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	25MHz
Connectivity	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	Cap Sense, POR, PWM, WDT
Number of I/O	39
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	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-TQFP
Supplier Device Package	48-TQFP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f707-gq

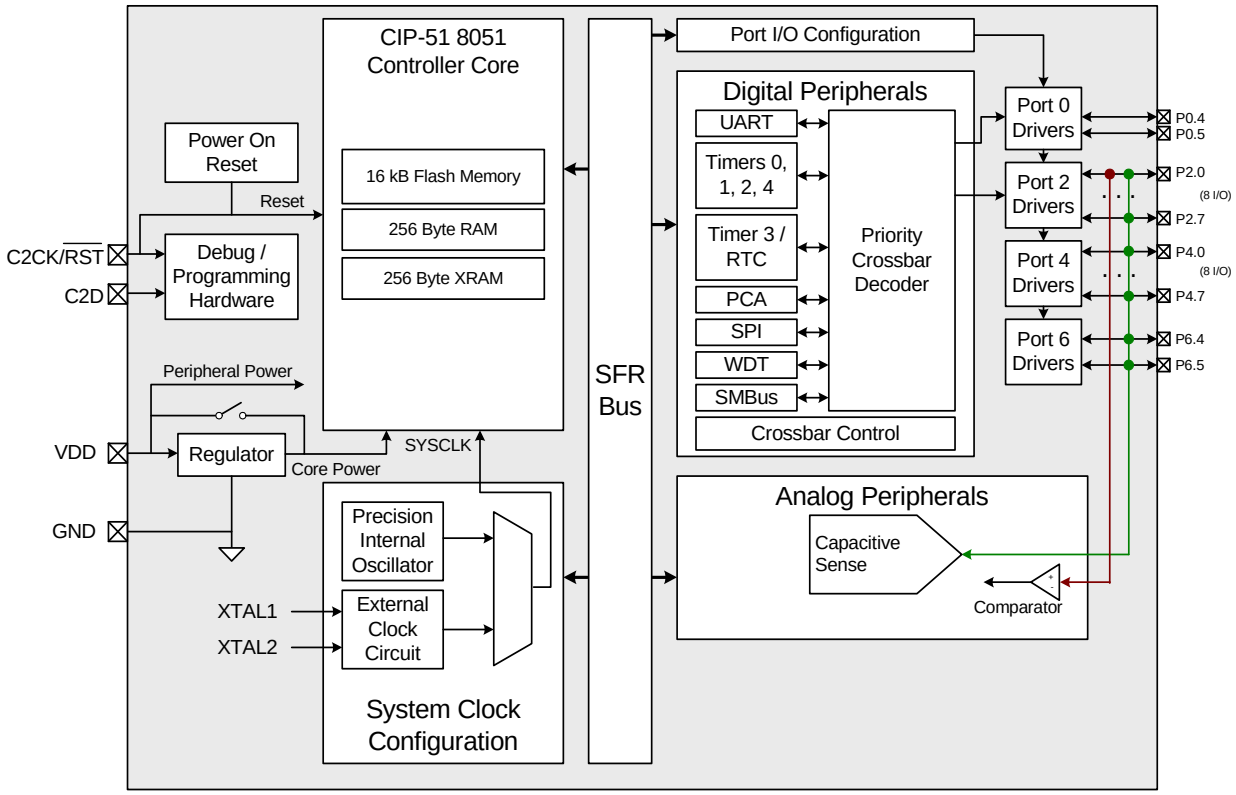

Figure 1.8. C8051F717 Block Diagram

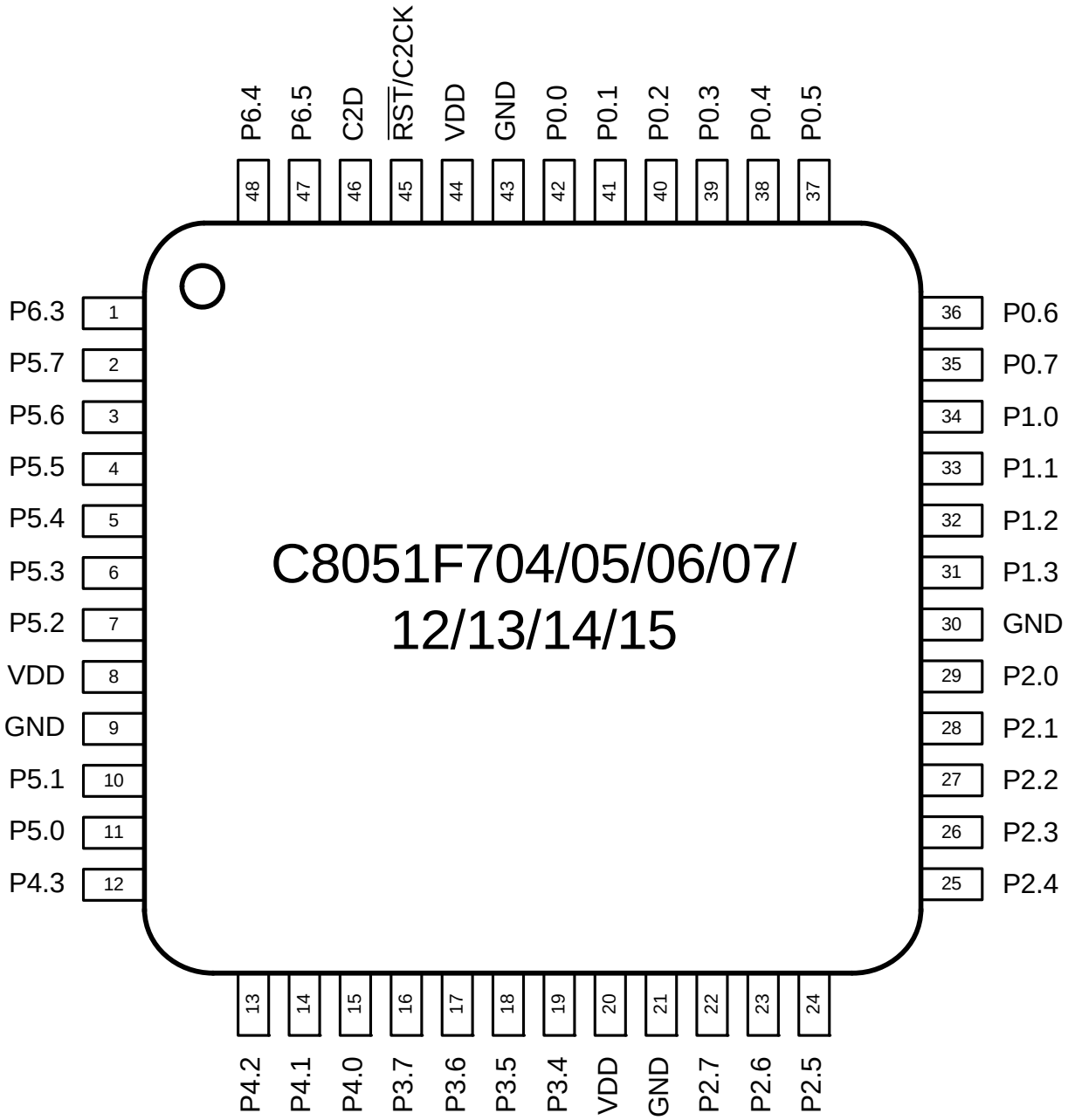

Figure 3.2. C8051F7xx-GQ QFP48 Pinout Diagram (Top View)

10.4.1. Window Detector Example

Figure 10.4 shows two example window comparisons for right-justified data, with ADC0LTH:ADC0LTL = 0x0080 (128d) and ADC0GTH:ADC0GTL = 0x0040 (64d). The input voltage can range from 0 to VREF x (1023/1024) with respect to GND, and is represented by a 10-bit unsigned integer value. In the left example, an AD0WINT interrupt will be generated if the ADC0 conversion word (ADC0H:ADC0L) is within the range defined by ADC0GTH:ADC0GTL and ADC0LTH:ADC0LTL (if 0x0040 < ADC0H:ADC0L < 0x0080). In the right example, and AD0WINT interrupt will be generated if the ADC0 conversion word is outside of the range defined by the ADC0GT and ADC0LT registers (if ADC0H:ADC0L < 0x0040 or ADC0H:ADC0L > 0x0080). Figure 10.5 shows an example using left-justified data with the same comparison values.

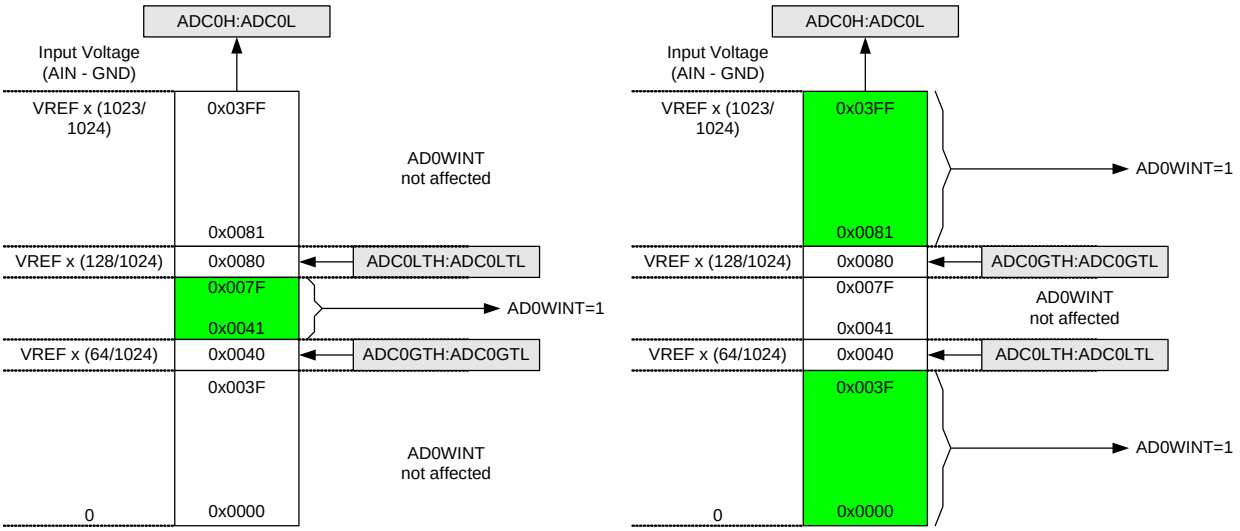

Figure 10.4. ADC Window Compare Example: Right-Justified Data

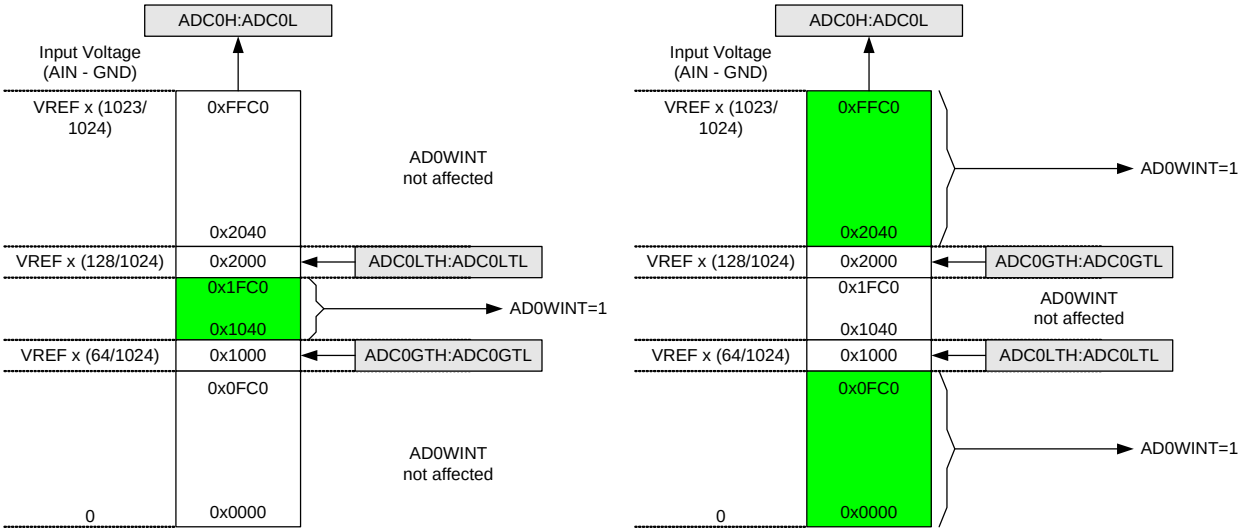

Figure 10.5. ADC Window Compare Example: Left-Justified Data

12.1. External Voltage References

To use an external voltage reference, REFSL[1:0] should be set to 00. Bypass capacitors should be added as recommended by the manufacturer of the external voltage reference.

12.2. Internal Voltage Reference Options

A 1.6 V high-speed reference is included on-chip. The high speed internal reference is selected by setting REFSL[1:0] to 11. When selected, the high-speed internal reference will be automatically enabled on an as-needed basis by ADC0.

For applications with a non-varying power supply voltage, using the power supply as the voltage reference can provide ADC0 with added dynamic range at the cost of reduced power supply noise rejection. To use the 1.8 to 3.6 V power supply voltage (V_{DD}) or the 1.8 V regulated digital supply voltage as the reference source, REFSL[1:0] should be set to 01 or 10, respectively.

12.3. Analog Ground Reference

To prevent ground noise generated by switching digital logic from affecting sensitive analog measurements, a separate analog ground reference option is available. When enabled, the ground reference for ADC0 is taken from the P0.1/AGND pin. Any external sensors sampled by ADC0 should be referenced to the P0.1/AGND pin. The separate analog ground reference option is enabled by setting REFGND to 1. Note that when using this option, P0.1/AGND must be connected to the same potential as GND.

12.4. Temperature Sensor Enable

The TEMPE bit in register REF0CN enables the temperature sensor. While disabled, the temperature sensor defaults to a high impedance state and any ADC0 measurements performed on the sensor result in meaningless data.

With the CIP-51's maximum system clock at 25 MHz, it has a peak throughput of 25 MIPS. The CIP-51 has a total of 109 instructions. The table below shows the total number of instructions that require each execution time.

Clocks to Execute	1	2	2/3	3	3/4	4	4/5	5	8
Number of Instructions	26	50	5	14	7	3	1	2	1

16.1. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

16.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 16.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

SFR Definition 16.5. B: B Register

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

SFR Address = 0xF0; SFR Page = All Pages; Bit-Addressable

Bit	Name	Function
7:0	B[7:0]	**B Register.** This register serves as a second accumulator for certain arithmetic operations.

SFR Definition 16.6. PSW: Program Status Word

Bit	7	6	5	4	3	2	1	0
Name	CY	AC	F0	RS[1:0]		OV	F1	PARITY
Type	R/W	R/W	R/W	R/W		R/W	R/W	R
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xD0; SFR Page = All Pages; Bit-Addressable

Bit	Name	Function
7	CY	**Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations.
6	AC	**Auxiliary Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations.
5	F0	**User Flag 0.** This is a bit-addressable, general purpose flag for use under software control.
4:3	RS[1:0]	**Register Bank Select.** These bits select which register bank is used during register accesses. 00: Bank 0, Addresses 0x00-0x07 01: Bank 1, Addresses 0x08-0x0F 10: Bank 2, Addresses 0x10-0x17 11: Bank 3, Addresses 0x18-0x1F
2	OV	**Overflow Flag.** This bit is set to 1 under the following circumstances: • An ADD, ADDC, or SUBB instruction causes a sign-change overflow. • A MUL instruction results in an overflow (result is greater than 255). • A DIV instruction causes a divide-by-zero condition. The OV bit is cleared to 0 by the ADD, ADDC, SUBB, MUL, and DIV instructions in all other cases.
1	F1	**User Flag 1.** This is a bit-addressable, general purpose flag for use under software control.
0	PARITY	**Parity Flag.** This bit is set to logic 1 if the sum of the eight bits in the accumulator is odd and cleared if the sum is even.

18.4.2. Non-multiplexed Configuration

In Non-multiplexed mode, the Data Bus and the Address Bus pins are not shared. An example of a Non-multiplexed Configuration is shown in Figure 18.2. See Section “18.6.1. Non-Multiplexed Mode” on page 120 for more information about Non-multiplexed operation.

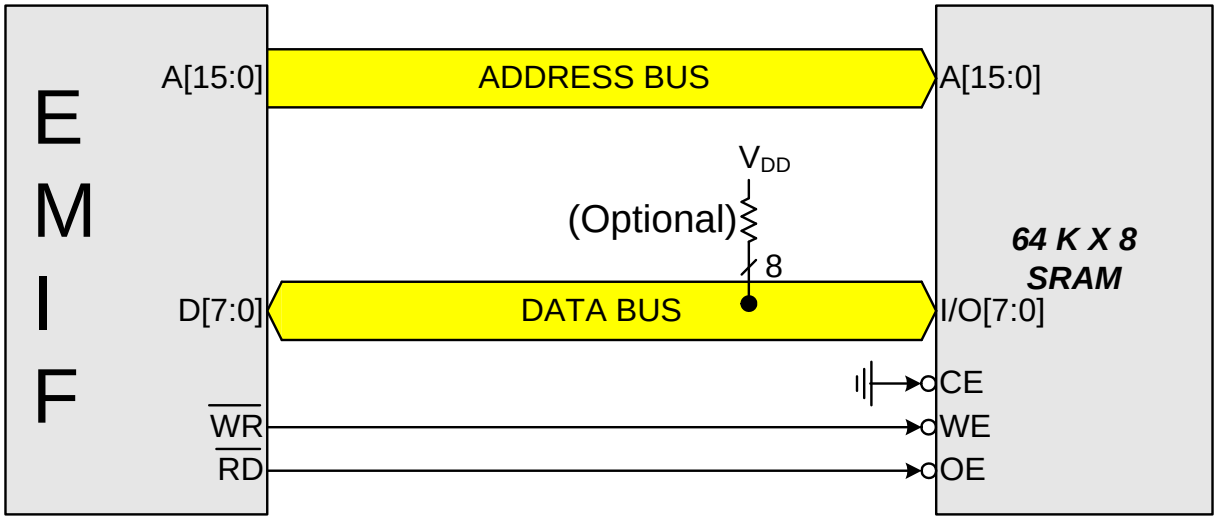

Figure 18.2. Non-multiplexed Configuration Example

21.3. $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ External Interrupts

The $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ external interrupt sources are configurable as active high or low, edge or level sensitive. The IN0PL ($\overline{\text{INT0}}$ Polarity) and IN1PL ($\overline{\text{INT1}}$ Polarity) bits in the IT01CF register select active high or active low; the IT0 and IT1 bits in TCON (Section “33.1. Timer 0 and Timer 1” on page 264) select level or edge sensitive. The table below lists the possible configurations.

IT0	IN0PL	$\overline{\text{INT0}}$ Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

IT1	IN1PL	$\overline{\text{INT1}}$ Interrupt
1	0	Active low, edge sensitive
1	1	Active high, edge sensitive
0	0	Active low, level sensitive
0	1	Active high, level sensitive

$\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are assigned to Port pins as defined in the IT01CF register (see SFR Definition 21.7). Note that $\overline{\text{INT0}}$ and $\overline{\text{INT0}}$ Port pin assignments are independent of any Crossbar assignments. $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ will monitor their assigned Port pins without disturbing the peripheral that was assigned the Port pin via the Crossbar. To assign a Port pin only to $\overline{\text{INT0}}$ and/or $\overline{\text{INT1}}$, configure the Crossbar to skip the selected pin(s). This is accomplished by setting the associated bit in register XBR0 (see Section “28.3. Priority Crossbar Decoder” on page 185 for complete details on configuring the Crossbar).

IE0 (TCON.1) and IE1 (TCON.3) serve as the interrupt-pending flags for the $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ external interrupts, respectively. If an $\overline{\text{INT0}}$ or $\overline{\text{INT1}}$ external interrupt is configured as edge-sensitive, the corresponding interrupt-pending flag is automatically cleared by the hardware when the CPU vectors to the ISR. When configured as level sensitive, the interrupt-pending flag remains logic 1 while the input is active as defined by the corresponding polarity bit (IN0PL or IN1PL); the flag remains logic 0 while the input is inactive. The external interrupt source must hold the input active until the interrupt request is recognized. It must then deactivate the interrupt request before execution of the ISR completes or another interrupt request will be generated.

23.4. EEPROM Security

RAM can only be downloaded to EEPROM after firmware writes a sequence of two bytes to EEKEY. In order to enable EEPROM writes:

1. Write the first EEPROM key code byte to EEKEY: 0x55
2. Write the second EEPROM key code byte to EEKEY: 0xAA

After a EEPROM writes have been enabled and a single write has executed, the control logic locks EEPROM writes until the two-byte unlock sequence has been entered into EEKEY again.

The protection state of the EEPROM can be observed by reading EEPSTATE (EEKEY2:0). This state can be read at any time without affecting the EEPROM's protection state.

If the two-byte unlock sequence is entered incorrectly, or if a write is attempted without first entering the two-byte sequence, EEPROM writes will be locked until the next power-on reset.

SFR Definition 23.1. EEADDR: EEPROM Byte Address

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

SFR Address = 0xB6; SFR Page = All Pages

Bit	Name	Description
7:5	Unused	Read = 000b; Write = Don't Care
4:0	EEADDR[4:0]	**EEPROM Byte Address** Selects one of 32 EEPROM bytes to read/write.

24.2. Stop Mode

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

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

24.3. Suspend Mode

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

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

Suspend mode can be terminated by five types of events, a port match (described in Section “28.5. Port Match” on page 192), a Timer 3 overflow (described in Section “33.3. Timer 3” on page 278), a comparator low output (if enabled), a capacitive sense greater-than comparator interrupt, or a device reset event. In order to run Timer 3 in Suspend mode, the timer must be configured to clock from the external clock source/8. 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.

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

30.4.4. Data Register

The SMBus Data register SMB0DAT holds a byte of serial data to be transmitted or one that has just been received. Software may safely read or write to the data register when the SI flag is set. Software should not attempt to access the SMB0DAT register when the SMBus is enabled and the SI flag is cleared to logic 0, as the interface may be in the process of shifting a byte of data into or out of the register.

Data in SMB0DAT is always shifted out MSB first. After a byte has been received, the first bit of received data is located at the MSB of SMB0DAT. While data is being shifted out, data on the bus is simultaneously being shifted in. SMB0DAT always contains the last data byte present on the bus. In the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data or address in SMB0DAT.

SFR Definition 30.5. SMB0DAT: SMBus Data

Bit	7	6	5	4	3	2	1	0
Name	SMB0DAT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xC2; SFR Page = 0

Bit	Name	Function
7:0	SMB0DAT[7:0]	**SMBus Data.** The SMB0DAT register contains a byte of data to be transmitted on the SMBus serial interface or a byte that has just been received on the SMBus serial interface. The CPU can read from or write to this register whenever the SI serial interrupt flag (SMB0CN.0) is set to logic 1. The serial data in the register remains stable as long as the SI flag is set. When the SI flag is not set, the system may be in the process of shifting data in/out and the CPU should not attempt to access this register.

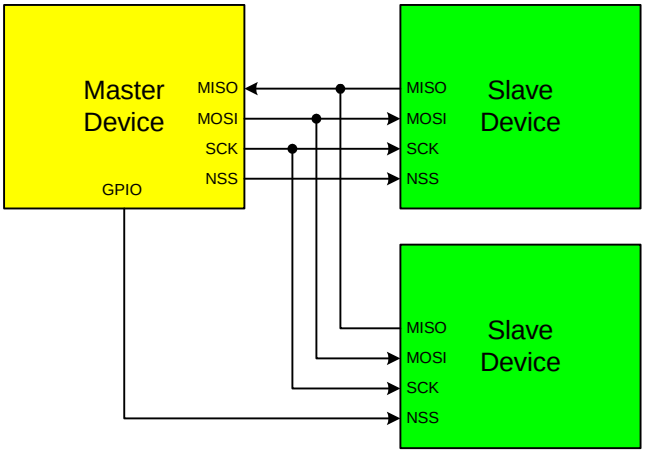

Figure 31.4. 4-Wire Single Master Mode and Slave Mode Connection Diagram

31.3. 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. The NSS signal must be driven low at least 2 system clocks before the first active edge of SCK for each byte transfer. Figure 31.4 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 31.3 shows a connection diagram between a slave device in 3-wire slave mode and a master device.

SFR Definition 33.9. TMR2RLL: Timer 2 Reload Register Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLL[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCA; SFR Page = 0

Bit	Name	Function
7:0	TMR2RLL[7:0]	**Timer 2 Reload Register Low Byte.** TMR2RLL holds the low byte of the reload value for Timer 2.

SFR Definition 33.10. TMR2RLH: Timer 2 Reload Register High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLH[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCB; SFR Page = 0

Bit	Name	Function
7:0	TMR2RLH[7:0]	**Timer 2 Reload Register High Byte.** TMR2RLH holds the high byte of the reload value for Timer 2.

SFR Definition 33.14. 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 Address = 0x92; SFR Page = 0

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 33.15. 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 Address = 0x93; SFR Page = 0

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.

34.1. PCA Counter/Timer

The 16-bit PCA counter/timer consists of two 8-bit SFRs: PCA0L and PCA0H. PCA0H is the high byte (MSB) of the 16-bit counter/timer and PCA0L is the low byte (LSB). Reading PCA0L automatically latches the value of PCA0H into a “snapshot” register; the following PCA0H read accesses this “snapshot” register. **Reading the PCA0L Register first guarantees an accurate reading of the entire 16-bit PCA0 counter.** Reading PCA0H or PCA0L does not disturb the counter operation. The CPS2–CPS0 bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 34.1.

When the counter/timer overflows from 0xFFFF to 0x0000, the Counter Overflow Flag (CF) in PCA0MD is set to logic 1 and an interrupt request is generated if CF interrupts are enabled. Setting the ECF bit in PCA0MD to logic 1 enables the CF flag to generate an interrupt request. The CF bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Clearing the CIDL bit in the PCA0MD register allows the PCA to continue normal operation while the CPU is in Idle mode.

Table 34.1. PCA Timebase Input Options

CPS2	CPS1	CPS0	Timebase
0	0	0	System clock divided by 12
0	0	1	System clock divided by 4
0	1	0	Timer 0 overflow
0	1	1	High-to-low transitions on ECI (max rate = system clock divided by 4)
1	0	0	System clock
1	0	1	External oscillator source divided by 8[*]
1	1	x	Reserved
Note: External oscillator source divided by 8 is synchronized with the system clock.			

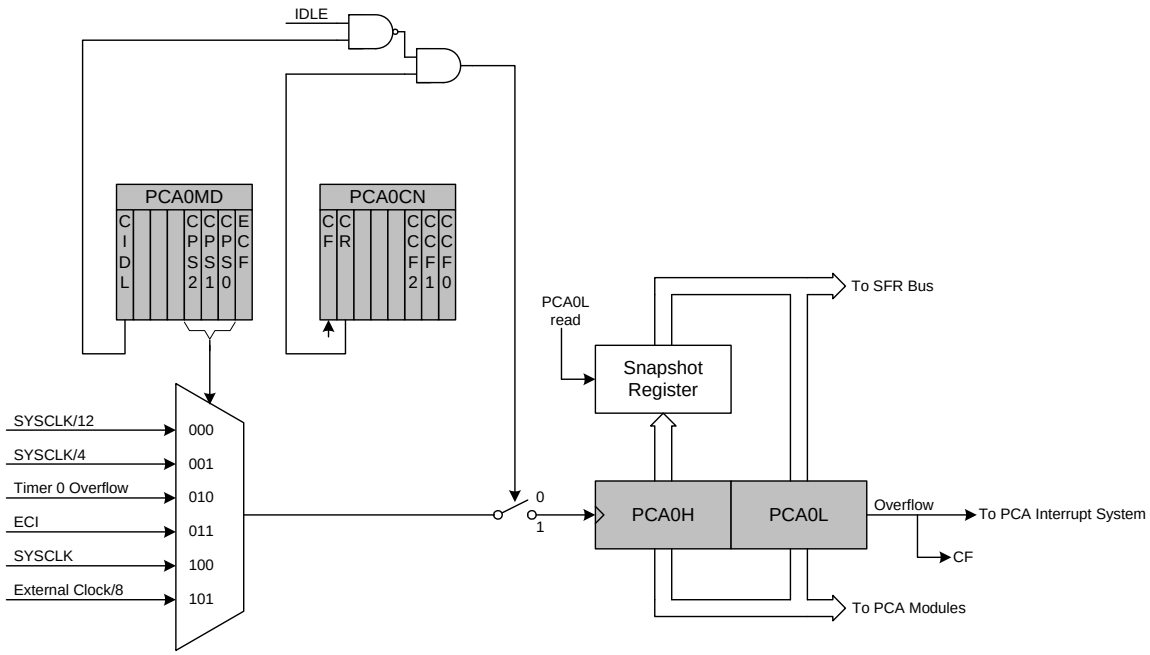

Figure 34.2. PCA Counter/Timer Block Diagram

34.3.2. Software Timer (Compare) Mode

In Software Timer mode, the PCA counter/timer value is compared to 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 ECOMn and MATn bits in the PCA0CPMn register enables Software Timer mode.

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.

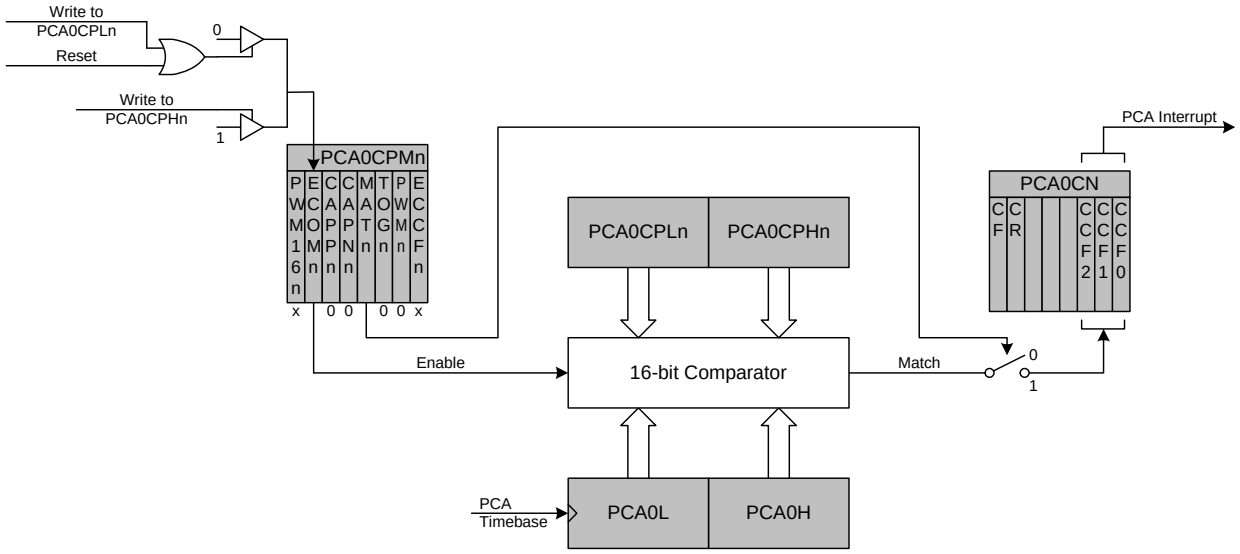

Figure 34.5. PCA Software Timer Mode Diagram

34.3.4. 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 34.1.

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

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

Equation 34.1. 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, CEXn 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. 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.

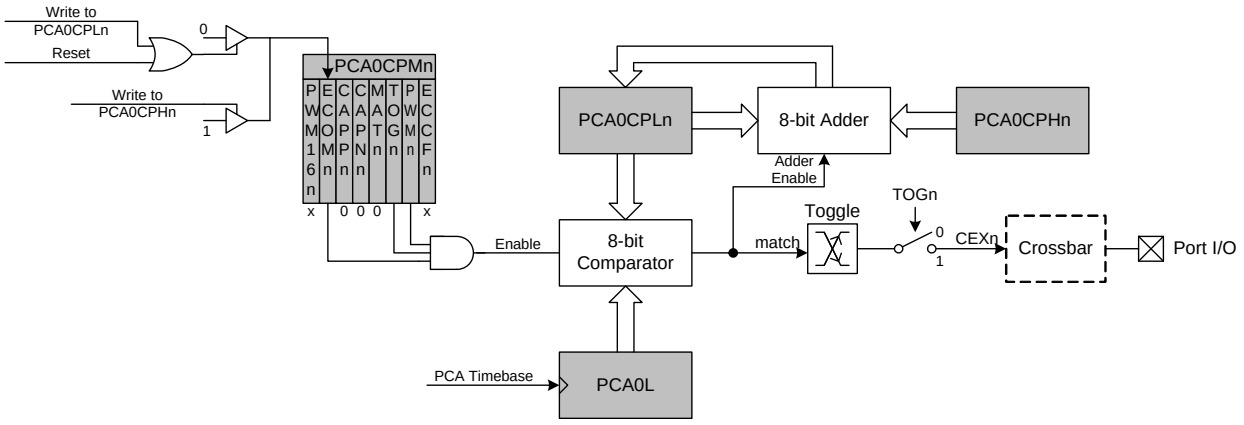

Figure 34.7. PCA Frequency Output Mode

34.3.6. 16-Bit Pulse Width Modulator Mode

A PCA module may also be operated in 16-Bit PWM mode. 16-bit PWM mode is independent of the other (8/9/10/11-bit) PWM modes. In this mode, the 16-bit capture/compare module defines the number of PCA clocks for the low time of the PWM signal. When the PCA counter matches the module contents, the output on CEXn is asserted high; when the 16-bit counter overflows, CEXn is asserted low. To output a varying duty cycle, new value writes should be synchronized with PCA CCFn match interrupts. 16-Bit PWM Mode is enabled by setting the ECOMn, PWMn, and PWM16n bits in the PCA0CPMn register. For a varying duty cycle, match interrupts should be enabled (ECCFn = 1 AND MATn = 1) to help synchronize the capture/compare register writes. If the MATn bit is set to 1, the CCFn flag for the module will be set each time a 16-bit comparator match (rising edge) occurs. The CF flag in PCA0CN can be used to detect the overflow (falling edge). The duty cycle for 16-Bit PWM Mode is given by Equation 34.4.

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.

$$\text{Duty Cycle} = \frac{(65536 - PCA0CPn)}{65536}$$

Equation 34.4. 16-Bit PWM Duty Cycle

Using Equation 34.4, the largest duty cycle is 100% (PCA0CPn = 0), and the smallest duty cycle is 0.0015% (PCA0CPn = 0xFFFF). A 0% duty cycle may be generated by clearing the ECOMn bit to 0.

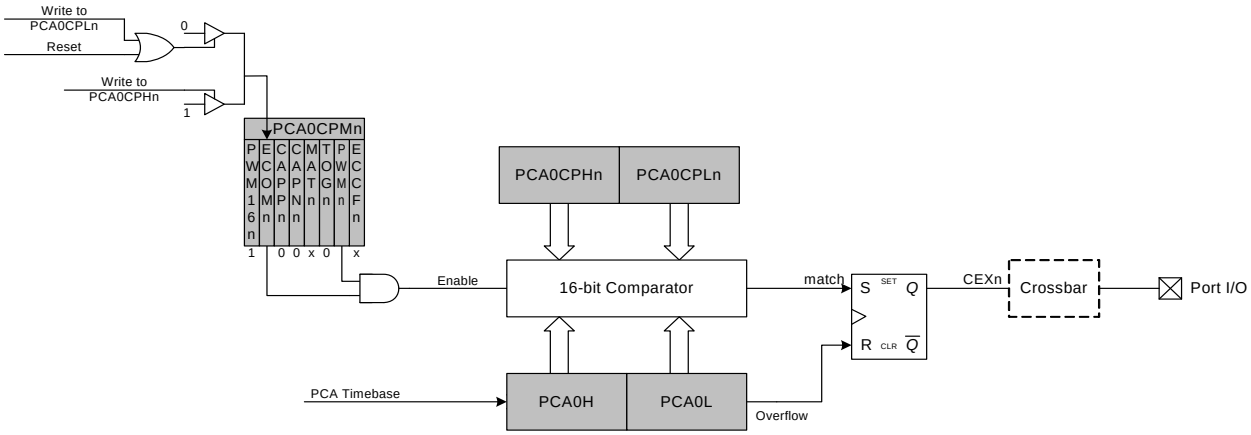

Figure 34.10. PCA 16-Bit PWM Mode