**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 | Not For New Designs |
| Core Processor | 8051 |
| Core Size | 8-Bit |
| Speed | 25MHz |
| Connectivity | EBI/EMI, SMBus (2-Wire/I²C), SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT |
| Number of I/O | 40 |
| Program Memory Size | 32KB (32K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 2.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 3.6V |
| Data Converters | A/D 20x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 48-TQFP |
| Supplier Device Package | 48-TQFP (7x7) |
| Purchase URL | https://www.e-xfl.com/product-detail/silicon-labs/c8051f345-gq |

## 5. 10-Bit ADC (ADC0, C8051F340/1/2/3/4/5/6/7/A/B Only)

The ADC0 subsystem for the C8051F34x devices consists of two analog multiplexers (referred to collectively as AMUX0), and a 200 ksps, 10-bit successive-approximation-register ADC with integrated track-and-hold and programmable window detector. The AMUX0, data conversion modes, and window detector are all configured under software control via the Special Function Registers shown in Figure 5.1. ADC0 operates in both Single-ended and Differential modes, and may be configured to measure voltages at port pins, the Temperature Sensor output, or $V_{DD}$ with respect to a port pin, VREF, or GND. The connection options for AMUX0 are detailed in SFR Definition 5.1 and SFR Definition 5.2. The ADC0 subsystem is enabled only when the AD0EN bit in the ADC0 Control register (ADC0CN) is set to logic 1. The ADC0 subsystem is in low power shutdown when this bit is logic 0.

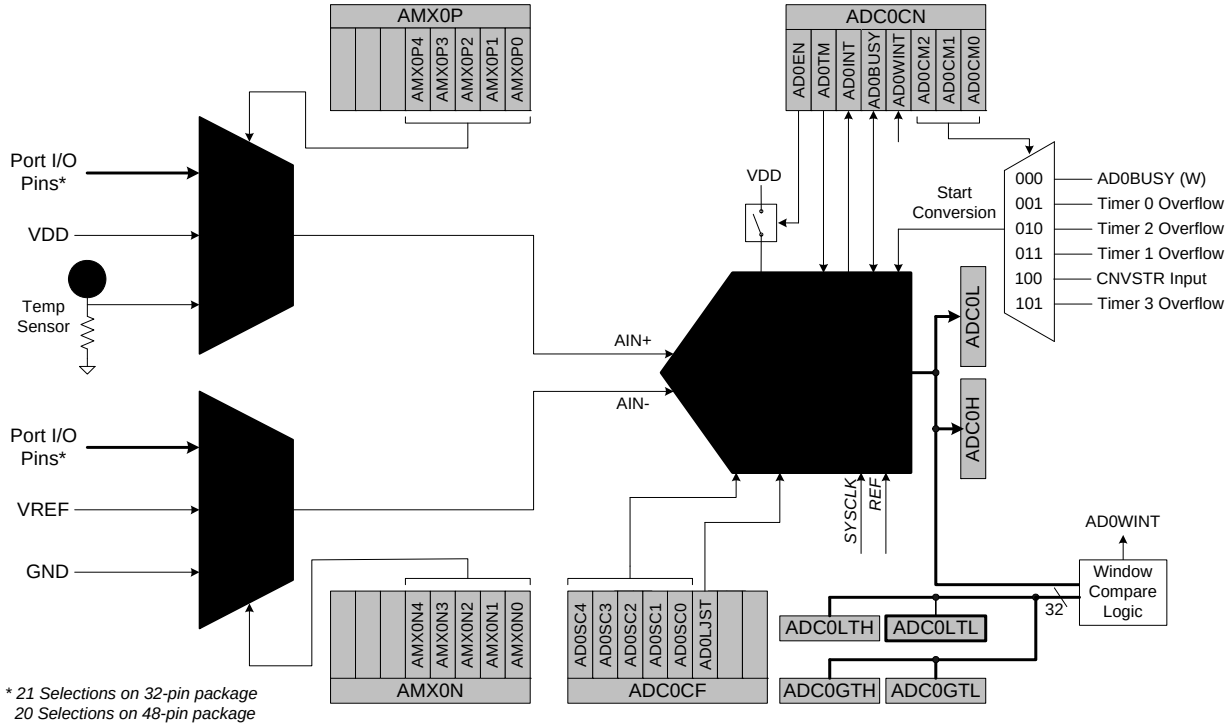



**Figure 5.1. ADC0 Functional Block Diagram**

## 5.2. Temperature Sensor

The temperature sensor transfer function is shown in Figure 5.2. The output voltage ($V_{TEMP}$) is the positive ADC input when the temperature sensor is selected by bits AMX0P4-0 in register AMX0P. Values for the Offset and Slope parameters can be found in Table 5.1.

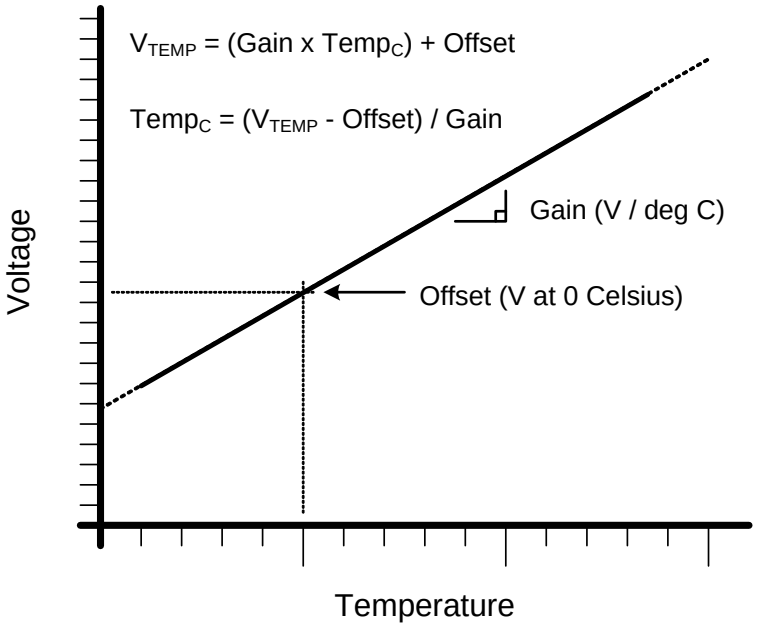



**Figure 5.2. Temperature Sensor Transfer Function**

The uncalibrated temperature sensor output is extremely linear and suitable for relative temperature measurements (see Table 5.1 for linearity specifications). For absolute temperature measurements, offset and/or gain calibration is recommended. Typically a 1-point (offset) calibration includes the following steps:

Step 1. Control/measure the ambient temperature (this temperature must be known).
Step 2. Power the device, and delay for a few seconds to allow for self-heating.
Step 3. Perform an ADC conversion with the temperature sensor selected as the positive input and GND selected as the negative input.
Step 4. Calculate the offset characteristics, and store this value in non-volatile memory for use with subsequent temperature sensor measurements.

Figure 5.3 shows the typical temperature sensor error assuming a 1-point calibration at 25 °C. **Note that parameters which affect ADC measurement, in particular the voltage reference value, will also affect temperature measurement.**

## SFR Definition 5.1. AMX0P: AMUX0 Positive Channel Select

| R | R | R | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| - | - | - | AMX0P4 | AMX0P3 | AMX0P2 | AMX0P1 | AMX0P0 | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xBB |

Bits7–5: UNUSED. Read = 000b; Write = don't care.
Bits4–0: AMX0P4–0: AMUX0 Positive Input Selection

| AMX0P4-0 | ADC0 Positive Input (32-pin Package) | ADC0 Positive Input (48-pin Package) |
|---|---|---|
| 00000 | P1.0 | P2.0 |
| 00001 | P1.1 | P2.1 |
| 00010 | P1.2 | P2.2 |
| 00011 | P1.3 | P2.3 |
| 00100 | P1.4 | P2.5 |
| 00101 | P1.5 | P2.6 |
| 00110 | P1.6 | P3.0 |
| 00111 | P1.7 | P3.1 |
| 01000 | P2.0 | P3.4 |
| 01001 | P2.1 | P3.5 |
| 01010 | P2.2 | P3.7 |
| 01011 | P2.3 | P4.0 |
| 01100 | P2.4 | P4.3 |
| 01101 | P2.5 | P4.4 |
| 01110 | P2.6 | P4.5 |
| 01111 | P2.7 | P4.6 |
| 10000 | P3.0 | RESERVED |
| 10001 | P0.0 | P0.3 |
| 10010 | P0.1 | P0.4 |
| 10011 | P0.4 | P1.1 |
| 10100 | P0.5 | P1.2 |
| 10101 - 11101 | RESERVED | RESERVED |
| 11110 | Temp Sensor | Temp Sensor |
| 11111 | $V_{DD}$ | $V_{DD}$ |

### SFR Definition 5.3. ADC0CF: ADC0 Configuration

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| AD0SC4 | AD0SC3 | AD0SC2 | AD0SC1 | AD0SC0 | AD0LJST | - | - | 11111000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: |
| | | | | | | | | 0xBC |

Bits7–3: AD0SC4–0: ADC0 SAR Conversion Clock Period Bits.
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 Table 5.1.

$$AD0SC = \frac{SYSCLK}{CLK_{SAR}} - 1$$

Bit2: AD0LJST: ADC0 Left Justify Select.
0: Data in ADC0H:ADC0L registers are right-justified.
1: Data in ADC0H:ADC0L registers are left-justified.

Bits1–0: UNUSED. Read = 00b; Write = don't care.

### SFR Definition 5.4. ADC0H: ADC0 Data Word MSB

| 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: |
| | | | | | | | | 0xBE |

Bits7–0: ADC0 Data Word High-Order Bits.
For AD0LJST = 0: Bits 7–2 are the sign extension of Bit1. Bits 1-0 are the upper 2 bits of the 10-bit ADC0 Data Word.
For AD0LJST = 1: Bits 7–0 are the most-significant bits of the 10-bit ADC0 Data Word.

### SFR Definition 5.5. ADC0L: ADC0 Data Word LSB

| 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: |
| | | | | | | | | 0xBD |

Bits7–0: ADC0 Data Word Low-Order Bits.
For AD0LJST = 0: Bits 7–0 are the lower 8 bits of the 10-bit Data Word.
For AD0LJST = 1: Bits 7–6 are the lower 2 bits of the 10-bit Data Word. Bits 5–0 will always read '0'.

### SFR Definition 5.9. ADC0LTH: ADC0 Less-Than Data 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: |
| | | | | | | | | 0xC6 |

Bits7–0: High byte of ADC0 Less-Than Data Word.

### SFR Definition 5.10. ADC0LTL: ADC0 Less-Than Data 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: |
| | | | | | | | | 0xC5 |

Bits7–0: Low byte of ADC0 Less-Than Data Word.

## SFR Definition 7.5. CPT1MX: Comparator1 MUX Selection

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| - | CMX1N2 | CMX1N1 | CMX1N0 | - | CMX1P2 | CMX1P1 | CMX1P0 | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: |
| | | | | | | | | 0x9E |

Bit7: UNUSED. Read = 0b, Write = don't care.
Bits6–4: CMX1N2–CMX1N0: Comparator1 Negative Input MUX Select.
These bits select which Port pin is used as the Comparator1 negative input.

| CMX1N2 | CMX1N1 | CMX1N0 | Negative Input (32-pin Package) | Negative Input (48-pin Package) |
|---|---|---|---|---|
| 0 | 0 | 0 | P1.3 | P2.3 |
| 0 | 0 | 1 | P1.7 | P3.1 |
| 0 | 1 | 0 | P2.3 | P4.0 |
| 0 | 1 | 1 | P2.7 | P4.6 |
| 1 | 0 | 0 | P0.5 | P1.2 |

Bit3: UNUSED. Read = 0b, Write = don't care.
Bits2–0: CMX1P1–CMX1P0: Comparator1 Positive Input MUX Select.
These bits select which Port pin is used as the Comparator1 positive input.

| CMX1P2 | CMX1P1 | CMX1P0 | Positive Input (32-pin Package) | Positive Input (48-pin Package) |
|---|---|---|---|---|
| 0 | 0 | 0 | P1.2 | P2.2 |
| 0 | 0 | 1 | P1.6 | P3.0 |
| 0 | 1 | 0 | P2.2 | P3.7 |
| 0 | 1 | 1 | P2.6 | P4.5 |
| 1 | 0 | 0 | P0.4 | P1.1 |

Note that the port pins used by the comparator depend on the package type (32-pin or 48-pin).

**SFR Definition 9.6. B: B Register**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| B.7 | B.6 | B.5 | B.4 | B.3 | B.2 | B.1 | B.0 | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: |
| | | | | | | | (bit addressable) | 0xF0 |

Bits7–0: B: B Register.
This register serves as a second accumulator for certain arithmetic operations.

## 9.3. Interrupt Handler

The CIP-51 includes an extended interrupt system supporting multiple 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-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.

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.

### 9.3.1. MCU Interrupt Sources and Vectors

The MCU supports multiple 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 9.4 on page 90. 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).

### 9.3.2. External Interrupts

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 “21.1. Timer 0 and Timer 1” on page 235**) select level or edge sensitive. The following table lists the possible configurations.

### 13.7.1. Non-multiplexed Mode

#### 13.7.1.1.16-bit MOVX: EMI0CF[4:2] = ‘101’, ‘110’, or ‘111’.

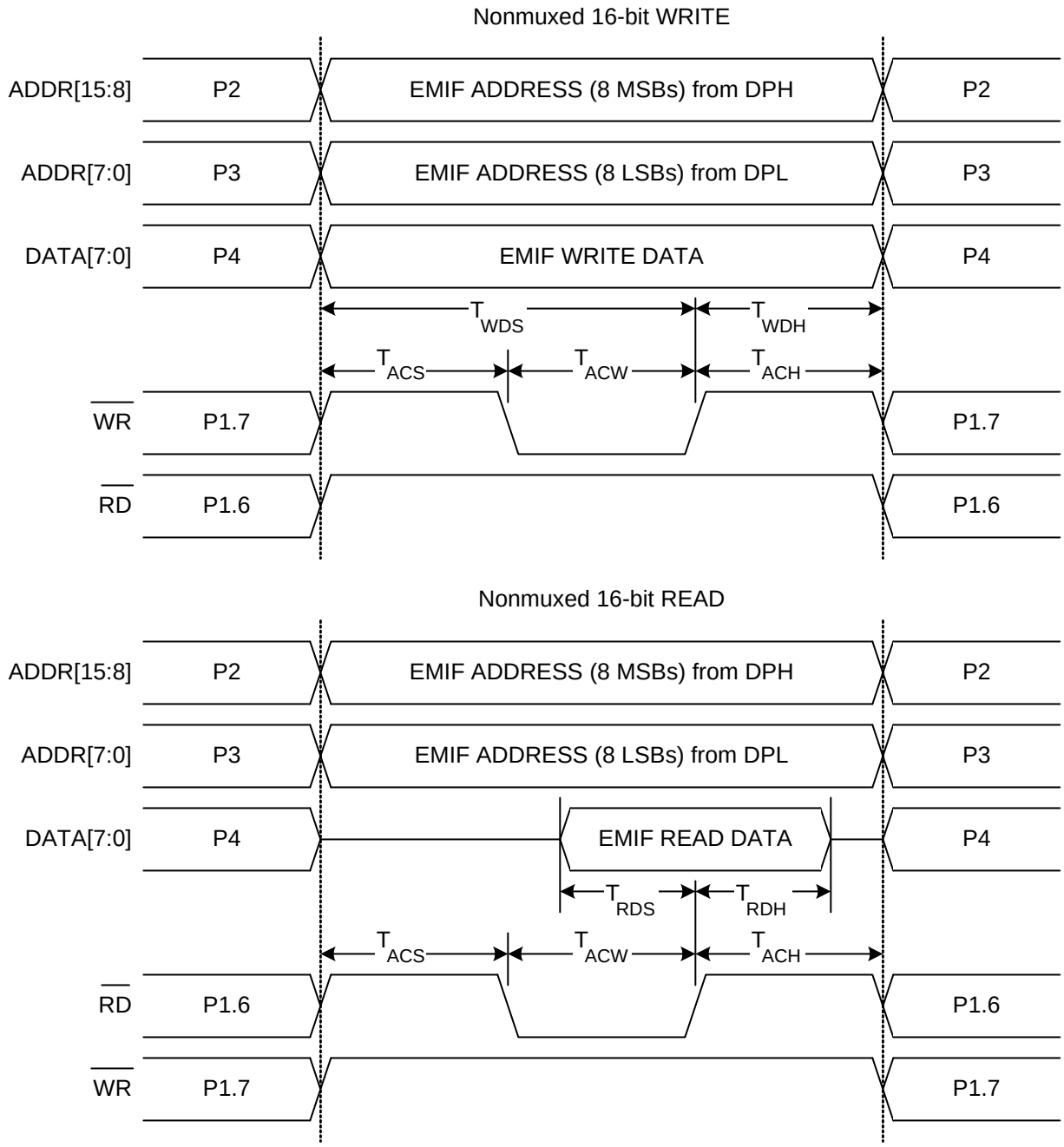



**Figure 13.5. Non-multiplexed 16-bit MOVX Timing**

### 14.3.3. 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 14.1, 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. If the frequency desired is 100 kHz, let R = 246 kΩ and C = 50 pF:

$$f = \frac{1.23(10^3)}{RC} = \frac{1.23(10^3)}{[246 \times 50]} = 0.1\ \text{MHz} = 100\ \text{kHz}$$

Referring to the table in SFR Definition 14.4, the required XFCN setting is 010b. Programming XFCN to a higher setting in RC mode will improve frequency accuracy at an increased external oscillator supply current.

### 14.3.4. External Capacitor Example

If a capacitor is used as an external oscillator for the MCU, the circuit should be configured as shown in Figure 14.1, 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 from the equations below. Assume $V_{DD}$ = 3.0 V and C = 50 pF:

$$f = \frac{KF}{(C \times V_{DD})} = \frac{KF}{(50\ \text{x}\ 3)\text{MHz}}$$

$$f = \frac{KF}{150\ \text{MHz}}$$

If a frequency of roughly 150 kHz is desired, select the K Factor from the table in SFR Definition 14.4 as KF = 22:

$$f = \frac{22}{150} = 0.146\ \text{MHz, or } 146\ \text{kHz}$$

Therefore, the XFCN value to use in this example is 011b.

## 14.5. System and USB Clock Selection

The internal oscillator requires little start-up time and may be selected as the system or USB clock immediately following the OSCICN write that enables the internal oscillator. External crystals and ceramic resonators 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 oscillator is settled. **To avoid reading a false XTLVLD, in crystal mode software should delay at least 1 ms between enabling the external oscillator and checking XTLVLD.** RC and C modes typically require no startup time.

### 14.5.1. 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, USB) 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 4x Clock Multiplier so long as the selected oscillator is enabled and has settled. C8051F340/1/2/3 devices can use the 48 MHz Clock Multiplier output as system clock. See Table 3.1, "Global DC Electrical Characteristics," on page 25 for system clock frequency specifications. When operating with a system clock of greater than 25 MHz (up to 48 MHz), the FLRT bit (FLSCL.4) should be set to '1'. See **Section "10. Prefetch Engine" on page 99** for more details.

### 14.5.2. USB Clock Selection

The USBCLK[2:0] bits in register CLKSEL select which oscillator source is used as the USB clock. The USB clock may be derived from the 4x Clock Multiplier output, a divided version of the internal oscillator, or a divided version of the external oscillator. Note that the USB clock must be 48 MHz when operating USB0 as a Full Speed Function; the USB clock must be 6 MHz when operating USB0 as a Low Speed Function. See SFR Definition 14.6 for USB clock selection options.

Some example USB clock configurations for Full and Low Speed mode are given below:

| **Internal Oscillator** | | |
|---|---|---|
| **Clock Signal** | **Input Source Selection** | **Register Bit Settings** |
| USB Clock | Clock Multiplier | USBCLK = 000b |
| Clock Multiplier Input | Internal Oscillator* | MULSEL = 00b |
| Internal Oscillator | Divide by 1 | IFCN = 11b |
| **External Oscillator** | | |
| **Clock Signal** | **Input Source Selection** | **Register Bit Settings** |
| USB Clock | Clock Multiplier | USBCLK = 000b |
| Clock Multiplier Input | External Oscillator | MULSEL = 01b |
| External Oscillator | Crystal Oscillator Mode<br>12 MHz Crystal | XOSCMD = 110b<br>XFCN = 111b |

***Note:** Clock Recovery must be enabled for this configuration.

| **Internal Oscillator** | | |
|---|---|---|
| **Clock Signal** | **Input Source Selection** | **Register Bit Settings** |
| USB Clock | Internal Oscillator / 2 | USBCLK = 001b |
| Internal Oscillator | Divide by 1 | IFCN = 11b |
| **External Oscillator** | | |
| **Clock Signal** | **Input Source Selection** | **Register Bit Settings** |

## 16.6. Function Addressing

The FADDR register holds the current USB0 function address. Software should write the host-assigned 7-bit function address to the FADDR register when received as part of a SET_ADDRESS command. A new address written to FADDR will not take effect (USB0 will not respond to the new address) until the end of the current transfer (typically following the status phase of the SET_ADDRESS command transfer). The UPDATE bit (FADDR.7) is set to '1' by hardware when software writes a new address to the FADDR register. Hardware clears the UPDATE bit when the new address takes effect as described above.

### USB Register Definition 16.7. FADDR: USB0 Function Address

| R | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| Update | Function Address | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | USB Address: 0x00 |

Bit7: Update: Function Address Update
Set to '1' when software writes the FADDR register. USB0 clears this bit to '0' when the new address takes effect.
0: The last address written to FADDR is in effect.
1: The last address written to FADDR is not yet in effect.

Bits6–0: Function Address
Holds the 7-bit function address for USB0. This address should be written by software when the SET_ADDRESS standard device request is received on Endpoint0. The new address takes effect when the device request completes.

## 16.7. Function Configuration and Control

The USB register POWER (SFR Definition 16.8) is used to configure and control USB0 at the device level (enable/disable, Reset/Suspend/Resume handling, etc.).

**USB Reset:** The USBRST bit (POWER.3) is set to '1' by hardware when Reset signaling is detected on the bus. Upon this detection, the following occur:

1. The USB0 Address is reset (FADDR = 0x00).
2. Endpoint FIFOs are flushed.
3. Control/status registers are reset to 0x00 (E0CSR, EINCSRL, EINCSRH, EOUTCSRL, EOUTCSRH).
4. USB register INDEX is reset to 0x00.
5. All USB interrupts (excluding the Suspend interrupt) are enabled and their corresponding flags cleared.
6. A USB Reset interrupt is generated if enabled.

Writing a '1' to the USBRST bit will generate an asynchronous USB0 reset. All USB registers are reset to their default values following this asynchronous reset.

**Suspend Mode:** With Suspend Detection enabled (SUSEN = '1'), USB0 will enter Suspend Mode when Suspend signaling is detected on the bus. An interrupt will be generated if enabled (SUSINTE = '1'). The Suspend Interrupt Service Routine (ISR) should perform application-specific configuration tasks such as disabling appropriate peripherals and/or configuring clock sources for low power modes. See **Section "14. Oscillators" on page 131** for more details on internal oscillator configuration, including the Suspend

### 16.10.3.Endpoint0 OUT Transactions

When a SETUP request is received that requires the host to transmit data to USB0, one or more OUT requests will be sent by the host. When an OUT packet is successfully received by USB0, hardware will set the OPRDY bit (E0CSR.0) to ‘1’ and generate an Endpoint0 interrupt. Following this interrupt, firmware should unload the OUT packet from the Endpoint0 FIFO and set the SOPRDY bit (E0CSR.6) to ‘1’.

If the amount of data required for the transfer exceeds the maximum packet size for Endpoint0, the data will be split into multiple packets. If the requested data is an integer multiple of the maximum packet size for Endpoint0 (as reported to the host), the host will send a zero-length data packet signaling the end of the transfer.

Upon reception of the first OUT token for a particular control transfer, Endpoint0 is said to be in Receive Mode. In this mode, only OUT tokens should be sent by the host to Endpoint0. The SUEND bit (E0CSR.4) is set to ‘1’ if a SETUP or IN token is received while Endpoint0 is in Receive Mode.

Endpoint0 will remain in Receive mode until:

1. The SIE receives a SETUP or IN token.
2. The host sends a packet less than the maximum Endpoint0 packet size.
3. The host sends a zero-length packet.

Firmware should set the DATAEND bit (E0CSR.3) to ‘1’ when the expected amount of data has been received. The SIE will transmit a STALL condition if the host sends an OUT packet after the DATAEND bit has been set by firmware. An interrupt will be generated with the STSTL bit (E0CSR.2) set to ‘1’ after the STALL is transmitted.

## Table 16.4. USB Transceiver Electrical Characteristics

**$V_{DD}$ = 3.0 to 3.6 V, –40 to +85 °C unless otherwise specified**

| Parameters | Symbol | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|---|
| **Transmitter** | | | | | | |
| Output High Voltage | $V_{OH}$ | | 2.8 | | | V |
| Output Low Voltage | $V_{OL}$ | | | | 0.8 | V |
| Output Crossover Point | $V_{CRS}$ | | 1.3 | | 2.0 | V |
| Output Impedance | $Z_{DRV}$ | Driving High<br>Driving Low | | 38<br>38 | | Ω |
| Pull-up Resistance | $R_{PU}$ | Full Speed (D+ Pull-up)<br>Low Speed (D– Pull-up) | 1.425 | 1.5 | 1.575 | kΩ |
| Output Rise Time | $T_R$ | Low Speed<br>Full Speed | 75<br>4 | | 300<br>20 | ns |
| Output Fall Time | $T_F$ | Low Speed<br>Full Speed | 75<br>4 | | 300<br>20 | ns |
| **Receiver** | | | | | | |
| Differential Input Sensitivity | $V_{DI}$ | \| (D+) – (D–) \| | 0.2 | | | V |
| Differential Input Common Mode Range | $V_{CM}$ | | 0.8 | | 2.5 | V |
| Input Leakage Current | $I_L$ | Pullups Disabled | | <1.0 | | µA |

**Note:** Refer to the USB Specification for timing diagrams and symbol definitions.

**Table 17.4. SMBus Status Decoding (Continued)**

| Mode | Values Read | | | | Current SMbus State | Typical Response Options | Values Written | | |
|---|---|---|---|---|---|---|---|---|---|
| | 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 |
| | | 0 | 0 | 1 | A slave byte was transmitted; ACK received. | Load SMB0DAT with next data byte to transmit. | 0 | 0 | X |
| | | 0 | 1 | X | A Slave byte was transmitted; error detected. | No action required (expecting Master to end transfer). | 0 | 0 | X |
| | 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 was received; ACK requested. | Acknowledge received address. | 0 | 0 | 1 |
| | | | | | | Do not acknowledge received address. | 0 | 0 | 0 |
| | | 1 | 1 | X | Lost arbitration as master; slave address received; ACK requested. | Acknowledge received address. | 0 | 0 | 1 |
| | | | | | | Do not acknowledge received address. | 0 | 0 | 0 |
| | | | | | | Reschedule failed transfer; do not acknowledge received address. | 1 | 0 | 0 |
| | 0010 | 0 | 1 | X | Lost arbitration while attempting a repeated START. | Abort failed transfer. | 0 | 0 | X |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X |
| | 0001 | 1 | 1 | X | Lost arbitration while attempting a STOP. | No action required (transfer complete/aborted). | 0 | 0 | 0 |
| | | 0 | 0 | X | A STOP was detected while addressed as a Slave Transmitter or Slave Receiver. | Clear STO. | 0 | 0 | X |
| | | 0 | 1 | X | Lost arbitration due to a detected STOP. | Abort transfer. | 0 | 0 | X |
| | | | | | | Reschedule failed transfer. | 1 | 0 | X |
| | 0000 | 1 | 0 | X | A slave byte was received; ACK requested. | Acknowledge received byte; Read SMB0DAT. | 0 | 0 | 1 |
| | | | | | | Do not acknowledge received byte. | 0 | 0 | 0 |
| | | 1 | 1 | X | Lost arbitration while transmitting a data byte as master. | Abort failed transfer. | 0 | 0 | 0 |
| | | | | | | Reschedule failed transfer. | 1 | 0 | 0 |

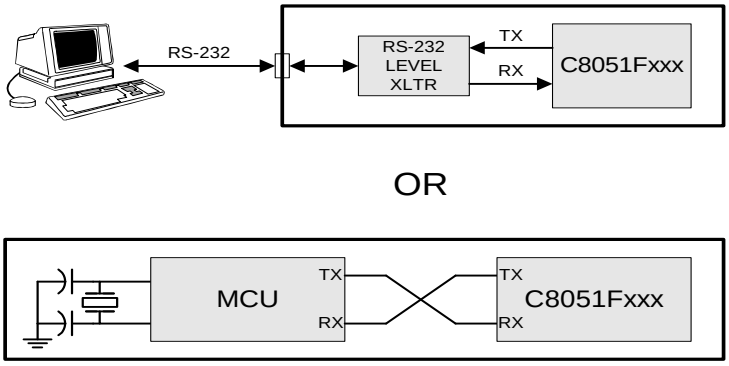



**Figure 18.3. UART Interconnect Diagram**

### 18.2.1. 8-Bit UART

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

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

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

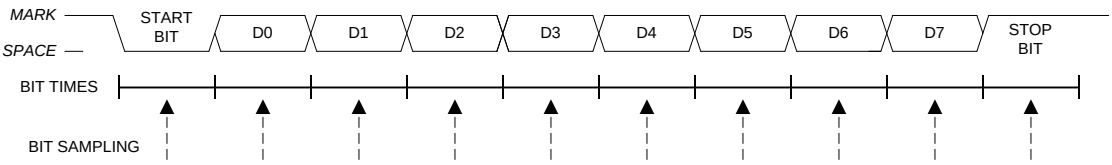



**Figure 18.4. 8-Bit UART Timing Diagram**

## SFR Definition 19.2. SMOD1: UART1 Mode

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| MCE1 | S1PT1 | S1PT0 | PE1 | S1DL1 | S1DL0 | XBE1 | SBL1 | 00001100 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | |

SFR Address: 0xE5

Bit7: MCE1: Multiprocessor Communication Enable.
0: RI will be activated if stop bit(s) are ‘1’.
1: RI will be activated if stop bit(s) and extra bit are ‘1’ (extra bit must be enabled using XBE1).
Note: This function is not available when hardware parity is enabled.

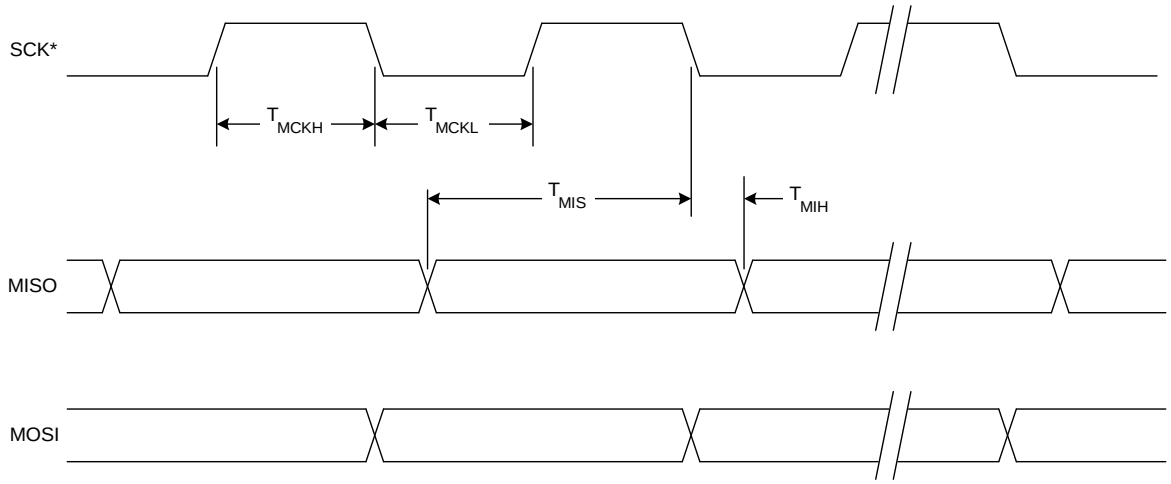
Bits6–5: S1PT[1:0]: Parity Type.
00: Odd
01: Even
10: Mark
11: Space

Bit4: PE1: Parity Enable.
This bit activates hardware parity generation and checking. The parity type is selected by bits S1PT1-0 when parity is enabled.
0: Hardware parity is disabled.
1: Hardware parity is enabled.

Bits3–2: S1DL[1:0]: Data Length.
00: 5-bit data
01: 6-bit data
10: 7-bit data
11: 8-bit data

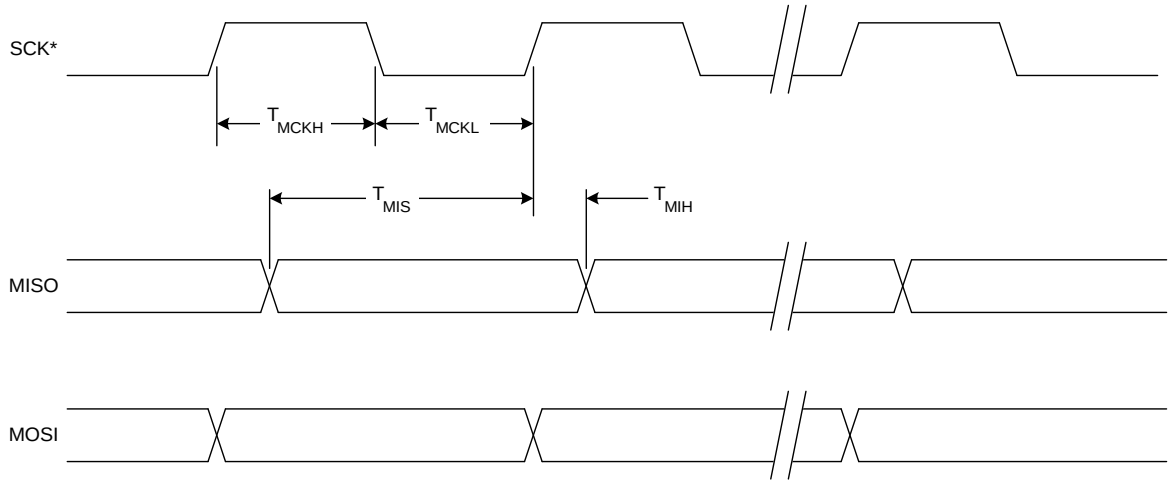
Bit1: XBE1: Extra Bit Enable
When enabled, the value of TBX1 will be appended to the data field.
0: Extra Bit Disabled.
1: Extra Bit Enabled.

Bit0: SBL1: Stop Bit Length
0: Short - Stop bit is active for one bit time.
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).




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

**Figure 20.8. SPI Master Timing (CKPHA = 0)**




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

**Figure 20.9. SPI Master Timing (CKPHA = 1)**

## 22.1. PCA Counter/Timer

The 16-bit PCA counter/timer consists of two 8-bit SFRs: PCA0L and PCA0H. PCA0H is the high byte (MSB) of the 16-bit counter/timer and PCA0L is the low byte (LSB). Reading PCA0L automatically latches the value of PCA0H into a “snapshot” register; the following PCA0H read accesses this “snapshot” register. **Reading the PCA0L Register first guarantees an accurate reading of the entire 16-bit PCA0 counter.** Reading PCA0H or PCA0L does not disturb the counter operation. The CPS2-CPS0 bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 22.1.

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 (Note: PCA0 interrupts must be globally enabled before CF interrupts are recognized. PCA0 interrupts are globally enabled by setting the EA bit (IE.7) and the EPCA0 bit in EIE1 to logic 1). Clearing the CIDL bit in the PCA0MD register allows the PCA to continue normal operation while the CPU is in Idle mode.

**Table 22.1. 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* |

***Note:** External oscillator source divided by 8 is synchronized with the system clock.

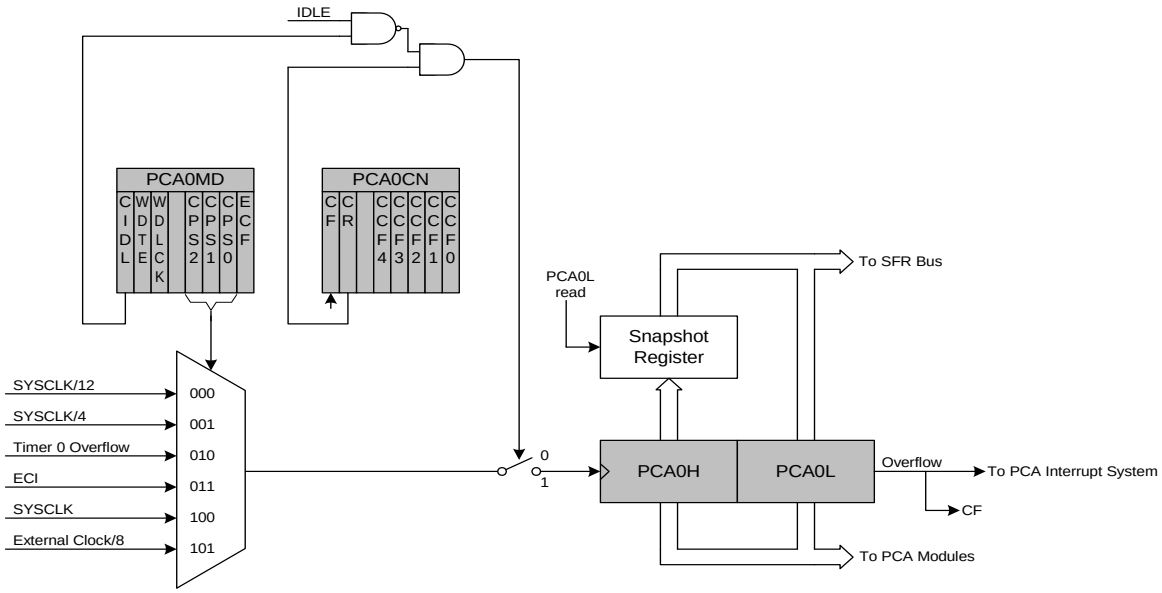



**Figure 22.2. PCA Counter/Timer Block Diagram**

$$Offset = (256 \times PCA0CPL4) + (256 - PCA0L)$$

**Equation 22.4. Watchdog Timer Offset in PCA Clocks**

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH4 and PCA0H. Software may force a WDT reset by writing a '1' to the CCF4 flag (PCA0CN.4) while the WDT is enabled.

### 22.3.2. Watchdog Timer Usage

To configure the WDT, perform the following tasks:

1. Disable the WDT by writing a '0' to the WDTE bit.
2. Select the desired PCA clock source (with the CPS2-CPS0 bits).
3. Load PCA0CPL4 with the desired WDT update offset value.
4. Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
5. Enable the WDT by setting the WDTE bit to '1'.
6. (optional) Lock the WDT (prevent WDT disable until the next system reset) by setting the WDLCK bit to '1'.
7. Write a value to PCA0CPH4 to reload the WDT.

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 PCA0CPL4 defaults to 0x00. Using Equation 22.4, this results in a WDT timeout interval of 256 PCA clocks. Table 22.3 lists some example timeout intervals for typical system clocks.

**Table 22.3. Watchdog Timer Timeout Intervals[1]**

| System Clock (Hz) | PCA0CPL4 | Timeout Interval (ms) |
|---|---|---|
| 12,000,000 | 255 | 65.5 |
| 12,000,000 | 128 | 33.0 |
| 12,000,000 | 32 | 8.4 |
| 24,000,000 | 255 | 32.8 |
| 24,000,000 | 128 | 16.5 |
| 24,000,000 | 32 | 4.2 |
| 1,500,000[2] | 255 | 524.3 |
| 1,500,000[2] | 128 | 264.2 |
| 1,500,000[2] | 32 | 67.6 |
| 32,768 | 255 | 24,000 |
| 32,768 | 128 | 12,093.75 |
| 32,768 | 32 | 3,093.75 |

**Notes:**

1. Assumes SYSCLK / 12 as the PCA clock source, and a PCA0L value of 0x00 at the update time.
2. System Clock reset frequency.

# 23. C2 Interface

C8051F34x 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.

## 23.1. C2 Interface Registers

The following describes the C2 registers necessary to perform Flash programming functions through the C2 interface. All C2 registers are accessed through the C2 interface as described in the C2 Interface Specification.

### C2 Register Definition 23.1. C2ADD: C2 Address

Reset Value: 00000000

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

Bits7–0: 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 |
| 0xAD | Selects the C2 Flash Programming Data register for Data Read/Write instructions |

### C2 Register Definition 23.2. DEVICEID: C2 Device ID

Reset Value: 00001111

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

This read-only register returns the 8-bit device ID: 0x0F (C8051F340/1/2/3/4/5/6/7/8/9/A/B/C/D).