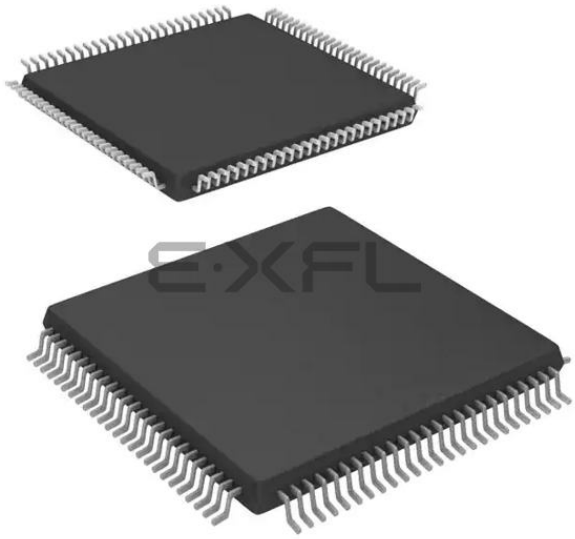


Welcome to E-XFL.COM

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	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 8x8b, 8x12b; D/A 2x12b
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/c8051f020-gqr

TABLE OF CONTENTS

1.3. JTAG Debug and Boundary Scan

The C8051F020 family has on-chip JTAG boundary scan and debug circuitry that provides *non-intrusive, full speed, in-circuit debugging using the production part installed in the end application*, via the four-pin JTAG interface. The JTAG port is fully compliant to IEEE 1149.1, providing full boundary scan for test and manufacturing purposes.

Silicon Labs' debugging system supports inspection and modification of memory and registers, breakpoints, watch-points, 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 and SMBus) are stalled when the MCU is halted, during single stepping, or at a breakpoint in order to keep them synchronized.

The C8051F020DK development kit provides all the hardware and software necessary to develop application code and perform in-circuit debugging with the C8051F020/1/2/3 MCUs. The kit includes software with a developer's studio and debugger, an integrated 8051 assembler, and an RS-232 to JTAG serial adapter. It also has a target application board with the associated MCU installed, plus the RS-232 and JTAG cables, and wall-mount power supply. The Development Kit requires a Windows 95/98/NT/ME/2000 computer with one available RS-232 serial port. As shown in Figure 1.8, the PC is connected via RS-232 to the Serial Adapter. A six-inch ribbon cable connects the Serial Adapter to the user's application board, picking up the four JTAG pins and VDD and GND. The Serial Adapter takes its power from the application board; it requires roughly 20 mA at 2.7-3.6 V. For applications where there is not sufficient power available from the target system, the provided power supply can be connected directly to the Serial Adapter.

Silicon Labs' debug environment 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 1.8. Development/In-System Debug Diagram

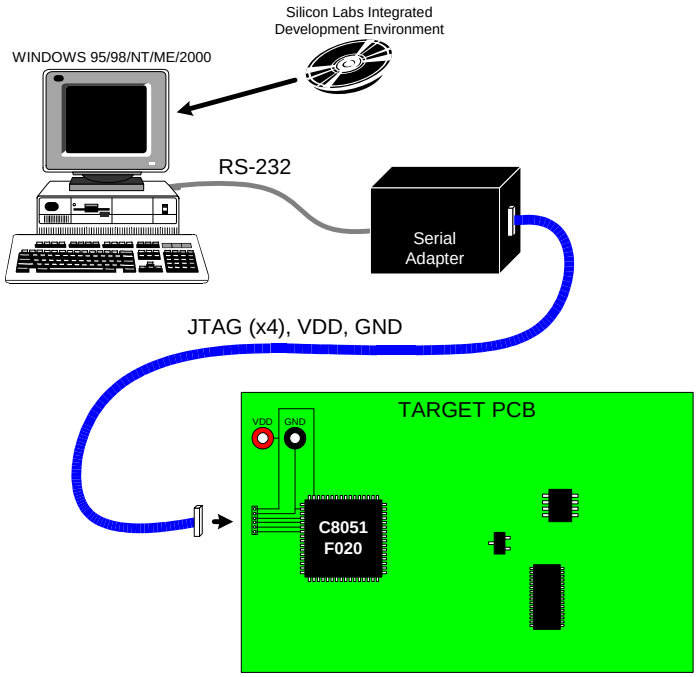


Figure 4.3. TQFP-64 Pinout Diagram

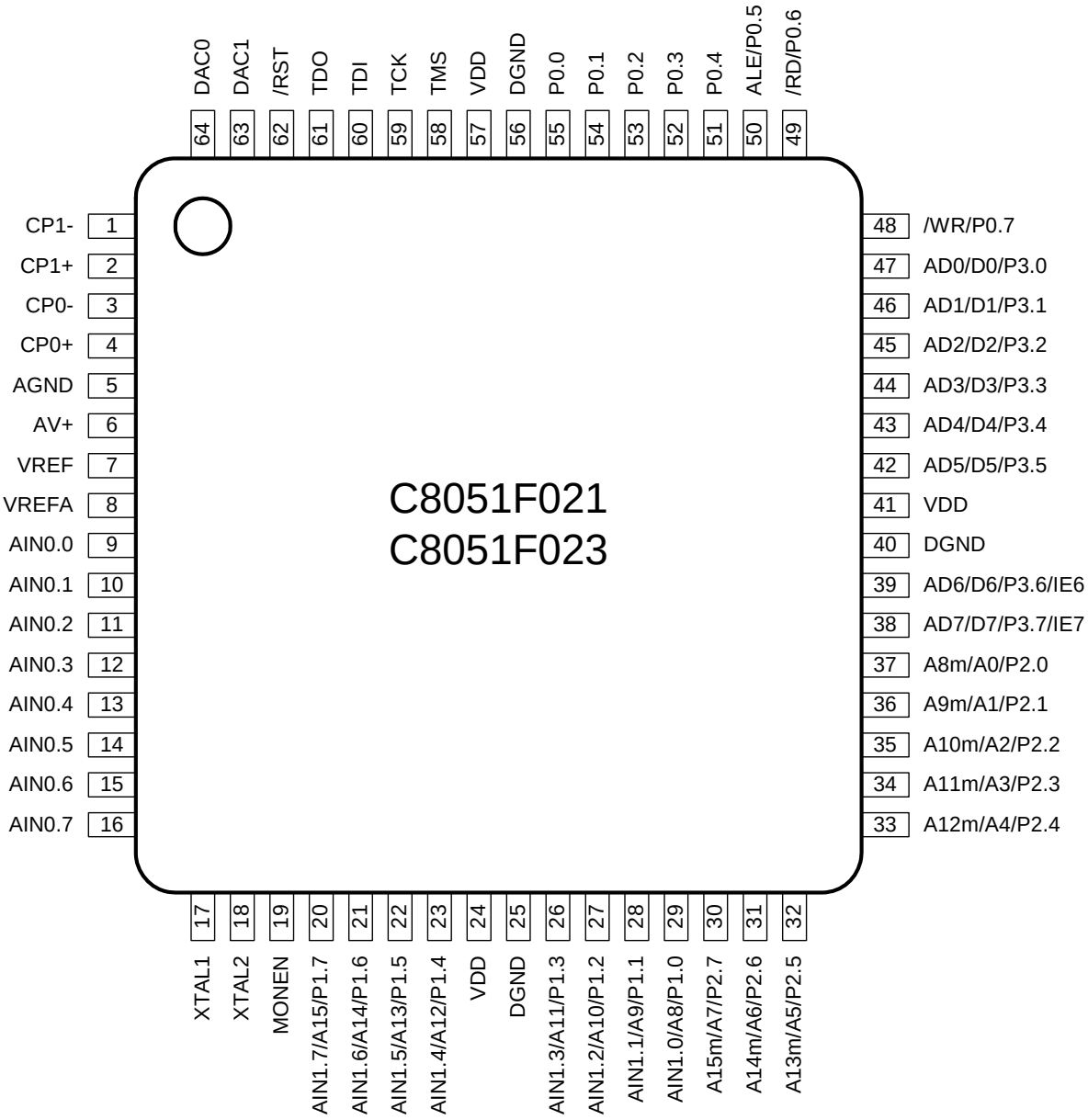


Figure 5.9. ADC0H: ADC0 Data Word MSB Register (C8051F020/1)

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

Bits7-0: ADC0 Data Word High-Order Bits.
For AD0LJST = 0: Bits 7-4 are the sign extension of Bit3. Bits 3-0 are the upper 4 bits of the 12-bit ADC0 Data Word.
For AD0LJST = 1: Bits 7-0 are the most-significant bits of the 12-bit ADC0 Data Word.

Figure 5.10. ADC0L: ADC0 Data Word LSB Register (C8051F020/1)

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

Bits7-0: ADC0 Data Word Low-Order Bits.
For AD0LJST = 0: Bits 7-0 are the lower 8 bits of the 12-bit ADC0 Data Word.
For AD0LJST = 1: Bits 7-4 are the lower 4 bits of the 12-bit ADC0 Data Word. Bits3-0 will always read ‘0’.

6.3. ADC0 Programmable Window Detector

The ADC0 Programmable Window Detector continuously compares the ADC0 output to user-programmed limits, and notifies the system when an out-of-bound condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (AD0WINT in ADC0CN) can also be used in polled mode. The high and low bytes of the reference words are loaded into the ADC0 Greater-Than and ADC0 Less-Than registers (ADC0GTH, ADC0GTL, ADC0LTH, and ADC0LTL). Reference comparisons are shown starting on page 70. Notice that the window detector flag can be asserted when the measured data is inside or outside the user-programmed limits, depending on the programming of the ADC0GTx and ADC0LTx registers.

Figure 6.12. ADC0GTH: ADC0 Greater-Than Data High Byte Register (C8051F022/3)

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

Bits7-0: High byte of ADC0 Greater-Than Data Word.

Figure 6.13. ADC0GTL: ADC0 Greater-Than Data Low Byte Register (C8051F022/3)

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

Bits7-0: Low byte of ADC0 Greater-Than Data Word.

Figure 6.14. ADC0LTH: ADC0 Less-Than Data High Byte Register (C8051F022/3)

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

Bits7-0: High byte of ADC0 Less-Than Data Word.

Figure 6.15. ADC0LTL: ADC0 Less-Than Data Low Byte Register (C8051F022/3)

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

Bits7-0: Low byte of ADC0 Less-Than Data Word.

Figure 11.4. CPT1CN: Comparator1 Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
CP1EN	CP1OUT	CP1RIF	CP1FIF	CP1HYP1	CP1HYP0	CP1HYN1	CP1HYN0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9F

Bit7: CP1EN: Comparator1 Enable Bit.
0: Comparator1 Disabled.
1: Comparator1 Enabled.

Bit6: CP1OUT: Comparator1 Output State Flag.
0: Voltage on CP1+ < CP1-.
1: Voltage on CP1+ > CP1-.

Bit5: CP1RIF: Comparator1 Rising-Edge Interrupt Flag.
0: No Comparator1 Rising Edge Interrupt has occurred since this flag was last cleared.
1: Comparator1 Rising Edge Interrupt has occurred.

Bit4: CP1FIF: Comparator1 Falling-Edge Interrupt Flag.
0: No Comparator1 Falling-Edge Interrupt has occurred since this flag was last cleared.
1: Comparator1 Falling-Edge Interrupt has occurred.

Bits3-2: CP1HYP1-0: Comparator1 Positive Hysteresis Control Bits.
00: Positive Hysteresis Disabled.
01: Positive Hysteresis = 2 mV.
10: Positive Hysteresis = 4 mV.
11: Positive Hysteresis = 10 mV.

Bits1-0: CP1HYN1-0: Comparator1 Negative Hysteresis Control Bits.
00: Negative Hysteresis Disabled.
01: Negative Hysteresis = 2 mV.
10: Negative Hysteresis = 4 mV.
11: Negative Hysteresis = 10 mV.

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.

16.5. Memory Mode Selection

The external data memory space can be configured in one of four modes, shown in Figure 16.5, based on the EMIF Mode bits in the EMI0CF register (Figure 16.2). These modes are summarized below. More information about the different modes can be found in **Section " ." on page 152**.

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.

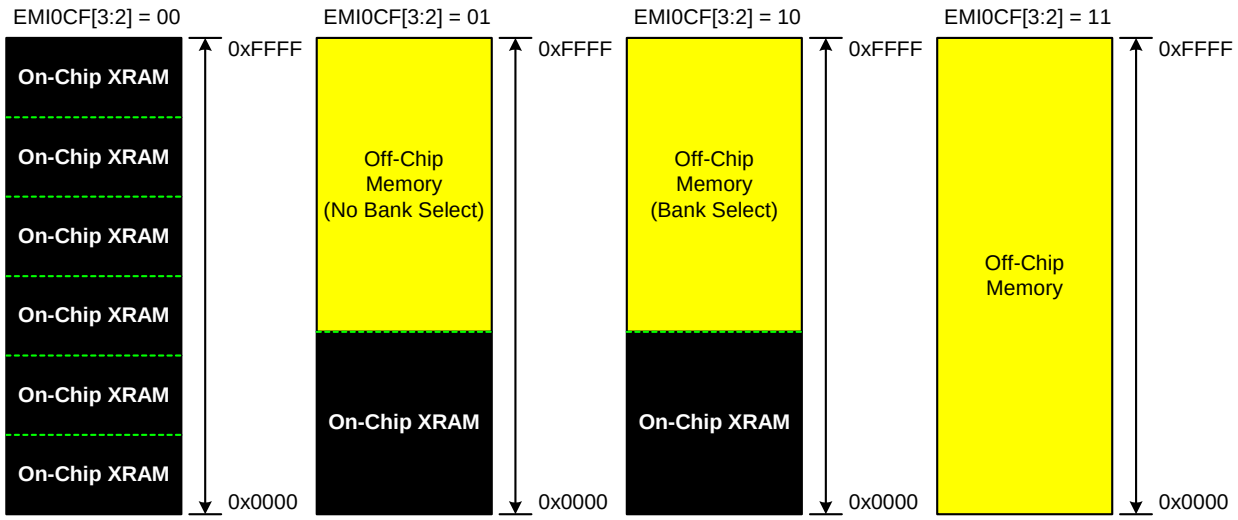
- 8-bit MOVX operations use the contents of EMI0CN to determine the high-byte of the effective address and R0 or R1 to determine the low-byte of the effective address.
- 16-bit MOVX operations use the contents of the 16-bit DPTR to determine the effective address.

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.

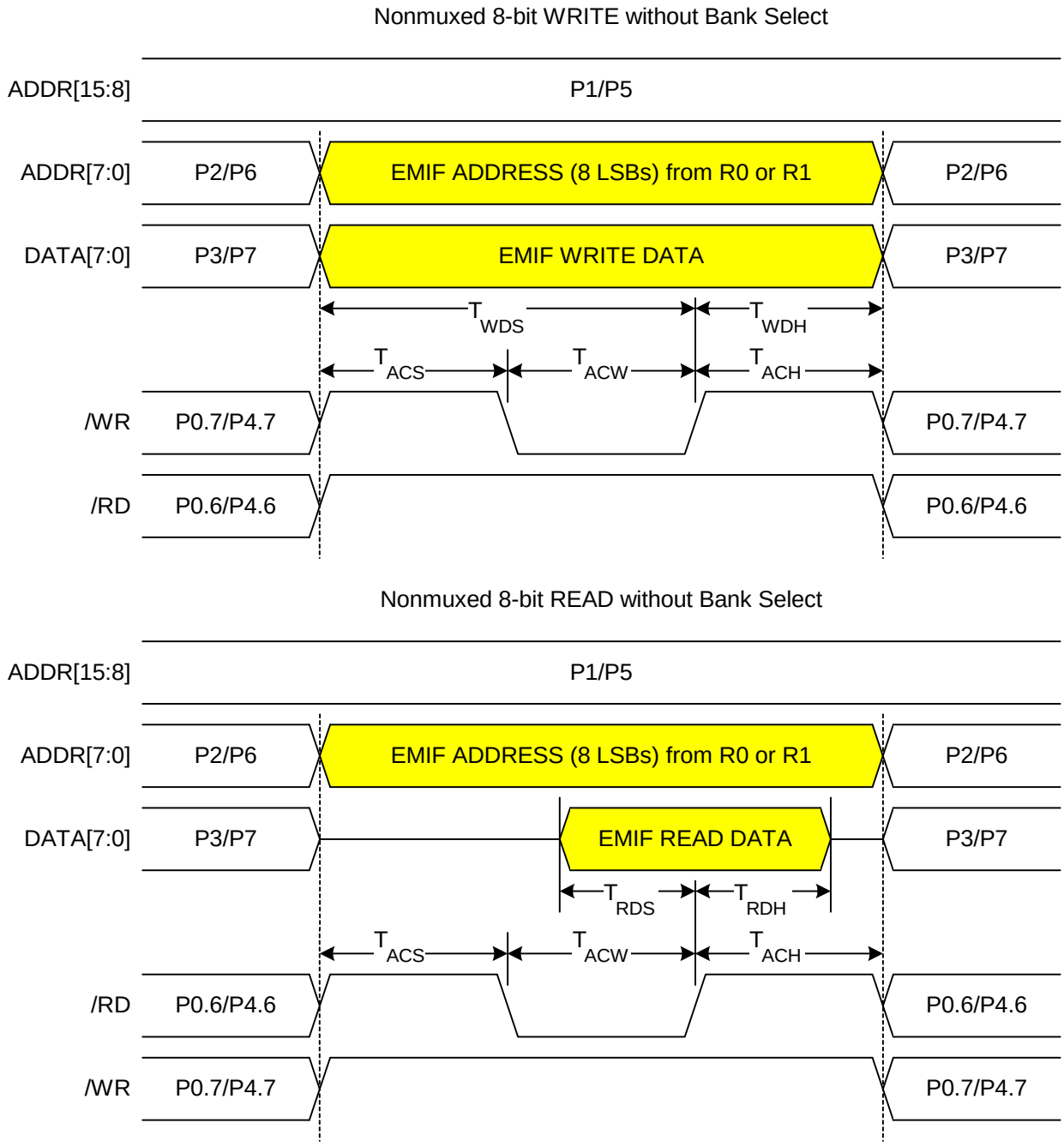
- Effective addresses below the 4k boundary will access on-chip XRAM space.
- Effective addresses beyond the 4k boundary will access off-chip space.
- 8-bit MOVX operations use the contents of EMI0CN to determine whether the memory access is on-chip or off-chip. The lower 8-bits of the Address Bus A[7:0] are driven as defined by R0 or R1. 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. This behavior is in contrast with "Split Mode with Bank Select" described below.
- 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 16.5. EMIF Operating Modes



16.6.1.2. 8-bit MOVX without Bank Select: EMI0CF[4:2] = ‘101’ or ‘111’.

Figure 16.8. Non-multiplexed 8-bit MOVX without Bank Select Timing



16.6.1.3. 8-bit MOVX with Bank Select: EMI0CF[4:2] = '110'.

Figure 16.9. Non-multiplexed 8-bit MOVX with Bank Select Timing

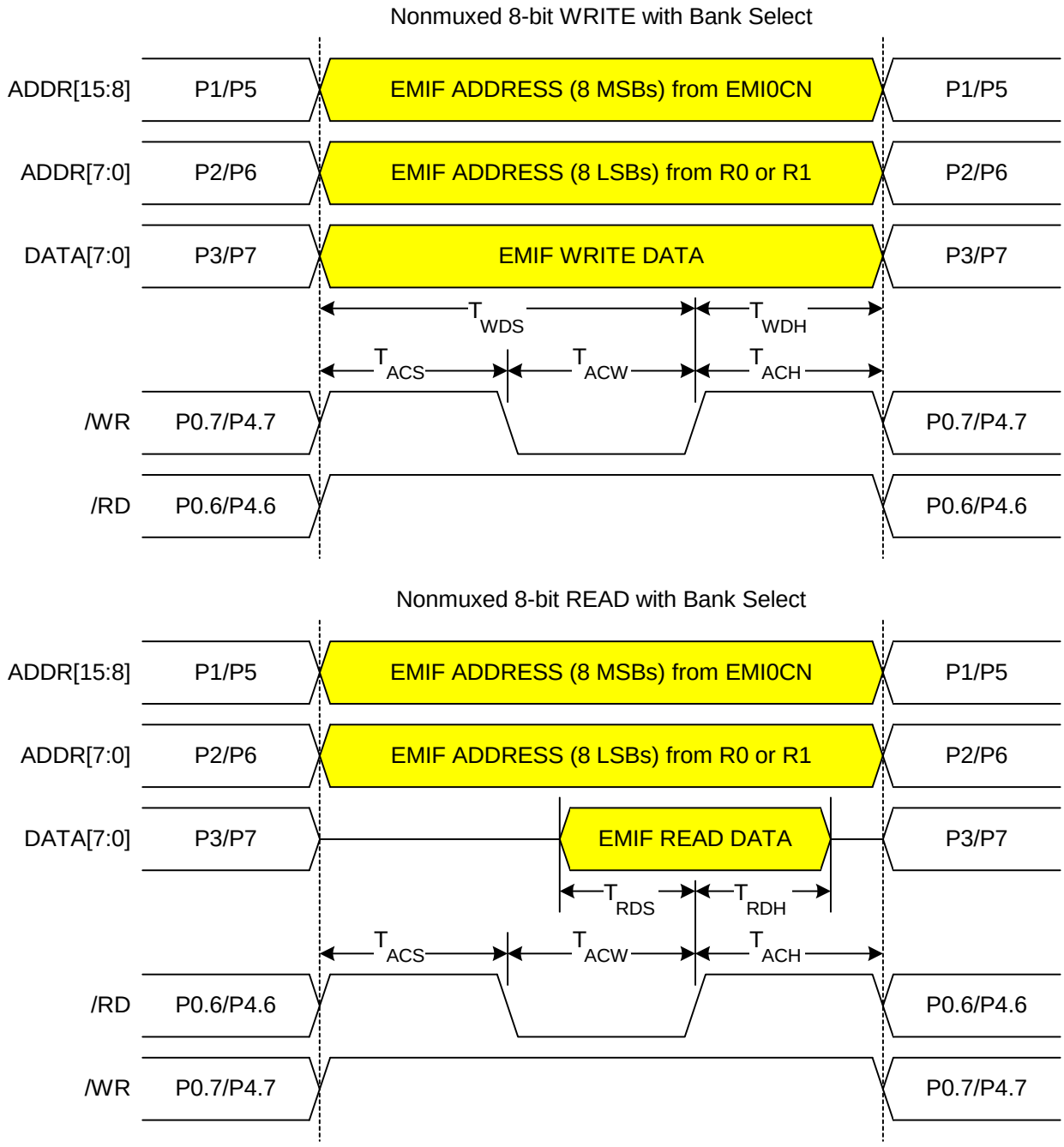


Figure 17.19. P3IF: Port3 Interrupt Flag Register

R/W	R/W	R	R	R/W	R/W	R/W	R/W	Reset Value
IE7	IE6	-	-	IE7CF	IE6CF	-	-	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xAD

Bit7: IE7: External Interrupt 7 Pending Flag
0: No falling edge has been detected on P3.7 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.7 is detected.
Bit6: IE6: External Interrupt 6 Pending Flag
0: No falling edge has been detected on P3.6 since this bit was last cleared.
1: This flag is set by hardware when a falling edge on P3.6 is detected.
Bits5-4: UNUSED. Read = 00b, Write = don't care.
Bit3: IE7CF: External Interrupt 7 Edge Configuration
0: External Interrupt 7 triggered by a falling edge on the IE7 input.
1: External Interrupt 7 triggered by a rising edge on the IE7 input.
Bit2: IE6CF: External Interrupt 6 Edge Configuration
0: External Interrupt 6 triggered by a falling edge on the IE6 input.
1: External Interrupt 6 triggered by a rising edge on the IE6 input.
Bits1-0: UNUSED. Read = 00b, Write = don't care.

17.2. Ports 4 through 7 (C8051F020/2 only)

All Port pins on Ports 4 through 7 can be accessed as General-Purpose I/O (GPIO) pins by reading and writing the associated Port Data registers (See Figure 17.21, Figure 17.22, Figure 17.23, and Figure 17.24), a set of SFRs which are byte-addressable.

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.

17.2.1. Configuring Ports which are not Pinned Out

Although P4, P5, P6, and P7 are not brought out to pins on the C8051F021/3 devices, the Port Data registers are still present and can be used by software. Because the digital input paths also remain active, it is recommended that these pins not be left in a 'floating' state in order to avoid unnecessary power dissipation arising from the inputs floating to non-valid logic levels. This condition can be prevented by any of the following:

1. Leave the weak pull-up devices enabled by setting WEAKPUD (XBR2.7) to a logic 0.
2. Configure the output modes of P4, P5, P6, and P7 to "Push-Pull" by writing P74OUT = 0xFF.
3. Force the output states of P4, P5, P6, and P7 to logic 0 by writing zeros to the Port Data registers: P4 = 0x00, P5 = 0x00, P6= 0x00, and P7 = 0x00.

17.2.2. Configuring the Output Modes of the Port Pins

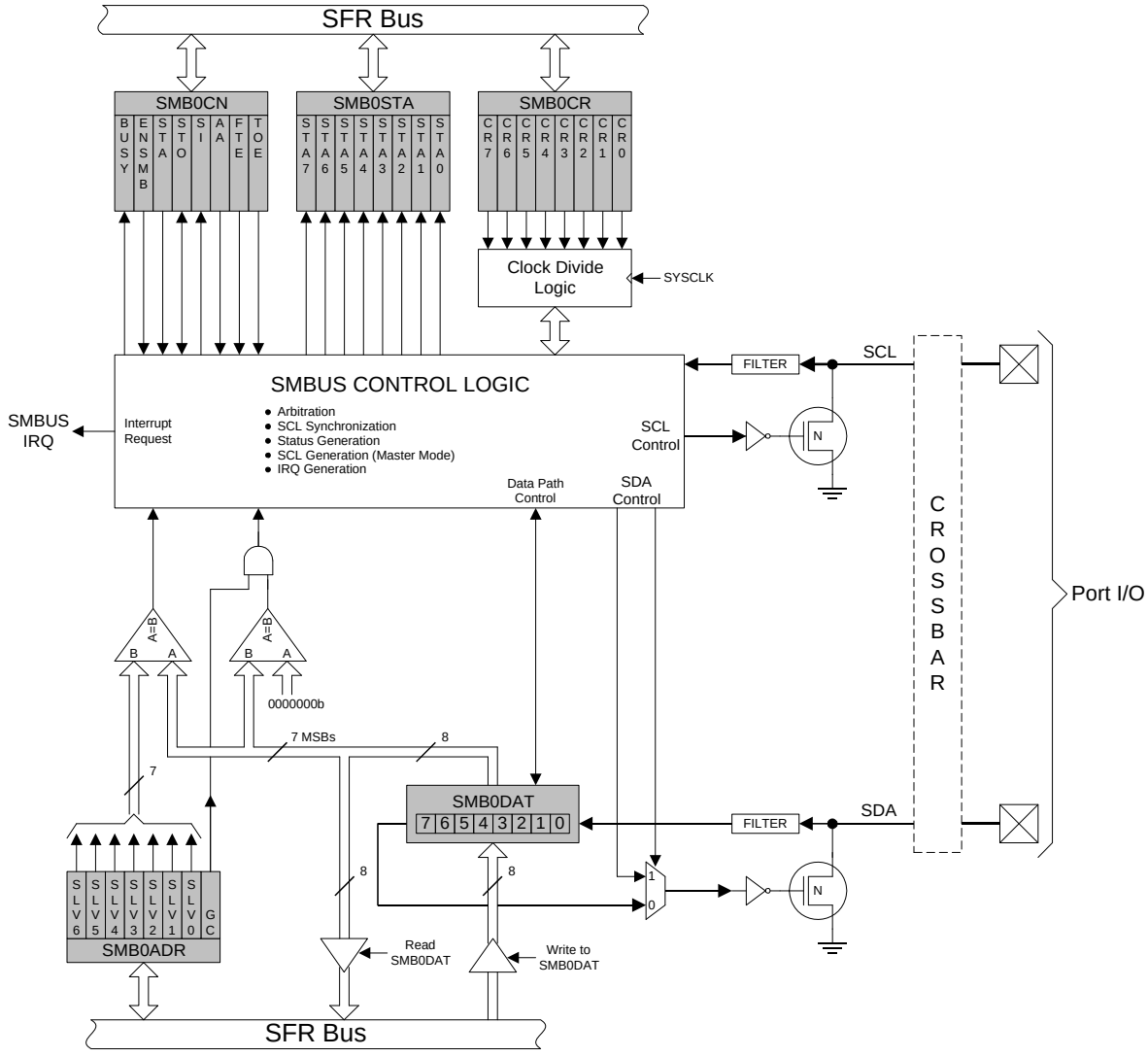
The output mode of each port pin can be configured to be either Open-Drain or Push-Pull. In the Push-Pull configuration, a logic 0 in the associated bit in the Port Data register will cause the Port pin to be driven to GND, and a logic 1 will cause the Port pin to be driven to VDD. In the Open-Drain configuration, a logic 0 in the associated bit in the

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



18.2.3. SCL Low Timeout

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

18.2.4. SCL High (SMBus Free) Timeout

The SMBus specification stipulates that if the SCL and SDA lines remain high for more that 50 µs, the bus is designated as free. If an SMBus device is waiting to generate a Master START, the START will be generated following a bus free timeout.

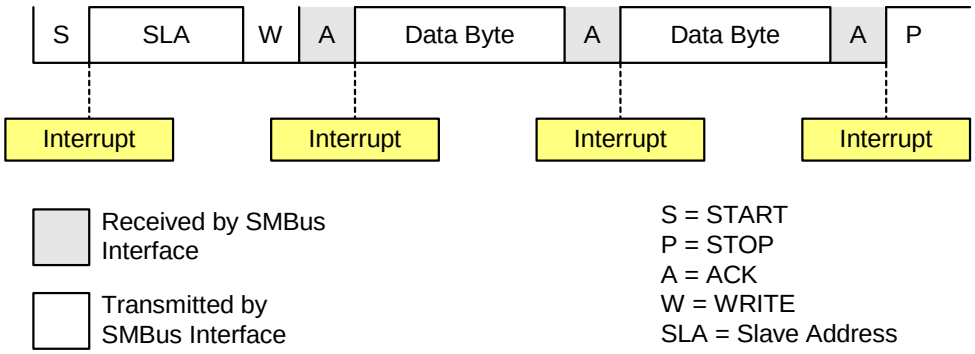
18.3. SMBus Transfer Modes

The SMBus0 interface may be configured to operate as a master and/or a slave. At any particular time, the interface will be operating in one of the following modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. See Table 18.1 for transfer mode status decoding using the SMB0STA status register. The following mode descriptions illustrate an interrupt-driven SMBus0 application; SMBus0 may alternatively be operated in polled mode.

18.3.1. Master Transmitter Mode

Serial data is transmitted on SDA while the serial clock is output on SCL. SMBus0 generates a START condition and then transmits the first byte containing the address of the target slave device and the data direction bit. In this case the data direction bit (R/W) will be logic 0 to indicate a "WRITE" operation. The SMBus0 interface transmits one or more bytes of serial data, waiting for an acknowledge (ACK) from the slave after each byte. To indicate the end of the serial transfer, SMBus0 generates a STOP condition.

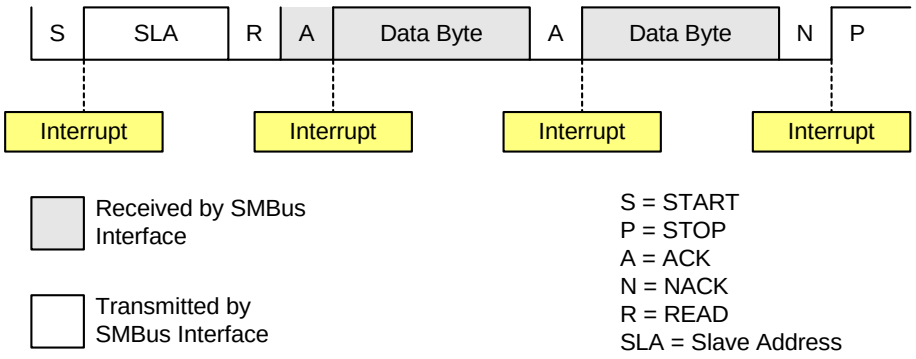
Figure 18.4. Typical Master Transmitter Sequence



18.3.2. Master Receiver Mode

Serial data is received on SDA while the serial clock is output on SCL. The SMBus0 interface generates a START followed by the first data byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 1 to indicate a "READ" operation. The SMBus0 interface receives serial data from the slave and generates the clock on SCL. After each byte is received, SMBus0 generates an ACK or NACK depending on the state of the AA bit in register SMB0CN. SMBus0 generates a STOP condition to indicate the end of the serial transfer.

Figure 18.5. Typical Master Receiver Sequence



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.

Table 18.1. SMB0STA Status Codes and States

Mode	Status Code	SMBus State	Typical Action
Slave Receiver	0x60	Own slave address + W received. ACK transmitted.	Wait for data.
	0x68	Arbitration lost in sending SLA + R/W as master. Own address + W received. ACK transmitted.	Save current data for retry when bus is free. Wait for data.
	0x70	General call address received. ACK transmitted.	Wait for data.
	0x78	Arbitration lost in sending SLA + R/W as master. General call address received. ACK transmitted.	Save current data for retry when bus is free.
	0x80	Data byte received. ACK transmitted.	Read SMB0DAT. Wait for next byte or STOP.
	0x88	Data byte received. NACK transmitted.	Set STO to reset SMBus.
	0x90	Data byte received after general call address. ACK transmitted.	Read SMB0DAT. Wait for next byte or STOP.
	0x98	Data byte received after general call address. NACK transmitted.	Set STO to reset SMBus.
	0xA0	STOP or repeated START received.	No action necessary.
Slave Transmitter	0xA8	Own address + R received. ACK transmitted.	Load SMB0DAT with data to transmit.
	0xB0	Arbitration lost in transmitting SLA + R/W as master. Own address + R received. ACK transmitted.	Save current data for retry when bus is free. Load SMB0DAT with data to transmit.
	0xB8	Data byte transmitted. ACK received.	Load SMB0DAT with data to transmit.
	0xC0	Data byte transmitted. NACK received.	Wait for STOP.
	0xC8	Last data byte transmitted (AA=0). ACK received.	Set STO to reset SMBus.
Slave	0xD0	SCL Clock High Timer per SMB0CR timed out	Set STO to reset SMBus.
All	0x00	Bus Error (illegal START or STOP)	Set STO to reset SMBus.
	0xF8	Idle	State does not set SI.

Figure 19.7. SPI0CKR: SPI0 Clock Rate Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9D

Bits7-0: SCR7-SCR0: SPI0 Clock Rate
These bits determine the frequency of the SCK output when the SPI0 module is configured for master mode operation. The SCK clock frequency is a divided down version of the system clock, and is given in the following equation, where *SYSCLK* is the system clock frequency and *SPI0CR* is the 8-bit value held in the SPI0CR register.

$$f_{SCK} = \frac{SYSCLK}{2 \times (SPI0CKR + 1)}$$

for 0 <= SPI0CKR <= 255

Example: If SYSCLK = 2 MHz and SPI0CKR = 0x04,

$$f_{SCK} = \frac{2000000}{2 \times (4 + 1)}$$

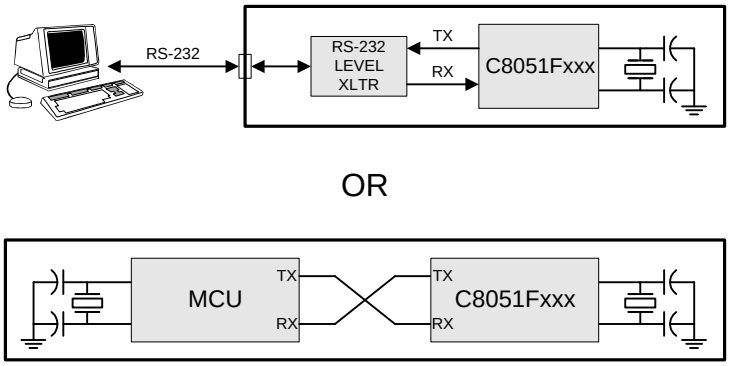
$$f_{SCK} = 200kHz$$

Figure 19.8. SPI0DAT: SPI0 Data Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0x9B

Bits7-0: SPI0DAT: SPI0 Transmit and Receive Data.
The SPI0DAT register is used to transmit and receive SPI0 data. Writing data to SPI0DAT places the data immediately into the shift register and initiates a transfer when in Master Mode. A read of SPI0-DAT returns the contents of the receive buffer.

Figure 20.6. UART Modes 1, 2, and 3 Interconnect Diagram



20.1.4. 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 2 overflows, as defined by Equation 20.1 and Equation 20.2. Multiprocessor communications and hardware address recognition are supported, as described in **Section 20.2**.

Important Note About the PCA0CN Register: If the main PCA counter (PCA0H : PCA0L) overflows during the execution phase of a read-modify-write instruction (bit-wise SETB or CLR, ANL, ORL, XRL) that targets the PCA0CN register, the CF (Counter Overflow) bit will not be set. The following steps should be taken when performing a bit-wise operation on the PCA0CN register:

Step 1. Disable global interrupts (EA = 0).
Step 2. Read PCA0L. This will latch the value of PCA0H.
Step 3. Read PCA0H, saving the value.
Step 4. Execute the bit-wise operation on CCFn (for example, CLR CCF0, or CCF0 = 0;).
Step 5. Read PCA0L.
Step 6. Read PCA0H, saving the value.
Step 7. If the value of PCA0H read in Step 3 is 0xFF and the value for PCA0H read in Step 6 is 0x00, then manually set the CF bit in software (for example, SETB CF, or CF = 1;).
Step 8. Re-enable interrupts (EA = 1).