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	Active
Core Processor	8051
Core Size	8-Bit
Speed	25MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	DMA, LCD, POR, PWM, WDT
Number of I/O	57
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8.25K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.8V
Data Converters	A/D 16x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f966-b-gq

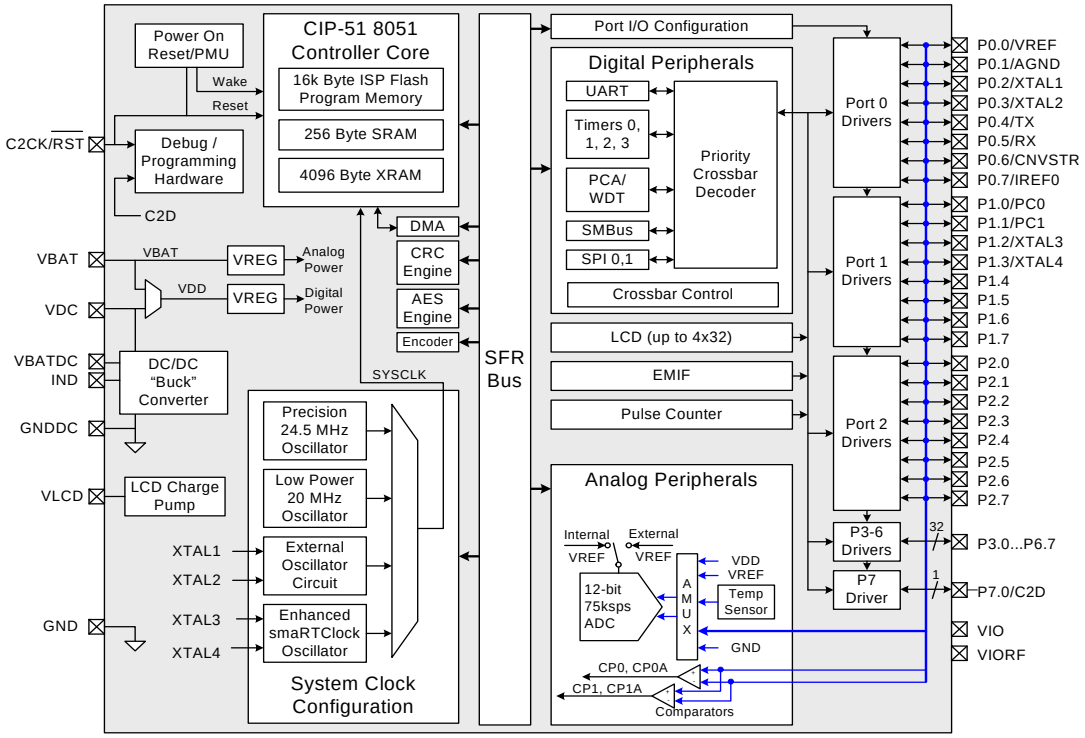

Figure 1.9. C8051F968 Block Diagram

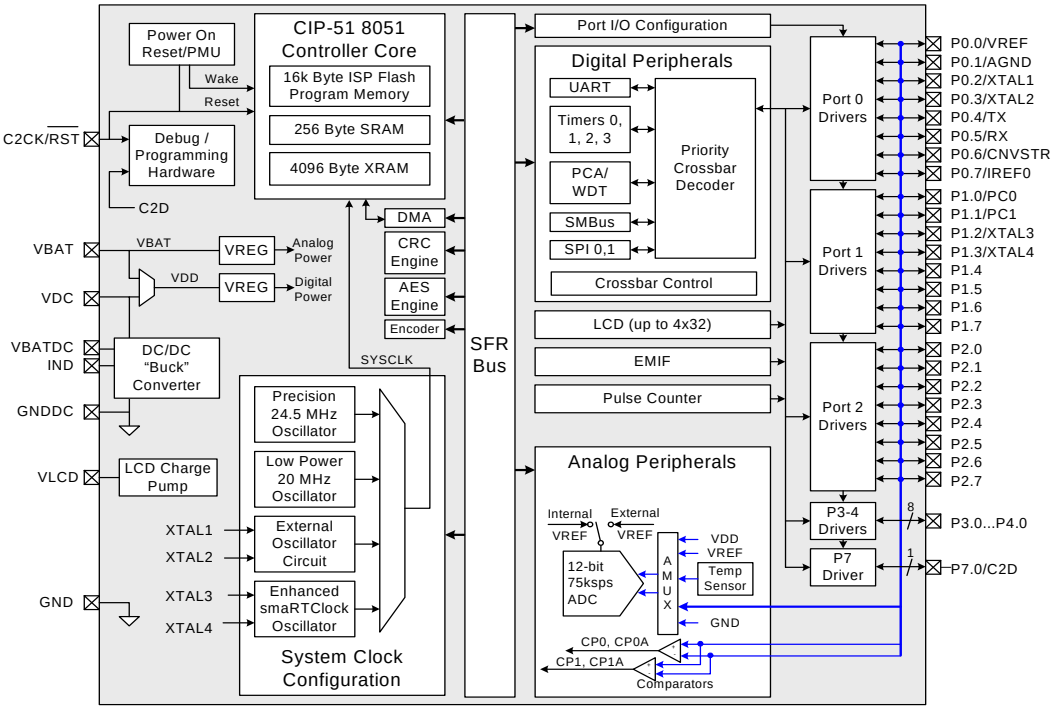

Figure 1.10. C8051F969 Block Diagram

Table 8.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/3
JNC rel	Jump if Carry is not set	2	2/3
JB bit, rel	Jump if direct bit is set	3	3/4
JNB bit, rel	Jump if direct bit is not set	3	3/4
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/4
Program Branching			
ACALL addr11	Absolute subroutine call	2	3
LCALL addr16	Long subroutine call	3	4
RET	Return from subroutine	1	5
RETI	Return from interrupt	1	5
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (relative address)	2	3
JMP @A+DPTR	Jump indirect relative to DPTR	1	3
JZ rel	Jump if A equals zero	2	2/3
JNZ rel	Jump if A does not equal zero	2	2/3
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/4
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/4
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/4
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/5
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/3
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/4
NOP	No operation	1	1

Notes on Registers, Operands and Addressing Modes:

Rn—Register R0–R7 of the currently selected register bank.

@Ri—Data RAM location addressed indirectly through R0 or R1.

rel—8-bit, signed (twos complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.

direct—8-bit internal data location's address. This could be a direct-access Data RAM location (0x00–0x7F) or an SFR (0x80–0xFF).

#data—8-bit constant

#data16—16-bit constant

bit—Direct-accessed bit in Data RAM or SFR

addr11—11-bit destination address used by ACALL and AJMP. The destination must be within the same 2 kB page of program memory as the first byte of the following instruction.

addr16—16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8 kB program memory space.

There is one unused opcode (0xA5) that performs the same function as NOP.

11. Direct Memory Access (DMA0)

An on-chip direct memory access (DMA0) is included on the C8051F96x devices. The DMA0 subsystem allows autonomous variable-length data transfers between XRAM and peripheral SFR registers without CPU intervention. During DMA0 operation, the CPU is free to perform some other tasks. In order to save total system power consumption, the CPU and flash can be powered down. DMA0 improves the system performance and efficiency with high data throughput peripherals.

DMA0 contains seven independent channels, common control registers, and a DMA0 Engine (see Figure 11.1). Each channel includes a register that assigns a peripheral to the channel, a channel control register, and a set of SFRs that include XRAM address information and SFR address information used by the channel during a data transfer. The DMA0 architecture is described in detail in Section 11.1.

The DMA0 in C8051F96x devices supports four peripherals: AES0, ENC0, CRC1, and SPI1. Peripherals with DMA0 capability should be configured to work with the DMA0 through their own registers. The DMA0 provides up to seven channels, and each channel can be configured for one of nine possible data transfer functions:

- XRAM to ENC0L/M/H
- ENC0L/M/H sfrs to XRAM
- XRAM to CRC1IN sfr
- XRAM to SPI1DAT sfr
- SPI1DAT sfr to XRAM
- XRAM to AES0KIN sfr
- XRAM to AES0BIN sfr
- XRAM to AES0XIN sfr
- AES0YOUT sfr to XRAM

The DMA0 subsystem signals the MCU through a set of interrupt service routine flags. Interrupts can be generated when the DMA0 transfers half of the data length or full data length on any channel.

The 16-bit C8051F96x 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 the final remainder (CRC value)
   return CRC_acc;
}
```

The following table lists several input values and the associated outputs using the 16-bit C8051F96x CRC algorithm:

Table 12.1. Example 16-bit CRC Outputs

Input	Output
0x63	0xBD35
0x8C	0xB1F4
0x7D	0x4ECA
0xAA, 0xBB, 0xCC	0x6CF6
0x00, 0x00, 0xAA, 0xBB, 0xCC	0xB166

SFR Definition 14.1. AES0BCFG: AES Block Configuration

Bit	7	6	5	4	3	2	1	0
Name			DONE	BUSY	EN	ENC	KSIZE	
Type	R	R	R/W	R	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xE9; SFR page = 0x2; Not bit-Addressable

Bit	Name	Function
5	DONE	**Done Flag.** This bit is set upon completion of an encryption operation. When used with the DMA, the DONE bit signals the start of the out transfer. When used without the DMA, the done flag indicates data is ready to be read from AES0YOUT. The DONE bit is not cleared by hardware and must be cleared to zero by software at the start of the next encryption operation.
4	BUSY	**AES BUSY.** This bit is set while the AES block is engaged in an encryption or decryption operation. This bit is read only.
3	EN	**AES Enable.** This bit should be set to 1 to initiate an encryption or decryption operation. Clearing this bit to 0 will reset the AES module.
2	ENC	**Encryption/Decryption Select.** This is set to 1 to select an encryption operation. Clearing this bit to 0 will select a decryption operation.
1:0	KSIZE[1:0]	**AES Key Size.** These bits select the key size for encryption/decryption. The encryption/decryption time depends on the key size selected. 00: Select 128-bits (16-bytes). Encryption/decryption takes 218 clocks. 01: Select 198-bits (24-bytes). Encryption/decryption takes 274 clocks. 10: Select 256-bits (32-bytes). Encryption/decryption takes 298 clocks. 11: Reserved

The level of flash security depends on the flash access method. The three flash access methods that can be restricted are reads, writes, and erases from the C2 debug interface, user firmware executing on unlocked pages, and user firmware executing on locked pages. Table 18.1 summarizes the flash security features of the C8051F96x devices.

Table 18.1. 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	Flash Error Reset	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	Flash Error Reset	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	Flash Error Reset	Permitted
Erase page containing Lock Byte (if no pages are locked)	Permitted	Flash Error Reset	Flash Error Reset
Erase page containing Lock Byte—Unlock all pages (if any page is locked)	C2 Device Erase Only	Flash Error Reset	Flash Error Reset
Lock additional pages (change 1s to 0s in the Lock Byte)	Not Permitted	Flash Error Reset	Flash Error Reset
Unlock individual pages (change 0s to 1s in the Lock Byte)	Not Permitted	Flash Error Reset	Flash Error Reset
Read, Write or Erase Reserved Area	Not Permitted	Flash Error Reset	Flash Error Reset

C2 Device Erase—Erases all flash pages including the page containing the Lock Byte.

Flash Error Reset—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.

18.5.2. PSWE Maintenance

1. Reduce the number of places in code where the PSWE bit (b0 in PSCTL) is set to a 1. There should be exactly one routine in code that sets PSWE to a 1 to write flash bytes and one routine in code that sets both PSWE and PSEE both to a 1 to erase flash pages.
2. Minimize the number of variable accesses while PSWE is set to a 1. Handle pointer address updates and loop maintenance outside the "PSWE = 1;... PSWE = 0;" area. Code examples showing this can be found in “AN201: Writing to Flash from Firmware,” available from the Silicon Laboratories web site.
3. Disable interrupts prior to setting PSWE to a 1 and leave them disabled until after PSWE has been reset to 0. Any interrupts posted during the flash write or erase operation will be serviced in priority order after the flash operation has been completed and interrupts have been re-enabled by software.
4. Make certain that the flash write and erase pointer variables are not located in XRAM. See your compiler documentation for instructions regarding how to explicitly locate variables in different memory areas.
5. Add address bounds checking to the routines that write or erase flash memory to ensure that a routine called with an illegal address does not result in modification of the flash.

18.5.3. System Clock

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

Additional flash recommendations and example code can be found in “AN201: Writing to Flash from Firmware," available from the Silicon Laboratories website.

SFR Definition 22.1. VDM0CN: VDD Supply Monitor Control

Bit	7	6	5	4	3	2	1	0
Name	VDMEN	VDDSTAT	VDDOK	VDDOKIE	VBMEN	VBSTAT	VBOK	VBOKIE
Type	R/W	R	R	R/W	R/W	R	R	R/W
Reset	1	Varies	Varies	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xFF

Bit	Name	Function
7	VDMEN	**Digital Supply Monitor Enable (Power Select Switch Output).** 0: Digital Supply Monitor Disabled. 1: Digital Supply Monitor Enabled.
6	VDDSTAT	**Digital Supply Status.** This bit indicates the current digital power supply status. 0: Digital supply is at or below the V_{RST} threshold. 1: Digital supply is above the V_{RST} threshold.
5	VDDOK	**Digital Supply Status (Early Warning).** This bit indicates the current digital power supply status. 0: Digital supply is at or below the VDD_{WARN} threshold. 1: Digital supply is above the VDD_{WARN} threshold.
4	VDDOKIE	**Digital Early Warning Interrupt Enable.** Enables the V_{DD} Early Warning Interrupt. 0: V_{DD} Early Warning Interrupt is disabled. 1: V_{DD} Early Warning Interrupt is enabled.
3	VBMEN	**Analog Supply Monitor Enable (VBAT Pin).** 0: Analog Supply Monitor Disabled. 1: Analog Supply Monitor Enabled.
2	VBSTAT	**Analog Supply Status.** This bit indicates the analog (VBAT) power supply status. 0: VBAT is at or below the V_{RST} threshold. 1: VBAT is above the V_{RST} threshold.
1	VBOK	**Analog Supply Status (Early Warning).** This bit indicates the current VBAT power supply status. 0: VBAT is at or below the VDD_{WARN} threshold. 1: VBAT is above the VDD_{WARN} threshold.
0	VBOKIE	**Analog Early Warning Interrupt Enable.** Enables the VBAT Early Warning Interrupt. 0: VBAT Early Warning Interrupt is disabled. 1: VBAT Early Warning Interrupt is enabled.

25. Low-Power Pulse Counter

The C8051F96x family of microcontrollers contains a low-power Pulse Counter module with advanced features, such as ultra low power input comparators, a wide range of pull up values with a self calibration engine, asymmetrical integrators for low pass filtering and switch debounce, single, dual, and quadrature modes of operation, two 24-bit counters, threshold comparators, and a variety of interrupt and sleep wake up capabilities. This combination of features provides water, gas, and heat metering system designers with an optimal tool for saving power while collecting meter usage data.

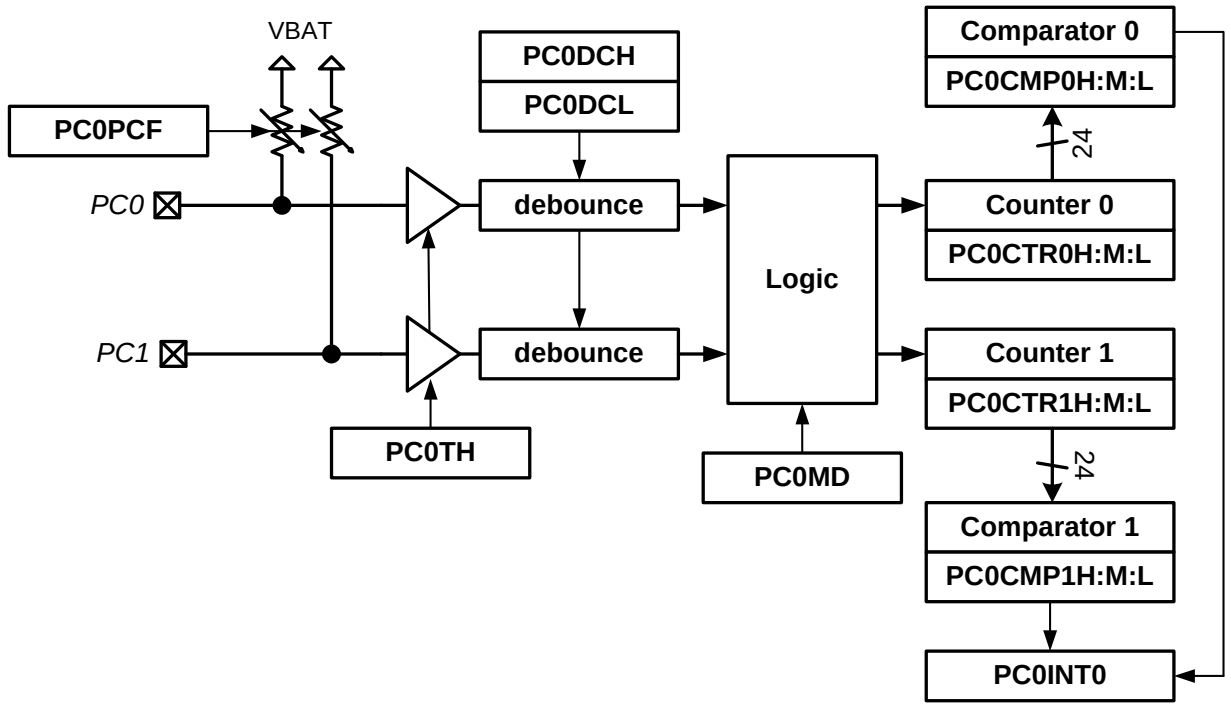

Figure 25.1. Pulse Counter Block Diagram

The low-power Pulse Counter is a low-power sleep-mode peripheral designed primarily to work meters using reed switches, including water and gas meters. The Pulse Counter is very flexible and can count pulses from many different types of sources.

The Pulse Counter operates in sleep mode to enable ultra-low power metering systems. The MCU does not have to wake up on every edge or transition and can remain in sleep mode while the Pulse Counter counts pulses for an extended period of time. The Pulse Counter includes two 24-bit counters. These counters can count up to 16,777,215 (2^{24}-1) transitions in sleep mode before overflowing. The Pulse Counter can wake up the MCU when one of the counters overflows. The Pulse Counter also has two 24-bit comparators. The comparators have the ability to wake up the MCU when the one of the counters reaches a predetermined threshold.

The Pulse Counter uses the RTC clock for sampling, de-bouncing, and managing the low-power pull-up resistors. The RTC must be enabled when counting pulses. The RTC alarms can wake up the MCU periodically to read the pulse counters, instead of using the Pulse Counter comparators. For example, the RTC can wake up the MCU every five minutes. The MCU can then read the Pulse Counter and transmit the information using the UART or a wireless transceiver.

25.1. Counting Modes

The Pulse Counter supports three different counting modes: single counter mode, dual counter mode, and quadrature counter mode. Figure 25.2 illustrates the three counter modes.

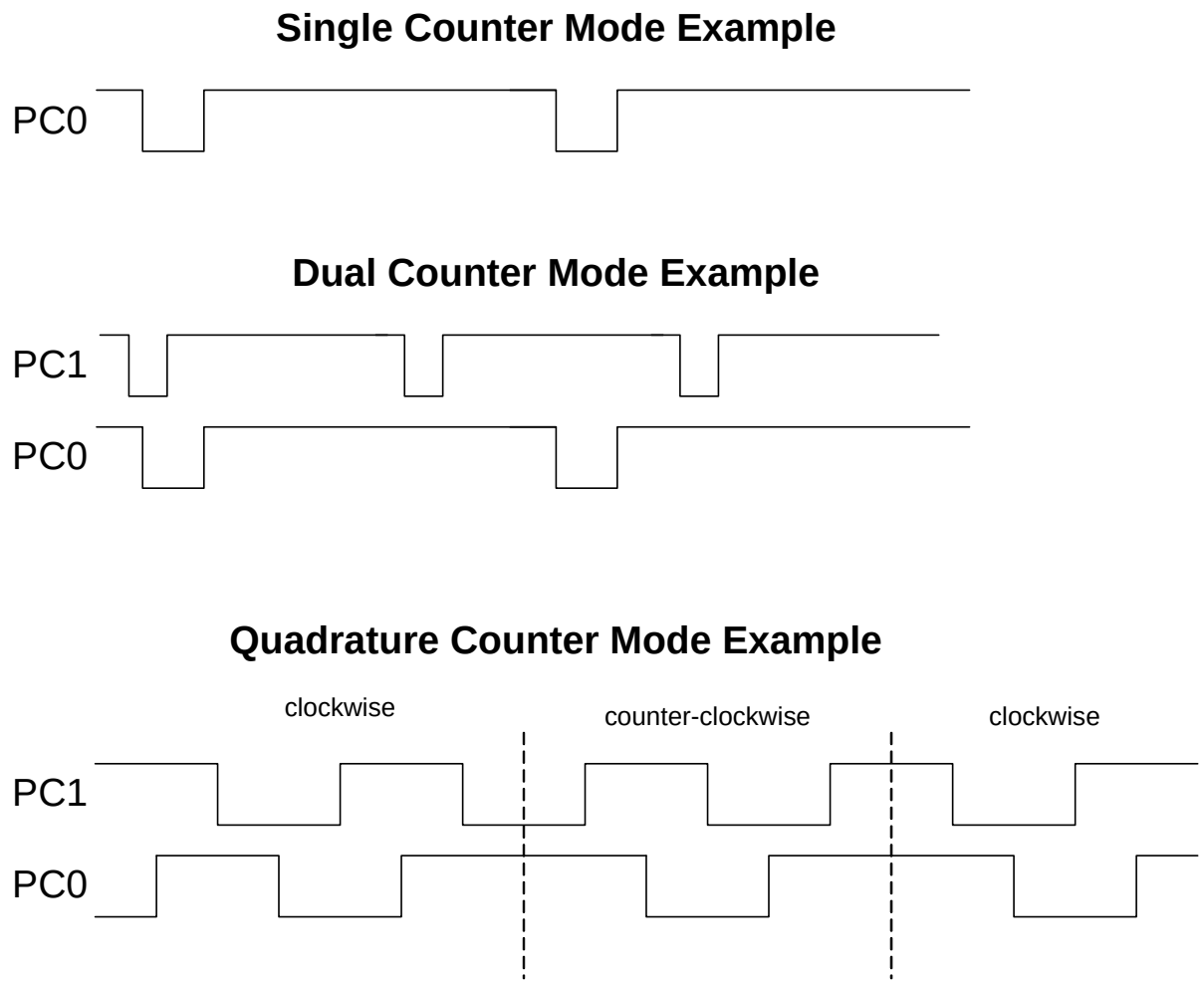

Figure 25.2. Mode Examples

The single counter mode uses only one Pulse Counter pin PC0 (P1.0) to count pulses from a single input channel. This mode uses only counter 0 and comparator. (Counter 1 and comparator 1 are not used.) The single counter mode supports only one meter-encoder with a single-channel output. A single-channel encoder is an effective solution when the metered fluid flows only in one direction. A single-channel encoder does not provide any direction information and does not support bidirectional fluid metering.

The dual counter mode supports two independent single-channel meters. Each meter has its own independent counter and comparator. Some of the global configuration settings apply to both channels, such as pull-up current, sampling rate, and debounce time. The dual mode may also be used for a redundant count using a two-channel non-quadrature encoder.

Quadrature counter mode supports a single two-channel quadrature meter encoder. The quadrature counter mode supports bidirectional encoders and applications with bidirectional fluid flow. In quadrature counter mode, clock-wise counts will increment counter 0, while counter clock-wise counts will increment counter 1. Subtracting counter 1 from counter 0 will yield the net position. If the normal fluid flow is clock-

SFR Definition 26.5. LCD0MSCF: LCD0 Master Configuration

Bit	7	6	5	4	3	2	1	0
Name							DCENSLP	CHPBYP
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	0

SFR Page = 0x2; SFR Address = 0xAC

Bit	Name	Function
7:2	Reserved	Read = 111111b. Must write 111111b.
1	DCENSLP	**DCDC Converter Enable in Sleep Mode** 0: DCDC is disabled in Sleep Mode. 1: DCDC is enabled in Sleep Mode.
0	CHPBYP	**LCD0 Charge Pump Bypass** This bit should be set to 1b in Contrast Control Mode 1 and Mode 2. 0: LCD0 Charge Pump is not bypassed. 1: LCD0 Charge Pump is bypassed.

SFR Definition 26.6. LCD0PWR: LCD0 Power

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

SFR Page = 0x2; SFR Address = 0xA4

Bit	Name	Function
7:4	Unused	Read = 0000b. Write = don't care.
3	MODE	**LCD0 Contrast Control Mode Selection.** 0: LCD0 Contrast Control Mode 1 or Mode 4 is selected. 1: LCD0 Contrast Control Mode 2 or Mode 3 is selected.
2:0	Reserved	Read = 001b. Must write 001b.

27.2.2. Assigning Port I/O Pins to Digital Functions

Any Port pins not assigned to analog functions may be assigned to digital functions or used as GPIO. Most digital functions rely on the Crossbar for pin assignment; however, some digital functions bypass the Crossbar in a manner similar to the analog functions listed above. **Port pins used by these digital functions and any Port pins selected for use as GPIO should have their corresponding bit in PnSKIP set to 1.** Table 27.2 shows all available digital functions and the potential mapping of Port I/O to each digital function.

Table 27.2. Port I/O Assignment for Digital Functions

Digital Function	Potentially Assignable Port Pins	SFR(s) used for Assignment
UART0, SPI0, SPI1, SMBus, CP0 and CP1 Outputs, System Clock Output, PCA0, Timer0 and Timer1 External Inputs.	Any Port pin available for assignment by the Crossbar. This includes P0.0–P2.7 pins which have their PnSKIP bit set to 0. **Note:** The Crossbar will always assign UART0 and SPI1 pins to fixed locations.	XBR0, XBR1, XBR2
Any pin used for GPIO	P0.0–P7.0	P0SKIP, P1SKIP, P2SKIP
External Memory Interface	P3.6–P6.7	EMI0CF

27.2.3. Assigning Port I/O Pins to External Digital Event Capture Functions

External digital event capture functions can be used to trigger an interrupt or wake the device from a low power mode when a transition occurs on a digital I/O pin. The digital event capture functions do not require dedicated pins and will function on both GPIO pins (PnSKIP = 1) and pins in use by the Crossbar (PnSKIP = 0). External digital even capture functions cannot be used on pins configured for analog I/O. Table 27.3 shows all available external digital event capture functions.

Table 27.3. Port I/O Assignment for External Digital Event Capture Functions

Digital Function	Potentially Assignable Port Pins	SFR(s) used for Assignment
External Interrupt 0	P0.0–P0.5, P1.6, P1.7	IT01CF
External Interrupt 1	P0.0–P0.4, P1.6, P1.7	IT01CF
Port Match	P0.0–P1.7	P0MASK, P0MAT P1MASK, P1MAT

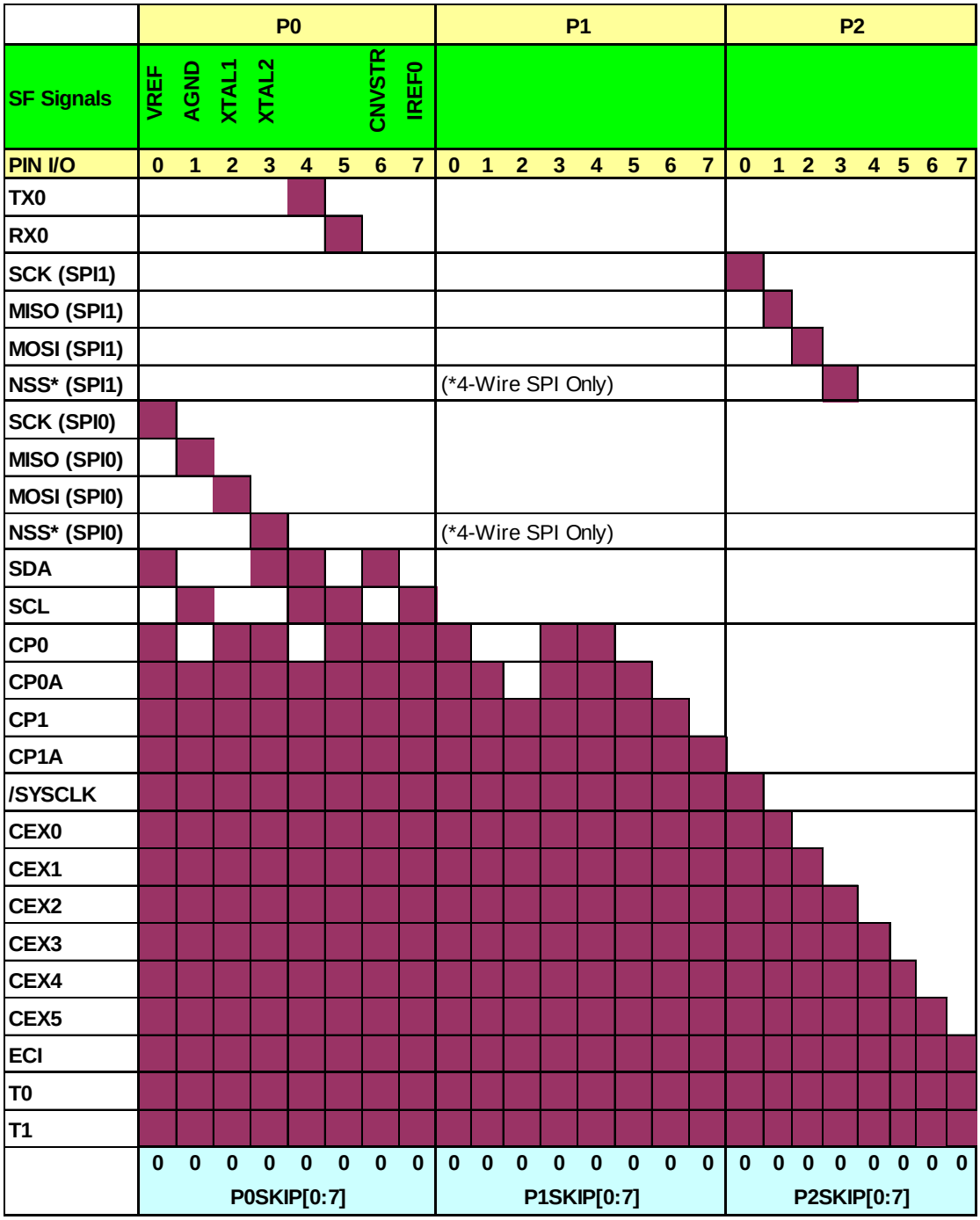

Figure 27.3. Crossbar Priority Decoder with No Pins Skipped

Table 28.1. SMBus Clock Source Selection

SMBCS1	SMBCS0	SMBus Clock Source
0	0	Timer 0 Overflow
0	1	Timer 1 Overflow
1	0	Timer 2 High Byte Overflow
1	1	Timer 2 Low Byte Overflow

The SMBCS1–0 bits select the SMBus clock source, which is used only when operating as a master or when the Free Timeout detection is enabled. When operating as a master, overflows from the selected source determine the absolute minimum SCL low and high times as defined in Equation 28.1. The selected clock source may be shared by other peripherals so long as the timer is left running at all times. For example, Timer 1 overflows may generate the SMBus and UART baud rates simultaneously. Timer configuration is covered in Section “32. Timers” on page 444.

$$T_{HighMin} = T_{LowMin} = \frac{1}{f_{ClockSourceOverflow}}$$

Equation 28.1. Minimum SCL High and Low Times

The selected clock source should be configured to establish the minimum SCL High and Low times as per Equation 28.1. When the interface is operating as a master (and SCL is not driven or extended by any other devices on the bus), the typical SMBus bit rate is approximated by Equation 28.1.

$$BitRate = \frac{f_{ClockSourceOverflow}}{3}$$

Equation 28.2. Typical SMBus Bit Rate

Figure 28.4 shows the typical SCL generation described by Equation 28.2. Notice that T_{HIGH} is typically twice as large as T_{LOW}. The actual SCL output may vary due to other devices on the bus (SCL may be extended low by slower slave devices, or driven low by contending master devices). The bit rate when operating as a master will never exceed the limits defined by Equation 28.2.

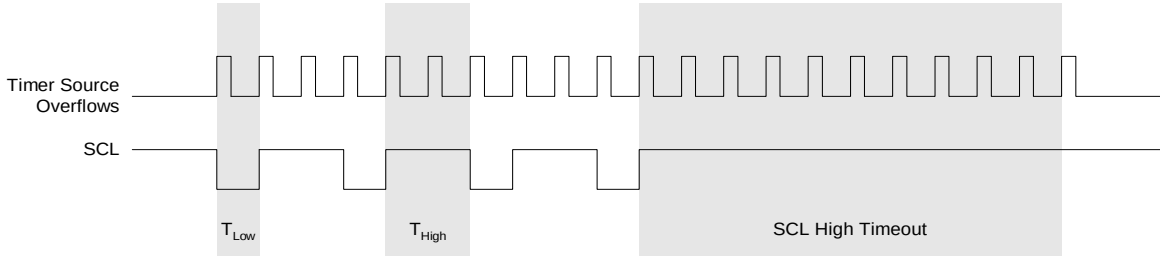

Figure 28.4. Typical SMBus SCL Generation

Setting the EXTHOLD bit extends the minimum setup and hold times for the SDA line. The minimum SDA setup time defines the absolute minimum time that SDA is stable before SCL transitions from low-to-high. The minimum SDA hold time defines the absolute minimum time that the current SDA value remains stable after SCL transitions from high-to-low. EXTHOLD should be set so that the minimum setup and hold times meet the SMBus Specification requirements of 250 ns and 300 ns, respectively. Table 28.2 shows the minimum setup and hold times for the two EXTHOLD settings. Setup and hold time extensions are typically necessary when SYSCLK is above 10 MHz.

Table 28.6. SMBus Status Decoding With Hardware ACK Generation Enabled (EHACK = 1)

Mode	Values Read				Current SMbus State	Typical Response Options	Values to Write			Next Status Vector Expected
	Status Vector	ACKRQ	ARBLOST	ACK			STA	STO	ACK	
Master Transmitter	1110	0	0	X	A master START was generated.	Load slave address + R/W into SMB0DAT.	0	0	X	1100
	1100	0	0	0	A master data or address byte was transmitted; NACK received.	Set STA to restart transfer.	1	0	X	1110
						Abort transfer.	0	1	X	-
		0	0	1	A master data or address byte was transmitted; ACK received.	Load next data byte into SMB0DAT.	0	0	X	1100
						End transfer with STOP.	0	1	X	-
						End transfer with STOP and start another transfer.	1	1	X	-
						Send repeated START.	1	0	X	1110
						Switch to Master Receiver Mode (clear SI without writing new data to SMB0DAT). Set ACK for initial data byte.	0	0	1	1000
Master Receiver	1000	0	0	1	A master data byte was received; ACK sent.	Set ACK for next data byte; Read SMB0DAT.	0	0	1	1000
						Set NACK to indicate next data byte as the last data byte; Read SMB0DAT.	0	0	0	1000
						Initiate repeated START.	1	0	0	1110
						Switch to Master Transmitter Mode (write to SMB0DAT before clearing SI).	0	0	X	1100
		0	0	0	A master data byte was received; NACK sent (last byte).	Read SMB0DAT; send STOP.	0	1	0	-
						Read SMB0DAT; Send STOP followed by START.	1	1	0	1110
						Initiate repeated START.	1	0	0	1110
						Switch to Master Transmitter Mode (write to SMB0DAT before clearing SI).	0	0	X	1100

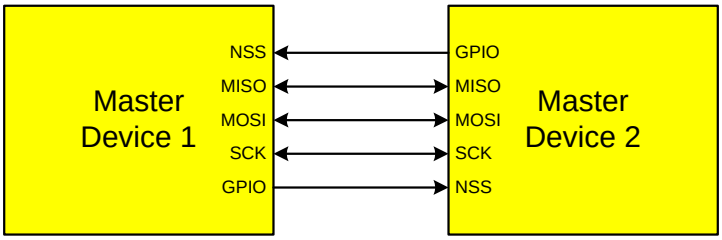

Figure 30.2. Multiple-Master Mode Connection Diagram

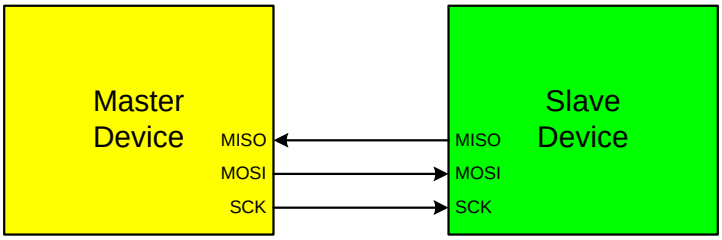

Figure 30.3. 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram

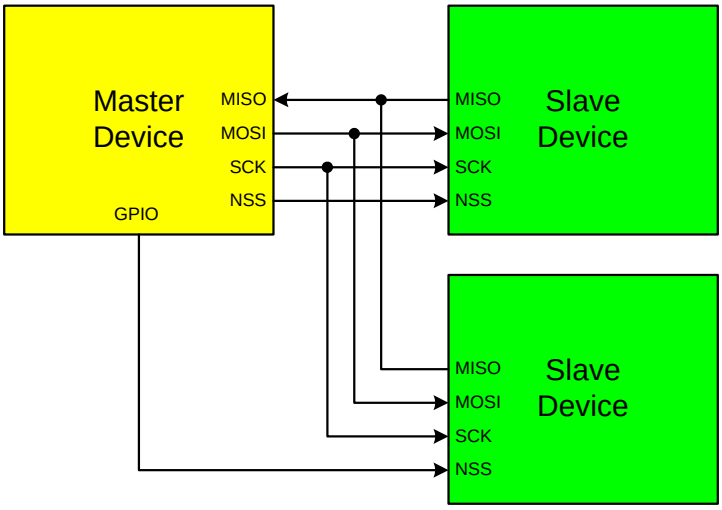

Figure 30.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram

30.3. 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,

Table 30.1. SPI Slave Timing Parameters

Parameter	Description	Min	Max	Units
Master Mode Timing (See Figure 30.8 and Figure 30.9)				
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 30.10 and Figure 30.11)				
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).				

To initiate a Master mode Bidirectional data transfer:

1. Configure the SPI1 SFRs normally for Master mode.
   a. Enable Master mode by setting bit 6 in SPI1CFG.
   b. Configure the clock polarity CKPOL and clock phase CKPHA as desired in SPI1CFG.
   c. Configure SPI1CKR for the desired SPI clock rate.
   d. Configure the desired 4-wire master or 3-wire master mode in SPI1CN.
   e. Enable the SPI by setting bit 0 of SPI1CN.
2. Configure the first DMA channel for the XRAM-to-SPI1DATA transfer:
   a. Disable the first DMA channel by clearing the corresponding bit in DMA0EN.
   b. Select the first DMA channel by writing to DMA0SEL.
   c. Configure the selected DMA channel to use the XRAM-to-SPI1DAT peripheral request by writing 0x03 to DMA0NCF.
   d. Write 0 to DMA0NMD to disable wrapping.
   e. Write the address of the first byte of master output (MOSI) data to DMA0NBAH:L.
   f. Write the size of the SPI transfer in bytes to DMA0NSZH:L.
   g. Clear the address offset SFRs CMA0A0H:L.
3. Configure the second DMA channel for the SPI1DAT-to-XRAM transfer:
   a. Disable the second DMA channel by clearing the corresponding bit in DMA0EN.
   b. Select the second DMA channel by writing to DMA0SEL.
   c. Configure the selected DMA channel to use the SPI1DAT-to-XRAM peripheral request by writing 0x04 to DMA0NCF.
   d. Enable DMA interrupts for the second channel by setting bit 7 of DMA0NCF.
   e. Write 0 to DMA0NMD to disable wrapping.
   f. Write the address for the first byte of master input (MISO) data to DMA0NBAH:L.
   g. Write the size of the SPI transfer in bytes to DMA0NSZH:L.
   h. Clear the address offset SFRs CMA0A0H:L.
   i. Enable the interrupt on the second channel by setting the corresponding bit in DMA0INT.
   j. Enable DMA interrupts by setting bit 5 of EIE2.
4. Clear the interrupt bits in DMA0INT for both channels.
5. Enable both channels by setting the corresponding bits in the DMA0EN SFR to initiate the SPI transfer operation.
6. Wait on the DMA interrupt.
7. Clear the DMA enables in the DMA0EN SFR.
8. Clear the DMA interrupts in the DMA0INT SFR.

SFR Definition 32.14. TMR3RLL: Timer 3 Reload Register Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR3RLL[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0x92

Bit	Name	Function
7:0	TMR3RLL[7:0]	**Timer 3 Reload Register Low Byte.** TMR3RLL holds the low byte of the reload value for Timer 3.

SFR Definition 32.15. TMR3RLH: Timer 3 Reload Register High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR3RLH[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0x93

Bit	Name	Function
7:0	TMR3RLH[7:0]	**Timer 3 Reload Register High Byte.** TMR3RLH holds the high byte of the reload value for Timer 3.