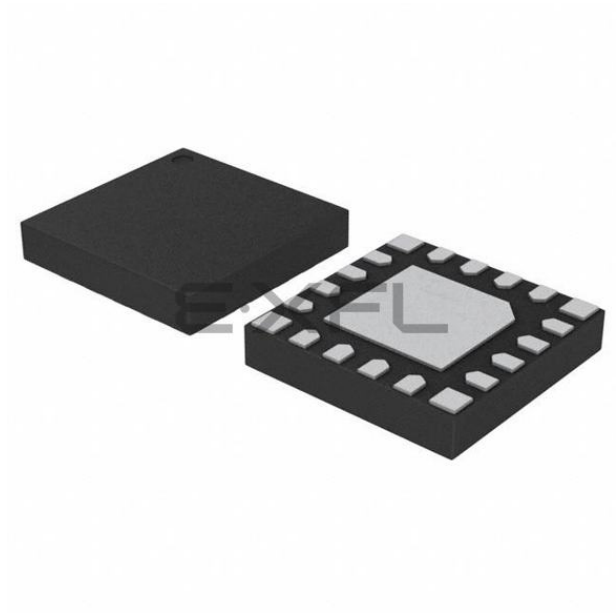


**Welcome to E-XFL.COM**

## 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 | Obsolete |
| Core Processor | CIP-51 8051 |
| Core Size | 8-Bit |
| Speed | 25MHz |
| Connectivity | SMBus (2-Wire/I²C), SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT |
| Number of I/O | 16 |
| Program Memory Size | 4KB (4K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 512 x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | A/D 10x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 20-UFQFN Exposed Pad |
| Supplier Device Package | 20-QFN (3x3) |
| Purchase URL | https://www.e-xfl.com/product-detail/silicon-labs/c8051f982-gm |

# 3. Pinout and Package Definitions

**Table 3.1. Pin Definitions for the C8051F99x-C8051F98x**

| Name | Pin Numbers | | | Type | Description |
|---|---|---|---|---|---|
| | ‘F980/1/2 ‘F983/5 ‘F990/1 -GM | ‘F986/7 ‘F988/9 ‘F996/7 -GM | ‘F986/7 ‘F988/9 ‘F996/7 -GU | | |
| $V_{DD}$ | 4 | 3 | 6 | P In | Power Supply Voltage. Must be 1.8 to 3.6 V. |
| GND | 3, 12 | 2 | 5 | G | Required Ground. |
| $\overline{RST}$/<br><br>C2CK | 5 | 6 | 9 | D I/O<br><br>D I/O | Device Reset. Open-drain output of internal POR or $V_{DD}$ monitor. An external source can initiate a system reset by driving this pin low for at least 15 µs. A 1 kΩ to 5 kΩ pullup to $V_{DD}$ is recommended. See Section “18. Reset Sources” on page 181 Section for a complete description.<br><br>Clock signal for the C2 Debug Interface. |
| P2.7/<br><br>C2D | 6 | 7 | 10 | D I/O<br><br>D I/O | Port 2.7. This pin can only be used as GPIO. The Crossbar cannot route signals to this pin and it cannot be configured as an analog input. See Port I/O Section for a complete description.<br><br>Bi-directional data signal for the C2 Debug Interface. |
| P1.6/<br><br>XTAL3 | 8 | 9 | 12 | D I/O<br><br>A In | Port 1.6. See Port I/O Section for a complete description.<br><br>SmaRTClock Oscillator Crystal Input.<br>See Section 20 for a complete description. |
| P1.7/<br><br>XTAL4 | 7 | 8 | 11 | D I/O<br><br>A Out | Port 1.7. See Port I/O Section for a complete description.<br><br>SmaRTClock Oscillator Crystal Output.<br>See Section 20 for a complete description. |
| P0.0/<br><br>$V_{REF}$* | 2 | 24 | 3 | D I/O or A In<br><br>A In | Port 0.0. See Port I/O Section for a complete description.<br><br>External $V_{REF}$ Input.<br>See Section “5.9. Voltage and Ground Reference Options” on page 88. |
| ***Note:** Available only on the C8051F980/2/6/8 and C8051F990/6 devices. | | | | | |

**Table 3.1. Pin Definitions for the C8051F99x-C8051F98x (Continued)**

| Name | Pin Numbers | | | Type | Description |
|---|---|---|---|---|---|
| | ‘F980/1/2 ‘F983/5 ‘F990/1 -GM | ‘F986/7 ‘F988/9 ‘F996/7 -GM | ‘F986/7 ‘F988/9 ‘F996/7 -GU | | |
| P0.1/ | 1 | 23 | 2 | D I/O or A In | Port 0.1. See Port I/O Section for a complete description. |
| AGND* | | | | G | Optional Analog Ground. See Section “5.9. Voltage and Ground Reference Options” on page 88. |
| P0.2/ | 20 | 22 | 1 | D I/O or A In | Port 0.2. See Port I/O Section for a complete description. |
| XTAL1/ | | | | A In | External Clock Input. This pin is the external oscillator return for a crystal or resonator. See Section “19. Clocking Sources” on page 188. |
| RTCOUT | | | | D Out | Buffered SmaRTClock oscillator output. |
| P0.3/ | 19 | 21 | 24 | D I/O or A In | Port 0.3. See Section “21. Port Input/Output” on page 215 for a complete description. |
| XTAL2/ | | | | A Out | External Clock Output. This pin is the excitation driver for an external crystal or resonator. |
| | | | | D In | External Clock Input. This pin is the external clock input in external CMOS clock mode. |
| | | | | A In | External Clock Input. This pin is the external clock input in capacitor or RC oscillator configurations. See Section “19. Clocking Sources” on page 188 for complete details. |
| WAKEOUT | | | | D Out | Wake-up request signal to wake up external devices. |
| P0.4/ | 18 | 20 | 23 | D I/O or A In | Port 0.4. See Section “21. Port Input/Output” on page 215 for a complete description. |
| TX | | | | D Out | UART TX Pin. See Section “21. Port Input/Output” on page 215. |
| P0.5/ | 17 | 19 | 22 | D I/O or A In | Port 0.5. See Section “21. Port Input/Output” on page 215 for a complete description. |
| RX | | | | D In | UART RX Pin. See Section “21. Port Input/Output” on page 215. |

***Note:** Available only on the C8051F980/2/6/8 and C8051F990/6 devices.

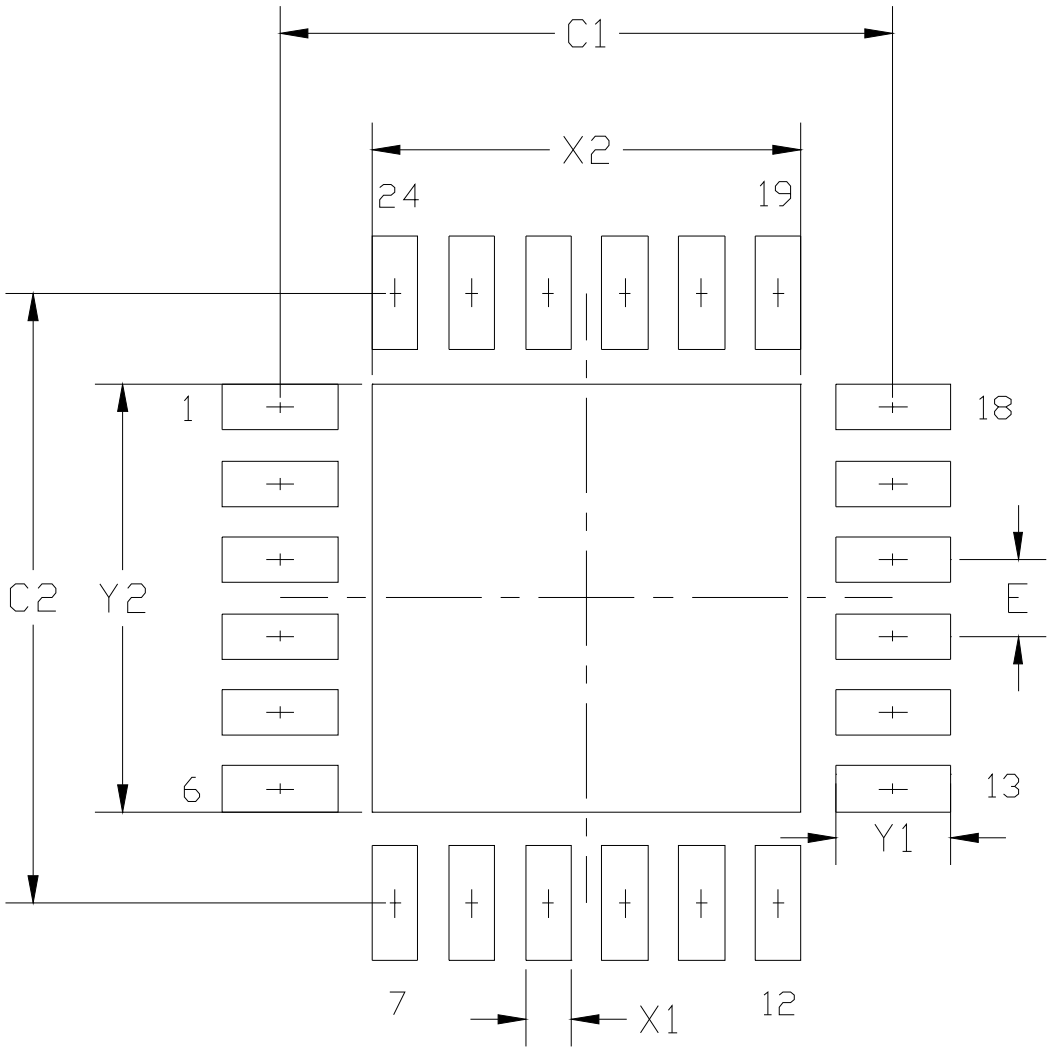



**Figure 3.10. Typical QFN-24 Landing Diagram**

### 5.2.3. Burst Mode

Burst Mode is a power saving feature that allows ADC0 to remain in a low power state between conversions. When Burst Mode is enabled, ADC0 wakes from a low power state, accumulates 1, 4, 8, 16, 32, or 64 using an internal Burst Mode clock (approximately 20 MHz), then re-enters a low power state. Since the Burst Mode clock is independent of the system clock, ADC0 can perform multiple conversions then enter a low power state within a single system clock cycle, even if the system clock is slow (e.g. 32.768 kHz), or suspended.

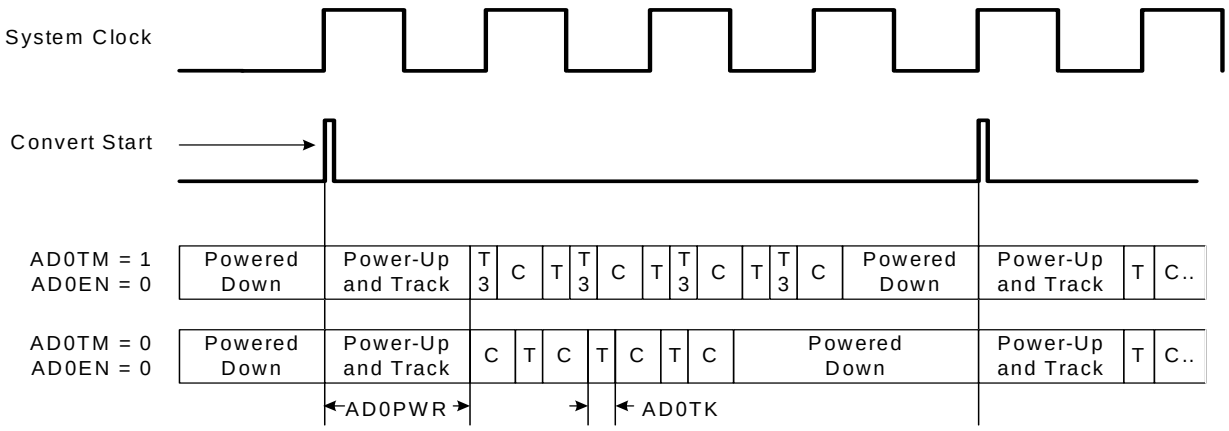
Burst Mode is enabled by setting BURSTEN to logic 1. When in Burst Mode, AD0EN controls the ADC0 idle power state (i.e. the state ADC0 enters when not tracking or performing conversions). If AD0EN is set to logic 0, ADC0 is powered down after each burst. If AD0EN is set to logic 1, ADC0 remains enabled 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 5.3 shows an example of Burst Mode Operation with a slow system clock and a repeat count of 4.

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 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.

In Burst Mode, tracking is determined by the settings in AD0PWR and AD0TK. The default settings for these registers will work in most applications without modification; however, settling time requirements may need adjustment in some applications. Refer to “5.2.4. Settling Time Requirements” on page 71 for more details.

**Notes:**

- Setting AD0TM to 1 will insert an additional 3 SAR clocks of tracking before each conversion, regardless of the settings of AD0PWR and AD0TK.
- 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.




T = Tracking set by AD0TK
T3 = Tracking set by AD0TM (3 SAR clocks)
C = Converting

**Figure 5.3. Burst Mode Tracking Example with Repeat Count Set to 4**

## SFR Definition 8.12. CS0MD2: Capacitive Sense Mode 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0CR[1:0] | | CS0DT[2:0] | | | CS0IA[2:0] | | |
| Type | R/W | | R/W | | | R/W | | |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = 0x0; SFR Address = 0xF3

| Bit | Name | Description |
|---|---|---|
| 7:6 | CS0CR[1:0] | **CS0 Conversion Rate.**<br>These bits control the conversion rate of the CS0 module. See the electrical specifications table for specific timing.<br>00: Conversions last 12 internal CS0 clocks and are 12 bits in length.<br>01: Conversions last 13 internal CS0 clocks and are 13 bits in length.<br>10: Conversions last 14 internal CS0 clocks and are 14 bits in length.<br>11: Conversions last 16 internal CS0 clocks.and are 16 bits in length. |
| 5:3 | CS0DT[2:0] | **CS0 Discharge Time.**<br>These bits adjust the primary CS0 reset time. For most touch-sensitive switches, the default (fastest) value is sufficient. See the discussion in Section 8.13 for more information.<br>000: Discharge time is 0.75 µs (recommended for most switches)<br>001: Discharge time is 1.0 µs<br>010: Discharge time is 1.2 µs<br>011: Discharge time is 1.5 µs<br>100: Discharge time is 2 µs<br>101: Discharge time is 3 µs<br>110: Discharge time is 6 µs<br>111: Discharge time is 12 µs |
| 2:0 | CS0IA[2:0] | **CS0 Output Current Adjustment.**<br>These bits allow the user to adjust the output current used to charge up the capacitive sensor element. For most touch-sensitive switches, the default (highest) current is sufficient. See the discussion in Section 8.13 for more information.<br>000: Full Current<br>001: 1/8 Current<br>010: 1/4 Current<br>011: 3/8 Current<br>100: 1/2 Current<br>101: 5/8 Current<br>110: 3/4 Current<br>111: 7/8 Current |

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

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | ECSEOS | ECSDC | ECSCPT | | ERTC0F | EMAT | EWARN |
| Type | R/W | R/W | R/W | R/W | R | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = All;SFR Address = 0xE7

| Bit | Name | Function |
|---|---|---|
| 7 | Unused | Read = 0b. Write = Don't care. |
| 6 | ECSEOS | **Enable Capacitive Sense End of Scan Interrupt.**<br>0: Disable Capacitive Sense End of Scan interrupt.<br>1: Enable interrupt requests generated by CS0EOS. |
| 5 | ECSDC | **Enable Capacitive Sense Digital Comparator Interrupt.**<br>0: Disable Capacitive Sense Digital Comparator interrupt.<br>1: Enable interrupt requests generated by CS0CMPF. |
| 4 | ECSCPT | **Enable Capacitive Sense Conversion Complete Interrupt.**<br>0: Disable Capacitive Sense Conversion Complete interrupt.<br>1: Enable interrupt requests generated by CS0INT. |
| 3 | Unused | Read = 0b. Write = Don't care. |
| 2 | ERTC0F | **Enable SmaRTClock Oscillator Fail Interrupt.**<br>This bit sets the masking of the SmaRTClock Alarm interrupt.<br>0: Disable SmaRTClock Alarm interrupts.<br>1: Enable interrupt requests generated by SmaRTClock Alarm. |
| 1 | EMAT | **Enable Port Match Interrupts.**<br>This bit sets the masking of the Port Match Event interrupt.<br>0: Disable all Port Match interrupts.<br>1: Enable interrupt requests generated by a Port Match. |
| 0 | EWARN | **Enable Supply Monitor Early Warning Interrupt.**<br>This bit sets the masking of the Supply Monitor Early Warning interrupt.<br>0: Disable the Supply Monitor Early Warning interrupt.<br>1: Enable interrupt requests generated by the Supply Monitor. |

## SFR Definition 15.1. PMU0CF: Power Management Unit Configuration[1,2,3]

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SLEEP | SUSPEND | CLEAR | RSTWK | RTCFWK | RTCAWK | PMATWK | CPT0WK |
| Type | W | W | W | R | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | Varies | Varies | Varies | Varies | Varies |

SFR Page = 0x0; SFR Address = 0xB5

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7 | SLEEP | **Sleep Mode Select** | Writing 1 places the device in Sleep Mode. | N/A |
| 6 | SUSPEND | **Suspend Mode Select** | Writing 1 places the device in Suspend Mode. | N/A |
| 5 | CLEAR | **Wake-up Flag Clear** | Writing 1 clears all wake-up flags. | N/A |
| 4 | RSTWK | **Reset Pin Wake-up Flag** | N/A | Set to 1 if a glitch has been detected on RST. |
| 3 | RTCFWK | **SmaRTClock Oscillator Fail Wake-up Source Enable and Flag** | 0: Disable wake-up on SmaRTClock Osc. Fail.<br>1: Enable wake-up on SmaRTClock Osc. Fail. | Set to 1 if the SmaRT-Clock Oscillator has failed. |
| 2 | RTCAWK | **SmaRTClock Alarm Wake-up Source Enable and Flag** | 0: Disable wake-up on SmaRTClock Alarm.<br>1: Enable wake-up on SmaRTClock Alarm. | Set to 1 if a SmaRTClock Alarm has occurred. |
| 1 | PMATWK | **Port Match Wake-up Source Enable and Flag** | 0: Disable wake-up on Port Match Event.<br>1: Enable wake-up on Port Match Event. | Set to 1 if a Port Match Event has occurred. |
| 0 | CPT0WK | **Comparator0 Wake-up Source Enable and Flag** | 0: Disable wake-up on Comparator0 rising edge.<br>1: Enable wake-up on Comparator0 rising edge. | Set to 1 if Comparator0 rising edge caused the last wake-up. |

**Notes:**

1. Read-modify-write operations (ORL, ANL, etc.) should not be used on this register. Wake-up sources must be re-enabled each time the SLEEP or SUSPEND bits are written to 1.
2. The Low Power Internal Oscillator cannot be disabled and the MCU cannot be placed in Suspend or Sleep Mode if any wake-up flags are set to 1. Software should clear all wake-up sources after each reset and after each wake-up from Suspend or Sleep Modes.
3. PMU0 requires two system clocks to update the wake-up source flags after waking from Suspend mode. The wake-up source flags will read ‘0’ during the first two system clocks following the wake from Suspend mode.

## SFR Definition 15.4. PCON: Power Management Control Register

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

SFR Page = All; SFR Address = 0x87

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:2 | GF[5:0] | **General Purpose Flags** | Sets the logic value. | Returns the logic value. |
| 1 | STOP | **Stop Mode Select** | Writing 1 places the device in Stop Mode. | N/A |
| 0 | IDLE | **Idle Mode Select** | Writing 1 places the device in Idle Mode. | N/A |

## 15.8. Power Management Specifications

See Table 4.5 on page 58 for detailed Power Management Specifications.

# 16. Cyclic Redundancy Check Unit (CRC0)

C8051F99x-C8051F98x devices include a cyclic redundancy check unit (CRC0) that can perform a CRC using a 16-bit polynomial. CRC0 accepts a stream of 8-bit data written to the CRC0IN register. CRC0 posts the 16-bit result to an internal register. The internal result register may be accessed indirectly using the CRC0PNT bits and CRC0DAT register, as shown in Figure 16.1. CRC0 also has a bit reverse register for quick data manipulation.

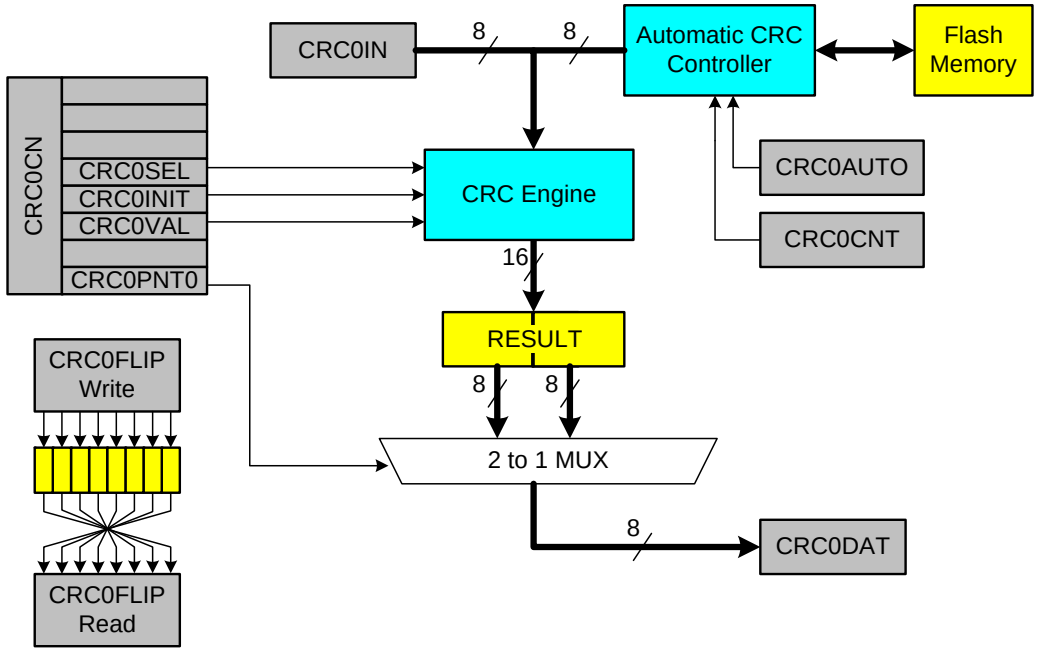



**Figure 16.1. CRC0 Block Diagram**

## 16.1. CRC Algorithm

The C8051F99x-C8051F98x CRC unit generates a CRC result equivalent to the following algorithm:

1. XOR the input with the most-significant bits of the current CRC result. If this is the first iteration of the CRC unit, the current CRC result will be the set initial value (0x0000 or 0xFFFF).

2a. If the MSB of the CRC result is set, shift the CRC result and XOR the result with the selected polynomial.

2b. If the MSB of the CRC result is not set, shift the CRC result.

Repeat Steps 2a/2b for the number of input bits (8). The algorithm is also described in the following example.

## 16.2. Preparing for a CRC Calculation

To prepare CRC0 for a CRC calculation, software should set the initial value of the result. The polynomial used for the CRC computation is 0x1021. The CRC0 result may be initialized to one of two values: 0x0000 or 0xFFFF. The following steps can be used to initialize CRC0.

1. Select the initial result value (Set CRC0VAL to 0 for 0x0000 or 1 for 0xFFFF).
2. Set the result to its initial value (Write 1 to CRC0INIT).

## 16.3. Performing a CRC Calculation

Once CRC0 is initialized, the input data stream is sequentially written to CRC0IN, one byte at a time. The CRC0 result is automatically updated after each byte is written. The CRC engine may also be configured to automatically perform a CRC on one or more 256 byte blocks. The following steps can be used to automatically perform a CRC on Flash memory.

1. Prepare CRC0 for a CRC calculation as shown above.
2. Write the index of the starting page to CRC0AUTO.
3. Set the AUTOEN bit in CRC0AUTO.
4. Write the number of 256 byte blocks to perform in the CRC calculation to CRC0CNT.
5. Write any value to CRC0CN (or OR its contents with 0x00) to initiate the CRC calculation. The CPU will not execute any additional code until the CRC operation completes. **See the note in SFR Definition 16.1. CRC0CN: CRC0 Control for more information on how to properly initiate a CRC calculation.**
6. Clear the AUTOEN bit in CRC0AUTO.
7. Read the CRC result using the procedure below.

## 16.4. Accessing the CRC0 Result

The internal CRC0 result is 16 bits. The CRC0PNT bits select the byte that is targeted by read and write operations on CRC0DAT and increment after each read or write. The calculation result will remain in the internal CR0 result register until it is set, overwritten, or additional data is written to CRC0IN.

## SFR Definition 16.2. CRC0IN: CRC0 Data Input

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CRC0IN[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = All; SFR Address = 0x85

| Bit | Name | Function |
|---|---|---|
| 7:0 | CRC0IN[7:0] | **CRC0 Data Input.**<br>Each write to CRC0IN results in the written data being computed into the existing CRC result according to the CRC algorithm described in Section 16.1 |

## SFR Definition 16.3. CRC0DAT: CRC0 Data Output

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CRC0DAT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = All; SFR Address = 0x86

| Bit | Name | Function |
|---|---|---|
| 7:0 | CRC0DAT[7:0] | **CRC0 Data Output.**<br>Each read or write performed on CRC0DAT targets the CRC result bits pointed to by the CRC0 Result Pointer (CRC0PNT bits in CRC0CN). |

### 20.3.2. Setting a SmaRTClock Alarm

The SmaRTClock alarm function compares the 32-bit value of SmaRTClock Timer to the value of the ALARMn registers. An alarm event is triggered if the SmaRTClock timer is **equal to** the ALARMn registers. If Auto Reset is enabled, the 32-bit timer will be cleared to zero one SmaRTClock cycle after the alarm event.

The SmaRTClock alarm event can be configured to reset the MCU, wake it up from a low power mode, or generate an interrupt. See Section “13. Interrupt Handler” on page 138, Section “15. Power Management” on page 162, and Section “18. Reset Sources” on page 181 for more information.

The following steps can be used to set up a SmaRTClock Alarm:

1. Disable SmaRTClock Alarm Events (RTC0AEN = 0).
2. Set the ALARMn registers to the desired value.
3. Enable SmaRTClock Alarm Events (RTC0AEN = 1).

**Notes:**

1. The ALRM bit, which is used as the SmaRTClock Alarm Event flag, is cleared by disabling SmaRTClock Alarm Events (RTC0AEN = 0).
2. If AutoReset is disabled, disabling (RTC0AEN = 0) then Re-enabling Alarm Events (RTC0AEN = 1) after a SmaRTClock Alarm without modifying ALARMn registers will automatically schedule the next alarm after 2^32 SmaRTClock cycles (approximately 36 hours using a 32.768 kHz crystal).
3. The SmaRTClock Alarm Event flag will remain asserted for a maximum of one SmaRTClock cycle. See Section “15. Power Management” on page 162 for information on how to capture a SmaRTClock Alarm event using a flag which is not automatically cleared by hardware.

## Internal Register Definition 20.4. RTC0CN: SmaRTClock Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RTC0EN | MCLKEN | OSCFAIL | RTC0TR | RTC0AEN | ALRM | RTC0SET | RTC0CAP |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | Varies | 0 | 0 | 0 | 0 | 0 |

SmaRTClock Address = 0x04

| Bit | Name | Function | | |
|---|---|---|---|---|
| 7 | RTC0EN | **SmaRTClock Enable.**<br>Enables/disables the SmaRTClock oscillator and associated bias currents.<br>0: SmaRTClock oscillator disabled.<br>1: SmaRTClock oscillator enabled. | | |
| 6 | MCLKEN | **Missing SmaRTClock Detector Enable.**<br>Enables/disables the missing SmaRTClock detector.<br>0: Missing SmaRTClock detector disabled.<br>1: Missing SmaRTClock detector enabled. | | |
| 5 | OSCFAIL | **SmaRTClock Oscillator Fail Event Flag.**<br>Set by hardware when a missing SmaRTClock detector timeout occurs. Must be cleared by software. The value of this bit is not defined when the SmaRTClock oscillator is disabled. | | |
| 4 | RTC0TR | **SmaRTClock Timer Run Control.**<br>Controls if the SmaRTClock timer is running or stopped (holds current value).<br>0: SmaRTClock timer is stopped.<br>1: SmaRTClock timer is running. | | |
| 3 | RTC0AEN | **SmaRTClock Alarm Enable.**<br>Enables/disables the SmaRTClock alarm function. Also clears the ALRM flag.<br>0: SmaRTClock alarm disabled.<br>1: SmaRTClock alarm enabled. | | |
| 2 | ALRM | **SmaRTClock Alarm Event Flag and Auto Reset Enable.**<br>Reads return the state of the alarm event flag.<br>Writes enable/disable the Auto Reset function. | Read:<br>0: SmaRTClock alarm event flag is de-asserted.<br>1: SmaRTClock alarm event flag is asserted. | Write:<br>0: Disable Auto Reset.<br>1: Enable Auto Reset. |
| 1 | RTC0SET | **SmaRTClock Timer Set.**<br>Writing 1 initiates a SmaRTClock timer set operation. This bit is cleared to 0 by hardware to indicate that the timer set operation is complete. | | |
| 0 | RTC0CAP | **SmaRTClock Timer Capture.**<br>Writing 1 initiates a SmaRTClock timer capture operation. This bit is cleared to 0 by hardware to indicate that the timer capture operation is complete. | | |
| **Note:** The ALRM flag will remain asserted for a maximum of one SmaRTClock cycle. See Section “Power Management” on page 162 for information on how to capture a SmaRTClock Alarm event using a flag which is not automatically cleared by hardware. | | | | |

## SFR Definition 21.3. XBR2: Port I/O Crossbar Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | WEAKPUD | XBARE | | | | | | |
| 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 Page = 0x0; SFR Address = 0xE3

| Bit | Name | Function |
|---|---|---|
| 7 | WEAKPUD | **Port I/O Weak Pullup Disable.**<br>0: Weak Pullups enabled (except for Port I/O pins configured for analog mode).<br>1: Weak Pullups disabled. |
| 6 | XBARE | **Crossbar Enable.**<br>0: Crossbar disabled.<br>1: Crossbar enabled. |
| 5:0 | Unused | Read = 000000b; Write = Don't Care. |
| **Note:** The Crossbar must be enabled (XBARE = 1) to use any Port pin as a digital output. | | |

## SFR Definition 21.17. P1DRV: Port1 Drive Strength

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P1DRV[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = 0xF; SFR Address = 0x9B

| Bit | Name | Function |
|---|---|---|
| 7:0 | P1DRV[7:0] | **Drive Strength Configuration Bits for P1.7–P1.0 (respectively).**<br>Configures digital I/O Port cells to high or low output drive strength.<br>0: Corresponding P1.n Output has low output drive strength.<br>1: Corresponding P1.n Output has high output drive strength. |

## SFR Definition 21.18. P2: Port2

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

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

| Bit | Name | Description | Read | Write |
|---|---|---|---|---|
| 7 | P2 | **Port 2 Data.**<br>Sets the Port latch logic value or reads the Port pin logic state in Port cells configured for digital I/O. | 0: Set output latch to logic LOW.<br>1: Set output latch to logic HIGH. | 0: P2.7 Port pin is logic LOW.<br>1: P2.7 Port pin is logic HIGH. |
| 6:0 | Unused | Read = 0000000b; Write = Don't Care. | | |

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 24.2 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 24.3 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 24.4 shows a connection diagram for a master device in 4-wire master mode and two slave devices.

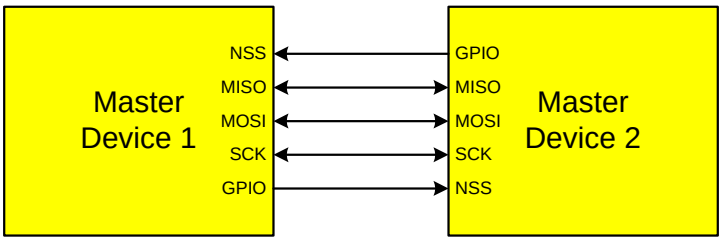



**Figure 24.2. Multiple-Master Mode Connection Diagram**

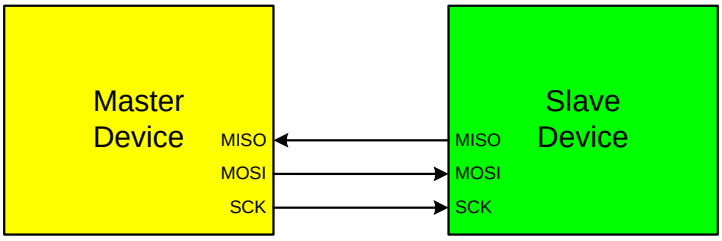



**Figure 24.3. 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram**

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

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

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

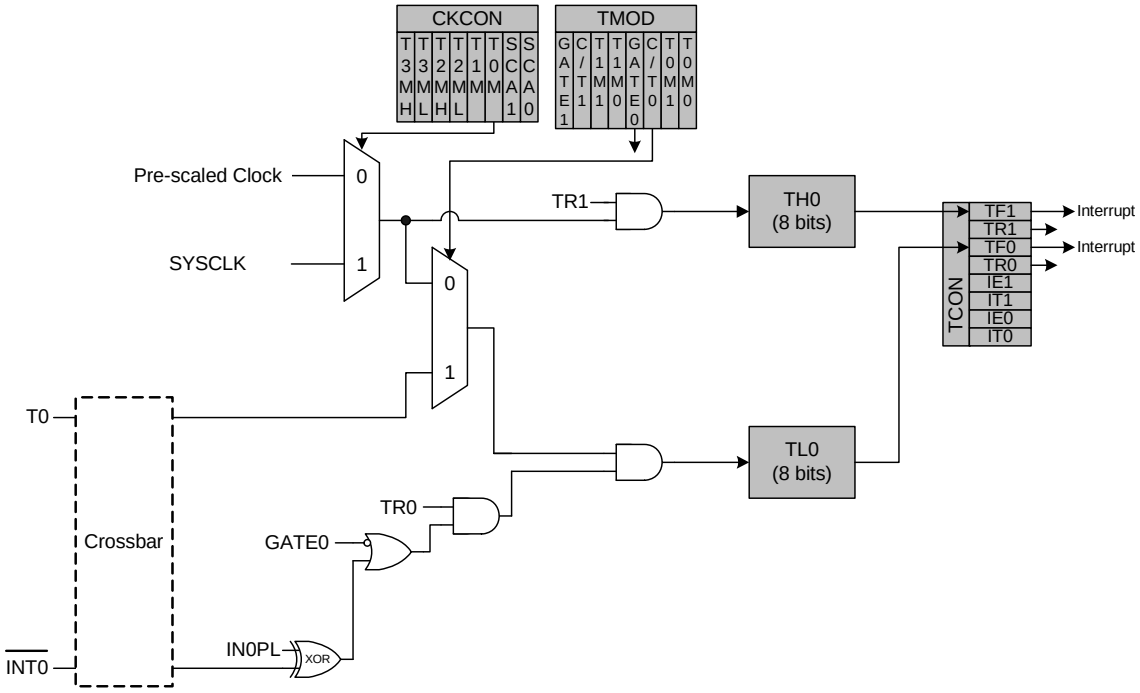



**Figure 25.3. T0 Mode 3 Block Diagram**

## 25.2. 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 can also be used in Capture Mode to measure the SmaRTClock or the Comparator 0 period with respect to another oscillator. The ability to measure the Comparator 0 period with respect to the system clock is makes using Touch Sense Switches very easy.

Timer 2 may be clocked by the system clock, the system clock divided by 12, SmaRTClock divided by 8, or Comparator 0 output. Note that the SmaRTClock divided by 8 and Comparator 0 output is synchronized with the system clock.

### 25.2.1. 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, SmaRTClock divided by 8, or Comparator 0 output. 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 25.4, 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.

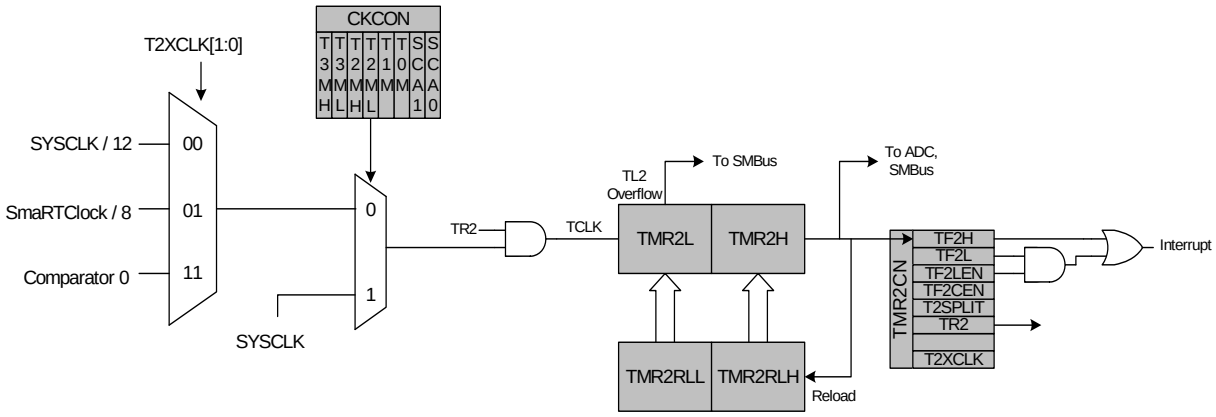



**Figure 25.4. Timer 2 16-Bit Mode Block Diagram**

The 8-bit offset held in PCA0CPH2 is compared to the upper byte of the 16-bit PCA counter. This offset value is the number of PCA0L overflows before a reset. Up to 256 PCA clocks may pass before the first PCA0L overflow occurs, depending on the value of the PCA0L when the update is performed. The total offset is then given (in PCA clocks) by Equation 26.5, where PCA0L is the value of the PCA0L register at the time of the update.

$$Offset = (256 \times PCA0CPL2) + (256 - PCA0L)$$

**Equation 26.5. Watchdog Timer Offset in PCA Clocks**

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH2 and PCA0H. Software may force a WDT reset by writing a 1 to the CCF2 flag (PCA0CN.2) while the WDT is enabled.

### 26.4.2. Watchdog Timer Usage

To configure the WDT, perform the following tasks:

1. Disable the WDT by writing a 0 to the WDTE bit.
2. Select the desired PCA clock source (with the CPS2–CPS0 bits).
3. Load PCA0CPL2 with the desired WDT update offset value.
4. Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
5. Enable the WDT by setting the WDTE bit to 1.
6. Reset the WDT timer by writing to PCA0CPH2.

The PCA clock source and Idle mode select cannot be changed while the WDT is enabled. The watchdog timer is enabled by setting the WDTE or WDLCK bits in the PCA0MD register. When WDLCK is set, the WDT cannot be disabled until the next system reset. If WDLCK is not set, the WDT is disabled by clearing the WDTE bit.

The WDT is enabled following any reset. The PCA0 counter clock defaults to the system clock divided by 12, PCA0L defaults to 0x00, and PCA0CPL2 defaults to 0x00. Using Equation 26.5, this results in a WDT timeout interval of 256 PCA clock cycles, or 3072 system clock cycles. Table 26.3 lists some example timeout intervals for typical system clocks.

**Table 26.3. Watchdog Timer Timeout Intervals[1]**

| System Clock (Hz) | PCA0CPL2 | Timeout Interval (ms) |
|---|---|---|
| 24,500,000 | 255 | 32.1 |
| 24,500,000 | 128 | 16.2 |
| 24,500,000 | 32 | 4.1 |
| 3,062,500[2] | 255 | 257 |
| 3,062,500[2] | 128 | 129.5 |
| 3,062,500[2] | 32 | 33.1 |
| 32,000 | 255 | 24576 |
| 32,000 | 128 | 12384 |
| 32,000 | 32 | 3168 |

**Notes:**
1. Assumes SYSCLK/12 as the PCA clock source, and a PCA0L value of 0x00 at the update time.
2. Internal SYSCLK reset frequency = Internal Oscillator divided by 8.

## 27.2. 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 27.1.

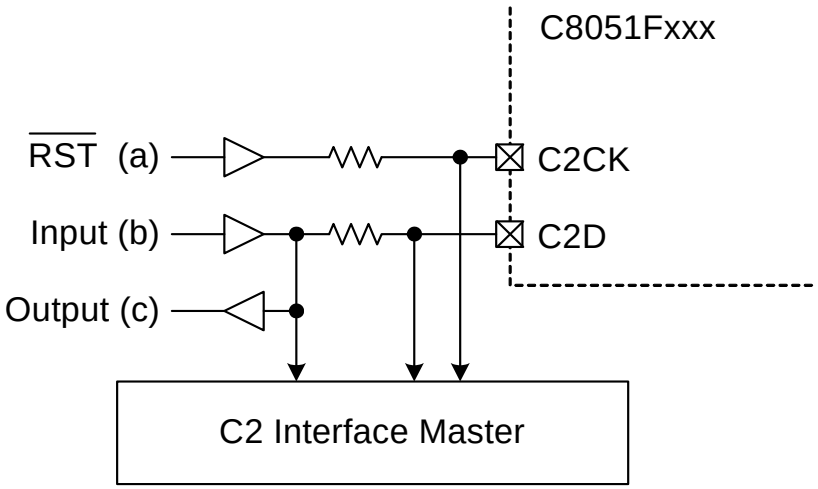



**Figure 27.1. Typical C2 Pin Sharing**

The configuration in Figure 27.1 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.