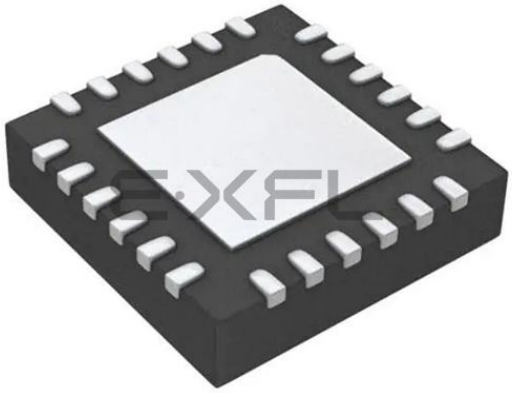

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	8051
Core Size	8-Bit
Speed	50MHz
Connectivity	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	21
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K 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	24-WFQFN Exposed Pad
Supplier Device Package	24-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f375-a-gmr

SFR Definition 8.2. TS0DATH: Temperature Sensor Output High Byte

Bit	7	6	5	4	3	2	1	0
Name	TS0DATH							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xD3; SFR Page = 0

Bit	Name	Function
7:0	TS0DATH	**Temperature Sensor Data Word (MSB).** This byte represents the MSB of the temperature sensor data word. The data word is a 16-bit, 2's complement number.

SFR Definition 8.3. TS0DATL: Temperature Sensor Output Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TS0DATL							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xD2; SFR Page = 0

Bit	Name	Function
7:0	TS0DATL	**Temperature Sensor Data Word (LSB).** This byte represents the LSB of the temperature sensor data word. The data word is a 16-bit, 2's complement number.

9.3. Programmable Window Detector

The ADC Programmable Window Detector continuously compares the ADC0 output registers to user-programmed limits, and notifies the system when a desired condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (AD0WINT in register ADC0CN) can also be used in polled mode. The ADC0 Greater-Than (ADC0GTH, ADC0GTL) and Less-Than (ADC0LTH, ADC0LTL) registers hold the comparison values. The window detector flag can be programmed to indicate when measured data is inside or outside of the user-programmed limits, depending on the contents of the ADC0 Less-Than and ADC0 Greater-Than registers.

SFR Definition 9.5. ADC0GTH: ADC0 Greater Than Data High Byte

Bit	7	6	5	4	3	2	1	0
Name	ADC0GTH[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xC4; SFR Page = All Pages

Bit	Name	Function
7:0	ADC0GTH[7:0]	**ADC0 Greater-Than Data Word High-Order Bits.**

SFR Definition 9.6. ADC0GTL: ADC0 Greater-Than Data Low Byte

Bit	7	6	5	4	3	2	1	0
Name	ADC0GTL[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xC3; SFR Page = All Pages

Bit	Name	Function
7:0	ADC0GTL[7:0]	**ADC0 Greater-Than Data Word Low-Order Bits.**

10. Temperature Sensor (C8051F390/2/4/6/8 and C8051F370/4 Only)

A fully C8051F33x-compatible temperature sensor is included on the C8051F390/2/4/6/8 and C8051F370/4 and accessed via the ADC multiplexer in single-ended configuration. For the self-contained precision temperature sensor, refer to Section 8.

To use the ADC to measure the temperature sensor, the ADC mux channel should be configured to connect to the temperature sensor. The temperature sensor transfer function is shown in Figure 10.1. The output voltage (V_{TEMP}) is the positive ADC input when the ADC multiplexer is set correctly. The TEMPE bit in register REF0CN enables/disables the temperature sensor, as described in SFR Definition 12.1. While disabled, the temperature sensor defaults to a high impedance state and any ADC measurements performed on the sensor will result in meaningless data. Refer to Table 7.11 for the slope and offset parameters of the temperature sensor.

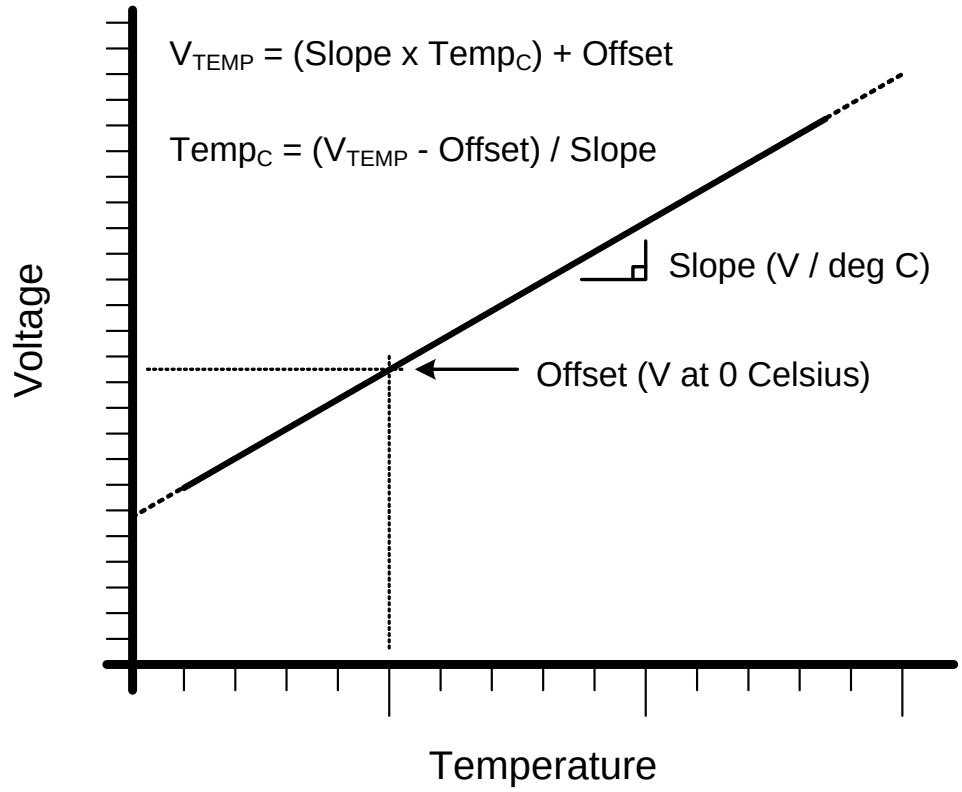

Figure 10.1. Temperature Sensor Transfer Function

11. 10-Bit Current Mode DACs (IDA0, IDA1, C8051F390/2/4/6/8 and C8051F370/4 Only)

The C8051F390/2/4/6/8 and C8051F370/4 devices include two 10-bit current-mode Digital-to-Analog Converters (IDACs). The maximum current output of the IDACs can be adjusted for three different current settings; 0.5 mA, 1 mA, and 2 mA. The IDACs are enabled or disabled with the IDAnEN bit in the Control Register for that IDAC (see SFR Definition 11.1 and SFR Definition 11.4). When IDAnEN is set to 0, the IDAC output behaves as a normal GPIO pin. When IDAnEN is set to 1, the digital output drivers and weak pullup for the IDAC pin are automatically disabled, and the pin is connected to the IDAC output. An internal bandgap bias generator is used to generate a reference current for the IDAC whenever it is enabled. When using an IDAC, the crossbar skip functionality should be enabled on the IDAC output pin, to force the Crossbar to skip the output pin.

11.1. IDAC Output Scheduling

The IDACs feature a flexible output update mechanism which allows for seamless full-scale changes and supports jitter-free updates for waveform generation. Three update modes are provided, allowing IDAC output updates on a write to IDAnH, on a Timer overflow, or on an external pin edge.

11.1.1. Update Output On-Demand

In its default mode (IDAnCN.[6:4] = 111) the IDAC output is updated “on-demand” on a write to the high-byte of the IDAC data register (IDAnH). It is important to note that writes to IDAnL are held in this mode, and have no effect on the IDAC output until a write to IDAnH takes place. If writing a full 10-bit word to the IDAC data registers, the 10-bit data word is written to the low byte (IDAnL) and high byte (IDAnH) data registers. Data is latched into the IDAC after a write to the IDAnH register, **so the write sequence should be IDAnL followed by IDAnH** if the full 10-bit resolution is required. The IDAC can be used in 8-bit mode by initializing IDAnL to the desired value (typically 0x00), and writing data to only IDAnH (see Section 11.3 for information on the format of the 10-bit IDAC data word within the 16-bit SFR space).

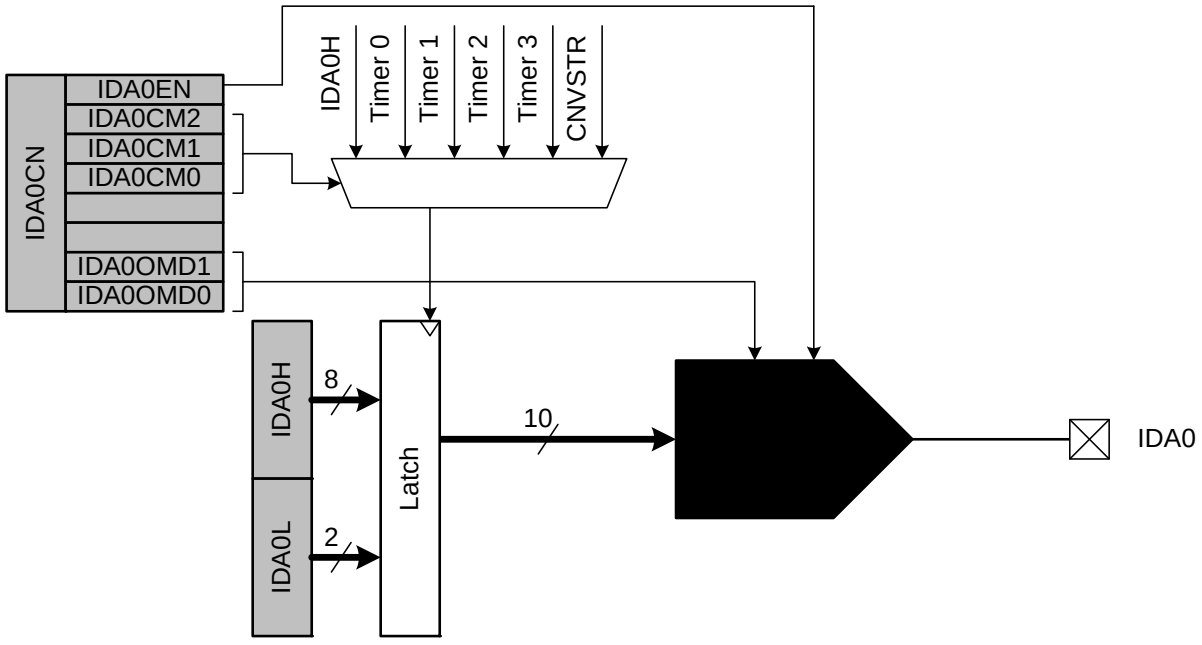

Figure 11.1. IDA0 Functional Block Diagram

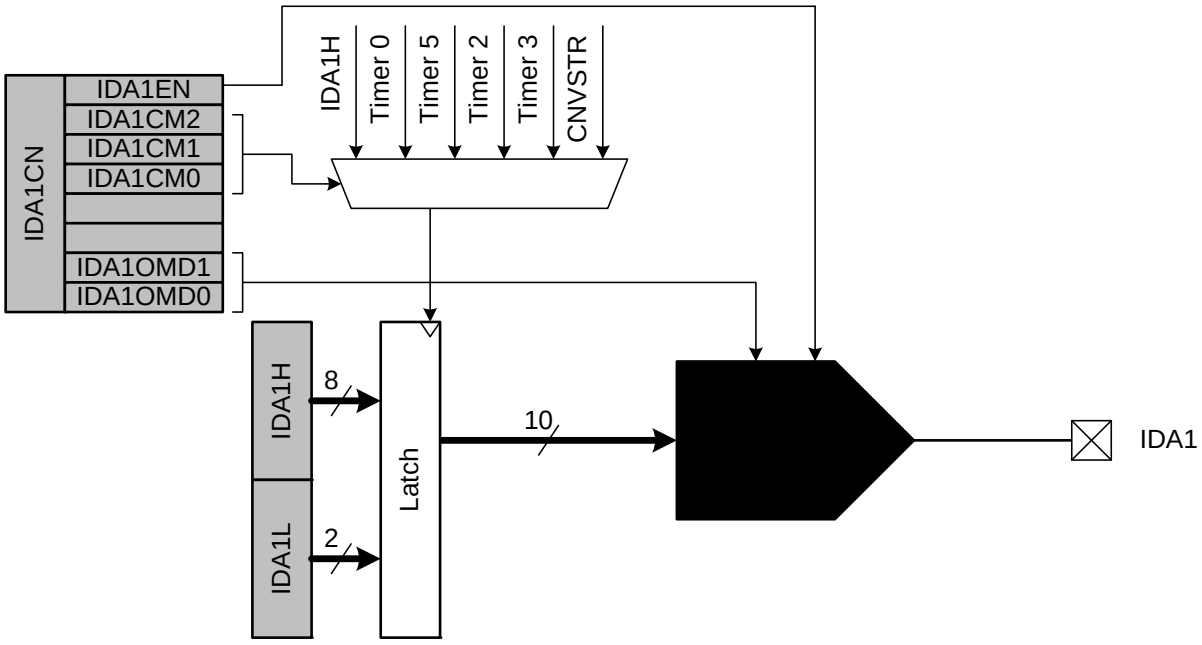

Figure 11.2. IDA1 Functional Block Diagram

18. Device ID Registers

The C8051F39x/37x has SFRs that identify the device family and derivative. These SFRs can be read by firmware at runtime to determine the capabilities of the MCU that is executing code. This allows the same firmware image to run on MCUs with different memory sizes and peripherals, and dynamically changing functionality to suit the capabilities of that MCU.

In order for firmware to identify the MCU, it must read two SFRs. DERIVID describes the specific derivative within that device family, and REVID describes the hardware revision of the MCU.

The C8051F39x/37x devices also include four SFRs, SN0 through SN3, that are pre-programmed during production with a unique, 32-bit serial number. The serial number provides a unique identification number for each device and can be read from the application firmware. If the serial number is not used in the application, these four registers can be used as general purpose SFRs.

SFR Definition 18.1. DERIVID: Device Derivative ID

Bit	7	6	5	4	3	2	1	0
Name	DERIVID							
Type	R							
Reset	Varies	Varies	Varies	Varies	Varies	Varies	Varies	Varies

SFR Address = 0xAB; SFR Page = 0

Bit	Name	Function
7:0	DERIVID	**Derivative ID.** This read-only register returns the 8-bit derivative ID, which can be used by firmware to identify which device in the product family is being used. 0xD0: C8051F390 0xD1: C8051F391 0xD2: C8051F392 0xD3: C8051F393 0xD4: C8051F394 0xD5: C8051F395 0xD6: C8051F396 0xD7: C8051F397 0xD8: C8051F398 0xD9: C8051F399 0xE0: C8051F370 0xE1: C8051F371 0xE4: C8051F374 0xE5: C8051F375

SFR Definition 19.1. SFRPAGE: SFR Page

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

SFR Address = 0xA7; SFR Page = All Pages

Bit	Name	Function
7:0	SFRPAGE[7:0]	**SFR Page Bits.** Represents the SFR Page the C8051 core uses when reading or modifying SFRs. Write: Sets the SFR Page. Read: Byte is the SFR page the C8051 core is using.

21.1.3. Flash Write Procedure

Flash bytes are programmed by software with the following sequence:

1. Disable interrupts (recommended).
2. Erase the 512-byte Flash page containing the target location, as described in Section 21.1.2.
3. Set the PSWE bit (register PSCTL).
4. Clear the PSEE bit (register PSCTL).
5. Write the first key code to FLKEY: 0xA5.
6. Write the second key code to FLKEY: 0xF1.
7. Using the MOVX instruction, write a single data byte to the desired location within the 512-byte sector.
8. Clear the PSWE bit.

Steps 5–7 must be repeated for each byte to be written. After Flash writes are complete, PSWE should be cleared so that MOVX instructions do not target program memory.

21.2. Non-volatile Data Storage

The Flash memory can be used for non-volatile data storage as well as program code. This allows data such as calibration coefficients to be calculated and stored at run time. Data is written using the MOVX write instruction and read using the MOVC instruction. Note: MOVX read instructions always target XRAM.

22.2. Write Operation

Up to sixteen successive bytes may be written to the EEPROM within a single write operation. To write to the EEPROM:

1. The master sends the START condition and the slave address byte with the R/W bit cleared to 0.
2. The EEPROM generates an ACK.
3. The master sends the write address location (A[7:0]) to the EEPROM.
4. The EEPROM stores the address location in its address counter and generates an ACK.
5. The master transmits the data byte (D[7:0]) to the EEPROM.
6. The EEPROM increments four least significant bits of the address counter and generates an ACK.
7. The master can repeat Steps 5 and 6 up to fifteen more times.
8. The master generates a STOP condition.
9. The EEPROM begins its internal programming cycle.
10. The master transmits a START condition and slave address with the R/W bit cleared to 0:
    a. If the EEPROM does not generate an ACK, repeat Steps 8 and 9.
    b. If the EEPROM does generate and ACK, the EEPROM internal programming cycle is complete.

Note: If the master transmits more than sixteen bytes prior to issuing a STOP condition, the last for bits of the address counter will roll over and the previously written data will be overwritten.

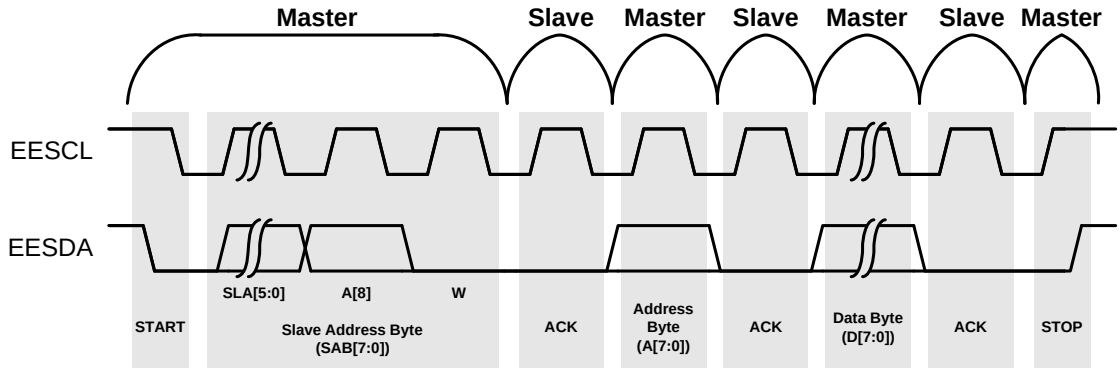

Figure 22.2. Write Operation (Single Byte)

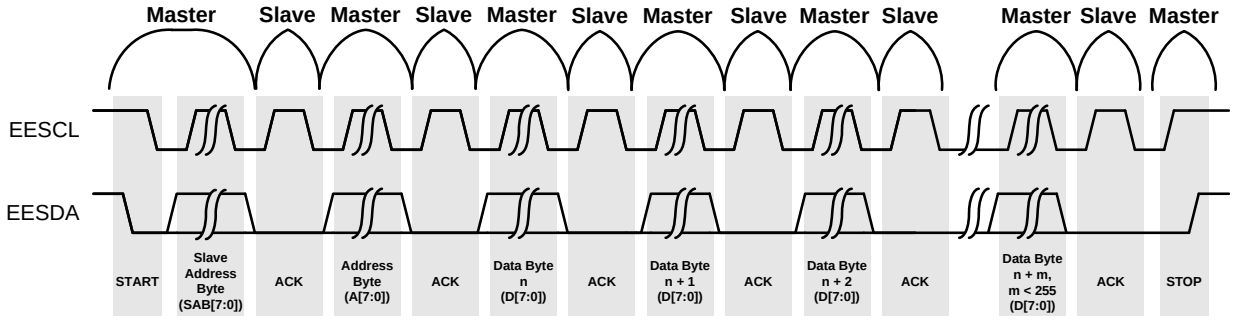

Figure 22.3. Write Operation (Multiple Bytes)

25.2. 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. Before entering stop mode, the system clock must be sourced by the internal high-frequency oscillator. 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.

By default, when in stop mode the internal regulator is still active. However, the regulator can be configured to shut down while in stop mode to save power. To shut down the regulator in stop mode, the STOPCF bit in register REG01CN should be set to 1 prior to setting the STOP bit (see SFR Definition 25.1). If the regulator is shut down using the STOPCF bit, only the $\overline{\text{RST}}$ pin or a full power cycle are capable of resetting the device.

25.3. Suspend Mode

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 or the internal low-frequency oscillator.

Suspend mode can be terminated by four types of events, a port match (described in Section “27.5. Port Match” on page 183), a Timer 3 overflow (described in Section “31.3. Timer 3” on page 259), a Comparator low output (if enabled), or a device reset event. Note that in order to run Timer 3 in suspend mode, the timer must be configured to clock from either the external clock source or the internal low-frequency oscillator 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 3 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.

27. Port Input/Output

Digital and analog resources are available through 17 (C8051F392/3/6/7/8/9) or 21 (C8051F390/1/4/5 and C8051F37x) I/O pins. Port pins P0.0-P2.3 can be defined as general-purpose I/O (GPIO), assigned to one of the internal digital resources, or assigned to an analog function as shown in Figure 27.3. Port pin P2.4 on the C8051F390/1/4/5 and C8051F37x and P2.0 on the C8051F392/3/6/7/8/9 can be used as GPIO and are shared with the C2 Interface Data signal (C2D). The designer has complete control over which functions are assigned, limited only by the number of physical I/O pins. This resource assignment flexibility is achieved through the use of a Priority Crossbar Decoder. Note that the state of a Port I/O pin can always be read in the corresponding Port latch, regardless of the Crossbar settings.

The Crossbar assigns the selected internal digital resources to the I/O pins based on the Priority Decoder (Figure 27.3 and Figure 27.4). The registers XBR0 and XBR1, defined in SFR Definition 27.1 and SFR Definition 27.2, are used to select internal digital functions.

The Port I/O cells are configured as either push-pull or open-drain in the Port Output Mode registers (PnMDOUT, where n = 0,1). Complete Electrical Specifications for Port I/O are given in Section "7. Electrical Characteristics" on page 32.

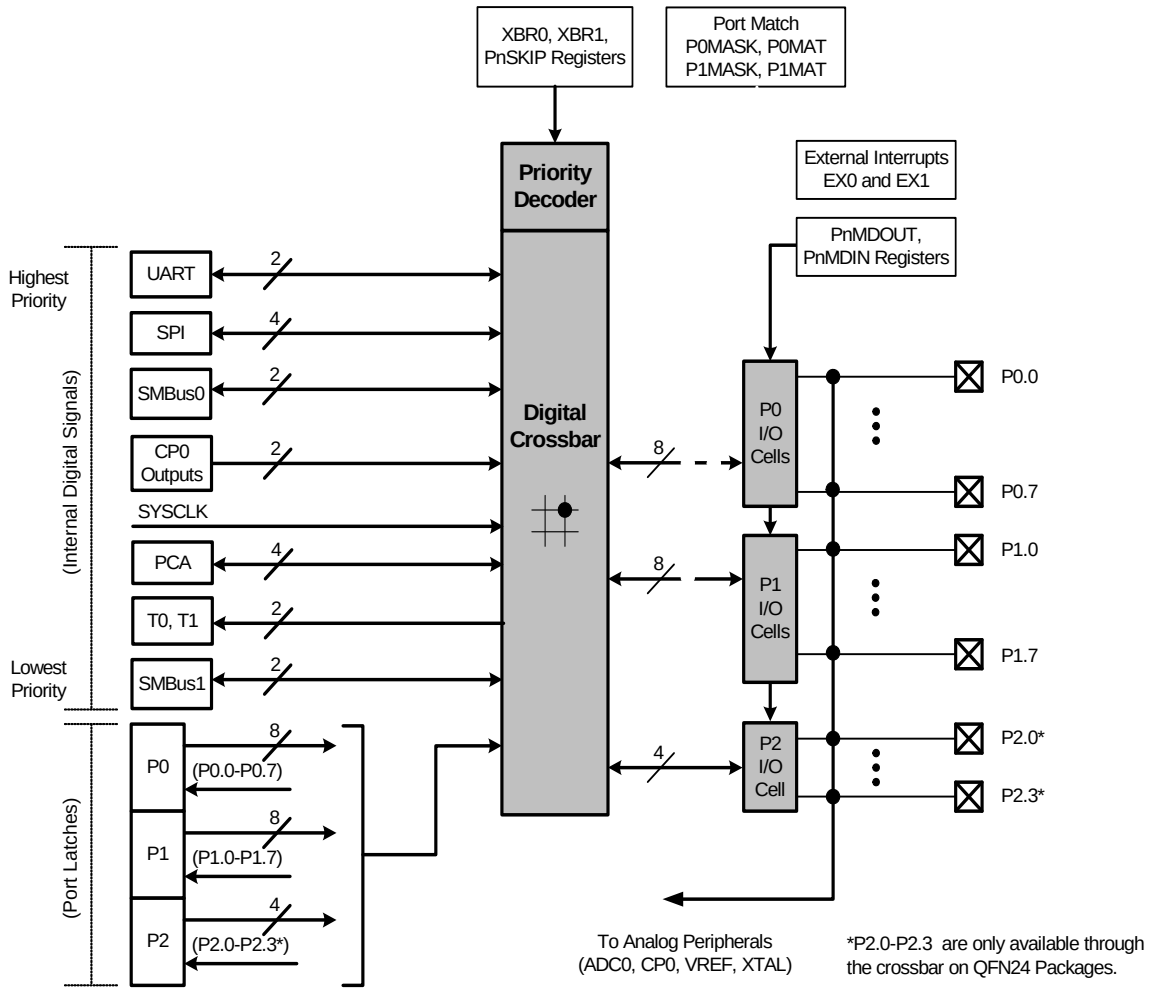

Figure 27.1. Port I/O Functional Block Diagram

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, SMBus0, SMBus1, CP0, CP0A, SYSCLK, PCA0 (CEX0-2 and ECI), T0 or T1.	Any Port pin available for assignment by the Crossbar. This includes P0.0 - P2.3 pins which have their PnSKIP bit set to '0'. **Note:** The Crossbar will always assign UART0 pins to P0.4 and P0.5.	XBR0, XBR1
Any pin used for GPIO	P0.0 - P2.4	P0SKIP, P1SKIP, P2SKIP

imum setup and hold times for the two EXTHOLD settings. Setup and hold time extensions are typically necessary when SYSCLK is above 10 MHz.

Table 28.2. Minimum SDA Setup and Hold Times

EXTHOLD	Minimum SDA Setup Time	Minimum SDA Hold Time
0	T_{low} – 4 system clocks or 1 system clock + s/w delay*	3 system clocks
1	11 system clocks	12 system clocks
Note: Setup Time for ACK bit transmissions and the MSB of all data transfers. When using software acknowledgement, the s/w delay occurs between the time SMB0DAT or ACK is written and when SI is cleared. Note that if SI is cleared in the same write that defines the outgoing ACK value, s/w delay is zero.		

With the SMBnTOE bit set, Timer 3 (SMBus0) and Timer 5 (SMBus1) should be configured to overflow after 25 ms in order to detect SCL low timeouts (see Section “28.3.4. SCL Low Timeout” on page 194). The SMBus interface will force the associated timer to reload while SCL is high, and allow the timer to count when SCL is low. The timer interrupt service routine should be used to reset SMBus communication by disabling and re-enabling the SMBus.

SMBus Free Timeout detection can be enabled by setting the SMBnFTE bit. When this bit is set, the bus will be considered free if SDA and SCL remain high for more than 10 SMBus clock source periods (see Figure 28.4).

28.4.2. SMBus Pin Swap

The SMBus peripherals are assigned to pins using the priority crossbar decoder. By default, the SMBus signals are assigned to port pins starting with SDA on the lower-numbered pin, and SCL on the next available pin. The SMBnSWAP bits in the SMBTC register can be set to 1 to reverse the order in which the SMBus signals are assigned.

28.4.3. SMBus Timing Control

The SMBnSDD field in the SMBTC register are used to restrict the detection of a START condition under certain circumstances. In some systems where there is significant mis-match between the impedance or the capacitance on the SDA and SCL lines, it may be possible for SCL to fall after SDA during an address or data transfer. Such an event can cause a false START detection on the bus. These kind of events are not expected in a standard SMBus or I2C-compliant system. **In most systems this parameter should not be adjusted, and it is recommended that it be left at its default value.**

By default, if the SCL falling edge is detected after the falling edge of SDA (i.e. one SYSCLK cycle or more), the device will detect this as a START condition. The SMBnSDD field is used to increase the amount of hold time that is required between SDA and SCL falling before a START is recognized. An additional 2, 4, or 8 SYSCLKs can be added to prevent false START detection in systems where the bus conditions warrant this.

SFR Definition 28.8. SMB1ADR: SMBus1 Slave Address

Bit	7	6	5	4	3	2	1	0
Name	SLV1[6:0]							GC1
Type	R/W							R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xD7; SFR Page = F

Bit	Name	Function
7:1	SLV1[6:0]	**SMBus1 Hardware Slave Address.** Defines the SMBus1 Slave Address(es) for automatic hardware acknowledgment. Only address bits which have a 1 in the corresponding bit position in SLVM1[6:0] are checked against the incoming address. This allows multiple addresses to be recognized.
0	GC1	**General Call Address Enable.** When hardware address recognition is enabled (EHACK1 = 1), this bit will determine whether the General Call Address (0x00) is also recognized by hardware. 0: General Call Address is ignored. 1: General Call Address is recognized.

Table 28.6. SMBus Status Decoding: Hardware ACK 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	
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
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	—

31.1.4. Mode 3: Two 8-bit Counter/Timers (Timer 0 Only)

In Mode 3, Timer 0 is configured as two separate 8-bit counter/timers held in TL0 and TH0. The counter/timer in TL0 is controlled using the Timer 0 control/status bits in TCON and TMOD: TR0, C/T0, GATE0 and TF0. TL0 can use either the system clock or an external input signal as its timebase. The TH0 register is restricted to a timer function sourced by the system clock or prescaled clock. TH0 is enabled using the Timer 1 run control bit TR1. TH0 sets the Timer 1 overflow flag TF1 on overflow and thus controls the Timer 1 interrupt.

Timer 1 is inactive in Mode 3. When Timer 0 is operating in Mode 3, Timer 1 can be operated in Modes 0, 1 or 2, but cannot be clocked by external signals nor set the TF1 flag and generate an interrupt. However, the Timer 1 overflow can be used to generate baud rates for the SMBus and/or UART, and/or initiate ADC conversions. While Timer 0 is operating in Mode 3, Timer 1 run control is handled through its mode settings. To run Timer 1 while Timer 0 is in Mode 3, set the Timer 1 Mode as 0, 1, or 2. To disable Timer 1, configure it for Mode 3.

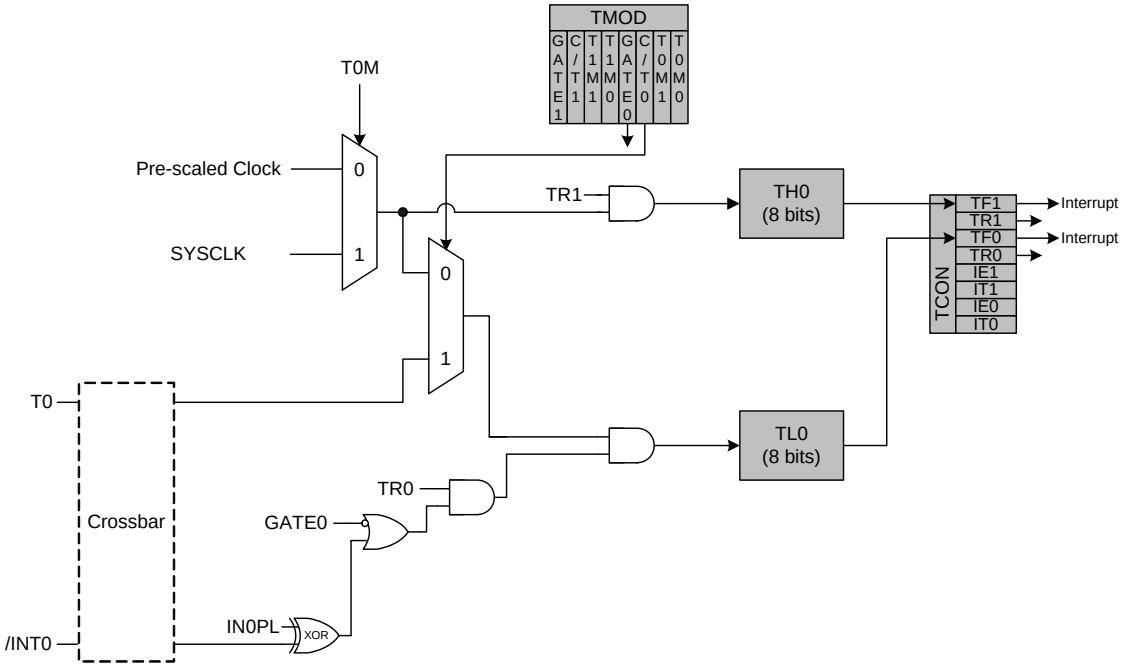

Figure 31.3. T0 Mode 3 Block Diagram

32.3.4. Frequency Output Mode

Frequency Output Mode produces a programmable-frequency square wave on the module's associated CEXn pin. The capture/compare module high byte holds the number of PCA clocks to count before the output is toggled. The frequency of the square wave is then defined by Equation 32.1.

$$F_{CEXn} = \frac{F_{PCA}}{2 \times PCA0CPHn}$$

Note: A value of 0x00 in the PCA0CPHn register is equal to 256 for this equation.

Equation 32.1. Square Wave Frequency Output

Where F_{PCA} is the frequency of the clock selected by the CPS2–0 bits in the PCA mode register, PCA0MD. The lower byte of the capture/compare module is compared to the PCA counter low byte; on a match, n is toggled and the offset held in the high byte is added to the matched value in PCA0CPLn. Frequency Output Mode is enabled by setting the ECOMn, TOGn, and PWMn bits in the PCA0CPMn register. Note that the MATn bit should normally be set to 0 in this mode. If the MATn bit is set to 1, the CCFn flag for the channel will be set when the 16-bit PCA0 counter and the 16-bit capture/compare register for the channel are equal.

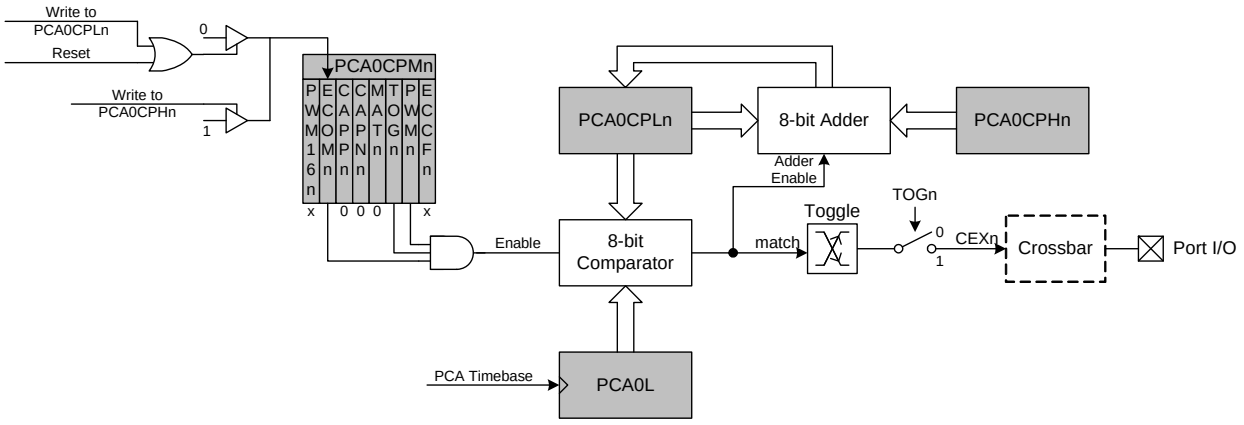

Figure 32.7. PCA Frequency Output Mode

32.3.5. 8-bit, 9-bit, 10-bit and 11-bit Pulse Width Modulator Modes

Each module can be used independently to generate a pulse width modulated (PWM) output on its associated CEXn pin. The frequency of the output is dependent on the timebase for the PCA counter/timer, and the setting of the PWM cycle length (8, 9, 10 or 11-bits). For backwards-compatibility with the 8-bit PWM mode available on other devices, the 8-bit PWM mode operates slightly different than 9, 10 and 11-bit PWM modes. **It is important to note that all channels configured for 8/9/10/11-bit PWM mode will use the same cycle length.** It is not possible to configure one channel for 8-bit PWM mode and another for 11-bit mode (for example). However, other PCA channels can be configured to Pin Capture, High-Speed Output, Software Timer, Frequency Output, or 16-bit PWM mode independently.

SFR Definition 32.3. PCA0PWM: PCA PWM Configuration

Bit	7	6	5	4	3	2	1	0
Name	ARSEL	ECOV	COVF				CLSEL[1:0]	
Type	R/W	R/W	R/W	R	R	R	R/W	
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xF7; SFR Page = All Pages

Bit	Name	Function
7	ARSEL	**Auto-Reload Register Select.** This bit selects whether to read and write the normal PCA capture/compare registers (PCA0CPn), or the Auto-Reload registers at the same SFR addresses. This function is used to define the reload value for 9, 10, and 11-bit PWM modes. In all other modes, the Auto-Reload registers have no function. 0: Read/Write Capture/Compare Registers at PCA0CPHn and PCA0CPLn. 1: Read/Write Auto-Reload Registers at PCA0CPHn and PCA0CPLn.
6	ECOV	**Cycle Overflow Interrupt Enable.** This bit sets the masking of the Cycle Overflow Flag (COVF) interrupt. 0: COVF will not generate PCA interrupts. 1: A PCA interrupt will be generated when COVF is set.
5	COVF	**Cycle Overflow Flag.** This bit indicates an overflow of the 8th, 9th, 10th, or 11th bit of the main PCA counter (PCA0). The specific bit used for this flag depends on the setting of the Cycle Length Select bits. The bit can be set by hardware or software, but must be cleared by software. 0: No overflow has occurred since the last time this bit was cleared. 1: An overflow has occurred since the last time this bit was cleared.
4:2	Unused	Unused. Read = 000b; Write = Don't care.
1:0	CLSEL[1:0]	**Cycle Length Select.** When 16-bit PWM mode is not selected, these bits select the length of the PWM cycle, between 8, 9, 10, or 11 bits. This affects all channels configured for PWM which are not using 16-bit PWM mode. These bits are ignored for individual channels configured to16-bit PWM mode. 00: 8 bits. 01: 9 bits. 10: 10 bits. 11: 11 bits.

C2 Register Definition 33.4. FPCTL: C2 Flash Programming Control

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

C2 Address: 0x02

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

C2 Register Definition 33.5. FPDAT: C2 Flash Programming Data

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

C2 Address: 0xB4

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