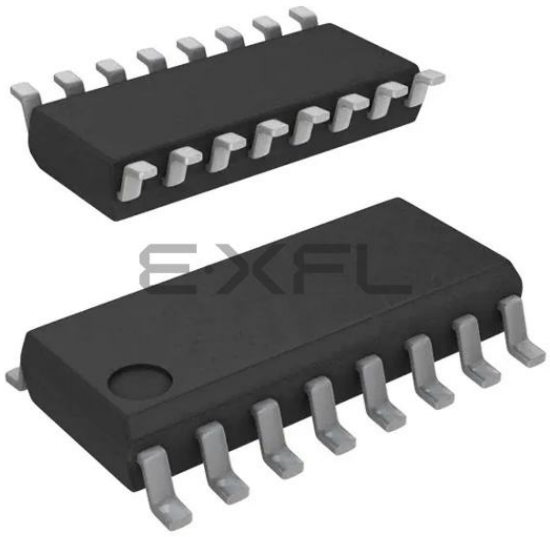


Welcome to E-XFL.COM

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	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	Cap Sense, POR, PWM, Temp Sensor, WDT
Number of I/O	13
Program Memory Size	16KB (16K 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 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	16-SOIC (0.154", 3.90mm Width)
Supplier Device Package	16-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f803-gs

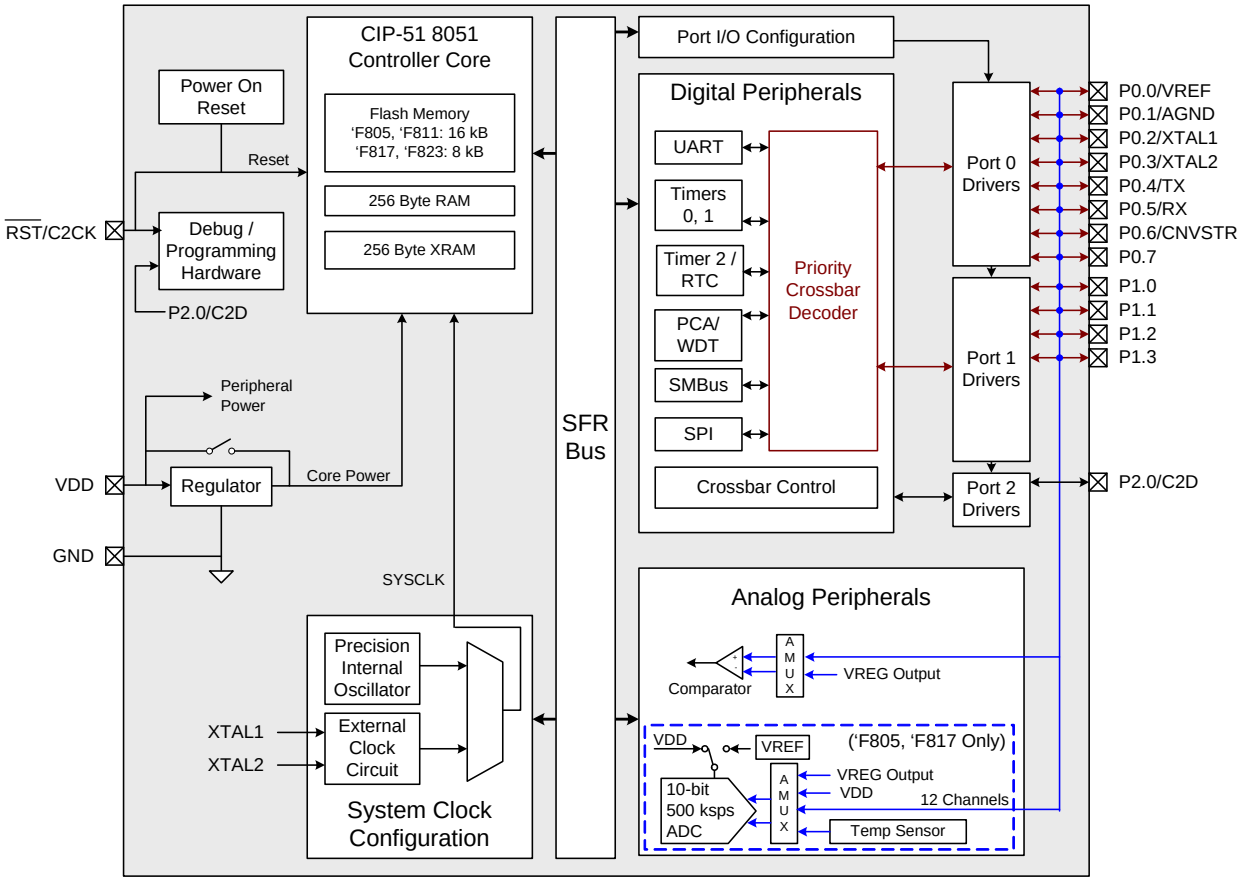


Figure 1.6. C8051F805, C8051F811, C8051F817, C8051F823 Block Diagram

8.3.3. Settling Time Requirements

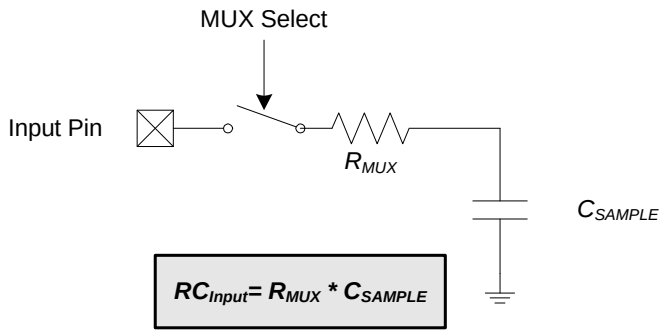
A minimum tracking time is required before each conversion to ensure that an accurate conversion is performed. This tracking time is determined by any series impedance, including the AMUX0 resistance, the the ADC0 sampling capacitance, and the accuracy required for the conversion. In delayed tracking mode, three SAR clocks are used for tracking at the start of every conversion. For many applications, these three SAR clocks will meet the minimum tracking time requirements.

Figure 8.3 shows the equivalent ADC0 input circuit. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 8.1. See Table 7.9 for ADC0 minimum settling time requirements as well as the mux impedance and sampling capacitor values.

$$t = \ln\left(\frac{2^n}{SA}\right) \times R_{TOTAL} C_{SAMPLE}$$

Equation 8.1. ADC0 Settling Time Requirements

Where:
SA is the settling accuracy, given as a fraction of an LSB (for example, 0.25 to settle within 1/4 LSB)
t is the required settling time in seconds
R_{TOTAL} is the sum of the AMUX0 resistance and any external source resistance.
n is the ADC resolution in bits (10).



Note: See electrical specification tables for R_{MUX} and C_{SAMPLE} parameters.

Figure 8.3. ADC0 Equivalent Input Circuits

SFR Definition 8.1. ADC0CF: ADC0 Configuration

Bit	7	6	5	4	3	2	1	0
Name	AD0SC[4:0]					AD0LJST	AD08BE	AMP0GN0
Type	R/W					R/W	R/W	R/W
Reset	1	1	1	1	1	0	0	1

SFR Address = 0xBC

Bit	Name	Function
7:3	AD0SC[4:0]	**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 the ADC specification table. $AD0SC = \frac{SYSCLK}{CLK_{SAR}} - 1$
2	AD0LJST	**ADC0 Left Justify Select.** 0: Data in ADC0H:ADC0L registers are right-justified. 1: Data in ADC0H:ADC0L registers are left-justified. **Note:** The AD0LJST bit is only valid for 10-bit mode (AD08BE = 0).
1	AD08BE	**8-Bit Mode Enable.** 0: ADC operates in 10-bit mode (normal). 1: ADC operates in 8-bit mode. **Note:** When AD08BE is set to 1, the AD0LJST bit is ignored.
0	AMP0GN0	**ADC Gain Control Bit.** 0: Gain = 0.5 1: Gain = 1

SFR Definition 8.9. ADC0MX: AMUX0 Channel Select

Bit	7	6	5	4	3	2	1	0
Name				AMX0P[3:0]				
Type	R	R	R	R/W				
Reset	0	0	0	1	1	1	1	1

SFR Address = 0xBB

Bit	Name	Function		
7:5	Unused	Read = 000b; Write = Don't Care.		
4:0	AMX0P[4:0]	**AMUX0 Positive Input Selection.**		
			20-Pin and 24-Pin Devices	**16-Pin Devices**
		00000:	P0.0	P0.0
		00001:	P0.1	P0.1
		00010:	P0.2	P0.2
		00011:	P0.3	P0.3
		00100:	P0.4	P0.4
		00101:	P0.5	P0.5
		00110:	P0.6	P0.6
		00111:	P0.7	P0.7
		01000	P1.0	P1.0
		01001	P1.1	P1.1
		01010	P1.2	P1.2
		01011	P1.3	P1.3
		01100	P1.4	Reserved.
		01101	P1.5	Reserved.
		01110	P1.6	Reserved.
		01111	P1.7	Reserved.
		10000:	Temp Sensor	Temp Sensor
		10001:	VREG Output	VREG Output
		10010:	VDD	VDD
		10011:	GND	GND
		10100 – 11111:	no input selected	

SFR Definition 13.5. CS0SS: Capacitive Sense Auto-Scan Start Channel

Bit	7	6	5	4	3	2	1	0
Name				CS0SS[4:0]				
Type	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xB9

Bit	Name	Description
7:5	Unused	Read = 000b; Write = Don't care
4:0	CS0SS[4:0]	**Starting Channel for Auto-Scan.** Sets the first CS0 channel to be selected by the mux for Capacitive Sense conversion when auto-scan is enabled and active. When auto-scan is enabled, a write to CS0SS will also update CS0MX.

SFR Definition 13.6. CS0SE: Capacitive Sense Auto-Scan End Channel

Bit	7	6	5	4	3	2	1	0
Name				CS0SE[4:0]				
Type	R	R	R	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xBA

Bit	Name	Description
7:5	Unused	Read = 000b; Write = Don't care
4:0	CS0SE[4:0]	**Ending Channel for Auto-Scan.** Sets the last CS0 channel to be selected by the mux for Capacitive Sense conversion when auto-scan is enabled and active.

14. CIP-51 Microcontroller

The MCU system controller core is the CIP-51 microcontroller. The CIP-51 is fully compatible with the MCS-51™ instruction set; standard 803x/805x assemblers and compilers can be used to develop software. The MCU family has a superset of all the peripherals included with a standard 8051. The CIP-51 also includes on-chip debug hardware (see description in Section 30), and interfaces directly with the analog and digital subsystems providing a complete data acquisition or control-system solution in a single integrated circuit.

The CIP-51 Microcontroller core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability (see Figure 14.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 25 MIPS Peak Throughput with 25 MHz Clock
- 0 to 25 MHz Clock Frequency
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- On-chip Debug Logic
- Program and Data Memory Security

Performance

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute, and usually have a maximum system clock of 12 MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with no instructions taking more than eight system clock cycles.

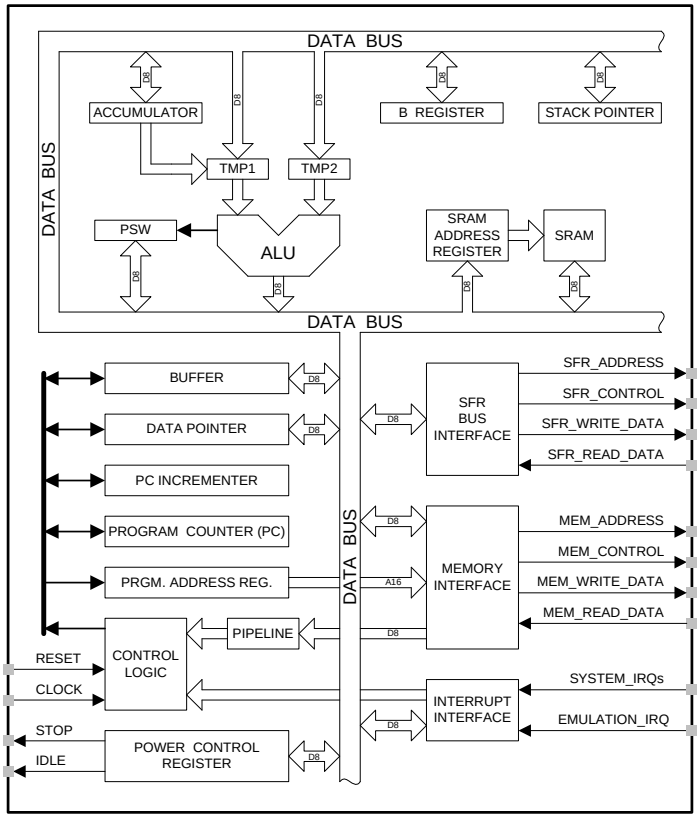


Figure 14.1. CIP-51 Block Diagram

With the CIP-51's maximum system clock at 25 MHz, it has a peak throughput of 25 MIPS. The CIP-51 has a total of 109 instructions. The table below shows the total number of instructions that require each execution time.

Clocks to Execute	1	2	2/3	3	3/4	4	4/5	5	8
Number of Instructions	26	50	5	14	6	3	2	2	1

14.1. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

14.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 14.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

Table 14.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	1
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1
Data Transfer			
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2
MOV @Ri, direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate to indirect RAM	2	2
MOV DPTR, #data16	Load DPTR with 16-bit constant	3	3
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	3
MOVC A, @A+PC	Move code byte relative PC to A	1	3
MOVX A, @Ri	Move external data (8-bit address) to A	1	3
MOVX @Ri, A	Move A to external data (8-bit address)	1	3
MOVX A, @DPTR	Move external data (16-bit address) to A	1	3
MOVX @DPTR, A	Move A to external data (16-bit address)	1	3
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A, Rn	Exchange Register with A	1	1
XCH A, direct	Exchange direct byte with A	2	2
XCH A, @Ri	Exchange indirect RAM with A	1	2
XCHD A, @Ri	Exchange low nibble of indirect RAM with A	1	2
Boolean Manipulation			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	2
SETB C	Set Carry	1	1
SETB bit	Set direct bit	2	2
CPL C	Complement Carry	1	1
CPL bit	Complement direct bit	2	2

SFR Definition 18.1. IE: Interrupt Enable

Bit	7	6	5	4	3	2	1	0
Name	EA	ESPI0	ET2	ES0	ET1	EX1	ET0	EX0
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xA8; Bit-Addressable

Bit	Name	Function
7	EA	**Enable All Interrupts.** Globally enables/disables all interrupts. It overrides individual interrupt mask settings. 0: Disable all interrupt sources. 1: Enable each interrupt according to its individual mask setting.
6	ESPI0	**Enable Serial Peripheral Interface (SPI0) Interrupt.** This bit sets the masking of the SPI0 interrupts. 0: Disable all SPI0 interrupts. 1: Enable interrupt requests generated by SPI0.
5	ET2	**Enable Timer 2 Interrupt.** This bit sets the masking of the Timer 2 interrupt. 0: Disable Timer 2 interrupt. 1: Enable interrupt requests generated by the TF2L or TF2H flags.
4	ES0	**Enable UART0 Interrupt.** This bit sets the masking of the UART0 interrupt. 0: Disable UART0 interrupt. 1: Enable UART0 interrupt.
3	ET1	**Enable Timer 1 Interrupt.** This bit sets the masking of the Timer 1 interrupt. 0: Disable all Timer 1 interrupt. 1: Enable interrupt requests generated by the TF1 flag.
2	EX1	**Enable External Interrupt 1.** This bit sets the masking of External Interrupt 1. 0: Disable external interrupt 1. 1: Enable interrupt requests generated by the $\overline{\text{INT1}}$ input.
1	ET0	**Enable Timer 0 Interrupt.** This bit sets the masking of the Timer 0 interrupt. 0: Disable all Timer 0 interrupt. 1: Enable interrupt requests generated by the TF0 flag.
0	EX0	**Enable External Interrupt 0.** This bit sets the masking of External Interrupt 0. 0: Disable external interrupt 0. 1: Enable interrupt requests generated by the $\overline{\text{INT0}}$ input.

19. Flash Memory

On-chip, re-programmable Flash memory is included for program code and non-volatile data storage. The Flash memory can be programmed in-system through the C2 interface or by software using the MOVX write instruction. Once cleared to logic 0, a Flash bit must be erased to set it back to logic 1. Flash bytes would typically be erased (set to 0xFF) before being reprogrammed. The write and erase operations are automatically timed by hardware for proper execution; data polling to determine the end of the write/erase operations is not required. Code execution is stalled during Flash write/erase operations. Refer to Table 7.6 for complete Flash memory electrical characteristics.

19.1. Programming The Flash Memory

The simplest means of programming the Flash memory is through the C2 interface using programming tools provided by Silicon Laboratories or a third party vendor. This is the only means for programming a non-initialized device. For details on the C2 commands to program Flash memory, see Section “30. C2 Interface” on page 244.

The Flash memory can be programmed by software using the MOVX write instruction with the address and data byte to be programmed provided as normal operands. Before programming Flash memory using MOVX, Flash programming operations must be enabled by: (1) setting the PSWE Program Store Write Enable bit (PSCTL.0) to logic 1 (this directs the MOVX writes to target Flash memory); and (2) Writing the Flash key codes in sequence to the Flash Lock register (FLKEY). The PSWE bit remains set until cleared by software. For detailed guidelines on programming Flash from firmware, please see Section “19.4. Flash Write and Erase Guidelines” on page 115.

Note: A minimum SYSCLK frequency is required for writing or erasing Flash memory, as detailed in “7. Electrical Characteristics” on page 39.

To ensure the integrity of the Flash contents, the on-chip VDD Monitor must be enabled and enabled as a reset source in any system that includes code that writes and/or erases Flash memory from software. Furthermore, there should be no delay between enabling the V_{DD} Monitor and enabling the V_{DD} Monitor as a reset source. Any attempt to write or erase Flash memory while the V_{DD} Monitor is disabled, or not enabled as a reset source, will cause a Flash Error device reset.

19.1.1. Flash Lock and Key Functions

Flash writes and erases by user software are protected with a lock and key function. The Flash Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before Flash operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, Flash writes and erases will be disabled until the next system reset. Flash writes and erases will also be disabled if a Flash write or erase is attempted before the key codes have been written properly. The Flash lock resets after each write or erase; the key codes must be written again before a following Flash operation can be performed. The FLKEY register is detailed in SFR Definition 19.2.

19.1.2. Flash Erase Procedure

The Flash memory is organized in 512-byte pages. The erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire 512-byte page, perform the following steps:

1. Save current interrupt state and disable interrupts.
2. Set the PSEE bit (register PSCTL).
3. Set the PSWE bit (register PSCTL).
4. Write the first key code to FLKEY: 0xA5.
5. Write the second key code to FLKEY: 0xF1.
6. Using the MOVX instruction, write a data byte to any location within the 512-byte page to be erased.
7. Clear the PSWE and PSEE bits.

22.3. External Oscillator Drive Circuit

The external oscillator circuit may drive an external crystal, ceramic resonator, capacitor, or RC network. A CMOS clock may also provide a clock input. For a crystal or ceramic resonator configuration, the crystal/resonator must be wired across the XTAL1 and XTAL2 pins as shown in Option 1 of Figure 22.1. A 10 MΩ resistor also must be wired across the XTAL2 and XTAL1 pins for the crystal/resonator configuration. In RC, capacitor, or CMOS clock configuration, the clock source should be wired to the XTAL2 pin as shown in Option 2, 3, or 4 of Figure 22.1. The type of external oscillator must be selected in the OSCXCN register, and the frequency control bits (XFCN) must be selected appropriately (see SFR Definition 22.4).

Important Note on External Oscillator Usage: Port pins must be configured when using the external oscillator circuit. When the external oscillator drive circuit is enabled in crystal/resonator mode, Port pins P0.2 and P0.3 are used as XTAL1 and XTAL2 respectively. When the external oscillator drive circuit is enabled in capacitor, RC, or CMOS clock mode, Port pin P0.3 is used as XTAL2. The Port I/O Crossbar should be configured to skip the Port pins used by the oscillator circuit; see Section “23.3. Priority Crossbar Decoder” on page 143 for Crossbar configuration. Additionally, when using the external oscillator circuit in crystal/resonator, capacitor, or RC mode, the associated Port pins should be configured as **analog inputs**. In CMOS clock mode, the associated pin should be configured as a **digital input**. See Section “23.4. Port I/O Initialization” on page 147 for details on Port input mode selection.

24.6. CRC0 Bit Reverse Feature

CRC0 includes hardware to reverse the bit order of each bit in a byte as shown in Figure 24.1. Each byte of data written to CRC0FLIP is read back bit reversed. For example, if 0xC0 is written to CRC0FLIP, the data read back is 0x03. Bit reversal is a useful mathematical function used in algorithms such as the FFT.

SFR Definition 24.6. CRC0FLIP: CRC Bit Flip

Bit	7	6	5	4	3	2	1	0
Name	CRC0FLIP[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCF

Bit	Name	Function
7:0	CRC0FLIP[7:0]	**CRC0 Bit Flip.** Any byte written to CRC0FLIP is read back in a bit-reversed order, i.e. the written LSB becomes the MSB. For example: If 0xC0 is written to CRC0FLIP, the data read back will be 0x03. If 0x05 is written to CRC0FLIP, the data read back will be 0xA0.

26. SMBus

The SMBus I/O interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the I^2C serial bus. Reads and writes to the interface by the system controller are byte oriented with the SMBus interface autonomously controlling the serial transfer of the data. Data can be transferred at up to 1/20th of the system clock as a master or slave (this can be faster than allowed by the SMBus specification, depending on the system clock used). A method of extending the clock-low duration is available to accommodate devices with different speed capabilities on the same bus.

The SMBus interface may operate as a master and/or slave, and may function on a bus with multiple masters. The SMBus provides control of SDA (serial data), SCL (serial clock) generation and synchronization, arbitration logic, and START/STOP control and generation. The SMBus peripheral can be fully driven by software (i.e., software accepts/rejects slave addresses, and generates ACKs), or hardware slave address recognition and automatic ACK generation can be enabled to minimize software overhead. A block diagram of the SMBus peripheral and the associated SFRs is shown in Figure 26.1.

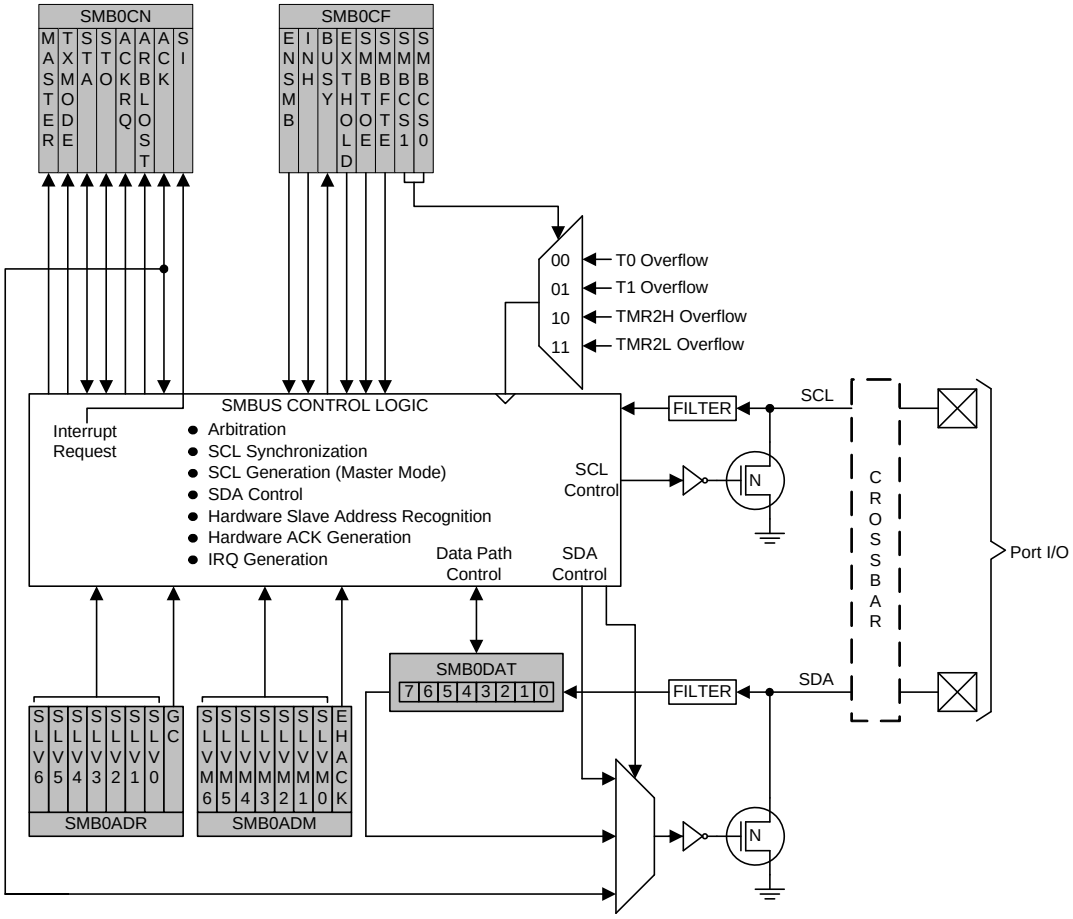


Figure 26.1. SMBus Block Diagram

SFR Definition 26.3. SMB0ADR: SMBus Slave Address

Bit	7	6	5	4	3	2	1	0
Name	SLV[6:0]							GC
Type	R/W							R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xD7

Bit	Name	Function
7:1	SLV[6:0]	**SMBus Hardware Slave Address.** Defines the SMBus Slave Address(es) for automatic hardware acknowledgement. Only address bits which have a 1 in the corresponding bit position in SLVM[6:0] are checked against the incoming address. This allows multiple addresses to be recognized.
0	GC	**General Call Address Enable.** When hardware address recognition is enabled (EHACK = 1), this bit will determine whether the General Call Address (0x00) is also recognized by hardware. 0: General Call Address is ignored. 1: General Call Address is recognized.

SFR Definition 26.4. SMB0ADM: SMBus Slave Address Mask

Bit	7	6	5	4	3	2	1	0
Name	SLVM[6:0]							EHACK
Type	R/W							R/W
Reset	1	1	1	1	1	1	1	0

SFR Address = 0xD6

Bit	Name	Function
7:1	SLVM[6:0]	**SMBus Slave Address Mask.** Defines which bits of register SMB0ADR are compared with an incoming address byte, and which bits are ignored. Any bit set to 1 in SLVM[6:0] enables comparisons with the corresponding bit in SLV[6:0]. Bits set to 0 are ignored (can be either 0 or 1 in the incoming address).
0	EHACK	**Hardware Acknowledge Enable.** Enables hardware acknowledgement of slave address and received data bytes. 0: Firmware must manually acknowledge all incoming address and data bytes. 1: Automatic Slave Address Recognition and Hardware Acknowledge is Enabled.

Table 26.6. SMBus Status Decoding With Hardware ACK Generation Enabled (EHACK = 1) (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	0	0	X	A slave address + R/W was received; ACK sent.	If Write, Set ACK for first data byte.	0	0	1	0000
						If Read, Load SMB0DAT with data byte	0	0	X	0100
		0	1	X	Lost arbitration as master; slave address + R/W received; ACK sent.	If Write, Set ACK for first data byte.	0	0	1	0000
						If Read, Load SMB0DAT with data byte	0	0	X	0100
						Reschedule failed transfer	1	0	X	1110
	0001	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 while attempting a STOP.	No action required (transfer complete/aborted).	0	0	0	—
	0000	0	0	X	A slave byte was received.	Set ACK for next data byte; Read SMB0DAT.	0	0	1	0000
						Set NACK for next data byte; Read SMB0DAT.	0	0	0	0000
Bus Error Condition	0010	0	1	X	Lost arbitration while attempting a repeated START.	Abort failed transfer.	0	0	X	—
						Reschedule failed transfer.	1	0	X	1110
	0001	0	1	X	Lost arbitration due to a detected STOP.	Abort failed transfer.	0	0	X	—
						Reschedule failed transfer.	1	0	X	1110
	0000	0	1	X	Lost arbitration while transmitting a data byte as master.	Abort failed transfer.	0	0	X	—
						Reschedule failed transfer.	1	0	X	1110

27.2. Operational Modes

UART0 provides standard asynchronous, full duplex communication. The UART mode (8-bit or 9-bit) is selected by the S0MODE bit (SCON0.7). Typical UART connection options are shown in Figure 27.3.

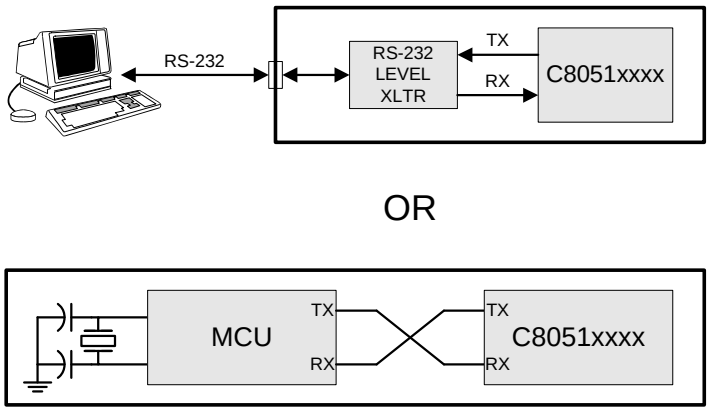


Figure 27.3. UART Interconnect Diagram

27.2.1. 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.

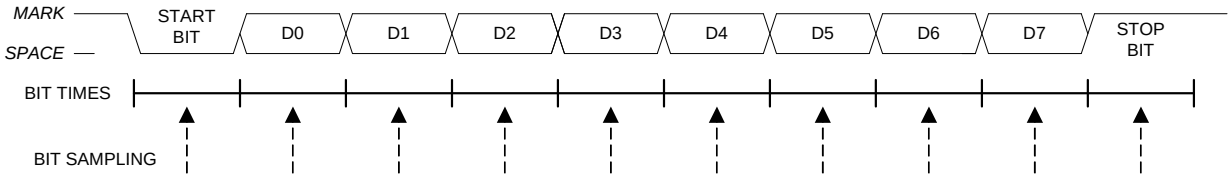


Figure 27.4. 8-Bit UART Timing Diagram

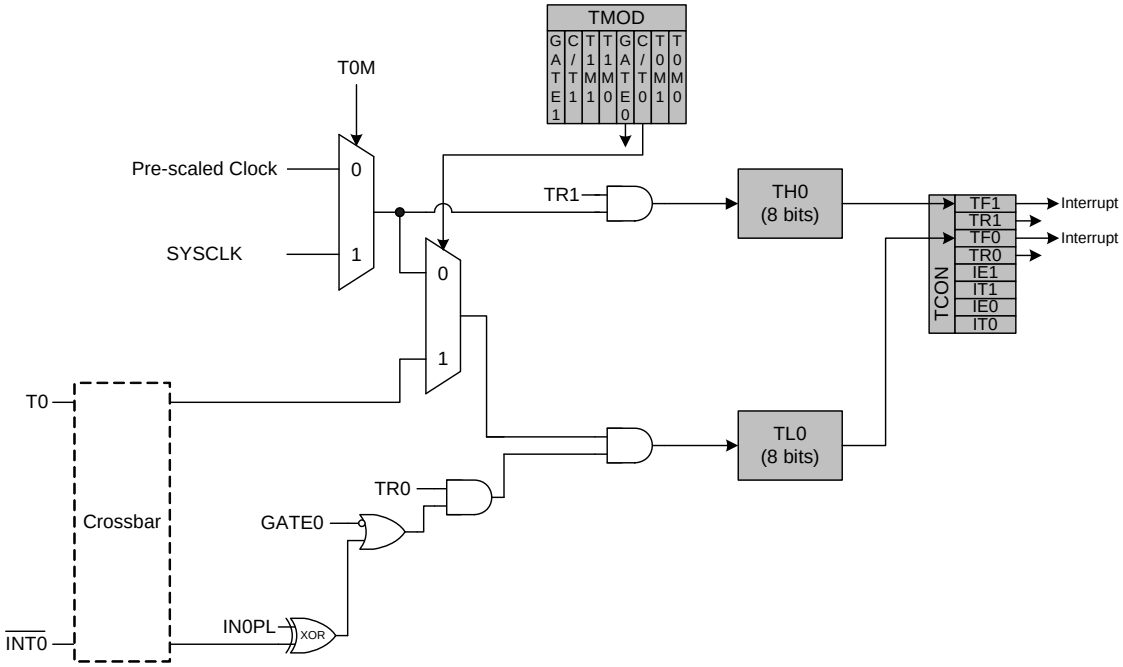


Figure 28.3. T0 Mode 3 Block Diagram

C2 Register Definition 30.2. DEVICEID: C2 Device ID

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

C2 Address: 0x00

Bit	Name	Function
7:0	DEVICEID[7:0]	**Device ID.** This read-only register returns the 8-bit device ID: 0x23 (C8051F80x-83x).

C2 Register Definition 30.3. REVID: C2 Revision ID

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

C2 Address: 0x01

Bit	Name	Function
7:0	REVID[7:0]	**Revision ID.** This read-only register returns the 8-bit revision ID. For example: 0x00 = Revision A.

C2 Register Definition 30.4. FPCTL: C2 Flash Programming Control

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

C2 Address: 0x02

Bit	Name	Function
7:0	FPCTL[7:0]	**C2 Flash Programming Control Register.** This register is used to enable Flash programming via the C2 interface. To enable C2 Flash programming, the following codes must be written in order: 0x02, 0x01. Once C2 Flash programming is enabled, a system reset must be issued to resume normal operation.

C2 Register Definition 30.5. FPDAT: C2 Flash Programming Data

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

C2 Address: 0xBF

Bit	Name	Function	
7:0	FPDAT[7:0]	**C2 Flash Programming Data Register.** This register is used to pass Flash commands, addresses, and data during C2 Flash accesses. Valid commands are listed below.	
		Code	**Command**
		0x06	Flash Block Read
		0x07	Flash Block Write
		0x08	Flash Page Erase
		0x03	Device Erase