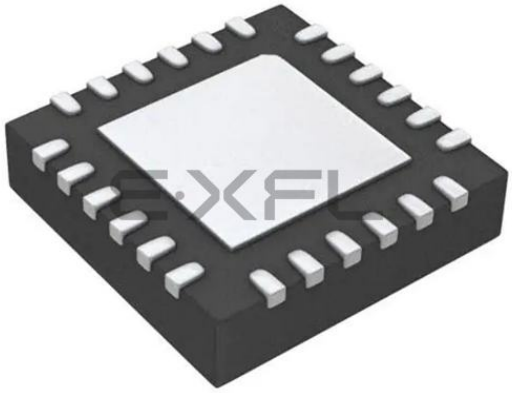

Welcome to E-XFL.COM

## 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 | 50MHz |
| Connectivity | SMBus (2-Wire/I²C), SPI, UART/USART |
| Peripherals | POR, PWM, WDT |
| Number of I/O | 21 |
| Program Memory Size | 16KB (16K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 3.6V |
| Data Converters | - |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°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/c8051f391-a-gmr |

## List of Tables

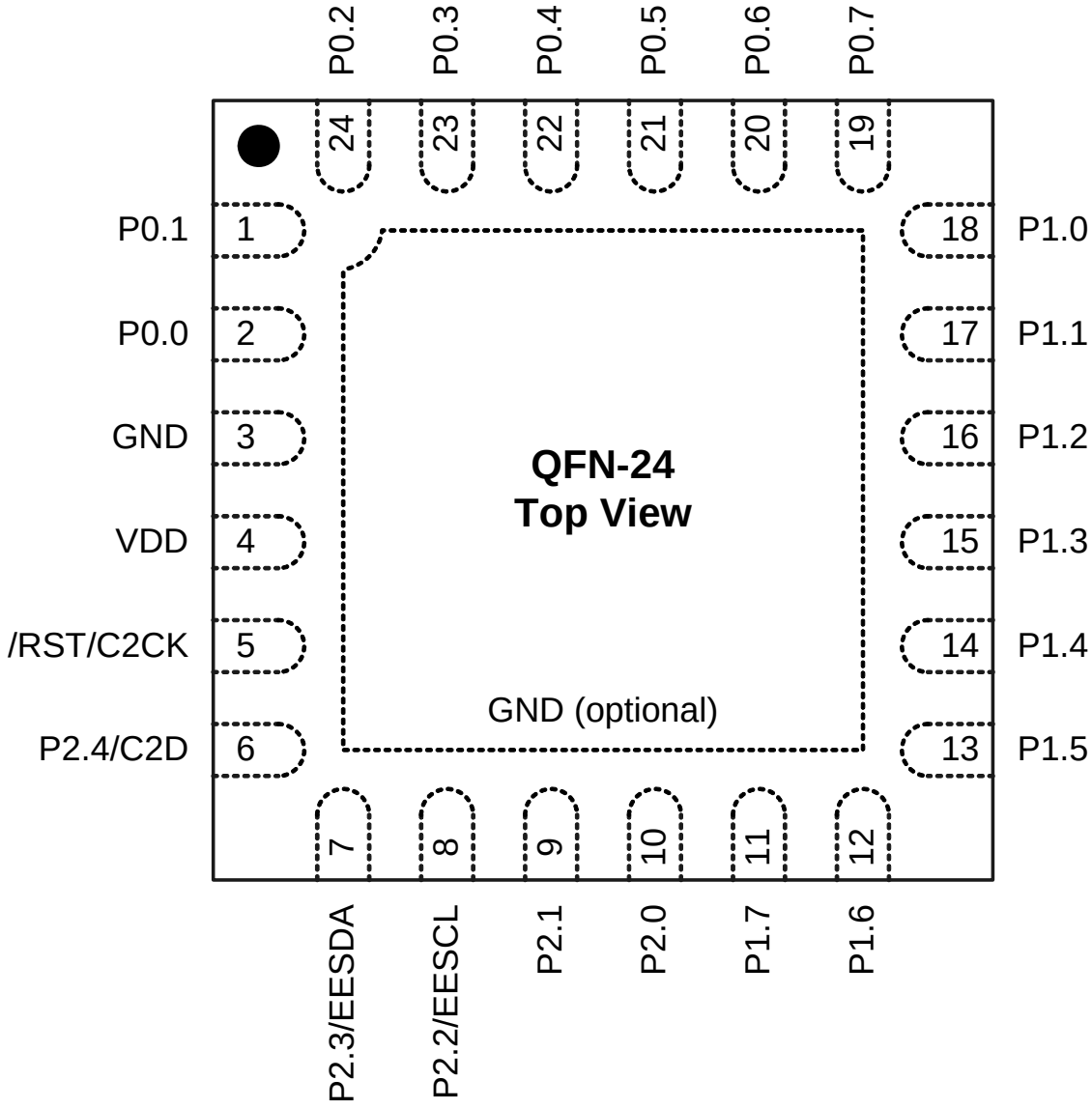


**Figure 4.3. C8051F370/1/4/5 Pinout Diagram (Top View)**

**Table 7.3. Port I/O DC Electrical Characteristics**

$V_{DD}$ = 1.8 to 3.6 V, –40 to +105 °C (C8051F39x), –40 to +85 °C (C8051F37x), unless otherwise specified.

| Parameters | Test Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| **Standard Port I/O** | | | | | |
| Output High Voltage | $I_{OH}$ = –3 mA, Port I/O push-pull | $V_{DD}$ – 0.7 | — | — | V |
| | $I_{OH}$ = –10 µA, Port I/O push-pull | $V_{DD}$ – 0.1 | — | — | V |
| | $I_{OH}$ = –10 mA, Port I/O push-pull | — | $V_{DD}$ – 0.8 | — | V |
| Output Low Voltage | $I_{OL}$ = 8.5 mA | — | — | 0.6 | V |
| | $I_{OL}$ = 10 µA | — | — | 0.1 | V |
| | $I_{OL}$ = 10 mA, 1.8 V ≤ $V_{DD}$ < 2.7 V | — | 0.8 | — | V |
| | $I_{OL}$ = 25 mA, 2.7 V ≤ $V_{DD}$ ≤ 3.6 V | — | 1.0 | — | V |
| Input High Voltage | 1.8 V ≤ $V_{DD}$ < 2.7 V | $V_{DD}$ – 0.4 | — | — | V |
| | 2.7 V ≤ $V_{DD}$ ≤ 3.6 V | $V_{DD}$ – 0.5 | — | — | V |
| Input Low Voltage | 1.8 V ≤ $V_{DD}$ < 2.7 V | — | — | 0.5 | V |
| | 2.7 V ≤ $V_{DD}$ ≤ 3.6 V | — | — | 0.6 | V |
| Input Leakage Current | Weak Pullup Off | — | — | ±1 | µA |
| | Weak Pullup On, $V_{IN}$ = 0 V | — | 20 | 100 | µA |
| **EESDA and EESCL (C8051F37x Only)*** | | | | | |
| Output Low Voltage (EESDA) | $I_{OL}$ = 0.15 mA, $V_{DD}$ = 1.8 V | — | — | 0.2 | V |
| Output Low Voltage (EESDA) | $I_{OL}$ = 2.1 mA, $V_{DD}$ = 3 V | — | — | 0.4 | V |
| Output Leakage Current (EESDA) | EEPUE = 0, $V_{DD}$ = 3.6 V, 0 V ≤ $V_{OUT}$ ≤ $V_{DD}$ | — | — | 2 | µA |
| Input High Voltage | | $V_{DD}$ x 0.7 | — | — | V |
| Input Low Voltage | | — | — | $V_{DD}$ x 0.3 | V |
| Input Leakage Current | EEPUE = 0, Standby, $V_{DD}$ = 3.6 V, 0 V ≤ $V_{IN}$ ≤ $V_{DD}$ | — | — | ±3 | µA |
| **Note:** Applicable when interfacing to the C8051F37x EEPROM. Otherwise, standard port I/O characteristics apply. | | | | | |

## SFR Definition 9.7. ADC0LTH: ADC0 Less-Than Data High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0LTH[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC6; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0LTH[7:0] | **ADC0 Less-Than Data Word High-Order Bits.** |

## SFR Definition 9.8. ADC0LTL: ADC0 Less-Than Data Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0LTL[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xC5; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | ADC0LTL[7:0] | **ADC0 Less-Than Data Word Low-Order Bits.** |

## 17. Memory Organization

The memory organization of the CIP-51 System Controller is similar to that of a standard 8051. There are two separate memory spaces: program memory and data memory. Program and data memory share the same address space but are accessed via different instruction types. The memory organization of the C8051F39x/37x device family is shown in Figure 17.1. Not shown in Figure 17.1 is 512 bytes of byte-addressable EEPROM available on C8051F37x, accessible by SMBUS/$I^2C$ (see Section 22).

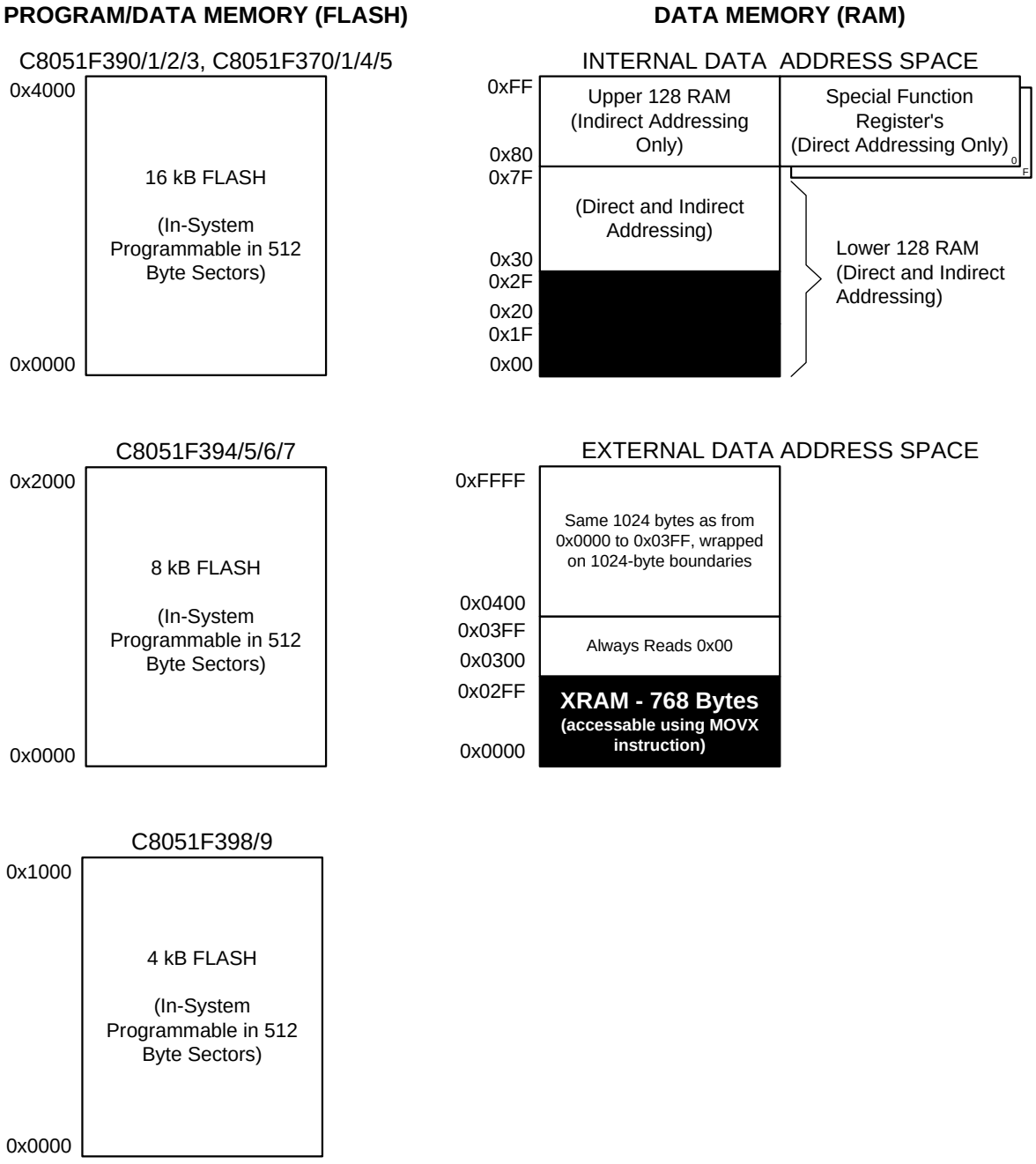


**Figure 17.1. C8051F39x/37x Memory Map**

The SPI0 interrupt occurs while the core executes in-line code by writing a value to TS0CN. The core vectors to the SPI0 ISR and pushes the current SFR page value (in this case SFR page 0x0F for TS0CN) into the 001b SFRPGIDX location in the SFR page stack. Also, the core automatically places the SFR page (0x00) needed to access the SPI0's special function registers into the SFRPAGE register. See Figure 19.3.

SFRPAGE is considered the top of the SFR page stack. Software may switch to any SFR page by writing a new value to the SFRPAGE register at any time during the SPI0 ISR.

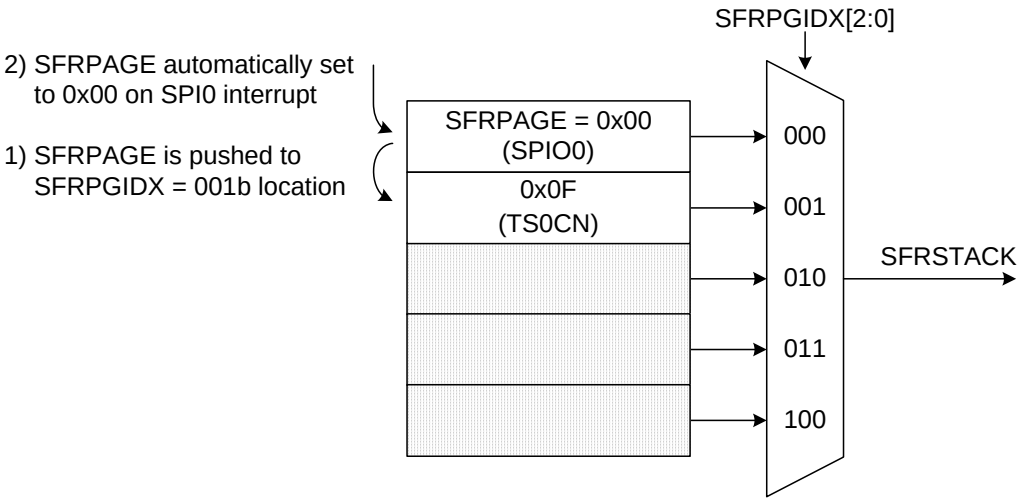


**Figure 19.3. SFR Page Stack After SPI0 Interrupt Occurs**

# 20. Interrupts

The C8051F39x/37x includes an extended interrupt system supporting multiple interrupt sources with four priority levels. The allocation of interrupt sources between on-chip peripherals and external input pins varies according to the specific version of the device. Each interrupt source has one or more associated interrupt-pending flag(s) located in an SFR. When a peripheral or external source meets a valid interrupt condition, the associated interrupt-pending flag is set to logic 1.

If interrupts are enabled for the source, an interrupt request is generated when the interrupt-pending flag is set. As soon as execution of the current instruction is complete, the CPU generates an LCALL to a predetermined address to begin execution of an interrupt service routine (ISR). Each ISR must end with an RETI instruction, which returns program execution to the next instruction that would have been executed if the interrupt request had not occurred. If interrupts are not enabled, the interrupt-pending flag is ignored by the hardware and program execution continues as normal. (The interrupt-pending flag is set to logic 1 regardless of the interrupt's enable/disable state.)

Each interrupt source can be individually enabled or disabled through the use of an associated interrupt enable bit in an SFR (IE, EIE1, and EIE2). However, interrupts must first be globally enabled by setting the EA bit (IE.7) to logic 1 before the individual interrupt enables are recognized. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings.

**Note:** Any instruction that clears a bit to disable an interrupt should be immediately followed by an instruction that has two or more opcode bytes. Using EA (global interrupt enable) as an example:

```
// in 'C':
EA = 0; // clear EA bit.
EA = 0; // this is a dummy instruction with two-byte opcode.

; in assembly:
CLR EA ; clear EA bit.
CLR EA ; this is a dummy instruction with two-byte opcode.
```

For example, if an interrupt is posted during the execution phase of a "CLR EA" opcode (or any instruction which clears a bit to disable an interrupt source), and the instruction is followed by a single-cycle instruction, the interrupt may be taken. However, a read of the enable bit will return a '0' inside the interrupt service routine. When the bit-clearing opcode is followed by a multi-cycle instruction, the interrupt will not be taken.

Some interrupt-pending flags are automatically cleared by the hardware when the CPU vectors to the ISR. However, most are not cleared by the hardware and must be cleared by software before returning from the ISR. If an interrupt-pending flag remains set after the CPU completes the return-from-interrupt (RETI) instruction, a new interrupt request will be generated immediately and the CPU will re-enter the ISR after the completion of the next instruction.

## 20.2. Interrupt Register Descriptions

The SFRs used to enable the interrupt sources and set their priority level are described in this section. Refer to the data sheet 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).

### SFR Definition 20.1. IE: Interrupt Enable

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

SFR Address = 0xA8; SFR Page = All Pages; Bit-Addressable

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

## SFR Definition 23.4. CRC0AUTO: CRC0 Automatic Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AUTOEN | | CRC0ST[5:0] | | | | | |
| Type | R/W | R/W | R/W | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xDD; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7 | AUTOEN | **Automatic CRC Calculation Enable.**<br>When AUTOEN is set to 1, any write to CRC0CN will initiate an automatic CRC starting at Flash sector CRC0ST and continuing for CRC0CNT sectors. |
| 6 | Reserved | Must write 0b. |
| 5:0 | CRC0ST[5:0] | **Automatic CRC Calculation Starting Block.**<br>These bits specify the Flash block to start the automatic CRC calculation. The starting address of the first Flash block included in the automatic CRC calculation is CRC0ST x Block Size.<br>**Note:** The block size is 256 bytes. |

## 26.2. Programmable Internal High-Frequency (H-F) Oscillator

All C8051F39x/37x devices include a programmable internal high-frequency oscillator that defaults as the system clock after a system reset. The internal oscillator period can be adjusted via the OSCICL register as defined by SFR Definition 26.2.

On C8051F39x/37x devices, OSCICL is factory calibrated to obtain a 49 MHz base frequency.

The system clock may be derived directly from the programmed internal oscillator, or from a divided version, with factors of 2, 4, 8, or 16, as defined by the IFCN bits in register OSCICN. The divide value defaults to 16 following a reset.

### 26.2.1. Internal Oscillator Suspend Mode

When software writes a logic 1 to SUSPEND (OSCICN.5), the internal oscillator is suspended. If the system clock is derived from the internal oscillator, the input clock to the peripheral or CIP-51 will be stopped until one of the following events occur:

- Port 0 Match Event.
- Port 1 Match Event.
- Comparator 0 enabled and output is logic 0.
- Timer3 Overflow Event.

When one of the oscillator awakening events occur, the internal oscillator, CIP-51, and affected peripherals resume normal operation, regardless of whether the event also causes an interrupt. The CPU resumes execution at the instruction following the write to SUSPEND.

## SFR Definition 26.2. OSCICL: Internal H-F Oscillator Calibration

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

SFR Address = 0xB3; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7 | Unused | Unused. Read = 0; Write = Don't Care |
| 6:0 | OSCICL[6:0] | **Internal Oscillator Calibration Bits.**<br>These bits determine the internal oscillator period. When set to 0000000b, the H-F oscillator operates at its fastest setting. When set to 1111111b, the H-F oscillator operates at its slowest setting. The reset value is factory calibrated to generate an internal oscillator frequency of 49 MHz. |

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<br>or<br>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.

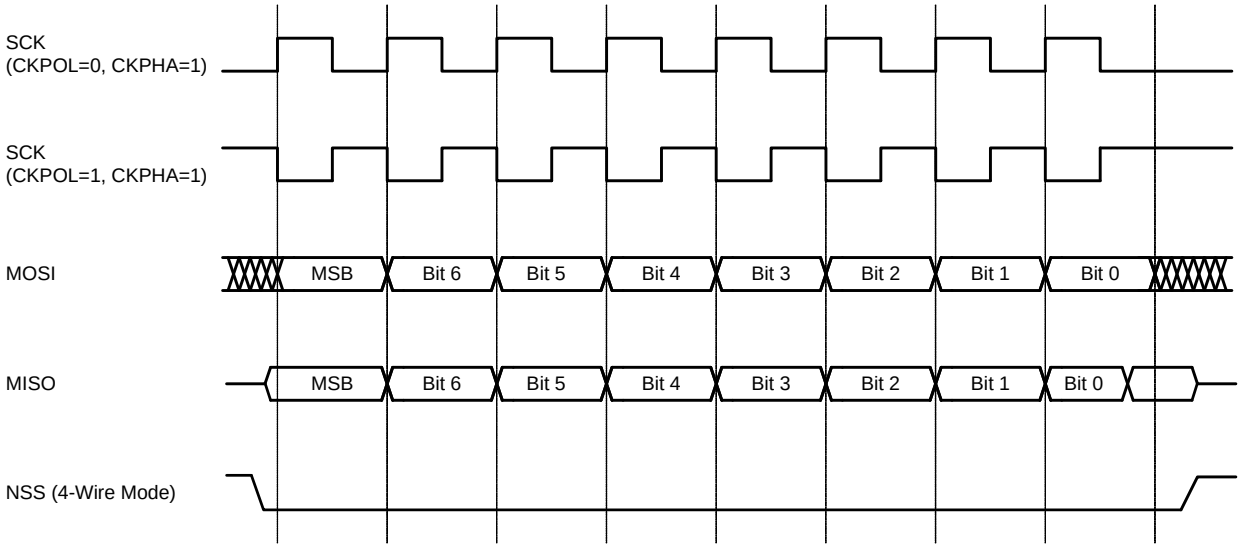


**Figure 30.7. Slave Mode Data/Clock Timing (CKPHA = 1)**

## 30.6. SPI Special Function Registers

SPI0 is accessed and controlled through four special function registers in the system controller: SPI0CN Control Register, SPI0DAT Data Register, SPI0CFG Configuration Register, and SPI0CKR Clock Rate Register. The four special function registers related to the operation of the SPI0 Bus are described in the following figures.

## SFR Definition 30.1. SPI0CFG: SPI0 Configuration

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

SFR Address = 0xA1; SFR Page = All Pages

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

**Note:** In slave mode, data on MOSI is sampled in the center of each data bit. In master mode, data on MISO is sampled one SYSCLK before the end of each data bit, to provide maximum settling time for the slave device. See Table 30.1 for timing parameters.

## SFR Definition 31.5. TL0: Timer 0 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TL0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8A; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | TL0[7:0] | **Timer 0 Low Byte.**<br>The TL0 register is the low byte of the 16-bit Timer 0. |

## SFR Definition 31.6. TL1: Timer 1 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TL1[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x8B; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | TL1[7:0] | **Timer 1 Low Byte.**<br>The TL1 register is the low byte of the 16-bit Timer 1. |

## 31.5. Timer 5

Timer 5 is a 16-bit timer formed by two 8-bit SFRs: TMR5L (low byte) and TMR5H (high byte). Timer 5 may operate in 16-bit auto-reload mode or (split) 8-bit auto-reload mode. The T5SPLIT bit (TMR5CN.3) defines

Timer 5 may be clocked by the system clock, the system clock divided by 12, or the external oscillator source divided by 8. Note that the external oscillator source divided by 8 is synchronized with the system clock.

### 31.5.1. 16-bit Timer with Auto-Reload

When T5SPLIT (TMR5CN.3) is zero, Timer 5 operates as a 16-bit timer with auto-reload. Timer 5 can be clocked by SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the Timer 5 reload registers (TMR5RLH and TMR5RLL) is loaded into the Timer 5 register as shown in Figure 31.12, and the Timer 5 High Byte Overflow Flag (TMR5CN.7) is set. If Timer 5 interrupts are enabled (if EIE1.7 is set), an interrupt will be generated on each Timer 5 overflow. Additionally, if Timer 5 interrupts are enabled and the TF5LEN bit is set (TMR5CN.5), an interrupt will be generated each time the lower 8 bits (TMR5L) overflow from 0xFF to 0x00.

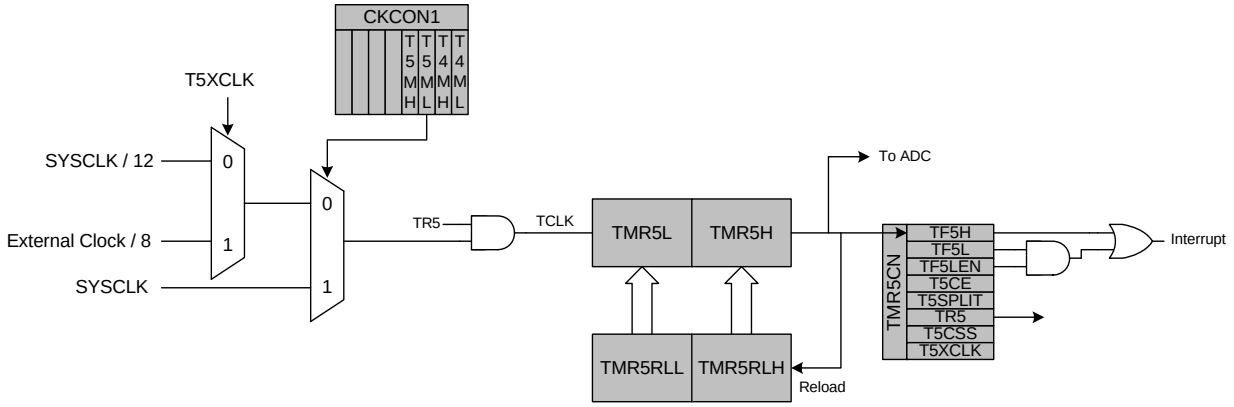


**Figure 31.12. Timer 5 16-Bit Mode Block Diagram**

## SFR Definition 31.28. TMR5H Timer 5 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR5H[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xCD; SFR Page = F

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR5H[7:0] | **Timer 5 High Byte.**<br>In 16-bit mode, the TMR5H register contains the high byte of the 16-bit Timer 5. In 8-bit mode, TMR5H contains the 8-bit high byte timer value. |

### 32.3.3. High-Speed Output Mode

In High-Speed Output mode, a module's associated CEXn pin is toggled each time a match occurs between the PCA Counter and the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1. An interrupt request is generated if the CCFn interrupt for that module is enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Setting the TOGn, MATn, and ECOMn bits in the PCA0CPMn register enables the High-Speed Output mode. If ECOMn is cleared, the associated pin will retain its state, and not toggle on the next match event.

**Important Note About Capture/Compare Registers**: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to 0; writing to PCA0CPHn sets ECOMn to 1.

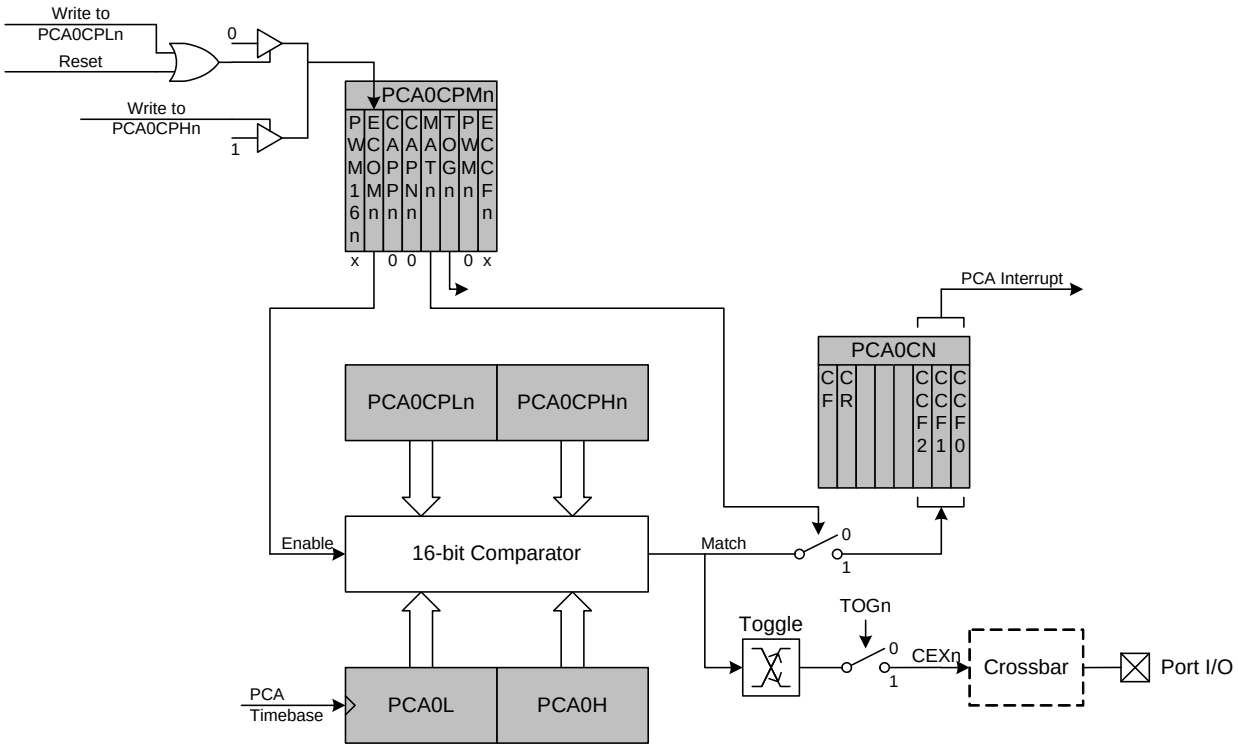


**Figure 32.6. PCA High-Speed Output Mode Diagram**

### 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.

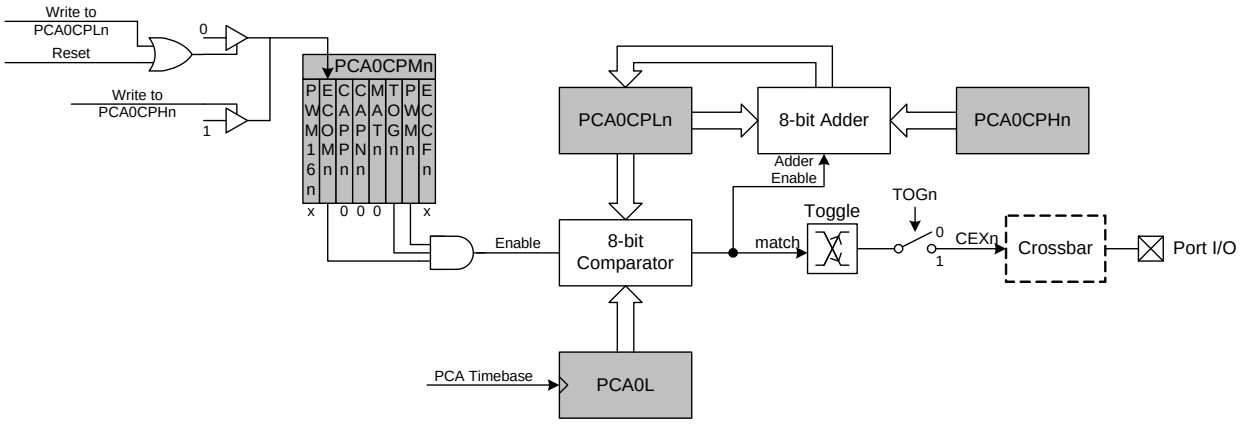


**Figure 32.7. PCA Frequency Output Mode**

### 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.

## 32.6. Register Descriptions for PCA0

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

### SFR Definition 32.1. PCA0CN: PCA Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CF | CR | | | | CCF2 | CCF1 | CCF0 |
| Type | R/W | R/W | R | R | R | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD8; SFR Page = All Pages; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | CF | **PCA Counter/Timer Overflow Flag.**<br>Set by hardware when the PCA Counter/Timer overflows from 0xFFFF to 0x0000. When the Counter/Timer Overflow (CF) interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |
| 6 | CR | **PCA Counter/Timer Run Control.**<br>This bit enables/disables the PCA Counter/Timer.<br>0: PCA Counter/Timer disabled.<br>1: PCA Counter/Timer enabled. |
| 5:3 | Unused | Unused. Read = 000b, Write = Don't care. |
| 2 | CCF2 | **PCA Module 2 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF2 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |
| 1 | CCF1 | **PCA Module 1 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF1 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |
| 0 | CCF0 | **PCA Module 0 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF0 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. |

## SFR Definition 32.4. PCA0CLR: PCA Comparator Clear Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CPCPOL | | | | | CPCE2 | CPCE1 | CPCE0 |
| 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 = 0xCE; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7 | CPCPOL | **Comparator Clear Polarity.**<br>Selects the polarity of the comparator result that will clear the PCA channel(s).<br>0: PCA channel(s) will be cleared when comparator result goes logic low<br>1: PCA channel(s) will be cleared when comparator result goes logic high |
| 6:3 | Reserved | Must write 0000b. |
| 2 | CPCE2 | **Comparator Clear Enable for CEX2.**<br>Enables the comparator clear function on PCA channel 2. |
| 1 | CPCE1 | **Comparator Clear Enable for CEX1.**<br>Enables the comparator clear function on PCA channel 1. |
| 0 | CPCE0 | **Comparator Clear Enable for CEX0.**<br>Enables the comparator clear function on PCA channel 0. |