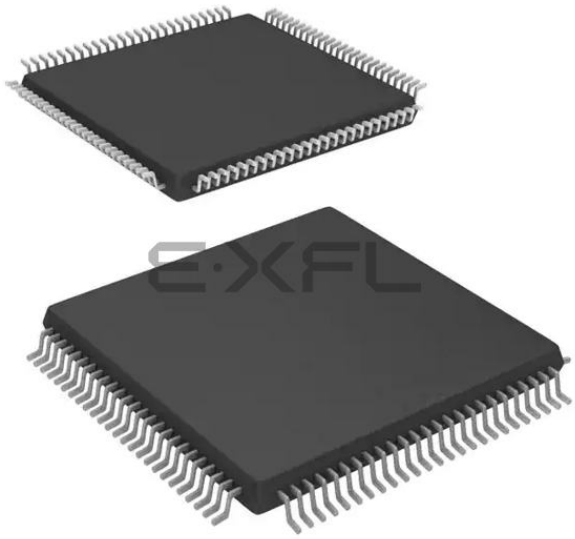


Welcome to E-XFL.COM

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	CANbus, EBI/EMI, SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, Temp Sensor, WDT
Number of I/O	64
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 13x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	100-TQFP
Supplier Device Package	100-TQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f044-gqr

1.4. Programmable Digital I/O and Crossbar

The standard 8051 Ports (0, 1, 2, and 3) are available on the MCUs. The C8051F040/2/4/6 have 4 additional 8-bit ports (4, 5, 6, and 7) for a total of 64 general-purpose I/O Ports. The Ports behave like the standard 8051 with a few enhancements.

Each port pin can be configured as either a push-pull or open-drain output. Also, the "weak pullups" which are normally fixed on an 8051 can be globally disabled, providing additional power saving capabilities for low-power applications.

Perhaps the most unique enhancement is the Digital Crossbar. This is essentially a large digital switching network that allows mapping of internal digital system resources to Port I/O pins on P0, P1, P2, and P3 (See Figure 1.9). Unlike microcontrollers with standard multiplexed digital I/O ports, all combinations of functions are supported with all package options offered.

The on-chip counter/timers, serial buses, HW interrupts, ADC Start of Conversion input, comparator outputs, and other digital signals in the controller can be configured to appear on the Port I/O pins specified in the Crossbar Control registers. This allows the user to select the exact mix of general purpose Port I/O and digital resources needed for the particular application.

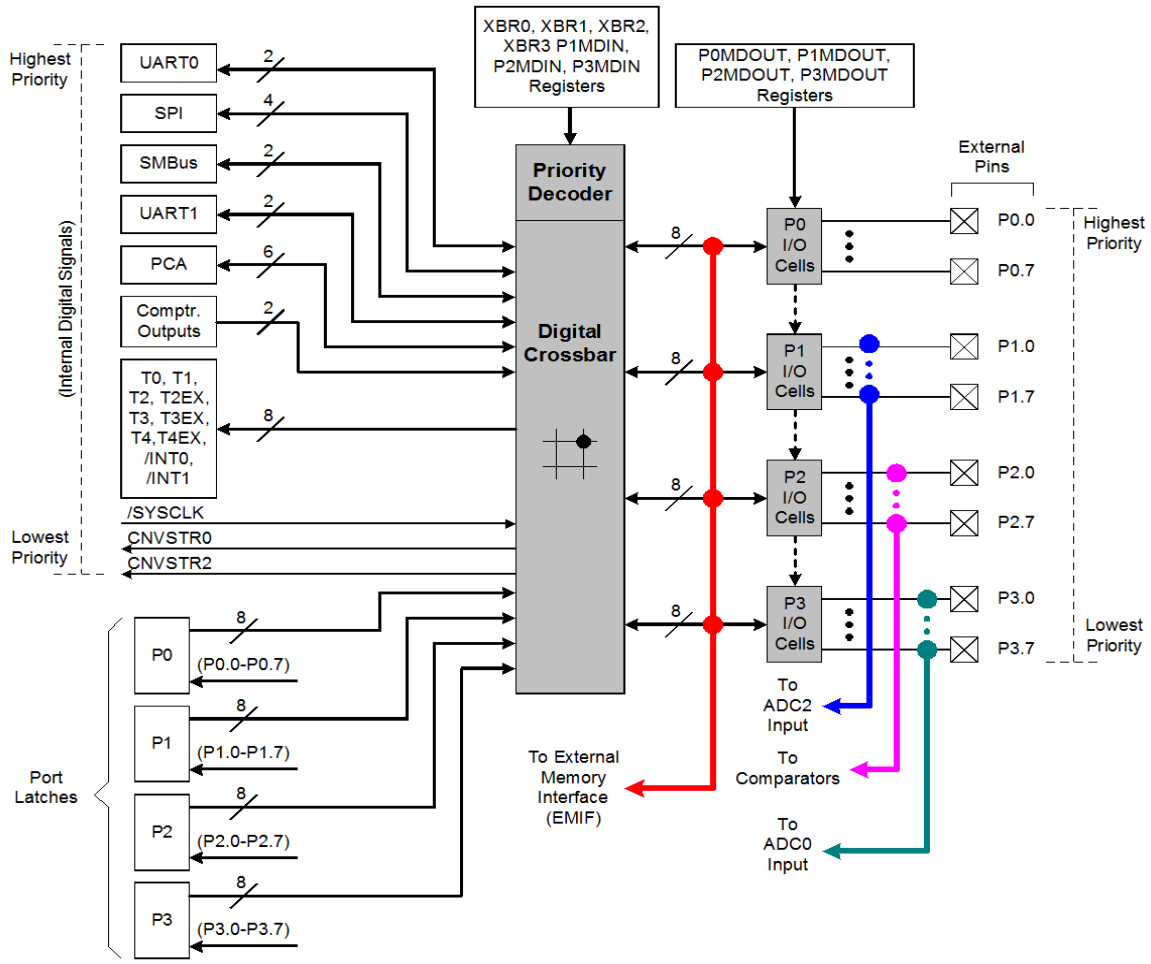


Figure 1.9. Digital Crossbar Diagram

2. Absolute Maximum Ratings

Table 2.1. Absolute Maximum Ratings*

Parameter	Conditions	Min	Typ	Max	Units
Ambient temperature under bias		–55	—	125	°C
Storage Temperature		–65	—	150	°C
Voltage on any Pin (except V_{DD}, Port I/O, and JTAG pins) with respect to DGND		–0.3	—	V_{DD} + 0.3	V
Voltage on any Port I/O Pin, /RST, and JTAG pins with respect to DGND		–0.3	—	5.8	V
Voltage on V_{DD} with respect to DGND		–0.3	—	4.2	V
Maximum Total current through V_{DD}, AV+, DGND, and AGND		—	—	800	mA
Maximum output current sunk by any Port pin		—	—	100	mA
Maximum output current sunk by any other I/O pin		—	—	50	mA
Maximum output current sourced by any Port pin		—	—	100	mA
Maximum output current sourced by any other I/O pin		—	—	50	mA

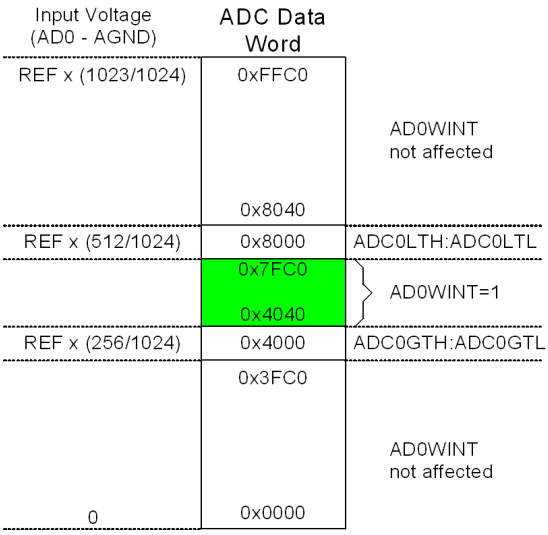
***Note:** Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the devices at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

Due to special I/O design requirements of the High Voltage Difference Amplifier, undue electrical over-voltage stress (i.e., ESD) experienced by these pads may result in impedance degradation of these inputs (HVAIN+ and HVAIN–). For this reason, care should be taken to ensure proper handling and use as typically required to prevent ESD damage to electrostatically sensitive CMOS devices (e.g., static-free workstations, use of grounding straps, over-voltage protection in end-applications, etc.)

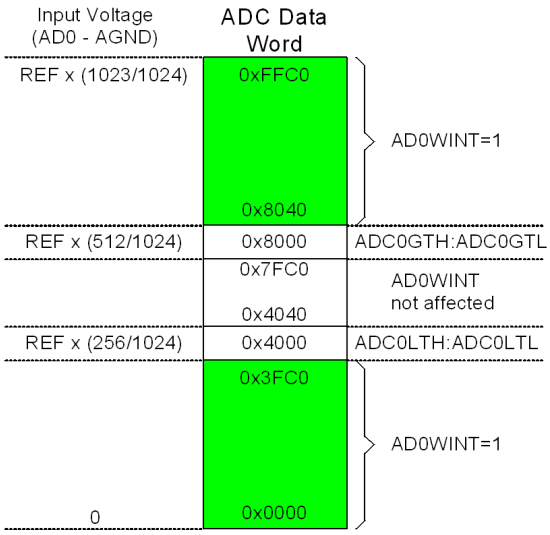
Table 5.1. AMUX Selection Chart (AMX0AD3–0 and AMX0CF3–0 bits)

		AMX0AD3-0								
		0000	0001	0010	0011	0100	0101	0110	0111	1xxx
AMX0CF Bits 3-0	0000	AIN0.0	AIN0.1	AIN0.2	AIN0.3	HVDA	AGND	P3EVEN	P3ODD	TEMP SENSOR
	0001	+(AIN0.0) -(AIN0.1)		AIN0.2	AIN0.3	HVDA	AGND	P3EVEN	P3ODD	TEMP SENSOR
	0010	AIN0.0	AIN0.1	+(AIN0.2) -(AIN0.3)		HVDA	AGND	P3EVEN	P3ODD	TEMP SENSOR
	0011	+(AIN0.0) -(AIN0.1)		+(AIN0.2) -(AIN0.3)		HVDA	AGND	P3EVEN	P3ODD	TEMP SENSOR
	0100	AIN0.0	AIN0.1	AIN0.2	AIN0.3	+(HVDA) -(HVREF)		P3EVEN	P3ODD	TEMP SENSOR
	0101	+(AIN0.0) -(AIN0.1)		AIN0.2	AIN0.3	+(HVDA) -(HVREF)		P3EVEN	P3ODD	TEMP SENSOR
	0110	AIN0.0	AIN0.1	+(AIN0.2) -(AIN0.3)		+(HVDA) -(HVREF)		P3EVEN	P3ODD	TEMP SENSOR
	0111	+(AIN0.0) -(AIN0.1)		+(AIN0.2) -(AIN0.3)		+(HVDA) -(HVREF)		P3EVEN	P3ODD	TEMP SENSOR
	1000	AIN0.0	AIN0.1	AIN0.2	AIN0.3	HVDA	AGND	+P3EVEN -P3ODD		TEMP SENSOR
	1001	+(AIN0.0) -(AIN0.1)		AIN0.2	AIN0.3	HVDA	AGND	+P3EVEN -P3ODD		TEMP SENSOR
	1010	AIN0.0	AIN0.1	+(AIN0.2) -(AIN0.3)		HVDA	AGND	+P3EVEN -P3ODD		TEMP SENSOR
	1011	+(AIN0.0) -(AIN0.1)		+(AIN0.2) -(AIN0.3)		HVDA	AGND	+P3EVEN -P3ODD		TEMP SENSOR
	1100	AIN0.0	AIN0.1	AIN0.2	AIN0.3	+(HVDA) -(HVREF)		+P3EVEN -P3ODD)		TEMP SENSOR
	1101	+(AIN0.0) -(AIN0.1)		AIN0.2	AIN0.3	+(HVDA) -(HVREF)		+P3EVEN -P3ODD		TEMP SENSOR
	1110	AIN0.0	AIN0.1	+(AIN0.2) -(AIN0.3)		+(HVDA) -(HVREF)		+P3EVEN -P3ODD		TEMP SENSOR
	1111	+(AIN0.0) -(AIN0.1)		+(AIN0.2) -(AIN0.3)		+(HVDA) -(HVREF)		+P3EVEN -P3ODD		TEMP SENSOR

Note: “P3EVEN” denotes even numbered and “P3ODD” odd numbered Port 3 pins selected in the AMX0PRT register.



Given:
AMX0SL = 0x00, AMX0CF = 0x00, ADLJST = 1,
ADC0LTH:ADC0LTL = 0x8000,
ADC0GTH:ADC0GTL = 0x4000.
An ADC End of Conversion will cause an ADC Window Compare Interrupt (ADWINT=1) if the resulting ADC Data Word is < 0x8000 and > 0x4000.
Given:



AMX0SL = 0x00, AMX0CF = 0x00, ADLJST = 1,
ADC0LTH:ADC0LTL = 0x4000,
ADC0GTH:ADC0GTL = 0x8000.
An ADC End of Conversion will cause an ADC Window Compare Interrupt (ADWINT=1) if the resulting ADC Data Word is < 0x4000 or > 0x8000.

Figure 6.10. 10-Bit ADC0 Window Interrupt Example: Left Justified Single-Ended Data

Table 7.2. ADC2 Electrical Characteristics

V_{DD} = 3.0 V, AV+ = 3.0 V, V_{REF2} = 2.40 V (REFBE = 0), PGA2 = 1, –40 to +85 °C unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
DC Accuracy					
Resolution			8		bits
Integral Nonlinearity		—	—	±1	LSB
Differential Nonlinearity	Guaranteed Monotonic	—	—	±1	LSB
Offset Error		—	0.5±0.3	—	LSB
Full Scale Error	Differential mode	—	–1±0.2	—	LSB
Dynamic Performance (10 kHz sine-wave input, 0 to 1 dB below Full Scale, 500 ksps)					
Signal-to-Noise Plus Distortion		45	47	—	dB
Total Harmonic Distortion	Up to the 5th harmonic	—	–51	—	dB
Spurious-Free Dynamic Range		—	52	—	dB
Conversion Rate					
SAR Conversion Clock Frequency		—	—	6	MHz
Conversion Time in SAR Clocks		8	—	—	clocks
Track/Hold Acquisition Time		300	—	—	ns
Throughput Rate		—	—	500	ksps
Analog Inputs					
Input Voltage Range	Single-ended	0	—	VREF	V
Common Mode Range		0	—	AV+	V
Input Capacitance		—	5	—	pF
Power Specifications					
Power Supply Current (AV+ supplied to ADC2)	Operating Mode, 500 ksps	—	420	900	µA
Power Supply Rejection		—	±0.3	—	mV/V

Table 12.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
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 DPTR 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 is 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
Program Branching			
ACALL addr11	Absolute subroutine call	2	3
LCALL addr16	Long subroutine call	3	4
RET	Return from subroutine	1	5
RETI	Return from interrupt	1	5
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (relative address)	2	3
JMP @A+DPTR	Jump indirect relative to DPTR	1	3
JZ rel	Jump if A equals zero	2	2/3

SFR Definition 12.12. IP: Interrupt Priority

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	PT2	PS0	PT1	PX1	PT0	PX0	11000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Bit Addressable

SFR Address: 0xB8
SFR Page: All Pages

Bits7-6: UNUSED. Read = 11b, Write = don't care.
Bit5: PT2: Timer 2 Interrupt Priority Control.
This bit sets the priority of the Timer 2 interrupt.
0: Timer 2 interrupt priority set to low priority level.
1: Timer 2 interrupts set to high priority level.
Bit4: PS0: UART0 Interrupt Priority Control.
This bit sets the priority of the UART0 interrupt.
0: UART0 interrupt priority set to low priority level.
1: UART0 interrupts set to high priority level.
Bit3: PT1: Timer 1 Interrupt Priority Control.
This bit sets the priority of the Timer 1 interrupt.
0: Timer 1 interrupt priority set to low priority level.
1: Timer 1 interrupts set to high priority level.
Bit2: PX1: External Interrupt 1 Priority Control.
This bit sets the priority of the External Interrupt 1 interrupt.
0: External Interrupt 1 priority set to low priority level.
1: External Interrupt 1 set to high priority level.
Bit1: PT0: Timer 0 Interrupt Priority Control.
This bit sets the priority of the Timer 0 interrupt.
0: Timer 0 interrupt priority set to low priority level.
1: Timer 0 interrupt set to high priority level.
Bit0: PX0: External Interrupt 0 Priority Control.
This bit sets the priority of the External Interrupt 0 interrupt.
0: External Interrupt 0 priority set to low priority level.
1: External Interrupt 0 set to high priority level.

14.4. External Crystal Example

If a crystal or ceramic resonator is used as an external oscillator source for the MCU, the circuit should be configured as shown in Figure 14.1, Option 1. The External Oscillator Frequency Control value (XFCN) should be chosen from the Crystal column of the table in SFR Definition 14.4 (OSCXCN register). For example, an 11.0592 MHz crystal requires an XFCN setting of 111b.

When the crystal oscillator is enabled, the oscillator amplitude detection circuit requires a settle time to achieve proper bias. Introducing a delay of at least 1 ms between enabling the oscillator and checking the XTLVLD bit will prevent a premature switch to the external oscillator as the system clock. Switching to the external oscillator before the crystal oscillator has stabilized can result in unpredictable behavior. The recommended procedure is:

Step 1. Enable the external oscillator in crystal oscillator mode.
Step 2. Wait at least 1 ms.
Step 3. Poll for XTLVLD => '1'.
Step 4. Switch the system clock to the external oscillator.

Note: Tuning-fork crystals may require additional settling time before XTLVLD returns a valid result.

The capacitors shown in the external crystal configuration provide the load capacitance required by the crystal for correct oscillation. These capacitors are "in series" as seen by the crystal and "in parallel" with the stray capacitance of the XTAL1 and XTAL2 pins.

Note: The load capacitance depends upon the crystal and the manufacturer. Please refer to the crystal data sheet when completing these calculations.

For example, a tuning-fork crystal of 32.768 kHz with a recommended load capacitance of 12.5 pF should use the configuration shown in Figure 14.1, Option 1. The total value of the capacitors and the stray capacitance of the XTAL pins should equal 25 pF. With a stray capacitance of 3 pF per pin, the 22 pF capacitors yield an equivalent capacitance of 12.5 pF across the crystal, as shown in Figure 14.2.

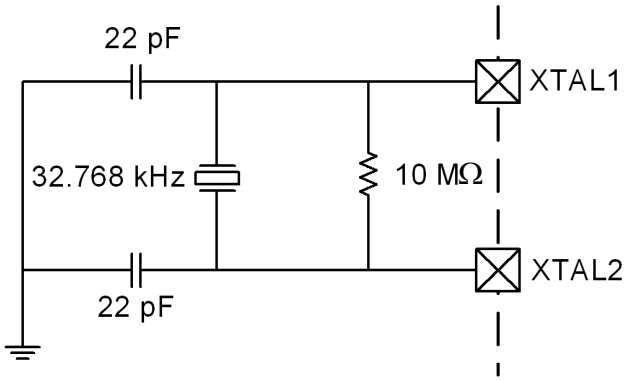


Figure 14.2. 32.768 kHz External Crystal Example

Important Note on External Crystals: Crystal oscillator circuits are quite sensitive to PCB layout. The crystal should be placed as close as possible to the XTAL pins on the device. The traces should be as short as possible and shielded with ground plane from any other traces which could introduce noise or interference.

3/4/5) and all locations above 0x8000 (C8051F046/7) are reserved. Flash writes and erases targeting the reserved area should be avoided.

Table 15.1. Flash Electrical Characteristics

V_{DD} = 2.7 to 3.6 V; T_a = –40 to +85 °C

Parameter	Conditions	Min	Typ	Max	Units
Flash Size[1]	C8051F040/1/2/3/4/5 C8051F046/7		65664[2] 32896		Bytes
Endurance		20 k	100 k	—	Erase/Write
Erase Cycle Time		10	12	14	ms
Write Cycle Time		40	50	60	µs

Notes:
1. Includes 128-byte scratchpad.
2. 512 bytes at locations 0xFE00 to 0xFFFF are reserved.

15.2. Non-volatile Data Storage

The Flash memory can be used for non-volatile data storage as well as program code. This allows data such as calibration coefficients to be calculated and stored at run time. Data is written using the MOVX write instruction (as described in the previous section) and read using the MOVC instruction.

An additional 128-byte sector of Flash memory is included for non-volatile data storage. Its smaller sector size makes it particularly well suited as general purpose, non-volatile scratchpad memory. Even though Flash memory can be written a single byte at a time, an entire sector must be erased first. In order to change a single byte of a multi-byte data set, the data must be moved to temporary storage. The 128-byte sector-size facilitates updating data without wasting program memory or RAM space. The 128-byte sector is double-mapped over the 64k byte Flash memory; its address ranges from 0x00 to 0x7F (see Figure 15.1). To access this 128-byte sector, the SFLE bit in PSCTL must be set to logic 1. Code execution from this 128-byte scratchpad sector is not permitted.

15.3. Security Options

The CIP-51 provides security options to protect the Flash memory from inadvertent modification by software as well as prevent the viewing of proprietary program code and constants. The Program Store Write Enable (PSCTL.0) and the Program Store Erase Enable (PSCTL.1) bits protect the Flash memory from accidental modification by software. These bits must be explicitly set to logic 1 before software can write or erase the Flash memory. Additional security features prevent proprietary program code and data constants from being read or altered across the JTAG interface or by software running on the system controller.

A set of security lock bytes stored at 0xFDFE and 0xFDFF (C8051F040/1/2/3/4/5) and at 0x7FFE and 0x7FFF (C8051F046/7) protect the Flash program memory from being read or altered across the JTAG interface. Each bit in a security lock-byte protects one 8k-byte block of memory. Clearing a bit to logic 0 in a Read Lock Byte prevents the corresponding block of Flash memory from being read across the JTAG interface. Clearing a bit in the Write/Erase Lock Byte protects the block from JTAG erasures and/or writes.

The Read Lock Byte is at locations 0xFDFF (C8051F040/1/2/3/4/5) and 0x7FFF (C8051F046/7). The Write/Erase Lock Byte is located at 0xFDFE (C8051F040/1/2/3/4/5) and 0x7FFE (C8051F046/7). Figure 15.1 shows the location and bit definitions of the security bytes. **The 512-byte sector containing the lock bytes can be written to, but not erased by software**. An attempted read of a read-locked byte returns undefined data. Debugging code in a read-locked sector is not possible through the JTAG interface.

SFR Definition 15.3. PSCTL: Program Store Read/Write Control

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
-	-	-	-	-	SFLE	PSEE	PSWE	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:

SFR Address: 0x8F
SFR Page: 0

Bits7-3: UNUSED. Read = 00000b, Write = don't care.

Bit2: SFLE: Scratchpad Flash Memory Access Enable
When this bit is set, Flash reads and writes from user software are directed to the 128-byte Scratchpad Flash sector. When SFLE is set to logic 1, Flash accesses out of the address range 0x00-0x7F should not be attempted. Reads/Writes out of this range will yield undefined results.
0: Flash access from user software directed to the Program/Data Flash sector.
1: Flash access from user software directed to the 128 byte Scratchpad sector.

Bit1: PSEE: Program Store Erase Enable.
Setting this bit allows an entire page of the Flash program memory to be erased provided the PSWE bit is also set. After setting this bit, a write to Flash memory using the MOVX instruction will erase the entire page that contains the location addressed by the MOVX instruction. The value of the data byte written does not matter. **Note: The Flash page containing the Read Lock Byte and Write/Erase Lock Bytes cannot be erased by software.**
0: Flash program memory erasure disabled.
1: Flash program memory erasure enabled.

Bit0: PSWE: Program Store Write Enable.
Setting this bit allows writing a byte of data to the Flash program memory using the MOVX write instruction. The location must be erased prior to writing data.
0: Write to Flash program memory disabled. MOVX write operations target External RAM.
1: Write to Flash program memory enabled. MOVX write operations target Flash memory.

16.5. Memory Mode Selection

The external data memory space can be configured in one of four modes, shown in Figure 16.3, based on the EMIF Mode bits in the EMI0CF register (SFR Definition 16.2). These modes are summarized below. More information about the different modes can be found in **Section "16.6. Timing" on page 194**.

16.5.1. Internal XRAM Only

When EMI0CF.[3:2] are set to '00', all MOVX instructions will target the internal XRAM space on the device. Memory accesses to addresses beyond the populated space will wrap on 4k boundaries. As an example, the addresses 0x1000 and 0x2000 both evaluate to address 0x0000 in on-chip XRAM space.

- 8-bit MOVX operations use the contents of EMI0CN to determine the high-byte of the effective address and R0 or R1 to determine the low-byte of the effective address.
- 16-bit MOVX operations use the contents of the 16-bit DPTR to determine the effective address.

16.5.2. Split Mode without Bank Select

When EMI0CF.[3:2] are set to '01', the XRAM memory map is split into two areas, on-chip space and off-chip space.

- Effective addresses below the 4k boundary will access on-chip XRAM space.
- Effective addresses above the 4k boundary will access off-chip space.
- 8-bit MOVX operations use the contents of EMI0CN to determine whether the memory access is on-chip or off-chip. However, in the "No Bank Select" mode, an 8-bit MOVX operation will not drive the upper 8-bits A[15:8] of the Address Bus during an off-chip access. This allows the user to manipulate the upper address bits at will by setting the Port state directly via the port latches. This behavior is in contrast with "Split Mode with Bank Select" described below. The lower 8-bits of the Address Bus A[7:0] are driven, determined by R0 or R1.
- 16-bit MOVX operations use the contents of DPTR to determine whether the memory access is on-chip or off-chip, and unlike 8-bit MOVX operations, the full 16-bits of the Address Bus A[15:0] are driven during the off-chip transaction.

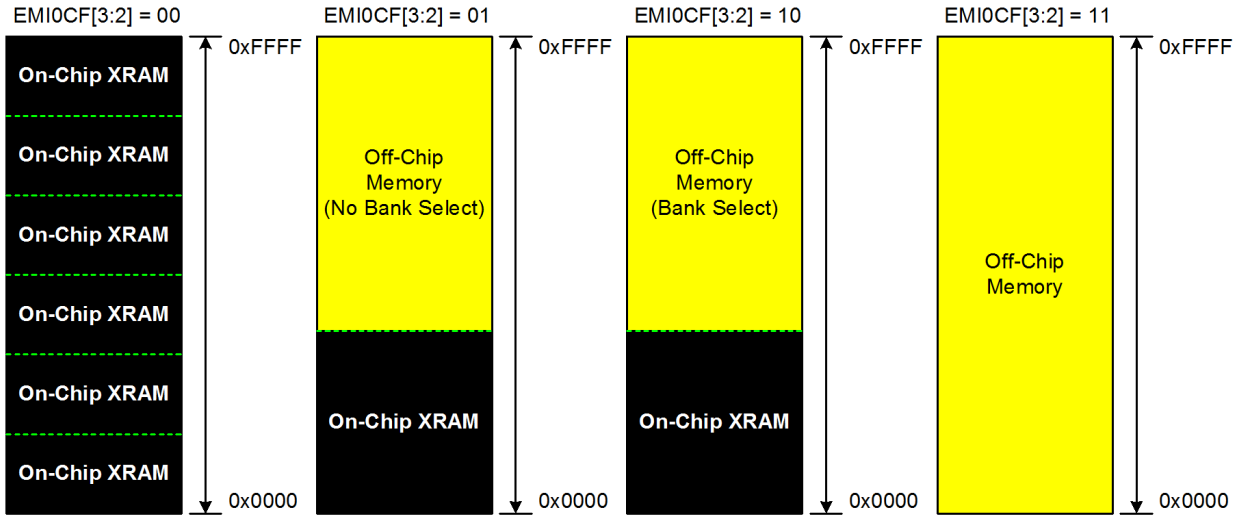


Figure 16.3. EMIF Operating Modes

a digital input by setting P3MDOUT.7 to a logic 0, which selects open-drain output mode, and P3.7 to a logic 1, which disables the low-side output driver.

If the Port pin has been assigned to a digital peripheral by the Crossbar and that pin functions as an input (for example RX0, the UART0 receive pin), then the output drivers on that pin are automatically disabled.

17.1.4. Weak Pullups

By default, each Port pin has an internal weak pullup device enabled which provides a resistive connection (about 100 kΩ) between the pin and V_{DD}. The weak pullup devices can be globally disabled by writing a logic 1 to the Weak Pullup Disable bit, (WEAKPUD, XBR2.7). The weak pullup is automatically deactivated on any pin that is driving a logic 0; that is, an output pin will not contend with its own pullup device. The weak pullup device can also be explicitly disabled on Ports 1, 2, and 3 pin by configuring the pin as an Analog Input, as described below.

17.1.5. Configuring Port 1, 2, and 3 Pins as Analog Inputs

The pins on Port 1 can serve as analog inputs to the ADC2 analog MUX (C8051F040/1/2/3 only), the pins on Port 2 can serve as analog inputs to the Comparators, and the pins on Port 3 can serve as inputs to ADC0. A Port pin is configured as an Analog Input by writing a logic 0 to the associated bit in the PnMDIN registers. All Port pins default to a Digital Input mode. Configuring a Port pin as an analog input:

1. Disables the digital input path from the pin. This prevents additional power supply current from being drawn when the voltage at the pin is near V_{DD} / 2. A read of the Port Data bit will return a logic 0 regardless of the voltage at the Port pin.
2. Disables the weak pullup device on the pin.
3. Causes the Crossbar to “skip over” the pin when allocating Port pins for digital peripherals, except for P2.0-P2.1.

Note that the output drivers on a pin configured as an Analog Input are not explicitly disabled. Therefore, the associated PnMDOUT bits of pins configured as Analog Inputs should explicitly be set to logic 0 (Open-Drain output mode), and the associated Port Data bits should be set to logic 1 (high-impedance). Also note that it is not required to configure a Port pin as an Analog Input in order to use it as an input to the ADC’s or Comparators; however, it is strongly recommended. See the analog peripheral’s corresponding section in this datasheet for further information.

17.1.6. External Memory Interface Pin Assignments

If the External Memory Interface (EMIF) is enabled on the Low ports (Ports 0 through 3), EMIFLE (XBR2.5) should be set to a logic 1 so that the Crossbar will not assign peripherals to P0.7 (/WR), P0.6 (/RD), and, if the External Memory Interface is in Multiplexed mode, P0.5 (ALE). Figure 17.4 shows an example Crossbar Decode Table with EMIFLE=1 and the EMIF in Multiplexed mode. Figure 17.5 shows an example Crossbar Decode Table with EMIFLE=1 and the EMIF in Non-multiplexed mode.

If the External Memory Interface is enabled on the Low ports and an off-chip MOVX operation occurs, the External Memory Interface will control the output states (logic 1 or logic 0) of the affected Port pins during the execution phase of the MOVX instruction, regardless of the settings of the Crossbar registers or the Port Data registers. The output configuration (push-pull or open-drain) of the Port pins is not affected by the EMIF operation, except that Read operations will explicitly disable the output drivers on the Data Bus. **In most cases, GPIO pins used in EMIF operations (especially the /WR and /RD lines) should be configured as push-pull and 'parked' at a logic 1 state.** See **Section "16. External Data Memory Interface and On-Chip XRAM" on page 187** for more information about the External Memory Interface.

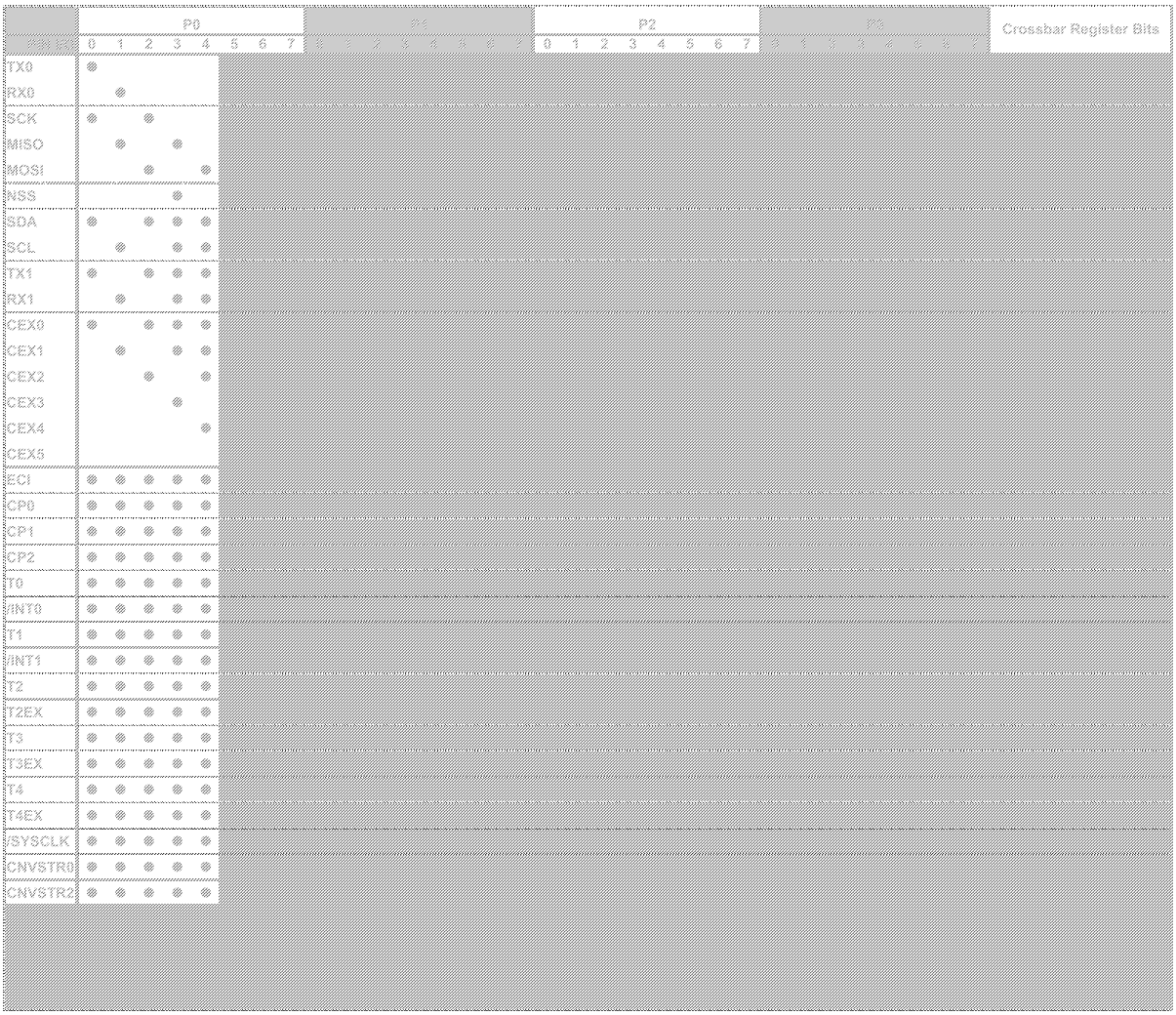


Figure 17.4. Priority Crossbar Decode Table
(EMIFLE = 1; EMIF in Multiplexed Mode; P1MDIN = 0xFF)

19.4. SMBus Special Function Registers

The SMBus0 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 five special function registers related to the operation of the SMBus0 interface are described in the following sections.

19.4.1. Control Register

The SMBus0 Control register SMB0CN is used to configure and control the SMBus0 interface. All of the bits in the register can be read or written by software. Two of the control bits are also affected by the SMBus0 hardware. The Serial Interrupt flag (SI, SMB0CN.3) is set to logic 1 by the hardware when a valid serial interrupt condition occurs. It can only be cleared by software. The Stop flag (STO, SMB0CN.4) is set to logic 1 by software. It is cleared to logic 0 by hardware when a STOP condition is detected on the bus.

Setting the ENSMB flag to logic 1 enables the SMBus0 interface. Clearing the ENSMB flag to logic 0 disables the SMBus0 interface and removes it from the bus. Momentarily clearing the ENSMB flag and then resetting it to logic 1 will reset SMBus0 communication. However, ENSMB should not be used to temporarily remove a device from the bus since the bus state information will be lost. Instead, the Assert Acknowledge (AA) flag should be used to temporarily remove the device from the bus (see description of AA flag below).

Setting the Start flag (STA, SMB0CN.5) to logic 1 will put SMBus0 in a master mode. If the bus is free, SMBus0 will generate a START condition. If the bus is not free, SMBus0 waits for a STOP condition to free the bus and then generates a START condition after a 5 µs delay per the SMB0CR value (In accordance with the SMBus protocol, the SMBus0 interface also considers the bus free if the bus is idle for 50 µs and no STOP condition was recognized). If STA is set to logic 1 while SMBus0 is in master mode and one or more bytes have been transferred, a repeated START condition will be generated.

When the Stop flag (STO, SMB0CN.4) is set to logic 1 while the SMBus0 interface is in master mode, the interface generates a STOP condition. In a slave mode, the STO flag may be used to recover from an error condition. In this case, a STOP condition is not generated on the bus, but the SMBus hardware behaves as if a STOP condition has been received and enters the "not addressed" slave receiver mode. Note that this simulated STOP will not cause the bus to appear free to SMBus0. The bus will remain occupied until a STOP appears on the bus or a Bus Free Timeout occurs. Hardware automatically clears the STO flag to logic 0 when a STOP condition is detected on the bus.

The Serial Interrupt flag (SI, SMB0CN.3) is set to logic 1 by hardware when the SMBus0 interface enters any one of the 28 possible states except the Idle state. If interrupts are enabled for the SMBus0 interface, an interrupt request is generated when the SI flag is set. The SI flag must be cleared by software.

Important Note: If SI is set to logic 1 while the SCL line is low, the clock-low period of the serial clock will be stretched and the serial transfer is suspended until SI is cleared to logic 0. A high level on SCL is not affected by the setting of the SI flag.

The Assert Acknowledge flag (AA, SMB0CN.2) is used to set the level of the SDA line during the acknowledge clock cycle on the SCL line. Setting the AA flag to logic 1 will cause an ACK (low level on SDA) to be sent during the acknowledge cycle if the device has been addressed. Setting the AA flag to logic 0 will cause a NACK (high level on SDA) to be sent during acknowledge cycle. After the transmission of a byte in slave mode, the slave can be temporarily removed from the bus by clearing the AA flag. The slave's own address and general call address will be ignored. To resume operation on the bus, the AA flag must be reset to logic 1 to allow the slave's address to be recognized.

SFR Definition 19.4. SMB0ADR: SMBus0 Address

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SLV6	SLV5	SLV4	SLV3	SLV2	SLV1	SLV0	GC	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: 0xC3
SFR Page: 0

Bits7-1: SLV6-SLV0: SMBus0 Slave Address.
These bits are loaded with the 7-bit slave address to which SMBus0 will respond when operating as a slave transmitter or slave receiver. SLV6 is the most significant bit of the address and corresponds to the first bit of the address byte received.

Bit0: GC: General Call Address Enable.
This bit is used to enable general call address (0x00) recognition.
0: General call address is ignored.
1: General call address is recognized.

19.4.5. 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 19.1.

21.3. Configuration of a Masked Address

The UART0 address is configured via two SFRs: SADDR0 (Serial Address) and SADEN0 (Serial Address Enable). SADEN0 sets the bit mask for the address held in SADDR0: bits set to logic 1 in SADEN0 correspond to bits in SADDR0 that are checked against the received address byte; bits set to logic 0 in SADEN0 correspond to “don’t care” bits in SADDR0.

Example 1, SLAVE #1	
SADDR0	= 00110101
SADEN0	= 00001111
UART0 Address	= xxxx0101

Example 2, SLAVE #2	
SADDR0	= 00110101
SADEN0	= 11110011
UART0 Address	= 0011xx01

Example 3, SLAVE #3	
SADDR0	= 00110101
SADEN0	= 11000000
UART0 Address	= 00xxxxxx

Setting the SM20 bit (SCON0.5) configures UART0 such that when a stop bit is received, UART0 will generate an interrupt only if the ninth bit is logic 1 (RB80 = ‘1’) and the received data byte matches the UART0 slave address. Following the received address interrupt, the slave will clear its SM20 bit to enable interrupts on the reception of the following data byte(s). Once the entire message is received, the addressed slave resets its SM20 bit to ignore all transmissions until it receives the next address byte. While SM20 is logic 1, UART0 ignores all bytes that do not match the UART0 address and include a ninth bit that is logic 1.

21.4. Broadcast Addressing

Multiple addresses can be assigned to a single slave and/or a single address can be assigned to multiple slaves, thereby enabling "broadcast" transmissions to more than one slave simultaneously. The broadcast address is the logical OR of registers SADDR0 and SADEN0, and ‘0’s of the result are treated as “don’t cares”. Typically a broadcast address of 0xFF (hexadecimal) is acknowledged by all slaves, assuming “don’t care” bits as ‘1’s. The master processor can be configured to receive all transmissions or a protocol can be implemented such that the master/slave role is temporarily reversed to enable half-duplex transmission between the original master and slave(s).

Example 4, SLAVE #1	
SADDR0	= 00110101
SADEN0	= 00001111
Broadcast Address	= 00111111

Example 5, SLAVE #2	
SADDR0	= 00110101
SADEN0	= 11110011
Broadcast Address	= 11110111

Example 6, SLAVE #3	
SADDR0	= 00110101
SADEN0	= 11000000
Broadcast Address	= 11110101

Where all ZEROES in the Broadcast address are don’t cares.

Note in the above examples 4, 5, and 6, each slave would recognize as “valid” an address of 0xFF as a broadcast address. Also note that examples 4, 5, and 6 uses the same SADDR0 and SADEN0 register values as shown in the examples 1, 2, and 3 respectively (slaves #1, 2, and 3). Thus, a master could address each slave device individually using a masked address, and also broadcast to all three slave devices. For example, if a Master were to send an address “11110101”, only slave #1 would recognize the address as valid. If a master were to then send an address of “11111111”, all three slave devices would recognize the address as a valid broadcast address.

22.2. Operational Modes

UART1 provides standard asynchronous, full duplex communication. The UART mode (8-bit or 9-bit) is selected by the S1MODE bit (SCON1.7). Typical UART connection options are shown below.

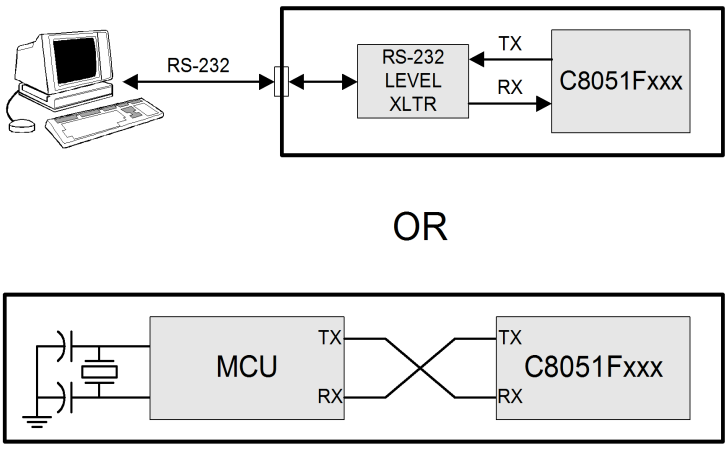


Figure 22.3. UART Interconnect Diagram

22.2.1. 8-Bit UART

8-Bit UART mode uses a total of 10 bits per data byte: one start bit, eight data bits (LSB first), and one stop bit. Data are transmitted LSB first 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 software 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 MCE1 is logic 1, the stop bit must be logic 1. In the event of a receive data overrun, the first received 8 bits are latched into the SBUF1 receive register and the following overrun data bits are lost.

If these conditions are met, the eight bits of data is 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.

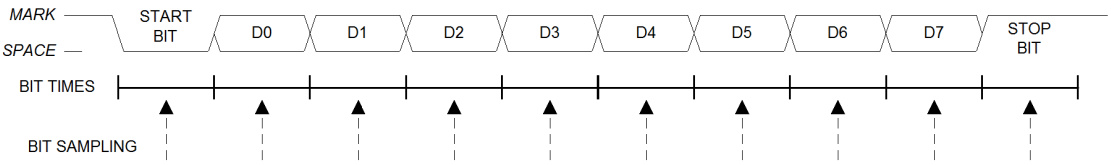


Figure 22.4. 8-Bit UART Timing Diagram

22.3. Multiprocessor Communications

9-Bit UART mode supports multiprocessor communication between a master processor and one or more slave processors by special use of the ninth data bit. When a master processor wants to transmit to one or more slaves, it first sends an address byte to select the target(s). An address byte differs from a data byte in that its ninth bit is logic 1; in a data byte, the ninth bit is always set to logic 0.

Setting the MCE1 bit (SCON1.5) of a slave processor configures its UART such that when a stop bit is received, the UART will generate an interrupt only if the ninth bit is logic one (RB81 = 1) signifying an address byte has been received. In the UART interrupt handler, software should compare the received address with the slave's own assigned 8-bit address. If the addresses match, the slave should clear its MCE1 bit to enable interrupts on the reception of the following data byte(s). Slaves that weren't addressed leave their MCE1 bits set and do not generate interrupts on the reception of the following data bytes, thereby ignoring the data. Once the entire message is received, the addressed slave should reset its MCE1 bit to ignore all transmissions until it receives the next address byte.

Multiple addresses can be assigned to a single slave and/or a single address can be assigned to multiple slaves, thereby enabling "broadcast" transmissions to more than one slave simultaneously. The master processor can be configured to receive all transmissions or a protocol can be implemented such that the master/slave role is temporarily reversed to enable half-duplex transmission between the original master and slave(s).

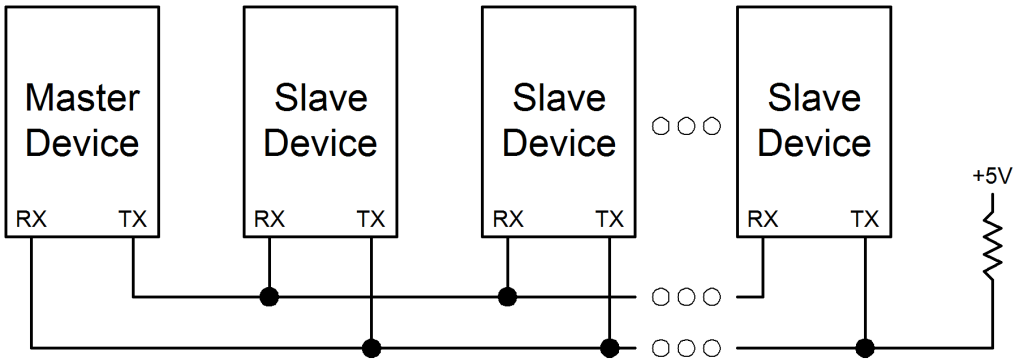


Figure 22.6. UART Multi-Processor Mode Interconnect Diagram

24.2.2. Software Timer (Compare) Mode

In Software Timer mode, the PCA0 counter/timer is compared to the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match 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. Setting the ECOMn and MATn bits in the PCA0CPMn register enables Software Timer mode.

Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to '0'; writing to PCA0CPHn sets ECOMn to '1'.

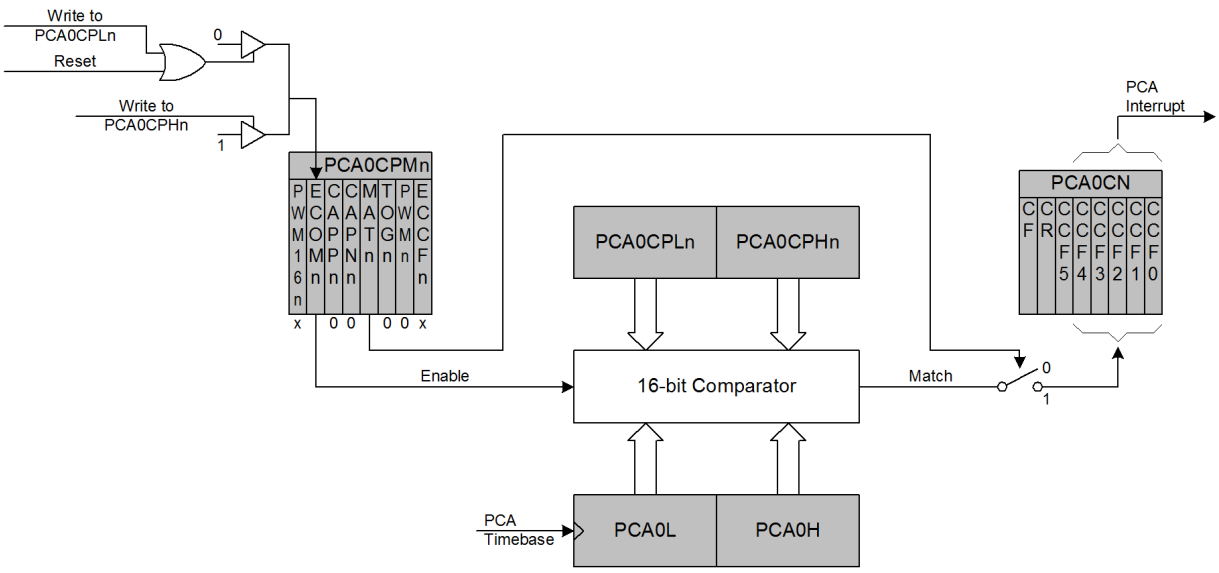


Figure 24.5. PCA Software Timer Mode Diagram

25.1. Boundary Scan

The DR in the Boundary Scan path is an 134-bit shift register. The Boundary DR provides control and observability of all the device pins as well as the SFR bus and Weak Pullup feature via the EXTEST and SAMPLE commands.

Table 25.1. Boundary Data Register Bit Definitions

EXTEST provides access to both capture and update actions, while Sample only performs a capture.

Bit	Action	Target
0	Capture	Reset Enable from MCU
	Update	Reset Enable to /RST pin
1	Capture	Reset input from /RST pin
	Update	Reset output to /RST pin
2	Capture	Reset Enable from MCU
	Update	Reset Enable to /RST pin
3	Capture	Reset input from /RST pin
	Update	Reset output to /RST pin
4	Capture	CANRX output enable to pin
	Update	CANRX output enable to pin
5	Capture	CANRX input from pin
	Update	CANRX output to pin
6	Capture	CANTX output enable to pin
	Update	CANTX output enable to pin
7	Capture	CANTX input from pin
	Update	CANTX output to pin
8	Capture	External Clock from XTAL1 pin
	Update	Not used
9	Capture	Weak pullup enable from MCU
	Update	Weak pullup enable to Port Pins
10, 12, 14, 16, 18, 20, 22, 24	Capture	P0.n output enable from MCU (e.g. Bit6=P0.0, Bit8=P0.1, etc.)
	Update	P0.n output enable to pin (e.g. Bit6=P0.0oe, Bit8=P0.1oe, etc.)
11, 13, 15, 17, 19, 21, 23, 25	Capture	P0.n input from pin (e.g. Bit7=P0.0, Bit9=P0.1, etc.)
	Update	P0.n output to pin (e.g. Bit7=P0.0, Bit9=P0.1, etc.)
26, 28, 30, 32, 34, 36, 38, 40	Capture	P1.n output enable from MCU
	Update	P1.n output enable to pin
27, 29, 31, 33, 35, 37, 39, 41	Capture	P1.n input from pin
	Update	P1.n output to pin
42, 44, 46, 48, 50, 52, 54, 56	Capture	P2.n output enable from MCU
	Update	P2.n output enable to pin
43, 45, 47, 49, 51, 53, 55, 57	Capture	P2.n input from pin
	Update	P2.n output to pin
58, 60, 62, 64, 66, 68, 70, 72	Capture	P3.n output enable from MCU
	Update	P3.n output enable to pin
59, 61, 63, 65, 67, 69, 71, 73	Capture	P3.n input from pin
	Update	P3.n output to pin
74, 76, 78, 80, 82, 84, 86, 88	Capture	P4.n output enable from MCU
	Update	P4.n output enable to pin