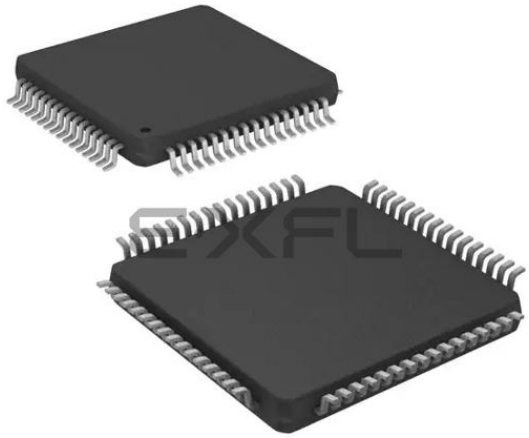


Welcome to E-XFL.COM

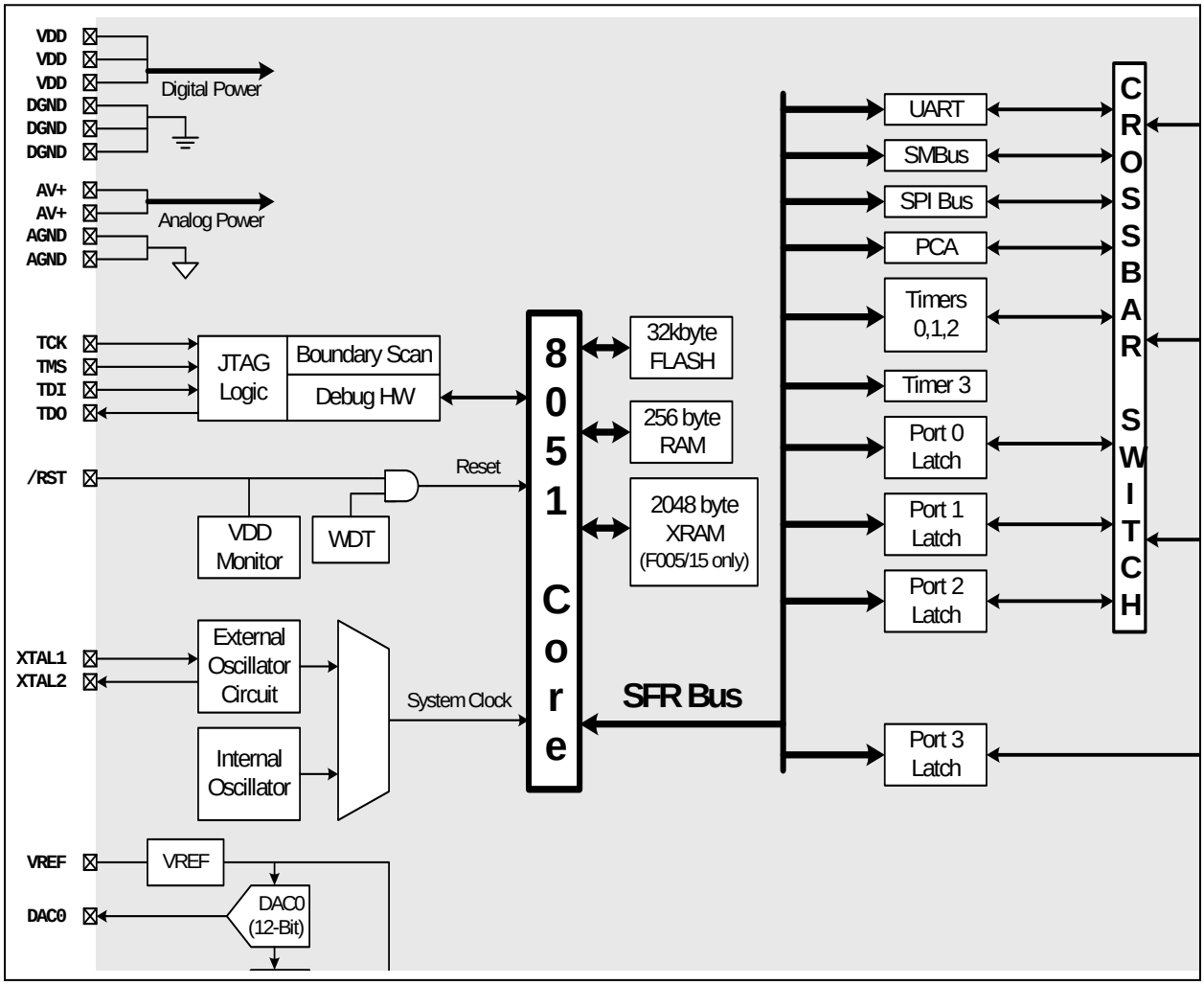
What is "Embedded - Microcontrollers"?

"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 8x10b; 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/c8051f010-gqr

Figure 1.1. C8051F000/05/10/15 Block Diagram



1.3. JTAG Debug and Boundary Scan

The C8051F000 family has on-chip JTAG and debug circuitry that provide *non-intrusive, full speed, in-circuit debug using the production part installed in the end application* using the four-pin JTAG I/F. The JTAG port is fully compliant to IEEE 1149.1, providing full boundary scan for test and manufacturing purposes.

Silicon Labs' debug system supports inspection and modification of memory and registers, breakpoints, watchpoints, a stack monitor, and single stepping. No additional target RAM, program memory, timers, or communications channels are required. All the digital and analog peripherals are functional and work correctly while debugging. All the peripherals (except for the ADC) are stalled when the MCU is halted, during single stepping, or at a breakpoint in order to keep them in sync.

The C8051F000DK, C8051F005DK, C8051F010DK, and C8051F015DK are development kits with all the hardware and software necessary to develop application code and perform in-circuit debug with the C8051F000/1/2, F005/6/7, F010/1/2, and F015/6/7 MCUs respectively. The kit includes software with a developer's studio and debugger, an integrated 8051 assembler, and an RS-232 to JTAG protocol translator module referred to as the EC. It also has a target application board with the associated MCU installed and a large prototyping area, plus the RS-232 and JTAG cables, and wall-mount power supply. The Development Kit requires a Windows 95/98/NT/2000/XP computer with one available RS-232 serial port. As shown in Figure 1.7, the PC is connected via RS-232 to the EC. A six-inch ribbon cable connects the EC to the user's application board, picking up the four JTAG pins and VDD and GND. The EC takes its power from the application board. It requires roughly 20mA at 2.7-3.6V. For applications where there is not sufficient power available from the target board, the provided power supply can be connected directly to the EC.

This is a vastly superior configuration for developing and debugging embedded applications compared to standard MCU Emulators, which use on-board "ICE Chips" and target cables and require the MCU in the application board to be socketed. Silicon Labs' debug environment both increases ease of use and preserves the performance of the precision analog peripherals.

Figure 1.7. Debug Environment Diagram

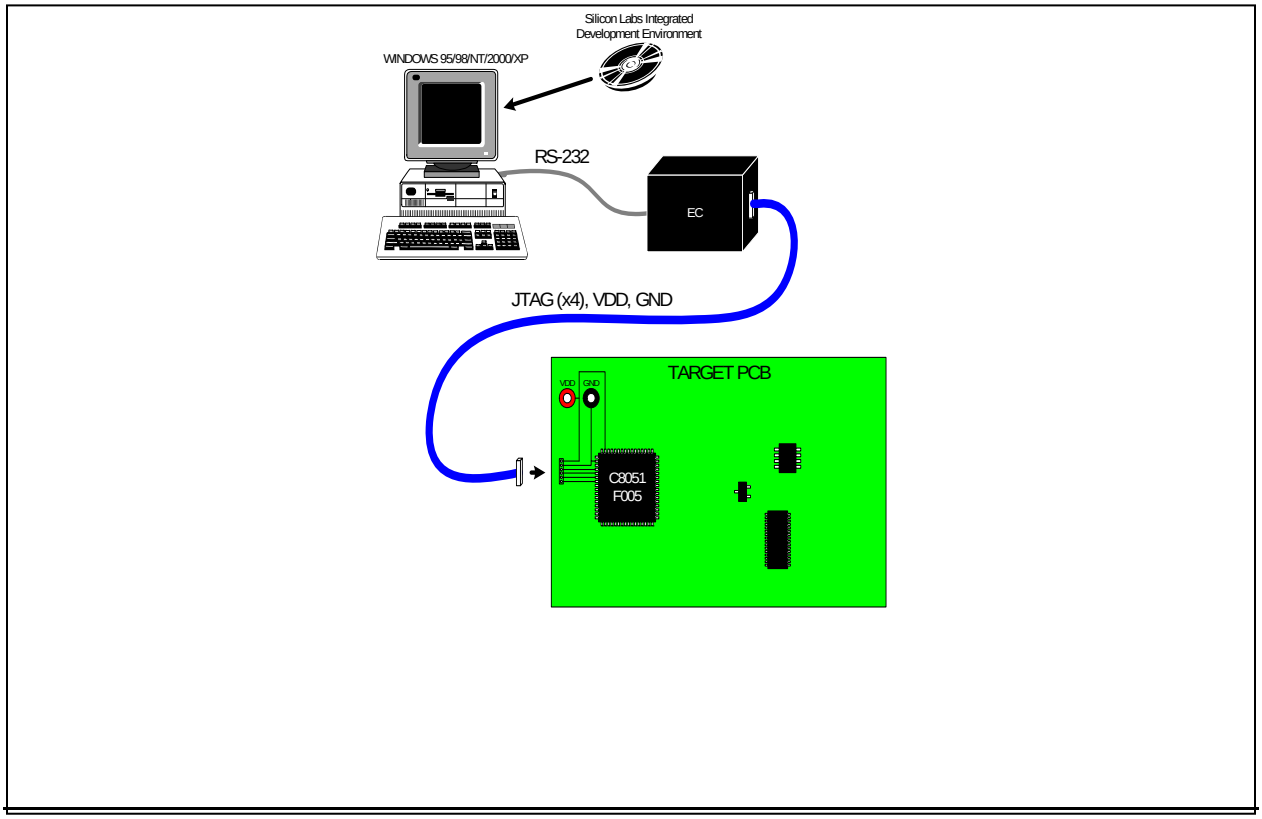
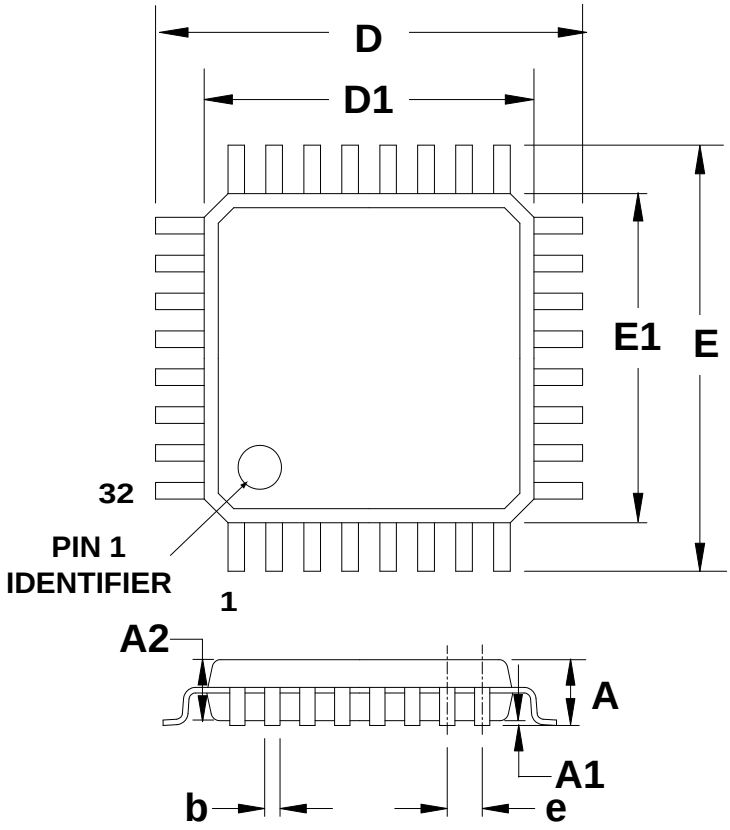


Figure 4.6. LQFP-32 Package Drawing



	MIN (mm)	NOM (mm)	MAX (mm)
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
b	0.30	0.37	0.45
D	-	9.00	-
D1	-	7.00	-
e	-	0.80	-
E	-	9.00	-
E1	-	7.00	-

7. DACs, 12 BIT VOLTAGE MODE

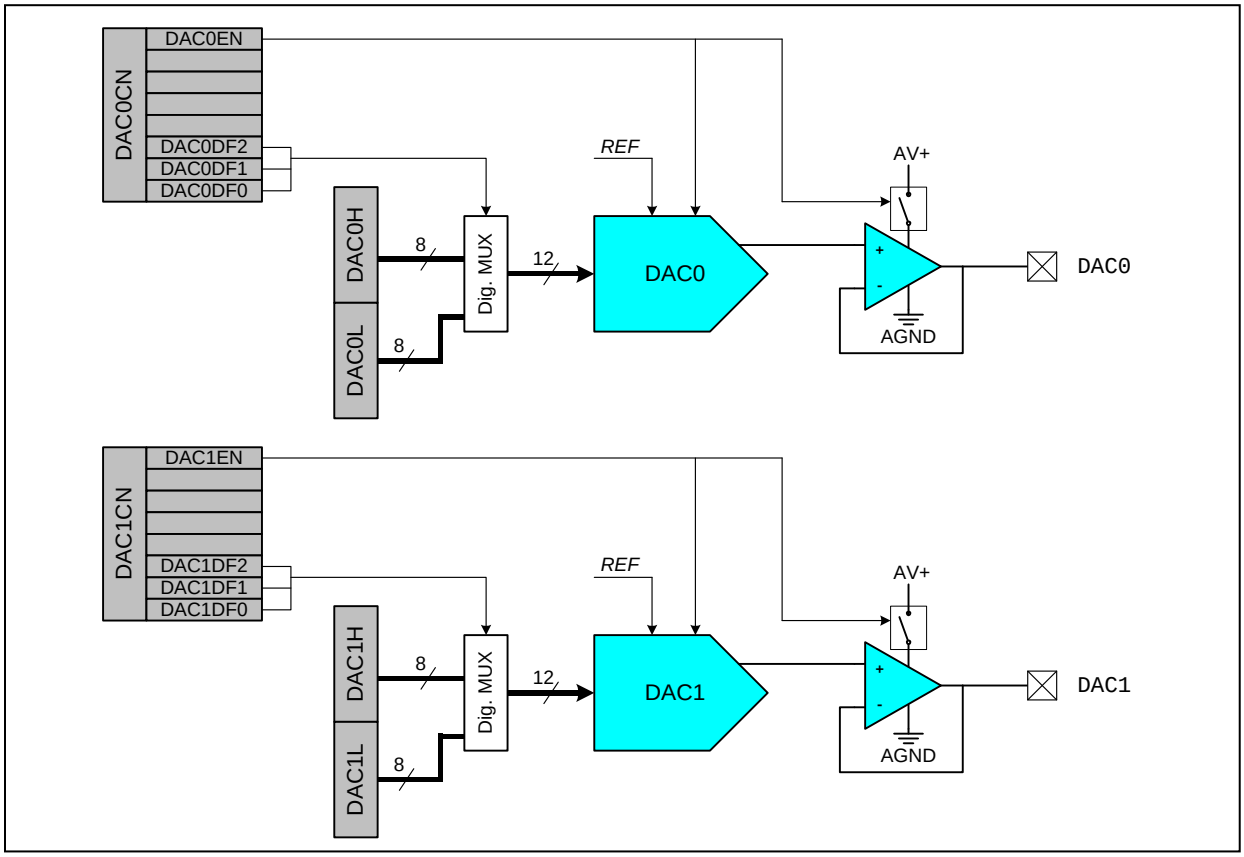
The C8051F000 MCU family has two 12-bit voltage-mode Digital to Analog Converters. Each DAC has an output swing of 0V to VREF-1LSB for a corresponding input code range of 0x000 to 0xFFF. Using DAC0 as an example, the 12-bit data word is written to the low byte (DAC0L) and high byte (DAC0H) data registers. 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



8. COMPARATORS

The MCU family has two on-chip analog voltage comparators as shown in Figure 8.1. The inputs of each Comparator are available at the package pins. The output of each comparator is optionally available at the package pins via the I/O crossbar (see Section 15.1). When assigned to package pins, each comparator output can be programmed to operate in open drain or push-pull modes (see section 15.3).

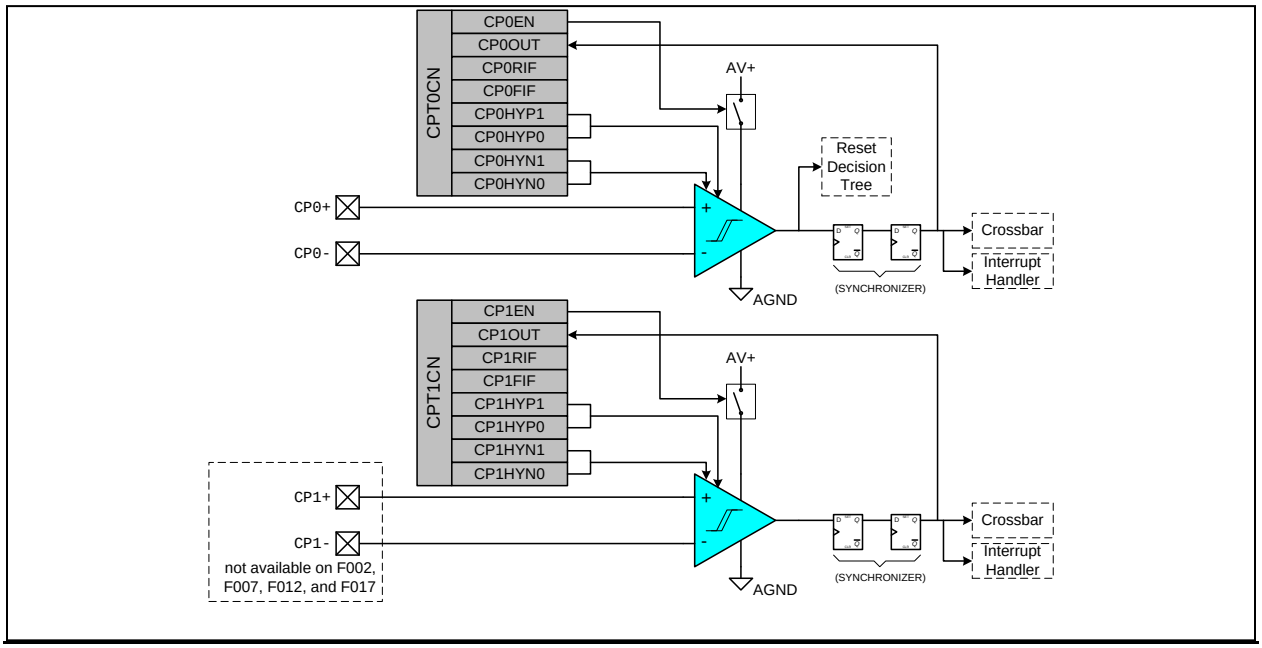
The hysteresis of each comparator is software-programmable via its respective Comparator control register (CPT0CN, CPT1CN). The user can program both the amount of hysteresis voltage (referred to the input voltage) and the positive and negative-going symmetry of this hysteresis around the threshold voltage. The output of the comparator can be polled in software, or can be used as an interrupt source. Each comparator can be individually enabled or disabled (shutdown). When disabled, the comparator output (if assigned to a Port I/O pin via the Crossbar) defaults to the logic low state, its interrupt capability is suspended and its supply current falls to less than 1μA. Comparator 0 inputs can be externally driven from -0.25V to (AV+) + 0.25V without damage or upset.

The Comparator 0 hysteresis is programmed using bits 3-0 in the Comparator 0 Control Register CPT0CN (shown in Figure 8.3). The amount of *negative* hysteresis voltage is determined by the settings of the CP0HYN bits. As shown in Figure 8.2, settings of 10, 4 or 2mV of negative hysteresis can be programmed, or negative hysteresis can be disabled. In a similar way, the amount of *positive* hysteresis is determined by the setting the CP0HYP bits.

Comparator interrupts can be generated on both rising-edge and falling-edge output transitions. (For Interrupt enable and priority control, see Section 10.4). The CP0FIF flag is set upon a Comparator 0 falling-edge interrupt, and the CP0RIF flag is set upon the Comparator 0 rising-edge interrupt. Once set, these bits remain set until cleared by the CPU. The Output State of Comparator 0 can be obtained at any time by reading the CP0OUT bit. Note the comparator output and interrupt should be ignored until the comparator settles after power-up. Comparator 0 is enabled by setting the CP0EN bit, and is disabled by clearing this bit. Note there is a 20usec settling time for the comparator output to stabilize after setting the CP0EN bit or a power-up. Comparator 0 can also be programmed as a reset source. For details, see Section 13.

The operation of Comparator 1 is identical to that of Comparator 0, except the Comparator 1 is controlled by the CPT1CN Register (Figure 8.4). Comparator 1 can not be programmed as a reset source. Also, the input pins for Comparator 1 are not pinned out on the F002, F007, F012, or F017 devices. The complete electrical specifications for the Comparators are given in Table 8.1.

Figure 8.1. Comparator Functional Block Diagram



10. CIP-51 CPU

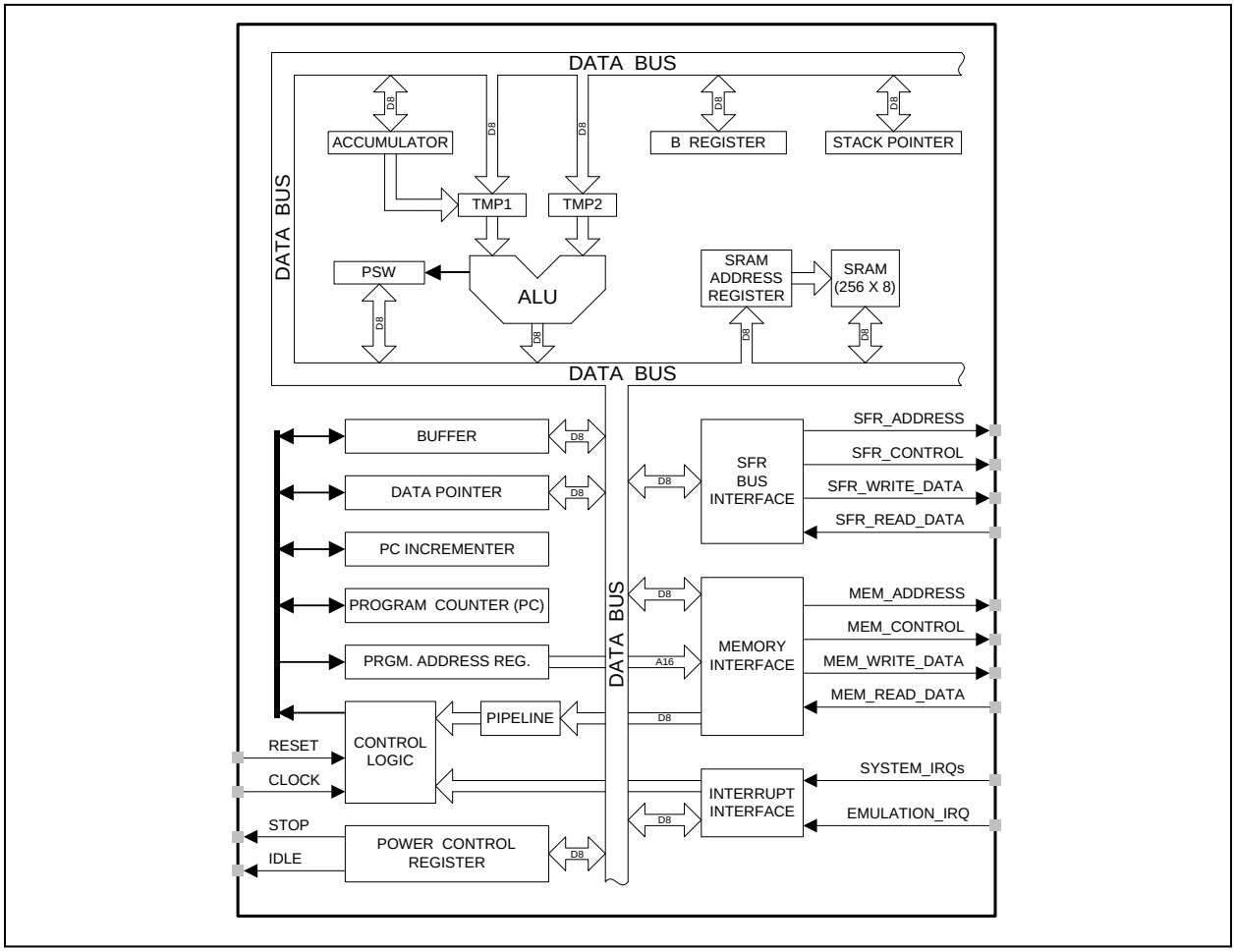
The MCUs' system CPU is the CIP-51. The CIP-51 is fully compatible with the MCS-51™ instruction set. Standard 803x/805x assemblers and compilers can be used to develop software. The MCU family has a superset of all the peripherals included with a standard 8051. Included are four 16-bit counter/timers (see description in Section 19), a full-duplex UART (see description in Section 18), 256 bytes of internal RAM, 128 byte Special Function Register (SFR) address space (see Section 10.3), and four byte-wide I/O Ports (see description in Section 14). The CIP-51 also includes on-chip debug hardware (see description in Section 21), and interfaces directly with the MCUs' analog and digital subsystems providing a complete data acquisition or control-system solution in a single integrated circuit.

Features

The CIP-51 Microcontroller core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability (see Figure 10.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 25 MIPS Peak Throughput with 25MHz Clock
- 0 to 25MHz Clock Frequency (on 'F0x5/6/7)
- Four Byte-Wide I/O Ports
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- On-chip Debug Circuitry
- Program and Data Memory Security

Figure 10.1. CIP-51 Block Diagram



Performance

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute, and usually have a maximum system clock of 12MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with no instructions taking more than eight system clock cycles.

With the CIP-51's maximum system clock at 25MHz, it has a peak throughput of 25MIPS. The CIP-51 has a total of 109 instructions. The number of instructions versus the system clock cycles required to execute them is as follows:

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

Programming and Debugging Support

A JTAG-based serial interface is provided for in-system programming of the Flash program memory and communication with on-chip debug support circuitry. The reprogrammable Flash can also be read and changed a single byte at a time by the application software using the MOVC and MOVX instructions. This feature allows program memory to be used for non-volatile data storage as well as updating program code under software control.

The on-chip debug support circuitry facilitates full speed in-circuit debugging, allowing the setting of hardware breakpoints and watch points, 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 and non-invasive, requiring no RAM, Stack, timers, or other on-chip resources.

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

10.1. 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.

10.1.1. 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 10.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

10.1.2. MOVX Instruction and Program Memory

The MOVX instruction is typically used to access external data memory. In the CIP-51, the MOVX instruction can access the on-chip program memory space implemented as reprogrammable Flash memory using the control bits in the PSCTL register (see Figure 11.1). This feature provides a mechanism for the CIP-51 to update program code and use the program memory space for non-volatile data storage. For the products with RAM mapped into external data memory space (C8051F005/06/07/15/16/17), MOVX is still used to read/write this memory with the PSCTL

Address	Register	Description	Page No.
0x89	TMOD	Counter/Timer Mode	143
0x91	TMR3CN	Timer 3 Control	152
0x95	TMR3H	Timer 3 High	153
0x94	TMR3L	Timer 3 Low	153
0x93	TMR3RLH	Timer 3 Reload High	153
0x92	TMR3RLL	Timer 3 Reload Low	153
0xFF	WDTCN	Watchdog Timer Control	96
0xE1	XBR0	Port I/O Crossbar Configuration 1	105
0xE2	XBR1	Port I/O Crossbar Configuration 2	107
0xE3	XBR2	Port I/O Crossbar Configuration 3	108
0x84-86, 0x96-97, 0x9C, 0xA1-A3, 0xA9-AC, 0xAE, 0xB3-B5, 0xB9, 0xBD, 0xC9, 0xCE, 0xDF, 0xE4-E5, 0xF1-F5		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.

Figure 10.7. ACC: Accumulator

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	ACC.0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (bit addressable)	SFR Address: 0xE0

Bits 7-0: ACC: Accumulator
This register is the accumulator for arithmetic operations.

Figure 10.8. B: B Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
B.7	B.6	B.5	B.4	B.3	B.2	B.1	B.0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (bit addressable)	SFR Address: 0xF0

Bits 7-0: B: B Register
This register serves as a second accumulator for certain arithmetic operations.

13. RESET SOURCES

The reset circuitry of the MCUs allows the controller to be easily placed in a predefined default condition. On entry to this reset state, the CIP-51 halts program execution, forces the external port pins to a known state and initializes the SFRs to their defined reset values. Interrupts and timers are disabled. On exit, the program counter (PC) is reset, and program execution starts at location 0x0000.

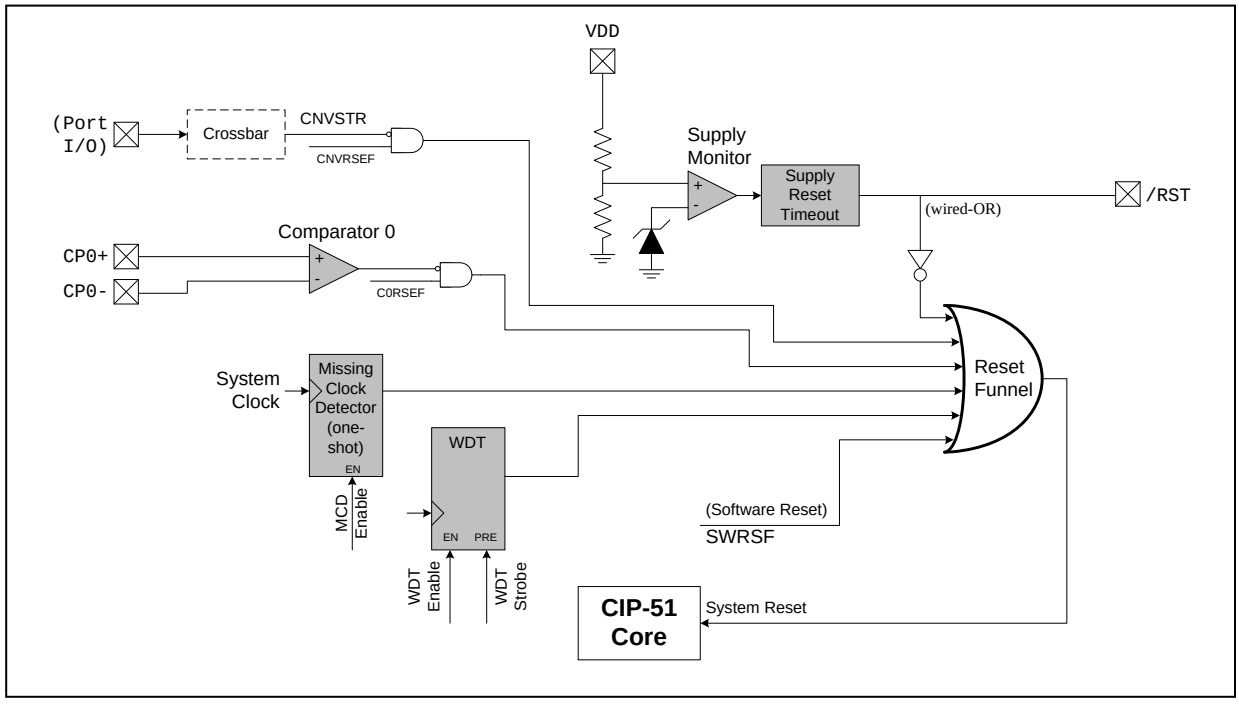
All of the SFRs are reset to predefined values. The reset values of the SFR bits are defined in the SFR detailed descriptions. The contents of internal data memory are not changed during a reset and any previously stored data is preserved. However, since the stack pointer SFR is reset, the stack is effectively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

The I/O port latches are reset to 0xFF (all logic ones), activating internal weak pull-ups which take the external I/O pins to a high state. The weak pull-ups are enabled during and after the reset. If the source of reset is from the VDD Monitor or writing a 1 to PORSF, the /RST pin is driven low until the end of the VDD reset timeout.

On exit from the reset state, the MCU uses the internal oscillator running at 2MHz as the system clock by default. Refer to Section 14 for information on selecting and configuring the system clock source. The Watchdog Timer is enabled using its longest timeout interval. (Section 13.8 details the use of the Watchdog Timer.)

There are seven sources for putting the MCU into the reset state: power-on/power-fail, external /RST pin, external CNVSTR signal, software commanded, Comparator 0, Missing Clock Detector, and Watchdog Timer. Each reset source is described below:

Figure 13.1. Reset Sources Diagram



13.4. External Reset

The external /RST pin provides a means for external circuitry to force the MCU into a reset state. Asserting an active-low signal on the /RST pin will cause the MCU to enter the reset state. Although there is a weak internal pullup, it may be desirable to provide an external pull-up and/or decoupling of the /RST pin to avoid erroneous noise-induced resets. The MCU will remain in reset until at least 12 clock cycles after the active-low /RST signal is removed. The PINRSF flag (RSTSRC.0) is set on exit from an external reset. The /RST pin is also 5V tolerant.

13.5. Missing Clock Detector Reset

The Missing Clock Detector is essentially a one-shot circuit that is triggered by the MCU system clock. If the system clock goes away for more than 100µs, the one-shot will time out and generate a reset. After a Missing Clock Detector reset, the MCDRSF flag (RSTSRC.2) will be set, signifying the MSD as the reset source; otherwise, this bit reads 0. The state of the /RST pin is unaffected by this reset. Setting the MSCLKE bit in the OSCICN register (see Figure 14.2) enables the Missing Clock Detector.

13.6. Comparator 0 Reset

Comparator 0 can be configured as an active-low reset input by writing a 1 to the C0RSEF flag (RSTSRC.5). Comparator 0 should be enabled using CPT0CN.7 (see Figure 8.3) at least 20µs prior to writing to C0RSEF to prevent any turn-on chatter on the output from generating an unwanted reset. When configured as a reset, if the non-inverting input voltage (on CP0+) is less than the inverting input voltage (on CP0-), the MCU is put into the reset state. After a Comparator 0 Reset, the C0RSEF flag (RSTSRC.5) will read 1 signifying Comparator 0 as the reset source; otherwise, this bit reads 0. The state of the /RST pin is unaffected by this reset. Also, Comparator 0 can generate a reset with or without the system clock.

13.7. External CNVSTR Pin Reset

The external CNVSTR signal can be configured as an active-low reset input by writing a 1 to the CNVRSEF flag (RSTSRC.6). The CNVSTR signal can appear on any of the P0, P1, or P2 I/O pins as described in Section 15.1. (Note that the Crossbar must be configured for the CNVSTR signal to be routed to the appropriate Port I/O.) The Crossbar should be configured and enabled before the CNVRSEF is set to configure CNVSTR as a reset source. When configured as a reset, CNVSTR is active-low and level sensitive. After a CNVSTR reset, the CNVRSEF flag (RSTSRC.6) will read 1 signifying CNVSTR as the reset source; otherwise, this bit reads 0. The state of the /RST pin is unaffected by this reset.

13.8. Watchdog Timer Reset

The MCU includes a programmable Watchdog Timer (WDT) running off the system clock. The WDT will force the MCU into the reset state when the watchdog timer overflows. To prevent the reset, the WDT must be restarted by application software before the overflow occurs. If the system experiences a software/hardware malfunction preventing the software from restarting the WDT, the WDT will overflow and cause a reset. This should prevent the system from running out of control.

The WDT is automatically enabled and started with the default maximum time interval on exit from all resets. If desired the WDT can be disabled by system software or locked on to prevent accidental disabling. Once locked, the WDT cannot be disabled until the next system reset. The state of the /RST pin is unaffected by this reset.

13.8.1. Watchdog Usage

The WDT consists of a 21-bit timer running from the programmed system clock. The timer measures the period between specific writes to its control register. If this period exceeds the programmed limit, a WDT reset is generated. The WDT can be enabled and disabled as needed in software, or can be permanently enabled if desired. Watchdog features are controlled via the Watchdog Timer Control Register (WDTCN) shown in Figure 13.3.

Enable/Reset WDT

The watchdog timer is both enabled and the countdown restarted by writing 0xA5 to the WDTCN register. The user's application software should include periodic writes of 0xA5 to WDTCN as needed to prevent a watchdog timer overflow. The WDT is enabled and restarted as a result of any system reset.

Disable WDT

Writing 0xDE followed by 0xAD to the WDTCN register disables the WDT. The following code segment illustrates disabling the WDT.

```
CLR   EA          ; disable all interrupts
MOV   WDTCN,#0DEh ; disable software
MOV   WDTCN,#0ADh ; watchdog timer
SETB  EA          ; re-enable interrupts
```

The writes of 0xDE and 0xAD must occur within 4 clock cycles of each other, or the disable operation is ignored. Interrupts should be disabled during this procedure to avoid delay between the two writes.

Disable WDT Lockout

Writing 0xFF to WDTCN locks out the disable feature. Once locked out, the disable operation is ignored until the next system reset. Writing 0xFF does not enable or reset the watchdog timer. Applications always intending to use the watchdog should write 0xFF to WDTCN in their initialization code.

Setting WDT Interval

WDTCN.[2:0] control the watchdog timeout interval. The interval is given by the following equation:

$$4^{3+WDTCN[2:0]} \times T_{SYSCLK}\text{ , (where } T_{SYSCLK} \text{ is the system clock period).}$$

For a 2MHz system clock, this provides an interval range of 0.032msec to 524msec. WDTCN.7 must be a 0 when setting this interval. Reading WDTCN returns the programmed interval. WDTCN.[2:0] is 111b after a system reset.

Figure 13.3. WDTCN: Watchdog Timer Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								xxxxx111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xFF

Bits7-0: WDT Control
Writing 0xA5 both enables and reloads the WDT.
Writing 0xDE followed within 4 clocks by 0xAD disables the WDT.
Writing 0xFF locks out the disable feature.

Bit4: Watchdog Status Bit (when Read)
Reading the WDTCN.[4] bit indicates the Watchdog Timer Status.
0: WDT is inactive
1: WDT is active

Bits2-0: Watchdog Timeout Interval Bits
The WDTCN.[2:0] bits set the Watchdog Timeout Interval. When writing these bits, WDTCN.7 must be set to 0.

Figure 14.3. OSCXCN: External Oscillator Control Register

R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
XTLVLD	XOSCMD2	XOSCMD1	XOSCMD0	-	XFCN2	XFCN1	XFCN0	00110000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB1

Bit7: XTLVLD: Crystal Oscillator Valid Flag
(Valid only when XOSCMD = 1xx.)
0: Crystal Oscillator is unused or not yet stable
1: Crystal Oscillator is running and stable (should read 1ms after Crystal Oscillator is enabled to avoid transient condition).

Bits6-4: XOSCMD2-0: External Oscillator Mode Bits
00x: Off. XTAL1 pin is grounded internally.
010: System Clock from External CMOS Clock on XTAL1 pin.
011: System Clock from External CMOS Clock on XTAL1 pin divided by 2.
10x: RC/C Oscillator Mode with divide by 2 stage.
110: Crystal Oscillator Mode
111: Crystal Oscillator Mode with divide by 2 stage.

Bit3: RESERVED. Read = undefined, Write = don't care

Bits2-0: XFCN2-0: External Oscillator Frequency Control Bits
000-111: see table below

XFCN	Crystal (XOSCMD = 11x)	RC (XOSCMD = 10x)	C (XOSCMD = 10x)
000	f ≤ 12.5kHz	f ≤ 25kHz	K Factor = 0.44
001	12.5kHz < f ≤ 30.3kHz	25kHz < f ≤ 50kHz	K Factor = 1.4
010	30.35kHz < f ≤ 93.8kHz	50kHz < f ≤ 100kHz	K Factor = 4.4
011	93.8kHz < f ≤ 267kHz	100kHz < f ≤ 200kHz	K Factor = 13
100	267kHz < f ≤ 722kHz	200kHz < f ≤ 400kHz	K Factor = 38
101	722kHz < f ≤ 2.23MHz	400kHz < f ≤ 800kHz	K Factor = 100
110	2.23MHz < f ≤ 6.74MHz	800kHz < f ≤ 1.6MHz	K Factor = 420
111	f > 6.74MHz	1.6MHz < f ≤ 3.2MHz	K Factor = 1400

CRYSTAL MODE (Circuit from Figure 14.1, Option 1; XOSCMD = 11x)
Choose XFCN value to match the crystal or ceramic resonator frequency.

RC MODE (Circuit from Figure 14.1, Option 2; XOSCMD = 10x)
Choose oscillation frequency range where:
$f = 1.23(10^3) / (R * C)$, where
f = frequency of oscillation in MHz
C = capacitor value in pF
R = Pull-up resistor value in kΩ

C MODE (Circuit from Figure 14.1, Option 3; XOSCMD = 10x)
Choose K Factor (KF) for the oscillation frequency desired:
f = KF / (C * AV+), where
f = frequency of oscillation in MHz
C = capacitor value on XTAL1, XTAL2 pins in pF
AV+ = Analog Power Supply on MCU in volts

Table 15.1. Crossbar Priority Decode

	P0								P1								P2							
PIN I/O	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
SDA	●																							
SCL		●																						
SCK	●		●																					
MISO		●		●																				
MOSI			●		●																			
NSS				●		●																		
TX	●		●		●		●																	
RX		●		●		●		●																
CEX0	●		●		●		●		●															
CEX1		●		●		●		●		●														
CEX2			●		●		●		●		●													
CEX3				●		●		●		●		●												
CEX4					●		●		●		●		●											
ECI	●	●	●	●	●	●	●	●	●	●	●	●	●	●										
CP0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●									
CP1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●								
T0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●							
/INT0	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●						
T1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●					
/INT1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●				
T2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●			
T2EX	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		
/SYSCLK	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	
CNVSTR	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

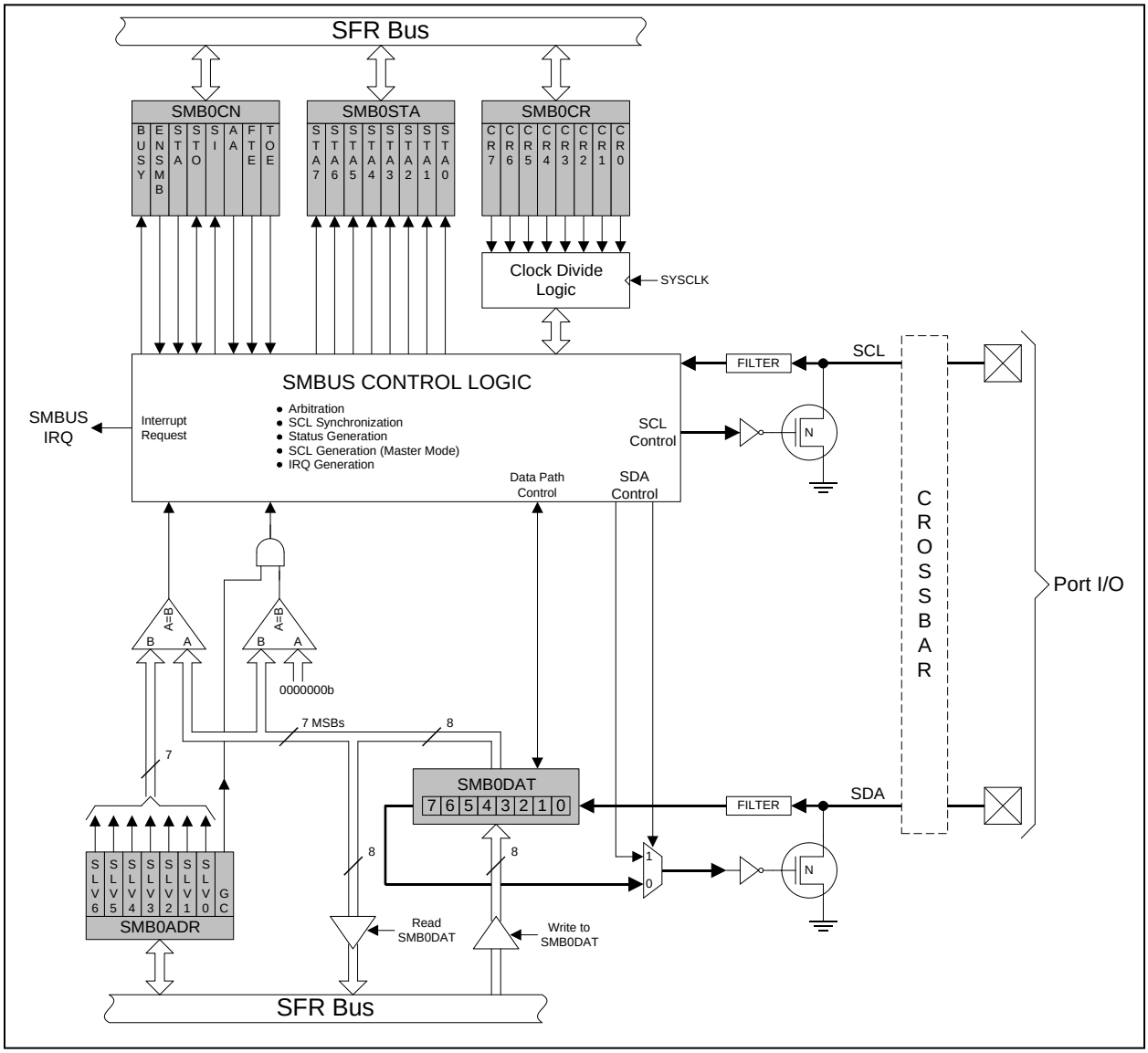
In the Priority Decode Table, a dot (•) is used to show the external Port I/O pin (column) to which each signal (row) can be assigned by the user application code via programming registers XBR2, XBR1, and XBR0.

16. SMBus / I2C Bus

The SMBus serial I/O interface is compliant with the System Management Bus Specification, version 1.1. It is a two-wire, bi-directional serial bus, which is also compatible with the I^2C serial bus. Reads and writes to the interface by the system controller are byte oriented with the SMBus interface autonomously controlling the serial transfer of the data. Data can be transferred at up to $1/8^{th}$ of the system clock if desired (this can be faster than allowed by the SMBus specification, depending on the system clock used). A method of extending the clock-low duration is used to accommodate devices with different speed capabilities on the same bus.

Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver, and data transfers from an addressed slave transmitter to a master receiver. The master device initiates both types of data transfers and provides the serial clock pulses. The SMBus interface may operate as a master or a slave. Multiple master devices on the same bus are also supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration.

Figure 16.1. SMBus Block Diagram



17. SERIAL PERIPHERAL INTERFACE BUS

The Serial Peripheral Interface (SPI) provides access to a four-wire, full-duplex, serial bus. SPI supports the connection of multiple slave devices to a master device on the same bus. A separate slave-select signal (NSS) is used to select a slave device and enable a data transfer between the master and the selected slave. Multiple masters on the same bus are also supported. Collision detection is provided when two or more masters attempt a data transfer at the same time. The SPI can operate as either a master or a slave. When the SPI is configured as a master, the maximum data transfer rate (bits/sec) is one-half the system clock frequency.

When the SPI is configured as a slave, the maximum data transfer rate (bits/sec) for full-duplex operation is 1/10 the system clock frequency, provided that the master issues SCK, NSS, and the serial input data synchronously with the system clock. If the master issues SCK, NSS, and the serial input data asynchronously, the maximum data transfer rate (bits/sec) must be less that 1/10 the system clock frequency. In the special case where the master only wants to transmit data to the slave and does not need to receive data from the slave (i.e. half-duplex operation), the SPI slave can receive data at a maximum data transfer rate (bits/sec) of ¼ the system clock frequency. This is provided that the master issues SCK, NSS, and the serial input data synchronously with the system clock.

Figure 17.1. SPI Block Diagram

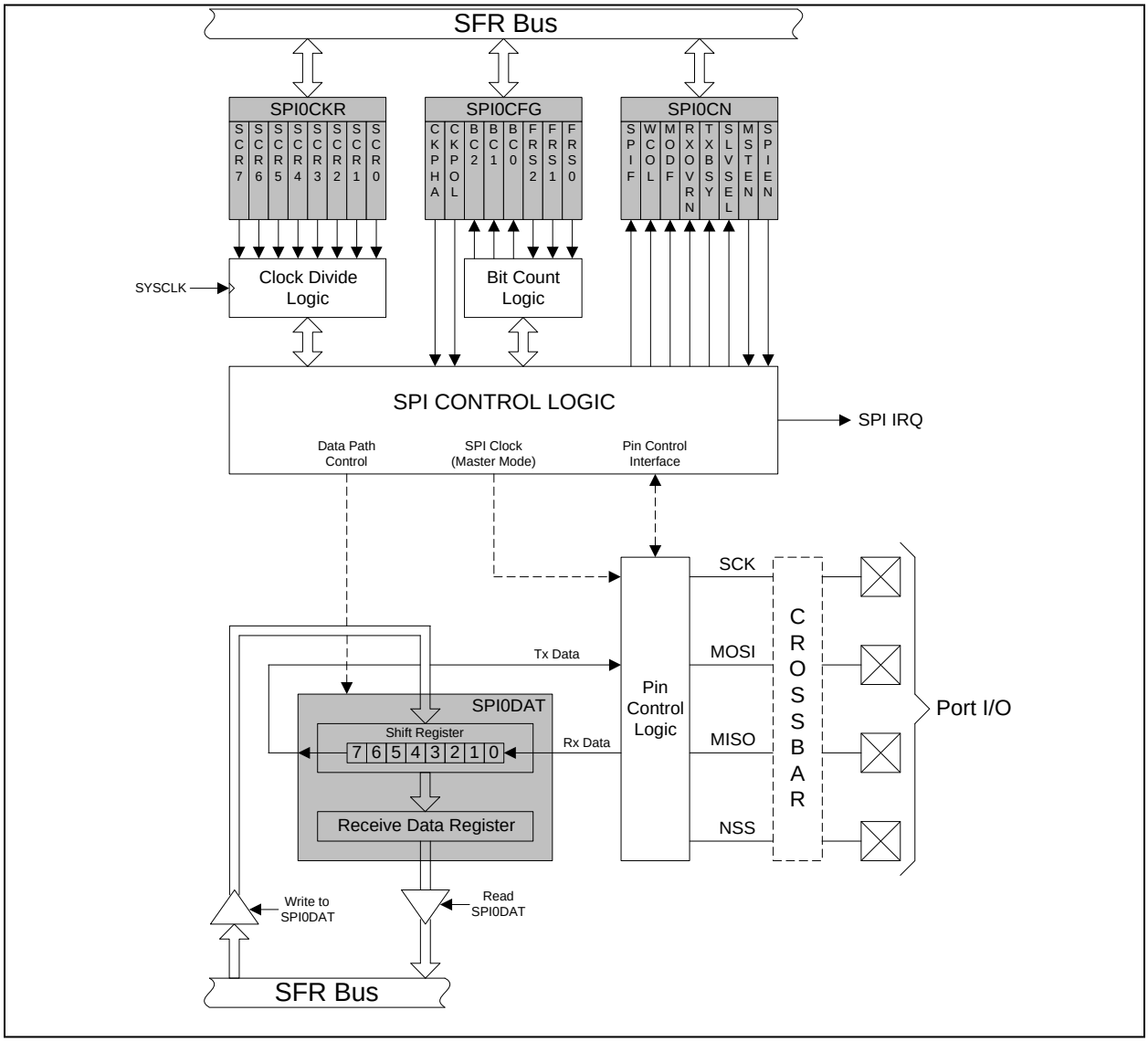


Figure 17.6. SPI0CN: SPI Control Register

R/W	R/W	R/W	R/W	R	R	R/W	R/W	Reset Value
SPIF	WCOL	MODF	RXOVRN	TXBSY	SLVSEL	MSTEN	SPIEN	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (bit addressable)	SFR Address: 0xF8

Bit7: SPIF: SPI Interrupt Flag.
This bit is set to logic 1 by hardware at the end of a data transfer. If interrupts are enabled, setting this bit causes the CPU to vector to the SPI0 interrupt service routine. This bit is not automatically cleared by hardware. It must be cleared by software.

Bit6: WCOL: Write Collision Flag.
This bit is set to logic 1 by hardware (and generates a SPI interrupt) to indicate a write to the SPI data register was attempted while a data transfer was in progress. It is cleared by software.

Bit5: MODF: Mode Fault Flag.
This bit is set to logic 1 by hardware (and generates a SPI interrupt) when a master mode collision is detected (NSS is low and MSTEN = 1). This bit is not automatically cleared by hardware. It must be cleared by software.

Bit4: RXOVRN: Receive Overrun Flag.
This bit is set to logic 1 by hardware (and generates a SPI interrupt) when the receive buffer still holds unread data from a previous transfer and the last bit of the current transfer is shifted into the SPI shift register. This bit is not automatically cleared by hardware. It must be cleared by software.

Bit3: TXBSY: Transmit Busy Flag.
This bit is set to logic 1 by hardware while a master mode transfer is in progress. It is cleared by hardware at the end of the transfer.

Bit2: SLVSEL: Slave Selected Flag.
This bit is set to logic 1 whenever the NSS pin is low indicating it is enabled as a slave. It is cleared to logic 0 when NSS is high (slave disabled).

Bit1: MSTEN: Master Mode Enable.
0: Disable master mode. Operate in slave mode.
1: Enable master mode. Operate as a master.

Bit0: SPIEN: SPI Enable.
This bit enables/disables the SPI.
0: SPI disabled.
1: SPI enabled.

18.1.3. Mode 2: 9-Bit UART, Fixed Baud Rate

Mode 2 provides asynchronous, full-duplex communication using a total of eleven bits per data byte: a start bit, 8 data bits (LSB first), a programmable ninth data bit, and a stop bit (see timing diagram in Figure 18.6). On transmit, the ninth data bit is determined by the value in TB8 (SCON.3). It can be assigned the value of the parity flag P in the PSW or used in multiprocessor communications. On receive, the ninth data bit goes into RB8 (SCON.2) and the stop bit is ignored.

Data transmission begins when an instruction writes a data byte to the SBUF register. The TI Transmit Interrupt Flag (SCON.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN Receive Enable bit (SCON.4) is set to logic 1. After the stop bit is received, the data byte will be loaded into the SBUF receive register if the following conditions are met: RI must be logic 0, and if SM2 is logic 1, the 9th bit must be logic 1.

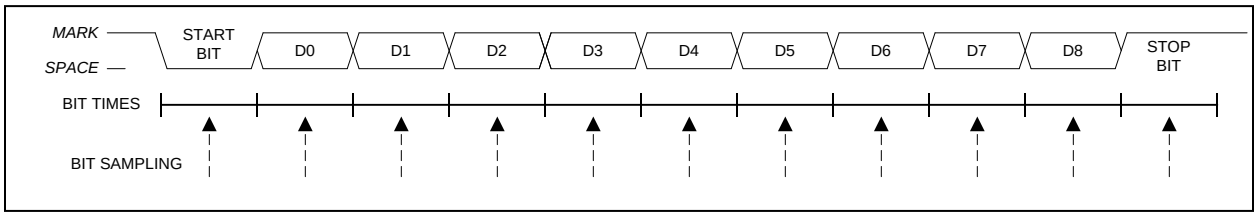
If these conditions are met, the eight bits of data are stored in SBUF, the ninth bit is stored in RB8 and the RI flag is set. If these conditions are not met, SBUF and RB8 will not be loaded and the RI flag will not be set. An interrupt will occur if enabled when either TI or RI are set.

The baud rate in Mode 2 is a direct function of the system clock frequency as follows:

$$\textit{Mode 2 Baud Rate} = 2^{SMOD} * (SYSCLK / 64).$$

The SMOD bit (PCON.7) selects whether to divide SYSCLK by 32 or 64. In the formula, 2 is raised to the power SMOD, resulting in a baud rate of either 1/32 or 1/64 of the system clock frequency. On reset, the SMOD bit is logic 0, thus selecting the lower speed baud rate by default.

Figure 18.6. UART Modes 2 and 3 Timing Diagram



18.1.4. Mode 3: 9-Bit UART, Variable Baud Rate

Mode 3 is the same as Mode 2 in all respects except the baud rate is variable. The baud rate is determined in the same manner as for Mode 1. Mode 3 operation transmits 11 bits: a start bit, 8 data bits (LSB first), a programmable ninth data bit, and a stop bit. Timer 1 or Timer 2 overflows generate the baud rate just as with Mode 1. In summary, Mode 3 transmits using the same protocol as Mode 2 but with Mode 1 baud rate generation.

Figure 18.9. SCON: Serial Port Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SM0	SM1	SM2	REN	TB8	RB8	TI	RI	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0 (bit addressable)	SFR Address: 0x98

Bits7-6: SM0-SM1: Serial Port Operation Mode.
These bits select the Serial Port Operation Mode.

SM0	SM1	Mode
0	0	Mode 0: Synchronous Mode
0	1	Mode 1: 8-Bit UART, Variable Baud Rate
1	0	Mode 2: 9-Bit UART, Fixed Baud Rate
1	1	Mode 3: 9-Bit UART, Variable Baud Rate

Bit5: SM2: Multiprocessor Communication Enable.
The function of this bit is dependent on the Serial Port Operation Mode.
Mode 0: No effect
Mode 1: Checks for valid stop bit.
0: Logic level of stop bit is ignored.
1: RI will only be activated if stop bit is logic level 1.
Mode 2 and 3: Multiprocessor Communications Enable.
0: Logic level of ninth bit is ignored.
1: RI is set and an interrupt is generated only when the ninth bit is logic 1.

Bit4: REN: Receive Enable.
This bit enables/disables the UART receiver.
0: UART reception disabled.
1: UART reception enabled.

Bit3: TB8: Ninth Transmission Bit.
The logic level of this bit will be assigned to the ninth transmission bit in Modes 2 and 3. It is not used in Modes 0 and 1. Set or cleared by software as required.

Bit2: RB8: Ninth Receive Bit.
The bit is assigned the logic level of the ninth bit received in Modes 2 and 3. In Mode 1, if SM2 is logic 0, RB8 is assigned the logic level of the received stop bit. RB8 is not used in Mode 0.

Bit1: TI: Transmit Interrupt Flag.
Set by hardware when a byte of data has been transmitted by the UART (after the 8th bit in Mode 0, or at the beginning of the stop bit in other modes). When the UART interrupt is enabled, setting this bit causes the CPU to vector to the UART interrupt service routine. This bit must be cleared manually by software

Bit0: RI: Receive Interrupt Flag.
Set by hardware when a byte of data has been received by the UART (after the 8th bit in Mode 0, or after the stop bit in other modes – see SM2 bit for exception). When the UART interrupt is enabled, setting this bit causes the CPU to vector to the UART interrupt service routine. This bit must be cleared manually by software.