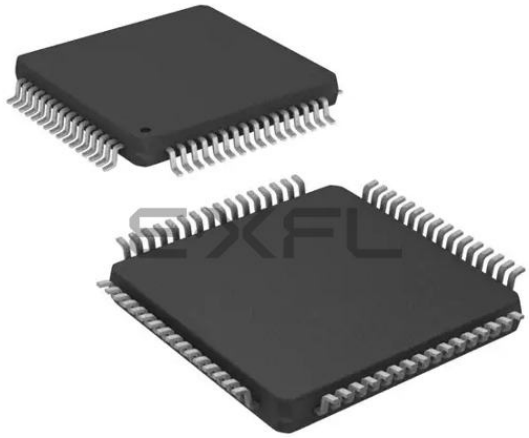


**Welcome to E-XFL.COM**

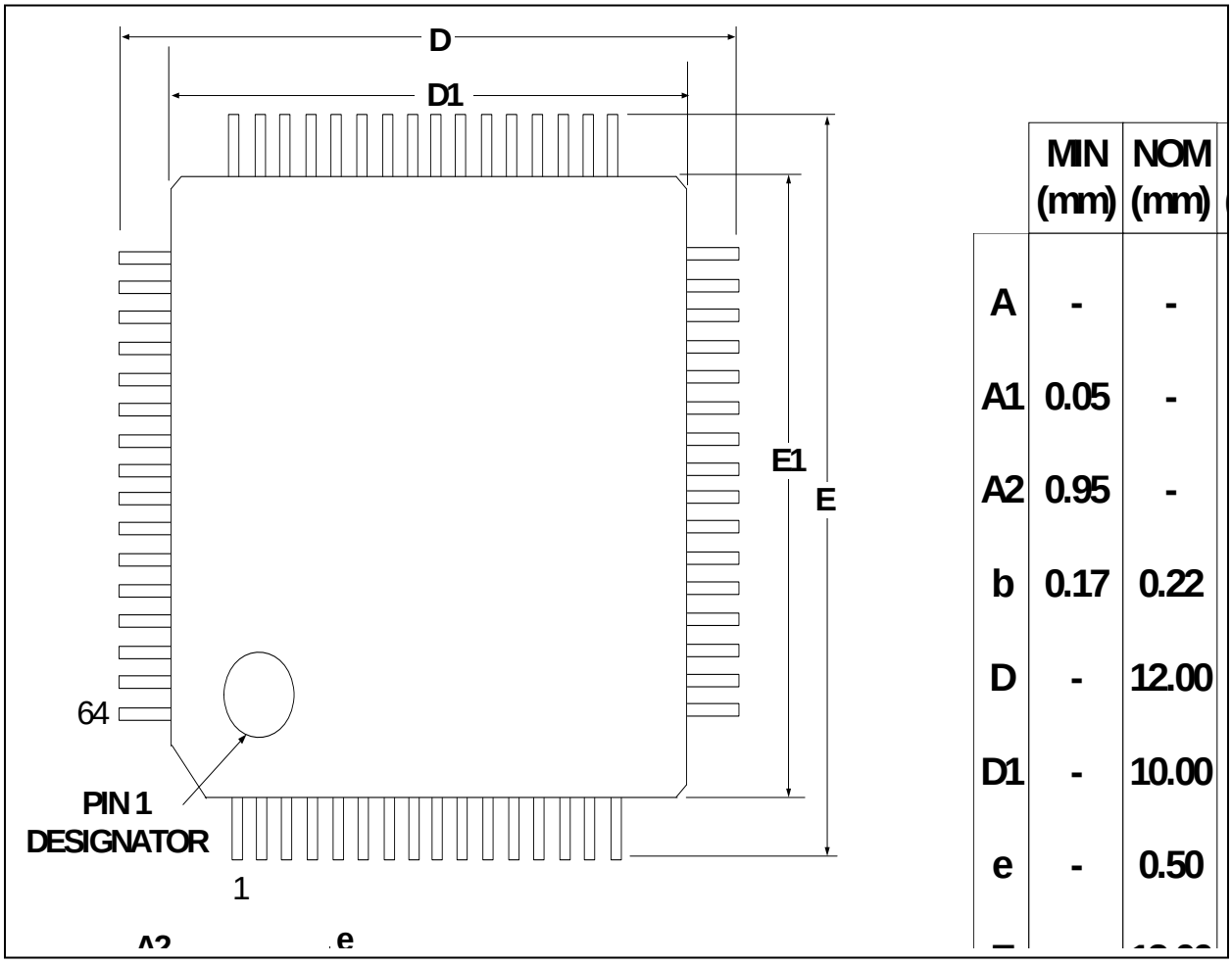
## What is "Embedded - Microcontrollers"?

"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 | Not For New Designs |
| Core Processor | 8051 |
| Core Size | 8-Bit |
| Speed | 20MHz |
| Connectivity | SMBus (2-Wire/I²C), SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT |
| Number of I/O | 32 |
| Program Memory Size | 32KB (32K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 256 x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 3.6V |
| Data Converters | A/D 8x12b; D/A 2x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-TQFP |
| Supplier Device Package | 64-TQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/silicon-labs/c8051f000-gqr |

**Figure 4.2. TQFP-64 Package Drawing**




| | MIN (mm) | NOM (mm) |
|---|---|---|
| A | - | - |
| A1 | 0.05 | - |
| A2 | 0.95 | - |
| b | 0.17 | 0.22 |
| D | - | 12.00 |
| D1 | - | 10.00 |
| e | - | 0.50 |

## 6.3. ADC Programmable Window Detector

The ADC programmable window detector is very useful in many applications. It continuously compares the ADC output to user-programmed limits and notifies the system when an out-of-band condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (ADWINT in ADC0CN) can also be used in polled mode. The high and low bytes of the reference words are loaded into the ADC Greater-Than and ADC Less-Than registers (ADC0GTH, ADC0GTL, ADC0LTH, and ADC0LTL). Figure 6.14 and Figure 6.15 show example comparisons for reference. Notice that the window detector flag can be asserted when the measured data is inside or outside the user-programmed limits, depending on the programming of the ADC0GTx and ADC0LTx registers.

**Figure 6.10. ADC0GTH: ADC Greater-Than Data High Byte Register (C8051F01x)**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 11111111 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xC5 |

Bits7-0:
The high byte of the ADC Greater-Than Data Word.

**Figure 6.11. ADC0GTL: ADC Greater-Than Data Low Byte Register (C8051F01x)**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 11111111 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xC4 |

Bits7-0:
The low byte of the ADC Greater-Than Data Word.
Definition:
ADC Greater-Than Data Word = ADC0GTH:ADC0GTL

**Figure 6.12. ADC0LTH: ADC Less-Than Data High Byte Register (C8051F01x)**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xC7 |

Bits7-0:
The high byte of the ADC Less-Than Data Word.

**Figure 6.13. ADC0LTL: ADC Less-Than Data Low Byte Register (C8051F01x)**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xC6 |

Bits7-0:
These bits are the low byte of the ADC Less-Than Data Word.

Definition:
ADC Less-Than Data Word = ADC0LTH:ADC0LTL

# 7. DACs, 12 BIT VOLTAGE MODE

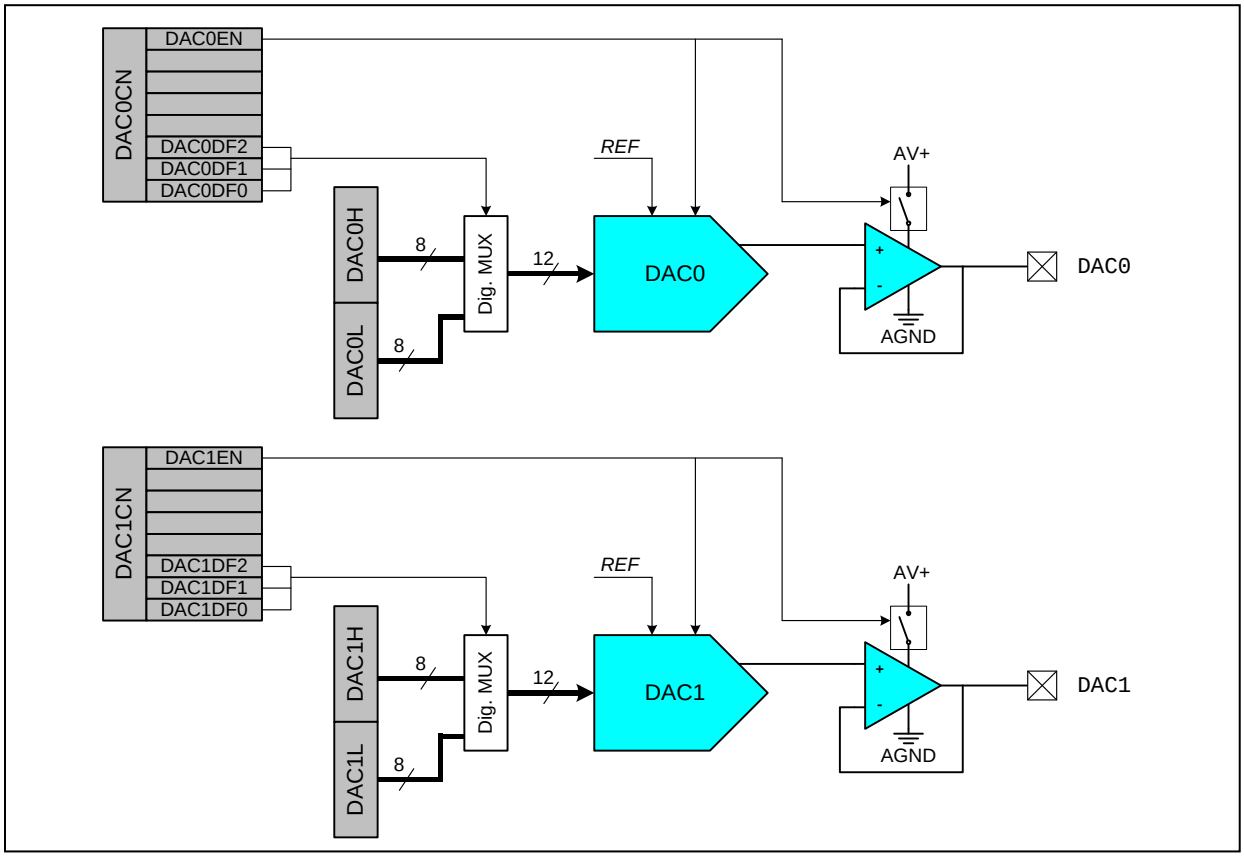
The C8051F000 MCU family has two 12-bit voltage-mode Digital to Analog Converters. Each DAC has an output swing of 0V to VREF-1LSB for a corresponding input code range of 0x000 to 0xFFF. Using DAC0 as an example, the 12-bit data word is written to the low byte (DAC0L) and high byte (DAC0H) data registers. Data is latched into DAC0 after a write to the corresponding DAC0H register, **so the write sequence should be DAC0L followed by DAC0H** if the full 12-bit resolution is required. The DAC can be used in 8-bit mode by initializing DAC0L to the desired value (typically 0x00), and writing data to only DAC0H with the data shifted to the left. DAC0 Control Register (DAC0CN) provides a means to enable/disable DAC0 and to modify its input data formatting.

The DAC0 enable/disable function is controlled by the DAC0EN bit (DAC0CN.7). Writing a 1 to DAC0EN enables DAC0 while writing a 0 to DAC0EN disables DAC0. While disabled, the output of DAC0 is maintained in a high-impedance state, and the DAC0 supply current falls to 1μA or less. Also, the Bias Enable bit (BIASE) in the REF0CN register (see Figure 9.2) must be set to 1 in order to supply bias to DAC0. The voltage reference for DAC0 must also be set properly (see Section 9).

In some instances, input data should be shifted prior to a DAC0 write operation to properly justify data within the DAC input registers. This action would typically require one or more load and shift operations, adding software overhead and slowing DAC throughput. To alleviate this problem, the data-formatting feature provides a means for the user to program the orientation of the DAC0 data word within data registers DAC0H and DAC0L. The three DAC0DF bits (DAC0CN.[2:0]) allow the user to specify one of five data word orientations as shown in the DAC0CN register definition.

DAC1 is functionally the same as DAC0 described above. The electrical specifications for both DAC0 and DAC1 are given in Table 7.1.

**Figure 7.1. DAC Functional Block Diagram**



## Table 7.1. DAC Electrical Characteristics

VDD = 3.0V, AV+ = 3.0V, REF = 2.40V (REFBE=0), No Output Load unless otherwise specified.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| **STATIC PERFORMANCE** | | | | | |
| Resolution | | | 12 | | bits |
| Integral Nonlinearity | For Data Word Range 0x014 to 0xFEB | | ±2 | | LSB |
| Differential Nonlinearity | Guaranteed Monotonic (codes 0x014 to 0xFEB) | | | ±1 | LSB |
| Output Noise | No Output Filter<br>100kHz Output Filter<br>10kHz Output Filter | | 250<br>128<br>41 | | µVrms |
| Offset Error | Data Word = 0x014 | | ±3 | ±30 | mV |
| Offset Tempco | | | 6 | | ppm/°C |
| Full-Scale Error | | | ±20 | ±60 | mV |
| Full-Scale Error Tempco | | | 10 | | ppm/°C |
| VDD Power-Supply Rejection Ratio | | | -60 | | dB |
| Output Impedance in Shutdown Mode | DACnEN=0 | | 100 | | kΩ |
| Output Current | | | ±300 | | µA |
| Output Short Circuit Current | Data Word = 0xFFF | | 15 | | mA |
| **DYNAMIC PERFORMANCE** | | | | | |
| Voltage Output Slew Rate | Load = 40pF | | 0.44 | | V/µs |
| Output Settling Time To ½ LSB | Load = 40pF, Output swing from code 0xFFF to 0x014 | | 10 | | µs |
| Output Voltage Swing | | 0 | | REF-1LSB | V |
| Startup Time | DAC Enable asserted | | 10 | | µs |
| **ANALOG OUTPUTS** | | | | | |
| Load Regulation | $I_L$ = 0.01mA to 0.3mA at code 0xFFF | | 60 | | ppm |
| **CURRENT CONSUMPTION (each DAC)** | | | | | |
| Power Supply Current (AV+ supplied to DAC) | Data Word = 0x7FF | | 110 | 400 | µA |

**Figure 9.2. REF0CN: Reference Control Register**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| - | - | - | - | - | TEMPE | BIASE | REFBE | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xD1 |

Bits7-3: UNUSED. Read = 00000b; Write = don't care
Bit2: TEMPE: Temperature Sensor Enable Bit
0: Internal Temperature Sensor Off.
1: Internal Temperature Sensor On.
Bit1: BIASE: Bias Enable Bit for ADC and DAC's
0: Internal Bias Off.
1: Internal Bias On (required for use of ADC or DAC's).
Bit0: REFBE: Internal Voltage Reference Buffer Enable Bit
0: Internal Reference Buffer Off. System reference can be driven from external source on VREF pin.
1: Internal Reference Buffer On. System reference provided by internal voltage reference.

**Table 9.1. Reference Electrical Characteristics**

VDD = 3.0V, AV+ = 3.0V, -40°C to +85°C unless otherwise specified.

| PARAMETER | CONDITIONS | MIN | TYP | MAX | UNITS |
|---|---|---|---|---|---|
| **INTERNAL REFERENCE (REFBE = 1)** | | | | | |
| Output Voltage | 25°C ambient | 2.34 | 2.43 | 2.50 | V |
| VREF Short Circuit Current | | | | 30 | mA |
| VREF Power Supply Current (supplied by AV+) | | | 50 | | µA |
| VREF Temperature Coefficient | | | 15 | | ppm/°C |
| Load Regulation | Load = (0-to-200µA) to AGND (Note 1) | | 0.5 | | ppm/µA |
| VREF Turn-on Time1 | 4.7µF tantalum, 0.1µF ceramic bypass | | 2 | | ms |
| VREF Turn-on Time2 | 0.1µF ceramic bypass | | 20 | | µs |
| VREF Turn-on Time3 | no bypass cap | | 10 | | µs |
| **EXTERNAL REFERENCE (REFBE = 0)** | | | | | |
| Input Voltage Range | | 1.00 | | (AV+) – 0.3V | V |
| Input Current | | | 0 | 1 | µA |

Note 1: The reference can only source current. When driving an external load, it is recommended to add a load resistor to AGND.

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| RR A | Rotate A right | 1 | 1 |
| RRC A | Rotate A right through carry | 1 | 1 |
| SWAP A | Swap nibbles of A | 1 | 1 |
| **DATA TRANSFER** | | | |
| MOV A,Rn | Move register to A | 1 | 1 |
| MOV A,direct | Move direct byte to A | 2 | 2 |
| MOV A,@Ri | Move indirect RAM to A | 1 | 2 |
| MOV A,#data | Move immediate to A | 2 | 2 |
| MOV Rn,A | Move A to register | 1 | 1 |
| MOV Rn,direct | Move direct byte to register | 2 | 2 |
| MOV Rn,#data | Move immediate to register | 2 | 2 |
| MOV direct,A | Move A to direct byte | 2 | 2 |
| MOV direct,Rn | Move register to direct byte | 2 | 2 |
| MOV direct,direct | Move direct byte to direct | 3 | 3 |
| MOV direct,@Ri | Move indirect RAM to direct byte | 2 | 2 |
| MOV direct,#data | Move immediate to direct byte | 3 | 3 |
| MOV @Ri,A | Move A to indirect RAM | 1 | 2 |
| MOV @Ri,direct | Move direct byte to indirect RAM | 2 | 2 |
| MOV @Ri,#data | Move immediate to indirect RAM | 2 | 2 |
| MOV DPTR,#data16 | Load data pointer with 16-bit constant | 3 | 3 |
| MOVC A,@A+DPTR | Move code byte relative DPTR to A | 1 | 3 |
| MOVC A,@A+PC | Move code byte relative PC to A | 1 | 3 |
| MOVX A,@Ri | Move external data (8-bit address) to A | 1 | 3 |
| MOVX @Ri,A | Move A to external data (8-bit address) | 1 | 3 |
| MOVX A,@DPTR | Move external data (16-bit address) to A | 1 | 3 |
| MOVX @DPTR,A | Move A to external data (16-bit address) | 1 | 3 |
| PUSH direct | Push direct byte onto stack | 2 | 2 |
| POP direct | Pop direct byte from stack | 2 | 2 |
| XCH A,Rn | Exchange register with A | 1 | 1 |
| XCH A,direct | Exchange direct byte with A | 2 | 2 |
| XCH A,@Ri | Exchange indirect RAM with A | 1 | 2 |
| XCHD A,@Ri | Exchange low nibble of indirect RAM with A | 1 | 2 |
| **BOOLEAN MANIPULATION** | | | |
| CLR C | Clear carry | 1 | 1 |
| CLR bit | Clear direct bit | 2 | 2 |
| SETB C | Set carry | 1 | 1 |
| SETB bit | Set direct bit | 2 | 2 |
| CPL C | Complement carry | 1 | 1 |
| CPL bit | Complement direct bit | 2 | 2 |
| ANL C,bit | AND direct bit to carry | 2 | 2 |
| ANL C,/bit | AND complement of direct bit to carry | 2 | 2 |
| ORL C,bit | OR direct bit to carry | 2 | 2 |
| ORL C,/bit | OR complement of direct bit to carry | 2 | 2 |
| MOV C,bit | Move direct bit to carry | 2 | 2 |
| MOV bit,C | Move carry to direct bit | 2 | 2 |
| JC rel | Jump if carry is set | 2 | 2/3 |
| JNC rel | Jump if carry not set | 2 | 2/3 |
| JB bit,rel | Jump if direct bit is set | 3 | 3/4 |
| JNB bit,rel | Jump if direct bit is not set | 3 | 3/4 |
| JBC bit,rel | Jump if direct bit is set and clear bit | 3 | 3/4 |

## 10.3. 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 CIP-51's resources and peripherals. The CIP-51 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 MCU. This allows the addition of new functionality while retaining compatibility with the MCS-51™ instruction set. Table 10.3 lists the SFRs implemented in the CIP-51 System Controller.

The SFR registers are accessed any time 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, P1, SCON, 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 datasheet, as indicated in Table 10.3, for a detailed description of each register.

**Table 10.2. Special Function Register Memory Map**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| F8 | SPI0CN | PCA0H | PCA0CPH0 | PCA0CPH1 | PCA0CPH2 | PCA0CPH3 | PCA0CPH4 | WDTCN |
| F0 | B | | | | | | EIP1 | EIP2 |
| E8 | ADC0CN | PCA0L | PCA0CPL0 | PCA0CPL1 | PCA0CPL2 | PCA0CPL3 | PCA0CPL4 | RSTSRC |
| E0 | ACC | XBR0 | XBR1 | XBR2 | | | EIE1 | EIE2 |
| D8 | PCA0CN | PCA0MD | PCA0CPM0 | PCA0CPM1 | PCA0CPM2 | PCA0CPM3 | PCA0CPM4 | |
| D0 | PSW | REF0CN | DAC0L | DAC0H | DAC0CN | DAC1L | DAC1H | DAC1CN |
| C8 | T2CON | | RCAP2L | RCAP2H | TL2 | TH2 | | SMB0CR |
| C0 | SMB0CN | SMB0STA | SMB0DAT | SMB0ADR | ADC0GTL | ADC0GTH | ADC0LTL | ADC0LTH |
| B8 | IP | | AMX0CF | AMX0SL | ADC0CF | | ADC0L | ADC0H |
| B0 | P3 | OSCXCN | OSCICN | | | | FLSCL | FLACL*** |
| A8 | IE | | | | | PRT1IF | | EMI0CN*** |
| A0 | P2 | | | | PRT0CF | PRT1CF | PRT2CF | PRT3CF |
| 98 | SCON | SBUF | SPI0CFG | SPI0DAT | | SPI0CKR | CPT0CN | CPT1CN |
| 90 | P1 | TMR3CN | TMR3RLL | TMR3RLH | TMR3L | TMR3H | | |
| 88 | TCON | TMOD | TL0 | TL1 | TH0 | TH1 | CKCON | PSCTL |
| 80 | P0 | SP | DPL | DPH | | | | PCON |
| | 0(8) | 1(9) | 2(A) | 3(B) | 4(C) | 5(D) | 6(E) | 7(F) |

Bit Addressable

**Table 10.3. Special Function Registers**

SFRs are listed in alphabetical order. All undefined SFR locations are reserved.
* Refers to a register in the C8051F000/1/2/5/6/7 only.
** Refers to a register in the C8051F010/1/2/5/6/7 only.
*** Refers to a register in the C8051F005/06/07/15/16/17 only.

| Address | Register | Description | Page No. |
|---|---|---|---|
| 0xE0 | ACC | Accumulator | 76 |
| 0xBC | ADC0CF | ADC Configuration | 33*, 42** |
| 0xE8 | ADC0CN | ADC Control | 34*, 45** |
| 0xC5 | ADC0GTH | ADC Greater-Than Data Word (High Byte) | 36*, 47** |
| 0xC4 | ADC0GTL | ADC Greater-Than Data Word (Low Byte) | 36*, 47** |
| 0xBF | ADC0H | ADC Data Word (High Byte) | 35*, 46** |
| 0xBE | ADC0L | ADC Data Word (Low Byte) | 35*, 46** |

### 10.3.1. Register Descriptions

Following are descriptions of SFRs related to the operation of the CIP-51 System Controller. Reserved bits should not be set to logic l. Future product versions may use these bits to implement new features in which case the reset value of the bit will be logic 0, selecting the feature's default state. Detailed descriptions of the remaining SFRs are included in the sections of the datasheet associated with their corresponding system function.

**Figure 10.3. SP: Stack Pointer**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000111 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0x81 |

Bits 7-0: SP: Stack Pointer.
The stack pointer holds the location of the top of the stack. The stack pointer is incremented before every PUSH operation. The SP register defaults to 0x07 after reset.

**Figure 10.4. DPL: Data Pointer Low Byte**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0x82 |

Bits 7-0: DPL: Data Pointer Low.
The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed RAM and Flash Memory.

**Figure 10.5. DPH: Data Pointer High Byte**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0x83 |

Bits 7-0: DPH: Data Pointer High.
The DPH register is the high byte of the 16-bit DPTR. DPTR is used to access indirectly addressed RAM and Flash Memory.

## 10.4. INTERRUPT HANDLER

The CIP-51 includes an extended interrupt system supporting a total of 22 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-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.

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.

### 10.4.1. MCU Interrupt Sources and Vectors

The MCUs allocate 12 interrupt sources to on-chip peripherals. Up to 10 additional external interrupt sources are available depending on the I/O pin configuration of the device. 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 10.4. 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).

### 10.4.2. External Interrupts

Two of the external interrupt sources (/INT0 and /INT1) are configurable as active-low level-sensitive or active-low edge-sensitive inputs depending on the setting of IT0 (TCON.0) and IT1 (TCON.2). IE0 (TCON.1) and IE1 (TCON.3) serve as the interrupt-pending flag for the /INT0 and /INT1 external interrupts, respectively. If an /INT0 or /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 follows the state of the external interrupt's input pin. 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.

The remaining four external interrupts (External Interrupts 4-7) are active-low, edge-sensitive inputs. The interrupt-pending flags for these interrupts are in the Port 1 Interrupt Flag Register shown in Figure 15.10.

prohibited from reading locations in the lower partition using the MOVC instruction. (Executing a MOVC instruction from the upper partition with a source address in the lower partition will always return a data value of 0x00.) Software running in the lower partition can access locations in both the upper and lower partition without restriction.

The Value-added firmware should be placed in the lower partition. On reset, control is passed to the value-added firmware via the reset vector. Once the value-added firmware completes its initial execution, it branches to a predetermined location in the upper partition. If entry points are published, software running in the upper partition may execute program code in the lower partition, but it cannot read the contents of the lower partition. Parameters may be passed to the program code running in the lower partition either through the typical method of placing them on the stack or in registers before the call or by placing them in prescribed memory locations in the upper partition.

The SRL address is specified using the contents of the Flash Access Register. The 16-bit SRL address is calculated as 0xNN00, where NN is the contents of the SRL Security Register. Thus, the SRL can be located on 256-byte boundaries anywhere in program memory space. However, the 512-byte erase sector size essentially requires that a 512 boundary be used. The contents of a non-initialized SRL security byte is 0x00, thereby setting the SRL address to 0x0000 and allowing read access to all locations in program memory space by default.

**Figure 11.3. FLACL: Flash Access Limit (C8051F005/06/07/15/16/17 only)**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xB7 |

Bits 7-0: FLACL: Flash Access Limit.
This register holds the high byte of the 16-bit program memory read/write/erase limit address. The entire 16-bit access limit address value is calculated as 0xNN00 where NN is replaced by contents of FLACL. A write to this register sets the Flash Access Limit. **This register can only be written once after any reset. Any subsequent writes are ignored until the next reset.**

**Figure 13.4. RSTSRC: Reset Source Register**

| R | R/W | R/W | R/W | R | R | R/W | R | Reset Value |
|---|---|---|---|---|---|---|---|---|
| JTAGRST | CNVRSEF | C0RSEF | SWRSEF | WDTRSF | MCDRSF | PORSF | PINRSF | xxxxxxxx |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xEF |

(Note: Do not use read-modify-write operations on this register.)

Bit7: JTAGRST. JTAG Reset Flag.
0: JTAG is not currently in reset state.
1: JTAG is in reset state.

Bit6: CNVRSEF: Convert Start Reset Source Enable and Flag
Write
0: CNVSTR is not a reset source
1: CNVSTR is a reset source (active low)
Read
0: Source of prior reset was not from CNVSTR
1: Source of prior reset was from CNVSTR

Bit5: C0RSEF: Comparator 0 Reset Enable and Flag
Write
0: Comparator 0 is not a reset source
1: Comparator 0 is a reset source (active low)
Read
Note: The value read from C0RSEF is not defined if Comparator 0 has not been enabled as a reset source.
0: Source of prior reset was not from Comparator 0
1: Source of prior reset was from Comparator 0

Bit4: SWRSF: Software Reset Force and Flag
Write
0: No Effect
1: Forces an internal reset. /RST pin is not effected.
Read
0: Prior reset source was not from write to the SWRSF bit.
1: Prior reset source was from write to the SWRSF bit.

Bit3: WDTRSF: Watchdog Timer Reset Flag
0: Source of prior reset was not from WDT timeout.
1: Source of prior reset was from WDT timeout.

Bit2: MCDRSF: Missing Clock Detector Flag
0: Source of prior reset was not from Missing Clock Detector timeout.
1: Source of prior reset was from Missing Clock Detector timeout.

Bit1: PORSF: Power-On Reset Force and Flag
Write
0: No effect
1: Forces a Power-On Reset. /RST is driven low.
Read
0: Source of prior reset was not from POR.
1: Source of prior reset was from POR.

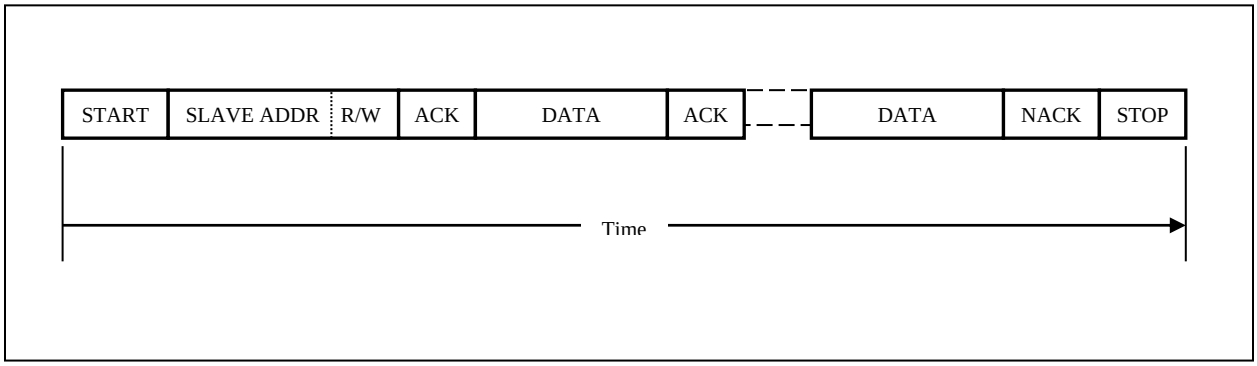
Bit0: PINRSF: HW Pin Reset Flag
0: Source of prior reset was not from /RST pin.
1: Source of prior reset was from /RST pin.

## 16.2. Operation

A typical SMBus transaction consists of a START condition, followed by an address byte, one or more bytes of data, and a STOP condition. The address byte and each of the data bytes are followed by an ACKNOWLEDGE bit from the receiver. The address byte consists of a 7-bit address plus a direction bit. The direction bit (R/W) occupies the least-significant bit position of the address. The direction bit is set to logic 1 to indicate a "READ" operation and cleared to logic 0 to indicate a "WRITE" operation. A general call address (0x00 +R/W) is recognized by all slave devices allowing a master to address multiple slave devices simultaneously.

All transactions are initiated by the master, with one or more addressed slave devices as the target. The master generates the START condition and then transmits the address and direction bit. If the transaction is a WRITE operation from the master to the slave, the master transmits the data a byte at a time waiting for an ACKNOWLEDGE from the slave at the end of each byte. If it is a READ operation, the slave transmits the data waiting for an ACKNOWLEDGE from the master at the end of each byte. At the end of the data transfer, the master generates a STOP condition to terminate the transaction and free the bus. Figure 16.3 illustrates a typical SMBus transaction.

**Figure 16.3. SMBus Transaction**




The SMBus interface may be configured to operate as either a master or a slave. At any particular time, it will be operating in one of the following four modes:

### 16.2.1. Master Transmitter Mode

Serial data is transmitted on SDA while the serial clock is output on SCL. The first byte transmitted contains the address of the target slave device and the data direction bit. In this case the data direction bit (R/W) will be logic 0 to indicate a "WRITE" operation. The master then transmits one or more bytes of serial data. After each byte is transmitted, an acknowledge bit is generated by the slave. To indicate the beginning and the end of the serial transfer, the master device outputs START and STOP conditions.

### 16.2.2. Master Receiver Mode

Serial data is received on SDA while the serial clock is output on SCL. The first byte is transmitted by the master and contains the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 1 to indicate a "READ" operation. Serial data is then received from the slave on SDA while the master outputs the serial clock. The slave transmits one or more bytes of serial data. After each byte is received, an acknowledge bit is transmitted by the master. The master outputs START and STOP conditions to indicate the beginning and end of the serial transfer.

### 16.2.3. Slave Transmitter Mode

Serial data is transmitted on SDA while the serial clock is received on SCL. First, a byte is received that contains an address and data direction bit. In this case the data direction bit (R/W) will be logic 1 to indicate a "READ" operation. If the received address matches the slave's assigned address (or a general call address is received) one or more bytes of serial data are transmitted to the master. After each byte is received, an acknowledge bit is transmitted by the master. The master outputs START and STOP conditions to indicate the beginning and end of the serial transfer.

### 16.2.4. Slave Receiver Mode

Serial data is received on SDA while the serial clock is received on SCL. First, a byte is received that contains an address and data direction bit. In this case the data direction bit (R/W) will be logic 0 to indicate a "WRITE" operation. If the received address matches the slave's assigned address (or a general call address is received) one or more bytes of serial data are received from the master. After each byte is received, an acknowledge bit is transmitted by the slave. The master outputs START and STOP conditions to indicate the beginning and end of the serial transfer.

## 16.3. Arbitration

A master may start a transfer only if the bus is free. The bus is free after a STOP condition or after the SCL and SDA lines remains high for a specified time. Two or more master devices may attempt to generate a START condition at the same time. Since the devices that generated the START condition may not be aware that other masters are contending for the bus, an arbitration scheme is employed. The master devices continue to transmit until one of the masters transmits a HIGH level, while the other(s) master transmits a LOW level on SDA. The first master(s) transmitting the HIGH level on SDA looses the arbitration and is required to give up the bus.

## 16.4. Clock Low Extension

SMBus provides a clock synchronization mechanism, similar to I2C, which allows devices with different speed capabilities to coexist on the bus. A clock-low extension is used during a transfer in order to allow slower slave devices to communicate with faster masters. The slave can hold the SCL line LOW to extend the clock low period, effectively decreasing the serial clock frequency.

## 16.5. Timeouts

### 16.5.1. SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than 25ms as a "timeout" condition. Devices that have detected the timeout condition must reset the communication no later than 10ms after detecting the timeout condition.

One of the MCU's general-purpose timers, operating in 16-bit auto-reload mode, can be used to monitor the SCL line for this timeout condition. Timer 3 is specifically designed for this purpose. (Refer to the Timer 3 Section 19.3. for detailed information on Timer 3 operation.)

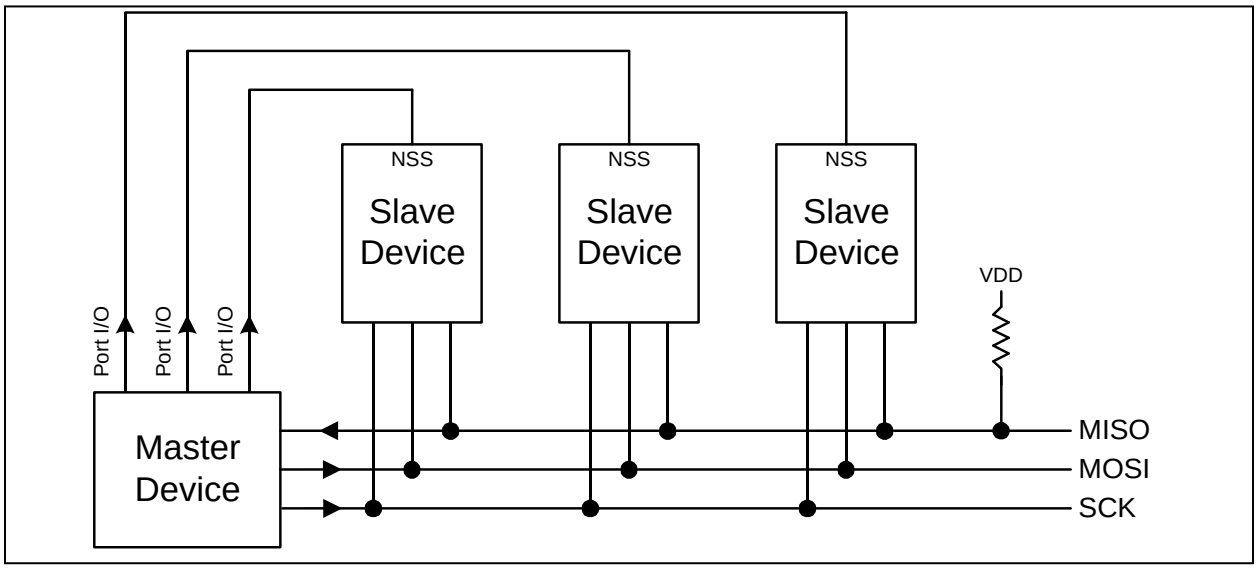
### 16.5.2. SCL High (SMBus Free) Timeout

The SMBus specification stipulates that if a device holds the SCL and SDA lines high for more that 50usec, the bus is designated as free. The SMB0CR register is used to detect this condition when the FTE bit in SMB0CN is set.

## 16.6. SMBus Special Function Registers

The SMBus serial interface is accessed and controlled through five SFRs: SMB0CN Control Register, SMB0CR Clock Rate Register, SMB0ADR Address Register, SMB0DAT Data Register and SMB0STA Status Register. The system device may have one or more SMBus serial interfaces implemented. The five special function registers related to the operation of the SMBus interface are described in the following section.

**Figure 17.2. Typical SPI Interconnection**




## 17.1. Signal Descriptions

The four signals used by the SPI (MOSI, MISO, SCK, NSS) are described below.

### 17.1.1. Master Out, Slave In

The master-out, slave-in (MOSI) signal is an output from a master device and an input to slave devices. It is used to serially transfer data from the master to the slave. Data is transferred most-significant bit first.

### 17.1.2. Master In, Slave Out

The master-in, slave-out (MISO) signal is an output from a slave device and an input to the master device. It is used to serially transfer data from the slave to the master. Data is transferred most-significant bit first. A SPI slave places the MISO pin in a high-impedance state when the slave is not selected.

### 17.1.3. Serial Clock

The serial clock (SCK) signal is an output from the master device and an input to slave devices. It is used to synchronize the transfer of data between the master and slave on the MOSI and MISO lines.

### 17.1.4. Slave Select

The slave select (NSS) signal is an input used to select the SPI module when in slave mode by a master, or to disable the SPI module when in master mode. When in slave mode, it is pulled low to initiate a data transfer and remains low for the duration of the transfer.

# 18. UART

The UART is a serial port capable of asynchronous transmission. The UART can function in full duplex mode. In all modes, receive data is buffered in a holding register. This allows the UART to start reception of a second incoming data byte before software has finished reading the previous data byte.

The UART has an associated Serial Control Register (SCON) and a Serial Data Buffer (SBUF) in the SFRs. The single SBUF location provides access to both transmit and receive registers. Reads access the Receive register and writes access the Transmit register automatically.

The UART is capable of generating interrupts if enabled. The UART has two sources of interrupts: a Transmit Interrupt flag, TI (SCON.1) set when transmission of a data byte is complete, and a Receive Interrupt flag, RI (SCON.0) set when reception of a data byte is complete. The UART interrupt flags are not cleared by hardware when the CPU vectors to the interrupt service routine. They must be cleared manually by software. This allows software to determine the cause of the UART interrupt (transmit complete or receive complete).

**Figure 18.1. UART Block Diagram**

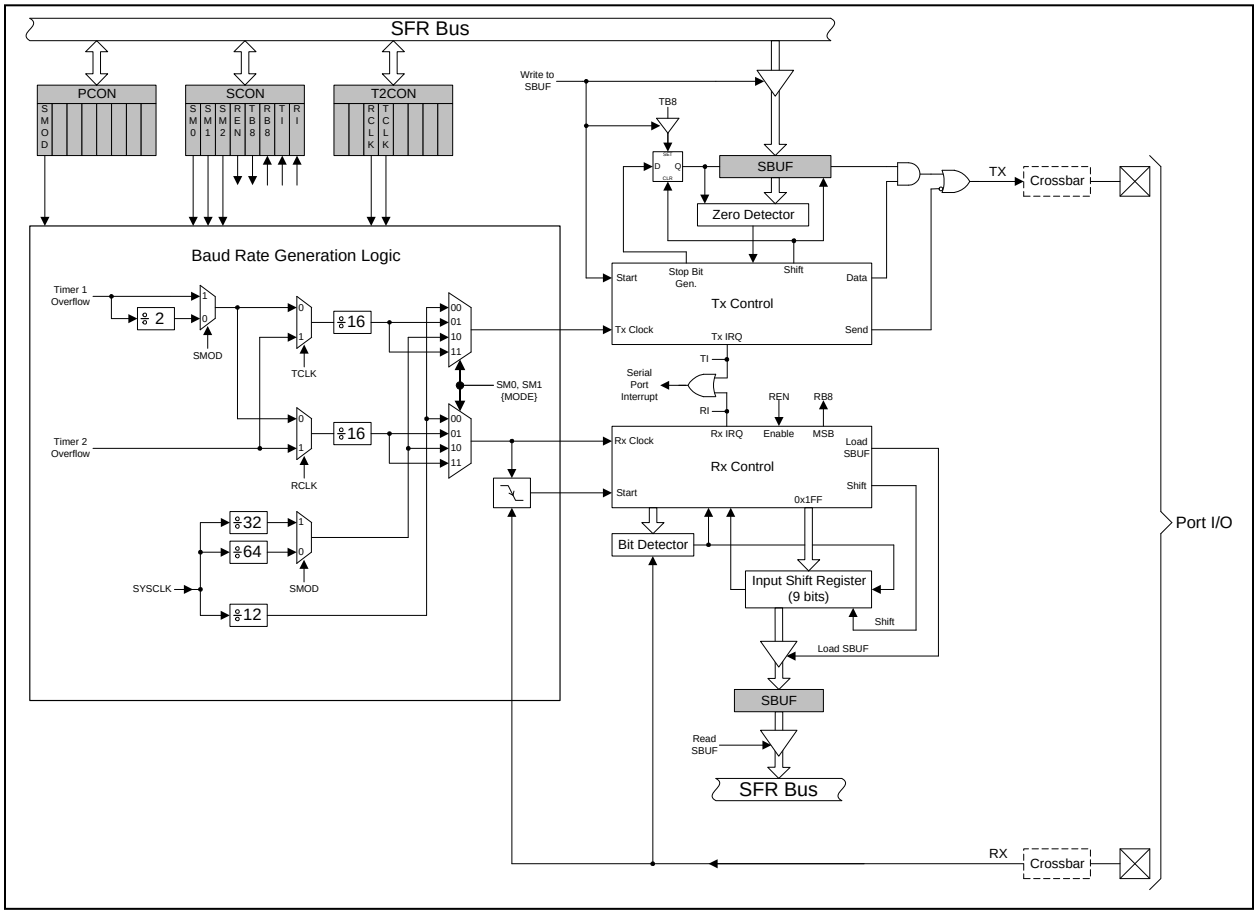
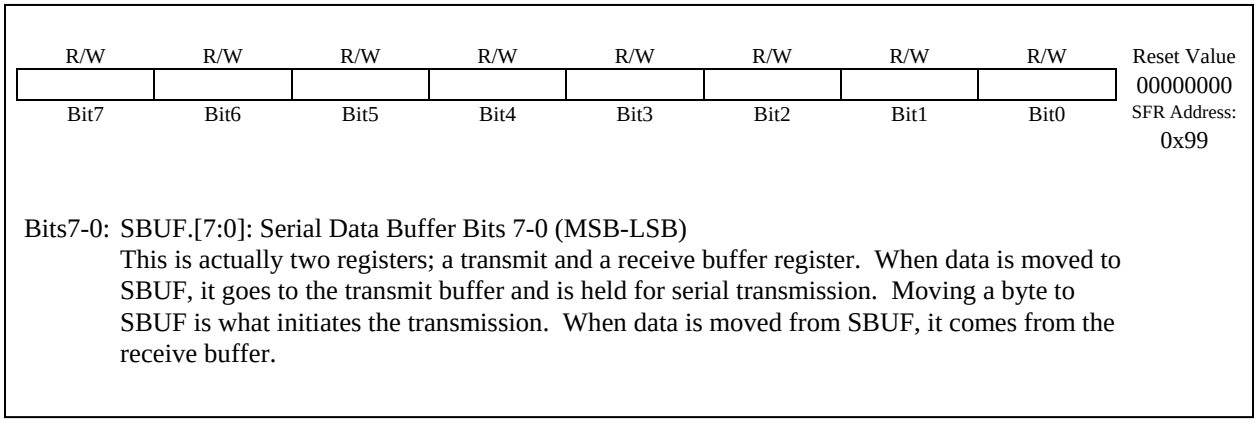


**Table 18.2. Oscillator Frequencies for Standard Baud Rates**

| Oscillator Frequency (MHz) | Divide Factor | Timer 1 Load Value* | Resulting Baud Rate** |
|---|---|---|---|
| 24.0 | 208 | 0xF3 | 115200 (115384) |
| 23.592 | 205 | 0xF3 | 115200 (113423) |
| 22.1184 | 192 | 0xF4 | 115200 |
| 18.432 | 160 | 0xF6 | 115200 |
| 16.5888 | 144 | 0xF7 | 115200 |
| 14.7456 | 128 | 0xF8 | 115200 |
| 12.9024 | 112 | 0xF9 | 115200 |
| 11.0592 | 96 | 0xFA | 115200 |
| 9.216 | 80 | 0xFB | 115200 |
| 7.3728 | 64 | 0xFC | 115200 |
| 5.5296 | 48 | 0xFD | 115200 |
| 3.6864 | 32 | 0xFE | 115200 |
| 1.8432 | 16 | 0xFF | 115200 |
| 24.576 | 320 | 0xEC | 76800 |
| 25.0 | 434 | 0xE5 | 57600 (57870) |
| 25.0 | 868 | 0xCA | 28800 |
| 24.576 | 848 | 0xCB | 28800 (28921) |
| 24.0 | 833 | 0xCC | 28800 (28846) |
| 23.592 | 819 | 0xCD | 28800 (28911) |
| 22.1184 | 768 | 0xD0 | 28800 |
| 18.432 | 640 | 0xD8 | 28800 |
| 16.5888 | 576 | 0xDC | 28800 |
| 14.7456 | 512 | 0xE0 | 28800 |
| 12.9024 | 448 | 0xE4 | 28800 |
| 11.0592 | 384 | 0xE8 | 28800 |
| 9.216 | 320 | 0xEC | 28800 |
| 7.3728 | 256 | 0xF0 | 28800 |
| 5.5296 | 192 | 0xF4 | 28800 |
| 3.6864 | 128 | 0xF8 | 28800 |
| 1.8432 | 64 | 0xFC | 28800 |

* Assumes SMOD=1 and T1M=1.
** Numbers in parenthesis show the actual baud rate.

**Figure 18.8. SBUF: Serial (UART) Data Buffer Register**




Bits7-0: SBUF.[7:0]: Serial Data Buffer Bits 7-0 (MSB-LSB)
This is actually two registers; a transmit and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer and is held for serial transmission. Moving a byte to SBUF is what initiates the transmission. When data is moved from SBUF, it comes from the receive buffer.

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

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

Timer 1 is inactive in Mode 3, so with Timer 0 in Mode 3, Timer 1 can be turned off and on by switching it into and out of its Mode 3. When Timer 0 is in Mode 3, Timer 1 can be operated in Modes 0, 1 or 2, but cannot be clocked by external signals nor set the TF1 flag and generate an interrupt. However, the Timer 1 overflow can be used for baud rate generation. Refer to Section 18 (UART) for information on configuring Timer 1 for baud rate generation.

**Figure 19.3. T0 Mode 3 Block Diagram**

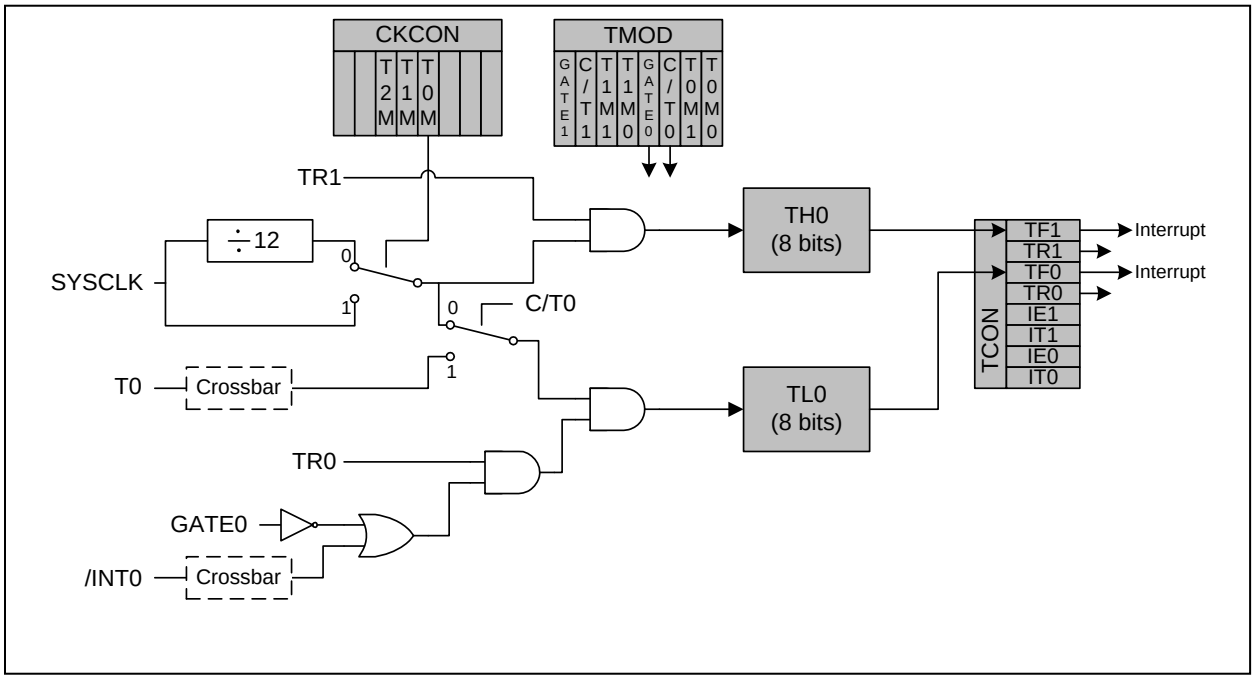


**Figure 19.15. RCAP2L: Timer 2 Capture Register Low Byte**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xCA |

Bits 7-0: RCAP2L: Timer 2 Capture Register Low Byte.
The RCAP2L register captures the low byte of Timer 2 when Timer 2 is configured in capture mode. When Timer 2 is configured in auto-reload mode, it holds the low byte of the reload value.

**Figure 19.16. RCAP2H: Timer 2 Capture Register High Byte**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xCB |

Bits 7-0: RCAP2H: Timer 2 Capture Register High Byte.
The RCAP2H register captures the high byte of Timer 2 when Timer 2 is configured in capture mode. When Timer 2 is configured in auto-reload mode, it holds the high byte of the reload value.

**Figure 19.17. TL2: Timer 2 Low Byte**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xCC |

Bits 7-0: TL2: Timer 2 Low Byte.
The TL2 register contains the low byte of the 16-bit Timer 2.

**Figure 19.18. TH2: Timer 2 High Byte**

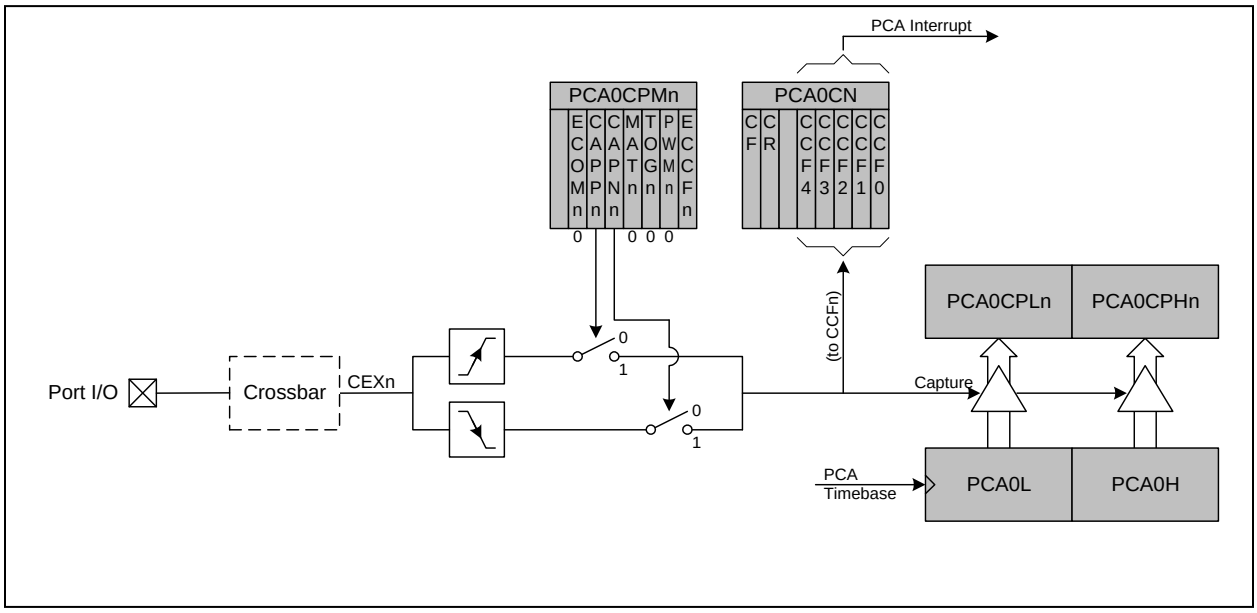
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 | SFR Address: 0xCD |

Bits 7-0: TH2: Timer 2 High Byte.
The TH2 register contains the high byte of the 16-bit Timer 2.

### 20.1.1. Edge-triggered Capture Mode

In this mode, a valid transition on the CEXn pin causes the PCA to capture the value of the PCA counter/timer and load it into the corresponding module's 16-bit capture/compare register (PCA0CPLn and PCA0CPHn). The CAPPn and CAPNn bits in the PCA0CPMn register are used to select the type of transition that triggers the capture: low-to-high transition (positive edge), high-to-low transition (negative edge), or either transition (positive or negative edge). When a capture occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1 and an interrupt request is generated if CCF interrupts are 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.

**Figure 20.3. PCA Capture Mode Diagram**



## 20.3. Register Descriptions for PCA

The system device may implement one or more Programmable Counter Arrays. Following are detailed descriptions of the special function registers related to the operation of the PCA. The CIP-51 System Controller section of the datasheet provides additional information on the SFRs and their use.

**Figure 20.8. PCA0CN: PCA Control Register**

| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | Reset Value |
|---|---|---|---|---|---|---|---|---|
| CF | CR | - | CCF4 | CCF3 | CCF2 | CCF1 | CCF0 | 00000000 |
| Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 (bit addressable) | SFR Address: 0xD8 |

Bit7: CF: PCA Counter/Timer Overflow Flag.
Set by hardware when the PCA Counter/Timer overflows from 0xFFFF to 0x0000. When the Counter/Timer Overflow (CF) interrupt is enabled, setting this bit causes the CPU to vector to the CF interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit6: CR: PCA Counter/Timer Run Control.
This bit enables/disables the PCA Counter/Timer.
0: PCA Counter/Timer disabled.
1: PCA Counter/Timer enabled.

Bit5: UNUSED. Read = 0, Write = don't care.

Bit4: CCF4: PCA Module 4 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF interrupt is enabled, setting this bit causes the CPU to vector to the CCF interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit3: CCF3: PCA Module 3 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF interrupt is enabled, setting this bit causes the CPU to vector to the CCF interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit2: CCF2: PCA Module 2 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF interrupt is enabled, setting this bit causes the CPU to vector to the CCF interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit1: CCF1: PCA Module 1 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF interrupt is enabled, setting this bit causes the CPU to vector to the CCF interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit0: CCF0: PCA Module 0 Capture/Compare Flag.
This bit is set by hardware when a match or capture occurs. When the CCF interrupt is enabled, setting this bit causes the CPU to vector to the CCF interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.