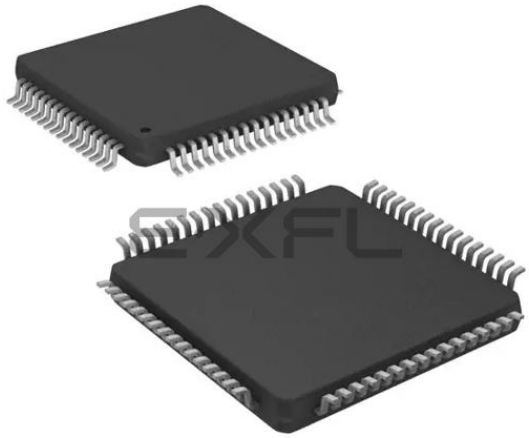


Welcome to E-XFL.COM

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.

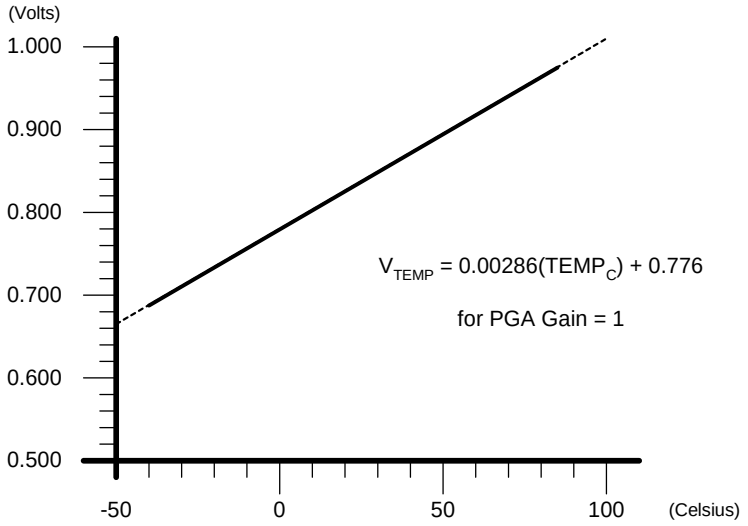
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Not For New Designs
Core Processor	8051
Core Size	8-Bit
Speed	25MHz
Connectivity	EBI/EMI, 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	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4.25K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 8x8b, 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/c8051f021-gqr

Notes

The Temperature Sensor transfer function is shown in Figure 6.2. The output voltage (V_{TEMP}) is the PGA input when the Temperature Sensor is selected by bits AMX0AD3-0 in register AMX0SL; this voltage will be amplified by the PGA according to the user-programmed PGA settings.

Figure 6.2. Temperature Sensor Transfer Function



6.2. ADC Modes of Operation

ADC0 has a maximum conversion speed of 100 ksps. The ADC0 conversion clock is derived from the system clock divided by the value held in the ADCSC bits of register ADC0CF.

6.2.1. Starting a Conversion

A conversion can be initiated in one of four ways, depending on the programmed states of the ADC0 Start of Conversion Mode bits (AD0CM1, AD0CM0) in ADC0CN. Conversions may be initiated by:

1. Writing a '1' to the AD0BUSY bit of ADC0CN;
2. A Timer 3 overflow (i.e. timed continuous conversions);
3. A rising edge detected on the external ADC convert start signal, CNVSTR;
4. A Timer 2 overflow (i.e. timed continuous conversions).

The AD0BUSY bit is set to logic 1 during conversion and restored to logic 0 when conversion is complete. The falling edge of AD0BUSY triggers an interrupt (when enabled) and sets the AD0INT interrupt flag (ADC0CN.5). Converted data is available in the ADC0 data word MSB and LSB registers, ADC0H, ADC0L. Converted data can be either left or right justified in the ADC0H:ADC0L register pair (see example in Figure 6.11) depending on the programmed state of the AD0LJST bit in the ADC0CN register.

When initiating conversions by writing a '1' to AD0BUSY, the AD0INT bit should be polled to determine when a conversion has completed (ADC0 interrupts may also be used). The recommended polling procedure is shown below.

Step 1. Write a '0' to AD0INT;
Step 2. Write a '1' to AD0BUSY;
Step 3. Poll AD0INT for '1';
Step 4. Process ADC0 data.

6.2.3. Settling Time Requirements

When the ADC0 input configuration is changed (i.e., a different MUX or PGA selection is made), a minimum settling (or tracking) time is required before an accurate conversion can be performed. This settling time is determined by the ADC0 MUX resistance, the ADC0 sampling capacitance, any external source resistance, and the accuracy required for the conversion. Figure 6.4 shows the equivalent ADC0 input circuits for both Differential and Single-ended modes. Notice that the equivalent time constant for both input circuits is the same. The required settling time for a given settling accuracy (*SA*) may be approximated by Equation 6.1. When measuring the Temperature Sensor output, R_{TOTAL} reduces to R_{MUX}. 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 settling time requirements. See Table 6.1 on page 74 for minimum settling/tracking time requirements.

Equation 6.1. ADC0 Settling Time Requirements

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

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 ADC0 MUX resistance and any external source resistance.
n is the ADC resolution in bits (10).

Figure 6.4. ADC0 Equivalent Input Circuits

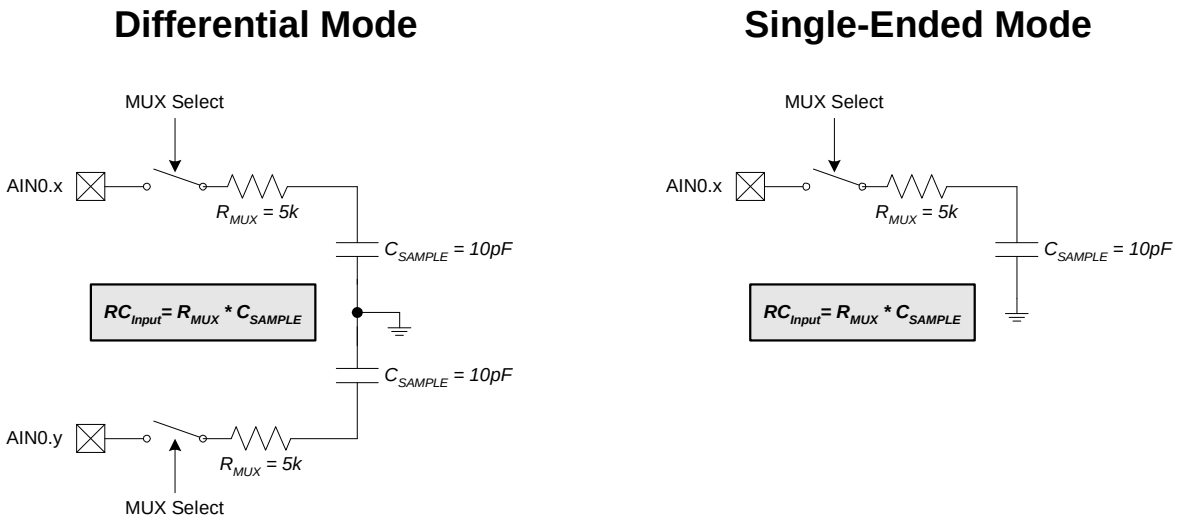
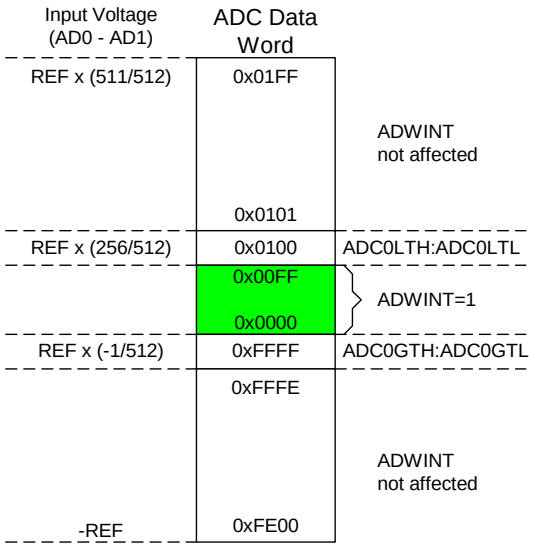
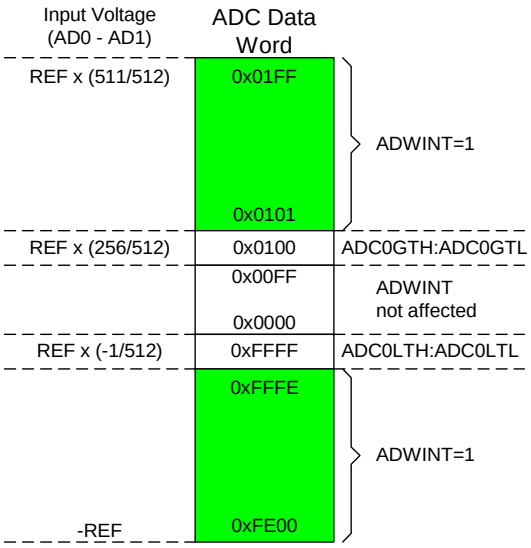


Figure 6.17. 10-Bit ADC0 Window Interrupt Example: Right Justified Differential Data



Given:
AMX0SL = 0x00, AMX0CF = 0x01, ADLJST = 0,
ADC0LTH:ADC0LTL = 0x0100,
ADC0GTH:ADC0GTL = 0xFFFF.
An ADC End of Conversion will cause an ADC Window Compare Interrupt (ADWINT=1) if the resulting ADC Data Word is < 0x0100 and > 0xFFFF. (In two's-complement math, 0xFFFF = -1.)



Given:
AMX0SL = 0x00, AMX0CF = 0x01, ADLJST = 0,
ADC0LTH:ADC0LTL = 0xFFFF,
ADC0GTH:ADC0GTL = 0x0100.
An ADC End of Conversion will cause an ADC Window Compare Interrupt (ADWINT=1) if the resulting ADC Data Word is < 0xFFFF or > 0x0100. (In two's-complement math, 0xFFFF = -1.)

Table 12.4. Interrupt Summary

Interrupt Source	Interrupt Vector	Priority Order	Pending Flag	Bit addressable?	Cleared by HW?	Enable Flag	Priority Control
Reset	0x0000	Top	None	N/A	N/A	Always Enabled	Always Highest
External Interrupt 0 (/INT0)	0x0003	0	IE0 (TCON.1)	Y	Y	EX0 (IE.0)	PX0 (IP.0)
Timer 0 Overflow	0x000B	1	TF0 (TCON.5)	Y	Y	ET0 (IE.1)	PT0 (IP.1)
External Interrupt 1 (/INT1)	0x0013	2	IE1 (TCON.3)	Y	Y	EX1 (IE.2)	PX1 (IP.2)
Timer 1 Overflow	0x001B	3	TF1 (TCON.7)	Y	Y	ET1 (IE.3)	PT1 (IP.3)
UART0	0x0023	4	RI0 (SCON0.0) TI0 (SCON0.1)	Y		ES0 (IE.4)	PS0 (IP.4)
Timer 2 Overflow (or EXF2)	0x002B	5	TF2 (T2CON.7)	Y		ET2 (IE.5)	PT2 (IP.5)
Serial Peripheral Interface	0x0033	6	SPIF (SPI0CN.7)	Y		ESPI0 (EIE1.0)	PSPI0 (EIP1.0)
SMBus Interface	0x003B	7	SI (SMB0CN.3)	Y		ESMB0 (EIE1.1)	PSMB0 (EIP1.1)
ADC0 Window Comparator	0x0043	8	AD0WINT (ADC0CN.2)	Y		EWADC0 (EIE1.2)	PWADC0 (EIP1.2)
Programmable Counter Array	0x004B	9	CF (PCA0CN.7) CCFn (PCA0CN.n)	Y		EPCA0 (EIE1.3)	PPCA0 (EIP1.3)
Comparator 0 Falling Edge	0x0053	10	CP0FIF (CPT0CN.4)			ECP0F (EIE1.4)	PCP0F (EIP1.4)
Comparator 0 Rising Edge	0x005B	11	CP0RIF (CPT0CN.5)			ECP0R (EIE1.5)	PCP0R (EIP1.5)
Comparator 1 Falling Edge	0x0063	12	CP1FIF (CPT1CN.4)			ECP1F (EIE1.6)	PCP1F (EIP1.6)
Comparator 1 Rising Edge	0x006B	13	CP1RIF (CPT1CN.5)			ECP1R (EIE1.7)	PCP1F (EIP1.7)
Timer 3 Overflow	0x0073	14	TF3 (TMR3CN.7)			ET3 (EIE2.0)	PT3 (EIP2.0)
ADC0 End of Conversion	0x007B	15	AD0INT (ADC0CN.5)	Y		EADC0 (EIE2.1)	PADC0 (EIP2.1)
Timer 4 Overflow	0x0083	16	TF4 (T4CON.7)			ET4 (EIE2.2)	PT4 (EIP2.2)
ADC1 End of Conversion	0x008B	17	AD1INT (ADC1CN.5)			EADC1 (EIE2.3)	PADC1 (EIP2.3)
External Interrupt 6	0x0093	18	IE6 (P3IF.5)			EX6 (EIE2.4)	PX6 (EIP2.4)
External Interrupt 7	0x009B	19	IE7 (P3IF.6)			EX7 (EIE2.5)	PX7 (EIP2.5)
UART1	0x00A3	20	RI1 (SCON1.0) TI1 (SCON1.1)			ES1	PS1
External Crystal OSC Ready	0x00AB	21	XTLVLD (OSCXCN.7)			EXVLD (EIE2.7)	PXVLD (EIP2.7)

12.4. Power Management Modes

The CIP-51 core has two software programmable power management modes: Idle and Stop. Idle mode halts the CPU while leaving the external peripherals and internal clocks active. In Stop mode, the CPU is halted, all interrupts and timers (except the Missing Clock Detector) are inactive, and the system clock is stopped. Since clocks are running in Idle mode, power consumption is dependent upon the system clock frequency and the number of peripherals left in active mode before entering Idle. Stop mode consumes the least power. Figure 12.15 describes the Power Control Register (PCON) used to control the CIP-51's power management modes.

Although the CIP-51 has Idle and Stop modes built in (as with any standard 8051 architecture), power management of the entire MCU is better accomplished by enabling/disabling individual peripherals as needed. Each analog peripheral can be disabled when not in use and put into low power mode. Digital peripherals, such as timers or serial buses, draw little power whenever they are not in use. Turning off the Flash memory saves power, similar to entering Idle mode. Turning off the oscillator saves even more power, but requires a reset to restart the MCU.

12.4.1. Idle Mode

Setting the Idle Mode Select bit (PCON.0) causes the CIP-51 to halt the CPU and enter Idle mode as soon as the instruction that sets the bit completes. All internal registers and memory maintain their original data. All analog and digital peripherals can remain active during Idle mode.

Idle mode is terminated when an enabled interrupt or /RST is asserted. The assertion of an enabled interrupt will cause the Idle Mode Selection bit (PCON.0) to be cleared and the CPU to resume operation. The pending interrupt will be serviced and the next instruction to be executed after the return from interrupt (RETI) will be the instruction immediately following the one that set the Idle Mode Select bit. If Idle mode is terminated by an internal or external reset, the CIP-51 performs a normal reset sequence and begins program execution at address 0x0000.

If enabled, the WDT will eventually cause an internal watchdog reset and thereby terminate the Idle mode. This feature protects the system from an unintended permanent shutdown in the event of an inadvertent write to the PCON register. If this behavior is not desired, the WDT may be disabled by software prior to entering the Idle mode if the WDT was initially configured to allow this operation. This provides the opportunity for additional power savings, allowing the system to remain in the Idle mode indefinitely, waiting for an external stimulus to wake up the system. Refer to **Section "13.8. Watchdog Timer Reset" on page 129** for more information on the use and configuration of the WDT.

12.4.2. Stop Mode

Setting the Stop Mode Select bit (PCON.1) causes the CIP-51 to enter Stop mode as soon as the instruction that sets the bit completes. In Stop mode, the CPU and internal oscillator are stopped, effectively shutting down all digital peripherals. Each analog peripheral must be shut down individually prior to entering Stop Mode. Stop mode can only be terminated by an internal or external reset. On reset, the CIP-51 performs the normal reset sequence and begins program execution at address 0x0000.

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

13. RESET SOURCES

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

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

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

The I/O port latches are reset to 0xFF (all logic 1's), activating internal weak pull-ups which take the external I/O pins to a high state. Note that weak pull-ups are disabled during the reset, and enabled when the device exits the reset state. This allows power to be conserved while the part is held in reset. For VDD Monitor resets, the /RST pin is driven low until the end of the VDD reset timeout.

On exit from the reset state, the program counter (PC) is reset, and the system clock defaults to the internal oscillator running at 2 MHz. Refer to Section "**14. OSCILLATORS**" on page **135** for information on selecting and configuring the system clock source. The Watchdog Timer is enabled using its longest timeout interval (see Section "**13.8. Watchdog Timer Reset**" on page **129**). Once the system clock source is stable, program execution begins at location 0x0000.

There are seven sources for putting the MCU into the reset state: power-on/power-fail, external /RST pin, external CNVSTR signal, software command, Comparator0, Missing Clock Detector, and Watchdog Timer. Each reset source is described in the following sections.

Figure 13.1. Reset Sources

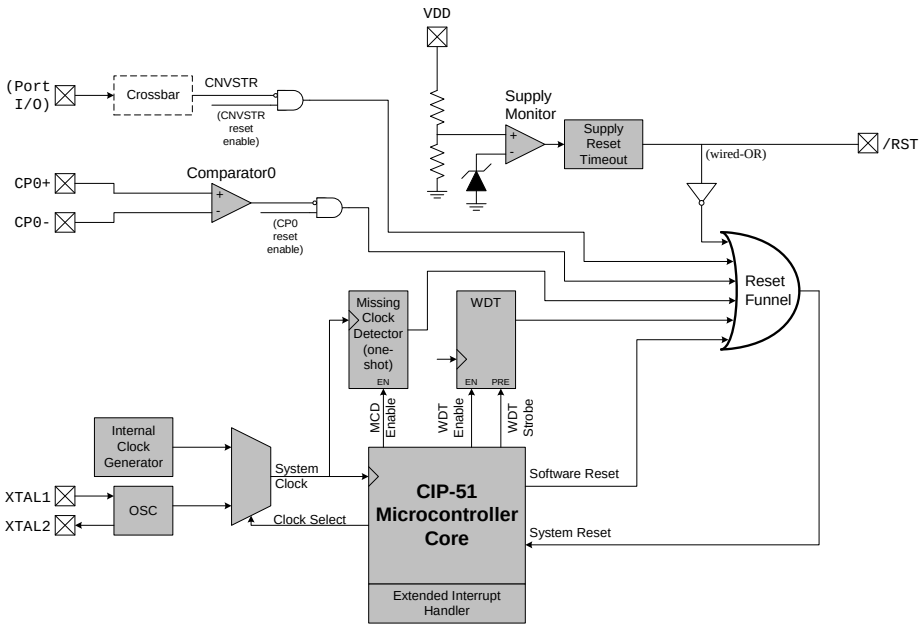


Figure 14.2. OSCICN: Internal Oscillator Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
MSCLKE	-	-	IFRDY	CLKSL	IOSCEN	IFCN1	IFCN0	00010100
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB2

Bit7: MSCLKE: Missing Clock Enable Bit
0: Missing Clock Detector Disabled
1: Missing Clock Detector Enabled; reset triggered if clock is missing for more than 100 µs
Bits6-5: UNUSED. Read = 00b, Write = don't care
Bit4: IFRDY: Internal Oscillator Frequency Ready Flag
0: Internal Oscillator Frequency not running at speed specified by the IFCN bits.
1: Internal Oscillator Frequency running at speed specified by the IFCN bits.
Bit3: CLKSL: System Clock Source Select Bit
0: Uses Internal Oscillator as System Clock.
1: Uses External Oscillator as System Clock.
Bit2: IOSCEN: Internal Oscillator Enable Bit
0: Internal Oscillator Disabled
1: Internal Oscillator Enabled
Bits1-0: IFCN1-0: Internal Oscillator Frequency Control Bits
00: Internal Oscillator typical frequency is 2 MHz.
01: Internal Oscillator typical frequency is 4 MHz.
10: Internal Oscillator typical frequency is 8 MHz.
11: Internal Oscillator typical frequency is 16 MHz.

Table 14.1. Internal Oscillator Electrical Characteristics

VDD = 2.7V to 3.6V; T_a = -40°C to +85°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Internal Oscillator Frequency	OSCICN.[1:0] = 00	1.5	2	2.4	MHz
	OSCICN.[1:0] = 01	3.1	4	4.8	
	OSCICN.[1:0] = 10	6.2	8	9.6	
	OSCICN.[1:0] = 11	12.3	16	19.2	
Internal Oscillator Current Consumption (from VDD)	OSCICN.2 = 1		200		µA

Figure 14.3. OSCXCN: External Oscillator Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
XTLVLD	XOSCMD2	XOSCMD1	XOSCMD0	-	XFCN2	XFCN1	XFCN0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
								0xB1

Bit7: XTLVLD: Crystal Oscillator Valid Flag
(Valid only when XOSCMD = 11x.)
0: Crystal Oscillator is unused or not yet stable
1: Crystal Oscillator is running and stable

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:

XFCN	Crystal (XOSCMD = 11x)	RC (XOSCMD = 10x)	C (XOSCMD = 10x)
000	f < 12 kHz	f < 25 kHz	K Factor = 0.44
001	12 kHz < f ≤ 30 kHz	25 kHz < f ≤ 50 kHz	K Factor = 1.4
010	30 kHz < f ≤ 95 kHz	50 kHz < f ≤ 100 kHz	K Factor = 4.4
011	95 kHz < f ≤ 270 kHz	100 kHz < f ≤ 200 kHz	K Factor = 13
100	270 kHz < f ≤ 720 kHz	200 kHz < f ≤ 400 kHz	K Factor = 38
101	720 kHz < f ≤ 2.2 MHz	400 kHz < f ≤ 800 kHz	K Factor = 100
110	2.2 MHz < f ≤ 6.7 MHz	800 kHz < f ≤ 1.6 MHz	K Factor = 420
111	f > 6.7 MHz	1.6 MHz < f ≤ 3.2 MHz	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

16. EXTERNAL DATA MEMORY INTERFACE AND ON-CHIP XRAM

The C8051F020/1/2/3 MCUs include 4k bytes of on-chip RAM mapped into the external data memory space (XRAM), as well as an External Data Memory Interface which can be used to access off-chip memories and memory-mapped devices connected to the GPIO ports. The external memory space may be accessed using the external move instruction (MOVX) and the data pointer (DPTR), or using the MOVX indirect addressing mode using R0 or R1. If the MOVX instruction is used with an 8-bit address operand (such as @R1), then the high byte of the 16-bit address is provided by the External Memory Interface Control Register (EMI0CN, shown in Figure 16.1). Note: the MOVX instruction can also be used for writing to the FLASH memory. See **Section "15. FLASH MEMORY" on page 139** for details. The MOVX instruction accesses XRAM by default. The EMIF can be configured to appear on the lower I/O ports (P0-P3) or the upper I/O ports (P4-P7).

16.1. Accessing XRAM

The XRAM memory space is accessed using the MOVX instruction. The MOVX instruction has two forms, both of which use an indirect addressing method. The first method uses the Data Pointer, DPTR, a 16-bit register which contains the effective address of the XRAM location to be read or written. The second method uses R0 or R1 in combination with the EMI0CN register to generate the effective XRAM address. Examples of both of these methods are given below.

16.1.1. 16-Bit MOVX Example

The 16-bit form of the MOVX instruction accesses the memory location pointed to by the contents of the DPTR register. The following series of instructions reads the value of the byte at address 0x1234 into the accumulator A:

```
MOV    DPTR, #1234h        ; load DPTR with 16-bit address to read (0x1234)
MOVX   A, @DPTR            ; load contents of 0x1234 into accumulator A
```

The above example uses the 16-bit immediate MOV instruction to set the contents of DPTR. Alternately, the DPTR can be accessed through the SFR registers DPH, which contains the upper 8-bits of DPTR, and DPL, which contains the lower 8-bits of DPTR.

16.1.2. 8-Bit MOVX Example

The 8-bit form of the MOVX instruction uses the contents of the EMI0CN SFR to determine the upper 8-bits of the effective address to be accessed and the contents of R0 or R1 to determine the lower 8-bits of the effective address to be accessed. The following series of instructions read the contents of the byte at address 0x1234 into the accumulator A.

```
MOV    EMI0CN, #12h        ; load high byte of address into EMI0CN
MOV    R0, #34h            ; load low byte of address into R0 (or R1)
MOVX   a, @R0              ; load contents of 0x1234 into accumulator A
```

16.2. Configuring the External Memory Interface

Configuring the External Memory Interface consists of four steps:

1. Select EMIF on Low Ports (P3, P2, P1, and P0) or High Ports (P7, P6, P5, and P4).
2. Select Multiplexed mode or Non-multiplexed mode.
3. Select the memory mode (on-chip only, split mode without bank select, split mode with bank select, or off-chip only).
4. Set up timing to interface with off-chip memory or peripherals.
5. Select the desired output mode for the associated Ports (registers PnMDOUT, P74OUT).

Each of these four steps is explained in detail in the following sections. The Port selection, Multiplexed mode selection, and Mode bits are located in the EMI0CF register shown in Figure 16.2.

16.3. Port Selection and Configuration

The External Memory Interface can appear on Ports 3, 2, 1, and 0 (C8051F020/1/2/3 devices) or on Ports 7, 6, 5, and 4 (C8051F020/2 devices only), depending on the state of the PRTSEL bit (EMI0CF.5). If the lower Ports are selected, the EMIFLE bit (XBR2.1) must be set to a '1' so that the Crossbar will skip over P0.7 (/WR), P0.6 (/RD), and if multiplexed mode is selected P0.5 (ALE). For more information about the configuring the Crossbar, see **Section "17. PORT INPUT/OUTPUT" on page 161**.

The External Memory Interface claims the associated Port pins for memory operations ONLY during the execution of an off-chip MOVX instruction. Once the MOVX instruction has completed, control of the Port pins reverts to the Port latches or to the Crossbar (on Ports 3, 2, 1, and 0). See **Section "17. PORT INPUT/OUTPUT" on page 161** for more information about the Crossbar and Port operation and configuration. **The Port latches should be explicitly configured to 'park' the External Memory Interface pins in a dormant state, most commonly by setting them to a logic 1**.

During the execution of the MOVX instruction, the External Memory Interface will explicitly disable the drivers on all Port pins that are acting as Inputs (Data[7:0] during a READ operation, for example). The Output mode of the Port pins (whether the pin is configured as Open-Drain or Push-Pull) is unaffected by the External Memory Interface operation, and remains controlled by the PnMDOUT registers. See **Section "17. PORT INPUT/OUTPUT" on page 161** for more information about Port output mode configuration.

to a Port pin without assigning RX0 as well. Each combination of enabled peripherals results in a unique device pinout.

All Port pins on Ports 0 through 3 that are not allocated by the Crossbar can be accessed as General-Purpose I/O (GPIO) pins by reading and writing the associated Port Data registers (See Figure 17.10, Figure 17.12, Figure 17.15, and Figure 17.17), a set of SFRs which are both byte- and bit-addressable. The output states of Port pins that are allocated by the Crossbar are controlled by the digital peripheral that is mapped to those pins. Writes to the Port Data registers (or associated Port bits) will have no effect on the states of these pins.

A Read of a Port Data register (or Port bit) will always return the logic state present at the pin itself, regardless of whether the Crossbar has allocated the pin for peripheral use or not. An exception to this occurs during the execution of a *read-modify-write* instruction (ANL, ORL, XRL, CPL, INC, DEC, DJNZ, JBC, CLR, SET, and the bitwise MOV operation). During the *read* cycle of the *read-modify-write* instruction, it is the contents of the Port Data register, not the state of the Port pins themselves, which is read.

Because the Crossbar registers affect the pinout of the peripherals of the device, they are typically configured in the initialization code of the system before the peripherals themselves are configured. Once configured, the Crossbar registers are typically left alone.

Once the Crossbar registers have been properly configured, the Crossbar is enabled by setting XBARE (XBR2.6) to a logic 1. **Until XBARE is set to a logic 1, the output drivers on Ports 0 through 3 are explicitly disabled in order to prevent possible contention on the Port pins while the Crossbar registers and other registers which can affect the device pinout are being written**.

The output drivers on Crossbar-assigned input signals (like RX0, for example) are explicitly disabled; thus the values of the Port Data registers and the PnMDOUT registers have no effect on the states of these pins.

17.1.2. Configuring the Output Modes of the Port Pins

The output drivers on Ports 0 through 3 remain disabled until the Crossbar is enabled by setting XBARE (XBR2.6) to a logic 1.

The output mode of each port pin can be configured as either Open-Drain or Push-Pull; the default state is Open-Drain. In the Push-Pull configuration, writing a logic 0 to the associated bit in the Port Data register will cause the Port pin to be driven to GND, and writing a logic 1 will cause the Port pin to be driven to VDD. In the Open-Drain configuration, writing a logic 0 to the associated bit in the Port Data register will cause the Port pin to be driven to GND, and a logic 1 will cause the Port pin to assume a high-impedance state. The Open-Drain configuration is useful to prevent contention between devices in systems where the Port pin participates in a shared interconnection in which multiple outputs are connected to the same physical wire (like the SDA signal on an SMBus connection).

The output modes of the Port pins on Ports 0 through 3 are determined by the bits in the associated PnMDOUT registers (See Figure 17.11, Figure 17.14, Figure 17.16, and Figure 17.18). For example, a logic 1 in P3MDOUT.7 will configure the output mode of P3.7 to Push-Pull; a logic 0 in P3MDOUT.7 will configure the output mode of P3.7 to Open-Drain. All Port pins default to Open-Drain output.

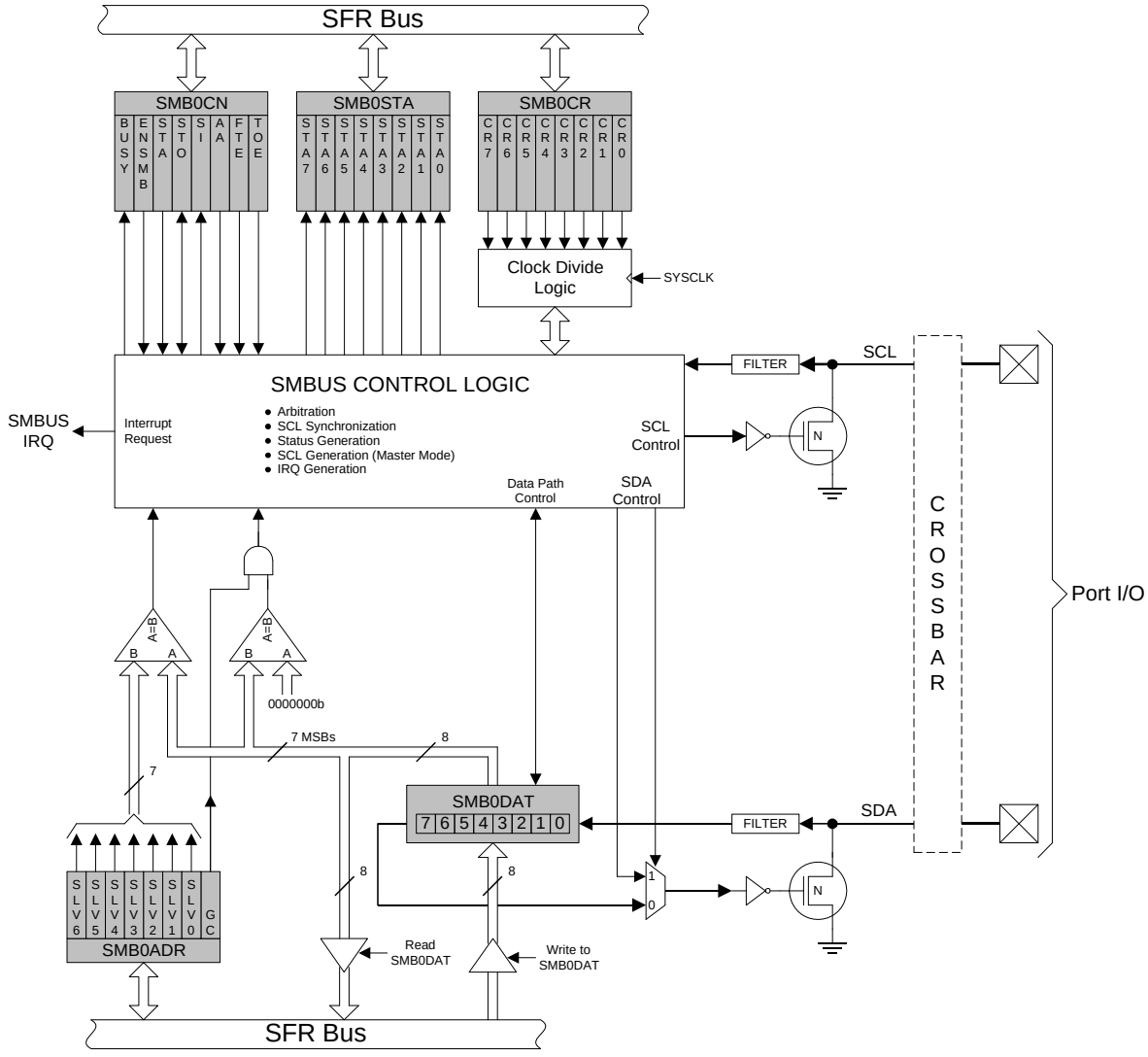
The PnMDOUT registers control the output modes of the port pins regardless of whether the Crossbar has allocated the Port pin for a digital peripheral or not. The exceptions to this rule are: the Port pins connected to SDA, SCL, RX0 (if UART0 is in Mode 0), and RX1 (if UART1 is in Mode 0) are always configured as Open-Drain outputs, regardless of the settings of the associated bits in the PnMDOUT registers.

18. SYSTEM MANAGEMENT BUS / I²C BUS (SMBUS0)

The SMBus0 I/O interface is a two-wire, bi-directional serial bus. SMBus0 is compliant with the System Management Bus Specification, version 1.1, and compatible with the I²C serial bus. Reads and writes to the interface by the system controller are byte oriented with the SMBus0 interface autonomously controlling the serial transfer of the data. Data can be transferred at up to 1/8th 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 available to accommodate devices with different speed capabilities on the same bus.

SMBus0 may operate as a master and/or slave, and may function on a bus with multiple masters. SMBus0 provides control of SDA (serial data), SCL (serial clock) generation and synchronization, arbitration logic, and START/STOP control and generation. SMBus0 is controlled by SFRs as described in **Section 18.4 on page 189**.

Figure 18.1. SMBus0 Block Diagram



18.4.5. Status Register

The SMB0STA Status register holds an 8-bit status code indicating the current state of the SMBus0 interface. There are 28 possible SMBus0 states, each with a corresponding unique status code. The five most significant bits of the status code vary while the three least-significant bits of a valid status code are fixed at zero when SI = '1'. Therefore, all possible status codes are multiples of eight. This facilitates the use of status codes in software as an index used to branch to appropriate service routines (allowing 8 bytes of code to service the state or jump to a more extensive service routine).

For the purposes of user software, the contents of the SMB0STA register is only defined when the SI flag is logic 1. Software should never write to the SMB0STA register; doing so will yield indeterminate results. The 28 SMBus0 states, along with their corresponding status codes, are given in Table 1.1.

Figure 18.12. SMB0STA: SMBus0 Status Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
STA7	STA6	STA5	STA4	STA3	STA2	STA1	STA0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xC1

Bits7-3: STA7-STA3: SMBus0 Status Code.
These bits contain the SMBus0 Status Code. There are 28 possible status codes; each status code corresponds to a single SMBus state. A valid status code is present in SMB0STA when the SI flag (SMB0CN.3) is set to logic 1. The content of SMB0STA is not defined when the SI flag is logic 0. Writing to the SMB0STA register at any time will yield indeterminate results.

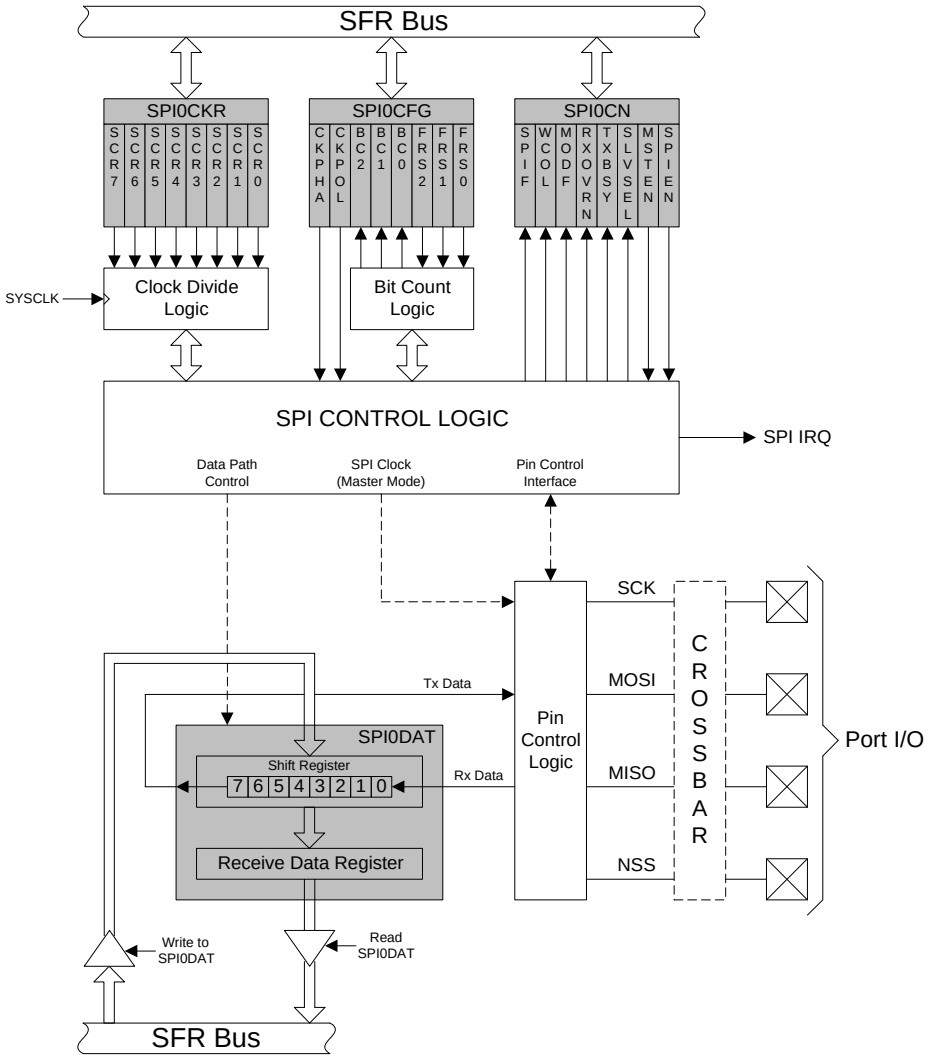
Bits2-0: STA2-STA0: The three least significant bits of SMB0STA are always read as logic 0 when the SI flag is logic 1.

19. SERIAL PERIPHERAL INTERFACE BUS (SPI0)

The Serial Peripheral Interface (SPI0) provides access to a four-wire, full-duplex, serial bus. SPI0 may operate as a master or a slave, and supports the connection of multiple slaves and masters on the same bus. A slave-select input (NSS) is included in the SPI0 interface to select SPI0 as a slave; additional general purpose port I/O can be used as slave-select outputs when SPI0 is operating as a master. Collision detection is provided when two or more masters attempt a data transfer at the same time. 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 1/4 the system clock frequency. This is provided that the master issues SCK, NSS, and the serial input data synchronously with the system clock.

Figure 19.1. SPI Block Diagram



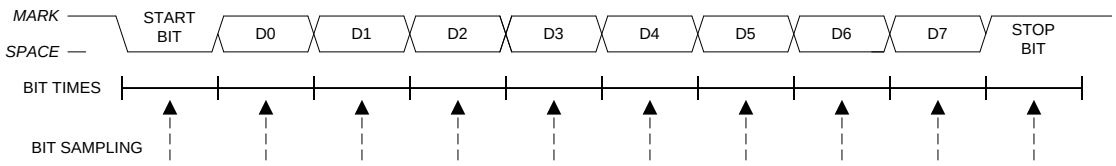
21.1.2. Mode 1: 8-Bit UART, Variable Baud Rate

Mode 1 provides standard asynchronous, full duplex communication using a total of 10 bits per data byte: one start bit, eight data bits (LSB first), and one stop bit. Data are transmitted from the TX1 pin and received at the RX1 pin. On receive, the eight data bits are stored in SBUF1 and the stop bit goes into RB81 (SCON1.2).

Data transmission begins when an instruction writes a data byte to the SBUF1 register. The TI1 Transmit Interrupt Flag (SCON1.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN1 Receive Enable bit (SCON1.4) is set to logic 1. After the stop bit is received, the data byte will be loaded into the SBUF1 receive register if the following conditions are met: RI1 must be logic 0, and if SM21 is logic 1, the stop bit must be logic 1.

If these conditions are met, the eight bits of data are stored in SBUF1, the stop bit is stored in RB81 and the RI1 flag is set. If these conditions are not met, SBUF1 and RB81 will not be loaded and the RI1 flag will not be set. An interrupt will occur if enabled when either TI1 or RI1 is set.

Figure 21.4. UART1 Mode 1 Timing Diagram



The baud rate generated in Mode 1 is a function of timer overflow, shown in Equation 21.1 and Equation 21.2. UART1 can use Timer 1 operating in *8-Bit Auto-Reload Mode*, or Timer 4 operating in *Baud Rate Generator Mode* to generate the baud rate (note that the TX and RX clocks are selected separately). On each timer overflow event (a rollover from all ones - (0xFF for Timer 1, 0xFFFF for Timer 4) - to zero) a clock is sent to the baud rate logic.

Timer 4 is selected as TX and/or RX baud clock source by setting the TCLK1 (T4CON.4) and/or RCLK1 (T4CON.5) bits, respectively (see **Section "22. TIMERS" on page 225** for complete timer configuration details). When either TCLK1 or RCLK1 is set to logic 1, Timer 4 is forced into *Baud Rate Generator Mode*, with SYSCLK / 2 as its clock source. If TCLK1 and/or RCLK1 is logic 0, Timer 1 acts as the baud clock source for the TX and/or RX circuits, respectively.

The Mode 1 baud rate equations are shown below, where T1M is the Timer 1 Clock Select bit (register CKCON), TH1 is the 8-bit reload register for Timer 1, SMOD1 is the UART1 baud rate doubler (register PCON), and [RCAP4H , RCAP4L] is the 16-bit reload register for Timer 4.

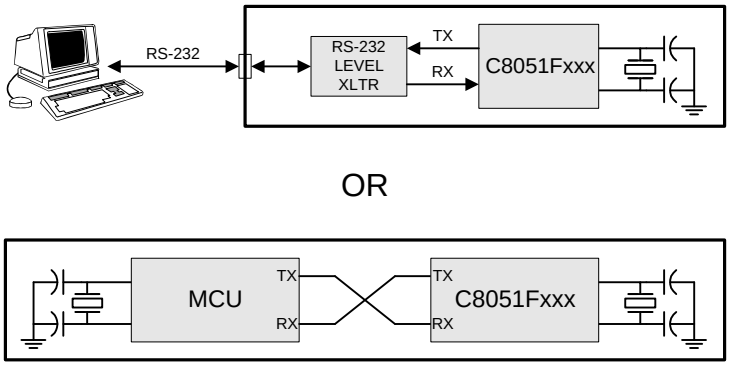
Equation 21.1. Mode 1 Baud Rate using Timer 1

$$BaudRate = \left(\frac{2^{SMOD1}}{32}\right) \times \left(\frac{SYSCLK \times 12^{(T1M-1))}}{(256 - TH1)}\right)$$

Equation 21.2. Mode 1 Baud Rate using Timer 4

$$BaudRate = \frac{SYSCLK}{[32 \times (65536 - [RCAP4H, RCAP4L])]}$$

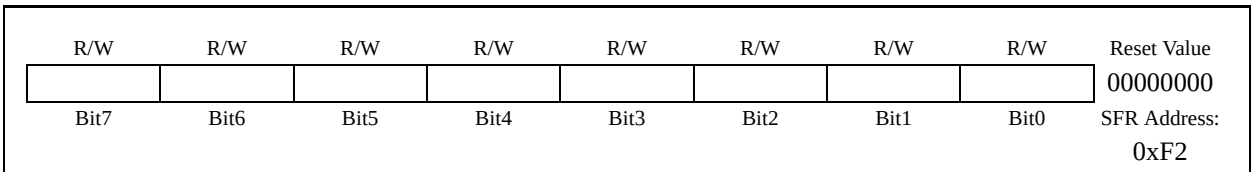
Figure 21.6. UART Modes 1, 2, and 3 Interconnect Diagram



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

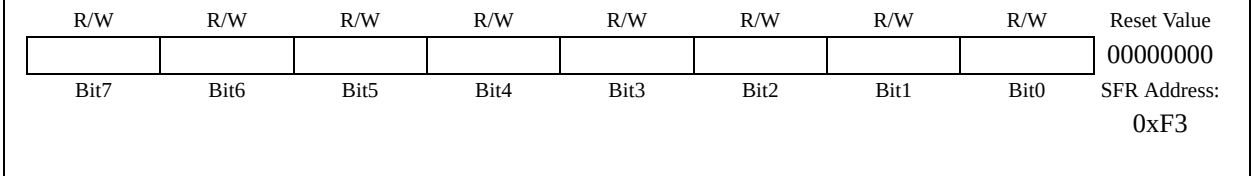
Mode 3 uses the Mode 2 transmission protocol with the Mode 1 baud rate generation. Mode 3 operation transmits 11 bits: a start bit, 8 data bits (LSB first), a programmable ninth data bit, and a stop bit. The baud rate is derived from Timer 1 or Timer 4 overflows, as defined by Equation 21.1 and Equation 21.2. Multiprocessor communications and hardware address recognition are supported, as described in **Section 21.2**.

Figure 21.9. SBUF1: UART1 Data Buffer Register



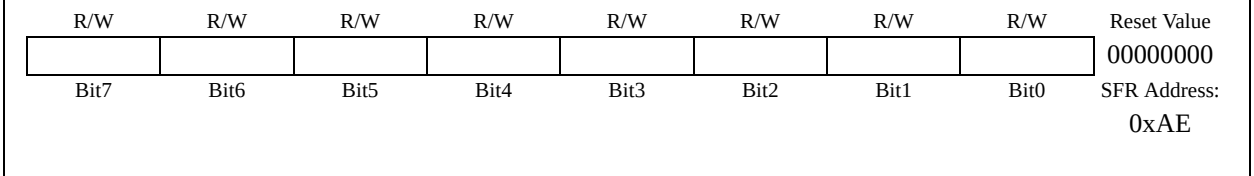
Bits7-0: SBUF1.[7:0]: UART1 Buffer Bits 7-0 (MSB-LSB)
This SFR accesses two registers; a transmit shift register and a receive latch register. When data is written to SBUF1, it goes to the transmit shift register and is held for serial transmission. Writing a byte to SBUF1 is what initiates the transmission. A read of SBUF1 returns the contents of the receive latch.

Figure 21.10. SADDR1: UART1 Slave Address Register



Bits7-0: SADDR1.[7:0]: UART1 Slave Address
The contents of this register are used to define the UART1 slave address. Register SADEN1 is a bit mask to determine which bits of SADDR1 are checked against a received address: corresponding bits set to logic 1 in SADEN1 are checked; corresponding bits set to logic 0 are “don’t cares”.

Figure 21.11. SADEN1: UART1 Slave Address Enable Register



Bits7-0: SADEN1.[7:0]: UART1 Slave Address Enable
Bits in this register enable corresponding bits in register SADDR1 to determine the UART1 slave address.
0: Corresponding bit in SADDR1 is a “don’t care”.
1: Corresponding bit in SADDR1 is checked against a received address.

Figure 22.28. T4CON: Timer 4 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
TF4	EXF4	RCLK1	TCLK1	EXEN4	TR4	C/T4	CP/RL4	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xC9

Bit7: TF4: Timer 4 Overflow Flag.
Set by hardware when Timer 4 overflows. When the Timer 4 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 4 interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software. TF4 will not be set when RCLK1 and/or TCLK1 are logic 1.

Bit6: EXF4: Timer 4 External Flag.
Set by hardware when either a capture or reload is caused by a high-to-low transition on the T4EX input pin and EXEN4 is logic 1. When the Timer 4 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 4 Interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by software.

Bit5: RCLK1: Receive Clock Flag for UART1.
Selects which timer is used for the UART1 receive clock in modes 1 or 3.
0: Timer 1 overflows used for receive clock.
1: Timer 4 overflows used for receive clock.

Bit4: TCLK1: Transmit Clock Flag for UART1.
Selects which timer is used for the UART1 transmit clock in modes 1 or 3.
0: Timer 1 overflows used for transmit clock.
1: Timer 4 overflows used for transmit clock.

Bit3: EXEN4: Timer 4 External Enable.
Enables high-to-low transitions on T4EX to trigger captures or reloads when Timer 4 is not operating in Baud Rate Generator mode.
0: High-to-low transitions on T4EX ignored.
1: High-to-low transitions on T4EX cause a capture or reload.

Bit2: TR4: Timer 4 Run Control.
This bit enables/disables Timer 4.
0: Timer 4 disabled.
1: Timer 4 enabled.

Bit1: C/T4: Counter/Timer Select.
0: Timer Function: Timer 4 incremented by clock defined by T4M (CKCON.6).
1: Counter Function: Timer 4 incremented by high-to-low transitions on external input pin (T2).

Bit0: CP/RL4: Capture/Reload Select.
This bit selects whether Timer 4 functions in capture or auto-reload mode. EXEN4 must be logic 1 for high-to-low transitions on T4EX to be recognized and used to trigger captures or reloads. If RCLK1 or TCLK1 is set, this bit is ignored and Timer 4 will function in auto-reload mode.
0: Auto-reload on Timer 4 overflow or high-to-low transition at T4EX (EXEN4 = 1).
1: Capture on high-to-low transition at T4EX (EXEN4 = 1).

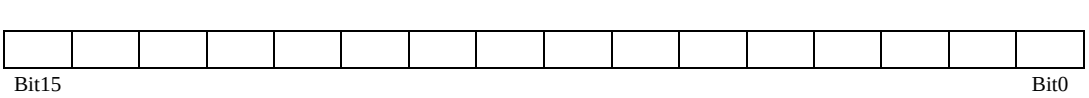
24. JTAG (IEEE 1149.1)

Each MCU has an on-chip JTAG interface and logic to support boundary scan for production and in-system testing, Flash read/write operations, and non-intrusive in-circuit debug. The JTAG interface is fully compliant with the IEEE 1149.1 specification. Refer to this specification for detailed descriptions of the Test Interface and Boundary-Scan Architecture. Access of the JTAG Instruction Register (IR) and Data Registers (DR) are as described in the Test Access Port and Operation of the IEEE 1149.1 specification.

The JTAG interface is accessed via four dedicated pins on the MCU: TCK, TMS, TDI, and TDO.

Through the 16-bit JTAG Instruction Register (IR), any of the seven instructions shown in Figure 24.1 can be commanded. There are three DR's associated with JTAG Boundary-Scan, and four associated with Flash read/write operations on the MCU.

Figure 24.1. IR: JTAG Instruction Register



IR Value	Instruction	Description
0x0000	EXTEST	Selects the Boundary Data Register for control and observability of all device pins
0x0002	SAMPLE/ PRELOAD	Selects the Boundary Data Register for observability and presetting the scan-path latches
0x0004	IDCODE	Selects device ID Register
0xFFFF	BYPASS	Selects Bypass Data Register
0x0082	Flash Control	Selects FLASHCON Register to control how the interface logic responds to reads and writes to the FLASHDAT Register
0x0083	Flash Data	Selects FLASHDAT Register for reads and writes to the Flash memory
0x0084	Flash Address	Selects FLASHADR Register which holds the address of all Flash read, write, and erase operations