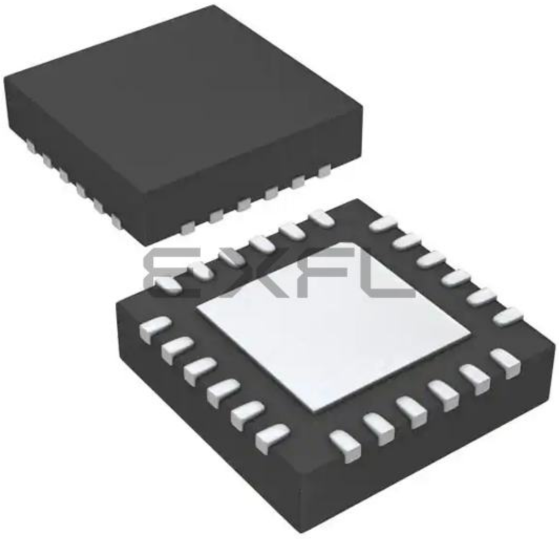
# Silicon Labs - C8051T623-GM Datasheet



**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 | 8051 |
| Core Size | 8-Bit |
| Speed | 48 MIPS |
| Connectivity | I²C, SPI, UART/USART, USB |
| Peripherals | POR, PWM, WDT |
| Number of I/O | 16 |
| Program Memory Size | 8KB (8K x 8) |
| Program Memory Type | OTP |
| EEPROM Size | - |
| RAM Size | 1.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 5.25V |
| Data Converters | - |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 24-WFQFN Exposed Pad |
| Supplier Device Package | 24-QFN (4x4) |
| Purchase URL | https://www.e-xfl.com/product-detail/silicon-labs/c8051t623-gm |

## Table of Contents

# 9. Prefetch Engine

The C8051T622/3 and C8051T326/7 family of devices incorporate a 2-byte prefetch engine. Because the access time of the EPROM memory is 40 ns, and the minimum instruction time is roughly 20 ns, the prefetch engine is necessary for full-speed code execution. Instructions are read from EPROM memory two bytes at a time by the prefetch engine and given to the CIP-51 processor core to execute. When running linear code (code without any jumps or branches), the prefetch engine allows instructions to be executed at full speed. When a code branch occurs, the processor may be stalled for up to two clock cycles while the next set of code bytes is retrieved from EPROM memory.

**Note:** The prefetch engine should be disabled when the device is in suspend mode to save power.

## SFR Definition 9.1. PFE0CN: Prefetch Engine Control

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

SFR Address = 0xAF

| **Bit** | **Name** | **Function** |
|---|---|---|
| 7:6 | Unused | Unused. Read = 00b, Write = don’t care. |
| 5 | PFEN | **Prefetch Enable.**<br>This bit enables the prefetch engine.<br>0: Prefetch engine is disabled.<br>1: Prefetch engine is enabled. |
| 4:0 | Unused | Unused. Read = 00000b. Write = don’t care. |

## 10. Memory Organization

The memory organization of the CIP-51 System Controller is similar to that of a standard 8051. There are two separate memory spaces: program memory and data memory. Program and data memory share the same address space but are accessed via different instruction types. The memory organization of the C8051T622/3 and C8051T326/7 device family is shown in Figure 10.1

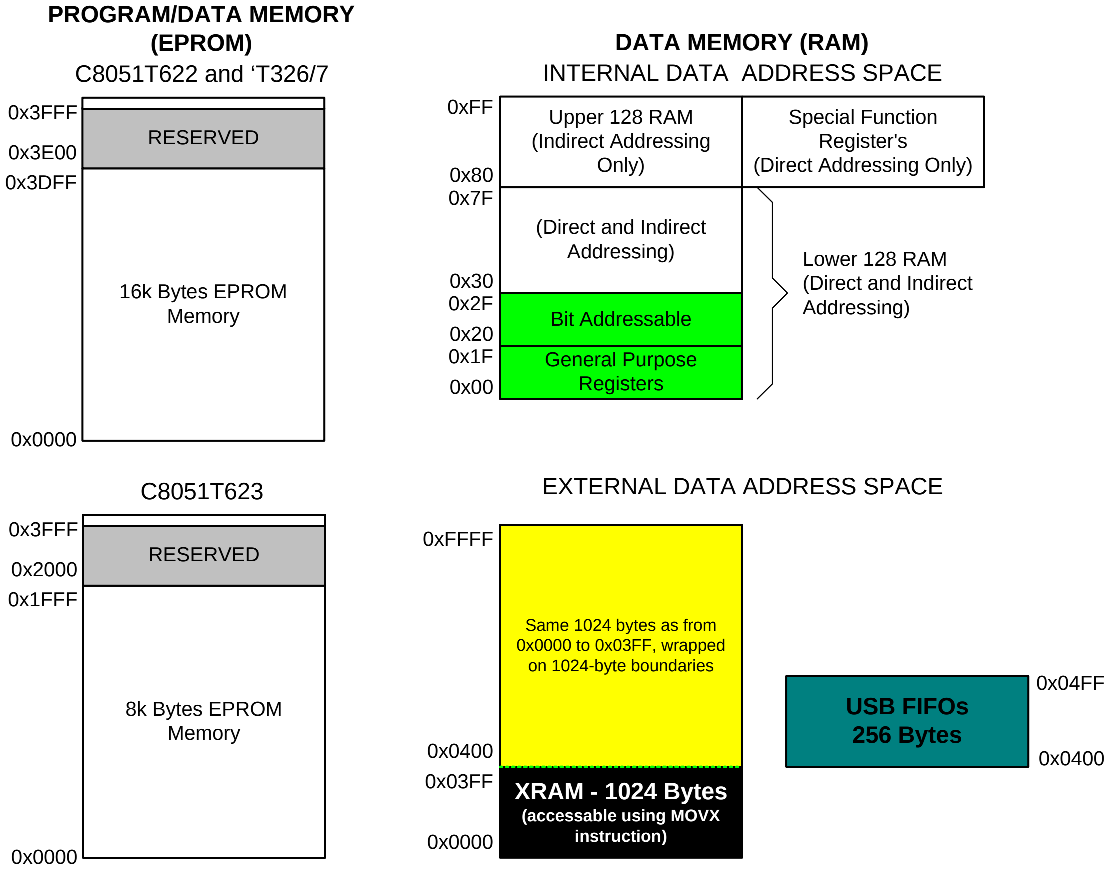

PROGRAM/DATA MEMORY
(EPROM)
C8051T622 and ‘T326/7
0x3FFF
0x3E00
0x3DFF
RESERVED
16k Bytes EPROM Memory
0x0000
C8051T623
0x3FFF
0x2000
0x1FFF
RESERVED
8k Bytes EPROM Memory
0x0000
DATA MEMORY (RAM)
INTERNAL DATA ADDRESS SPACE
0xFF
0x80
0x7F
0x30
0x2F
0x20
0x1F
0x00
Upper 128 RAM (Indirect Addressing Only)
Special Function Register's (Direct Addressing Only)
(Direct and Indirect Addressing)
Bit Addressable
General Purpose Registers
Lower 128 RAM (Direct and Indirect Addressing)
EXTERNAL DATA ADDRESS SPACE
0xFFFF
Same 1024 bytes as from 0x0000 to 0x03FF, wrapped on 1024-byte boundaries
0x0400
0x03FF
XRAM - 1024 Bytes (accessable using MOVX instruction)
0x0000
USB FIFOs 256 Bytes
0x04FF
0x0400

**Figure 10.1. Memory Map**

### 10.1. Program Memory

The CIP-51 core has a 64 kB program memory space. The C8051T622/3 and C8051T326/7 implements 16384 or 8192 bytes of this program memory space as in-system byte-programmable EPROM organized in a contiguous block from addresses 0x0000 to 0x3FFF or 0x0000 to 0x1FFF.

**Note:** 512 bytes (0x3E00 – 0x3FFF) of this memory are reserved for factory use and are not available for user program storage. C2 Register Definition 10.2 shows the program memory maps for C8051T622/3 and C8051T326/7 devices.

## SFR Definition 12.2. IP: Interrupt Priority

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

SFR Address = 0xB8; Bit-Addressable

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

example, and instructions which force a Software Reset. A global search on "RSTSRC" can quickly verify this.

### 13.3.2. PSWE Maintenance

7. Reduce the number of places in code where the PSWE bit (PSCTL.0) is set to a 1. There should be exactly one routine in code that sets PSWE to a 1 to write EPROM bytes.
8. Minimize the number of variable accesses while PSWE is set to a 1. Handle pointer address updates and loop variable maintenance outside the "PSWE = 1;... PSWE = 0;" area.
9. Disable interrupts prior to setting PSWE to a '1' and leave them disabled until after PSWE has been reset to '0'. Any interrupts posted during the EPROM write operation will be serviced in priority order after the EPROM operation has been completed and interrupts have been re-enabled by software.
10. Make certain that the EPROM write pointer variables are not located in XRAM. See your compiler documentation for instructions regarding how to explicitly locate variables in different memory areas.
11. Add address bounds checking to the routines that write EPROM memory to ensure that a routine called with an illegal address does not result in modification of the EPROM.

### 13.3.3. System Clock

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

## 13.4. Program Memory CRC

A CRC engine is included on-chip which provides a means of verifying EPROM contents once the device has been programmed. The CRC engine is available for EPROM verification even if the device is fully read and write locked, allowing for verification of code contents at any time.

The CRC engine is operated through the C2 debug and programming interface, and performs 16-bit CRCs on individual 256-Byte blocks of program memory, or a 32-bit CRC on the entire memory space. To prevent hacking and extrapolation of security-locked source code, the CRC engine will only allow CRCs to be performed on contiguous 256-Byte blocks beginning on 256-Byte boundaries (lowest 8-bits of address are 0x00). For example, the CRC engine can perform a CRC for locations 0x0400 through 0x04FF, but it cannot perform a CRC for locations 0x0401 through 0x0500, or on block sizes smaller or larger than 256 Bytes.

### 13.4.1. Performing 32-bit CRCs on Full EPROM Content

A 32-bit CRC on the entire EPROM space is initiated by writing to the CRC1 byte over the C2 interface. The CRC calculation begins at address 0x0000, and ends at the end of user EPROM space. The EPBusy bit in register C2ADD will be set during the CRC operation, and cleared once the operation is complete. The 32-bit results will be available in the CRC3-0 registers. CRC3 is the MSB, and CRC0 is the LSB. The polynomial used for the 32-bit CRC calculation is 0x04C11DB7. Note: If a 16-bit CRC has been performed since the last device reset, a device reset should be initiated before performing a 32-bit CRC operation.

### 13.4.2. Performing 16-bit CRCs on 256-Byte EPROM Blocks

A 16-bit CRC of individual 256-byte blocks of EPROM can be initiated by writing to the CRC0 byte over the C2 interface. The value written to CRC0 is the high byte of the beginning address for the CRC. For example, if CRC0 is written to 0x02, the CRC will be performed on the 256-bytes beginning at address 0x0200, and ending at address 0x2FF. The EPBusy bit in register C2ADD will be set during the CRC operation, and cleared once the operation is complete. The 16-bit results will be available in the CRC1-0 registers. CRC1 is the MSB, and CRC0 is the LSB. The polynomial for the 16-bit CRC calculation is 0x1021.

## 15.6. EPROM Error Reset

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

- Programming hardware attempts to write or read an EPROM location which is above the user code space address limit.
- An EPROM read from firmware is attempted above user code space. This occurs when a MOVC operation is attempted above the user code space address limit.
- A Program read is attempted above user code space. This occurs when user code attempts to branch to an address above the user code space address limit.

The MEMERR bit (RSTSRC.6) is set following an EPROM error reset. The state of the $\overline{\text{RST}}$ pin is unaffected by this reset.

## 15.7. Software Reset

Software may force a reset by writing a 1 to the SWRSF bit (RSTSRC.4). The SWRSF bit will read 1 following a software forced reset. The state of the $\overline{\text{RST}}$ pin is unaffected by this reset.

## 15.8. USB Reset

Writing 1 to the USBRSF bit in register RSTSRC selects USB0 as a reset source. With USB0 selected as a reset source, a system reset will be generated when either of the following occur:

1. RESET signaling is detected on the USB network. The USB Function Controller (USB0) must be enabled for RESET signaling to be detected. See Section “18. Universal Serial Bus Controller (USB0)” on page 116 for information on the USB Function Controller.
2. A falling or rising voltage on the VBUS pin matches the edge polarity selected by the VBPOL bit in register REG01CN. See Section “7. Voltage Regulators (REG0 and REG1)” on page 35 for details on the VBUS detection circuit.

The USBRSF bit will read 1 following a USB reset. The state of the /RST pin is unaffected by this reset.

## 16. Oscillators and Clock Selection

C8051T622/3 and C8051T326/7 devices include a programmable internal high-frequency oscillator, a programmable internal low-frequency oscillator, and an external oscillator drive circuit. The internal high-frequency oscillator can be enabled/disabled and calibrated using the OSCICN and OSCICL registers, as shown in Figure 16.1. The internal low-frequency oscillator can be enabled/disabled and calibrated using the OSCLCN register. The system clock can be sourced by the external oscillator circuit or either internal oscillator. Both internal oscillators offer a selectable post-scaling feature. The USB clock (USBCLK) can be derived from the internal oscillators or external oscillator.

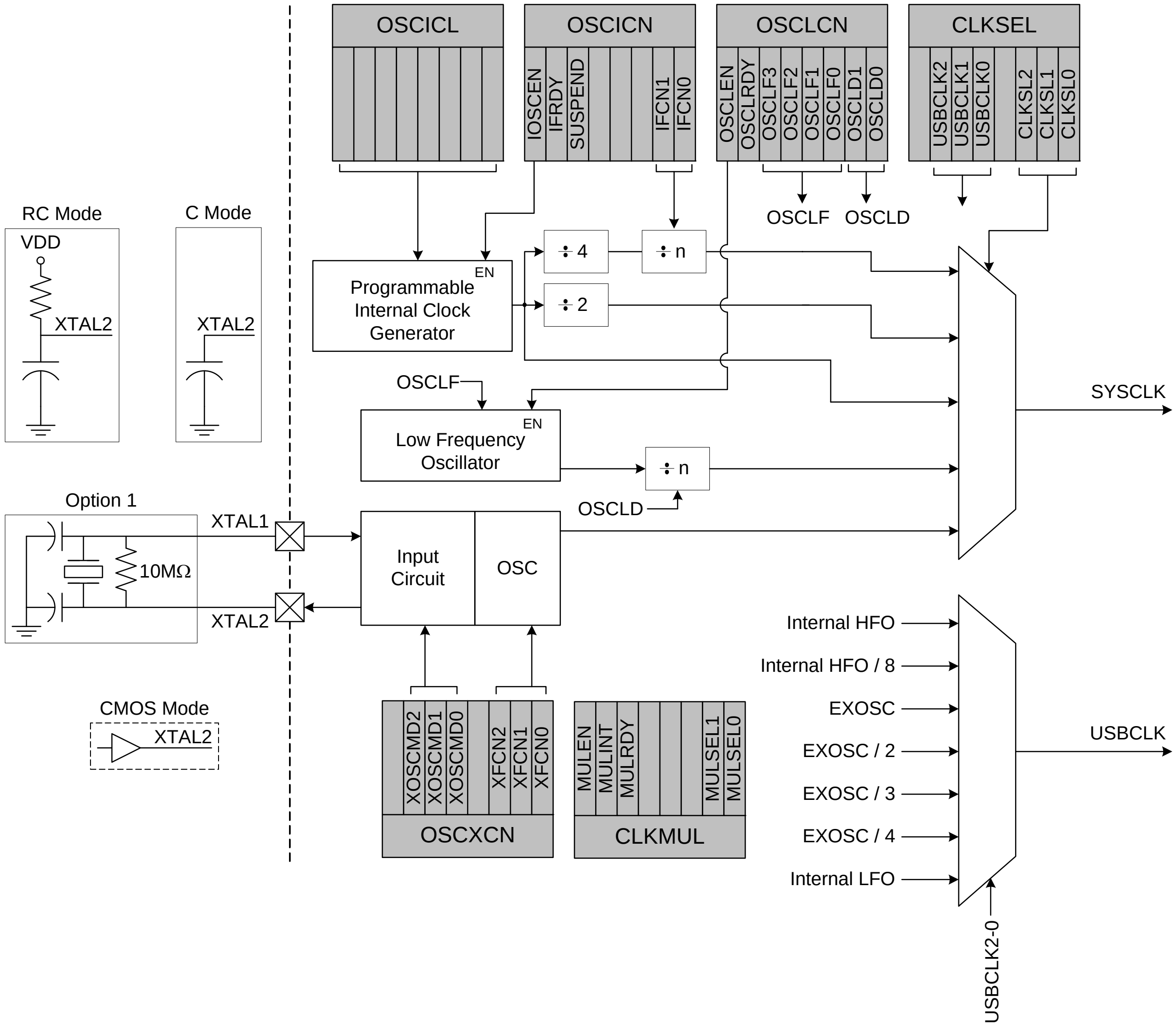


**Figure 16.1. Oscillator Options**

## 16.4. Clock Multiplier

The C8051T622/3 and C8051T326/7 device includes a 48 MHz high-frequency oscillator instead of a 12 MHz oscillator and a 4x Clock Multiplier, so the USB0 module can be run directly from the internal high-frequency oscillator. For compatibility with the Flash development platform, however, the CLKMUL register (SFR Definition 16.4) behaves as if the Clock Multiplier is present.

## SFR Definition 16.4. CLKMUL: Clock Multiplier Control

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

SFR Address = 0xB9

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7 | MULEN | **Clock Multiplier Enable Bit.**<br>0: Clock Multiplier disabled.<br>1: Clock Multiplier enabled.<br>This bit always reads 1. | | |
| 6 | MULINIT | **Clock Multiplier Initialize Bit.** | This bit should be a 0 when the Clock Multiplier is enabled. Once enabled, writing a 1 to this bit will initialize the Clock Multiplier. | The MULRDY bit reads 1 when the Clock Multiplier is stabilized.<br>This bit always reads 1. |
| 5 | MULRDY | **Clock Multiplier Ready Bit.**<br>0: Clock Multiplier not ready.<br>1: Clock Multiplier ready (locked).<br>This bit always reads 1. | | |
| 4:2 | Unused | Unused. Read = 000b; Write = Don’t Care | | |
| 1:0 | MULSEL[1:0] | **Clock Multiplier Input Select Bits.**<br>These bits select the clock supplied to the Clock Multiplier.<br>00: Internal High-Frequency Oscillator<br>01: External Oscillator<br>10: External Oscillator/2<br>11: Reserved.<br>These bits always read 00. | | |

## SFR Definition 17.1. XBR0: Port I/O Crossbar Register 0

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

SFR Address = 0xE1

| **Bit** | **Name** | **Function** |
|---|---|---|
| 7:4 | Unused | Unused. Read = 0000b. Write = don’t care. |
| 3 | SYSCKE | **/SYSCLK Output Enable.**<br>The source of this signal is determined by the OUTCLK bit (see SFR Definition 16.1).<br>0: /SYSCLK unavailable at Port pin.<br>1: /SYSCLK output routed to Port pin. |
| 2 | SMB0E | **SMBus I/O Enable.**<br>0: SMBus I/O unavailable at Port pins.<br>1: SMBus I/O routed to Port pins. |
| 1 | SPI0E | **SPI I/O Enable.**<br>0: SPI I/O unavailable at Port pins.<br>1: SPI I/O routed to Port pins. Note that the SPI can be assigned either 3 or 4 GPIO pins. |
| 0 | URT0E | **UART I/O Output Enable.**<br>0: UART I/O unavailable at Port pin.<br>1: UART TX0, RX0 routed to Port pins P0.4 and P0.5. |

## 18.3. USB Register Access

The USB0 controller registers listed in Table 18.2 are accessed through two SFRs: USB0 Address (USB0ADR) and USB0 Data (USB0DAT). The USB0ADR register selects which USB register is targeted by reads/writes of the USB0DAT register. See Figure 18.2.

Endpoint control/status registers are accessed by first writing the USB register INDEX with the target endpoint number. Once the target endpoint number is written to the INDEX register, the control/status registers associated with the target endpoint may be accessed. See the "Indexed Registers" section of Table 18.2 for a list of endpoint control/status registers.

**Important Note**: The USB clock must be active when accessing USB registers.

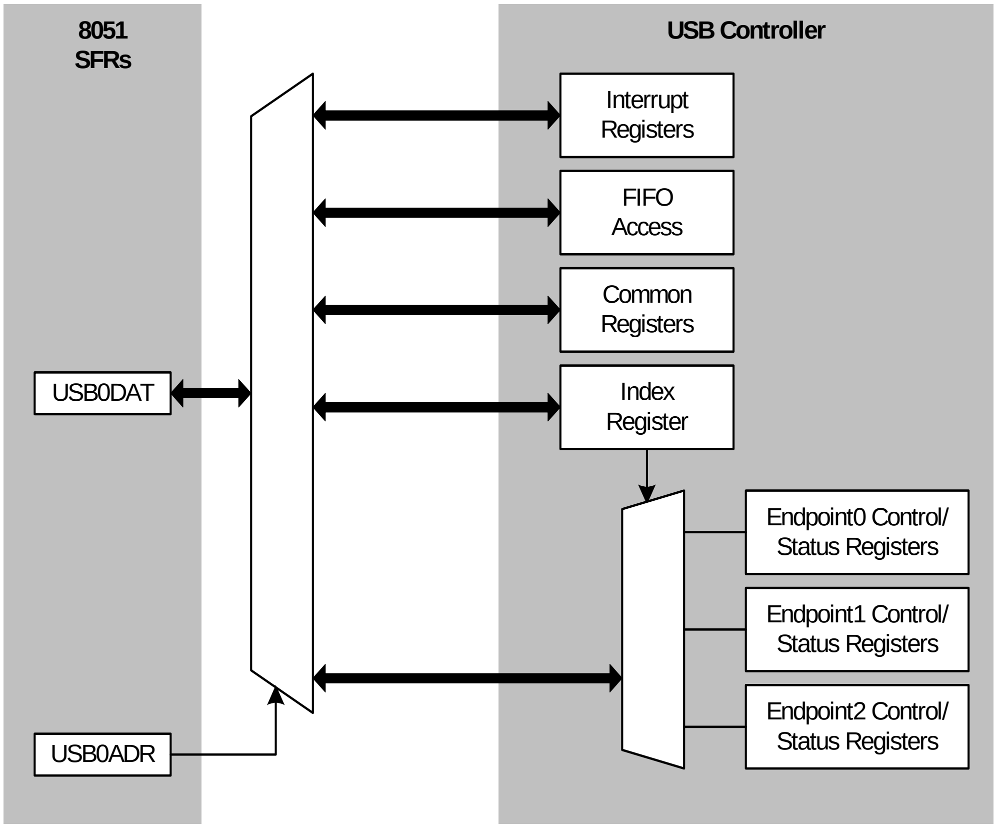

8051
SFRs
USB Controller
Interrupt Registers
FIFO Access
Common Registers
Index Register
USB0DAT
USB0ADR
Endpoint0 Control/ Status Registers
Endpoint1 Control/ Status Registers
Endpoint2 Control/ Status Registers

**Figure 18.2. USB0 Register Access Scheme**

## USB Register Definition 18.4. INDEX: USB0 Endpoint Index

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

USB Register Address = 0x0E

| Bit | Name | Function |
|---|---|---|
| 7:4 | Unused | Unused. Read = 0000b. Write = don’t care. |
| 3:0 | EPSEL[3:0] | **Endpoint Select Bits.**<br>These bits select which endpoint is targeted when indexed USB0 registers are accessed.<br>0000: Endpoint 0<br>0001: Endpoint 1<br>0010: Endpoint 2<br>0011-1111: Reserved. |

## 18.4. USB Clock Configuration

USB0 is capable of communication as a Full or Low Speed USB function. Communication speed is selected via the SPEED bit in SFR USB0XCN. When operating as a Low Speed function, the USB0 clock must be 6 MHz. When operating as a Full Speed function, the USB0 clock must be 48 MHz. Clock options are described in Section “16. Oscillators and Clock Selection” on page 86. The USB0 clock is selected via SFR CLKSEL (see SFR Definition 16.1).

Clock Recovery circuitry uses the incoming USB data stream to adjust the internal oscillator; this allows the internal oscillator to meet the requirements for USB clock tolerance. Clock Recovery should be used in the following configurations:

| Communication Speed | USB Clock |
|---|---|
| Full Speed | Internal Oscillator |
| Low Speed | Internal Oscillator / 8 |

When operating USB0 as a Low Speed function with Clock Recovery, software must write 1 to the CRLOW bit to enable Low Speed Clock Recovery. Clock Recovery is typically not necessary in Low Speed mode.

Single Step Mode can be used to help the Clock Recovery circuitry to lock when high noise levels are present on the USB network. This mode is not required (or recommended) in typical USB environments.

**Table 18.3. FIFO Configurations**

| Endpoint Number | Split Mode Enabled? | Maximum IN Packet Size (Double Buffer Disabled / Enabled) | Maximum OUT Packet Size (Double Buffer Disabled / Enabled) |
|---|---|---|---|
| 0 | N/A | 64 | |
| 1 | N | 128 / 64 | |
| | Y | 64 / 32 | 64 / 32 |
| 2 | N | 64/ 32 | |
| | Y | 32 / 16 | 32 / 16 |

### 18.5.1. FIFO Access

Each endpoint FIFO is accessed through a corresponding FIFOn register. A read of an endpoint FIFOn register unloads one byte from the FIFO; a write of an endpoint FIFOn register loads one byte into the endpoint FIFO. When an endpoint FIFO is configured for Split Mode, a read of the endpoint FIFOn register unloads one byte from the OUT endpoint FIFO; a write of the endpoint FIFOn register loads one byte into the IN endpoint FIFO.

## USB Register Definition 18.6. FIFOn: USB0 Endpoint FIFO Access

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

USB Register Address = 0x20-0x22

| Bit | Name | Function |
|---|---|---|
| 7:0 | FIFODATA[7:0] | **Endpoint FIFO Access Bits.**<br>USB Addresses 0x20-0x22 provide access to the 4 pairs of endpoint FIFOs:<br>0x20: Endpoint 0<br>0x21: Endpoint 1<br>0x22: Endpoint 2<br>Writing to the FIFO address loads data into the IN FIFO for the corresponding endpoint. Reading from the FIFO address unloads data from the OUT FIFO for the corresponding endpoint. |

## USB Register Definition 18.23. EOUTCSRH: USB0 OUT Endpoint Control High Byte

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

USB Register Address = 0x15

| Bit | Name | Function |
|---|---|---|
| 7 | DBOEN | **Double-buffer Enable.**<br>0: Double-buffering disabled for the selected OUT endpoint.<br>1: Double-buffering enabled for the selected OUT endpoint. |
| 6 | ISO | **Isochronous Transfer Enable.**<br>This bit enables/disables isochronous transfers on the current endpoint.<br>0: Endpoint configured for bulk/interrupt transfers.<br>1: Endpoint configured for isochronous transfers. |
| 5:0 | Unused | Unused. Read = 000000b. Write = don’t care. |

## USB Register Definition 18.24. EOUTCNTL: USB0 OUT Endpoint Count Low

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

USB Register Address = 0x16

| Bit | Name | Function |
|---|---|---|
| 7:0 | EOCL[7:0] | **OUT Endpoint Count Low Byte.**<br>EOCL holds the lower 8-bits of the 10-bit number of data bytes in the last received packet in the current OUT endpoint FIFO. This number is only valid while OPRDY = 1. |

**Table 19.5. SMBus Status Decoding With Hardware ACK Generation Disabled (EHACK = 0) (Continued)**

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

$$\text{Baud Rate} = \frac{\text{SYSCLK}}{(65536 - (\text{SBRLH1:SBRLL1}))} \times \frac{1}{2} \times \frac{1}{\text{Prescaler}}$$

**Equation 21.1. UART1 Baud Rate**

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

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

| | Target Baud Rate (bps) | Actual Baud Rate (bps) | Baud Rate Error | Oscillator Divide Factor | SB1PS[1:0] (Prescaler Bits) | Reload Value in SBRLH1:SBRLL1 |
|---|---|---|---|---|---|---|
| SYSCLK = 12 MHz | 230400 | 230769 | 0.16% | 52 | 11 | 0xFFE6 |
| | 115200 | 115385 | 0.16% | 104 | 11 | 0xFFCC |
| | 57600 | 57692 | 0.16% | 208 | 11 | 0xFF98 |
| | 28800 | 28846 | 0.16% | 416 | 11 | 0xFF30 |
| | 14400 | 14388 | 0.08% | 834 | 11 | 0xFE5F |
| | 9600 | 9600 | 0.0% | 1250 | 11 | 0xFD8F |
| | 2400 | 2400 | 0.0% | 5000 | 11 | 0xF63C |
| | 1200 | 1200 | 0.0% | 10000 | 11 | 0xEC78 |
| SYSCLK = 24 MHz | 230400 | 230769 | 0.16% | 104 | 11 | 0xFFCC |
| | 115200 | 115385 | 0.16% | 208 | 11 | 0xFF98 |
| | 57600 | 57692 | 0.16% | 416 | 11 | 0xFF30 |
| | 28800 | 28777 | 0.08% | 834 | 11 | 0xFE5F |
| | 14400 | 14406 | 0.04% | 1666 | 11 | 0xFCBF |
| | 9600 | 9600 | 0.0% | 2500 | 11 | 0xFB1E |
| | 2400 | 2400 | 0.0% | 10000 | 11 | 0xEC78 |
| | 1200 | 1200 | 0.0% | 20000 | 11 | 0xD8F0 |
| SYSCLK = 48 MHz | 230400 | 230769 | 0.16% | 208 | 11 | 0xFF98 |
| | 115200 | 115385 | 0.16% | 416 | 11 | 0xFF30 |
| | 57600 | 57554 | 0.08% | 834 | 11 | 0xFE5F |
| | 28800 | 28812 | 0.04% | 1666 | 11 | 0xFCBF |
| | 14400 | 14397 | 0.02% | 3334 | 11 | 0xF97D |
| | 9600 | 9600 | 0.0% | 5000 | 11 | 0xF63C |
| | 2400 | 2400 | 0.0% | 20000 | 11 | 0xD8F0 |
| | 1200 | 1200 | 0.0% | 40000 | 11 | 0xB1E0 |

## 21.2. Data Format

UART1 has a number of available options for data formatting. Data transfers begin with a start bit (logic low), followed by the data bits (sent LSB-first), a parity or extra bit (if selected), and end with one or two stop bits (logic high). The data length is variable between 5 and 8 bits. A parity bit can be appended to the data, and automatically generated and detected by hardware for even, odd, mark, or space parity. The stop

## SFR Definition 21.4. SBCON1: UART1 Baud Rate Generator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | SB1RUN | Reserved | Reserved | Reserved | Reserved | SB1PS[1:0] | |
| Type | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xAC

| Bit | Name | Function |
|---|---|---|
| 7 | Reserved | Reserved. Read = 0b. Must Write 0b. |
| 6 | SB1RUN | **Baud Rate Generator Enable.**<br>0: Baud Rate Generator is disabled. UART1 will not function.<br>1: Baud Rate Generator is enabled. |
| 5:2 | Reserved | Reserved. Read = 0000b. Must Write 0000b. |
| 1:0 | SB1PS[1:0] | **Baud Rate Prescaler Select.**<br>00: Prescaler = 12<br>01: Prescaler = 4<br>10: Prescaler = 48<br>11: Prescaler = 1 |

## SFR Definition 21.5. SBRLH1: UART1 Baud Rate Generator High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SBRLH1[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xB5

| Bit | Name | Function |
|---|---|---|
| 7:0 | SBRLH1[7:0] | **UART1 Baud Rate Reload High Bits.**<br>High Byte of reload value for UART1 Baud Rate Generator. |

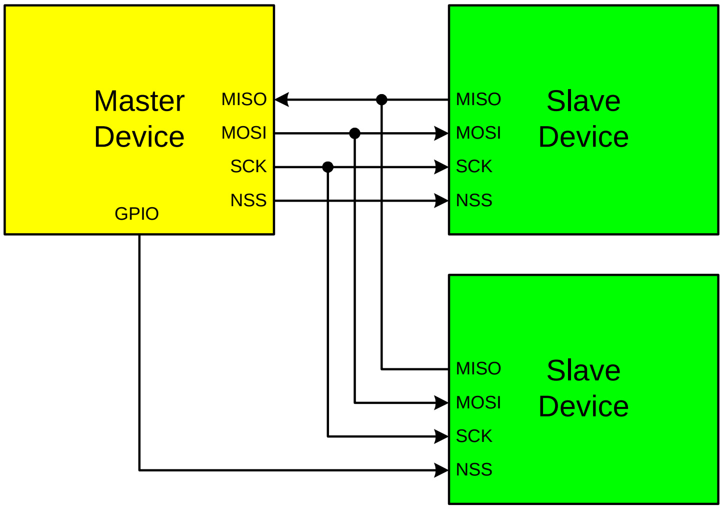


**Figure 22.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram**

## 22.3. SPI0 Slave Mode Operation

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

When configured as a slave, SPI0 can be configured for 4-wire or 3-wire operation. The default, 4-wire slave mode, is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 1. In 4-wire mode, the NSS signal is routed to a port pin and configured as a digital input. SPI0 is enabled when NSS is logic 0, and disabled when NSS is logic 1. The bit counter is reset on a falling edge of NSS. Note that the NSS signal must be driven low at least 2 system clocks before the first active edge of SCK for each byte transfer. Figure 22.4 shows a connection diagram between two slave devices in 4-wire slave mode and a master device.

3-wire slave mode is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 0. NSS is not used in this mode, and is not mapped to an external port pin through the crossbar. Since there is no way of uniquely addressing the device in 3-wire slave mode, SPI0 must be the only slave device present on the bus. It is important to note that in 3-wire slave mode there is no external means of resetting the bit counter that determines when a full byte has been received. The bit counter can only be reset by disabling and re-enabling SPI0 with the SPIEN bit. Figure 22.3 shows a connection diagram between a slave device in 3-wire slave mode and a master device.

## 22.4. SPI0 Interrupt Sources

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

All of the following bits must be cleared by software.

## SFR Definition 22.3. SPI0CKR: SPI0 Clock Rate

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

SFR Address = 0xA2

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

## SFR Definition 22.4. SPI0DAT: SPI0 Data

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

SFR Address = 0xA3

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

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

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

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

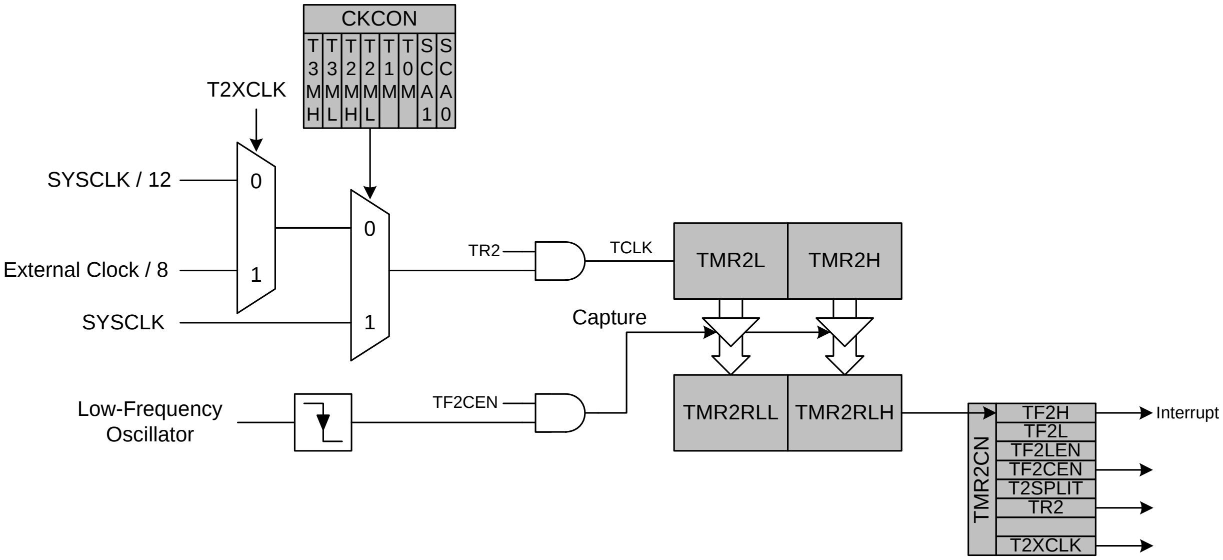


**Figure 23.6. Timer 2 Low-Frequency Oscillation Capture Mode Block Diagram**

## C2 Register Definition 25.4. DEVCTL: C2 Device Control

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

C2 Address: 0x02

| Bit | Name | Function |
|---|---|---|
| 7:0 | DEVCTL[7:0] | **Device Control Register.**<br>This register is used to halt the device for EPROM operations via the C2 interface. Refer to the EPROM chapter for more information. |

## C2 Register Definition 25.5. EPCTL: EPROM Programming Control Register

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

C2 Address: 0xDF

| Bit | Name | Function |
|---|---|---|
| 7:0 | EPCTL[7:0] | **EPROM Programming Control Register.**<br>This register is used to enable EPROM programming via the C2 interface. Refer to the EPROM chapter for more information. |