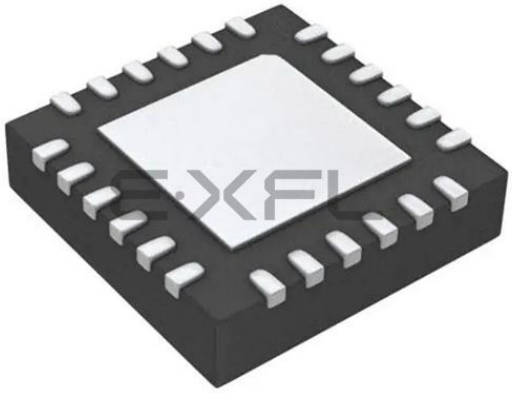

**Welcome to E-XFL.COM**

## 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, Temp Sensor, WDT |
| Number of I/O | 18 |
| Program Memory Size | 16KB (16K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 1.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.8V ~ 5.25V |
| Data Converters | A/D 18x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 125°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/c8051f543-imr |

## SFR Definition 5.11. 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 = 0x00

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

## SFR Definition 5.12. 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 = 0x00

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

### 5.4.1. Window Detector In Single-Ended Mode

Figure 5.6 shows two example window comparisons for right-justified data with ADC0LTH:ADC0LTL = 0x0200 (512d) and ADC0GTH:ADC0GTL = 0x0100 (256d). The input voltage can range from 0 to $V_{REF}$ x (4095/4096) with respect to GND, and is represented by a 12-bit unsigned integer value. The repeat count is set to one. 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 0x0100 < ADC0H:ADC0L < 0x0200). 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 < 0x0100 or ADC0H:ADC0L > 0x0200). Figure 5.7 shows an example using left-justified data with the same comparison values.

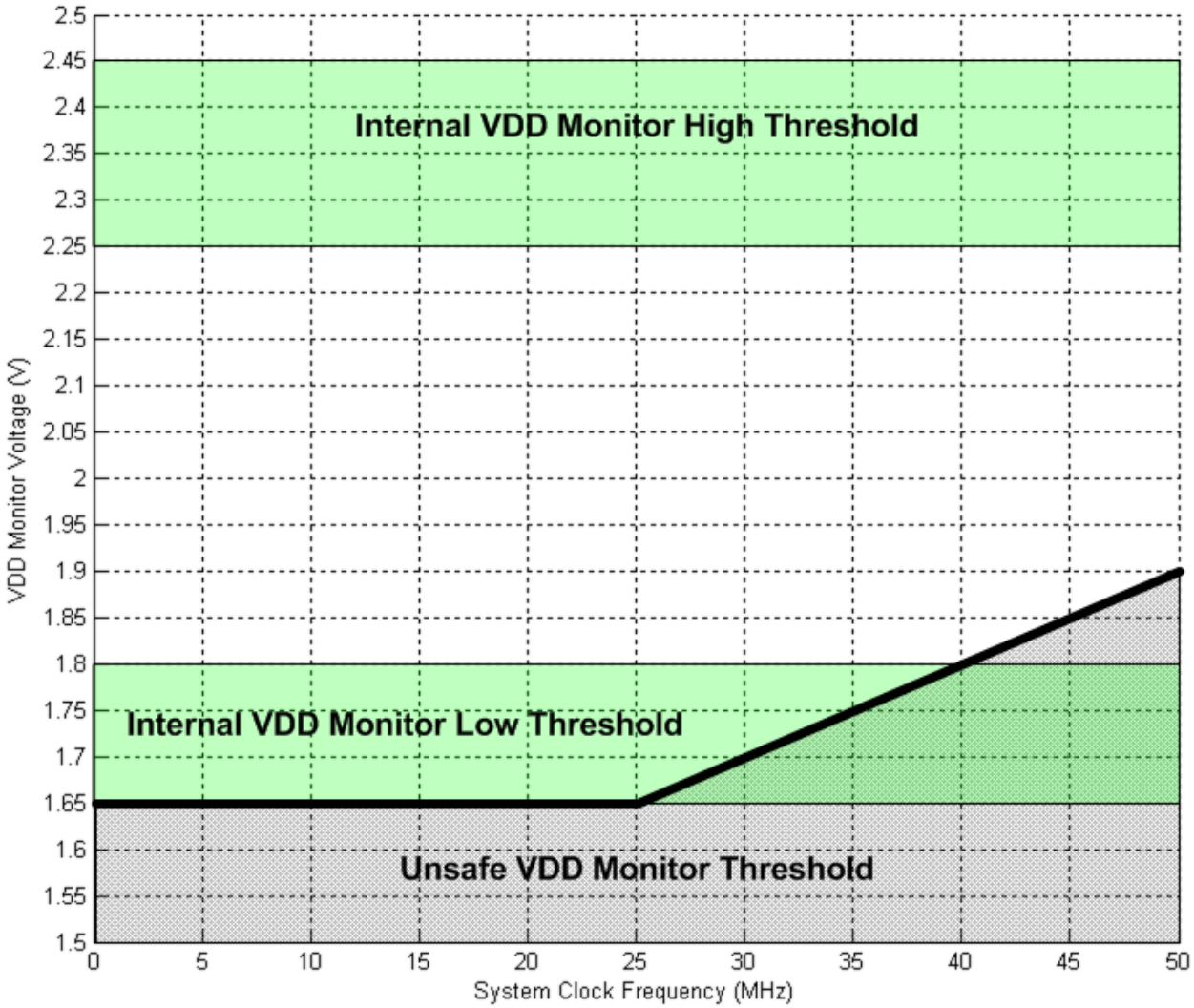


**Figure 6.1. Minimum VDD Monitor Threshold vs. System Clock Frequency**

**Note:** With system clock frequencies greater than 25 MHz, the $V_{DD}$ monitor level should be set to the high threshold (VDMLVL = 1b in SFR VDM0CN) to prevent undefined CPU operation. The high threshold should only be used with an external regulator powering $V_{DD}$ directly. See Figure 9.2 on page 73 for the recommended power supply connections.

## 6.2. Temperature Sensor

An on-chip temperature sensor is included on the C8051F54x devices which can be directly accessed via the ADC multiplexer in single-ended configuration. To use the ADC to measure the temperature sensor, the ADC multiplexer channel should be configured to connect to the temperature sensor. The temperature sensor transfer function is shown in Figure 6.3. The output voltage ($V_{TEMP}$) is the positive ADC input is selected by bits AD0MX[4:0] in register ADC0MX. The TEMPE bit in register REF0CN enables/disables the temperature sensor, as described in SFR Definition 7.1. 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 6.10 for the slope and offset parameters of the temperature sensor.

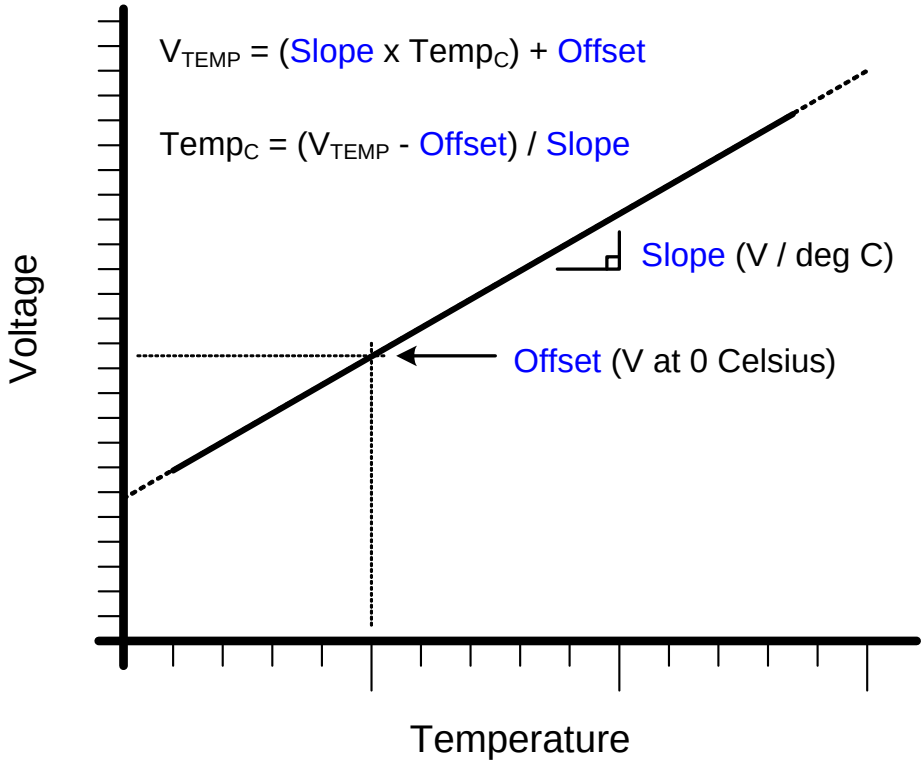


**Figure 6.3. Temperature Sensor Transfer Function**

## SFR Definition 8.2. CPT0MD: Comparator0 Mode Selection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | CP0RIE | CP0FIE | | | CP0MD[1:0] | |
| Type | R | R | R/W | R/W | R | R | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

SFR Address = 0x9B; SFR Page = 0x00

| Bit | Name | Function |
|---|---|---|
| 7:6 | Unused | Read = 00b, Write = Don't Care. |
| 5 | CP0RIE | **Comparator0 Rising-Edge Interrupt Enable.**<br>0: Comparator0 Rising-edge interrupt disabled.<br>1: Comparator0 Rising-edge interrupt enabled. |
| 4 | CP0FIE | **Comparator0 Falling-Edge Interrupt Enable.**<br>0: Comparator0 Falling-edge interrupt disabled.<br>1: Comparator0 Falling-edge interrupt enabled. |
| 3:2 | Unused | Read = 00b, Write = don't care. |
| 1:0 | CP0MD[1:0] | **Comparator0 Mode Select.**<br>These bits affect the response time and power consumption for Comparator0.<br>00: Mode 0 (Fastest Response Time, Highest Power Consumption)<br>01: Mode 1<br>10: Mode 2<br>11: Mode 3 (Slowest Response Time, Lowest Power Consumption) |

# 12. Special Function Registers

The direct-access data memory locations from 0x80 to 0xFF constitute the special function registers (SFRs). The SFRs provide control and data exchange with the C8051F54x's resources and peripherals. The CIP-51 controller core duplicates the SFRs found in a typical 8051 implementation as well as implementing additional SFRs used to configure and access the sub-systems unique to the C8051F54x. This allows the addition of new functionality while retaining compatibility with the MCS-51™ instruction set. Table 12.2 lists the SFRs implemented in the C8051F54x device family.

The SFR registers are accessed anytime the direct addressing mode is used to access memory locations from 0x80 to 0xFF. SFRs with addresses ending in 0x0 or 0x8 (e.g., P0, TCON, SCON0, IE, etc.) are bit-addressable as well as byte-addressable. All other SFRs are byte-addressable only. Unoccupied addresses in the SFR space are reserved for future use. Accessing unoccupied addresses in the SFR space will have an indeterminate effect and should be avoided. Refer to the corresponding pages of the data sheet, as indicated in Table 12.2, for a detailed description of each register.

## 12.1. SFR Paging

The CIP-51 features SFR paging, allowing the device to map many SFRs into the 0x80 to 0xFF memory address space. The SFR memory space has 256 *pages*. In this way, each memory location from 0x80 to 0xFF can access up to 256 SFRs. The C8051F54x family of devices utilizes two SFR pages: 0x00 and 0x0F. SFR pages are selected using the Special Function Register Page Selection register, SFRPAGE (see SFR Definition 11.3). The procedure for reading and writing an SFR is as follows:

1. Select the appropriate SFR page number using the SFRPAGE register.
2. Use direct accessing mode to read or write the special function register (MOV instruction).

## 12.2. Interrupts and SFR Paging

When an interrupt occurs, the SFR Page Register will automatically switch to the SFR page containing the flag bit that caused the interrupt. The automatic SFR Page switch function conveniently removes the burden of switching SFR pages from the interrupt service routine. Upon execution of the RETI instruction, the SFR page is automatically restored to the SFR Page in use prior to the interrupt. This is accomplished via a three-byte *SFR Page Stack*. The top byte of the stack is SFRPAGE, the current SFR Page. The second byte of the SFR Page Stack is SFRNEXT. The third, or bottom byte of the SFR Page Stack is SFRLAST. Upon an interrupt, the current SFRPAGE value is pushed to the SFRNEXT byte, and the value of SFRNEXT is pushed to SFRLAST. Hardware then loads SFRPAGE with the SFR Page containing the flag bit associated with the interrupt. On a return from interrupt, the SFR Page Stack is popped resulting in the value of SFRNEXT returning to the SFRPAGE register, thereby restoring the SFR page context without software intervention. The value in SFRLAST (0x00 if there is no SFR Page value in the bottom of the stack) of the stack is placed in SFRNEXT register. If desired, the values stored in SFRNEXT and SFRLAST may be modified during an interrupt, enabling the CPU to return to a different SFR Page upon execution of the RETI instruction (on interrupt exit). Modifying registers in the SFR Page Stack does not cause a push or pop of the stack. Only interrupt calls and returns will cause push/pop operations on the SFR Page Stack.

On the C8051F54x devices, vectoring to an interrupt will switch SFRPAGE to page 0x00.

**Table 12.1. Special Function Register (SFR) Memory Map for Pages 0x0 and 0xF**

| Address | Page | 0(8) | 1(9) | 2(A) | 3(B) | 4(C) | 5(D) | 6(E) | 7(F) |
|---|---|---|---|---|---|---|---|---|---|
| F8 | 0F | SPI0CN | PCA0L<br>SN0 | PCA0H<br>SN1 | PCA0CPL0<br>SN2 | PCA0CPH0<br>SN3 | PCACPL4 | PCACPH4 | VDM0CN |
| F0 | 0F | B<br>(All Pages) | P0MAT<br>P0MDIN | P0MASK<br>P1MDIN | P1MAT<br>P2MDIN | P1MASK<br>P3MDIN | | EIP1<br>EIP1 | EIP2<br>EIP2 |
| E8 | 0F | ADC0CN | PCA0CPL1 | PCA0CPH1 | PCA0CPL2 | PCA0CPH2 | PCA0CPL3 | PCA0CPL3 | RSTSRC |
| E0 | 0F | ACC<br>(All Pages) | XBR0 | XBR1 | CCH0CN | IT01CF | | EIE1<br>(All Pages) | EIE2<br>(All Pages) |
| D8 | 0F | PCA0CN | PCA0MD<br>PCA0PWM | PCA0CPM0 | PCA0CPM1 | PCA0CPM2 | PCA0CPM3 | PCA0CPM4 | PCA0CPM5 |
| D0 | 0F | PSW<br>(All Pages) | REF0CN | LIN0DATA | LIN0ADDR | P0SKIP | P1SKIP | P2SKIP | P3SKIP |
| C8 | 0F | TMR2CN | REG0CN<br>LIN0CF | TMR2RLL | TMR2RLH | TMR2L | TMR2H | PCA0CPL5 | PCA0CPH5 |
| C0 | 0F | SMB0CN | SMB0CF | SMB0DAT | ADC0GTL | ADC0GTH | ADC0LTL | ADC0LTH | XBR2 |
| B8 | 0F | IP<br>(All Pages) | | ADC0TK | ADC0MX | ADC0CF | ADC0L | ADC0H | |
| B0 | 0F | P3<br>(All Pages) | P2MAT | P2MASK | | | | FLSCL<br>(All Pages) | FLKEY<br>(All Pages) |
| A8 | 0F | IE<br>(All Pages) | SMOD0 | EMI0CN | SBCON0 | SBRLL0 | SBRLH0 | P3MAT<br>P3MDOUT | P3MASK |
| A0 | 0F | P2<br>(All Pages) | SPI0CFG<br>OSCICN | SPI0CKR<br>OSCICRS | SPI0DAT | P0MDOUT | P1MDOUT | P2MDOUT | SFRPAGE<br>(All Pages) |
| 98 | 0F | SCON0 | SBUF0 | CPT0CN | CPT0MD | CPT0MX | CPT1CN | CPT1MD<br>OSCIFIN | CPT1MX<br>OSCXCN |
| 90 | 0F | P1<br>(All Pages) | TMR3CN | TMR3RLL | TMR3RLH | TMR3L | TMR3H | | CLKMUL |
| 88 | 0F | TCON<br>(All Pages) | TMOD<br>(All Pages) | TL0<br>(All Pages) | TL1<br>(All Pages) | TH0<br>(All Pages) | TH1<br>(All Pages) | CKCON<br>(All Pages) | PSCTL<br>CLKSEL |
| 80 | 0F | P0<br>(All Pages) | SP<br>(All Pages) | DPL<br>(All Pages) | DPH<br>(All Pages) | SFR0CN | SFRNEXT<br>(All Pages) | SFRLAST<br>(All Pages) | PCON<br>(All Pages) |
| | | 0(8) | 1(9) | 2(A) | 3(B) | 4(C) | 5(D) | 6(E) | 7(F) |

(bit addressable)

# 13. Interrupts

The C8051F54x devices include an extended interrupt system supporting a total of 14 interrupt sources with two priority levels. The allocation of interrupt sources between on-chip peripherals and external inputs 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, or 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.

## 13.1. MCU Interrupt Sources and Vectors

The C8051F54x MCUs support 17 interrupt sources. Software can simulate an interrupt by setting any interrupt-pending flag to logic 1. If interrupts are enabled for the flag, an interrupt request will be generated and the CPU will vector to the ISR address associated with the interrupt-pending flag. MCU interrupt sources, associated vector addresses, priority order and control bits are summarized in Table 13.1. Refer to the datasheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

## SFR Definition 13.1. 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; Bit-Addressable; SFR Page = All Pages

| 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 $\overline{\text{INT1}}$ 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. |

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

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 “23.1. Timer 0 and Timer 1” on page 229) 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 13.7). 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 “18.3. Priority Crossbar Decoder” on page 150 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.

## 14.4. Flash Write and Erase Guidelines

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

The following guidelines are recommended for any system which contains routines which write or erase Flash from code.

### 14.4.1. $V_{DD}$ Maintenance and the $V_{DD}$ monitor

1. If the system power supply is subject to voltage or current "spikes," add sufficient transient protection devices to the power supply to ensure that the supply voltages listed in the Absolute Maximum Ratings table are not exceeded.
2. Enable the on-chip $V_{DD}$ monitor and enable the $V_{DD}$ monitor as a reset source as early in code as possible. This should be the first set of instructions executed after the Reset Vector. For C-based systems, this will involve modifying the startup code added by the C compiler. See your compiler documentation for more details. Make certain that there are no delays in software between enabling the $V_{DD}$ monitor and enabling the $V_{DD}$ monitor as a reset source. Code examples showing this can be found in “AN201: Writing to Flash from Firmware", available from the Silicon Laboratories web site.
3. As an added precaution, explicitly enable the $V_{DD}$ monitor and enable the $V_{DD}$ monitor as a reset source inside the functions that write and erase Flash memory. The $V_{DD}$ monitor enable instructions should be placed just after the instruction to set PSWE to a 1, but before the Flash write or erase operation instruction.
4. Make certain that all writes to the RSTSRC (Reset Sources) register use direct assignment operators and explicitly DO NOT use the bit-wise operators (such as AND or OR). For example, "RSTSRC = 0x02" is correct. "RSTSRC |= 0x02" is incorrect.
5. Make certain that all writes to the RSTSRC register explicitly set the PORSF bit to a 1. Areas to check are initialization code which enables other reset sources, such as the Missing Clock Detector or Comparator, for example, and instructions which force a Software Reset. A global search on "RSTSRC" can quickly verify this.

### 14.4.2. 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 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 variable 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.

# 16. Reset Sources

Reset circuitry allows the controller to be easily placed in a predefined default condition. On entry to this reset state, the following occur:

- CIP-51 halts program execution
- Special Function Registers (SFRs) are initialized to their defined reset values
- External Port pins are forced to a known state
- Interrupts and timers are disabled.

All SFRs are reset to the predefined values noted in the SFR detailed descriptions. The contents of internal data memory are unaffected during a reset; any previously stored data is preserved. However, since the stack pointer SFR is reset, the stack is effectively lost, even though the data on the stack is not altered.

The Port I/O latches are reset to 0xFF (all logic ones) in open-drain mode. Weak pullups are enabled during and after the reset. For $V_{DD}$ Monitor and power-on resets, the $\overline{RST}$ pin is driven low until the device exits the reset state.

On exit from the reset state, the program counter (PC) is reset, and the system clock defaults to the internal oscillator. The Watchdog Timer is enabled with the system clock divided by 12 as its clock source. Program execution begins at location 0x0000.

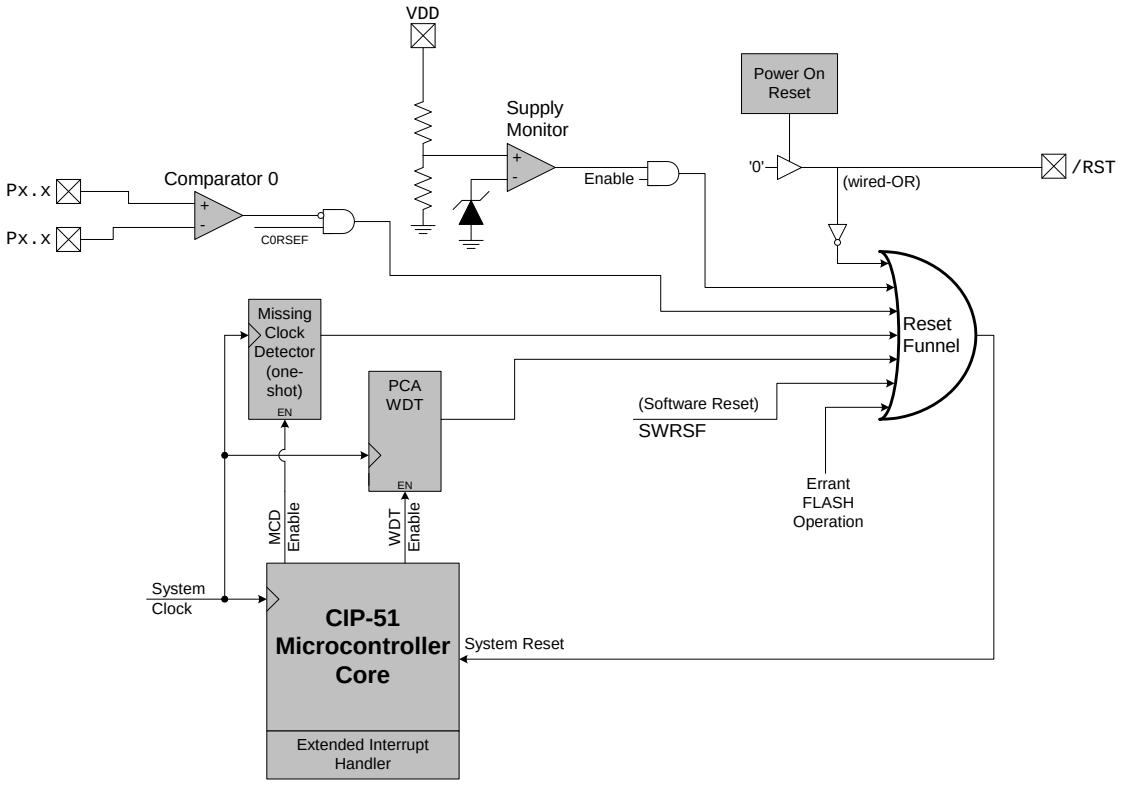


**Figure 16.1. Reset Sources**

## 16.5. Comparator0 Reset

Comparator0 can be configured as a reset source by writing a 1 to the C0RSEF flag (RSTSRC.5). Comparator0 should be enabled and allowed to settle prior to writing to C0RSEF to prevent any turn-on chatter on the output from generating an unwanted reset. The Comparator0 reset is active-low: if the non-inverting input voltage (on CP0+) is less than the inverting input voltage (on CP0–), the device is put into the reset state. After a Comparator0 reset, the C0RSEF flag (RSTSRC.5) will read 1 signifying Comparator0 as the reset source; otherwise, this bit reads 0. The state of the RST pin is unaffected by this reset.

## 16.6. PCA Watchdog Timer Reset

The programmable Watchdog Timer (WDT) function of the Programmable Counter Array (PCA) can be used to prevent software from running out of control during a system malfunction. The PCA WDT function can be enabled or disabled by software as described in Section “24.4. Watchdog Timer Mode” on page 260; the WDT is enabled and clocked by SYSCLK/12 following any reset. If a system malfunction prevents user software from updating the WDT, a reset is generated and the WDTRSF bit (RSTSRC.5) is set to 1. The state of the RST pin is unaffected by this reset.

## 16.7. Flash Error Reset

If a Flash read/write/erase or program read targets an illegal address, a system reset is generated. This may occur due to any of the following:

- A Flash write or erase is attempted above user code space. This occurs when PSWE is set to 1 and a MOVX write operation targets an address in or above the reserved space.
- A Flash read is attempted above user code space. This occurs when a MOVC operation targets an address in or above the reserved space.
- A Program read is attempted above user code space. This occurs when user code attempts to branch to an address in or above the reserved space.
- A Flash read, write or erase attempt is restricted due to a Flash security setting (see Section “14.3. Security Options” on page 119).
- A Flash read, write, or erase is attempted when the VDD Monitor is not enabled to the high threshold and set as a reset source.

The FERROR bit (RSTSRC.6) is set following a Flash error reset. The state of the RST pin is unaffected by this reset.

## 16.8. Software Reset

Software may force a reset by writing a 1 to the SWRSF bit (RSTSRC.4). The SWRSF bit will read 1 following a software forced reset. The state of the RST pin is unaffected by this reset.

## SFR Definition 18.21. P2MDIN: Port 2 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P2MDIN[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xF3; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MDIN[7:0] | **Analog Configuration Bits for P2.7–P2.0 (respectively).**<br>Port pins configured for analog mode have their weak pull-up and digital receiver disabled. For analog mode, the pin also needs to be configured for open-drain mode in the P2MDOUT register.<br>0: Corresponding P2.n pin is configured for analog mode.<br>1: Corresponding P2.n pin is not configured for analog mode. |
| **Note:** P2.2-P2.7 are only available on the 32-pin packages. | | |

## SFR Definition 18.22. P2MDOUT: Port 2 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P2MDOUT[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xA6; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2MDOUT[7:0] | **Output Configuration Bits for P2.7–P2.0 (respectively).**<br>These bits are ignored if the corresponding bit in register P2MDIN is logic 0.<br>0: Corresponding P2.n Output is open-drain.<br>1: Corresponding P2.n Output is push-pull. |
| **Note:** P2.2-P2.7 are only available on the 32-pin packages. | | |

## SFR Definition 18.23. P2SKIP: Port 2 Skip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P2SKIP[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xD6; SFR Page = 0x0F

| Bit | Name | Function |
|---|---|---|
| 7:0 | P2SKIP[7:0] | **Port 2 Crossbar Skip Enable Bits.**<br>These bits select Port 2 pins to be skipped by the Crossbar Decoder. Port pins used for analog, special functions or GPIO should be skipped by the Crossbar.<br>0: Corresponding P2.n pin is not skipped by the Crossbar.<br>1: Corresponding P2.n pin is skipped by the Crossbar. |
| **Note:** P2.2-P2.7 are only available on the 32-pin packages. | | |

## SFR Definition 18.24. P3: Port 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | | | | P3 |
| Type | R | R | R | R | R | R | R | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xB0; SFR Page = All Pages; Bit-Addressable

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:1 | Unused | Read = 0000000b; Write = Don't Care. | | |
| 0 | P3[0] | **Port 3 Data.**<br>Sets the Port latch logic value or reads the Port pin logic state in Port cells configured for digital I/O. | 0: Set output latch to logic LOW.<br>1: Set output latch to logic HIGH. | 0: P3.n Port pin is logic LOW.<br>1: P3.n Port pin is logic HIGH. |
| **Note:** Port P3.0 is only available on the 32-pin packages. | | | | |

## LIN Register Definition 19.8. LIN0SIZE: LIN0 Message Size Register

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

Indirect Address = 0x0B

| Bit | Name | Function |
|---|---|---|
| 7 | ENHCHK | **Checksum Selection Bit.**<br>0: Use the classic, specification 1.3 compliant checksum. Checksum covers the data bytes.<br>1: Use the enhanced, specification 2.0 compliant checksum. Checksum covers data bytes and protected identifier. |
| 6:4 | Unused | Read = 000b; Write = Don't Care |
| 3:0 | LINSIZE[3:0] | **Data Field Size.**<br>0000: 0 data bytes<br>0001: 1 data byte<br>0010: 2 data bytes<br>0011: 3 data bytes<br>0100: 4 data bytes<br>0101: 5 data bytes<br>0110: 6 data bytes<br>0111: 7 data bytes<br>1000: 8 data bytes<br>1001-1110: RESERVED<br>1111: Use the ID[1:0] bits (LIN0ID[5:4]) to determine the data length. |

### 20.5.2. Read Sequence (Master)

During a read sequence, an SMBus master reads data from a slave device. The master in this transfer will be a transmitter during the address byte, and a receiver during all data bytes. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 1 (READ). Serial data is then received from the slave on SDA while the SMBus outputs the serial clock. The slave transmits one or more bytes of serial data. An interrupt is generated after each received byte.

Software must write the ACK bit at that time to ACK or NACK the received byte. Writing a 1 to the ACK bit generates an ACK; writing a 0 generates a NACK. Software should write a 0 to the ACK bit for the last data transfer, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. The interface will switch to Master Transmitter Mode if SMB0DAT is written while an active Master Receiver. Figure 20.6 shows a typical master read sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the ‘data byte transferred’ interrupts occur **before** the ACK cycle in this mode.

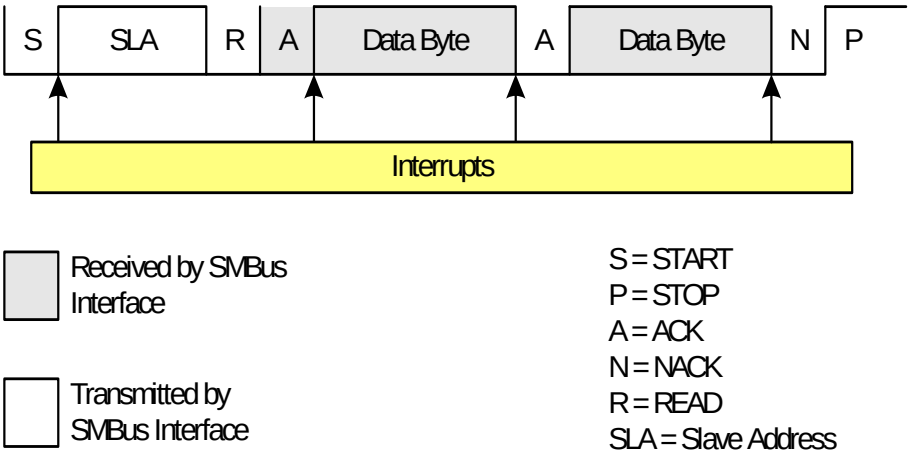


**Figure 20.6. Typical Master Read Sequence**

# 23. Timers

Each MCU includes four counter/timers: two are 16-bit counter/timers compatible with those found in the standard 8051, and two are 16-bit auto-reload timer for use with the ADC, SMBus, or for general purpose use. These timers can be used to measure time intervals, count external events and generate periodic interrupt requests. Timer 0 and Timer 1 are nearly identical and have four primary modes of operation. Timer 2 and Timer 3 offer 16-bit and split 8-bit timer functionality with auto-reload.

| Timer 0 and Timer 1 Modes | Timer 2 Modes | Timer 3 Modes |
|---|---|---|
| 13-bit counter/timer | 16-bit timer with auto-reload | 16-bit timer with auto-reload |
| 16-bit counter/timer | | |
| 8-bit counter/timer with auto-reload | Two 8-bit timers with auto-reload | Two 8-bit timers with auto-reload |
| Two 8-bit counter/timers (Timer 0 only) | | |

Timers 0 and 1 may be clocked by one of five sources, determined by the Timer Mode Select bits (T1M–T0M) and the Clock Scale bits (SCA1–SCA0). The Clock Scale bits define a pre-scaled clock from which Timer 0 and/or Timer 1 may be clocked (See SFR Definition 23.1 for pre-scaled clock selection).Timer 0/1 may then be configured to use this pre-scaled clock signal or the system clock.

Timer 2 and Timer 3 may be clocked by the system clock, the system clock divided by 12, or the external oscillator clock source divided by 8.

Timer 0 and Timer 1 may also be operated as counters. When functioning as a counter, a counter/timer register is incremented on each high-to-low transition at the selected input pin (T0 or T1). Events with a frequency of up to one-fourth the system clock frequency can be counted. The input signal need not be periodic, but it should be held at a given level for at least two full system clock cycles to ensure the level is properly sampled.

## 24.1. 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 CPS[2:0] bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 24.1.

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 24.1. 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.

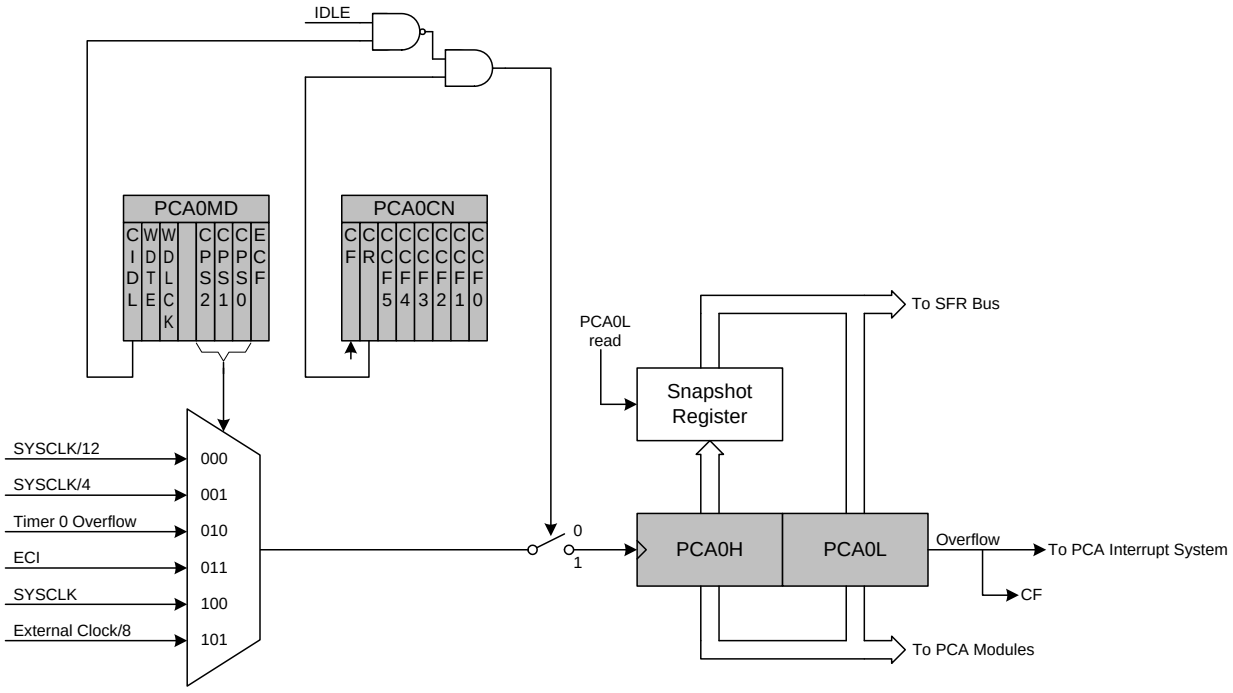


**Figure 24.2. PCA Counter/Timer Block Diagram**

## 24.2. PCA0 Interrupt Sources

Figure 24.3 shows a diagram of the PCA interrupt tree. There are five independent event flags that can be used to generate a PCA0 interrupt. They are as follows: the main PCA counter overflow flag (CF), which is set upon a 16-bit overflow of the PCA0 counter, an intermediate overflow flag (COVF), which can be set on an overflow from the 8th, 9th, 10th, or 11th bit of the PCA0 counter, and the individual flags for each PCA channel (CCF0, CCF1, CCF2, CCF3, CCF4, and CCF5), which are set according to the operation mode of that module. These event flags are always set when the trigger condition occurs. Each of these flags can be individually selected to generate a PCA0 interrupt, using the corresponding interrupt enable flag (ECF for CF, ECOV for COVF, and ECCFn for each CCFn). PCA0 interrupts must be globally enabled before any individual interrupt sources are recognized by the processor. PCA0 interrupts are globally enabled by setting the EA bit and the EPCA0 bit to logic 1.