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	Active
Core Processor	8051
Core Size	8-Bit
Speed	25MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.2V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	24-SSOP (0.154", 3.90mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f853-c-iu

Table 3.1. Pin Definitions for C8051F850/1/2/3/4/5-GU and C8051F850/1/2/3/4/5-IU

Pin Name	Type	Pin Numbers	Crossbar Capability	Additional Digital Functions	Analog Functions
P1.0	Standard I/O	18	Yes	P1MAT.0	ADC0.8 CP1P.0 CP1N.0
P1.1	Standard I/O	17	Yes	P1MAT.1	ADC0.9 CP1P.1 CP1N.1
P1.2	Standard I/O	16	Yes	P1MAT.2	ADC0.10 CP1P.2 CP1N.2
P1.3	Standard I/O	15	Yes	P1MAT.3	ADC0.11 CP1P.3 CP1N.3
P1.4	Standard I/O	14	Yes	P1MAT.4	ADC0.12 CP1P.4 CP1N.4
P1.5	Standard I/O	11	Yes	P1MAT.5	ADC0.13 CP1P.5 CP1N.5
P1.6	Standard I/O	10	Yes	P1MAT.6	ADC0.14 CP1P.6 CP1N.6
P1.7	Standard I/O	9	Yes	P1MAT.7	ADC0.15 CP1P.7 CP1N.7
P2.0 / C2D	Standard I/O / C2 Debug Data	8			
P2.1	Standard I/O	12			

The upper 128 bytes of data memory are accessible only by indirect addressing. This region occupies the same address space as the Special Function Registers (SFR) but is physically separate from the SFR space. The addressing mode used by an instruction when accessing locations above 0x7F determines whether the CPU accesses the upper 128 bytes of data memory space or the SFRs. Instructions that use direct addressing will access the SFR space. Instructions using indirect addressing above 0x7F access the upper 128 bytes of data memory. Figure 8.1 illustrates the data memory organization of the C8051F85x/86x.

Revision C C8051F852/5 and C8051F862/5 devices implement the upper four bytes of internal RAM as a 32-bit Unique Identifier. More information can be found in “Device Identification and Unique Identifier” on page 68.

8.2.1.1. General Purpose Registers

The lower 32 bytes of data memory, locations 0x00 through 0x1F, may be addressed as four banks of general-purpose registers. Each bank consists of eight byte-wide registers designated R0 through R7. Only one of these banks may be enabled at a time. Two bits in the program status word (PSW) register, RS0 and RS1, select the active register bank. This allows fast context switching when entering subroutines and interrupt service routines. Indirect addressing modes use registers R0 and R1 as index registers.

8.2.1.2. Bit Addressable Locations

In addition to direct access to data memory organized as bytes, the sixteen data memory locations at 0x20 through 0x2F are also accessible as 128 individually addressable bits. Each bit has a bit address from 0x00 to 0x7F. Bit 0 of the byte at 0x20 has bit address 0x00 while bit7 of the byte at 0x20 has bit address 0x07. Bit 7 of the byte at 0x2F has bit address 0x7F. A bit access is distinguished from a full byte access by the type of instruction used (bit source or destination operands as opposed to a byte source or destination).

The MCS-51™ assembly language allows an alternate notation for bit addressing of the form XX.B where XX is the byte address and B is the bit position within the byte. For example, the instruction:

```
MOV    C, 22.3h
```

moves the Boolean value at 0x13 (bit 3 of the byte at location 0x22) into the Carry flag.

8.2.1.3. Stack

A programmer's stack can be located anywhere in the 256-byte data memory. The stack area is designated using the Stack Pointer (SP) SFR. The SP will point to the last location used. The next value pushed on the stack is placed at SP+1 and then SP is incremented. A reset initializes the stack pointer to location 0x07. Therefore, the first value pushed on the stack is placed at location 0x08, which is also the first register (R0) of register bank 1. Thus, if more than one register bank is to be used, the SP should be initialized to a location in the data memory not being used for data storage. The stack depth can extend up to 256 bytes.

8.2.2. External RAM

On devices with 512 bytes total RAM, there are 256 bytes of on-chip RAM mapped into the external data memory space. All of these address locations may be accessed using the external move instruction (MOVX) and the data pointer (DPTR), or using MOVX indirect addressing mode. Note: The 16-bit MOVX instruction is also used for writes to the flash memory. See Section “10. Flash Memory” on page 61 for details. The MOVX instruction accesses XRAM by default.

For a 16-bit MOVX operation (@DPTR), the upper 8 bits of the 16-bit external data memory address word are "don't cares". As a result, addresses 0x0000 through 0x00FF are mapped modulo style over the entire 64 k external data memory address range. For example, the XRAM byte at address 0x0000 is shadowed at addresses 0x0100, 0x0200, 0x0300, 0x0400, etc.

Revision C C8051F850/1/3/4 and C8051F860/1/3/4 devices implement the upper four bytes of external RAM as a 32-bit Unique Identifier. More information can be found in “Device Identification and Unique Identifier” on page 68.

Table 9.2. Special Function Registers (Continued)

Register	Address	Register Description	Page
IE	0xA8	Interrupt Enable	75
IP	0xB8	Interrupt Priority	77
IT01CF	0xE4	INT0 / INT1 Configuration	150
OSCICL	0xC7	High Frequency Oscillator Calibration	127
OSCLCN	0xB1	Low Frequency Oscillator Control	128
P0	0x80	Port 0 Pin Latch	199
P0MASK	0xFE	Port 0 Mask	197
P0MAT	0xFD	Port 0 Match	198
P0MDIN	0xF1	Port 0 Input Mode	200
P0MDOUT	0xA4	Port 0 Output Mode	201
P0SKIP	0xD4	Port 0 Skip	202
P1	0x90	Port 1 Pin Latch	205
P1MASK	0xEE	Port 1 Mask	203
P1MAT	0xED	Port 1 Match	204
P1MDIN	0xF2	Port 1 Input Mode	206
P1MDOUT	0xA5	Port 1 Output Mode	207
P1SKIP	0xD5	Port 1 Skip	208
P2	0xA0	Port 2 Pin Latch	209
P2MDOUT	0xA6	Port 2 Output Mode	210
PCA0CENT	0x9E	PCA Center Alignment Enable	177
PCA0CLR	0x9C	PCA Comparator Clear Control	170
PCA0CN	0xD8	PCA Control	167
PCA0CPH0	0xFC	PCA Capture Module High Byte 0	175
PCA0CPH1	0xEA	PCA Capture Module High Byte 1	181
PCA0CPH2	0xEC	PCA Capture Module High Byte 2	183
PCA0CPL0	0xFB	PCA Capture Module Low Byte 0	174
PCA0CPL1	0xE9	PCA Capture Module Low Byte 1	180
PCA0CPL2	0xEB	PCA Capture Module Low Byte 2	182

10.2. Programming the Flash Memory

Writes to flash memory clear bits from logic 1 to logic 0, and can be performed on single byte locations. Flash erasures set bits back to logic 1, and occur only on full pages. The write and erase operations are automatically timed by hardware for proper execution; data polling to determine the end of the write/erase operation is not required. Code execution is stalled during a flash write/erase operation.

The simplest means of programming the flash memory is through the C2 interface using programming tools provided by Silicon Labs or a third party vendor. This is the only means for programming a non-initialized device.

To ensure the integrity of flash contents, it is strongly recommended that the on-chip supply monitor be enabled in any system that includes code that writes and/or erases flash memory from software.

10.2.1. Flash Lock and Key Functions

Flash writes and erases by user software are protected with a lock and key function. The Flash Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before flash operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, flash writes and erases will be disabled until the next system reset. Flash writes and erases will also be disabled if a flash write or erase is attempted before the key codes have been written properly. The flash lock resets after each write or erase; the key codes must be written again before a following flash operation can be performed.

10.2.2. Flash Erase Procedure

The flash memory can be programmed by software using the MOVX write instruction with the address and data byte to be programmed provided as normal operands. Before writing to flash memory using MOVX, flash write operations must be enabled by: (1) setting the PSWE Program Store Write Enable bit in the PSCTL register to logic 1 (this directs the MOVX writes to target flash memory); and (2) Writing the flash key codes in sequence to the Flash Lock register (FLKEY). The PSWE bit remains set until cleared by software.

A write to flash memory can clear bits to logic 0 but cannot set them; only an erase operation can set bits to logic 1 in flash. **A byte location to be programmed should be erased before a new value is written.** Erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire page, perform the following steps:

1. Disable interrupts (recommended).
2. Set the PSEE bit (register PSCTL).
3. Set the PSWE bit (register PSCTL).
4. Write the first key code to FLKEY: 0xA5.
5. Write the second key code to FLKEY: 0xF1.
6. Using the MOVX instruction, write a data byte to any location within the page to be erased.
7. Clear the PSWE and PSEE bits.

10.2.3. Flash Write Procedure

Flash bytes are programmed by software with the following sequence:

1. Disable interrupts (recommended).
2. Erase the flash page containing the target location, as described in Section 10.2.2.
3. Set the PSWE bit (register PSCTL).
4. Clear the PSEE bit (register PSCTL).
5. Write the first key code to FLKEY: 0xA5.
6. Write the second key code to FLKEY: 0xF1.
7. Using the MOVX instruction, write a single data byte to the desired location within the desired

In Burst Mode, tracking is determined by the settings in ADPWR and ADTK. Settling time requirements may need adjustment in some applications. Refer to “14.2.4. Settling Time Requirements” on page 90 for more details.

Notes:

- Setting ADTM to 1 will insert an additional 4 SAR clocks of tracking before each conversion, regardless of the settings of ADPWR and ADTK.
- When using Burst Mode, care must be taken to issue a convert start signal no faster than once every four SYSCLK periods. This includes external convert start signals. The ADC will ignore convert start signals which arrive before a burst is finished.

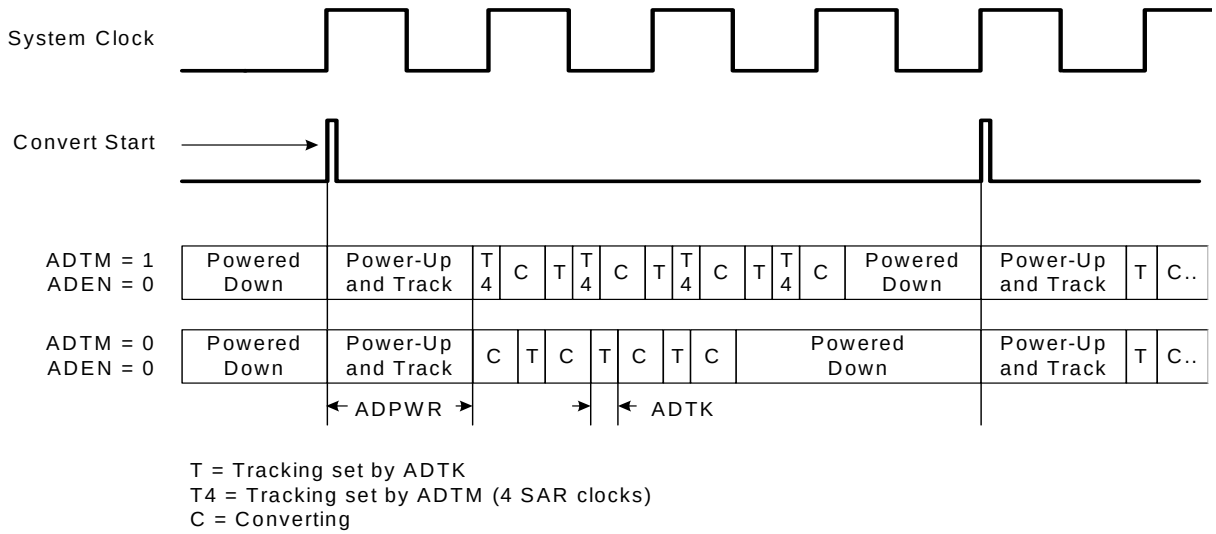

Figure 14.3. Burst Mode Tracking Example with Repeat Count Set to 4

14.2.4. Settling Time Requirements

A minimum amount of tracking time is required before each conversion can be performed, to allow the sampling capacitor voltage to settle. This tracking time is determined by the AMUX0 resistance, the ADC0 sampling capacitance, any external source resistance, and the accuracy required for the conversion. Note that when ADTM is set to 1, four SAR clocks are used for tracking at the start of every conversion. Large external source impedance will increase the required tracking time.

Figure 14.4 shows the equivalent ADC0 input circuit. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 14.1. When measuring any internal source, R_{TOTAL} reduces to R_{MUX}. See the electrical specification tables for ADC0 minimum settling time requirements as well as the mux impedance and sampling capacitor values.

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

Equation 14.1. ADC0 Settling Time Requirements

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 AMUX0 resistance and any external source resistance.

The ADSJST bits can be used to format the contents of the 16-bit accumulator. The accumulated result can be shifted right by 1, 2, or 3 bit positions. Based on the principles of oversampling and averaging, the effective ADC resolution increases by 1 bit each time the oversampling rate is increased by a factor of 4. The example below shows how to increase the effective ADC resolution by 1, 2, and 3 bits to obtain an effective ADC resolution of 11-bit, 12-bit, or 13-bit respectively without CPU intervention.

Input Voltage	**Repeat Count = 4** **Shift Right = 1** **11-Bit Result**	**Repeat Count = 16** **Shift Right = 2** **12-Bit Result**	**Repeat Count = 64** **Shift Right = 3** **13-Bit Result**
V_{REF} x 1023/1024	0x07F7	0x0FFC	0x1FF8
V_{REF} x 512/1024	0x0400	0x0800	0x1000
V_{REF} x 511/1024	0x03FE	0x04FC	0x0FF8
0	0x0000	0x0000	0x0000

14.7. Programmable Window Detector

The ADC Programmable Window Detector continuously compares the ADC0 output registers to user-programmed limits, and notifies the system when a desired condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (ADWINT in register ADC0CN0) can also be used in polled mode. The ADC0 Greater-Than (ADC0GTH, ADC0GTL) and Less-Than (ADC0LTH, ADC0LTL) registers hold the comparison values. The window detector flag can be programmed to indicate when measured data is inside or outside of the user-programmed limits, depending on the contents of the ADC0 Less-Than and ADC0 Greater-Than registers.

14.7.1. Window Detector In Single-Ended Mode

Figure 14.6 shows two example window comparisons for right-justified data, with ADC0LTH:ADC0LTL = 0x0080 (128d) and ADC0GTH:ADC0GTL = 0x0040 (64d). The input voltage can range from 0 to VREF x (1023/1024) with respect to GND, and is represented by a 10-bit unsigned integer value. In the left example, an ADWINT interrupt will be generated if the ADC0 conversion word (ADC0H:ADC0L) is within the range defined by ADC0GTH:ADC0GTL and ADC0LTH:ADC0LTL (if 0x0040 < ADC0H:ADC0L < 0x0080). In the right example, and ADWINT interrupt will be generated if the ADC0 conversion word is outside of the range defined by the ADC0GT and ADC0LT registers (if ADC0H:ADC0L < 0x0040 or ADC0H:ADC0L > 0x0080). Figure 14.7 shows an example using left-justified data with the same comparison values.

Register 14.2. ADC0CN1: ADC0 Control 1

Bit	7	6	5	4	3	2	1	0
Name	Reserved							ADCMBE
Type	R							RW
Reset	0	0	0	0	0	0	0	0
SFR Address: 0xB2								

Table 14.5. ADC0CN1 Register Bit Descriptions

Bit	Name	Function
7:1	Reserved	Must write reset value.
0	ADCMBE	**Common Mode Buffer Enable.** 0: Disable the common mode buffer. This setting should be used only if the tracking time of the signal is greater than 1.5 us. 1: Enable the common mode buffer. This setting should be used in most cases, and will give the best dynamic ADC performance. The common mode buffer must be enabled if signal tracking time is less than or equal to 1.5 us.

Register 14.4. ADC0AC: ADC0 Accumulator Configuration

Bit	7	6	5	4	3	2	1	0
Name	AD12BE	ADAE	ADSJST			ADRPT		
Type	RW	RW	RW			RW		
Reset	0	0	0	0	0	0	0	0
SFR Address: 0xB3								

Table 14.7. ADC0AC Register Bit Descriptions

Bit	Name	Function
7	AD12BE	**12-Bit Mode Enable.** Enables 12-bit Mode. In 12-bit mode, the ADC throughput is reduced by a factor of 4. 0: 12-bit Mode Disabled. 1: 12-bit Mode Enabled.
6	ADAE	**Accumulate Enable.** Enables multiple conversions to be accumulated when burst mode is disabled. 0: ADC0H:ADC0L contain the result of the latest conversion when Burst Mode is disabled. 1: ADC0H:ADC0L contain the accumulated conversion results when Burst Mode is disabled. Software must write 0x0000 to ADC0H:ADC0L to clear the accumulated result.
5:3	ADSJST	**Accumulator Shift and Justify.** Specifies the format of data read from ADC0H:ADC0L. All remaining bit combinations are reserved. 000: Right justified. No shifting applied. 001: Right justified. Shifted right by 1 bit. 010: Right justified. Shifted right by 2 bits. 011: Right justified. Shifted right by 3 bits. 100: Left justified. No shifting applied. 101-111: Reserved.
2:0	ADRPT	**Repeat Count.** Selects the number of conversions to perform and accumulate in Burst Mode. This bit field must be set to 000 if Burst Mode is disabled. 000: Perform and Accumulate 1 conversion (not used in 12-bit mode). 001: Perform and Accumulate 4 conversions (1 conversion in 12-bit mode). 010: Perform and Accumulate 8 conversions (2 conversions in 12-bit mode). 011: Perform and Accumulate 16 conversions (4 conversions in 12-bit mode). 100: Perform and Accumulate 32 conversions (8 conversions in 12-bit mode). 101: Perform and Accumulate 64 conversions (16 conversions in 12-bit mode). 110-111: Reserved.

15.4. CPU Core Registers

Register 15.1. DPL: Data Pointer Low

Bit	7	6	5	4	3	2	1	0
Name	DPL							
Type	RW							
Reset	0	0	0	0	0	0	0	0
SFR Address: 0x82								

Table 15.2. DPL Register Bit Descriptions

Bit	Name	Function
7:0	DPL	**Data Pointer Low.** The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed flash memory or XRAM.

Register 15.6. PSW: Program Status Word

Bit	7	6	5	4	3	2	1	0
Name	CY	AC	F0	RS		OV	F1	PARITY
Type	RW	RW	RW	RW		RW	RW	R
Reset	0	0	0	0	0	0	0	0
SFR Address: 0xD0 (bit-addressable)								

Table 15.7. PSW Register Bit Descriptions

Bit	Name	Function
7	CY	**Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations.
6	AC	**Auxiliary Carry Flag.** This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations.
5	F0	**User Flag 0.** This is a bit-addressable, general purpose flag for use under software control.
4:3	RS	**Register Bank Select.** These bits select which register bank is used during register accesses. 00: Bank 0, Addresses 0x00-0x07 01: Bank 1, Addresses 0x08-0x0F 10: Bank 2, Addresses 0x10-0x17 11: Bank 3, Addresses 0x18-0x1F
2	OV	**Overflow Flag.** This bit is set to 1 under the following circumstances: 1. An ADD, ADDC, or SUBB instruction causes a sign-change overflow. 2. A MUL instruction results in an overflow (result is greater than 255). 3. A DIV instruction causes a divide-by-zero condition. The OV bit is cleared to 0 by the ADD, ADDC, SUBB, MUL, and DIV instructions in all other cases.
1	F1	**User Flag 1.** This is a bit-addressable, general purpose flag for use under software control.
0	PARITY	**Parity Flag.** This bit is set to logic 1 if the sum of the eight bits in the accumulator is odd and cleared if the sum is even.

Register 18.5. CRC0CNT: CRC0 Automatic Flash Sector Count

Bit	7	6	5	4	3	2	1	0
Name	CRCDN	Reserved		CRCCNT				
Type	R	R		RW				
Reset	1	0	0	0	0	0	0	0
SFR Address: 0xD3								

Table 18.6. CRC0CNT Register Bit Descriptions

Bit	Name	Function
7	CRCDN	**Automatic CRC Calculation Complete.** Set to 0 when a CRC calculation is in progress. Code execution is stopped during a CRC calculation; therefore, reads from firmware will always return 1.
6:5	Reserved	Must write reset value.
4:0	CRCCNT	**Automatic CRC Calculation Block Count.** These bits specify the number of flash blocks to include in an automatic CRC calculation. The last address of the last flash block included in the automatic CRC calculation is (CRCST+CRCCNT) x Block Size - 1. The block size is 256 bytes.

21.4.3. Port Drive Strength

Port drive strength can be controlled on a port-by-port basis using the PRTDRV register. Each port has a bit in PRTDRV to select the high or low drive strength setting for all pins on that port. By default, all ports are configured for high drive strength.

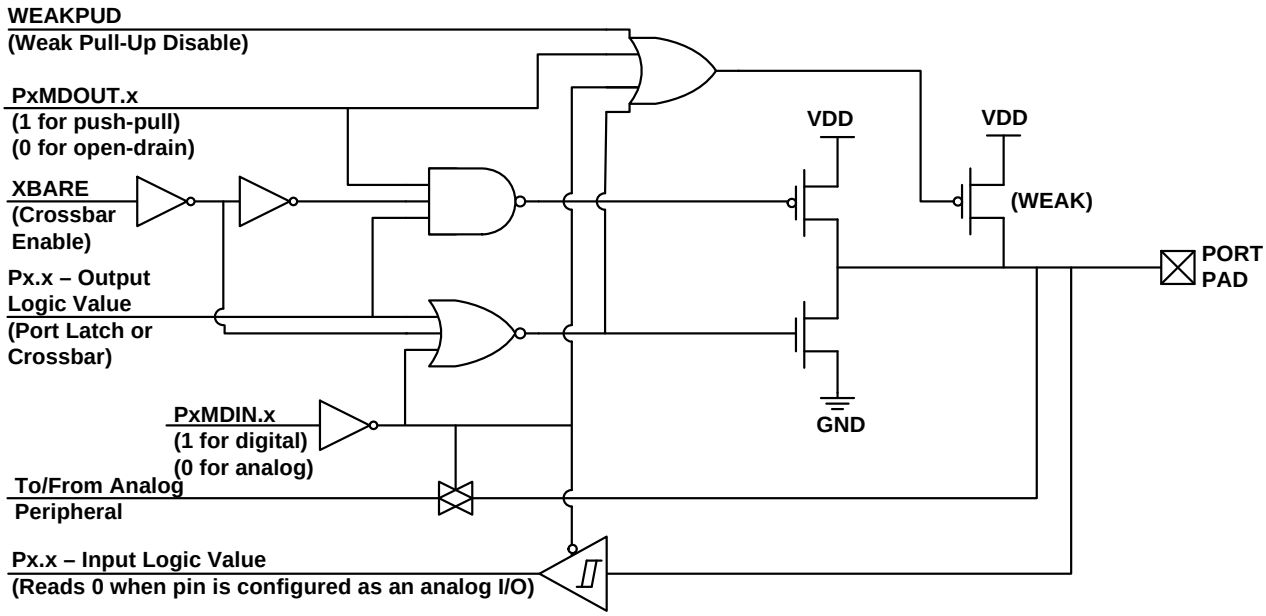

Figure 21.4. Port I/O Cell Block Diagram

21.5. Port Match

Port match functionality allows system events to be triggered by a logic value change on one or more port I/O pins. A software controlled value stored in the PnMATCH registers specifies the expected or normal logic values of the associated port pins (for example, P0MATCH.0 would correspond to P0.0). A port mismatch event occurs if the logic levels of the port's input pins no longer match the software controlled value. This allows software to be notified if a certain change or pattern occurs on the input pins regardless of the XBRn settings.

The PnMASK registers can be used to individually select which pins should be compared against the PnMATCH registers. A port mismatch event is generated if (Pn & PnMASK) does not equal (PnMATCH & PnMASK) for all ports with a PnMAT and PnMASK register.

A port mismatch event may be used to generate an interrupt or wake the device from idle mode. See the interrupts and power options chapters for more details on interrupt and wake-up sources.

21.6. Direct Read/Write Access to Port I/O Pins

All port I/O are accessed through corresponding special function registers (SFRs) that are both byte addressable and bit addressable. When writing to a port, the value written to the SFR is latched to maintain the output data value at each pin. When reading, the logic levels of the port's input pins are returned regardless of the XBRn settings (i.e., even when the pin is assigned to another signal by the crossbar, the port register can always read its corresponding port I/O pin). The exception to this is the execution of the read-modify-write instructions that target a Port Latch register as the destination. The read-modify-write instructions when operating on a port SFR are the following: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ and MOV, CLR or SETB, when the destination is an individual bit in a port SFR. For these instructions, the value of the latch register (not the pin) is read, modified, and written back to the SFR.

Register 21.12. P1MAT: Port 1 Match

Bit	7	6	5	4	3	2	1	0
Name	P1MAT							
Type	RW							
Reset	1	1	1	1	1	1	1	1
SFR Address: 0xED								

Table 21.15. P1MAT Register Bit Descriptions

Bit	Name	Function
7:0	P1MAT	**Port 1 Match Value.** Match comparison value used on P1 pins for bits in P1MASK which are set to 1. 0: P1.x pin logic value is compared with logic LOW. 1: P1.x pin logic value is compared with logic HIGH.
Note: Port 1 consists of 8 bits (P1.0-P1.7) on QSOP24 packages and 7 bits (P1.0-P1.6) on QFN20 packages and 4 bits (P1.0-P1.3) on SOIC16 packages.		

Table 23.3. SPI0CN Register Bit Descriptions

Bit	Name	Function
0	SPIEN	**SPI0 Enable.** 0: SPI disabled. 1: SPI enabled.

24.5.3. Write Sequence (Slave)

During a write sequence, an SMBus master writes data to a slave device. The slave in this transfer will be a receiver during the address byte, and a receiver during all data bytes. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode when a START followed by a slave address and direction bit (WRITE in this case) is received. If hardware ACK generation is disabled, upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. The software must respond to the received slave address with an ACK, or ignore the received slave address with a NACK. If hardware ACK generation is enabled, the hardware will apply the ACK for a slave address which matches the criteria set up by SMB0ADR and SMB0ADM. The interrupt will occur after the ACK cycle.

If the received slave address is ignored (by software or hardware), slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are received.

If hardware ACK generation is disabled, the ACKRQ is set to 1 and an interrupt is generated after each received byte. Software must write the ACK bit at that time to ACK or NACK the received byte.

With hardware ACK generation enabled, the SMBus hardware will automatically generate the ACK/NACK, and then post the interrupt. **It is important to note that the appropriate ACK or NACK value should be set up by the software prior to receiving the byte when hardware ACK generation is enabled.**

The interface exits Slave Receiver Mode after receiving a STOP. The interface will switch to Slave Transmitter Mode if SMB0DAT is written while an active Slave Receiver. Figure 24.7 shows a typical slave write sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the ‘data byte transferred’ interrupts occur at different places in the sequence, depending on whether hardware ACK generation is enabled. The interrupt occurs **before** the ACK with hardware ACK generation disabled, and **after** the ACK when hardware ACK generation is enabled.

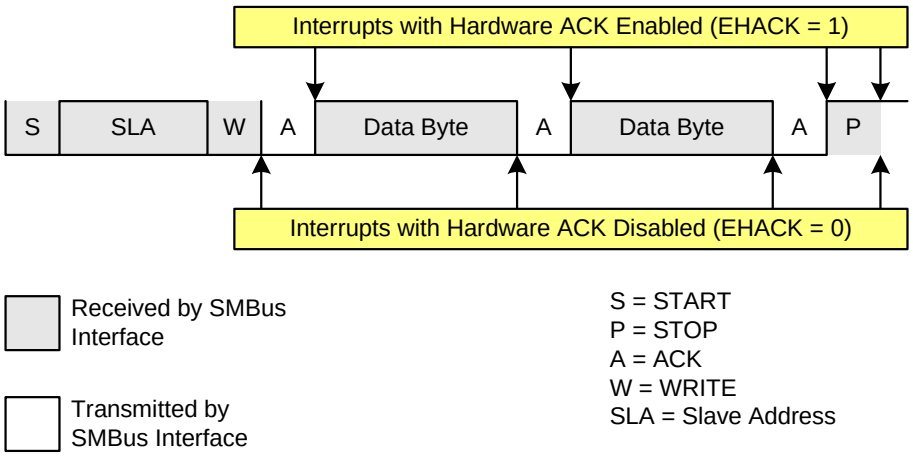

Figure 24.7. Typical Slave Write Sequence

Table 24.7. SMB0CF Register Bit Descriptions

Bit	Name	Function
1:0	SMBCS	**SMBus0 Clock Source Selection.** These two bits select the SMBus0 clock source, which is used to generate the SMBus0 bit rate. See the SMBus clock timing section for additional details. 00: Timer 0 Overflow 01: Timer 1 Overflow 10: Timer 2 High Byte Overflow 11: Timer 2 Low Byte Overflow

Register 25.5. TL1: Timer 1 Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TL1							
Type	RW							
Reset	0	0	0	0	0	0	0	0
SFR Address: 0x8B								

Table 25.7. TL1 Register Bit Descriptions

Bit	Name	Function
7:0	TL1	**Timer 1 Low Byte.** The TL1 register is the low byte of the 16-bit Timer 1.

Register 25.10. TMR2RLH: Timer 2 Reload High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLH							
Type	RW							
Reset	0	0	0	0	0	0	0	0
SFR Address: 0xCB								

Table 25.12. TMR2RLH Register Bit Descriptions

Bit	Name	Function
7:0	TMR2RLH	**Timer 2 Reload High Byte.** When operating in one of the auto-reload modes, TMR2RLH holds the reload value for the high byte of Timer 2 (TMR2H). When oeprating in capture mode, TMR2RLH is the captured value of TMR2H.

26. Universal Asynchronous Receiver/Transmitter (UART0)

UART0 is an asynchronous, full duplex serial port offering modes 1 and 3 of the standard 8051 UART. Enhanced baud rate support allows a wide range of clock sources to generate standard baud rates (details in Section “26.1. Enhanced Baud Rate Generation” on page 289). Received data buffering allows UART0 to start reception of a second incoming data byte before software has finished reading the previous data byte.

UART0 has two associated SFRs: Serial Control Register 0 (SCON0) and Serial Data Buffer 0 (SBUF0). The single SBUF0 location provides access to both transmit and receive registers. **Writes to SBUF0 always access the transmit register. Reads of SBUF0 always access the buffered receive register; it is not possible to read data from the transmit register.**

With UART0 interrupts enabled, an interrupt is generated each time a transmit is completed (TI is set in SCON0), or a data byte has been received (RI is set in SCON0). The UART0 interrupt flags are not cleared by hardware when the CPU vectors to the interrupt service routine. They must be cleared manually by software, allowing software to determine the cause of the UART0 interrupt (transmit complete or receive complete).

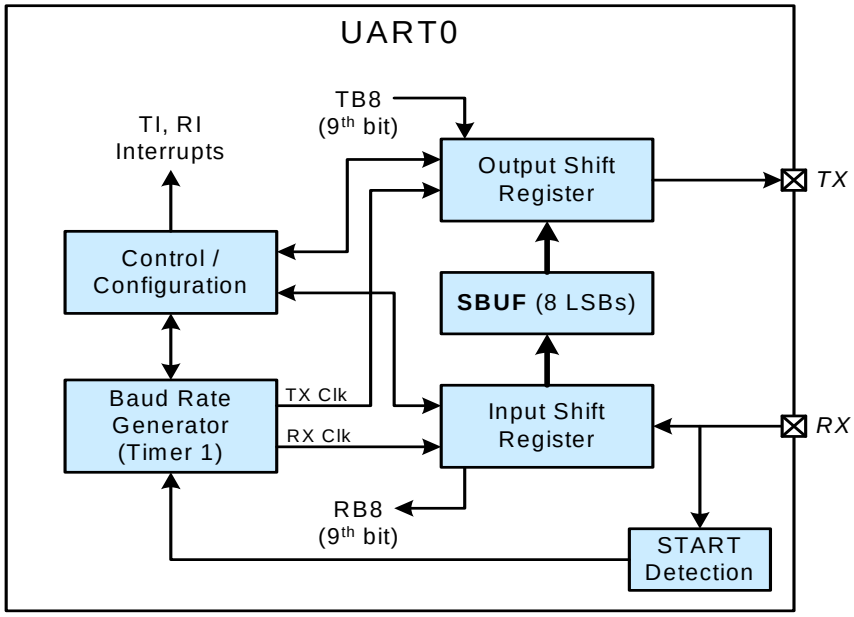

Figure 26.1. UART0 Block Diagram

26.1. Enhanced Baud Rate Generation

The UART0 baud rate is generated by Timer 1 in 8-bit auto-reload mode. The TX clock is generated by TL1; the RX clock is generated by a copy of TL1 (shown as RX Timer in Figure 26.2), which is not user-accessible. Both TX and RX Timer overflows are divided by two to generate the TX and RX baud rates. The RX Timer runs when Timer 1 is enabled, and uses the same reload value (TH1). However, an RX Timer reload is forced when a START condition is detected on the RX pin. This allows a receive to begin any time a START is detected, independent of the TX Timer state.

27.5. Watchdog Timer Control Registers

Register 27.1. WDTCN: Watchdog Timer Control

Bit	7	6	5	4	3	2	1	0
Name	WDTCN							
Type	RW							
Reset	0	0	0	1	0	1	1	1
SFR Address: 0x97								

Table 27.1. WDTCN Register Bit Descriptions

Bit	Name	Function
7:0	WDTCN	**WDT Control.** The WDT control field has different behavior for reads and writes. Read: When reading the WDTCN register, the lower three bits (WDTCN[2:0]) indicate the current timeout interval. Bit WDTCN.4 indicates whether the WDT is active (logic 1) or inactive (logic 0). Write: Writing the WDTCN register can set the timeout interval, enable the WDT, disable the WDT, reset the WDT, or lock the WDT to prevent disabling. Writing to WDTCN with the MSB (WDTCN.7) cleared to 0 will set the timeout interval to the value in bits WDTCN[2:0]. Writing 0xA5 both enables and reloads the WDT. Writing 0xDE followed within 4 system clocks by 0xAD disables the WDT. Writing 0xFF locks out the disable feature until the next device reset.