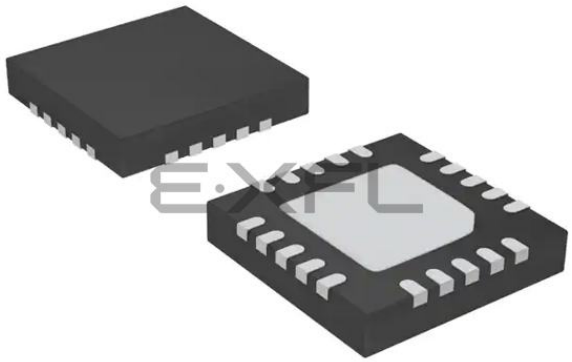

Welcome to E-XFL.COM

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	POR, PWM, Temp Sensor, WDT
Number of I/O	17
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 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-VFQFN Exposed Pad
Supplier Device Package	20-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f802-gm

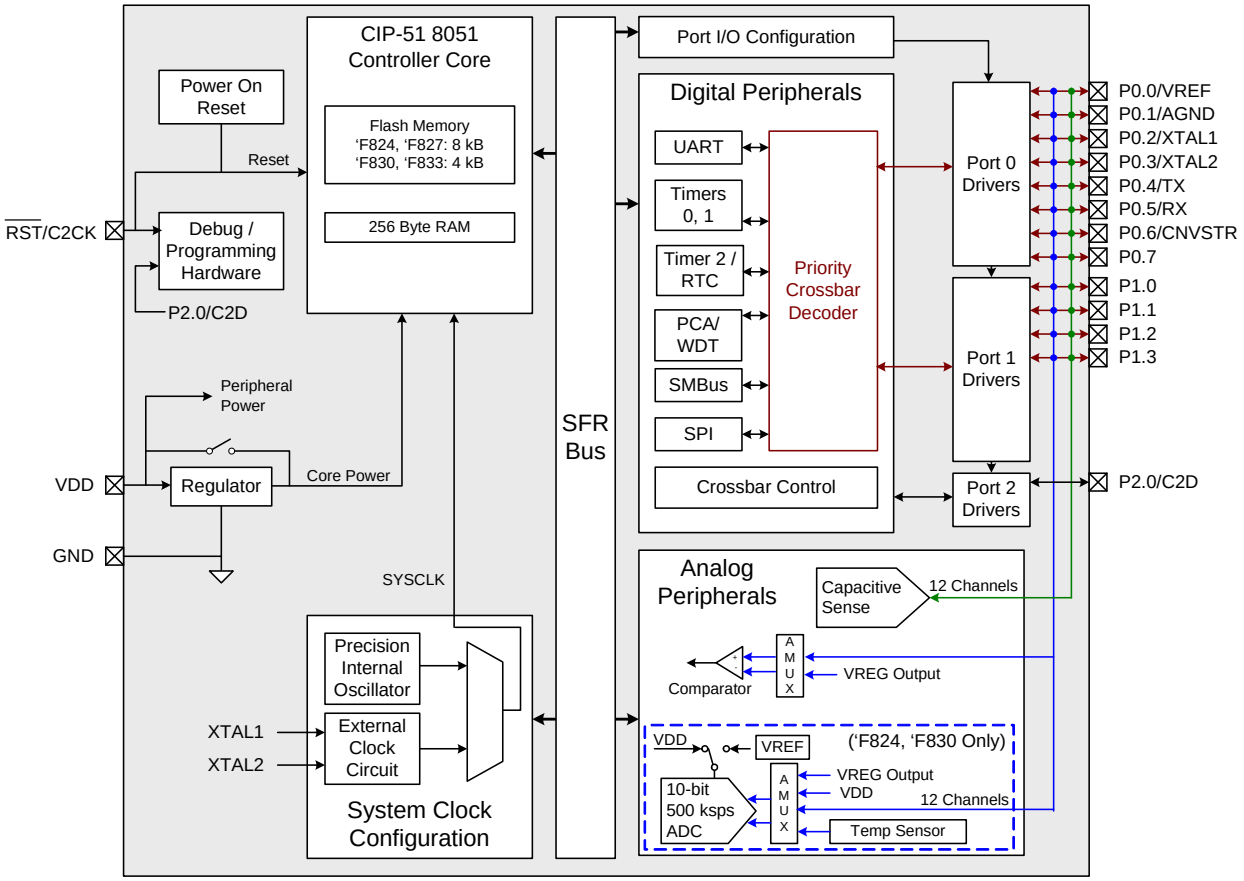

Figure 1.7. C8051F824, C8051F827, C8051F830, C8051F833 Block Diagram

Table 7.6. Flash Electrical Characteristics

Parameter	Conditions	Min	Typ	Max	Units
Flash Size (Note 1)	C8051F80x and C8051F810/1 C8051F812/3/4/5/6/7/8/9 and C8051F82x C8051F830/1/2/3/4/5		16384 8192 4096		bytes bytes bytes
Endurance (Erase/Write)		10000	—	—	cycles
Erase Cycle Time	25 MHz Clock	15	20	26	ms
Write Cycle Time	25 MHz Clock	15	20	26	µs
Clock Speed during Flash Write/Erase Operations		1	—	—	MHz

Note: Includes Security Lock Byte.

Table 7.7. Internal High-Frequency Oscillator Electrical Characteristics

V_{DD} = 1.8 to 3.6 V; T_A = –40 to +85 °C unless otherwise specified. Use factory-calibrated settings.

Parameter	Conditions	Min	Typ	Max	Units
Oscillator Frequency	IFCN = 11b	24	24.5	25	MHz
Oscillator Supply Current	25 °C, V_{DD} = 3.0 V, OSCICN.7 = 1, OCSICN.5 = 0	—	350	650	µA

Table 7.8. Capacitive Sense Electrical Characteristics

V_{DD} = 1.8 to 3.6 V; T_A = –40 to +85 °C unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Conversion Time	Single Conversion	26	38	50	µs
Capacitance per Code		—	1	—	fF
External Capacitive Load		—	—	45	pF
Quantization Noise[1]	RMS Peak-to-Peak	— —	3 20	— —	fF fF
Supply Current	CS module bias current, 25 °C CS module alone, maximum code output, 25 °C Wake-on-CS Threshold[2], 25 °C	— — —	40 75 150	60 105 165	µA µA µA

Notes:
1. RMS Noise is equivalent to one standard deviation. Peak-to-peak noise encompasses ±3.3 standard deviations.
2. Includes only current from regulator, CS module, and MCU in suspend mode.

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. $\mathrm{AD0SC} = \frac{\mathrm{SYSCLK}}{\mathrm{CLK_{SAR}}} - 1$
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 13.7. CS0THH: Capacitive Sense Comparator Threshold High Byte

Bit	7	6	5	4	3	2	1	0
Name	CS0THH[7:0]							
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 = 0x97

Bit	Name	Description
7:0	CS0THH[7:0]	**CS0 Comparator Threshold High Byte.** High byte of the 16-bit value compared to the Capacitive Sense conversion result.

SFR Definition 13.8. CS0THL: Capacitive Sense Comparator Threshold Low Byte

Bit	7	6	5	4	3	2	1	0
Name	CS0THL[7:0]							
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 = 0x96

Bit	Name	Description
7:0	CS0THL[7:0]	**CS0 Comparator Threshold Low Byte.** Low byte of the 16-bit value compared to the Capacitive Sense conversion result.

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

Table 17.2. Special Function Registers

SFRs are listed in alphabetical order. All undefined SFR locations are reserved

Register	Address	Description	Page
ACC	0xE0	Accumulator	89
ADC0CF	0xBC	ADC0 Configuration	50
ADC0CN	0xE8	ADC0 Control	52
ADC0GTH	0xC4	ADC0 Greater-Than Compare High	53
ADC0GTL	0xC3	ADC0 Greater-Than Compare Low	53
ADC0H	0xBE	ADC0 High	51
ADC0L	0xBD	ADC0 Low	51
ADC0LTH	0xC6	ADC0 Less-Than Compare Word High	54
ADC0LTL	0xC5	ADC0 Less-Than Compare Word Low	54
ADC0MX	0xBB	AMUX0 Multiplexer Channel Select	57
B	0xF0	B Register	90
CKCON	0x8E	Clock Control	210
CLKSEL	0xA9	Clock Select	210
CPT0CN	0x9B	Comparator0 Control	67
CPT0MD	0x9D	Comparator0 Mode Selection	68
CPT0MX	0x9F	Comparator0 MUX Selection	70
CRC0AUTO	0xD2	CRC0 Automatic Control Register	165
CRC0CN	0xCE	CRC0 Control	163
CRC0CNT	0xD3	CRC0 Automatic Flash Sector Count	165
CRC0DATA	0xDE	CRC0 Data Output	164
CRC0FLIP	0xCF	CRC0 Bit Flip	166
CRC0IN	0xDD	CRC Data Input	164
CS0THH	0x97	CS0 Digital Compare Threshold High	79
CS0THL	0x96	CS0 Digital Compare Threshold High	79
CS0CN	0xB0	CS0 Control	75
CS0DH	0xAC	CS0 Data High	77
CS0DL	0xAB	CS0 Data Low	77

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.

features of the C8051F80x-83x devices.

Table 19.1. Flash Security Summary

Action	C2 Debug Interface	User Firmware executing from:	
		an unlocked page	a locked page
Read, Write or Erase unlocked pages (except page with Lock Byte)	Permitted	Permitted	Permitted
Read, Write or Erase locked pages (except page with Lock Byte)	Not Permitted	FEDR	Permitted
Read or Write page containing Lock Byte (if no pages are locked)	Permitted	Permitted	Permitted
Read or Write page containing Lock Byte (if any page is locked)	Not Permitted	FEDR	Permitted
Read contents of Lock Byte (if no pages are locked)	Permitted	Permitted	Permitted
Read contents of Lock Byte (if any page is locked)	Not Permitted	FEDR	Permitted
Erase page containing Lock Byte (if no pages are locked)	Permitted	FEDR	FEDR
Erase page containing Lock Byte—Unlock all pages (if any page is locked)	Only by C2DE	FEDR	FEDR
Lock additional pages (change 1s to 0s in the Lock Byte)	Not Permitted	FEDR	FEDR
Unlock individual pages (change 0s to 1s in the Lock Byte)	Not Permitted	FEDR	FEDR
Read, Write or Erase Reserved Area	Not Permitted	FEDR	FEDR

C2DE—C2 Device Erase (Erases all Flash pages including the page containing the Lock Byte)
FEDR—Not permitted; Causes Flash Error Device Reset (FERROR bit in RSTSRC is 1 after reset)

- All prohibited operations that are performed via the C2 interface are ignored (do not cause device reset).
- Locking any Flash page also locks the page containing the Lock Byte.
- Once written to, the Lock Byte cannot be modified except by performing a C2 Device Erase.
- If user code writes to the Lock Byte, the Lock does not take effect until the next device reset.

19.4. Flash Write and Erase Guidelines

Any system which contains routines which write or erase Flash memory from software involves some risk that the write or erase routines will execute unintentionally if the CPU is operating outside its specified operating range of VDD, system clock frequency, or temperature. This accidental execution of Flash modifying code can result in alteration of Flash memory contents causing a system failure that is only recoverable by re-Flashing the code in the device.

To help prevent the accidental modification of Flash by firmware, the VDD Monitor must be enabled and enabled as a reset source on C8051F80x-83x devices for the Flash to be successfully modified. **If either the VDD Monitor or the VDD Monitor reset source is not enabled, a Flash Error Device Reset will be generated when the firmware attempts to modify the Flash.**

21.2. Power-Fail Reset / V_{DD} Monitor

When a power-down transition or power irregularity causes V_{DD} to drop below V_{RST}, the power supply monitor will drive the $\overline{RST}$ pin low and hold the CIP-51 in a reset state (see Figure 21.2). When V_{DD} returns to a level above V_{RST}, the CIP-51 will be released from the reset state. Even though internal data memory contents are not altered by the power-fail reset, it is impossible to determine if V_{DD} dropped below the level required for data retention. If the PORSF flag reads 1, the data may no longer be valid. The V_{DD} monitor is enabled and selected as a reset source after power-on resets. Its defined state (enabled/disabled) is not altered by any other reset source. For example, if the V_{DD} monitor is disabled by code and a software reset is performed, the V_{DD} monitor will still be disabled after the reset.

Important Note: If the V_{DD} monitor is being turned on from a disabled state, it should be enabled before it is selected as a reset source. Selecting the V_{DD} monitor as a reset source before it is enabled and stabilized may cause a system reset. In some applications, this reset may be undesirable. If this is not desirable in the application, a delay should be introduced between enabling the monitor and selecting it as a reset source. The procedure for enabling the V_{DD} monitor and configuring it as a reset source from a disabled state is shown below:

1. Enable the V_{DD} monitor (VDMEN bit in VDM0CN = 1).
2. If necessary, wait for the V_{DD} monitor to stabilize.
3. Select the V_{DD} monitor as a reset source (PORSF bit in RSTSRC = 1).

See Figure 21.2 for V_{DD} monitor timing; note that the power-on-reset delay is not incurred after a V_{DD} monitor reset. See Section “7. Electrical Characteristics” on page 39 for complete electrical characteristics of the V_{DD} monitor.

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 24.1. CRC0CN: CRC0 Control

Bit	7	6	5	4	3	2	1	0
Name				CRC0SEL	CRC0INIT	CRC0VAL	CRC0PNT[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 = 0xCE

Bit	Name	Function
7:5	Unused	Read = 000b; Write = Don't Care.
4	CRC0SEL	**CRC0 Polynomial Select Bit.** This bit selects the CRC0 polynomial and result length (32-bit or 16-bit). 0: CRC0 uses the 32-bit polynomial 0x04C11DB7 for calculating the CRC result. 1: CRC0 uses the 16-bit polynomial 0x1021 for calculating the CRC result.
3	CRC0INIT	**CRC0 Result Initialization Bit.** Writing a 1 to this bit initializes the entire CRC result based on CRC0VAL.
2	CRC0VAL	**CRC0 Set Value Initialization Bit.** This bit selects the set value of the CRC result. 0: CRC result is set to 0x00000000 on write of 1 to CRC0INIT. 1: CRC result is set to 0xFFFFFFFF on write of 1 to CRC0INIT.
1:0	CRC0PNT[1:0]	**CRC0 Result Pointer.** Specifies the byte of the CRC result to be read/written on the next access to CRC0DAT. The value of these bits will auto-increment upon each read or write. For CRC0SEL = 0: 00: CRC0DAT accesses bits 7–0 of the 32-bit CRC result. 01: CRC0DAT accesses bits 15–8 of the 32-bit CRC result. 10: CRC0DAT accesses bits 23–16 of the 32-bit CRC result. 11: CRC0DAT accesses bits 31–24 of the 32-bit CRC result. For CRC0SEL = 1: 00: CRC0DAT accesses bits 7–0 of the 16-bit CRC result. 01: CRC0DAT accesses bits 15–8 of the 16-bit CRC result. 10: CRC0DAT accesses bits 7–0 of the 16-bit CRC result. 11: CRC0DAT accesses bits 15–8 of the 16-bit CRC result.

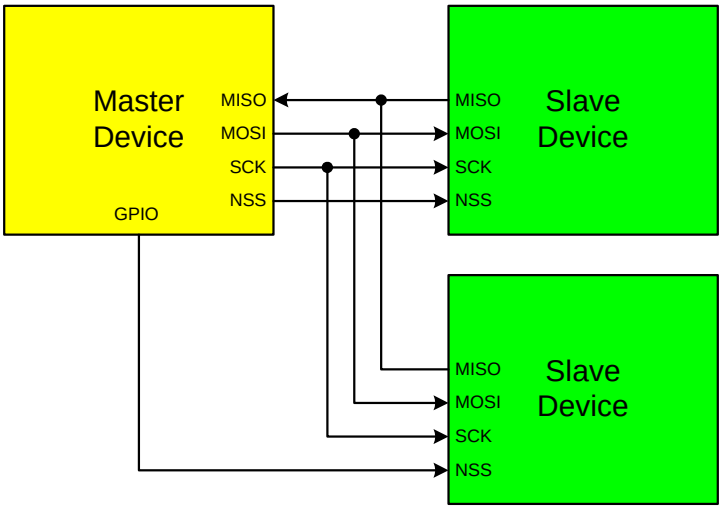

Figure 25.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram

25.3. 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 25.4 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 25.3 shows a connection diagram between a slave device in 3-wire slave mode and a master device.

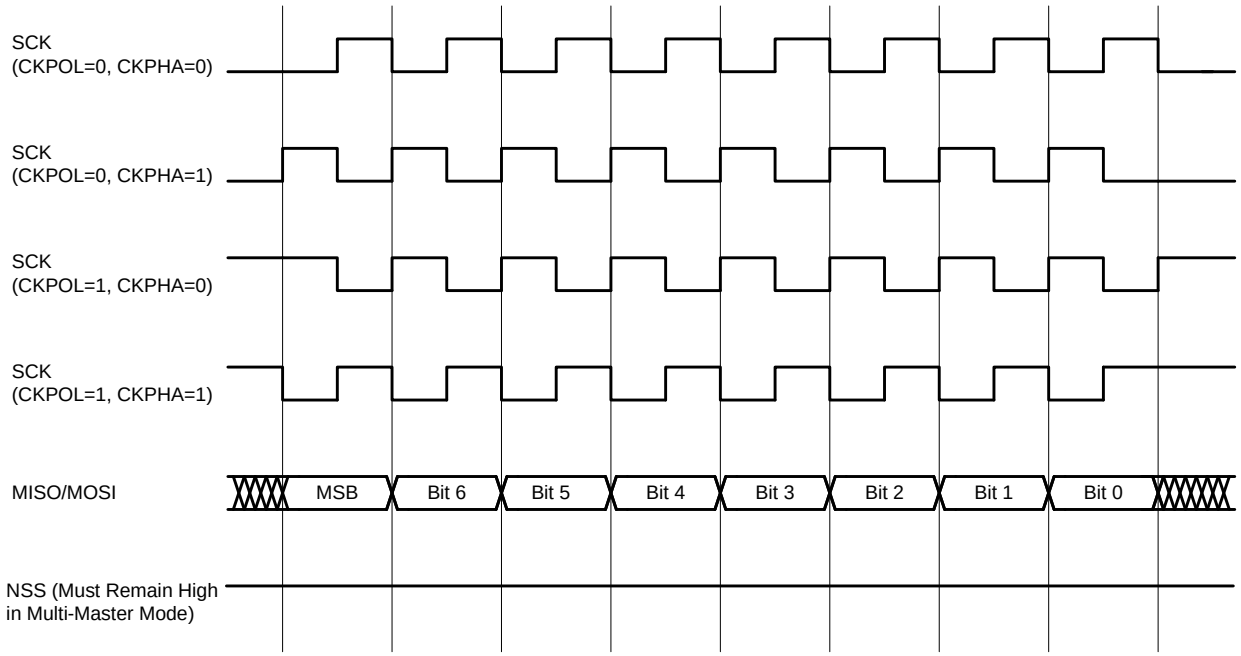

Figure 25.5. Master Mode Data/Clock Timing

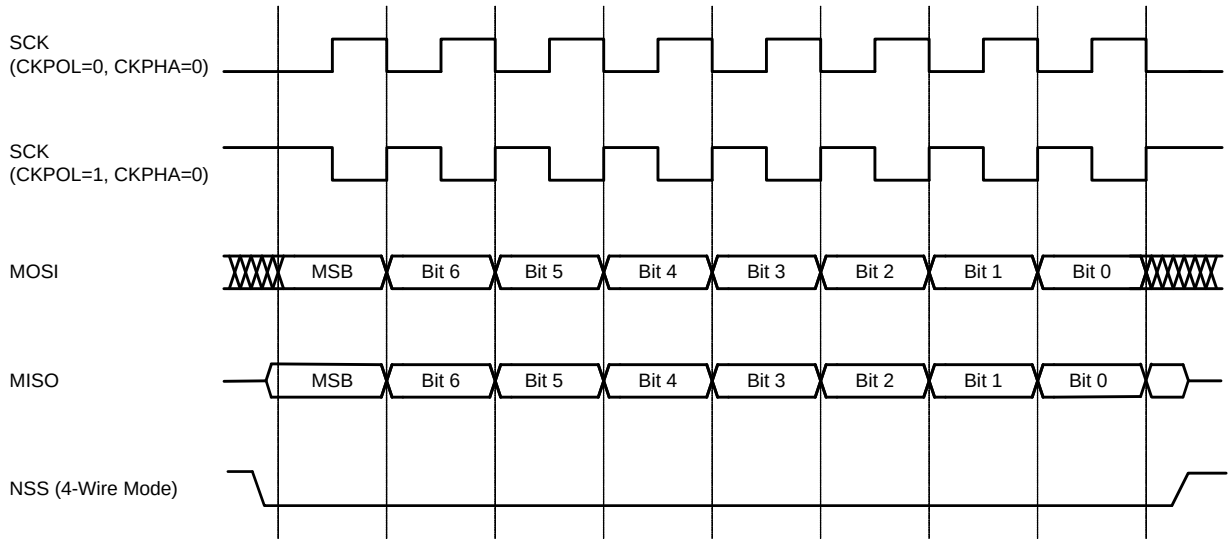

Figure 25.6. Slave Mode Data/Clock Timing (CKPHA = 0)

Table 25.1. SPI Slave Timing Parameters

Parameter	Description	Min	Max	Units
Master Mode Timing (See Figure 25.8 and Figure 25.9)				
T_{MCKH}	SCK High Time	1 x T_{SYSCLK}	—	ns
T_{MCKL}	SCK Low Time	1 x T_{SYSCLK}	—	ns
T_{MIS}	MISO Valid to SCK Shift Edge	1 x T_{SYSCLK} + 20	—	ns
T_{MIH}	SCK Shift Edge to MISO Change	0	—	ns
Slave Mode Timing (See Figure 25.10 and Figure 25.11)				
T_{SE}	NSS Falling to First SCK Edge	2 x T_{SYSCLK}	—	ns
T_{SD}	Last SCK Edge to NSS Rising	2 x T_{SYSCLK}	—	ns
T_{SEZ}	NSS Falling to MISO Valid	—	4 x T_{SYSCLK}	ns
T_{SDZ}	NSS Rising to MISO High-Z	—	4 x T_{SYSCLK}	ns
T_{CKH}	SCK High Time	5 x T_{SYSCLK}	—	ns
T_{CKL}	SCK Low Time	5 x T_{SYSCLK}	—	ns
T_{SIS}	MOSI Valid to SCK Sample Edge	2 x T_{SYSCLK}	—	ns
T_{SIH}	SCK Sample Edge to MOSI Change	2 x T_{SYSCLK}	—	ns
T_{SOH}	SCK Shift Edge to MISO Change	—	4 x T_{SYSCLK}	ns
T_{SLH}	Last SCK Edge to MISO Change (CKPHA = 1 ONLY)	6 x T_{SYSCLK}	8 x T_{SYSCLK}	ns
Note: T_{SYSCLK} is equal to one period of the device system clock (SYSCLK).				

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).

26.5. SMBus Transfer Modes

The SMBus interface may be configured to operate as master and/or slave. At any particular time, it will be operating in one of the following four modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. The SMBus interface enters Master Mode any time a START is generated, and remains in Master Mode until it loses an arbitration or generates a STOP. An SMBus interrupt is generated at the end of all SMBus byte frames. Note that the position of the ACK interrupt when operating as a receiver depends on whether hardware ACK generation is enabled. As a receiver, the interrupt for an ACK occurs **before** the ACK with hardware ACK generation disabled, and **after** the ACK when hardware ACK generation is enabled. As a transmitter, interrupts occur **after** the ACK, regardless of whether hardware ACK generation is enabled or not.

26.5.1. Write Sequence (Master)

During a write sequence, an SMBus master writes data to a slave device. The master in this transfer will be a transmitter during the address byte, and a transmitter during all data bytes. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 0 (WRITE). The master then transmits one or more bytes of serial data. After each byte is transmitted, an acknowledge bit is generated by the slave. The transfer is ended when the STO bit is set and a STOP is generated. Note that the interface will switch to Master Receiver Mode if SMB0DAT is not written following a Master Transmitter interrupt. Figure 26.5 shows a typical master write sequence. Two transmit data bytes are shown, though any number of bytes may be transmitted. Notice that all of the “data byte transferred” interrupts occur **after** the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

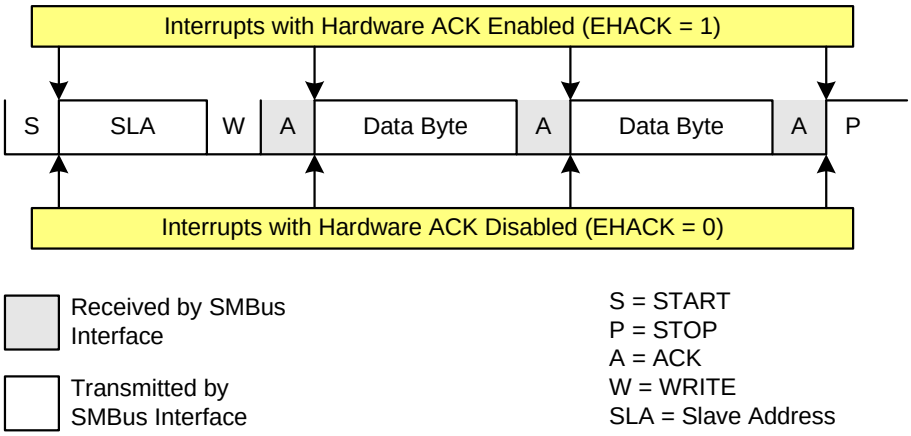

Figure 26.5. Typical Master Write Sequence

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

SFR Definition 28.3. TMOD: Timer Mode

Bit	7	6	5	4	3	2	1	0
Name	GATE1	C/T1	T1M[1:0]		GATE0	C/T0	T0M[1:0]	
Type	R/W	R/W	R/W		R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x89

Bit	Name	Function
7	GATE1	**Timer 1 Gate Control.** 0: Timer 1 enabled when TR1 = 1 irrespective of $\overline{\text{INT1}}$ logic level. 1: Timer 1 enabled only when TR1 = 1 AND $\overline{\text{INT1}}$ is active as defined by bit IN1PL in register IT01CF (see SFR Definition 18.7).
6	C/T1	**Counter/Timer 1 Select.** 0: Timer: Timer 1 incremented by clock defined by T1M bit in register CKCON. 1: Counter: Timer 1 incremented by high-to-low transitions on external pin (T1).
5:4	T1M[1:0]	**Timer 1 Mode Select.** These bits select the Timer 1 operation mode. 00: Mode 0, 13-bit Counter/Timer 01: Mode 1, 16-bit Counter/Timer 10: Mode 2, 8-bit Counter/Timer with Auto-Reload 11: Mode 3, Timer 1 Inactive
3	GATE0	**Timer 0 Gate Control.** 0: Timer 0 enabled when TR0 = 1 irrespective of $\overline{\text{INT0}}$ logic level. 1: Timer 0 enabled only when TR0 = 1 AND $\overline{\text{INT0}}$ is active as defined by bit IN0PL in register IT01CF (see SFR Definition 18.7).
2	C/T0	**Counter/Timer 0 Select.** 0: Timer: Timer 0 incremented by clock defined by T0M bit in register CKCON. 1: Counter: Timer 0 incremented by high-to-low transitions on external pin (T0).
1:0	T0M[1:0]	**Timer 0 Mode Select.** These bits select the Timer 0 operation mode. 00: Mode 0, 13-bit Counter/Timer 01: Mode 1, 16-bit Counter/Timer 10: Mode 2, 8-bit Counter/Timer with Auto-Reload 11: Mode 3, Two 8-bit Counter/Timers

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.

NOTES: