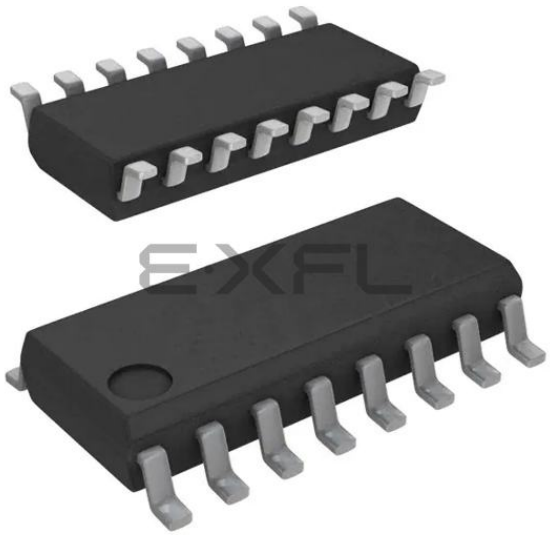

Welcome to E-XFL.COM

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, WDT
Number of I/O	13
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	-
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/c8051f821-gsr

1. System Overview

C8051F80x-83x devices are fully integrated, mixed-signal, system-on-a-chip capacitive sensing MCUs. Highlighted features are listed below. Refer to Table 2.1 for specific product feature selection and part ordering numbers.

- High-speed pipelined 8051-compatible microcontroller core (up to 25 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- Capacitive sense interface with 16 input channels
- 10-bit 500 ksps single-ended ADC with 16-channel analog multiplexer and integrated temperature sensor
- Precision calibrated 24.5 MHz internal oscillator
- 16 kb of on-chip Flash memory
- 512 bytes of on-chip RAM
- SMBus/I^2C, Enhanced UART, and Enhanced SPI serial interfaces implemented in hardware
- Three general-purpose 16-bit timers
- Programmable counter/timer array (PCA) with three capture/compare modules
- On-chip internal voltage reference
- On-chip Watchdog timer
- On-chip power-on reset and supply monitor
- On-chip voltage comparator
- 17 general purpose I/O

With on-chip power-on reset, V_{DD} monitor, watchdog timer, and clock oscillator, the C8051F80x-83x devices are truly stand-alone, system-on-a-chip solutions. The Flash memory can be reprogrammed even in-circuit, providing non-volatile data storage, and also allowing field upgrades of the 8051 firmware. User software has complete control of all peripherals, and may individually shut down any or all peripherals for power savings.

The C8051F80x-83x processors include Silicon Laboratories' 2-Wire C2 Debug and Programming interface, which allows non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production MCU installed in the final application. This debug logic supports inspection of memory, viewing and modification of special function registers, setting breakpoints, single stepping, and run and halt commands. All analog and digital peripherals are fully functional while debugging using C2. The two C2 interface pins can be shared with user functions, allowing in-system debugging without occupying package pins.

Each device is specified for 1.8–3.6 V operation over the industrial temperature range (–45 to +85 °C). An internal LDO regulator is used to supply the processor core voltage at 1.8 V. The Port I/O and $\overline{\text{RST}}$ pins are tolerant of input signals up to 5 V. See Table 2.1 for ordering information. Block diagrams of the devices in the C8051F80x-83x family are shown in Figure 1.1 through Figure 1.9.

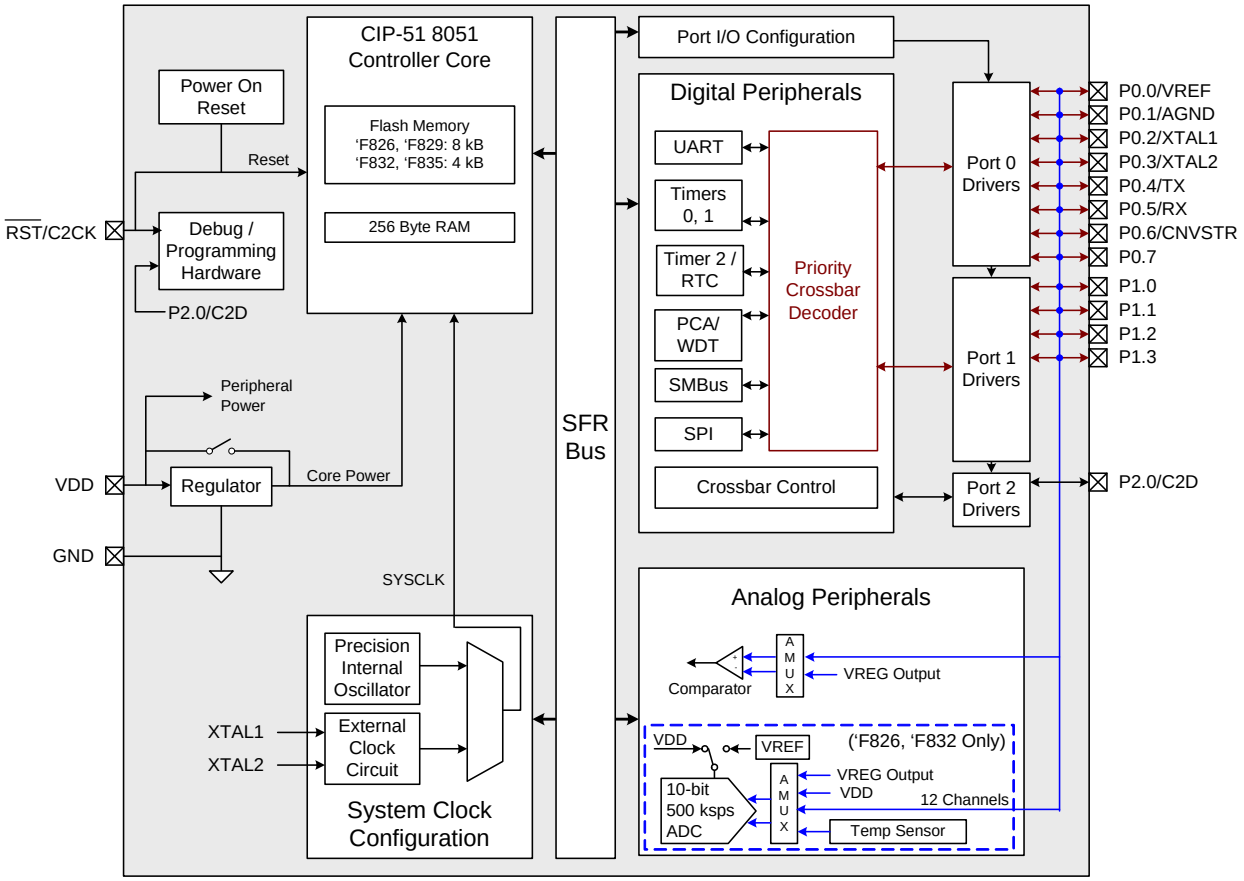

Figure 1.9. C8051F826, C8051F829, C8051F832, C8051F835 Block Diagram

4. QFN-20 Package Specifications

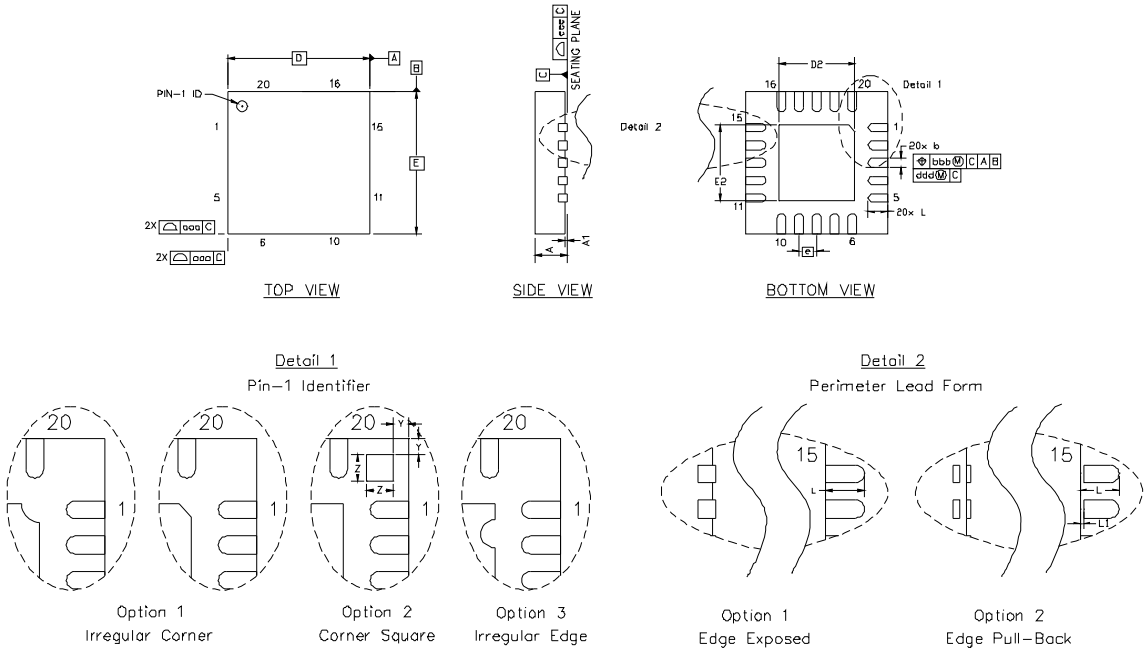

Figure 4.1. QFN-20 Package Drawing

Table 4.1. QFN-20 Package Dimensions

Dimension	Min	Typ	Max
A	0.80	0.90	1.00
A1	0.00	0.02	0.05
b	0.18	0.25	0.30
D		4.00 BSC.	
D2	2.00	2.15	2.25
e		0.50 BSC.	
E		4.00 BSC.	
E2	2.00	2.15	2.25

Dimension	Min	Typ	Max
L	0.45	0.55	0.65
L1	0.00	—	0.15
aaa	—	—	0.15
bbb	—	—	0.10
ddd	—	—	0.05
eee	—	—	0.08
Z	—	0.43	—
Y	—	0.18	—

Notes:

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing per ANSI Y14.5M-1994.
3. This drawing conforms to the JEDEC Solid State Outline MO-220, variation VGGD except for custom features D2, E2, Z, Y, and L which are toleranced per supplier designation.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

SFR Definition 8.2. ADC0H: ADC0 Data Word MSB

Bit	7	6	5	4	3	2	1	0
Name	ADC0H[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xBE

Bit	Name	Function
7:0	ADC0H[7:0]	**ADC0 Data Word High-Order Bits.** For AD0LJST = 0: Bits 7–2 will read 000000b. Bits 1–0 are the upper 2 bits of the 10-bit ADC0 Data Word. For AD0LJST = 1: Bits 7–0 are the most-significant bits of the 10-bit ADC0 Data Word. **Note:** In 8-bit mode AD0LJST is ignored, and ADC0H holds the 8-bit data word.

SFR Definition 8.3. ADC0L: ADC0 Data Word LSB

Bit	7	6	5	4	3	2	1	0
Name	ADC0L[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xBD

Bit	Name	Function
7:0	ADC0L[7:0]	**ADC0 Data Word Low-Order Bits.** For AD0LJST = 0: Bits 7–0 are the lower 8 bits of the 10-bit Data Word. For AD0LJST = 1: Bits 7–6 are the lower 2 bits of the 10-bit Data Word. Bits 5–0 will always read 0. **Note:** In 8-bit mode AD0LJST is ignored, and ADC0L will read back 00000000b.

8.5. ADC0 Analog Multiplexer

ADC0 on the C8051F800/1/2/3/4/5, C8051F812/3/4/5/6/7, C8051F824/5/6, and C8051F830/1/2 uses an analog input multiplexer to select the positive input to the ADC. Any of the following may be selected as the positive input: Port 0 or Port 1 I/O pins, the on-chip temperature sensor, or the positive power supply (V_{DD}). The ADC0 input channel is selected in the ADC0MX register described in SFR Definition 8.9.

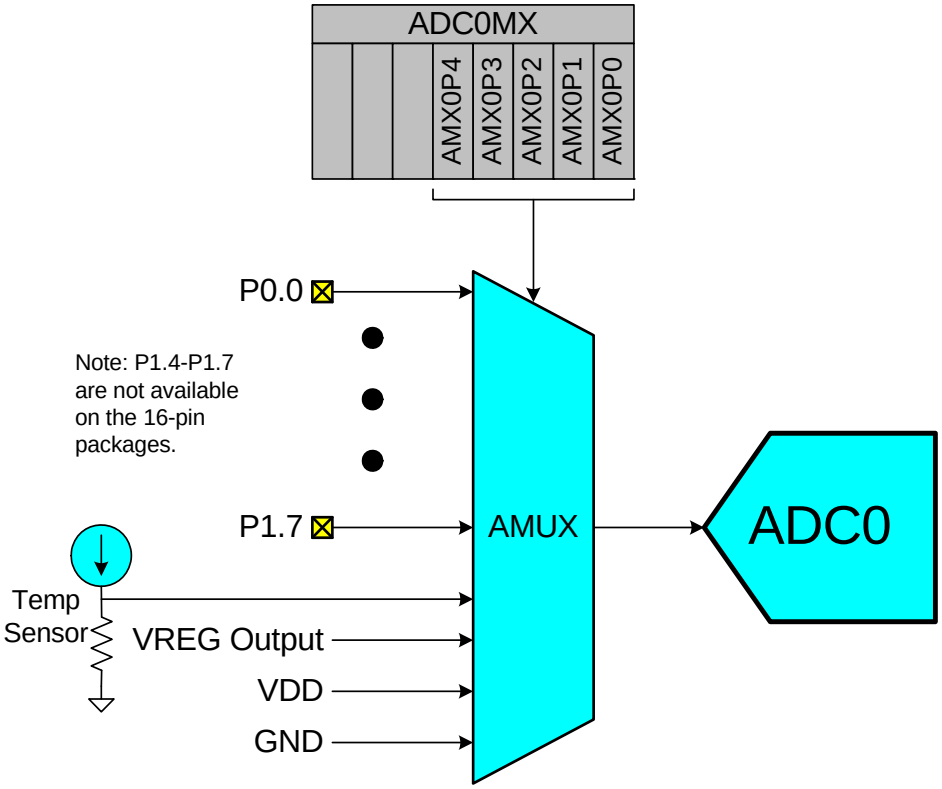

Figure 8.6. ADC0 Multiplexer Block Diagram

Important Note About ADC0 Input Configuration: Port pins selected as ADC0 inputs should be configured as analog inputs, and should be skipped by the Digital Crossbar. To configure a Port pin for analog input, set the corresponding bit in register PnMDIN to 0. To force the Crossbar to skip a Port pin, set the corresponding bit in register PnSKIP to 1. See Section “23. Port Input/Output” on page 138 for more Port I/O configuration details.

18.1. MCU Interrupt Sources and Vectors

The C8051F80x-83x MCUs support 15 interrupt sources. Software can simulate an interrupt by setting an interrupt-pending flag to logic 1. If interrupts are enabled for the flag, an interrupt request will be generated and the CPU will vector to the ISR address associated with the interrupt-pending flag. MCU interrupt sources, associated vector addresses, priority order and control bits are summarized in Table 18.1. Refer to the datasheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

18.1.1. Interrupt Priorities

Each interrupt source can be individually programmed to one of two priority levels: low or high. A low priority interrupt service routine can be preempted by a high priority interrupt. A high priority interrupt cannot be preempted. Each interrupt has an associated interrupt priority bit in an SFR (IP or EIP1) used to configure its priority level. Low priority is the default. If two interrupts are recognized simultaneously, the interrupt with the higher priority is serviced first. If both interrupts have the same priority level, a fixed priority order is used to arbitrate, given in Table 18.1.

18.1.2. Interrupt Latency

Interrupt response time depends on the state of the CPU when the interrupt occurs. Pending interrupts are sampled and priority decoded each system clock cycle. Therefore, the fastest possible response time is 5 system clock cycles: 1 clock cycle to detect the interrupt and 4 clock cycles to complete the LCALL to the ISR. If an interrupt is pending when a RETI is executed, a single instruction is executed before an LCALL is made to service the pending interrupt. Therefore, the maximum response time for an interrupt (when no other interrupt is currently being serviced or the new interrupt is of greater priority) occurs when the CPU is performing an RETI instruction followed by a DIV as the next instruction. In this case, the response time is 18 system clock cycles: 1 clock cycle to detect the interrupt, 5 clock cycles to execute the RETI, 8 clock cycles to complete the DIV instruction and 4 clock cycles to execute the LCALL to the ISR. If the CPU is executing an ISR for an interrupt with equal or higher priority, the new interrupt will not be serviced until the current ISR completes, including the RETI and following instruction.

SFR Definition 19.1. PSCTL: Program Store R/W Control

Bit	7	6	5	4	3	2	1	0
Name							PSEE	PSWE
Type	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address =0x8F

Bit	Name	Function
7:2	Unused	Read = 000000b, Write = don't care.
1	PSEE	**Program Store Erase Enable.** Setting this bit (in combination with PSWE) allows an entire page of Flash program memory to be erased. If this bit is logic 1 and Flash writes are enabled (PSWE is logic 1), a write to Flash memory using the MOVX instruction will erase the entire page that contains the location addressed by the MOVX instruction. The value of the data byte written does not matter. 0: Flash program memory erasure disabled. 1: Flash program memory erasure enabled.
0	PSWE	**Program Store Write Enable.** Setting this bit allows writing a byte of data to the Flash program memory using the MOVX write instruction. The Flash location should be erased before writing data. 0: Writes to Flash program memory disabled. 1: Writes to Flash program memory enabled; the MOVX write instruction targets Flash memory.

20.2. Stop Mode

Setting the Stop Mode Select bit (PCON.1) causes the controller core to enter Stop mode as soon as the instruction that sets the bit completes execution. In Stop mode the internal oscillator, CPU, and all digital peripherals are stopped; the state of the external oscillator circuit is not affected. Each analog peripheral (including the external oscillator circuit) may be shut down individually prior to entering Stop Mode. Stop mode can only be terminated by an internal or external reset. On reset, the device performs the normal reset sequence and begins program execution at address 0x0000.

If enabled, the Missing Clock Detector will cause an internal reset and thereby terminate the Stop mode. The Missing Clock Detector should be disabled if the CPU is to be put to in STOP mode for longer than the MCD timeout of 100 µs.

20.3. Suspend Mode

Suspend mode allows a system running from the internal oscillator to go to a very low power state similar to Stop mode, but the processor can be awakened by certain events without requiring a reset of the device. Setting the SUSPEND bit (OSCICN.5) causes the hardware to halt the CPU and the high-frequency internal oscillator, and go into Suspend mode as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data. Most digital peripherals are not active in Suspend mode. The exception to this is the Port Match feature and Timer 3, when it is run from an external oscillator source.

The clock divider bits CLKDIV[2:0] in register CLKSEL must be set to "divide by 1" when entering suspend mode.

Suspend mode can be terminated by five types of events, a port match (described in Section “23.5. Port Match” on page 150), a Timer 2 overflow (described in Section “28.2. Timer 2” on page 219), a comparator low output (if enabled), a capacitive sense greater-than comparator event, or a device reset event. In order to run Timer 3 in suspend mode, the timer must be configured to clock from the external clock source. When suspend mode is terminated, the device will continue execution on the instruction following the one that set the SUSPEND bit. If the wake event (port match or Timer 2 overflow) was configured to generate an interrupt, the interrupt will be serviced upon waking the device. If suspend mode is terminated by an internal or external reset, the CIP-51 performs a normal reset sequence and begins program execution at address 0x0000.

Note: The device will still enter suspend mode if a wake source is "pending", and the device will not wake on such pending sources. It is important to ensure that the intended wake source will trigger after the device enters suspend mode. For example, if a CS0 conversion completes and the interrupt fires before the device is in suspend mode, that interrupt cannot trigger the wake event. Because port match events are level-sensitive, pre-existing port match events will trigger a wake, as long as the match condition is still present when the device enters suspend.

SFR Definition 21.1. VDM0CN: V_{DD} Monitor Control

Bit	7	6	5	4	3	2	1	0
Name	VDMEN	VDDSTAT						
Type	R/W	R	R	R	R	R	R	R
Reset	Varies	Varies	Varies	Varies	Varies	Varies	Varies	Varies

SFR Address = 0xFF

Bit	Name	Function
7	VDMEN	**V_{DD} Monitor Enable.** This bit turns the V_{DD} monitor circuit on/off. The V_{DD} Monitor cannot generate system resets until it is also selected as a reset source in register RSTSRC (SFR Definition 21.2). Selecting the V_{DD} monitor as a reset source before it has stabilized may generate a system reset. In systems where this reset would be undesirable, a delay should be introduced between enabling the V_{DD} Monitor and selecting it as a reset source. After a power-on reset, the VDD monitor is enabled, and this bit will read 1. The state of this bit is sticky through any other reset source. 0: V_{DD} Monitor Disabled. 1: V_{DD} Monitor Enabled.
6	VDDSTAT	**V_{DD} Status.** This bit indicates the current power supply status (V_{DD} Monitor output). 0: V_{DD} is at or below the V_{DD} monitor threshold. 1: V_{DD} is above the V_{DD} monitor threshold.
5:0	Unused	Read = Varies; Write = Don't care.

21.3. External Reset

The external $\overline{\text{RST}}$ pin provides a means for external circuitry to force the device into a reset state. Asserting an active-low signal on the $\overline{\text{RST}}$ pin generates a reset; an external pullup and/or decoupling of the $\overline{\text{RST}}$ pin may be necessary to avoid erroneous noise-induced resets. See Section "7. Electrical Characteristics" on page 39 for complete $\overline{\text{RST}}$ pin specifications. The PINRSF flag (RSTSRC.0) is set on exit from an external reset.

21.4. Missing Clock Detector Reset

The Missing Clock Detector (MCD) is a one-shot circuit that is triggered by the system clock. If the system clock remains high or low for more than the MCD timeout, the one-shot will time out and generate a reset. After a MCD reset, the MCDRSF flag (RSTSRC.2) will read 1, signifying the MCD as the reset source; otherwise, this bit reads 0. Writing a 1 to the MCDRSF bit enables the Missing Clock Detector; writing a 0 disables it. The state of the $\overline{\text{RST}}$ pin is unaffected by this reset.

23.5. Port Match

Port match functionality allows system events to be triggered by a logic value change on P0 or P1. A software controlled value stored in the PnMATCH registers specifies the expected or normal logic values of P0 and P1. A Port mismatch event occurs if the logic levels of the Port's input pins no longer match the software controlled value. This allows Software to be notified if a certain change or pattern occurs on P0 or P1 input pins regardless of the XBRn settings.

The PnMASK registers can be used to individually select which P0 and P1 pins should be compared against the PnMATCH registers. A Port mismatch event is generated if (P0 & P0MASK) does not equal (P0MATCH & P0MASK) or if (P1 & P1MASK) does not equal (P1MATCH & P1MASK).

A Port mismatch event may be used to generate an interrupt or wake the device from a low power mode, such as IDLE or SUSPEND. See the Interrupts and Power Options chapters for more details on interrupt and wake-up sources.

24.1. 16-bit CRC Algorithm

The C8051F80x-83x CRC unit calculates the 16-bit CRC MSB-first, using a poly of 0x1021. The following describes the 16-bit CRC algorithm performed by the hardware:

1. XOR the most-significant byte of the current CRC result with the input byte. If this is the first iteration of the CRC unit, the current CRC result will be the set initial value (0x0000 or 0xFFFF).
2. If the MSB of the CRC result is set, left-shift the CRC result, and then XOR the CRC result with the polynomial (0x1021).
3. If the MSB of the CRC result is not set, left-shift the CRC result.
4. Repeat at Step 2 for the number of input bits (8).

For example, the 16-bit C8051F80x-83x CRC algorithm can be described by the following code:

```
unsigned short UpdateCRC (unsigned short CRC_acc, unsigned char CRC_input){
   unsigned char i;                    // loop counter
   #define POLY 0x1021
   // Create the CRC "dividend" for polynomial arithmetic (binary arithmetic
   // with no carries)
   CRC_acc = CRC_acc ^ (CRC_input << 8);
   // "Divide" the poly into the dividend using CRC XOR subtraction
   // CRC_acc holds the "remainder" of each divide
   // Only complete this division for 8 bits since input is 1 byte
   for (i = 0; i < 8; i++)
   {
      // Check if the MSB is set (if MSB is 1, then the POLY can "divide"
      // into the "dividend")
      if ((CRC_acc & 0x8000) == 0x8000)
      {
         // if so, shift the CRC value, and XOR "subtract" the poly
         CRC_acc = CRC_acc << 1;
         CRC_acc ^= POLY;
      }
      else
      {
         // if not, just shift the CRC value
         CRC_acc = CRC_acc << 1;
      }
   }
   return CRC_acc; // Return the final remainder (CRC value)
}
```

Table 24.1 lists example input values and the associated outputs using the 16-bit C8051F80x-83x CRC algorithm (an initial value of 0xFFFF is used):

Table 24.1. Example 16-bit CRC Outputs

Input	Output
0x63	0xBD35
0xAA, 0xBB, 0xCC	0x6CF6
0x00, 0x00, 0xAA, 0xBB, 0xCC	0xB166

SFR Definition 25.1. SPI0CFG: SPI0 Configuration

Bit	7	6	5	4	3	2	1	0
Name	SPIBSY	MSTEN	CKPHA	CKPOL	SLVSEL	NSSIN	SRMT	RXBMT
Type	R	R/W	R/W	R/W	R	R	R	R
Reset	0	0	0	0	0	1	1	1

SFR Address = 0xA1

Bit	Name	Function
7	SPIBSY	**SPI Busy.** This bit is set to logic 1 when a SPI transfer is in progress (master or slave mode).
6	MSTEN	**Master Mode Enable.** 0: Disable master mode. Operate in slave mode. 1: Enable master mode. Operate as a master.
5	CKPHA	**SPI0 Clock Phase.** 0: Data centered on first edge of SCK period.* 1: Data centered on second edge of SCK period.*
4	CKPOL	**SPI0 Clock Polarity.** 0: SCK line low in idle state. 1: SCK line high in idle state.
3	SLVSEL	**Slave Selected Flag.** This bit is set to logic 1 whenever the NSS pin is low indicating SPI0 is the selected slave. It is cleared to logic 0 when NSS is high (slave not selected). This bit does not indicate the instantaneous value at the NSS pin, but rather a de-glitched version of the pin input.
2	NSSIN	**NSS Instantaneous Pin Input.** This bit mimics the instantaneous value that is present on the NSS port pin at the time that the register is read. This input is not de-glitched.
1	SRMT	**Shift Register Empty (valid in slave mode only).** This bit will be set to logic 1 when all data has been transferred in/out of the shift register, and there is no new information available to read from the transmit buffer or write to the receive buffer. It returns to logic 0 when a data byte is transferred to the shift register from the transmit buffer or by a transition on SCK. SRMT = 1 when in Master Mode.
0	RXBMT	**Receive Buffer Empty (valid in slave mode only).** This bit will be set to logic 1 when the receive buffer has been read and contains no new information. If there is new information available in the receive buffer that has not been read, this bit will return to logic 0. RXBMT = 1 when in Master Mode.

Note: In slave mode, data on MOSI is sampled in the center of each data bit. In master mode, data on MISO is sampled one SYSCLK before the end of each data bit, to provide maximum settling time for the slave device. See Table 25.1 for timing parameters.

26.1. Supporting Documents

It is assumed the reader is familiar with or has access to the following supporting documents:

1. The I2C-Bus and How to Use It (including specifications), Philips Semiconductor.
2. The I2C-Bus Specification—Version 2.0, Philips Semiconductor.
3. System Management Bus Specification—Version 1.1, SBS Implementers Forum.

26.2. SMBus Configuration

Figure 26.2 shows a typical SMBus configuration. The SMBus specification allows any recessive voltage between 3.0 V and 5.0 V; different devices on the bus may operate at different voltage levels. The bi-directional SCL (serial clock) and SDA (serial data) lines must be connected to a positive power supply voltage through a pullup resistor or similar circuit. Every device connected to the bus must have an open-drain or open-collector output for both the SCL and SDA lines, so that both are pulled high (recessive state) when the bus is free. The maximum number of devices on the bus is limited only by the requirement that the rise and fall times on the bus not exceed 300 ns and 1000 ns, respectively.

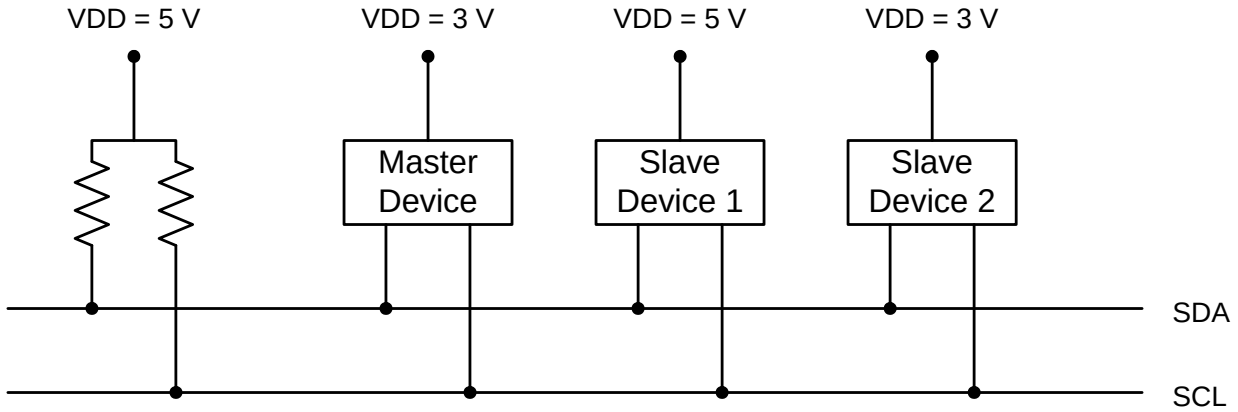

Figure 26.2. Typical SMBus Configuration

26.3. SMBus Operation

Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver (WRITE), and data transfers from an addressed slave transmitter to a master receiver (READ). The master device initiates both types of data transfers and provides the serial clock pulses on SCL. The SMBus interface may operate as a master or a slave, and multiple master devices on the same bus are supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration. Note that it is not necessary to specify one device as the Master in a system; any device who transmits a START and a slave address becomes the master for the duration of that transfer.

A typical SMBus transaction consists of a START condition followed by an address byte (Bits7–1: 7-bit slave address; Bit0: R/W direction bit), one or more bytes of data, and a STOP condition. Bytes that are received (by a master or slave) are acknowledged (ACK) with a low SDA during a high SCL (see Figure 26.3). If the receiving device does not ACK, the transmitting device will read a NACK (not acknowledge), which is a high SDA during a high SCL.

The direction bit (R/W) occupies the least-significant bit position of the address byte. The direction bit is set to logic 1 to indicate a "READ" operation and cleared to logic 0 to indicate a "WRITE" operation.

overflow after 25 ms (and SMBTOE set), the Timer 3 interrupt service routine can be used to reset (disable and re-enable) the SMBus in the event of an SCL low timeout.

26.3.5. SCL High (SMBus Free) Timeout

The SMBus specification stipulates that if the SCL and SDA lines remain high for more that 50 µs, the bus is designated as free. When the SMBFTE bit in SMB0CF is set, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods (as defined by the timer configured for the SMBus clock source). If the SMBus is waiting to generate a Master START, the START will be generated following this timeout. A clock source is required for free timeout detection, even in a slave-only implementation.

26.4. Using the SMBus

The SMBus can operate in both Master and Slave modes. The interface provides timing and shifting control for serial transfers; higher level protocol is determined by user software. The SMBus interface provides the following application-independent features:

- Byte-wise serial data transfers
- Clock signal generation on SCL (Master Mode only) and SDA data synchronization
- Timeout/bus error recognition, as defined by the SMB0CF configuration register
- START/STOP timing, detection, and generation
- Bus arbitration
- Interrupt generation
- Status information
- Optional hardware recognition of slave address and automatic acknowledgement of address/data

SMBus interrupts are generated for each data byte or slave address that is transferred. When hardware acknowledgement is disabled, the point at which the interrupt is generated depends on whether the hardware is acting as a data transmitter or receiver. When a transmitter (i.e., sending address/data, receiving an ACK), this interrupt is generated after the ACK cycle so that software may read the received ACK value; when receiving data (i.e., receiving address/data, sending an ACK), this interrupt is generated before the ACK cycle so that software may define the outgoing ACK value. If hardware acknowledgement is enabled, these interrupts are always generated after the ACK cycle. See Section 26.5 for more details on transmission sequences.

Interrupts are also generated to indicate the beginning of a transfer when a master (START generated), or the end of a transfer when a slave (STOP detected). Software should read the SMB0CN (SMBus Control register) to find the cause of the SMBus interrupt. The SMB0CN register is described in Section 26.4.2; Table 26.5 provides a quick SMB0CN decoding reference.

26.4.1. SMBus Configuration Register

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

SFR Definition 26.1. SMB0CF: SMBus Clock/Configuration

Bit	7	6	5	4	3	2	1	0
Name	ENSMB	INH	BUSY	EXTHOLD	SMBTOE	SMBFTE	SMBCS[1:0]	
Type	R/W	R/W	R	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xC1

Bit	Name	Function
7	ENSMB	**SMBus Enable.** This bit enables the SMBus interface when set to 1. When enabled, the interface constantly monitors the SDA and SCL pins.
6	INH	**SMBus Slave Inhibit.** When this bit is set to logic 1, the SMBus does not generate an interrupt when slave events occur. This effectively removes the SMBus slave from the bus. Master Mode interrupts are not affected.
5	BUSY	**SMBus Busy Indicator.** This bit is set to logic 1 by hardware when a transfer is in progress. It is cleared to logic 0 when a STOP or free-timeout is sensed.
4	EXTHOLD	**SMBus Setup and Hold Time Extension Enable.** This bit controls the SDA setup and hold times according to Table 26.2. 0: SDA Extended Setup and Hold Times disabled. 1: SDA Extended Setup and Hold Times enabled.
3	SMBTOE	**SMBus SCL Timeout Detection Enable.** This bit enables SCL low timeout detection. If set to logic 1, the SMBus forces Timer 3 to reload while SCL is high and allows Timer 3 to count when SCL goes low. If Timer 3 is configured to Split Mode, only the High Byte of the timer is held in reload while SCL is high. Timer 3 should be programmed to generate interrupts at 25 ms, and the Timer 3 interrupt service routine should reset SMBus communication.
2	SMBFTE	**SMBus Free Timeout Detection Enable.** When this bit is set to logic 1, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods.
1:0	SMBCS[1:0]	**SMBus Clock Source Selection.** These two bits select the SMBus clock source, which is used to generate the SMBus bit rate. The selected device should be configured according to Equation 26.1. 00: Timer 0 Overflow 01: Timer 1 Overflow 10: Timer 2 High Byte Overflow 11: Timer 2 Low Byte Overflow

Table 26.3. Sources for Hardware Changes to SMB0CN

Bit	Set by Hardware When:	Cleared by Hardware When:
MASTER	■ A START is generated.	■ A STOP is generated. ■ Arbitration is lost.
TXMODE	■ START is generated. ■ SMB0DAT is written before the start of an SMBus frame.	■ A START is detected. ■ Arbitration is lost. ■ SMB0DAT is not written before the start of an SMBus frame.
STA	■ A START followed by an address byte is received.	■ Must be cleared by software.
STO	■ A STOP is detected while addressed as a slave. ■ Arbitration is lost due to a detected STOP.	■ A pending STOP is generated.
ACKRQ	■ A byte has been received and an ACK response value is needed (only when hardware ACK is not enabled).	■ After each ACK cycle.
ARBLOST	■ A repeated START is detected as a MASTER when STA is low (unwanted repeated START). ■ SCL is sensed low while attempting to generate a STOP or repeated START condition. ■ SDA is sensed low while transmitting a 1 (excluding ACK bits).	■ Each time SI is cleared.
ACK	■ The incoming ACK value is low (ACKNOWLEDGE).	■ The incoming ACK value is high (NOT ACKNOWLEDGE).
SI	■ A START has been generated. ■ Lost arbitration. ■ A byte has been transmitted and an ACK/NACK received. ■ A byte has been received. ■ A START or repeated START followed by a slave address + R/W has been received. ■ A STOP has been received.	■ Must be cleared by software.

26.4.3. Hardware Slave Address Recognition

The SMBus hardware has the capability to automatically recognize incoming slave addresses and send an ACK without software intervention. Automatic slave address recognition is enabled by setting the EHACK bit in register SMB0ADM to 1. This will enable both automatic slave address recognition and automatic hardware ACK generation for received bytes (as a master or slave). More detail on automatic hardware ACK generation can be found in Section 26.4.2.2.

The registers used to define which address(es) are recognized by the hardware are the SMBus Slave Address register (SFR Definition 26.3) and the SMBus Slave Address Mask register (SFR Definition 26.4). A single address or range of addresses (including the General Call Address 0x00) can be specified using these two registers. The most-significant seven bits of the two registers are used to define which addresses will be ACKed. A 1 in bit positions of the slave address mask SLVM[6:0] enable a comparison between the received slave address and the hardware's slave address SLV[6:0] for those bits. A 0 in a bit of the slave address mask means that bit will be treated as a "don't care" for comparison purposes. In this

SFR Definition 28.11. TMR2L: Timer 2 Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2L[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCC

Bit	Name	Function
7:0	TMR2L[7:0]	**Timer 2 Low Byte.** In 16-bit mode, the TMR2L register contains the low byte of the 16-bit Timer 2. In 8-bit mode, TMR2L contains the 8-bit low byte timer value.

SFR Definition 28.12. TMR2H Timer 2 High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2H[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCD

Bit	Name	Function
7:0	TMR2H[7:0]	**Timer 2 Low Byte.** In 16-bit mode, the TMR2H register contains the high byte of the 16-bit Timer 2. In 8-bit mode, TMR2H contains the 8-bit high byte timer value.

set is then given (in PCA clocks) by Equation 29.5, where PCA0L is the value of the PCA0L register at the time of the update.

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

Equation 29.5. 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.

29.4.2. 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 29.5, this results in a WDT timeout interval of 256 PCA clock cycles, or 3072 system clock cycles. Table 29.3 lists some example timeout intervals for typical system clocks.

Table 29.3. 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.

29.5. Register Descriptions for PCA0

Following are detailed descriptions of the special function registers related to the operation of the PCA.