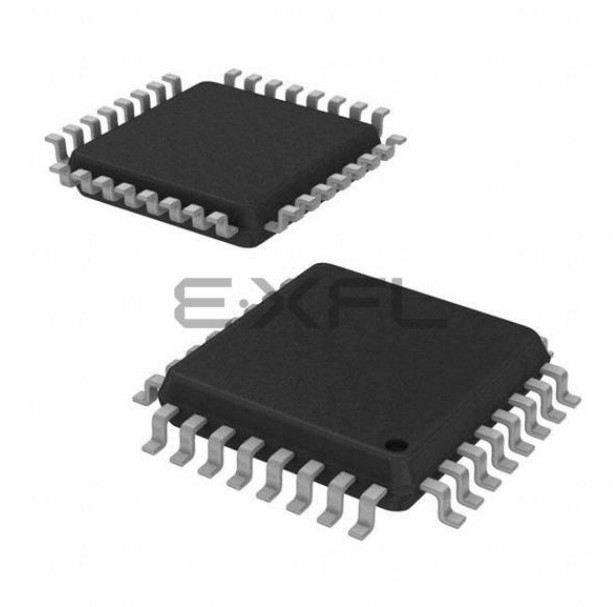

**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Active |
| Core Processor | 8051 |
| Core Size | 8-Bit |
| Speed | 50MHz |
| Connectivity | SMBus (2-Wire/I²C), SPI, UART/USART |
| Peripherals | POR, PWM, Temp Sensor, WDT |
| Number of I/O | 17 |
| Program Memory Size | 8KB (8K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 768 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 3.6V |
| Data Converters | A/D 8x24b; D/A 2x8b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 32-LQFP |
| Supplier Device Package | 32-LQFP (7x7) |
| Purchase URL | https://www.e-xfl.com/product-detail/silicon-labs/c8051f350-gq |

## 1.4. 24 or 16-Bit Analog to Digital Converter (ADC0)

The C8051F350/1/2/3 include a fully-differential, 24-bit (C8051F350/1) or 16-bit (C8051F352/3) Sigma-Delta Analog to Digital Converter (ADC) with on-chip calibration capabiliites. Two separate decimation filters can be programmed for throughputs of up to 1 kHz. An internal 2.5 V reference is available, or a differential external reference can be used for ratiometric measurements. A Programmable Gain Amplifier (PGA) is included, with eight gain settings up to 128x. An analog front-end multiplexer connects the differential inputs to eight external pins, the internal temperature sensor, or AGND. The on-chip input buffers can be used to provide a high input impedance for direct connection to sensitive transducers. An 8-bit offset DAC allows for correction of large input offset voltages.

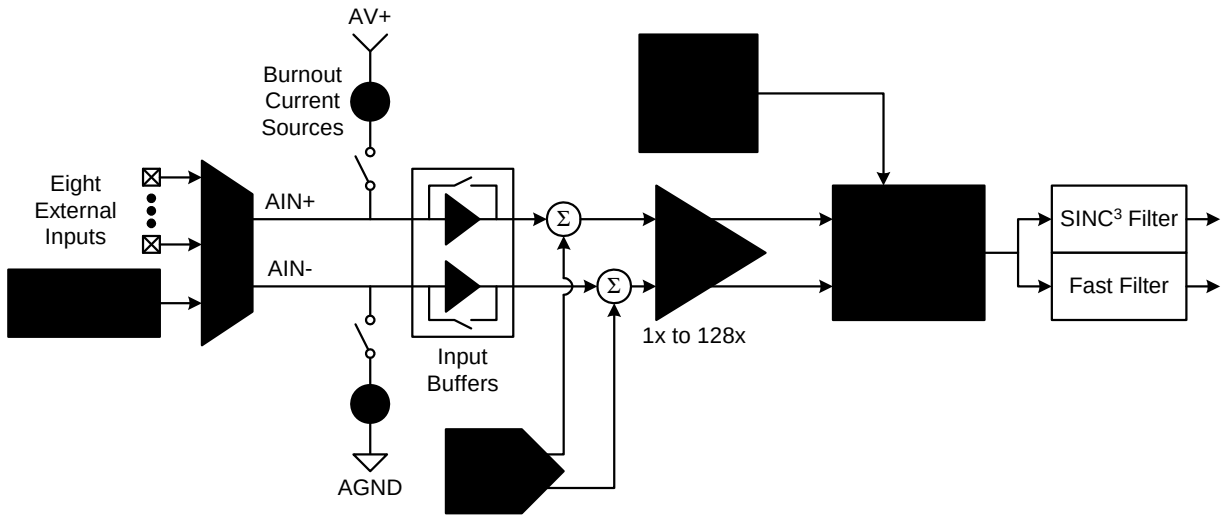


**Figure 1.7. ADC0 Block Diagram**

## NOTES:

# 5. 24 or 16-Bit Analog to Digital Converter (ADC0)

The C8051F350/1/2/3 include a fully-differential, 24-bit (C8051F350/1) or 16-bit (C8051F352/3) Sigma-Delta Analog to Digital Converter (ADC) with on-chip calibration capabiliites. Two separate decimation filters can be programmed for throughputs of up to 1 kHz. An internal reference is available, or a differential external reference can be used for ratiometric measurements. A Programmable Gain Amplifier (PGA) is included, with eight gain settings up to 128x. The on-chip input buffers can be used to provide a high input impedance for direct connection to sensitive transducers. An 8-bit offset DAC allows for correction of large input offset voltages.

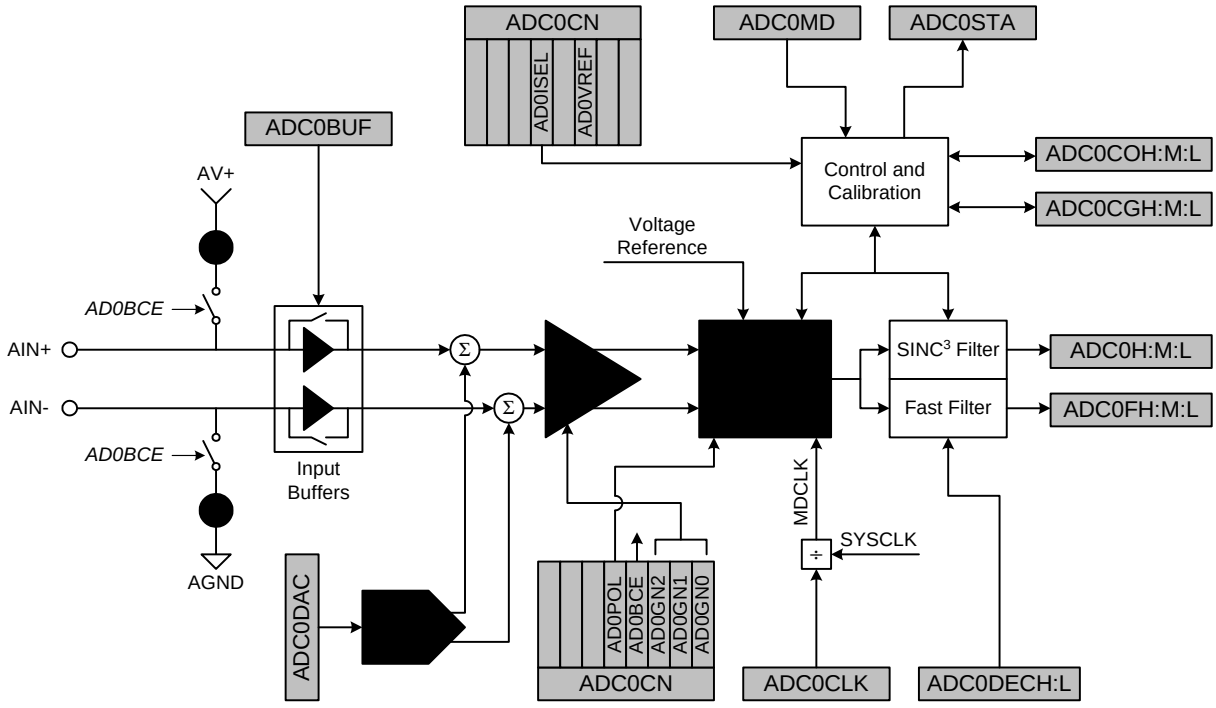


**Figure 5.1. ADC0 Block Diagram**

## SFR Definition 5.1. ADC0CN: ADC0 Control

| R | R | R | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| — | — | — | AD0POL | AD0BCE | AD0GN | | | 00010000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | |

SFR Address: 0xF4

Bits 7–5: Unused: Read = 000b, Write = don't care.
Bit 4: AD0POL: ADC0 Polarity.
0: ADC operates in Unipolar mode (straight binary result).
1: ADC operates in Bipolar mode (2's compliment result).
Bit 3: AD0BCE: ADC0 Burnout Current Source Enable.
0: ADC Burnout current sources disabled.
1: ADC Burnout current sources enabled.
Bits 2:0 AD0GN: ADC0 Programmable Gain Setting.
000: PGA Gain = 1.
001: PGA Gain = 2.
010: PGA Gain = 4.
011: PGA Gain = 8.
100: PGA Gain = 16.
101: PGA Gain = 32.
110: PGA Gain = 64.
111: PGA Gain = 128.

**This SFR can only be modified when ADC0 is in IDLE mode.**

**Table 5.9. ADC0 Fast Filter Flicker-Free (Noise-Free) Resolution[1] in Unipolar Mode (bits)**

| Decimation Ratio | Output Word Rate[2] | PGA Gain Setting | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| 1920 | 10 Hz | 16.26 | 16.11 | 15.90 | 15.49 | 14.95 | 14.24 | 13.37 | 12.32 |
| 768 | 25 Hz | 14.37 | 14.24 | 13.98 | 13.64 | 13.05 | 12.24 | 11.34 | 10.39 |
| 640 | 30 Hz | 13.63 | 13.64 | 13.57 | 13.10 | 12.57 | 11.82 | 10.86 | 9.93 |
| 384 | 50 Hz | 11.83 | 11.93 | 11.85 | 11.72 | 11.32 | 10.69 | 9.84 | 8.93 |
| 320 | 60 Hz | 11.11 | 11.04 | 11.11 | 11.00 | 10.74 | 10.16 | 9.36 | 8.44 |
| 192 | 100 Hz | 9.43 | 9.28 | 9.40 | 9.34 | 9.17 | 8.86 | 8.21 | 7.39 |

**Notes:**

**1.** *Flicker-free (Noise-free) Resolution* = $\log_2\left(\frac{FullInputRange(V)}{6.6 \times RMS\ Noise(V)}\right)$

where *Full Input Range* = $\frac{V_{REF}}{PGA\ Gain}$ in Unipolar mode and *RMS Noise* is obtained from Table 5.7.

**2.** Output Word Rate assuming Modular Clock frequency = 2.4576 MHz (sampling clock frequency = 19.2 kHz)

## SFR Definition 9.2. CPT0MD: Comparator0 Mode Selection

| R | R | R/W | R/W | R | R | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| — | — | CP0RIE | CP0FIE | — | — | CP0MD1 | CP0MD0 | 00000010 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | |

SFR Address: 0x9D

Bits7–6: UNUSED. Read = 00b, Write = don't care.
Bit5: CP0RIE: Comparator0 Rising-Edge Interrupt Enable.
0: Comparator0 Rising-edge interrupt disabled.
1: Comparator0 Rising-edge interrupt enabled.
Bit4: CP0FIE: Comparator0 Falling-Edge Interrupt Enable.
0: Comparator0 Falling-edge interrupt disabled.
1: Comparator0 Falling-edge interrupt enabled.
Bits3–2: UNUSED. Read = 00b, Write = don't care.
Bits1–0: CP0MD1–CP0MD0: Comparator0 Mode Select
These bits select the response time for Comparator0.

| Mode | CP0MD1 | CP0MD0 | Notes |
|---|---|---|---|
| 0 | 0 | 0 | Fastest Response Time |
| 1 | 0 | 1 | — |
| 2 | 1 | 0 | — |
| 3 | 1 | 1 | Lowest Power Consumption |

## 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 C8051F35x family has a superset of all the peripherals included with a standard 8051. See Section “1. System Overview’ on page 17 for more information about the available peripherals. The CIP-51 includes on-chip debug hardware which 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 core includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 50 MIPS Peak Throughput
- 256 Bytes of Internal RAM
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- Integrated Debug Logic

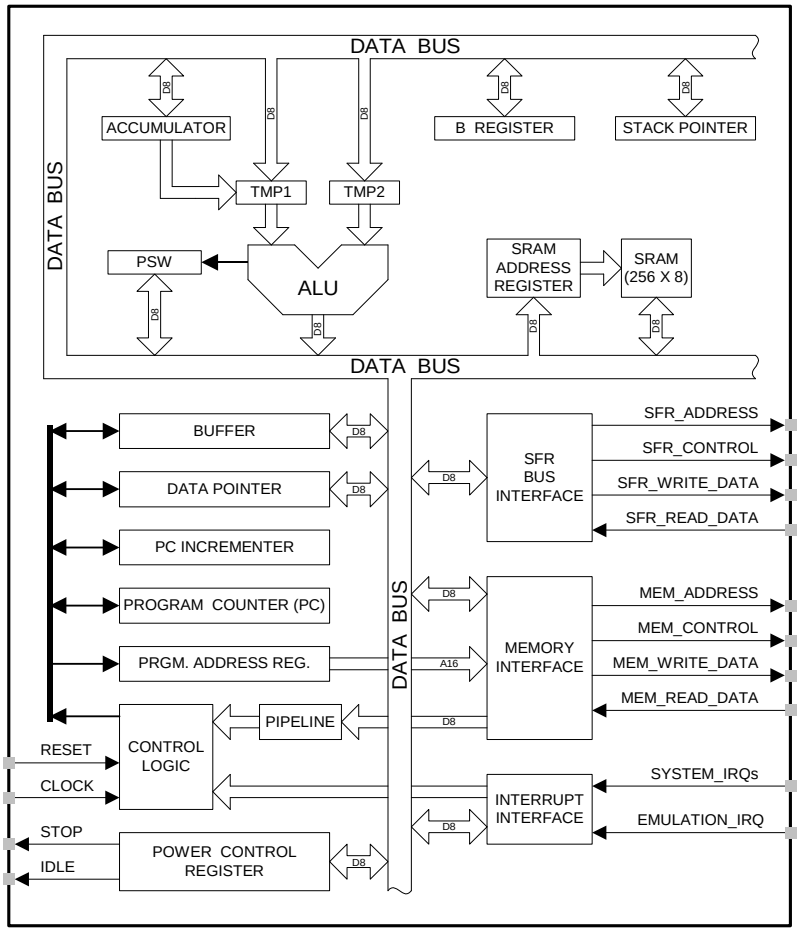


**Figure 10.1. CIP-51 Block Diagram**

## NOTES:

## NOTES:

### 19.4.1. SMBus Configuration Register

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

**Table 19.1. 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 19.1. 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 “22. Timers’ on page 195.

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

**Equation 19.1. 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 19.1. 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 19.2.

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

**Equation 19.2. Typical SMBus Bit Rate**

### 19.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see SFR Definition 19.2). 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 and TXMODE indicate the master/slave state and transmit/receive modes, respectively.

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 on 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 19.3 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.

Table 19.3 lists all sources for hardware changes to the SMB0CN bits. Refer to Table 19.4 for SMBus status decoding using the SMB0CN register.

## 19.5. 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; however, note that the interrupt is generated before the ACK cycle when operating as a receiver, and after the ACK cycle when operating as a transmitter.

### 19.5.1. Master Transmitter Mode

Serial data is transmitted on SDA while the serial clock is output on SCL. 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 19.5 shows a typical Master Transmitter sequence. Two transmit data bytes are shown, though any number of bytes may be transmitted. Notice that the 'data byte transferred' interrupts occur **after** the ACK cycle in this mode.

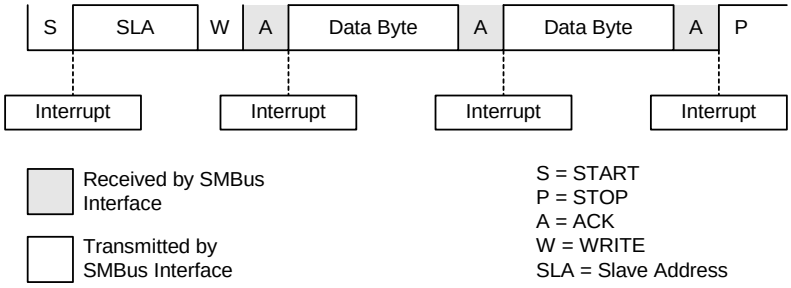


**Figure 19.5. Typical Master Transmitter Sequence**

### 19.5.2. Master Receiver Mode

Serial data is received on SDA while the serial clock is output on SCL. 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. After each byte is received, ACKRQ is set to ‘1’ and an interrupt is generated. Software must write the ACK bit (SMB0CN.1) to define the outgoing acknowledge value (Note: 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 after the last byte is received, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. Note that the interface will switch to Master Transmitter Mode if SMB0DAT is written while an active Master Receiver. Figure 19.6 shows a typical Master Receiver 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.

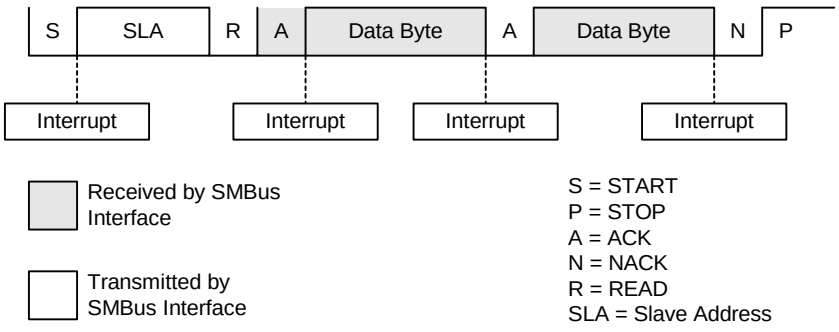


**Figure 19.6. Typical Master Receiver Sequence**

## SFR Definition 20.2. SBUF0: Serial (UART0) Port Data Buffer

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | |

SFR Address: 0x99

Bits7–0: SBUF0[7:0]: Serial Data Buffer Bits 7–0 (MSB–LSB)
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.

## 21.1. Signal Descriptions

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

### 21.1.1. 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.

### 21.1.2. 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.

### 21.1.3. 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.

### 21.1.4. 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 21.2, Figure 21.3, and Figure 21.4 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 137 for general purpose port I/O and crossbar information.

## 21.2. 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 data 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 21.2 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 21.3 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 21.4 shows a connection diagram for a master device in 4-wire master mode and two slave devices.

## SFR Definition 21.4. SPI0DAT: SPI0 Data

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | |

SFR Address: 0xA3

Bits 7–0: SPI0DAT: SPI0 Transmit and Receive Data.
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.

### 23.2.1. 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 and an interrupt request is generated if CCF interrupts are enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. If both CAPPn and CAPNn bits are set to logic 1, then the state of the Port pin associated with CEXn can be read directly to determine whether a rising-edge or falling-edge caused the capture.

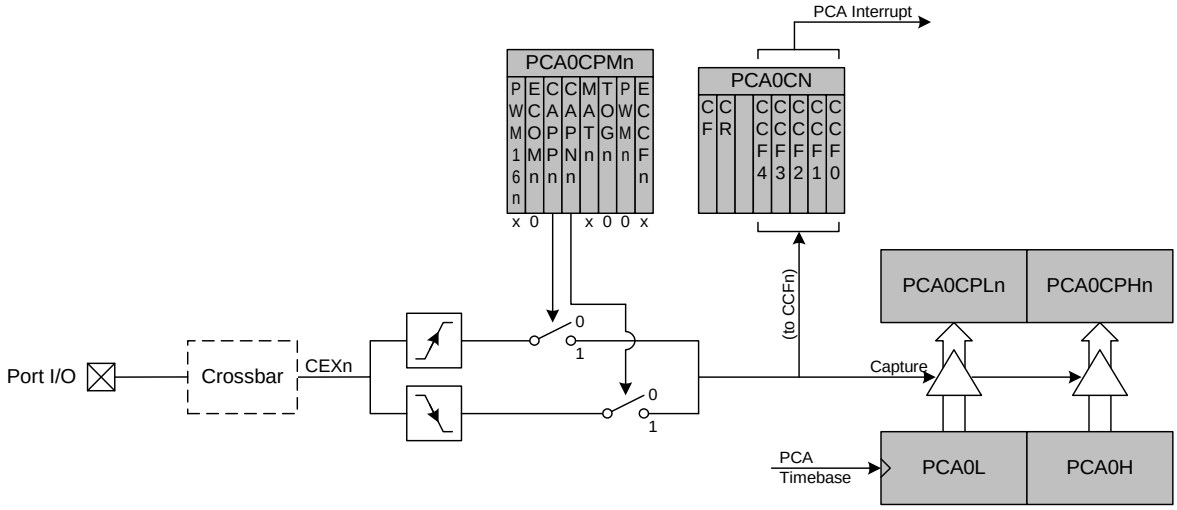


**Figure 23.4. PCA Capture Mode Diagram**

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

### SFR Definition 23.6. PCA0CPLn: PCA Capture Module Low Byte

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | |

SFR Address: PCA0CPL0: 0xE9, PCA0CPL1: 0xEB, PCA0CPL2: 0xED

Bits7–0: PCA0CPLn: PCA Capture Module Low Byte.
The PCA0CPLn register holds the low byte (LSB) of the 16-bit capture module n.

### SFR Definition 23.7. PCA0CPHn: PCA Capture Module High Byte

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | |

SFR Address: PCA0CPH0: 0xEA, PCA0CPH1: 0xEB, PCA0CPH2: 0xEE

Bits7–0: PCA0CPHn: PCA Capture Module High Byte.
The PCA0CPHn register holds the high byte (MSB) of the 16-bit capture module n.

### C2 Register Definition 25.3. REVID: C2 Revision ID

Reset Value
Variable

| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

This read-only register returns the 8-bit revision ID.

### C2 Register Definition 25.4. FPCTL: C2 Flash Programming Control

Reset Value
00000000

| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Bits7–0 FPCTL: Flash Programming Control Register.
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

Reset Value
00000000

| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|
| | | | | | | | |

Bits7–0: FPDAT: C2 Flash Programming Data Register.
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 |