 | TXRX | **Transmit / Receive Selection Bit.**<br>0: Current frame is a receive operation.<br>1: Current frame is a transmit operation. |
| 4 | DTACK | **Data Acknowledge Bit. (slave mode only)**<br>Set to 1 after handling a data request interrupt to acknowledge the transfer. The bit will automatically be cleared to 0 by the LIN controller. |
| 3 | RSTINT | **Reset Interrupt Bit.**<br>This bit always reads as 0.<br>0: No effect.<br>1: Reset the LININT bit (LIN0ST.3). |
| 2 | RSTERR | **Reset Error Bit.**<br>This bit always reads as 0.<br>0: No effect.<br>1: Reset the error bits in LIN0ST and LIN0ERR. |
| 1 | WUPREQ | **Wakeup Request Bit.**<br>Set to 1 to terminate sleep mode by sending a wakeup signal. The bit will automatically be cleared to 0 by the LIN controller. |
| 0 | STREQ | **Start Request Bit. (master mode only)**<br>1: Start a LIN transmission. This should be set only after loading the identifier, data length and data buffer if necessary.<br>The bit is reset to 0 upon transmission completion or error detection. |



Downloaded from Arrow.com.

## LIN Register Definition 19.6. LIN0ST: LIN0 Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ACTIVE | IDLTOUT | ABORT | DTREQ | LININT | ERROR | WAKEUP | DONE |
| Type | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Indirect Address = 0x09

| Bit | Name | Function |
|---|---|---|
| 7 | ACTIVE | **LIN Active Indicator Bit.**<br>0: No transmission activity detected on the LIN bus.<br>1: Transmission activity detected on the LIN bus. |
| 6 | IDLT | **Bus Idle Timeout Bit. (slave mode only)**<br>0: The bus has not been idle for four seconds.<br>1: No bus activity has been detected for four seconds, but the bus is not yet in Sleep mode. |
| 5 | ABORT | **Aborted Transmission Bit. (slave mode only)**<br>0: The current transmission has not been interrupted or stopped. This bit is reset to 0 after receiving a SYNCH BREAK that does not interrupt a pending transmission.<br>1: New SYNCH BREAK detected before the end of the last transmission or the STOP bit (LIN0CTRL.7) has been set. |
| 4 | DTREQ | **Data Request Bit. (slave mode only)**<br>0: Data identifier has not been received.<br>1: Data identifier has been received. |
| 3 | LININT | **Interrupt Request Bit.**<br>0: An interrupt is not pending. This bit is cleared by setting RSTINT (LIN0CTRL.3)<br>1: There is a pending LIN0 interrupt. |
| 2 | ERROR | **Communication Error Bit.**<br>0: No error has been detected. This bit is cleared by setting RSTERR (LIN0CTRL.2)<br>1: An error has been detected. |
| 1 | WAKEUP | **Wakeup Bit.**<br>0: A wakeup signal is not being transmitted and has not been received.<br>1: A wakeup signal is being transmitted or has been received |
| 0 | DONE | **Transmission Complete Bit.**<br>0: A transmission is not in progress or has not been started. This bit is cleared at the start of a transmission.<br>1: The current transmission is complete. |



Downloaded from Arrow.com.

## LIN Register Definition 19.7. LIN0ERR: LIN0 Error Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | SYNCH | PRTY | TOUT | CHK | BITERR |
| Type | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Indirect Address = 0x0A

| Bit | Name | Function |
|---|---|---|
| 7:5 | Unused | Read = 000b; Write = Don't Care |
| 4 | SYNCH | **Synchronization Error Bit (slave mode only).**<br>0: No error with the SYNCH FIELD has been detected.<br>1: Edges of the SYNCH FIELD are outside of the maximum tolerance. |
| 3 | PRTY | **Parity Error Bit (slave mode only).**<br>0: No parity error has been detected.<br>1: A parity error has been detected. |
| 2 | TOUT | **Timeout Error Bit.**<br>0: A timeout error has not been detected.<br>1: A timeout error has been detected. This error is detected whenever one of the following conditions is met:<br>• The master is expecting data from a slave and the slave does not respond.<br>• The slave is expecting data but no data is transmitted on the bus.<br>• A frame is not finished within the maximum frame length.<br>• The application does not set the DTACK bit (LIN0CTRL.4) or STOP bit (LIN0CTRL.7) until the end of the reception of the first byte after the identifier. |
| 1 | CHK | **Checksum Error Bit.**<br>0: Checksum error has not been detected.<br>1: Checksum error has been detected. |
| 0 | BITERR | **Bit Transmission Error Bit.**<br>0: No error in transmission has been detected.<br>1: The bit value monitored during transmission is different than the bit value sent. |



Downloaded from Arrow.com.

## LIN Register Definition 19.8. LIN0SIZE: LIN0 Message Size Register

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

Indirect Address = 0x0B

| Bit | Name | Function |
|---|---|---|
| 7 | ENHCHK | **Checksum Selection Bit.**<br>0: Use the classic, specification 1.3 compliant checksum. Checksum covers the data bytes.<br>1: Use the enhanced, specification 2.0 compliant checksum. Checksum covers data bytes and protected identifier. |
| 6:4 | Unused | Read = 000b; Write = Don't Care |
| 3:0 | LINSIZE[3:0] | **Data Field Size.**<br>0000: 0 data bytes<br>0001: 1 data byte<br>0010: 2 data bytes<br>0011: 3 data bytes<br>0100: 4 data bytes<br>0101: 5 data bytes<br>0110: 6 data bytes<br>0111: 7 data bytes<br>1000: 8 data bytes<br>1001-1110: RESERVED<br>1111: Use the ID[1:0] bits (LIN0ID[5:4]) to determine the data length. |



Downloaded from Arrow.com.

## LIN Register Definition 19.9. LIN0DIV: LIN0 Divider Register

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

Indirect Address = 0x0C

| Bit | Name | Function |
|---|---|---|
| 7:0 | DIVLSB | **LIN Baud Rate Divider Least Significant Bits.**<br>The 8 least significant bits for the baud rate divider. The 9th and most significant bit is the DIV9 bit (LIN0MUL.0). The valid range for the divider is 200 to 511. |

## LIN Register Definition 19.10. LIN0MUL: LIN0 Multiplier Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PRESCL[1:0] | | LINMUL[4:0] | | | | | DIV9 |
| Type | R/W | | R/W | | | | | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Indirect Address = 0x0D

| Bit | Name | Function |
|---|---|---|
| 7:6 | PRESCL[1:0] | **LIN Baud Rate Prescaler Bits.**<br>These bits are the baud rate prescaler bits. |
| 5:1 | LINMUL[4:0] | **LIN Baud Rate Multiplier Bits.**<br>These bits are the baud rate multiplier bits. These bits are not used in slave mode. |
| 0 | DIV9 | **LIN Baud Rate Divider Most Significant Bit.**<br>The most significant bit of the baud rate divider. The 8 least significant bits are in LIN0DIV. The valid range for the divider is 200 to 511. |



Downloaded from Arrow.com.

## LIN Register Definition 19.11. LIN0ID: LIN0 Identifier Register

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

Indirect Address = 0x0E

| Bit | Name | Function |
|---|---|---|
| 7:6 | Unused | Read = 00b; Write = Don't Care. |
| 5:0 | ID[5:0] | **LIN Identifier Bits.**<br>These bits form the data identifier.<br><br>If the LINSIZE bits (LIN0SIZE[3:0]) are 1111b, bits ID[5:4] are used to determine the data size and are interpreted as follows:<br>00: 2 bytes<br>01: 2 bytes<br>10: 4 bytes<br>11: 8 bytes |



Downloaded from Arrow.com.

## 20. 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 I2C 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. A block diagram of the SMBus peripheral and the associated SFRs is shown in Figure 20.1.

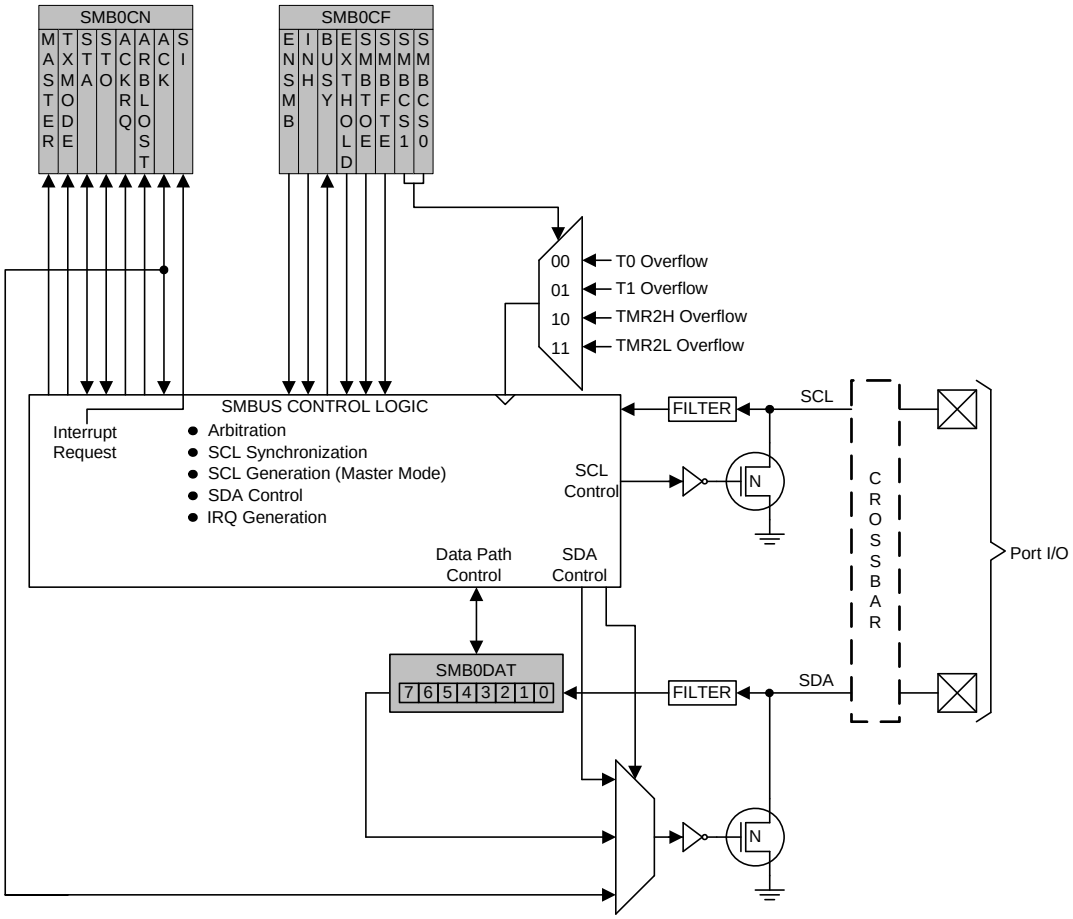



**Figure 20.1. SMBus Block Diagram**



Downloaded from Arrow.com.

## 20.1. Supporting Documents

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

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

## 20.2. SMBus Configuration

Figure 20.2 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.

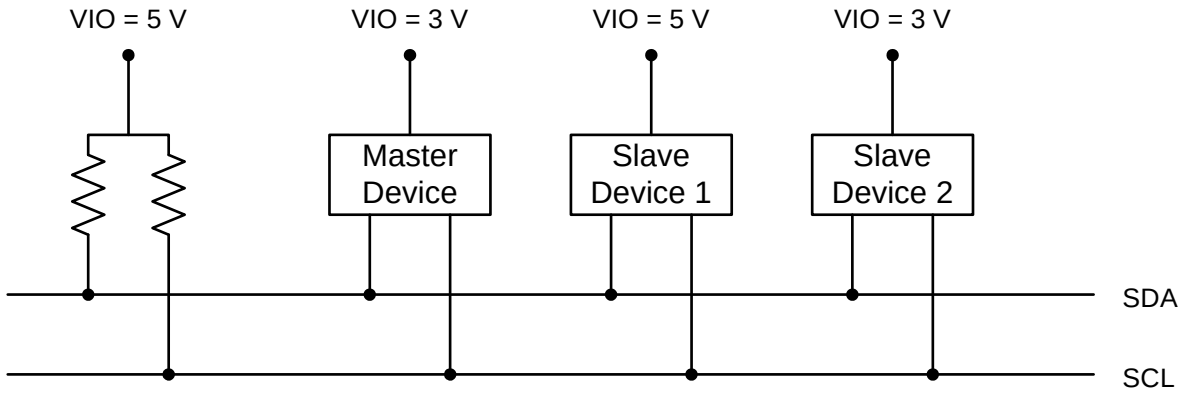



**Figure 20.2. Typical SMBus Configuration**

## 20.3. 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. 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 20.3). 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.



Downloaded from Arrow.com.

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 20.3 illustrates a typical SMBus transaction.

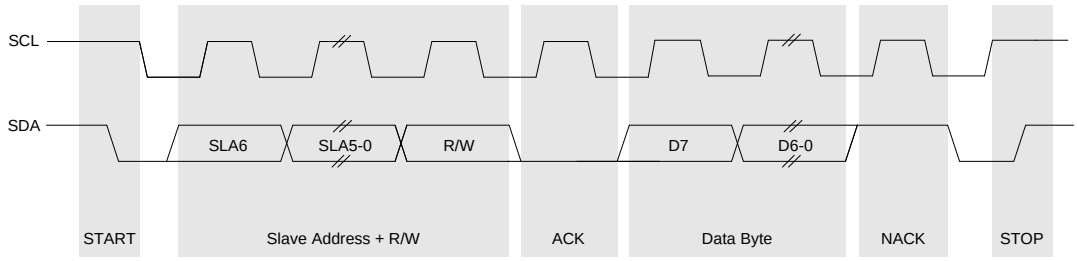



**Figure 20.3. SMBus Transaction**

### 20.3.1. 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.

### 20.3.2. 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 “20.3.5. SCL High (SMBus Free) Timeout” on page 190). 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.

### 20.3.3. Clock Low Extension

SMBus provides a clock synchronization mechanism, similar to $I^2C$, 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.

### 20.3.4. 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



Downloaded from Arrow.com.

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.

### 20.3.5. 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. Note that a clock source is required for free timeout detection, even in a slave-only implementation.

## 20.4. 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

SMBus interrupts are generated for each data byte or slave address that is transferred. 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. See Section 20.5 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 20.4.2; Table 20.4 provides a quick SMB0CN decoding reference.

### 20.4.1. 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).



Downloaded from Arrow.com.

**Table 20.1. 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 20.1. 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 “23. Timers” on page 227.

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

**Equation 20.1. 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 20.1. 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 20.2.

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

**Equation 20.2. Typical SMBus Bit Rate**

Figure 20.4 shows the typical SCL generation described by Equation 20.2. 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 20.1.

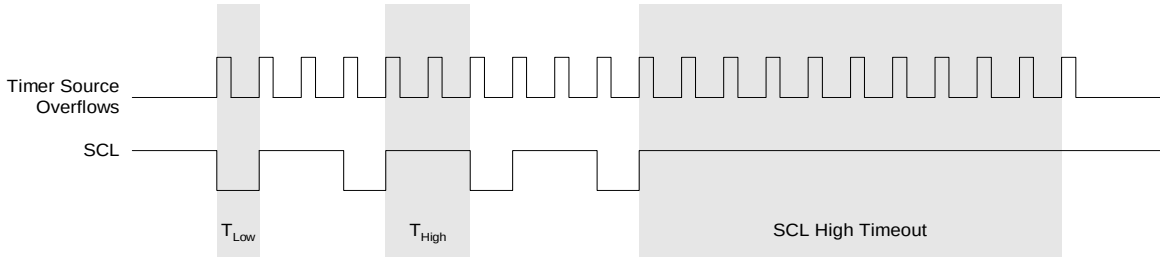



**Figure 20.4. Typical SMBus SCL Generation**

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



Downloaded from Arrow.com.

meet the SMBus Specification requirements of 250 ns and 300 ns, respectively. Table 20.2 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 20.2. 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 “20.3.4. SCL Low Timeout” on page 189). 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 20.4).



Downloaded from Arrow.com.

## SFR Definition 20.1. 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; SFR Page = 0x00

| 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 20.2.<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 20.1.<br>00: Timer 0 Overflow<br>01: Timer 1 Overflow<br>10:Timer 2 High Byte Overflow<br>11: Timer 2 Low Byte Overflow |



Downloaded from Arrow.com.

### 20.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see SFR Definition 20.2). 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.

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.

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 20.3 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.



Downloaded from Arrow.com.

## SFR Definition 20.2. 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; SFR Page =0x00

| 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.**<br>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. |



Downloaded from Arrow.com.

**Table 20.3. 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. | ■ 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. |



Downloaded from Arrow.com.

### 20.4.3. 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 20.3. 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; SMB0DAT = 0x00

| 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. |

## 20.5. 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. As a receiver, the interrupt for an ACK occurs **before** the ACK. As a transmitter, interrupts occur **after** the ACK.



Downloaded from Arrow.com.

### 20.5.1. 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 20.5 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.

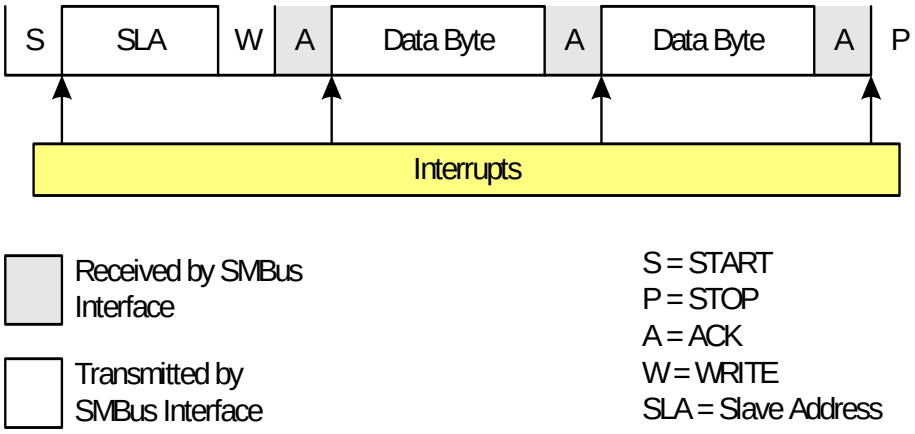



**Figure 20.5. Typical Master Write Sequence**



Downloaded from Arrow.com.

### 20.5.2. 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. An interrupt is generated after each received byte.

Software must write the ACK bit at that time to ACK or NACK the received byte. 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 20.6 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 **before** the ACK cycle in this mode.

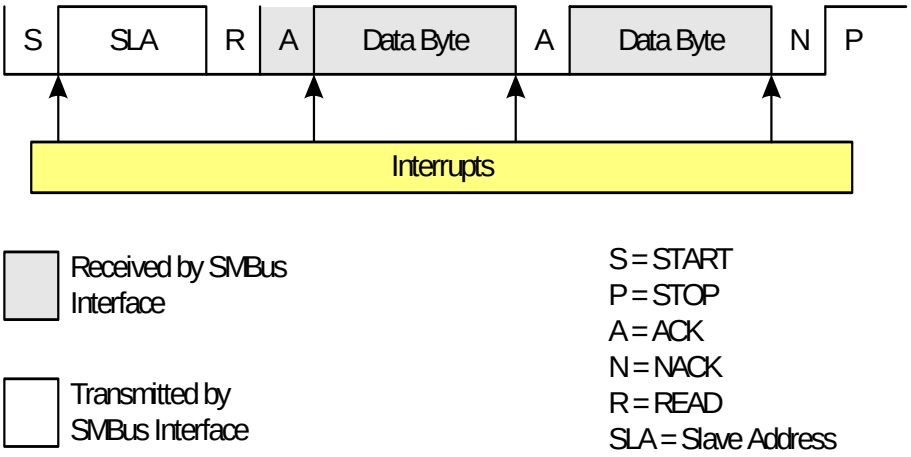



**Figure 20.6. Typical Master Read Sequence**



Downloaded from Arrow.com.

### 20.5.3. 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. 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 the received slave address is ignored, 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. Software must write the ACK bit at that time to ACK or NACK the received byte.

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 20.7 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 **before** the ACK in this mode.

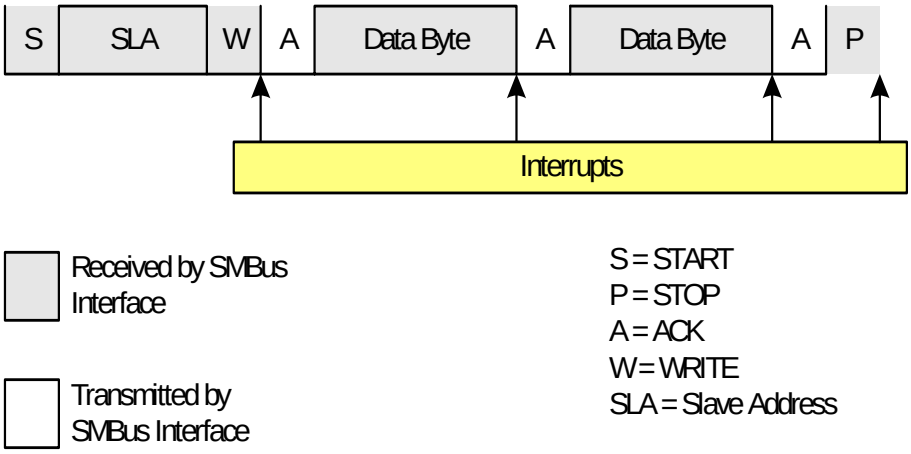



**Figure 20.7. Typical Slave Write Sequence**



Downloaded from Arrow.com.

### 20.5.4. 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. 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. The interrupt will occur after the ACK cycle.

If the received slave address is ignored, 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 (Note: 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 20.8 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.

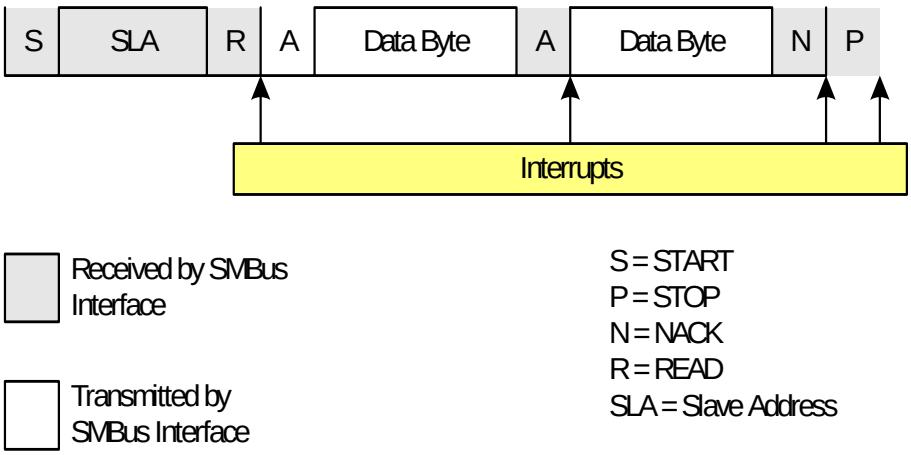



**Figure 20.8. Typical Slave Read Sequence**

## 20.6. SMBus Status Decoding

The current SMBus status can be easily decoded using the SMB0CN register. 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.



Downloaded from Arrow.com.

**Table 20.4. SMBus Status Decoding**

| 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 |



Downloaded from Arrow.com.

**Table 20.4. SMBus Status Decoding**

| 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 | — |



Downloaded from Arrow.com.

**Table 20.4. SMBus Status Decoding**

| Mode | Values Read | | | | Current SMbus State | Typical Response Options | Values to Write | | | Next Status Vector Expected |
|---|---|---|---|---|---|---|---|---|---|---|
| | Status Vector | ACKRQ | ARBLOST | ACK | | | STA | STO | ACK | |
| 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 |



Downloaded from Arrow.com.

# 21. UART0

UART0 is an asynchronous, full duplex serial port offering a variety of data formatting options. A dedicated baud rate generator with a 16-bit timer and selectable prescaler is included, which can generate a wide range of baud rates (details in Section “21.1. Baud Rate Generator” on page 205). A received data FIFO allows UART0 to receive up to three data bytes before data is lost and an overflow occurs.

UART0 has six associated SFRs. Three are used for the Baud Rate Generator (SBCON0, SBRLH0, and SBRLL0), two are used for data formatting, control, and status functions (SCON0, SMOD0), and one is used to send and receive data (SBUF0). The single SBUF0 location provides access to both the transmit holding register and the receive FIFO. **Writes to SBUF0 always access the Transmit register. Reads of SBUF0 always access the first byte of the Receive FIFO; it is not possible to read data from the Transmit Holding 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). If additional bytes are available in the Receive FIFO, the RI0 bit cannot be cleared by software.

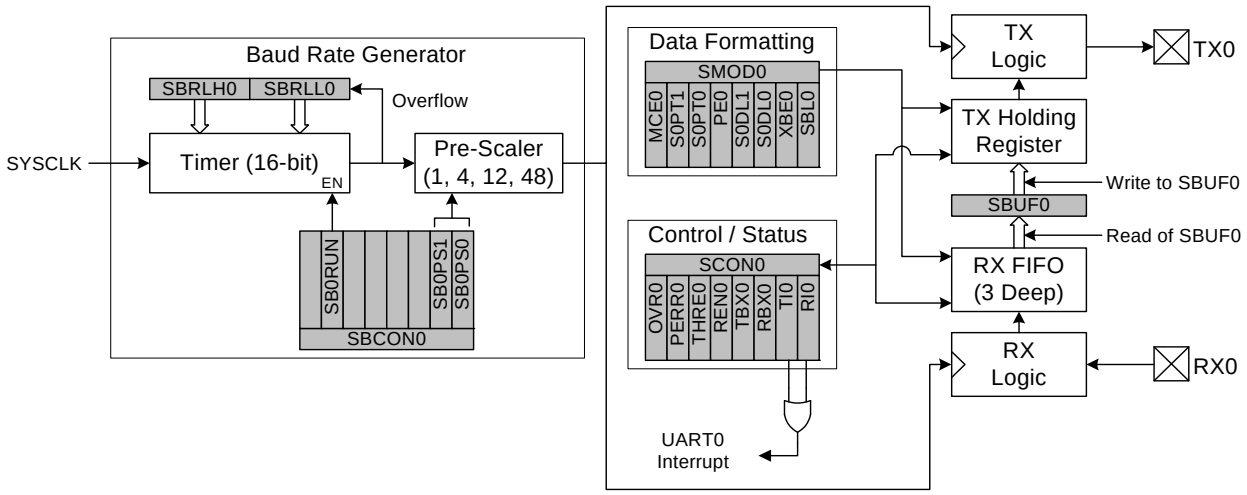



**Figure 21.1. UART0 Block Diagram**

## 21.1. Baud Rate Generator

The UART0 baud rate is generated by a dedicated 16-bit timer which runs from the controller’s core clock (SYSCLK) and has prescaler options of 1, 4, 12, or 48. The timer and prescaler options combined allow for a wide selection of baud rates over many clock frequencies.

The baud rate generator is configured using three registers: SBCON0, SBRLH0, and SBRLL0. The UART0 Baud Rate Generator Control Register (SBCON0, SFR Definition 21.4) enables or disables the baud rate generator and selects the prescaler value for the timer. The baud rate generator must be enabled for UART0 to function. Registers SBRLH0 and SBRLL0 contain a 16-bit reload value for the dedicated 16-bit timer. The internal timer counts up from the reload value on every clock tick. On timer overflows (0xFFFF to 0x0000), the timer is reloaded. The baud rate for UART0 is defined in Equation 21.1, where “BRG Clock” is the baud rate generator’s selected clock source. For reliable UART operation, it is recommended that the UART baud rate is not configured for baud rates faster than SYSCLK/16.



Downloaded from Arrow.com.

$$\text{Baud Rate} = \frac{\text{SYSCLK}}{(65536 - (\text{SBRLH0:SBRLL0}))} \times \frac{1}{2} \times \frac{1}{\text{Prescaler}}$$

**Equation 21.1. UART0 Baud Rate**

A quick reference for typical baud rates and clock frequencies is given in Table 21.1.

**Table 21.1. Baud Rate Generator Settings for Standard Baud Rates**

| | Target Baud Rate (bps) | Actual Baud Rate (bps) | Baud Rate Error | Oscillator Divide Factor | SB0PS[1:0] (Prescaler Bits) | Reload Value in SBRLH0:SBRLL0 |
|---|---|---|---|---|---|---|
| SYSCLK = 48 | 230400 | 230769 | 0.16% | 208 | 11 | 0xFF98 |
| | 115200 | 115385 | 0.16% | 416 | 11 | 0xFF30 |
| | 57600 | 57554 | 0.08% | 834 | 11 | 0xFE5F |
| | 28800 | 28812 | 0.04% | 1666 | 11 | 0xFCBF |
| | 14400 | 14397 | 0.02% | 3334 | 11 | 0xF97D |
| | 9600 | 9600 | 0.00% | 5000 | 11 | 0xF63C |
| | 2400 | 2400 | 0.00% | 20000 | 11 | 0xD8F0 |
| | 1200 | 1200 | 0.00% | 40000 | 11 | 0xB1E0 |
| SYSCLK = 24 | 230400 | 230769 | 0.16% | 104 | 11 | 0xFFCC |
| | 115200 | 115385 | 0.16% | 208 | 11 | 0xFF98 |
| | 57600 | 57692 | 0.16% | 416 | 11 | 0xFF30 |
| | 28800 | 28777 | 0.08% | 834 | 11 | 0xFE5F |
| | 14400 | 14406 | 0.04% | 1666 | 11 | 0xFCBF |
| | 9600 | 9600 | 0.00% | 2500 | 11 | 0xFB1E |
| | 2400 | 2400 | 0.00% | 10000 | 11 | 0xEC78 |
| | 1200 | 1200 | 0.00% | 20000 | 11 | 0xD8F0 |
| SYSCLK = 12 | 230400 | 230769 | 0.16% | 52 | 11 | 0xFFE6 |
| | 115200 | 115385 | 0.16% | 104 | 11 | 0xFFCC |
| | 57600 | 57692 | 0.16% | 208 | 11 | 0xFF98 |
| | 28800 | 28846 | 0.16% | 416 | 11 | 0xFF30 |
| | 14400 | 14388 | 0.08% | 834 | 11 | 0xFE5F |
| | 9600 | 9600 | 0.00% | 1250 | 11 | 0xFD8F |
| | 2400 | 2400 | 0.00% | 5000 | 11 | 0xF63C |
| | 1200 | 1200 | 0.00% | 10000 | 11 | 0xEC78 |



Downloaded from Arrow.com.

## 21.2. Data Format

UART0 has a number of available options for data formatting. Data transfers begin with a start bit (logic low), followed by the data bits (sent LSB-first), a parity or extra bit (if selected), and end with one or two stop bits (logic high). The data length is variable between 5 and 8 bits. A parity bit can be appended to the data, and automatically generated and detected by hardware for even, odd, mark, or space parity. The stop bit length is selectable between 1 and 2 bit times, and a multi-processor communication mode is available for implementing networked UART buses. All of the data formatting options can be configured using the SMOD0 register, shown in SFR Definition 21.2. Figure 21.2 shows the timing for a UART0 transaction without parity or an extra bit enabled. Figure 21.3 shows the timing for a UART0 transaction with parity enabled (PE0 = 1). Figure 21.4 is an example of a UART0 transaction when the extra bit is enabled (XBE0 = 1). Note that the extra bit feature is not available when parity is enabled, and the second stop bit is only an option for data lengths of 6, 7, or 8 bits.

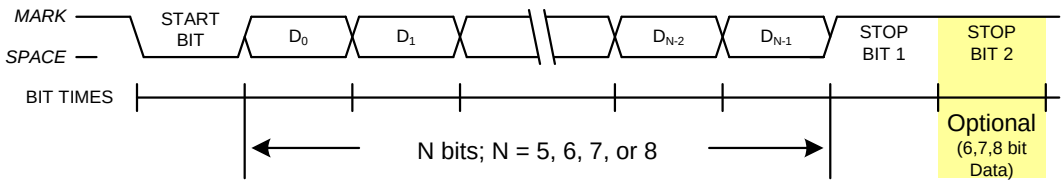



**Figure 21.2. UART0 Timing Without Parity or Extra Bit**

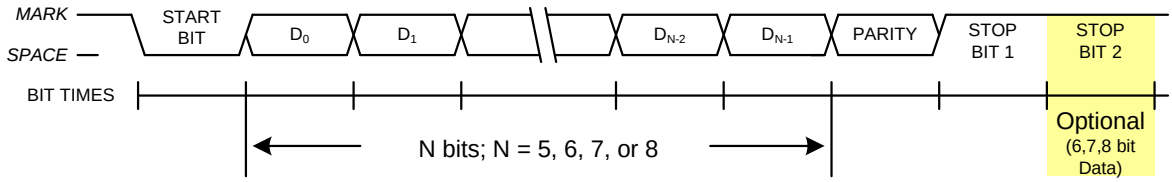



**Figure 21.3. UART0 Timing With Parity**

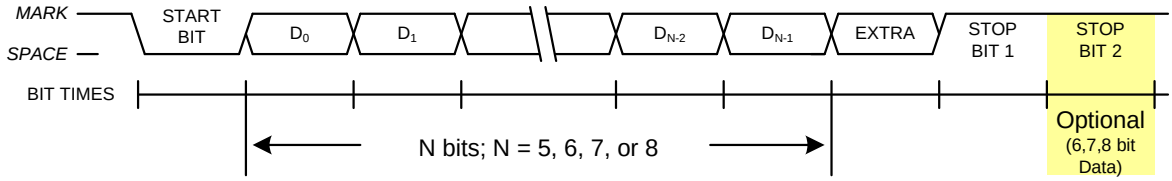



**Figure 21.4. UART0 Timing With Extra Bit**



Downloaded from Arrow.com.

## 21.3. Configuration and Operation

UART0 provides standard asynchronous, full duplex communication. It can operate in a point-to-point serial communications application, or as a node on a multi-processor serial interface. To operate in a point-to-point application, where there are only two devices on the serial bus, the MCE0 bit in SMOD0 should be cleared to 0. For operation as part of a multi-processor communications bus, the MCE0 and XBE0 bits should both be set to 1. In both types of applications, data is transmitted from the microcontroller on the TX0 pin, and received on the RX0 pin. The TX0 and RX0 pins are configured using the crossbar and the Port I/O registers, as detailed in Section “18. Port Input/Output” on page 147.

In typical UART communications, The transmit (TX) output of one device is connected to the receive (RX) input of the other device, either directly or through a bus transceiver, as shown in Figure 21.5.

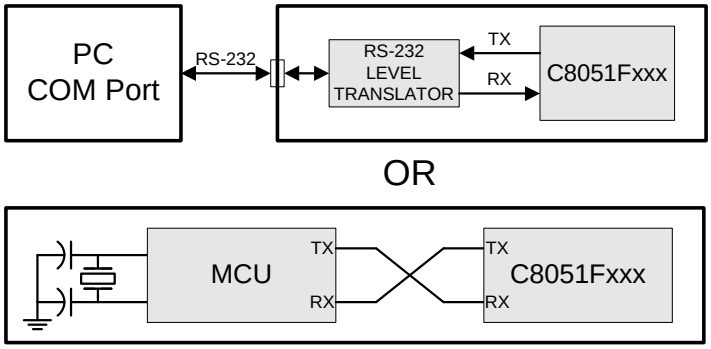



**Figure 21.5. Typical UART Interconnect Diagram**

### 21.3.1. Data Transmission

Data transmission is double-buffered and begins when software writes a data byte to the SBUF0 register. Writing to SBUF0 places data in the Transmit Holding Register, and the Transmit Holding Register Empty flag (THRE0) will be cleared to ‘0’. If the UART’s shift register is empty (i.e. no transmission in progress), the data will be placed in the Transmit Holding Register until the current transmission is complete. The TI0 Transmit Interrupt Flag (SCON0.1) will be set at the end of any transmission (the beginning of the stop-bit time). If enabled, an interrupt will occur when TI0 is set.

If the extra bit function is enabled (XBE0 = ‘1’) and the parity function is disabled (PE0 = ‘0’), the value of the TBX0 (SCON0.3) bit will be sent in the extra bit position. When the parity function is enabled (PE0 = 1), hardware will generate the parity bit according to the selected parity type (selected with S0PT[1:0]), and append it to the data field. Note: when parity is enabled, the extra bit function is not available.

### 21.3.2. Data Reception

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 stored in the receive FIFO if the following conditions are met: the receive FIFO (3 bytes deep) must not be full, and the stop bit(s) must be logic 1. In the event that the receive FIFO is full, the incoming byte will be lost, and a Receive FIFO Overrun Error will be generated (OVR0 in register SCON0 will be set to logic 1). If the stop bit(s) were logic 0, the incoming data will not be stored in the receive FIFO. If the reception conditions are met, the data is stored in the receive FIFO, and the RI0 flag will be set. Note: when MCE0 = 1, RI0 will only be set if the extra bit was equal to 1. Data can be read from the receive FIFO by reading the SBUF0 register. The SBUF0 register represents the oldest byte in the FIFO. After SBUF0 is read, the next byte in the FIFO is immediately loaded into SBUF0, and space is made available in the FIFO for another incoming byte. If enabled, an interrupt will occur when RI0 is set. RI0 can only be cleared to ‘0’ by software when there is no more information in the FIFO. The recommended procedure to empty the FIFO contents is:



Downloaded from Arrow.com.

1. Clear RI0 to 0.
2. Read SBUF0.
3. Check RI0, and repeat at step 1 if RI0 is set to 1.

If the extra bit function is enabled (XBE0 = 1) and the parity function is disabled (PE0 = 0), the extra bit for the oldest byte in the FIFO can be read from the RBX0 bit (SCON0.2). If the extra bit function is not enabled, the value of the stop bit for the oldest FIFO byte will be presented in RBX0. When the parity function is enabled (PE0 = 1), hardware will check the received parity bit against the selected parity type (selected with S0PT[1:0]) when receiving data. If a byte with parity error is received, the PERR0 flag will be set to 1. This flag must be cleared by software. Note: when parity is enabled, the extra bit function is not available.

### 21.3.3. Multiprocessor Communications

UART0 supports multiprocessor communication between a master processor and one or more slave processors by special use of the extra 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 extra bit is logic 1; in a data byte, the extra bit is always set to logic 0.

Setting the MCE0 bit (SMOD0.7) of a slave processor configures its UART such that when a stop bit is received, the UART will generate an interrupt only if the extra bit is logic 1 (RBX0 = 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 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).

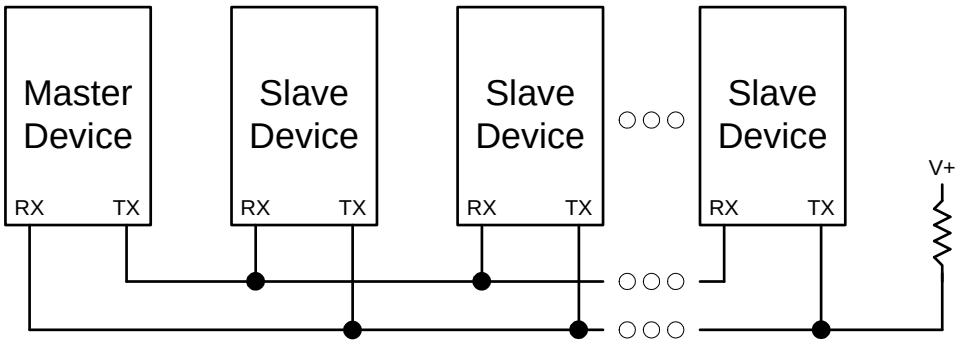



**Figure 21.6. UART Multi-Processor Mode Interconnect Diagram**



Downloaded from Arrow.com.

## SFR Definition 21.1. SCON0: Serial Port 0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OVR0 | PERR0 | THRE0 | REN0 | TBX0 | RBX0 | TI0 | RI0 |
| Type | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x98; Bit-Addressable; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7 | OVR0 | **Receive FIFO Overrun Flag.**<br>0: Receive FIFO Overrun has not occurred<br>1: Receive FIFO Overrun has occurred; A received character has been discarded due to a full FIFO. |
| 6 | PERR0 | **Parity Error Flag.**<br>When parity is enabled, this bit indicates that a parity error has occurred. It is set to 1 when the parity of the oldest byte in the FIFO does not match the selected Parity Type.<br>0: Parity error has not occurred<br>1: Parity error has occurred.<br>This bit must be cleared by software. |
| 5 | THRE0 | **Transmit Holding Register Empty Flag.**<br>0: Transmit Holding Register not Empty—do not write to SBUF0.<br>1: Transmit Holding Register Empty—it is safe to write to SBUF0. |
| 4 | REN0 | **Receive Enable.**<br>This bit enables/disables the UART receiver. When disabled, bytes can still be read from the receive FIFO.<br>0: UART1 reception disabled.<br>1: UART1 reception enabled. |
| 3 | TBX0 | **Extra Transmission Bit.**<br>The logic level of this bit will be assigned to the extra transmission bit when XBE0 is set to 1. This bit is not used when Parity is enabled. |
| 2 | RBX0 | **Extra Receive Bit.**<br>RBX0 is assigned the value of the extra bit when XBE1 is set to 1. If XBE1 is cleared to 0, RBX1 will be assigned the logic level of the first stop bit. This bit is not valid when Parity is enabled. |
| 1 | TI0 | **Transmit Interrupt Flag.**<br>Set to a 1 by hardware after data has been transmitted, at the beginning of the STOP bit. 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 soft-ware. Note that RI0 will remain set to ‘1’ as long as there is data still in the UART FIFO. After the last byte has been shifted from the FIFO to SBUF0, RI0 can be cleared. |



Downloaded from Arrow.com.

## SFR Definition 21.2. SMOD0: Serial Port 0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MCE0 | S0PT[1:0] | | PE0 | S0DL[1:0] | | XBE0 | SBL0 |
| Type | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

SFR Address = 0xA9; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7 | MCE0 | **Multiprocessor Communication Enable.**<br>0: RI0 will be activated if stop bit(s) are 1.<br>1: RI0 will be activated if stop bit(s) and extra bit are 1. Extra bit must be enabled using XBE0. |
| 6:5 | S0PT[1:0] | **Parity Type Select Bits.**<br>00: Odd Parity<br>01: Even Parity<br>10: Mark Parity<br>11: Space Parity. |
| 4 | PE0 | **Parity Enable.**<br>This bit enables hardware parity generation and checking. The parity type is selected by bits S0PT[1:0] when parity is enabled.<br>0: Hardware parity is disabled.<br>1: Hardware parity is enabled. |
| 3:2 | S0DL[1:0] | **Data Length.**<br>00: 5-bit data<br>01: 6-bit data<br>10: 7-bit data<br>11: 8-bit data |
| 1 | XBE0 | **Extra Bit Enable.**<br>When enabled, the value of TBX0 will be appended to the data field<br>0: Extra Bit is disabled.<br>1: Extra Bit is enabled. |
| 0 | SBL0 | **Stop Bit Length.**<br>0: Short—stop bit is active for one bit time<br>1: Long—stop bit is active for two bit times (data length = 6, 7, or 8 bits), or 1.5 bit times (data length = 5 bits). |



Downloaded from Arrow.com.

## SFR Definition 21.3. 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; SFR Page = 0x00

| 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. |

## SFR Definition 21.4. SBCON0: UART0 Baud Rate Generator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | SB0RUN | Reserved | Reserved | Reserved | Reserved | SB0PS[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 = 0xAB; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7 | Reserved | Read = 0b; Must Write 0b; |
| 6 | SB0RUN | **Baud Rate Generator Enable.**<br>0: Baud Rate Generator disabled. UART0 will not function.<br>1: Baud Rate Generator enabled. |
| 5:2 | Reserved | Read = 0000b; Must Write = 0000b; |
| 1:0 | SB0PS[1:0] | **Baud Rate Prescaler Select.**<br>00: Prescaler = 12.<br>01: Prescaler = 4.<br>10: Prescaler = 48.<br>11: Prescaler = 1. |



Downloaded from Arrow.com.

## SFR Definition 21.5. SBRLH0: UART0 Baud Rate Generator Reload High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SBRLH0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xAD; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | SBRLH0[7:0] | **High Byte of Reload Value for UART0 Baud Rate Generator.**<br>This value is loaded into the high byte of the UART0 baud rate generator when the counter overflows from 0xFFFF to 0x0000. |

## SFR Definition 21.6. SBRLL0: UART0 Baud Rate Generator Reload Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SBRLL0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xAC; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | SBRLL0[7:0] | **Low Byte of Reload Value for UART0 Baud Rate Generator.**<br>This value is loaded into the low byte of the UART0 baud rate generator when the counter overflows from 0xFFFF to 0x0000. |



Downloaded from Arrow.com.

## 22. 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.

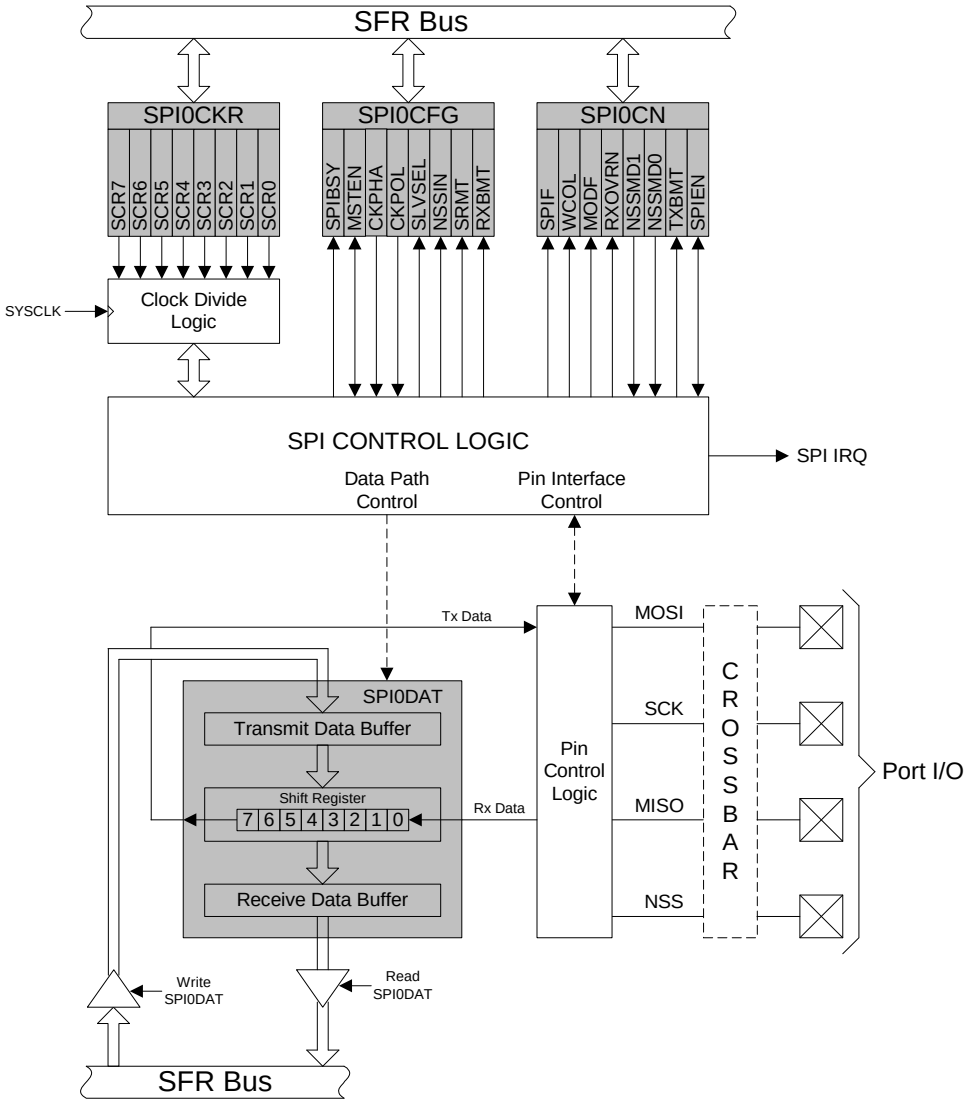



**Figure 22.1. SPI Block Diagram**



Downloaded from Arrow.com.

## 22.1. Signal Descriptions

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

### 22.1.1. 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.

### 22.1.2. 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.

### 22.1.3. 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.

### 22.1.4. 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 22.2, Figure 22.3, and Figure 22.4 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 “18. Port Input/Output” on page 147 for general purpose port I/O and crossbar information.



Downloaded from Arrow.com.

## 22.2. 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 22.2 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 22.3 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 22.4 shows a connection diagram for a master device in 4-wire master mode and two slave devices.



Downloaded from Arrow.com.

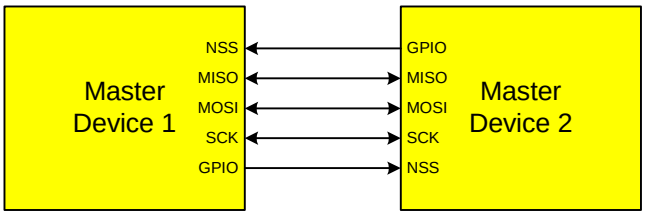



**Figure 22.2. Multiple-Master Mode Connection Diagram**

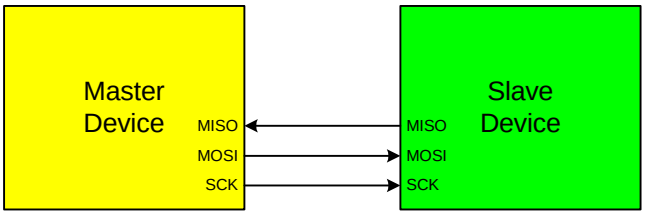



**Figure 22.3. 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram**

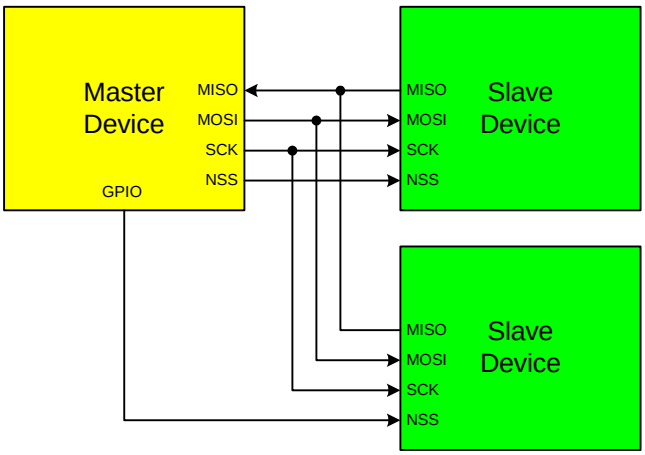



**Figure 22.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram**



Downloaded from Arrow.com.

## 22.3. 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 22.4 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 22.3 shows a connection diagram between a slave device in 3-wire slave mode and a master device.

## 22.4. 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.

1. 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.
2. 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.
3. 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.
4. 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.



Downloaded from Arrow.com.

## 22.5. 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 22.5. For slave mode, the clock and data relationships are shown in Figure 22.6 and Figure 22.7. CKPHA must be set to 0 on both the master and slave SPI when communicating between two of the following devices: C8051F04x, C8051F06x, C8051F12x, C8051F31x, C8051F32x, and C8051F33x.

The SPI0 Clock Rate Register (SPI0CKR) as shown in SFR Definition 22.3 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.

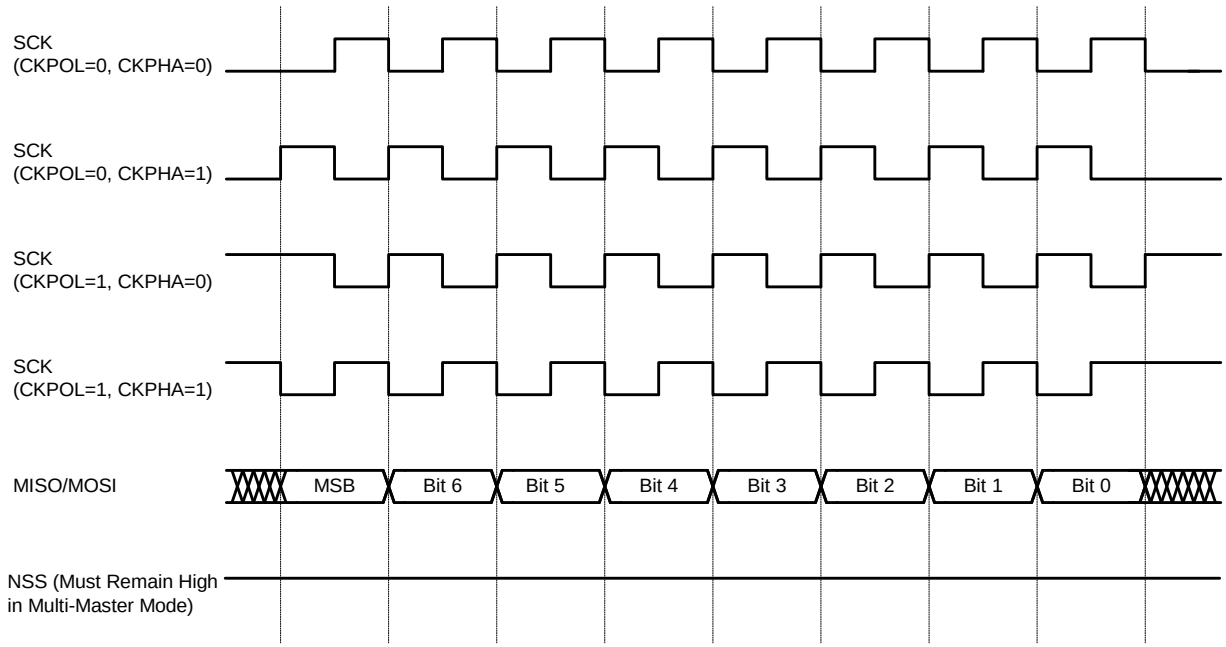



**Figure 22.5. Master Mode Data/Clock Timing**



Downloaded from Arrow.com.

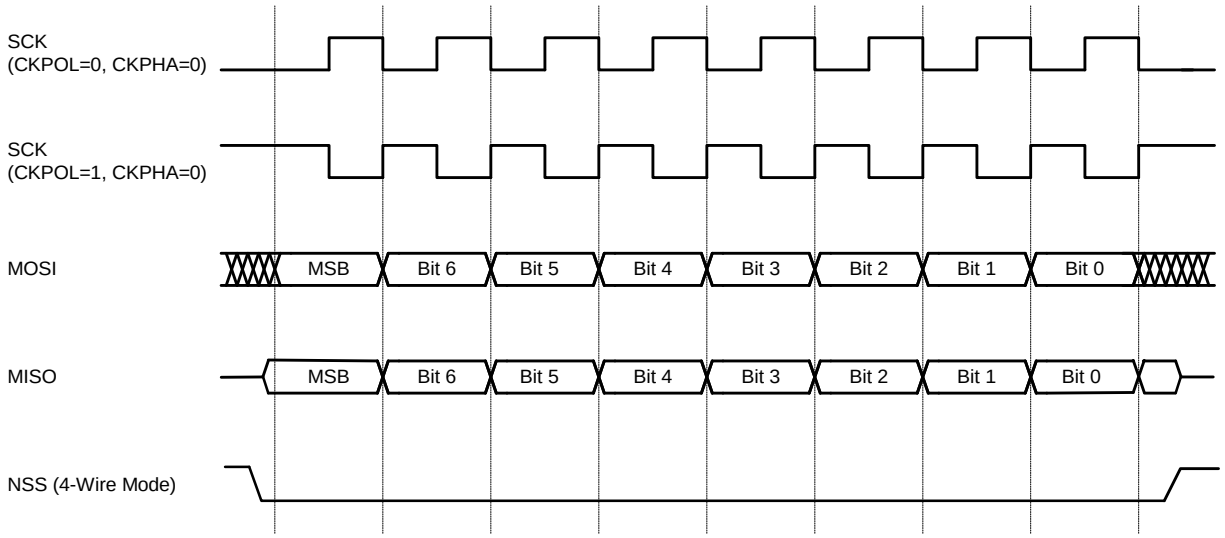



**Figure 22.6. Slave Mode Data/Clock Timing (CKPHA = 0)**

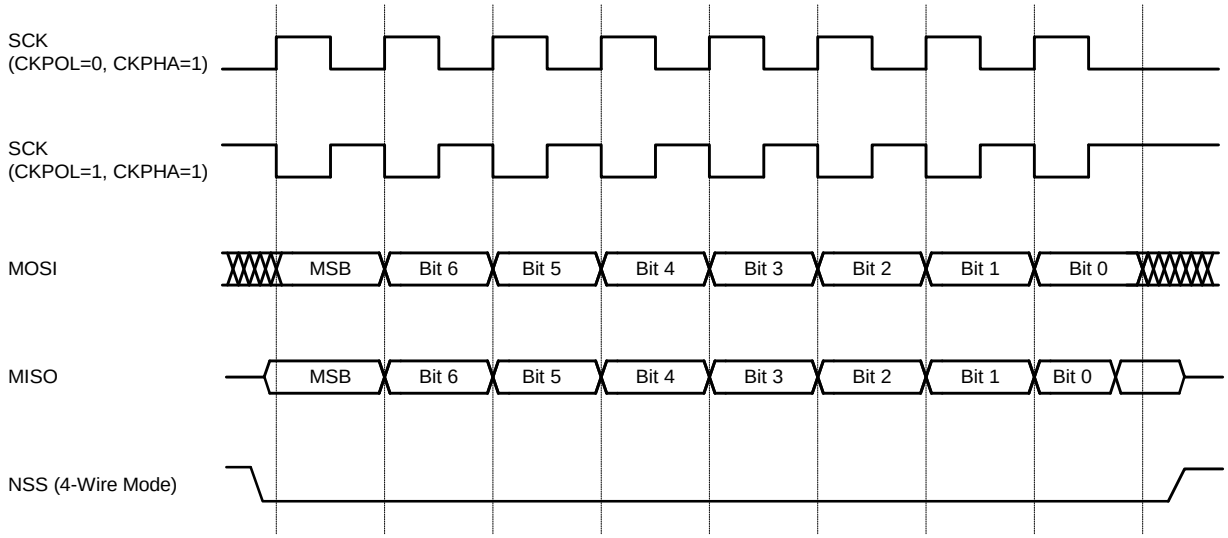



**Figure 22.7. Slave Mode Data/Clock Timing (CKPHA = 1)**

## 22.6. 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.



Downloaded from Arrow.com.

## SFR Definition 22.1. 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; SFR Page = 0x00

| 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 22.1 for timing parameters.



Downloaded from Arrow.com.

## SFR Definition 22.2. 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; SFR Page = 0x00

| 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 interrupts are enabled, setting this bit causes the CPU to vector to the SPI0 interrupt service routine. This bit is not automatically cleared by hardware. It must be cleared by software. |
| 6 | WCOL | **Write Collision Flag.**<br>This bit is set to logic 1 by hardware (and generates a SPI0 interrupt) to indicate a write to the SPI0 data register was attempted while a data transfer was in progress. It must be cleared by software. |
| 5 | MODF | **Mode Fault Flag.**<br>This bit is set to logic 1 by hardware (and generates a SPI0 interrupt) when a master mode collision is detected (NSS is low, MSTEN = 1, and NSSMD[1:0] = 01). This bit is not automatically cleared by hardware. It 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 (and generates a SPI0 interrupt) 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. This bit is not automatically cleared by hardware. It must be cleared by software. |
| 3:2 | NSSMD[1:0] | **Slave Select Mode.**<br>Selects between the following NSS operation modes:<br>(See Section 22.2 and Section 22.3).<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. |



Downloaded from Arrow.com.

## SFR Definition 22.3. 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; SFR Page = 0x00

| 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>$f_{SCK} = 200\ kHz$ |

## SFR Definition 22.4. SPI0DAT: SPI0 Data

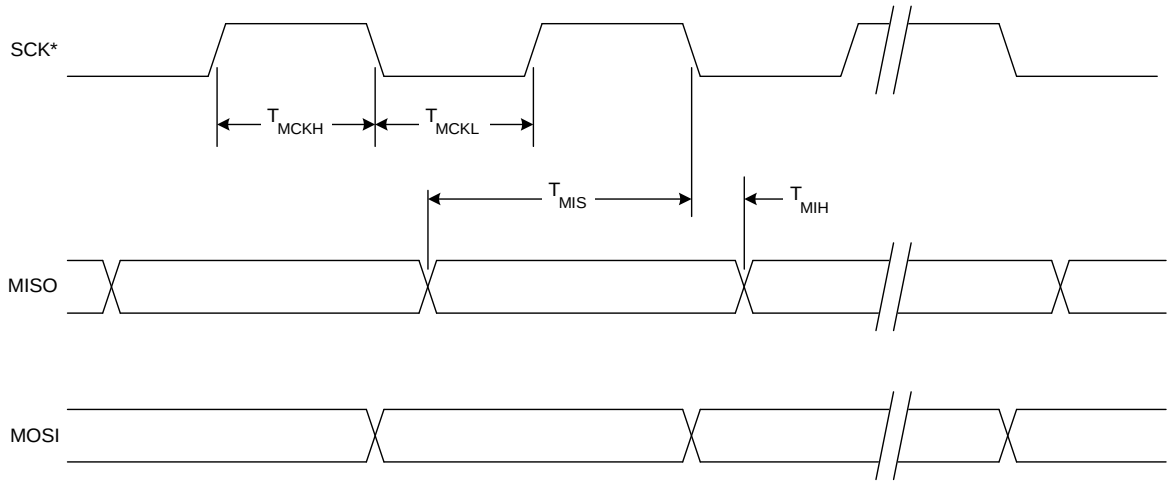
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPI0DAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA3; SFR Page = 0x00

| 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. |

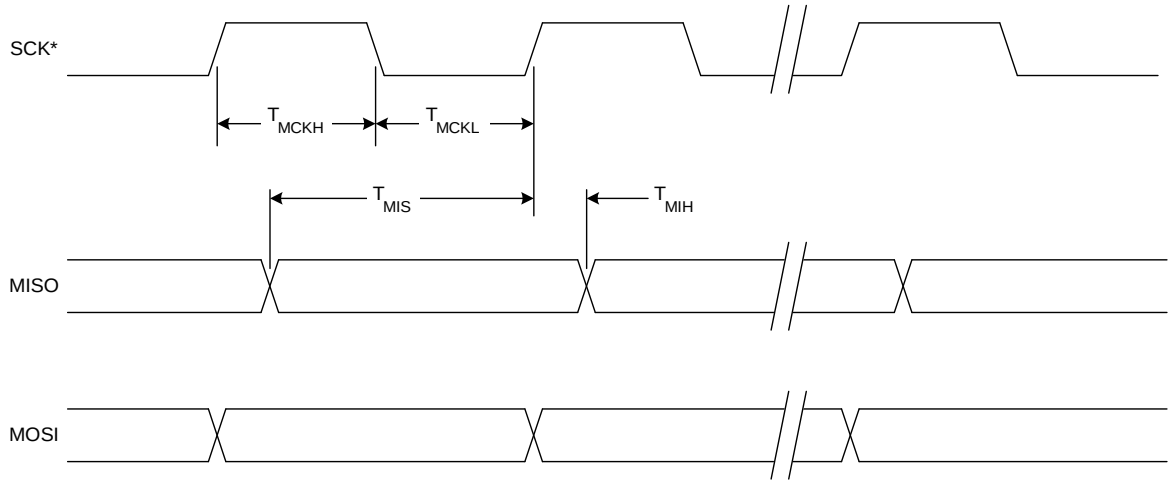


Downloaded from Arrow.com.




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

**Figure 22.8. SPI Master Timing (CKPHA = 0)**

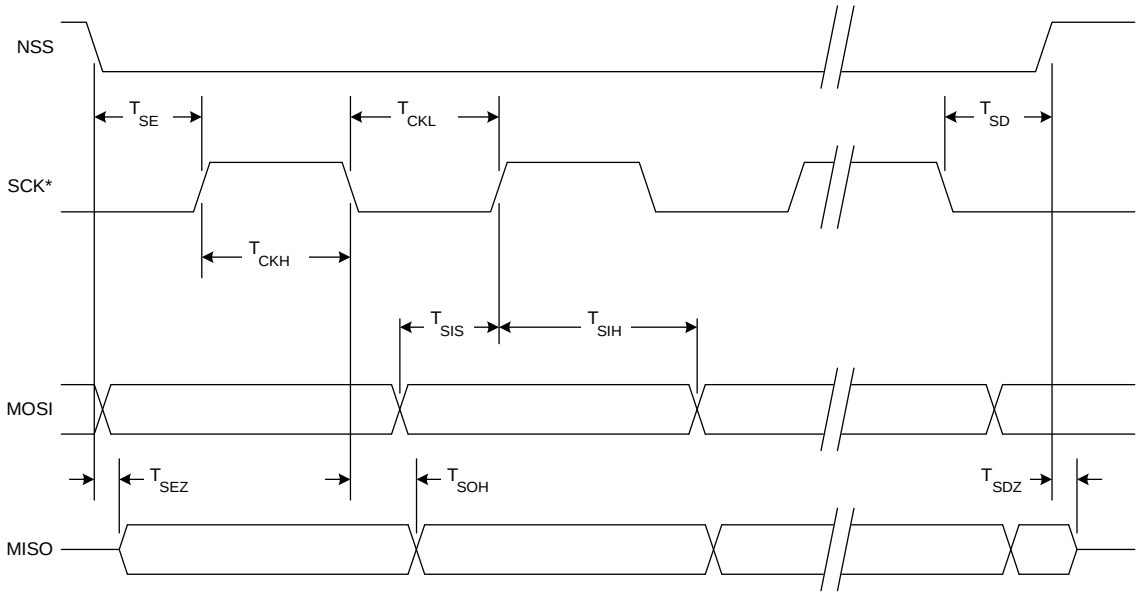



* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

**Figure 22.9. SPI Master Timing (CKPHA = 1)**

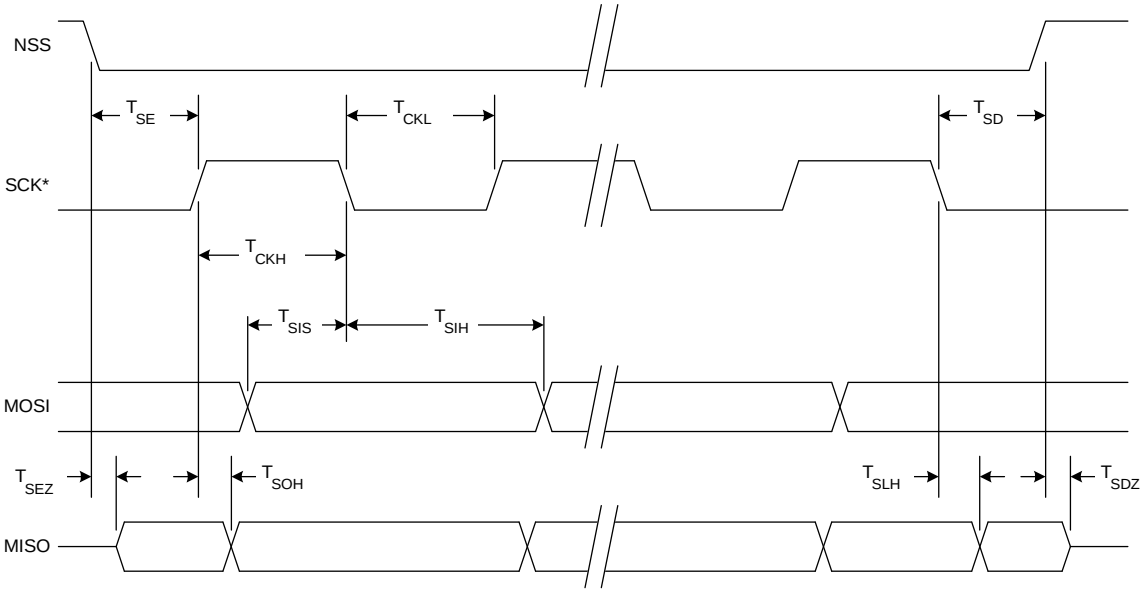


Downloaded from Arrow.com.




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

**Figure 22.10. SPI Slave Timing (CKPHA = 0)**




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

**Figure 22.11. SPI Slave Timing (CKPHA = 1)**



Downloaded from Arrow.com.

**Table 22.1. SPI Slave Timing Parameters**

| Parameter | Description | Min | Max | Units |
|---|---|---|---|---|
| **Master Mode Timing*** (See Figure 22.8 and Figure 22.9) | | | | |
| $T_{MCKH}$ | 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 22.10 and Figure 22.11) | | | | |
| $T_{SE}$ | 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).



Downloaded from Arrow.com.

# 23. 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.

| Timer 0 and Timer 1 Modes | Timer 2 Modes | Timer 3 Modes |
|---|---|---|
| 13-bit counter/timer | 16-bit timer with auto-reload | 16-bit timer with auto-reload |
| 16-bit counter/timer | | |
| 8-bit counter/timer with auto-reload | Two 8-bit timers with auto-reload | Two 8-bit timers with auto-reload |
| Two 8-bit counter/timers (Timer 0 only) | | |

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 23.1 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.



Downloaded from Arrow.com.

## SFR Definition 23.1. 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; SFR Page = All Pages

| 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) |



Downloaded from Arrow.com.

## 23.1. 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 “13.2. Interrupt Register Descriptions” on page 108); Timer 1 interrupts can be enabled by setting the ET1 bit in the IE register (Section “13.2. Interrupt Register Descriptions” on page 108). 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.

### 23.1.1. 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 (TCON.5) is set and an interrupt will occur if Timer 0 interrupts are enabled.

The C/T0 bit (TMOD.2) 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 “18.3. Priority Crossbar Decoder” on page 150 for information on selecting and configuring external I/O pins). Clearing C/T selects the clock defined by the T0M bit (CKCON.3). 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 23.1).

Setting the TR0 bit (TCON.4) enables the timer when either GATE0 (TMOD.3) is logic 0 or the input signal $\overline{\text{INT0}}$ is active as defined by bit IN0PL in register IT01CF (see SFR Definition 13.7). Setting GATE0 to 1 allows the timer to be controlled by the external input signal $\overline{\text{INT0}}$ (see Section “13.2. Interrupt Register Descriptions” on page 108), facilitating pulse width measurements.

| TR0 | GATE0 | $\overline{\text{INT0}}$ | 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 $\overline{\text{INT1}}$ is used with Timer 1; the $\overline{\text{INT1}}$ polarity is defined by bit IN1PL in register IT01CF (see SFR Definition 13.7).



Downloaded from Arrow.com.

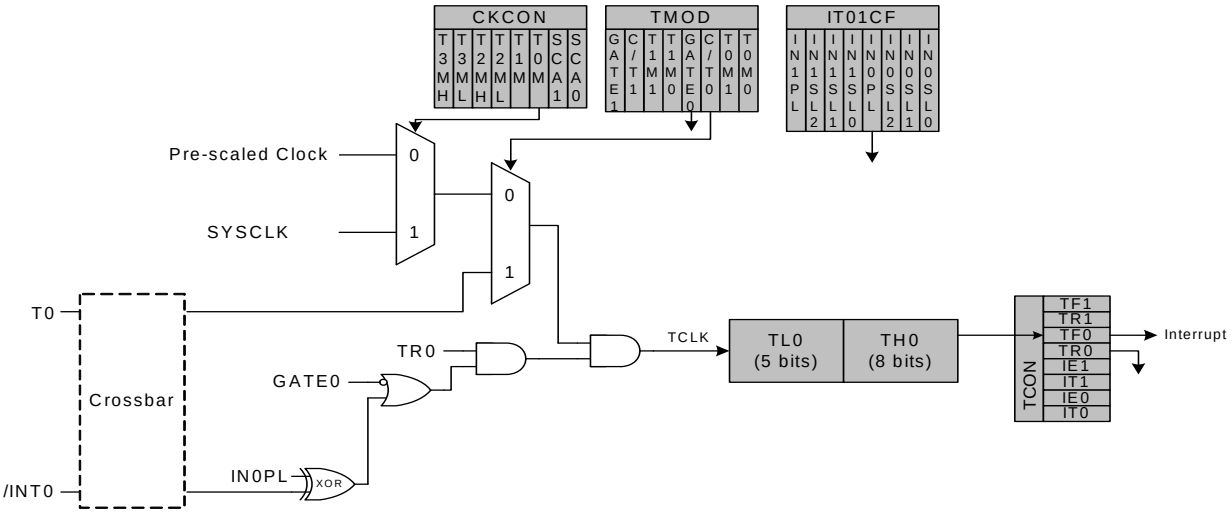



**Figure 23.1. T0 Mode 0 Block Diagram**

### 23.1.2. 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.

### 23.1.3. 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 (TCON.5) 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 (TMOD.3) is logic 0 or when the input signal INT0 is active as defined by bit IN0PL in register IT01CF (see Section “13.3. External Interrupts INT0 and INT1” on page 115 for details on the external input signals INT0 and INT1).



Downloaded from Arrow.com.

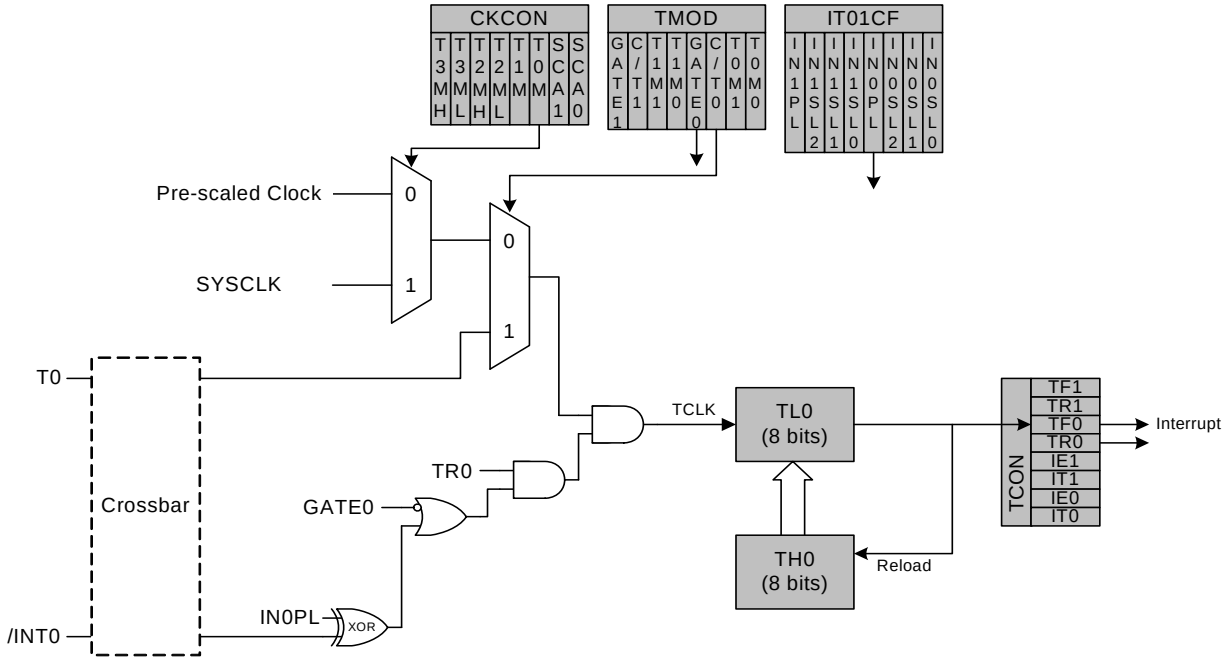



**Figure 23.2. T0 Mode 2 Block Diagram**

### 23.1.4. 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.



Downloaded from Arrow.com.

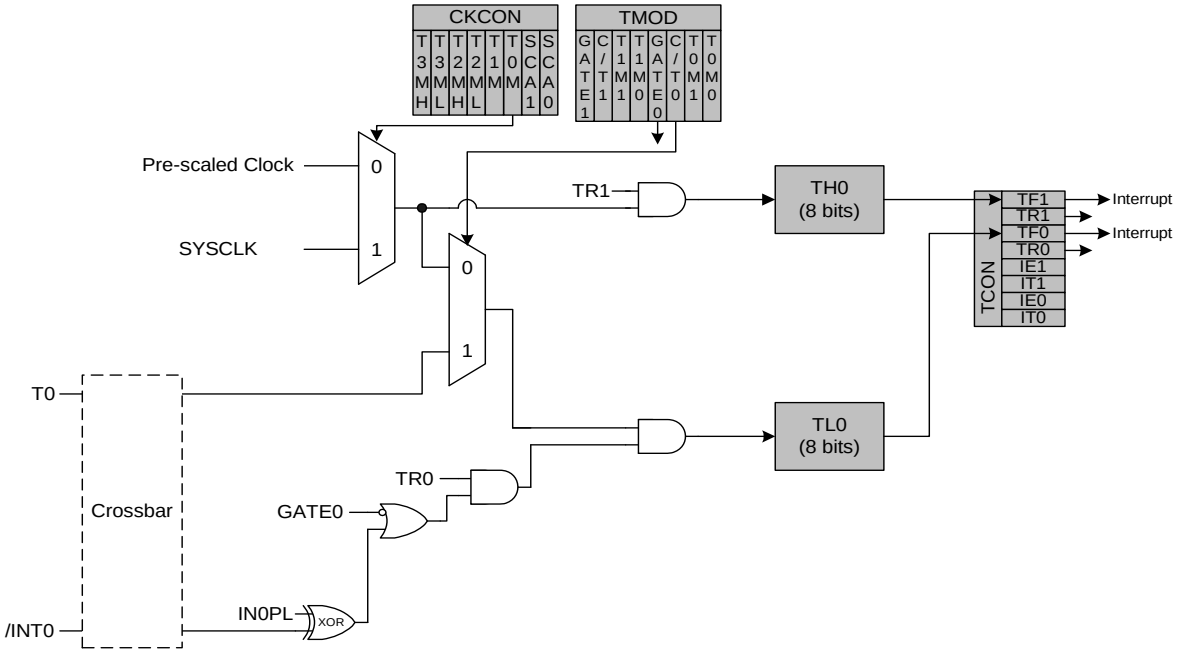



**Figure 23.3. T0 Mode 3 Block Diagram**



Downloaded from Arrow.com.

## SFR Definition 23.2. 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; SFR Page = All Pages

| 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 $\overline{\text{INT1}}$ interrupt will be edge or level sensitive. $\overline{\text{INT1}}$ is configured active low or high by the IN1PL bit in the IT01CF register (see SFR Definition 13.7).<br>0: $\overline{\text{INT1}}$ is level triggered.<br>1: $\overline{\text{INT1}}$ 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 13.7).<br>0: $\overline{\text{INT0}}$ is level triggered.<br>1: $\overline{\text{INT0}}$ is edge triggered. |



Downloaded from Arrow.com.

## SFR Definition 23.3. 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; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7 | GATE1 | **Timer 1 Gate Control.**<br>0: Timer 1 enabled when TR1 = 1 irrespective of $\overline{\text{INT1}}$ logic level.<br>1: Timer 1 enabled only when TR1 = 1 AND $\overline{\text{INT1}}$ is active as defined by bit IN1PL in register IT01CF (see SFR Definition 13.7). |
| 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 $\overline{\text{INT0}}$ logic level.<br>1: Timer 0 enabled only when TR0 = 1 AND $\overline{\text{INT0}}$ is active as defined by bit IN0PL in register IT01CF (see SFR Definition 13.7). |
| 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 |



Downloaded from Arrow.com.

## SFR Definition 23.4. 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; SFR Page = All Pages

| 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 23.5. 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; SFR Page = All Pages

| 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. |



Downloaded from Arrow.com.

## SFR Definition 23.6. 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; SFR Page = All Pages

| 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 23.7. 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; SFR Page = All Pages

| 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. |



Downloaded from Arrow.com.

## 23.2. 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.

### 23.2.1. 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 23.4, 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.

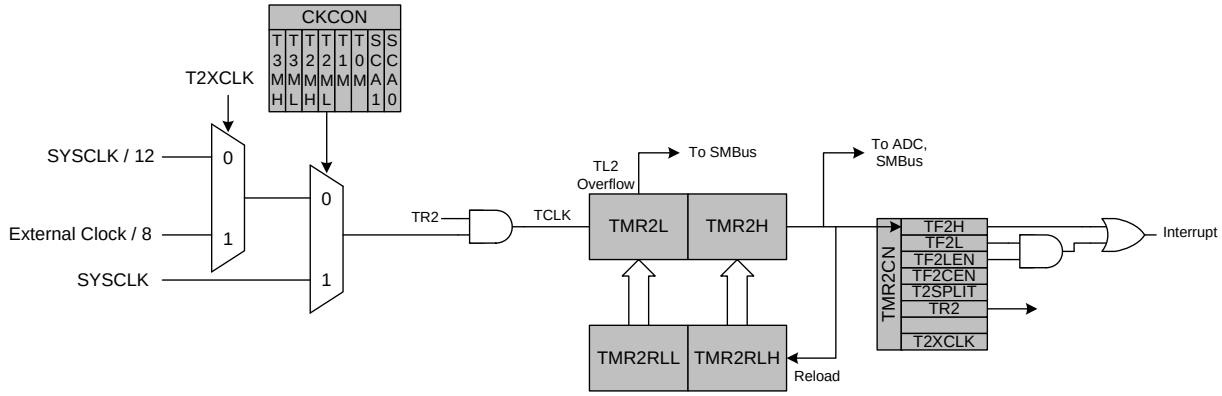



**Figure 23.4. Timer 2 16-Bit Mode Block Diagram**

### 23.2.2. 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 23.5. 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:



Downloaded from Arrow.com.

| 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.

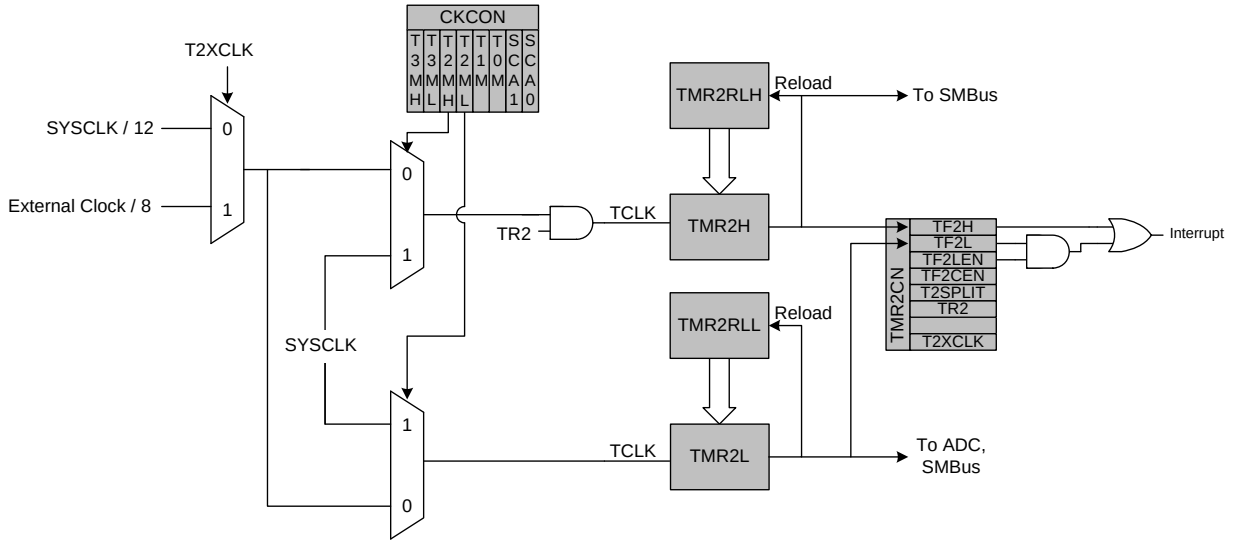



**Figure 23.5. Timer 2 8-Bit Mode Block Diagram**

### 23.2.3. External Oscillator Capture Mode

Capture Mode allows the external oscillator to be measured against the system clock. Timer 2 can be clocked from the system clock, or the system clock divided by 12, depending on the T2ML (CKCON.4), and T2XCLK bits. When a capture event is generated, the contents of Timer 2 (TMR2H:TMR2L) are loaded into the Timer 2 reload registers (TMR2RLH:TMR2RLL) and the TF2H flag is set. A capture event is generated by the falling edge of the clock source being measured, which is the external oscillator / 8. By recording the difference between two successive timer capture values, the external oscillator frequency can be determined with respect to the Timer 2 clock. The Timer 2 clock should be much faster than the capture clock to achieve an accurate reading. Timer 2 should be in 16-bit auto-reload mode when using Capture Mode.

For example, if T2ML = 1b and TF2CEN = 1b, Timer 2 will clock every SYSCLK and capture every external clock divided by 8. If the SYSCLK is 24 MHz and the difference between two successive captures is 5984, then the external clock frequency is as follows:

24 MHz/(5984/8) = 0.032086 MHz or 32.086 kHz



Downloaded from Arrow.com.

This mode allows software to determine the external oscillator frequency when an RC network or capacitor is used to generate the clock source.

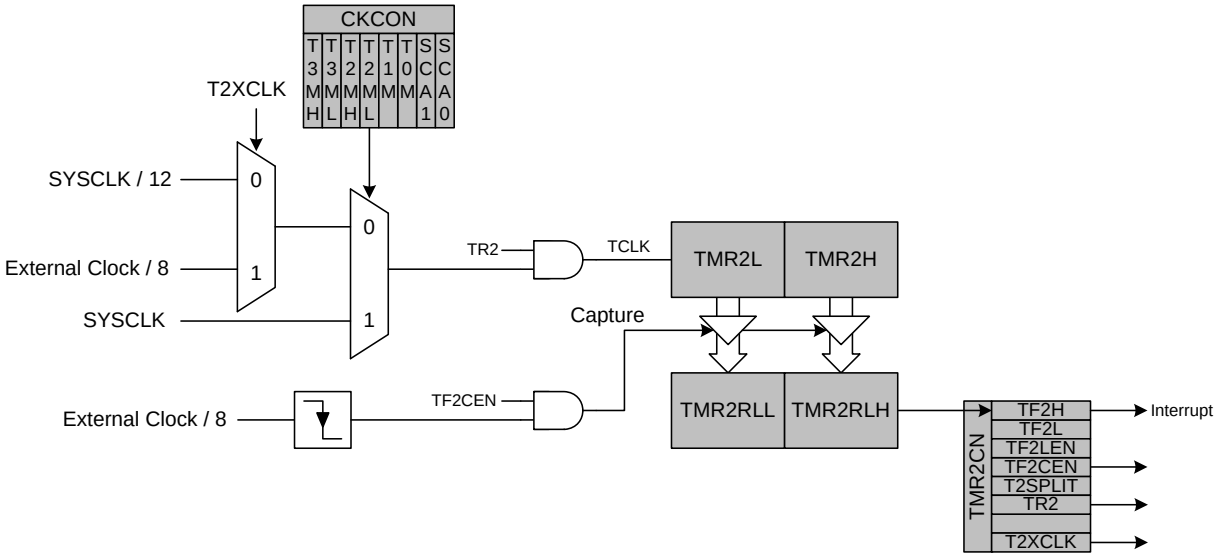



**Figure 23.6. Timer 2 External Oscillator Capture Mode Block Diagram**



Downloaded from Arrow.com.

## SFR Definition 23.8. 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; SFR Page = 0x00

| 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 Capture Mode Enable.**<br>0: Timer 2 Capture Mode is disabled.<br>1: Timer 2 Capture Mode is enabled. |
| 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 | 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). |



Downloaded from Arrow.com.

## SFR Definition 23.9. 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; SFR Page = 0x00

| 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 23.10. 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; SFR Page = 0x00

| 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. |



Downloaded from Arrow.com.

## SFR Definition 23.11. 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; SFR Page = 0x00

| 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 23.12. 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; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR2H[7:0] | **Timer 2 High 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. |



Downloaded from Arrow.com.

## 23.3. 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, 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 3 (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.

### 23.3.1. 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 23.7, and the Timer 3 High Byte Overflow Flag (TMR3CN.7) is set. If Timer 3 interrupts are enabled, 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.

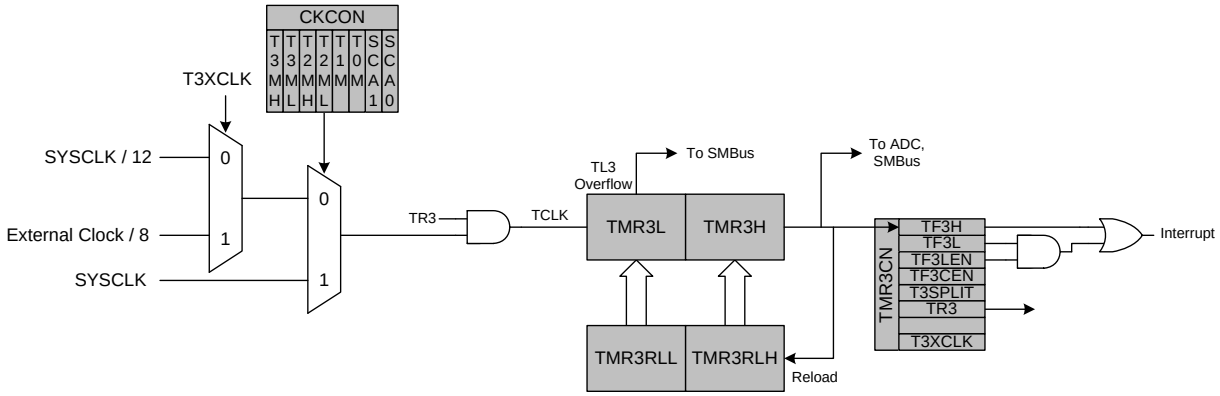



**Figure 23.7. Timer 3 16-Bit Mode Block Diagram**

### 23.3.2. 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 23.8. 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, or the external oscillator clock source divided by 8. 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 bit (T3XCLK in TMR3CN), as follows:



Downloaded from Arrow.com.

| T3MH | T3XCLK | TMR3H Clock Source |
|---|---|---|
| 0 | 0 | SYSCLK/12 |
| 0 | 1 | External Clock/8 |
| 1 | X | SYSCLK |

| T3ML | T3XCLK | TMR3L Clock Source |
|---|---|---|
| 0 | 0 | SYSCLK/12 |
| 0 | 1 | External Clock/8 |
| 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.

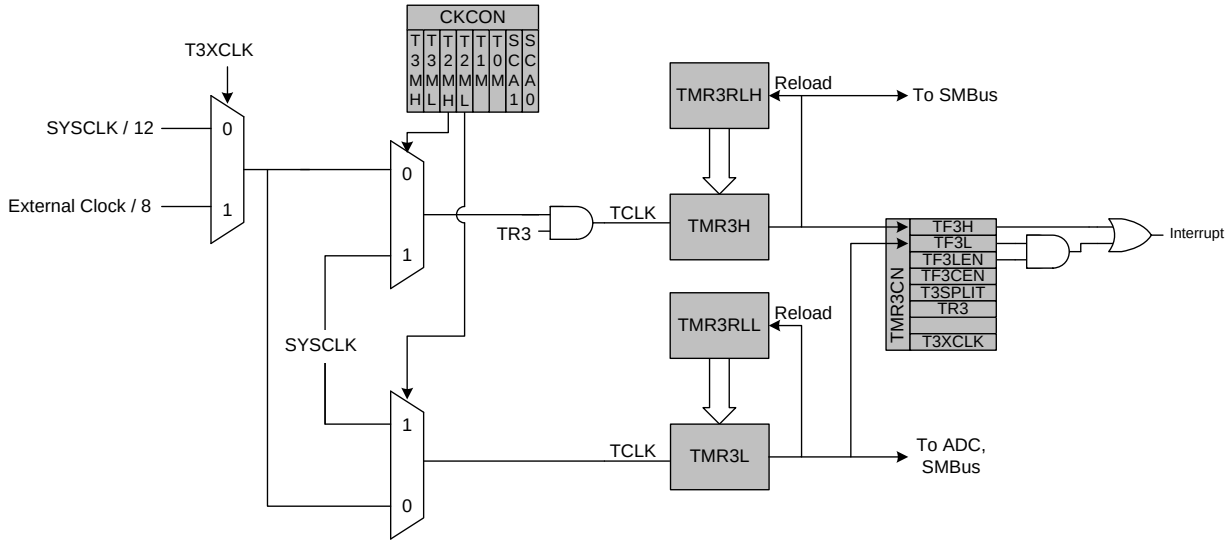



**Figure 23.8. Timer 3 8-Bit Mode Block Diagram**

### 23.3.3. External Oscillator Capture Mode

Capture Mode allows the external oscillator to be measured against the system clock. Timer 3 can be clocked from the system clock, or the system clock divided by 12, depending on the T3ML (CKCON.6), and T3XCLK bits. When a capture event is generated, the contents of Timer 3 (TMR3H:TMR3L) are loaded into the Timer 3 reload registers (TMR3RLH:TMR3RLL) and the TF3H flag is set. A capture event is generated by the falling edge of the clock source being measured, which is the external oscillator/8. By recording the difference between two successive timer capture values, the external oscillator frequency can be determined with respect to the Timer 3 clock. The Timer 3 clock should be much faster than the capture clock to achieve an accurate reading. Timer 3 should be in 16-bit auto-reload mode when using Capture Mode.

If the SYSCLK is 24 MHz and the difference between two successive captures is 5861, then the external clock frequency is as follows:

24 MHz/(5861/8) = 0.032754 MHz or 32.754 kHz

This mode allows software to determine the external oscillator frequency when an RC network or capacitor is used to generate the clock source.



Downloaded from Arrow.com.

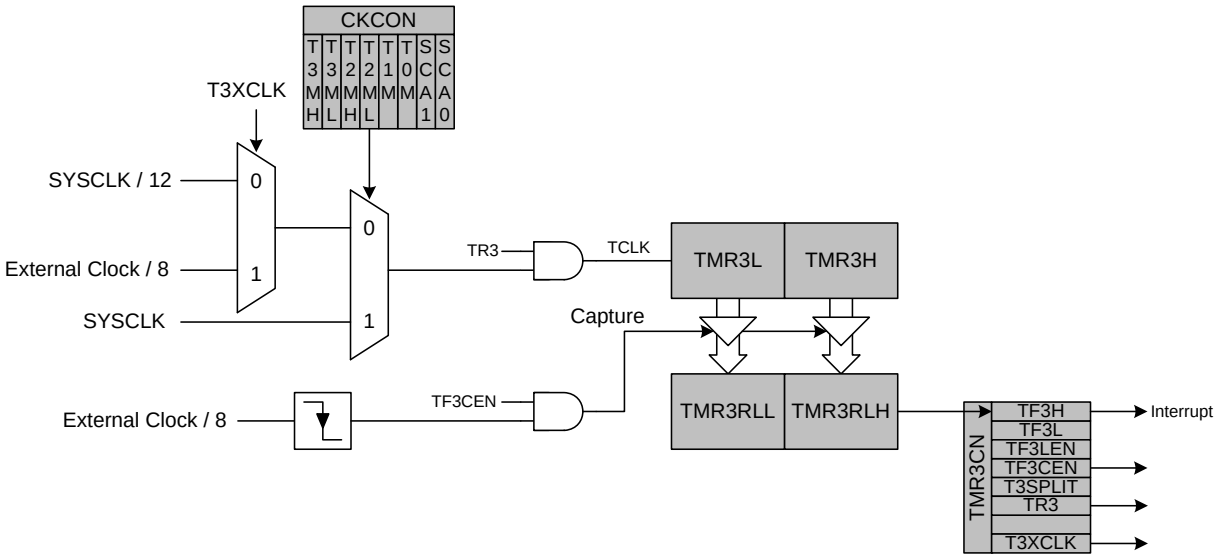



**Figure 23.9. Timer 3 External Oscillator Capture Mode Block Diagram**



Downloaded from Arrow.com.

## SFR Definition 23.13. TMR3CN: Timer 3 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF3H | TF3L | TF3LEN | TF3CEN | T3SPLIT | TR3 | | T3XCLK |
| 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 = 0x91; SFR Page = 0x00

| 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 Capture Mode Enable.**<br>0: Timer 3 Capture Mode is disabled.<br>1: Timer 3 Capture Mode is enabled. |
| 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 | Unused | Read = 0b; Write = Don't Care |
| 0 | T3XCLK | **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>0: Timer 3 clock is the system clock divided by 12.<br>1: Timer 3 clock is the external clock divided by 8 (synchronized with SYSCLK). |



Downloaded from Arrow.com.

## SFR Definition 23.14. 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; SFR Page = 0x00

| 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 23.15. 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; SFR Page = 0x00

| 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. |



Downloaded from Arrow.com.

## SFR Definition 23.16. 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; SFR Page = 0x00

| 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 23.17. 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; SFR Page = 0x00

| 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. |



Downloaded from Arrow.com.

## 24. 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 six 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 "24.3. Capture/Compare Modules" on page 252). 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 24.1

**Important Note:** The PCA Module 5 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 24.4 for details.

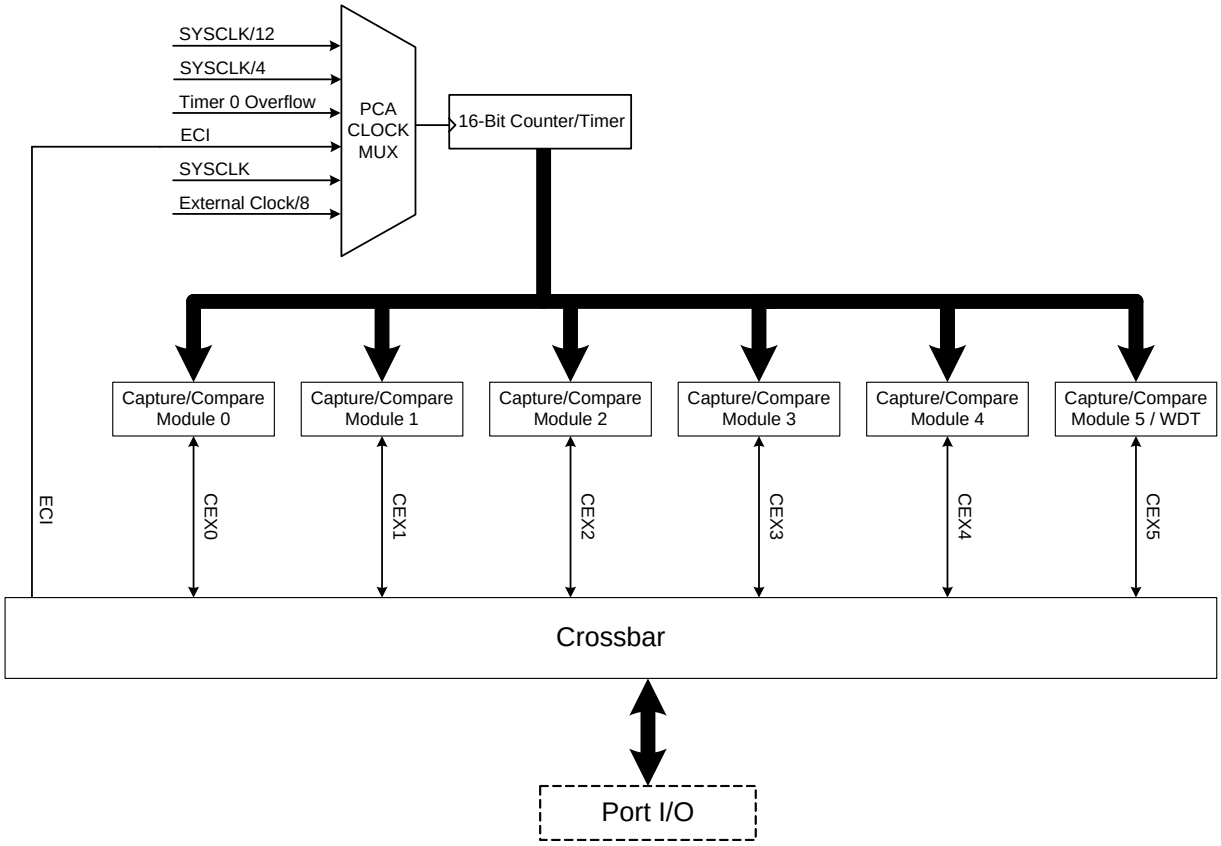



**Figure 24.1. PCA Block Diagram**



Downloaded from Arrow.com.

## 24.1. 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 CPS[2:0] bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 24.1.

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 24.1. 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.



Downloaded from Arrow.com.

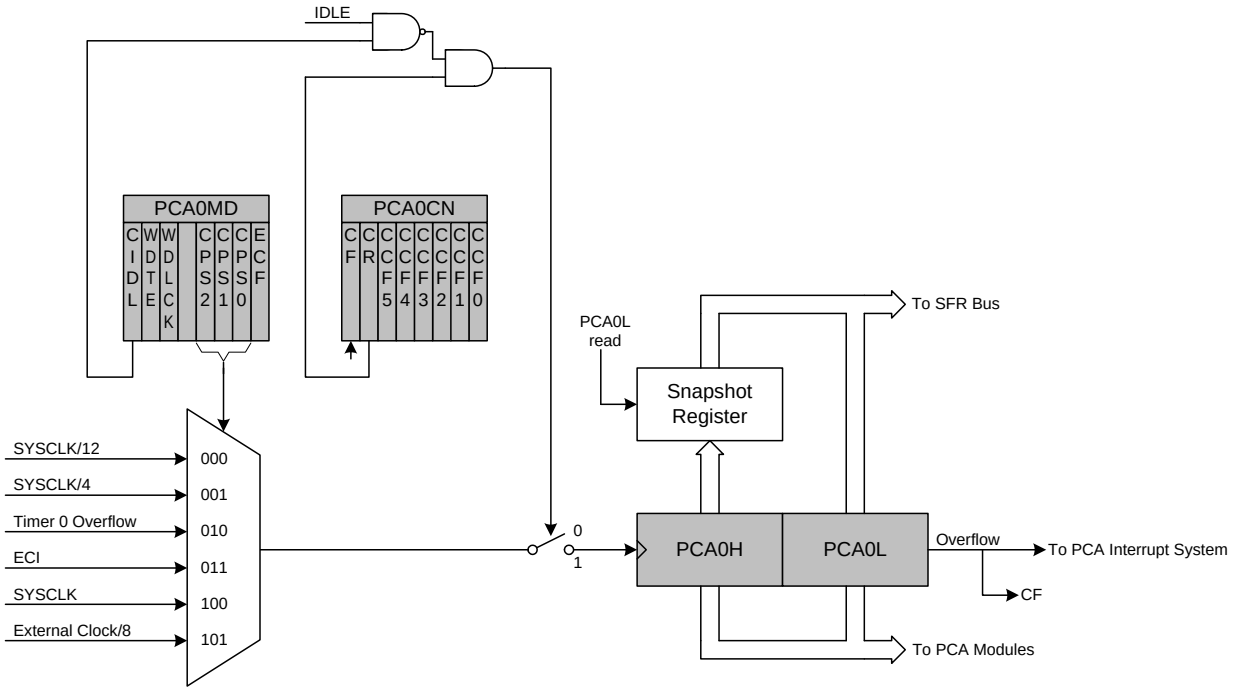



**Figure 24.2. PCA Counter/Timer Block Diagram**

## 24.2. PCA0 Interrupt Sources

Figure 24.3 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 as follows: 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, CCF2, CCF3, CCF4, and CCF5), 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.



Downloaded from Arrow.com.

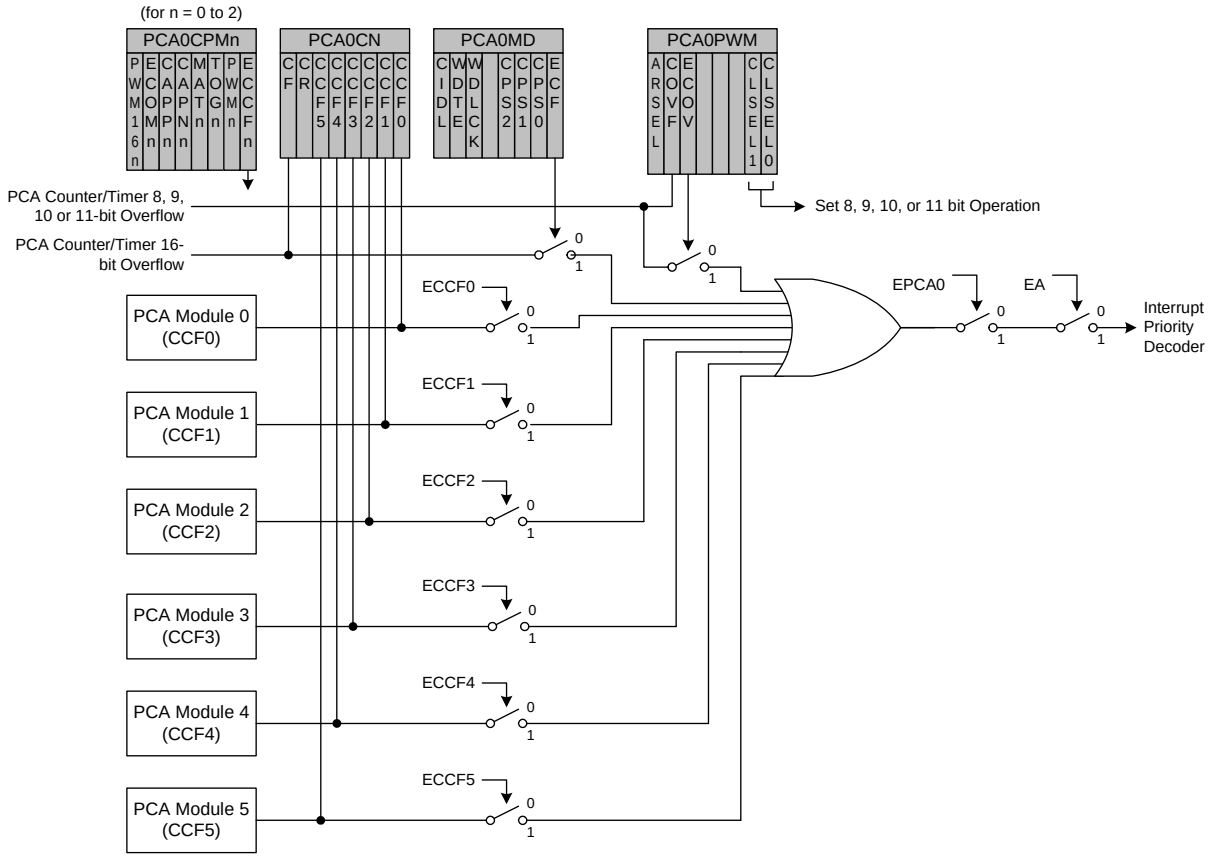



**Figure 24.3. PCA Interrupt Block Diagram**

## 24.3. 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 24.2 summarizes the bit settings in the PCA0CPMn and PCA0PWM registers used to select the PCA capture/compare module's operating mode. 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.



Downloaded from Arrow.com.

**Table 24.2. 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 = 1 to enable interrupts for this module (PCA interrupt triggered on CCFn set to 1).
3. B = 1 to 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 to 1, 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.

### 24.3.1. 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.



Downloaded from Arrow.com.

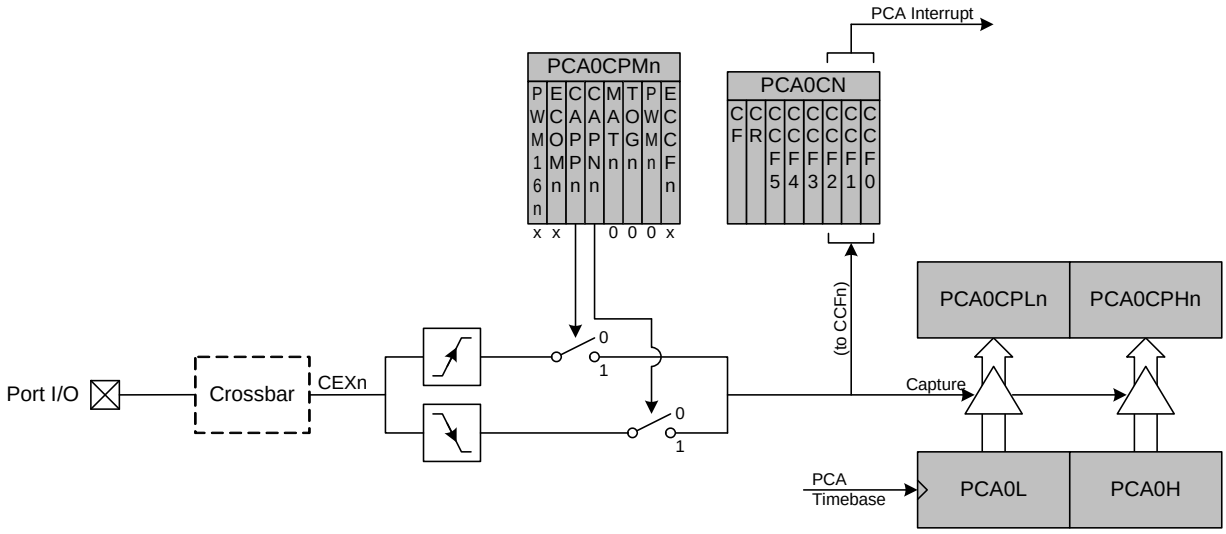



**Figure 24.4. PCA Capture Mode Diagram**

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

### 24.3.2. 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.



Downloaded from Arrow.com.

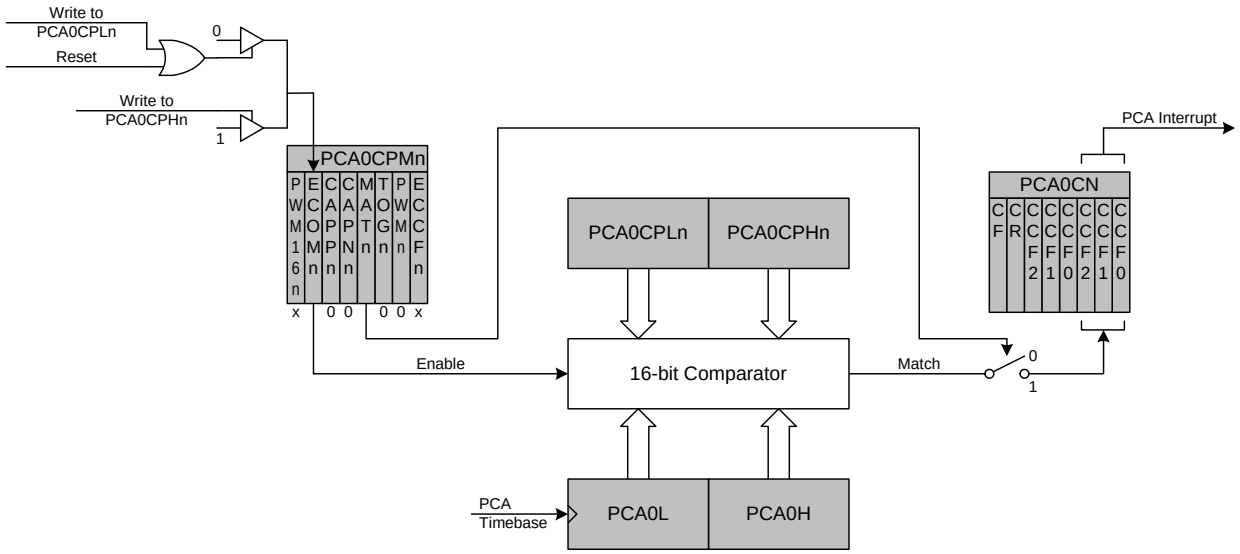



**Figure 24.5. PCA Software Timer Mode Diagram**

### 24.3.3. 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.



Downloaded from Arrow.com.

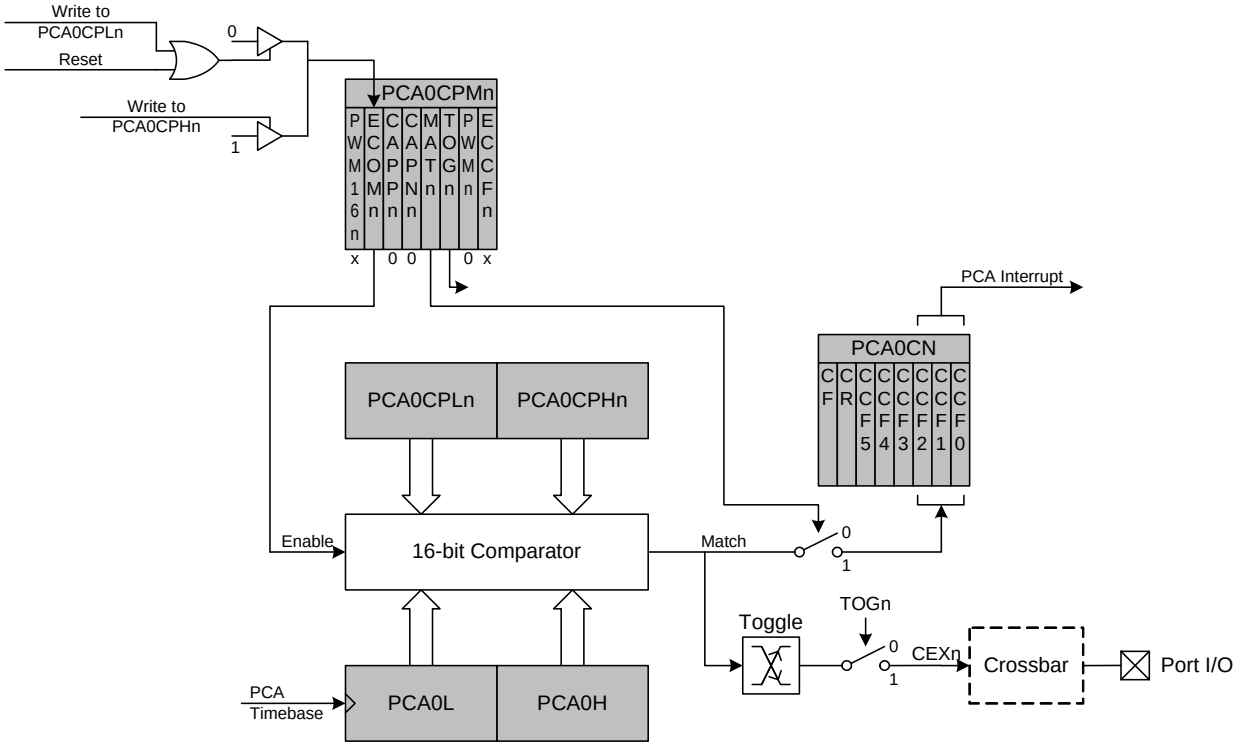



**Figure 24.6. PCA High-Speed Output Mode Diagram**

### 24.3.4. 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 24.1.

$$F_{CEXn} = \frac{F_{PCA}}{2 \times PCA0CPHn}$$

Note: A value of 0x00 in the PCA0CPHn register is equal to 256 for this equation.

**Equation 24.1. Square Wave Frequency Output**

Where $F_{PCA}$ is the frequency of the clock selected by the CPS[2:0] 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.



Downloaded from Arrow.com.

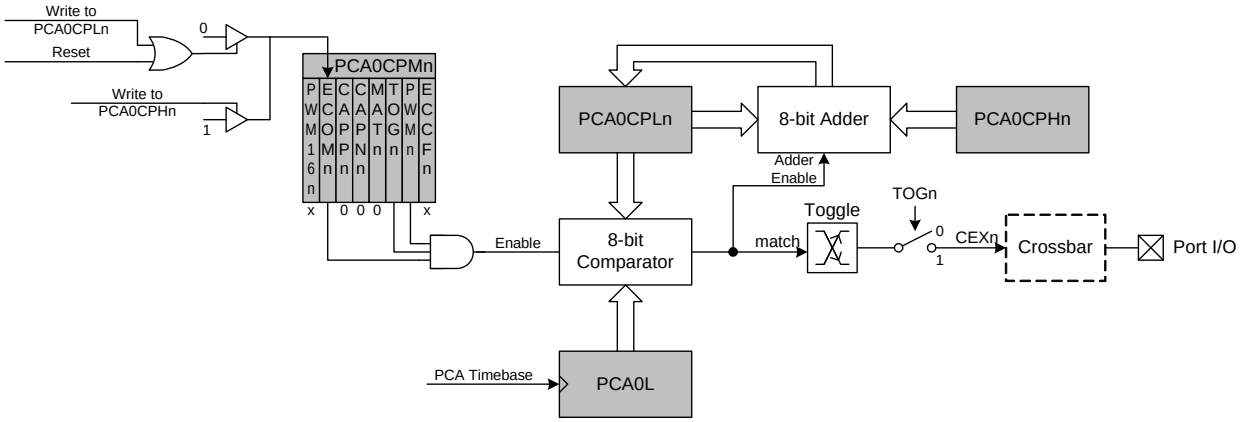



**Figure 24.7. PCA Frequency Output Mode**

### 24.3.5. 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.

#### 24.3.5.1. 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 24.8). 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 24.2.

**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.

$$\text{Duty Cycle} = \frac{(256 - \text{PCA0CPHn})}{256}$$

**Equation 24.2. 8-Bit PWM Duty Cycle**

Using Equation 24.2, 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.



Downloaded from Arrow.com.

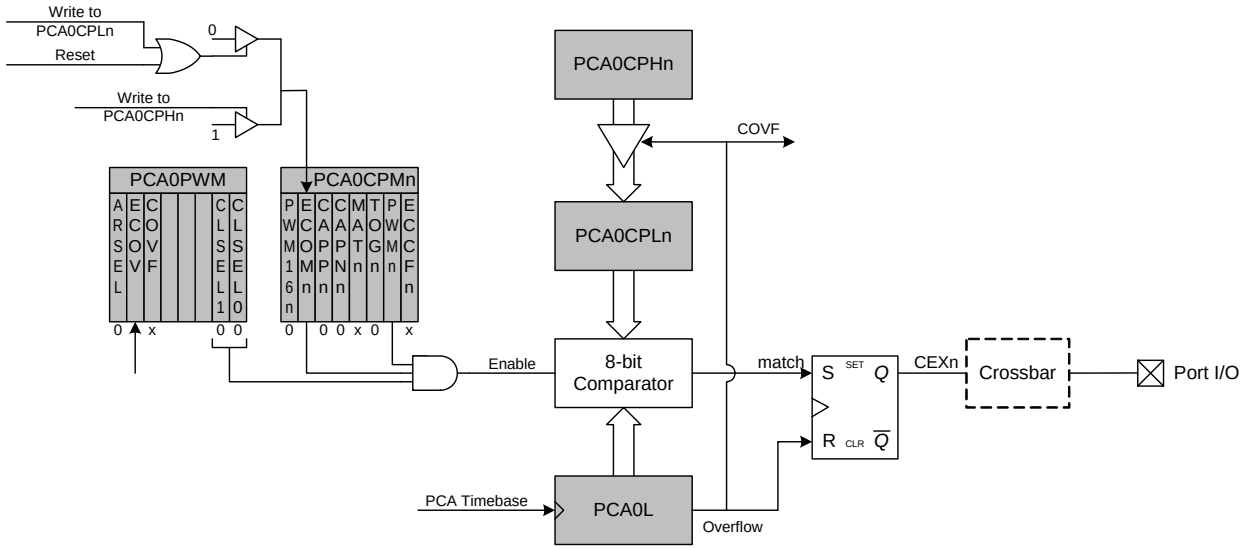



**Figure 24.8. PCA 8-Bit PWM Mode Diagram**

### 24.3.5.2. 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 24.9). 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 24.2, 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.

$$\text{Duty Cycle} = \frac{(2^N - \text{PCA0CPn})}{2^N}$$

**Equation 24.3. 9, 10, and 11-Bit PWM Duty Cycle**

A 0% duty cycle may be generated by clearing the ECOMn bit to 0.



Downloaded from Arrow.com.

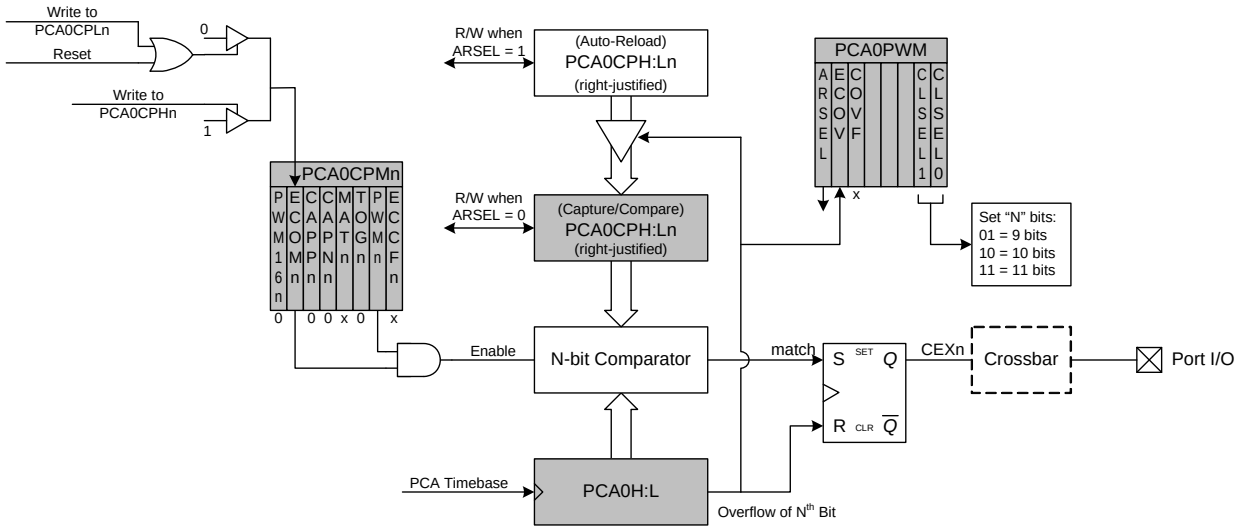



**Figure 24.9. PCA 9, 10 and 11-Bit PWM Mode Diagram**

### 24.3.6. 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 24.4.

**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.

$$\text{Duty Cycle} = \frac{(65536 - \text{PCA0CPn})}{65536}$$

**Equation 24.4. 16-Bit PWM Duty Cycle**

Using Equation 24.4, 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.



Downloaded from Arrow.com.

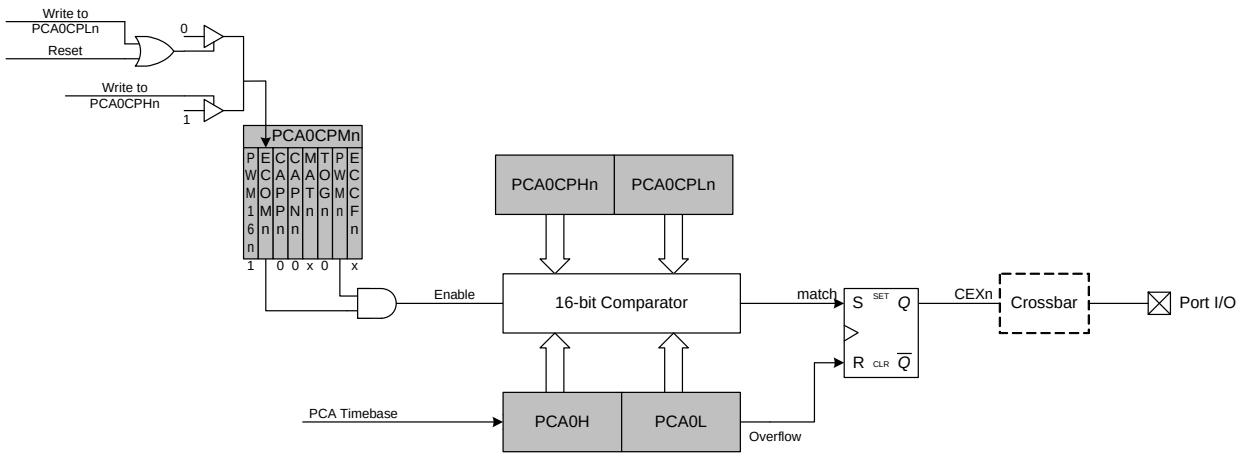



**Figure 24.10. PCA 16-Bit PWM Mode**

## 24.4. Watchdog Timer Mode

A programmable watchdog timer (WDT) function is available through the PCA Module 5. The WDT is used to generate a reset if the time between writes to the WDT update register (PCA0CPH5) 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 5 operates as a watchdog timer (WDT). The Module 5 high byte is compared to the PCA counter high byte; the Module 5 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).

### 24.4.1. 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 (CPS[2:0]) are frozen.
- PCA Idle control bit (CIDL) is frozen.
- Module 5 is forced into software timer mode.
- Writes to the Module 5 mode register (PCA0CPM5) 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 PCA0CPH5 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 PCA0CPH5. Upon a PCA0CPH5 write, PCA0H plus the offset held in PCA0CPL5 is loaded into PCA0CPH5 (See Figure 24.11).



Downloaded from Arrow.com.

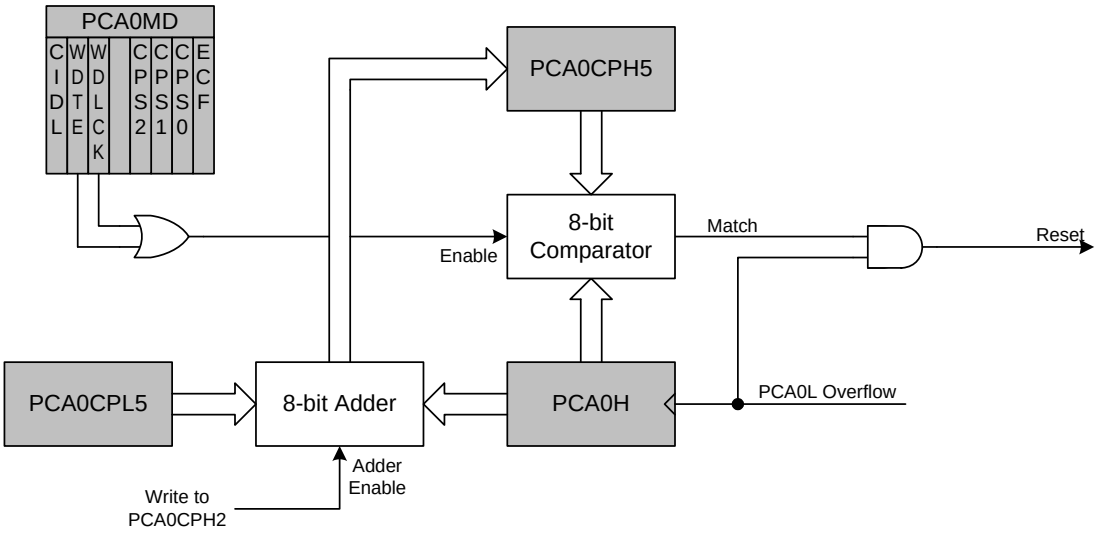



**Figure 24.11. PCA Module 2 with Watchdog Timer Enabled**

Note that the 8-bit offset held in PCA0CPH5 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 24.5, where PCA0L is the value of the PCA0L register at the time of the update.

$$Offset = (256 \times PCA0CPL5) + (256 - PCA0L)$$

**Equation 24.5. Watchdog Timer Offset in PCA Clocks**

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH5 and PCA0H. Software may force a WDT reset by writing a 1 to the CCF5 flag (PCA0CN.5) while the WDT is enabled.

### 24.4.2. Watchdog Timer Usage

To configure the WDT, perform the following tasks:

- Disable the WDT by writing a 0 to the WDTE bit.
- Select the desired PCA clock source (with the CPS[2:0] bits).
- Load PCA0CPL5 with the desired WDT update offset value.
- Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
- Enable the WDT by setting the WDTE bit to 1.
- Reset the WDT timer by writing to PCA0CPH5.

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 PCA0CPL5 defaults to 0x00. Using Equation 24.5, this results in a WDT timeout interval of 256 PCA clock cycles, or 3072 system clock cycles. Table 24.3 lists some example timeout intervals for typical system clocks.



Downloaded from Arrow.com.

**Table 24.3. Watchdog Timer Timeout Intervals[1]**

| System Clock (Hz) | PCA0CPL5 | Timeout Interval (ms) |
|---|---|---|
| 24,000,000 | 255 | 32.8 |
| 24,000,000 | 128 | 16.5 |
| 24,000,000 | 32 | 4.2 |
| 3,000,000 | 255 | 262.1 |
| 3,000,000 | 128 | 132.1 |
| 3,000,000 | 32 | 33.8 |
| 187,500[2] | 255 | 4194 |
| 187,500[2] | 128 | 2114 |
| 187,500[2] | 32 | 541 |

**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 128.



Downloaded from Arrow.com.

## 24.5. Register Descriptions for PCA0

Following are detailed descriptions of the special function registers related to the operation of the PCA.

### SFR Definition 24.1. PCA0CN: PCA Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CF | CR | CCF5 | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 |
| 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 = 0xD8; Bit-Addressable; SFR Page = 0x00

| 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 | CCF5 | **PCA Module 5 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF5 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. |
| 4 | CCF4 | **PCA Module 4 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF4 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. |
| 3 | CCF3 | **PCA Module 3 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF3 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. |
| 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. |



Downloaded from Arrow.com.

## SFR Definition 24.2. PCA0MD: PCA Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CIDL | WDTE | WDLCK | | CPS[2:0] | | | 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; SFR Page = 0x00

| 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 5 is used as the watchdog timer.<br>0: Watchdog Timer disabled.<br>1: PCA Module 5 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 | 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. | | |



Downloaded from Arrow.com.

## SFR Definition 24.3. 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 = 0xD9; SFR Page = 0x0F

| 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 | 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. |



Downloaded from Arrow.com.

## SFR Definition 24.4. 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; PCA0CPM3 = 0xDD, PCA0CPM4 = 0xDE, PCA0CPM5 = 0xDF, SFR Page (all registers) = 0x00

| 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 PCA0CPM5 register cannot be modified, and module 5 acts as the watchdog timer. To change the contents of the PCA0CPM5 register or the function of module 5, the Watchdog Timer must be disabled. | | |



Downloaded from Arrow.com.

## SFR Definition 24.5. 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; SFR Page = 0x00

| 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 24.6. 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; SFR Page = 0x00

| 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 24.1). |

**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.



Downloaded from Arrow.com.

## SFR Definition 24.7. 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, PCA0CPL3 = 0xED, PCA0CPL4 = 0xFD, PCA0CPL5 = 0xCE; SFR Page (all registers) = 0x00

| 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 24.8. 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, PCA0CPH3 = 0xEE, PCA0CPH4 = 0xFE, PCA0CPH5 = 0xCF; SFR Page (all registers) = 0x00

| 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. | | |



Downloaded from Arrow.com.

# 25. C2 Interface

C8051F54x 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.

## 25.1. 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 25.1. 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 |



Downloaded from Arrow.com.

## C2 Register Definition 25.2. 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 = 0xFD; SFR Address = 0xFD; SFR Page = 0xF

| Bit | Name | Function |
|---|---|---|
| 7:0 | DEVICEID[7:0] | **Device ID.**<br>This read-only register returns the 8-bit device ID: 0x22 (C8051F54x). |

## C2 Register Definition 25.3. 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 = 0xFE; SFR Address = 0xFE; SFR Page = 0xF

| 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. |



Downloaded from Arrow.com.

## C2 Register Definition 25.4. 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 25.5. 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 |



Downloaded from Arrow.com.

## 25.2. 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 25.1.

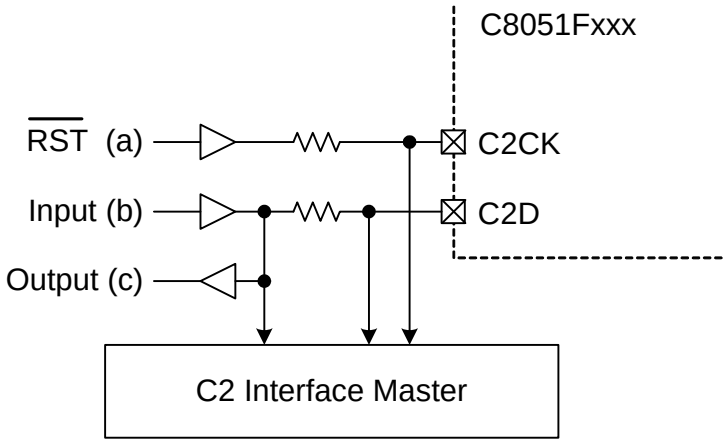



**Figure 25.1. Typical C2 Pin Sharing**

The configuration in Figure 25.1 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.



Downloaded from Arrow.com.

# DOCUMENT CHANGE LIST

## Revision 0.1 to Revision 1.0

- Updated “2. Ordering Information” to include -A (Automotive) devices and automotive qualification information.
- Updated Figure 4.6.
- Updated supply current related specifications throughout “6. Electrical Characteristics” .
- Updated SFR Definition 7.1 (REF0CN) to change VREF high setting to 2.20 V from 2.25 V.
- Updated Figure 8.1 to indicate that Comparators are powered from $V_{IO}$ and not $V_{DDA}$.
- Updated the Gain Table in “5.3.1. Calculating the Gain Value” to fix the ADC0GNH Value in the last row.
- Updated Table 10.1 with correct timing for all branch instructions, MOVC, and CPL A.
- Updated Table 14.1 to indicate behavior when performing a Flash operation in reserved space.
- Updated “14.1. Programming the Flash Memory” to clarify behavior of 8-bit MOVX instructions and when writing/erasing Flash.
- Updated SFR Definition 14.3 (FLSCL) to include FLEWT bit definition. This bit must be set before writing or erasing Flash. Also updated Table 6.5 to reflect new Flash Write and Erase timing.
- Updated “16.7. Flash Error Reset” with an additional cause of a Flash Error reset.
- Updated “18.1.3. Interfacing Port I/O in a Multi-Voltage System” to remove note regarding interfacing to voltages above VIO.
- Updated “20. SMBus” to remove all hardware ACK features, including SMB0ADM and SMB0ADR SFRs.
- Updated SFR Definition 21.1(SCON0) to correct SFR Page to 0x00 from All Pages.
- Updated CP Register Definition 24.2 with proper Device ID.

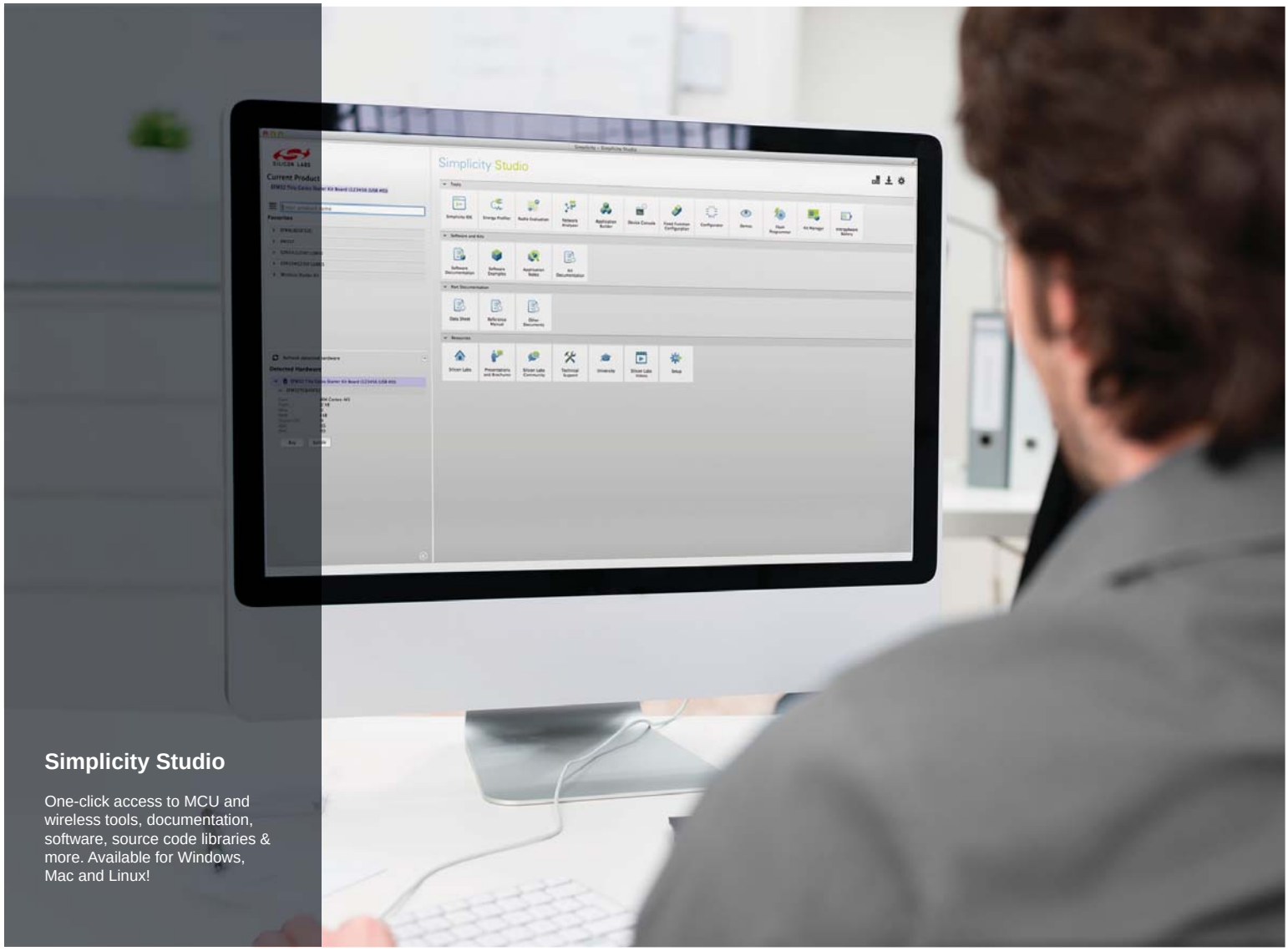
**Note:** All items from the C8051F54x Errata dated November 5th, 2009 are incorporated into this data sheet.

## Revision 1.0 to Revision 1.1

- Updated “1. System Overview” with a voltage range specification for the internal oscillator.
- Updated Figure 5.4, “12-Bit ADC Burst Mode Example With Repeat Count Set to 4,” on page 33 with new timing diagram when using CNVSTR pin.
- Updated Table 6.6, “Internal High-Frequency Oscillator Electrical Characteristics,” on page 53 with new conditions for the internal oscillator accuracy. The internal oscillator accuracy is dependent on the operating voltage range.
- Updated “6. Electrical Characteristics” to remove the internal oscillator curve across temperature diagram.
- Updated SFR Definition 7.1 (REF0CN) with oscillator suspend requirement for ZTCEN.
- Fixed incorrect cross references in “8. Comparators” .
- Updated SFR Definition 9.1 (REG0CN) with a new definition for Bit 6. The bit 6 reset value is 1b and must be written to 1b.
- Updated Figure 11.2, “Flash Program Memory Map,” on page 86 with correct address for start of lock byte page from 0x3900 to 0x3A00.
- Updated “15.3. Suspend Mode” with note regarding ZTCEN.
- Added Port 2 Event and Port 3 Event to wake-up sources in “17.2.1. Internal Oscillator Suspend Mode”
- Updated “19. Local Interconnect Network (LIN)” with a voltage range specification for the internal oscillator.
- Updated LIN Register Definitions for LIN0MUL and LIN0DIV to correct the reset value.
- Updated C2 Register Definitions 25.2 and 25.3 with correct C2 and SFR addresses.



Downloaded from Arrow.com.




## 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**

Downloaded from Arrow.com.