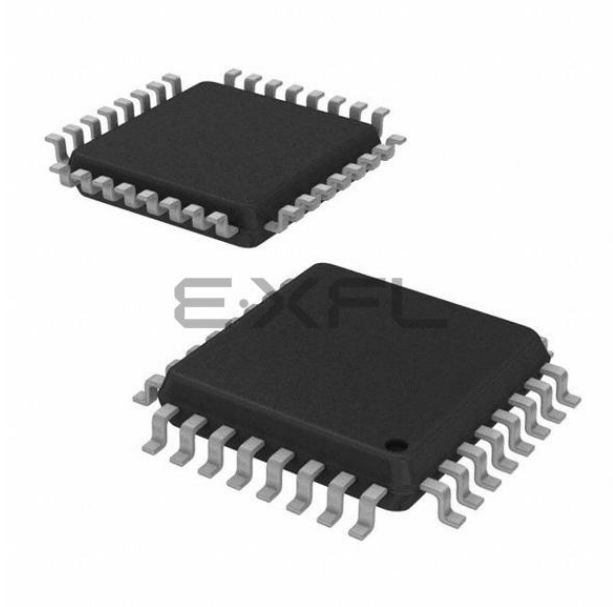

**Welcome to E-XFL.COM**

## 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 | 48 MIPS |
| Connectivity | I²C, SPI, UART/USART, USB |
| Peripherals | Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT |
| Number of I/O | 25 |
| Program Memory Size | 64KB (64K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 4.25K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 5.25V |
| Data Converters | A/D 21x10b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 32-LQFP |
| Supplier Device Package | 32-LQFP (7x7) |
| Purchase URL | https://www.e-xfl.com/product-detail/silicon-labs/c8051f381-gqr |

## 6. 10-Bit ADC (ADC0, C8051F380/1/2/3/C only)

ADC0 on the C8051F380/1/2/3/C is a 500 ksps, 10-bit successive-approximation-register (SAR) ADC with integrated track-and-hold, and a programmable window detector. The ADC is fully configurable under software control via Special Function Registers. The ADC may be configured to measure various different signals using the analog multiplexer described in Section “6.5. ADC0 Analog Multiplexer (C8051F380/1/2/3/C only)” on page 59. The voltage reference for the ADC is selected as described in Section “7. Voltage Reference Options” on page 62. The ADC0 subsystem is enabled only when the AD0EN bit in the ADC0 Control register (ADC0CN) is set to logic 1. The ADC0 subsystem is in low power shutdown when this bit is logic 0.

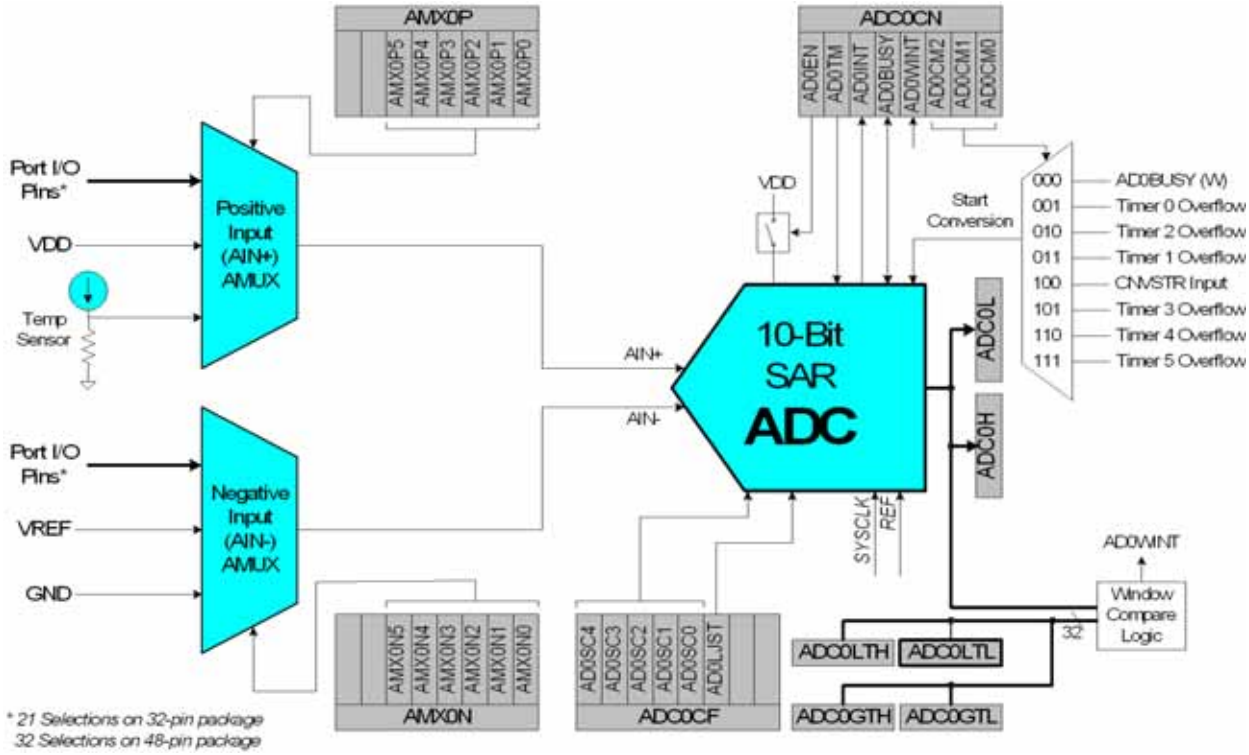


**Figure 6.1. ADC0 Functional Block Diagram**

## 6.3. Modes of Operation

ADC0 has a maximum conversion speed of 500 ksps. The ADC0 conversion clock is a divided version of the system clock, determined by the AD0SC bits in the ADC0CF register.

### 6.3.1. Starting a Conversion

A conversion can be initiated in one of several ways, depending on the programmed states of the ADC0 Start of Conversion Mode bits (AD0CM2–0) in register ADC0CN. Conversions may be initiated by one of the following:

1. Writing a 1 to the AD0BUSY bit of register ADC0CN
2. A Timer 0 overflow (i.e., timed continuous conversions)
3. A Timer 2 overflow
4. A Timer 1 overflow
5. A rising edge on the CNVSTR input signal
6. A Timer 3 overflow
7. A Timer 4 overflow
8. A Timer 5 overflow

Writing a 1 to AD0BUSY provides software control of ADC0 whereby conversions are performed "on-demand". During conversion, the AD0BUSY bit is set to logic 1 and reset to logic 0 when the conversion is complete. The falling edge of AD0BUSY triggers an interrupt (when enabled) and sets the ADC0 interrupt flag (AD0INT). Note: When polling for ADC conversion completions, the ADC0 interrupt flag (AD0INT) should be used. Converted data is available in the ADC0 data registers, ADC0H:ADC0L, when bit AD0INT is logic 1. Note that when Timer 2, 3, 4, or 5 overflows are used as the conversion source, Low Byte overflows are used if the timer is in 8-bit mode; High byte overflows are used if the timer is in 16-bit mode. See Section “26. Timers” on page 263 for timer configuration.

**Important Note About Using CNVSTR:** The CNVSTR input pin also functions as a Port I/O pin. When the CNVSTR input is used as the ADC0 conversion source, the associated pin should be skipped by the Digital Crossbar. See Section “20. Port Input/Output” on page 153 for details on Port I/O configuration.

### 6.3.2. Tracking Modes

The AD0TM bit in register ADC0CN controls the ADC0 track-and-hold mode. In its default state, the ADC0 input is continuously tracked, except when a conversion is in progress. When the AD0TM bit is logic 1, ADC0 operates in low-power track-and-hold mode. In this mode, each conversion is preceded by a tracking period of 3 SAR clocks (after the start-of-conversion signal). When the CNVSTR signal is used to initiate conversions in low-power tracking mode, ADC0 tracks only when CNVSTR is low; conversion begins on the rising edge of CNVSTR. See Figure 6.4 for track and convert timing details. Tracking can also be disabled (shutdown) when the device is in low power standby or sleep modes. Low-power track-and-hold mode is also useful when AMUX settings are frequently changed, due to the settling time requirements described in Section “6.3.3. Settling Time Requirements” on page 52.

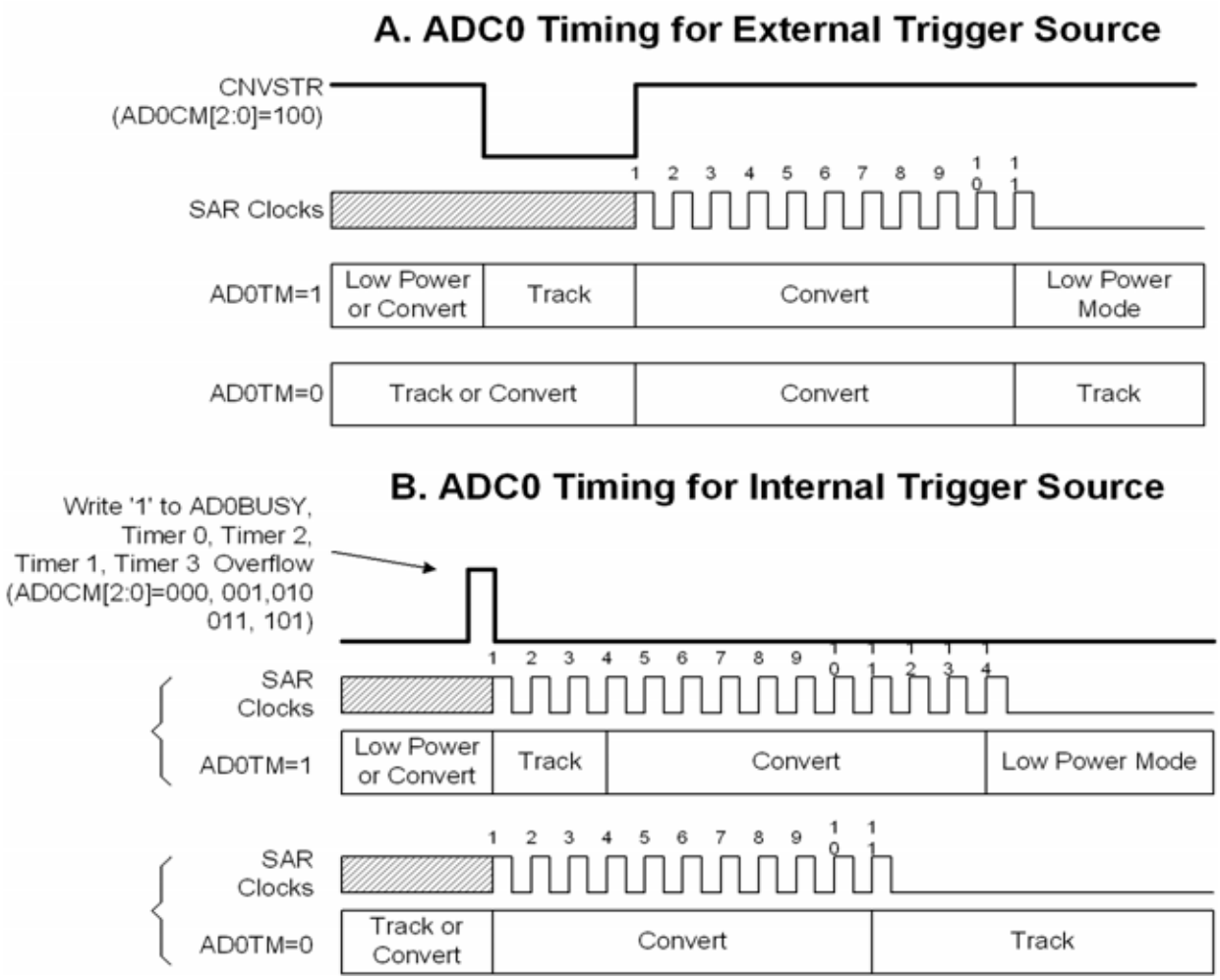


**Figure 6.4. 10-Bit ADC Track and Conversion Example Timing**

### 6.3.3. Settling Time Requirements

A minimum tracking time is required before each conversion to ensure that an accurate conversion is performed. This tracking time is determined by the AMUX0 resistance, the ADC0 sampling capacitance, any external source resistance, and the accuracy required for the conversion. Note that in low-power tracking mode, three SAR clocks are used for tracking at the start of every conversion. For most applications, these three SAR clocks will meet the minimum tracking time requirements.

Figure 6.5 shows the equivalent ADC0 input circuit. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 6.1. See Table 5.10 for ADC0 minimum settling time requirements as well as the mux impedance and sampling capacitor values.

$$t = \ln\left(\frac{2^n}{SA}\right) \times R_{TOTAL} C_{SAMPLE}$$

**Equation 6.1. ADC0 Settling Time Requirements**

Where:
*SA* is the settling accuracy, given as a fraction of an LSB (for example, 0.25 to settle within 1/4 LSB)
*t* is the required settling time in seconds
$R_{TOTAL}$ is the sum of the AMUX0 resistance and any external source resistance.
*n* is the ADC resolution in bits (10).

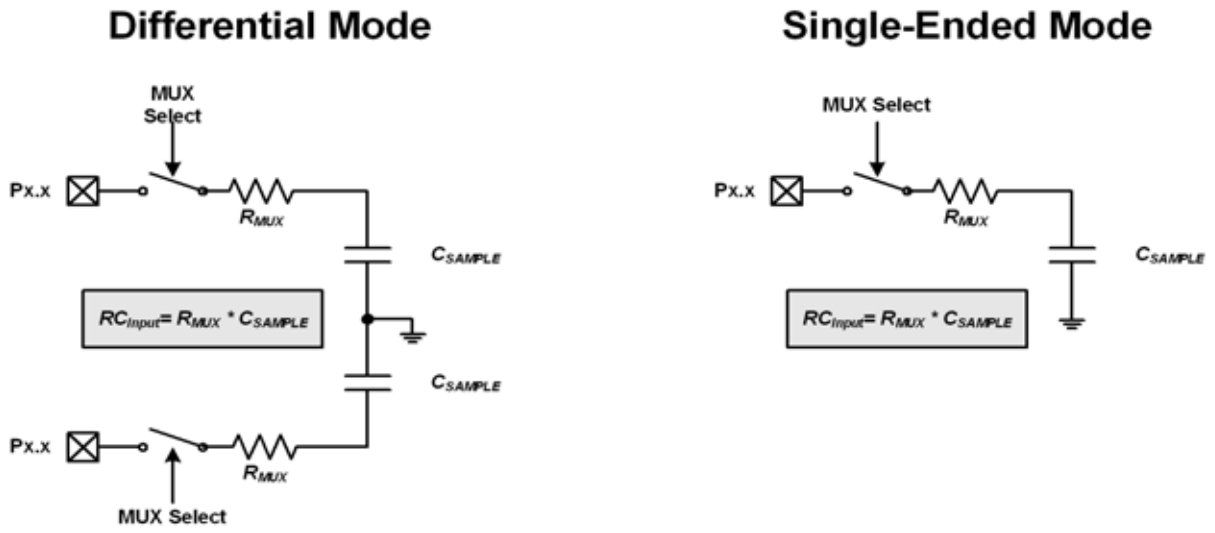


**Figure 6.5. ADC0 Equivalent Input Circuits**

With the CIP-51's maximum system clock at 48 MHz, it has a peak throughput of 48 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 | 2/4 | 3 | 3/5 | 4 | 5 | 4/6 | 6 | 8 |
|---|---|---|---|---|---|---|---|---|---|---|
| Number of Instructions | 26 | 50 | 5 | 10 | 6 | 5 | 2 | 2 | 2 | 1 |

**Programming and Debugging Support**

In-system programming of the Flash program memory and communication with on-chip debug support logic is accomplished via the Silicon Labs 2-Wire Development Interface (C2).

The on-chip debug support logic facilitates full speed in-circuit debugging, allowing the setting of hardware breakpoints, starting, stopping and single stepping through program execution (including interrupt service routines), examination of the program's call stack, and reading/writing the contents of registers and memory. This method of on-chip debugging is completely non-intrusive, requiring no RAM, Stack, timers, or other on-chip resources. C2 details can be found in Section “28. C2 Interface” on page 316.

The CIP-51 is supported by development tools from Silicon Labs and third party vendors. Silicon Labs provides an integrated development environment (IDE) including editor, debugger and programmer. The IDE's debugger and programmer interface to the CIP-51 via the C2 interface to provide fast and efficient in-system device programming and debugging. Third party macro assemblers and C compilers are also available.

## 11.1. 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.

### 11.1.1. 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 11.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

**Table 11.1. CIP-51 Instruction Set Summary**

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| **Arithmetic Operations** | | | |
| ADD A, Rn | Add register to A | 1 | 1 |
| ADD A, direct | Add direct byte to A | 2 | 2 |
| ADD A, @Ri | Add indirect RAM to A | 1 | 2 |
| ADD A, #data | Add immediate to A | 2 | 2 |
| ADDC A, Rn | Add register to A with carry | 1 | 1 |
| ADDC A, direct | Add direct byte to A with carry | 2 | 2 |
| ADDC A, @Ri | Add indirect RAM to A with carry | 1 | 2 |
| ADDC A, #data | Add immediate to A with carry | 2 | 2 |
| SUBB A, Rn | Subtract register from A with borrow | 1 | 1 |
| SUBB A, direct | Subtract direct byte from A with borrow | 2 | 2 |
| SUBB A, @Ri | Subtract indirect RAM from A with borrow | 1 | 2 |
| SUBB A, #data | Subtract immediate from A with borrow | 2 | 2 |
| INC A | Increment A | 1 | 1 |
| INC Rn | Increment register | 1 | 1 |
| INC direct | Increment direct byte | 2 | 2 |
| INC @Ri | Increment indirect RAM | 1 | 2 |
| DEC A | Decrement A | 1 | 1 |
| DEC Rn | Decrement register | 1 | 1 |
| DEC direct | Decrement direct byte | 2 | 2 |
| DEC @Ri | Decrement indirect RAM | 1 | 2 |
| INC DPTR | Increment Data Pointer | 1 | 1 |
| MUL AB | Multiply A and B | 1 | 4 |
| DIV AB | Divide A by B | 1 | 8 |
| DA A | Decimal adjust A | 1 | 1 |
| **Logical Operations** | | | |
| ANL A, Rn | AND Register to A | 1 | 1 |
| ANL A, direct | AND direct byte to A | 2 | 2 |
| ANL A, @Ri | AND indirect RAM to A | 1 | 2 |
| ANL A, #data | AND immediate to A | 2 | 2 |
| ANL direct, A | AND A to direct byte | 2 | 2 |
| ANL direct, #data | AND immediate to direct byte | 3 | 3 |
| ORL A, Rn | OR Register to A | 1 | 1 |
| ORL A, direct | OR direct byte to A | 2 | 2 |
| ORL A, @Ri | OR indirect RAM to A | 1 | 2 |
| ORL A, #data | OR immediate to A | 2 | 2 |
| ORL direct, A | OR A to direct byte | 2 | 2 |
| ORL direct, #data | OR immediate to direct byte | 3 | 3 |
| XRL A, Rn | Exclusive-OR Register to A | 1 | 1 |
| XRL A, direct | Exclusive-OR direct byte to A | 2 | 2 |
| XRL A, @Ri | Exclusive-OR indirect RAM to A | 1 | 2 |
| XRL A, #data | Exclusive-OR immediate to A | 2 | 2 |
| XRL direct, A | Exclusive-OR A to direct byte | 2 | 2 |

| Notes on Registers, Operands and Addressing Modes: |
|---|
| **Rn** - Register R0–R7 of the currently selected register bank.<br><br>**@Ri** - Data RAM location addressed indirectly through R0 or R1.<br><br>**rel** - 8-bit, signed (two's complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.<br><br>**direct** - 8-bit internal data location's address. This could be a direct-access Data RAM location (0x00–0x7F) or an SFR (0x80–0xFF).<br><br>**#data** - 8-bit constant<br><br>**#data16** - 16-bit constant<br><br>**bit** - Direct-accessed bit in Data RAM or SFR<br><br>**addr11** - 11-bit destination address used by ACALL and AJMP. The destination must be within the same 2 kB page of program memory as the first byte of the following instruction.<br><br>**addr16** - 16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8 kB program memory space.<br><br>There is one unused opcode (0xA5) that performs the same function as NOP.<br>All mnemonics copyrighted © Intel Corporation 1980. |

## 14.7. Timing

The timing parameters of the External Memory Interface can be configured to enable connection to devices having different setup and hold time requirements. The Address Setup time, Address Hold time, $\overline{RD}$ and $\overline{WR}$ strobe widths, and in multiplexed mode, the width of the ALE pulse are all programmable in units of SYSCLK periods through EMI0TC, shown in SFR Definition 14.3, and EMI0CF[1:0].

The timing for an off-chip MOVX instruction can be calculated by adding 4 SYSCLK cycles to the timing parameters defined by the EMI0TC register. Assuming non-multiplexed operation, the minimum execution time for an off-chip XRAM operation is 5 SYSCLK cycles (1 SYSCLK for $\overline{RD}$ or $\overline{WR}$ pulse + 4 SYSCLKs). For multiplexed operations, the Address Latch Enable signal will require a minimum of 2 additional SYSCLK cycles. Therefore, the minimum execution time for an off-chip XRAM operation in multiplexed mode is 7 SYSCLK cycles (2 for $\overline{ALE}$ + 1 for $\overline{RD}$ or $\overline{WR}$ + 4). The programmable setup and hold times default to the maximum delay settings after a reset. Table 14.1 lists the AC parameters for the External Memory Interface, and Figure 14.5 through Figure 14.10 show the timing diagrams for the different External Memory Interface modes and MOVX operations.

## SFR Definition 14.3. EMI0TC: External Memory TIming Control

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

SFR Address = 0x84; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:6 | EAS[1:0] | **EMIF Address Setup Time Bits.**<br>00: Address setup time = 0 SYSCLK cycles.<br>01: Address setup time = 1 SYSCLK cycle.<br>10: Address setup time = 2 SYSCLK cycles.<br>11: Address setup time = 3 SYSCLK cycles. |
| 5:2 | EWR[3:0] | **EMIF $\overline{WR}$ and $\overline{RD}$ Pulse-Width Control Bits.**<br>0000: $\overline{WR}$ and $\overline{RD}$ pulse width = 1 SYSCLK cycle.<br>0001: $\overline{WR}$ and $\overline{RD}$ pulse width = 2 SYSCLK cycles.<br>0010: $\overline{WR}$ and $\overline{RD}$ pulse width = 3 SYSCLK cycles.<br>0011: $\overline{WR}$ and $\overline{RD}$ pulse width = 4 SYSCLK cycles.<br>0100: $\overline{WR}$ and $\overline{RD}$ pulse width = 5 SYSCLK cycles.<br>0101: $\overline{WR}$ and $\overline{RD}$ pulse width = 6 SYSCLK cycles.<br>0110: $\overline{WR}$ and $\overline{RD}$ pulse width = 7 SYSCLK cycles.<br>0111: $\overline{WR}$ and $\overline{RD}$ pulse width = 8 SYSCLK cycles.<br>1000: $\overline{WR}$ and $\overline{RD}$ pulse width = 9 SYSCLK cycles.<br>1001: $\overline{WR}$ and $\overline{RD}$ pulse width = 10 SYSCLK cycles.<br>1010: $\overline{WR}$ and $\overline{RD}$ pulse width = 11 SYSCLK cycles.<br>1011: $\overline{WR}$ and $\overline{RD}$ pulse width = 12 SYSCLK cycles.<br>1100: $\overline{WR}$ and $\overline{RD}$ pulse width = 13 SYSCLK cycles.<br>1101: $\overline{WR}$ and $\overline{RD}$ pulse width = 14 SYSCLK cycles.<br>1110: $\overline{WR}$ and $\overline{RD}$ pulse width = 15 SYSCLK cycles.<br>1111: $\overline{WR}$ and $\overline{RD}$ pulse width = 16 SYSCLK cycles. |
| 1:0 | EAH[1:0] | **EMIF Address Hold Time Bits.**<br>00: Address hold time = 0 SYSCLK cycles.<br>01: Address hold time = 1 SYSCLK cycle.<br>10: Address hold time = 2 SYSCLK cycles.<br>11: Address hold time = 3 SYSCLK cycles. |

**Table 14.1. AC Parameters for External Memory Interface**

| Parameter | Description | Min* | Max* | Units |
|---|---|---|---|---|
| $T_{ACS}$ | Address/Control Setup Time | 0 | 3 x $T_{SYSCLK}$ | ns |
| $T_{ACW}$ | Address/Control Pulse Width | 1 x $T_{SYSCLK}$ | 16 x $T_{SYSCLK}$ | ns |
| $T_{ACH}$ | Address/Control Hold Time | 0 | 3 x $T_{SYSCLK}$ | ns |
| $T_{ALEH}$ | Address Latch Enable High Time | 1 x $T_{SYSCLK}$ | 4 x $T_{SYSCLK}$ | ns |
| $T_{ALEL}$ | Address Latch Enable Low Time | 1 x $T_{SYSCLK}$ | 4 x $T_{SYSCLK}$ | ns |
| $T_{WDS}$ | Write Data Setup Time | 1 x $T_{SYSCLK}$ | 19 x $T_{SYSCLK}$ | ns |
| $T_{WDH}$ | Write Data Hold Time | 0 | 3 x $T_{SYSCLK}$ | ns |
| $T_{RDS}$ | Read Data Setup Time | 20 | | ns |
| $T_{RDH}$ | Read Data Hold Time | 0 | | ns |
| **Note:** $T_{SYSCLK}$ is equal to one period of the device system clock (SYSCLK). | | | | |

# 15. 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 C8051F380/1/2/3/4/5/6/7/C'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 C8051F380/1/2/3/4/5/6/7/C. This allows the addition of new functionality while retaining compatibility with the MCS-51™ instruction set. Table 15.1 lists the SFRs implemented in the C8051F380/1/2/3/4/5/6/7/C 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 these areas will have an indeterminate effect and should be avoided. Refer to the corresponding pages of the data sheet, as indicated in Table 15.2, for a detailed description of each register.

## 15.1. 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 C8051F380/1/2/3/4/5/6/7/C devices utilize two SFR pages: 0x0, and 0xF. Most SFRs are available on both pages. SFR pages are selected using the Special Function Register Page Selection register, SFRPAGE. 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).

**Important Note:** When reading or writing SFRs that are not available on all pages within an ISR, it is recommended to save the state of the SFRPAGE register on ISR entry, and restore state on exit.

### SFR Definition 15.1. SFRPAGE: SFR Page

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

SFR Address = 0xBF; SFR Page = All Pages

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

## SFR Definition 17.2. RSTSRC: Reset Source

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | USBRSF | FERROR | C0RSEF | SWRSF | WDTRSF | MCDRSF | PORSF | PINRSF |
| Type | R/W | R | R/W | R/W | R | R/W | R/W | R |
| Reset | Varies | Varies | Varies | Varies | Varies | Varies | Varies | Varies |

SFR Address = 0xEF; SFR Page = All Pages

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7 | USBRSF | **USB Reset Flag** | Writing a 1 enables USB as a reset source. | Set to 1 if USB caused the last reset. |
| 6 | FERROR | **Flash Error Reset Flag.** | N/A | Set to 1 if Flash read/write/erase error caused the last reset. |
| 5 | C0RSEF | **Comparator0 Reset Enable and Flag.** | Writing a 1 enables Comparator0 as a reset source (active-low). | Set to 1 if Comparator0 caused the last reset. |
| 4 | SWRSF | **Software Reset Force and Flag.** | Writing a 1 forces a system reset. | Set to 1 if last reset was caused by a write to SWRSF. |
| 3 | WDTRSF | **Watchdog Timer Reset Flag.** | N/A | Set to 1 if Watchdog Timer overflow caused the last reset. |
| 2 | MCDRSF | **Missing Clock Detector Enable and Flag.** | Writing a 1 enables the Missing Clock Detector. The MCD triggers a reset if a missing clock condition is detected. | Set to 1 if Missing Clock Detector timeout caused the last reset. |
| 1 | PORSF | **Power-On / $V_{DD}$ Monitor Reset Flag, and $V_{DD}$ monitor Reset Enable.** | Writing a 1 enables the $V_{DD}$ monitor as a reset source. **Writing 1 to this bit before the $V_{DD}$ monitor is enabled and stabilized may cause a system reset.** | Set to 1 anytime a power-on or $V_{DD}$ monitor reset occurs. **When set to 1 all other RSTSRC flags are indeterminate.** |
| 0 | PINRSF | **HW Pin Reset Flag.** | N/A | Set to 1 if $\overline{\text{RST}}$ pin caused the last reset. |

**Note:** Do not use read-modify-write operations on this register

## SFR Definition 20.4. P0: Port 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0x80; SFR Page = All Pages; Bit Addressable

| Bit | Name | Description | Write | Read |
|---|---|---|---|---|
| 7:0 | P0[7:0] | **Port 0 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: P0.n Port pin is logic LOW.<br>1: P0.n Port pin is logic HIGH. |

## SFR Definition 20.5. P0MDIN: Port 0 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | P0MDIN[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Address = 0xF1; SFR Page = All Pages

| Bit | Name | Function |
|---|---|---|
| 7:0 | P0MDIN[7:0] | **Analog Configuration Bits for P0.7–P0.0 (respectively).**<br>Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled.<br>0: Corresponding P0.n pin is configured for analog mode.<br>1: Corresponding P0.n pin is not configured for analog mode. |

A Bulk or Interrupt pipe can be shut down (or Halted) by writing 1 to the SDSTL bit (EINCSRL.4). While SDSTL = 1, hardware will respond to all IN requests with a STALL condition. Each time hardware generates a STALL condition, an interrupt will be generated and the STSTL bit (EINCSRL.5) set to 1. The STSTL bit must be reset to 0 by firmware.

Hardware will automatically reset INPRDY to 0 when a packet slot is open in the endpoint FIFO. Note that if double buffering is enabled for the target endpoint, it is possible for firmware to load two packets into the IN FIFO at a time. In this case, hardware will reset INPRDY to 0 immediately after firmware loads the first packet into the FIFO and sets INPRDY to 1. An interrupt will not be generated in this case; an interrupt will only be generated when a data packet is transmitted.

When firmware writes 1 to the FCDT bit (EINCSRH.3), the data toggle for each IN packet will be toggled continuously, regardless of the handshake received from the host. This feature is typically used by Interrupt endpoints functioning as rate feedback communication for Isochronous endpoints. When FCDT = 0, the data toggle bit will only be toggled when an ACK is sent from the host in response to an IN packet.

### 21.12.2. Endpoints1-3 IN Isochronous Mode

When the ISO bit (EINCSRH.6) is set to 1, the target endpoint operates in Isochronous (ISO) mode. Once an endpoint has been configured for ISO IN mode, the host will send one IN token (data request) per frame; the location of data within each frame may vary. Because of this, it is recommended that double buffering be enabled for ISO IN endpoints.

Hardware will automatically reset INPRDY (EINCSRL.0) to 0 when a packet slot is open in the endpoint FIFO. Note that if double buffering is enabled for the target endpoint, it is possible for firmware to load two packets into the IN FIFO at a time. In this case, hardware will reset INPRDY to 0 immediately after firmware loads the first packet into the FIFO and sets INPRDY to 1. An interrupt will not be generated in this case; an interrupt will only be generated when a data packet is transmitted.

If there is not a data packet ready in the endpoint FIFO when USB0 receives an IN token from the host, USB0 will transmit a zero-length data packet and set the UNDRUN bit (EINCSRL.2) to 1.

The ISO Update feature (see Section 21.7) can be useful in starting a double buffered ISO IN endpoint. If the host has already set up the ISO IN pipe (has begun transmitting IN tokens) when firmware writes the first data packet to the endpoint FIFO, the next IN token may arrive and the first data packet sent before firmware has written the second (double buffered) data packet to the FIFO. The ISO Update feature ensures that any data packet written to the endpoint FIFO will not be transmitted during the current frame; the packet will only be sent after a SOF signal has been received.

## 21.13. Controlling Endpoints1-3 OUT

Endpoints1-3 OUT are managed via USB registers EOUTCSRL and EOUTCSRH. All OUT endpoints can be used for Interrupt, Bulk, or Isochronous transfers. Isochronous (ISO) mode is enabled by writing 1 to the ISO bit in register EOUTCSRH. Bulk and Interrupt transfers are handled identically by hardware.

An Endpoint1-3 OUT interrupt may be generated by the following:

1. Hardware sets the OPRDY bit (EINCSRL.0) to 1.
2. Hardware generates a STALL condition.

### 21.13.1. Endpoints1-3 OUT Interrupt or Bulk Mode

When the ISO bit (EOUTCSRH.6) = 0 the target endpoint operates in Bulk or Interrupt mode. Once an endpoint has been configured to operate in Bulk/Interrupt OUT mode (typically following an Endpoint0 SET_INTERFACE command), hardware will set the OPRDY bit (EOUTCSRL.0) to 1 and generate an interrupt upon reception of an OUT token and data packet. The number of bytes in the current OUT data packet (the packet ready to be unloaded from the FIFO) is given in the EOUTCNTH and EOUTCNTL registers. In response to this interrupt, firmware should unload the data packet from the OUT FIFO and reset the OPRDY bit to 0.

A Bulk or Interrupt pipe can be shut down (or Halted) by writing 1 to the SDSTL bit (EOUTCSRL.5). While SDSTL = 1, hardware will respond to all OUT requests with a STALL condition. Each time hardware generates a STALL condition, an interrupt will be generated and the STSTL bit (EOUTCSRL.6) set to 1. The STSTL bit must be reset to 0 by firmware.

Hardware will automatically set OPRDY when a packet is ready in the OUT FIFO. Note that if double buffering is enabled for the target endpoint, it is possible for two packets to be ready in the OUT FIFO at a time. In this case, hardware will set OPRDY to 1 immediately after firmware unloads the first packet and resets OPRDY to 0. A second interrupt will be generated in this case.

### 21.13.2. Endpoints1-3 OUT Isochronous Mode

When the ISO bit (EOUTCSRH.6) is set to 1, the target endpoint operates in Isochronous (ISO) mode. Once an endpoint has been configured for ISO OUT mode, the host will send exactly one data per USB frame; the location of the data packet within each frame may vary, however. Because of this, it is recommended that double buffering be enabled for ISO OUT endpoints.

Each time a data packet is received, hardware will load the received data packet into the endpoint FIFO, set the OPRDY bit (EOUTCSRL.0) to 1, and generate an interrupt (if enabled). Firmware would typically use this interrupt to unload the data packet from the endpoint FIFO and reset the OPRDY bit to 0.

If a data packet is received when there is no room in the endpoint FIFO, an interrupt will be generated and the OVRUN bit (EOUTCSRL.2) set to 1. If USB0 receives an ISO data packet with a CRC error, the data packet will be loaded into the endpoint FIFO, OPRDY will be set to 1, an interrupt (if enabled) will be generated, and the DATAERR bit (EOUTCSRL.3) will be set to 1. Software should check the DATAERR bit each time a data packet is unloaded from an ISO OUT endpoint FIFO.

## SFR Definition 23.1. SCON0: Serial Port 0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | S0MODE | - | MCE0 | REN0 | TB80 | RB80 | TI0 | RI0 |
| Type | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x98; SFR Page = All Pages; Bit-Addressable

| Bit | Name | Function |
|---|---|---|
| 7 | S0MODE | **Serial Port 0 Operation Mode.**<br>Selects the UART0 Operation Mode.<br>0: 8-bit UART with Variable Baud Rate.<br>1: 9-bit UART with Variable Baud Rate. |
| 6 | Unused | Read = 1b, Write = don't care. |
| 5 | MCE0 | **Multiprocessor Communication Enable.**<br>The function of this bit is dependent on the Serial Port 0 Operation Mode:<br>**Mode 0: Checks for valid stop bit.**<br>0: Logic level of stop bit is ignored.<br>1: RI0 will only be activated if stop bit is logic level 1.<br>**Mode 1: Multiprocessor Communications Enable.**<br>0: Logic level of ninth bit is ignored.<br>1: RI0 is set and an interrupt is generated only when the ninth bit is logic 1. |
| 4 | REN0 | **Receive Enable.**<br>0: UART0 reception disabled.<br>1: UART0 reception enabled. |
| 3 | TB80 | **Ninth Transmission Bit.**<br>The logic level of this bit will be sent as the ninth transmission bit in 9-bit UART Mode (Mode 1). Unused in 8-bit mode (Mode 0). |
| 2 | RB80 | **Ninth Receive Bit.**<br>RB80 is assigned the value of the STOP bit in Mode 0; it is assigned the value of the 9th data bit in Mode 1. |
| 1 | TI0 | **Transmit Interrupt Flag.**<br>Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in 8-bit UART Mode, or at the beginning of the STOP bit in 9-bit UART Mode). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software. |
| 0 | RI0 | **Receive Interrupt Flag.**<br>Set to 1 by hardware when a byte of data has been received by UART0 (set at the STOP bit sampling time). When the UART0 interrupt is enabled, setting this bit to 1 causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by software. |

## SFR Definition 26.2. CKCON1: Clock Control 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | | | | | T5MH | T5ML | T4MH | T4ML |
| Type | R | R | R | R | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0xE4; SFR Page = F

| Bit | Name | Function |
|---|---|---|
| 7:4 | Unused | Read = 0000b; Write = don't care |
| 3 | T5MH | **Timer 5 High Byte Clock Select.**<br>Selects the clock supplied to the Timer 5 high byte (split 8-bit timer mode only).<br>0: Timer 5 high byte uses the clock defined by the T5XCLK bit in TMR5CN.<br>1: Timer 5 high byte uses the system clock. |
| 2 | T5ML | **Timer 5 Low Byte Clock Select.**<br>Selects the clock supplied to Timer 5. Selects the clock supplied to the lower 8-bit timer in split 8-bit timer mode.<br>0: Timer 5 low byte uses the clock defined by the T5XCLK bit in TMR5CN.<br>1: Timer 5 low byte uses the system clock. |
| 1 | T4MH | **Timer 4 High Byte Clock Select.**<br>Selects the clock supplied to the Timer 4 high byte (split 8-bit timer mode only).<br>0: Timer 4 high byte uses the clock defined by the T4XCLK bit in TMR4CN.<br>1: Timer 4 high byte uses the system clock. |
| 0 | T4ML | **Timer 4 Low Byte Clock Select.**<br>Selects the clock supplied to Timer 4. If Timer 4 is configured in split 8-bit timer mode, this bit selects the clock supplied to the lower 8-bit timer.<br>0: Timer 4 low byte uses the clock defined by the T4XCLK bit in TMR4CN.<br>1: Timer 4 low byte uses the system clock. |

## SFR Definition 26.15. 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.**<br>TMR3RLL holds the low byte of the reload value for Timer 3. |

## SFR Definition 26.16. 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.**<br>TMR3RLH holds the high byte of the reload value for Timer 3. |

## SFR Definition 26.17. TMR3L: Timer 3 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3L[7:0] | | | | | | | |
| Type | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Address = 0x94; SFR Page = 0

| Bit | Name | Function |
|---|---|---|
| 7:0 | TMR3L[7:0] | **Timer 3 Low Byte.**<br>In 16-bit mode, the TMR3L register contains the low byte of the 16-bit Timer 3. In 8-bit mode, TMR3L contains the 8-bit low byte timer value. |

## C2 Register Definition 28.2. DEVICEID: C2 Device ID

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

C2 Address: 0x00

| Bit | Name | Function |
|---|---|---|
| 7:0 | DEVICEID[7:0] | **Device ID.**<br>This read-only register returns the 8-bit device ID: 0x28 (C8051F380/1/2/3/4/5/6/7/C). |

## C2 Register Definition 28.3. REVID: C2 Revision ID

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

C2 Address: 0x01

| Bit | Name | Function |
|---|---|---|
| 7:0 | REVID[7:0] | **Revision ID.**<br>This read-only register returns the 8-bit revision ID. For example: 0x00 = Revision A. |