

Welcome to [E-XFL.COM](https://www.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	I ² C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	13
Program Memory Size	2KB (2K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 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	16-SOIC (0.154", 3.90mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f865-c-is

C8051F85x-86x

Table of Contents

1. Electrical Specifications.....	8
1.1. Electrical Characteristics	8
1.2. Typical Performance Curves	19
1.2.1. Operating Supply Current	19
1.2.2. ADC Supply Current.....	20
1.2.3. Port I/O Output Drive.....	21
1.3. Thermal Conditions	21
1.4. Absolute Maximum Ratings.....	22
2. System Overview	23
2.1. Power	25
2.1.1. LDO	25
2.1.2. Voltage Supply Monitor (VMON0).....	25
2.1.3. Device Power Modes	25
2.2. I/O	26
2.2.1. General Features	26
2.2.2. Crossbar.....	26
2.3. Clocking	27
2.4. Counters/Timers and PWM	27
2.4.1. Programmable Counter Array (PCA0)	27
2.4.2. Timers (Timer 0, Timer 1, Timer 2 and Timer 3)	27
2.4.3. Watchdog Timer (WDT0)	27
2.5. Communications and other Digital Peripherals	28
2.5.1. Universal Asynchronous Receiver/Transmitter (UART0).....	28
2.5.2. Serial Peripheral Interface (SPI0)	28
2.5.3. System Management Bus / I2C (SMBus0)	28
2.5.4. 16/32-bit CRC (CRC0)	28
2.6. Analog Peripherals	30
2.6.1. 12-Bit Analog-to-Digital Converter (ADC0)	30
2.6.2. Low Current Comparators (CMP0, CMP1)	30
2.7. Reset Sources	31
2.8. On-Chip Debugging.....	31
3. Pin Definitions.....	32
3.1. C8051F850/1/2/3/4/5 QSOP24 Pin Definitions	32
3.2. C8051F850/1/2/3/4/5 QFN20 Pin Definitions	36
3.3. C8051F860/1/2/3/4/5 SOIC16 Pin Definitions	39
4. Ordering Information	42
5. QSOP-24 Package Specifications	45
6. QFN-20 Package Specifications	47
7. SOIC-16 Package Specifications	50
8. Memory Organization	52
8.1. Program Memory.....	53
8.1.1. MOVX Instruction and Program Memory	53
8.2. Data Memory	53

1.2.2. ADC Supply Current

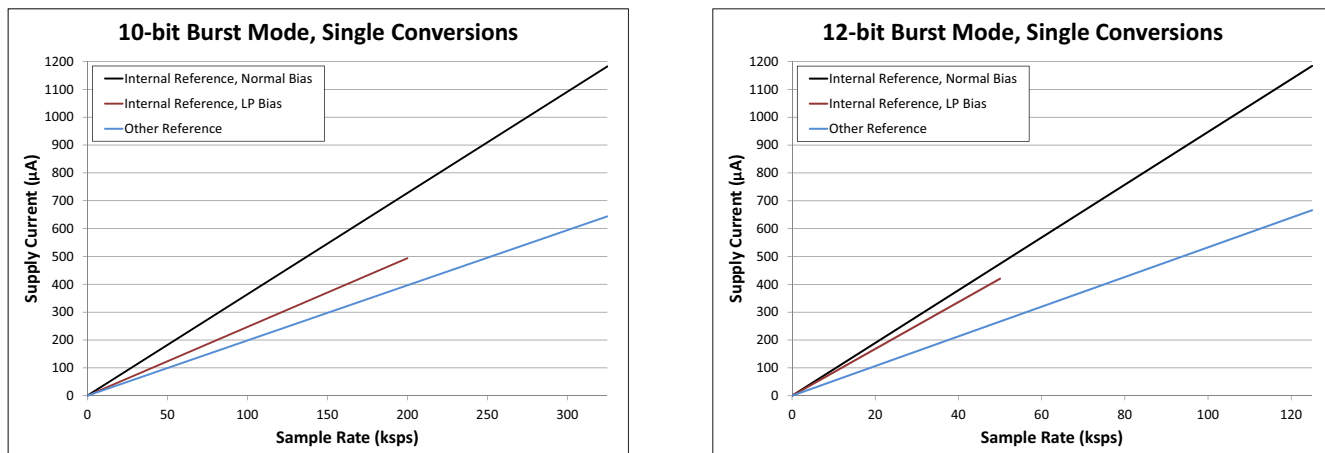


Figure 1.3. Typical ADC and Internal Reference Power Consumption in Burst Mode

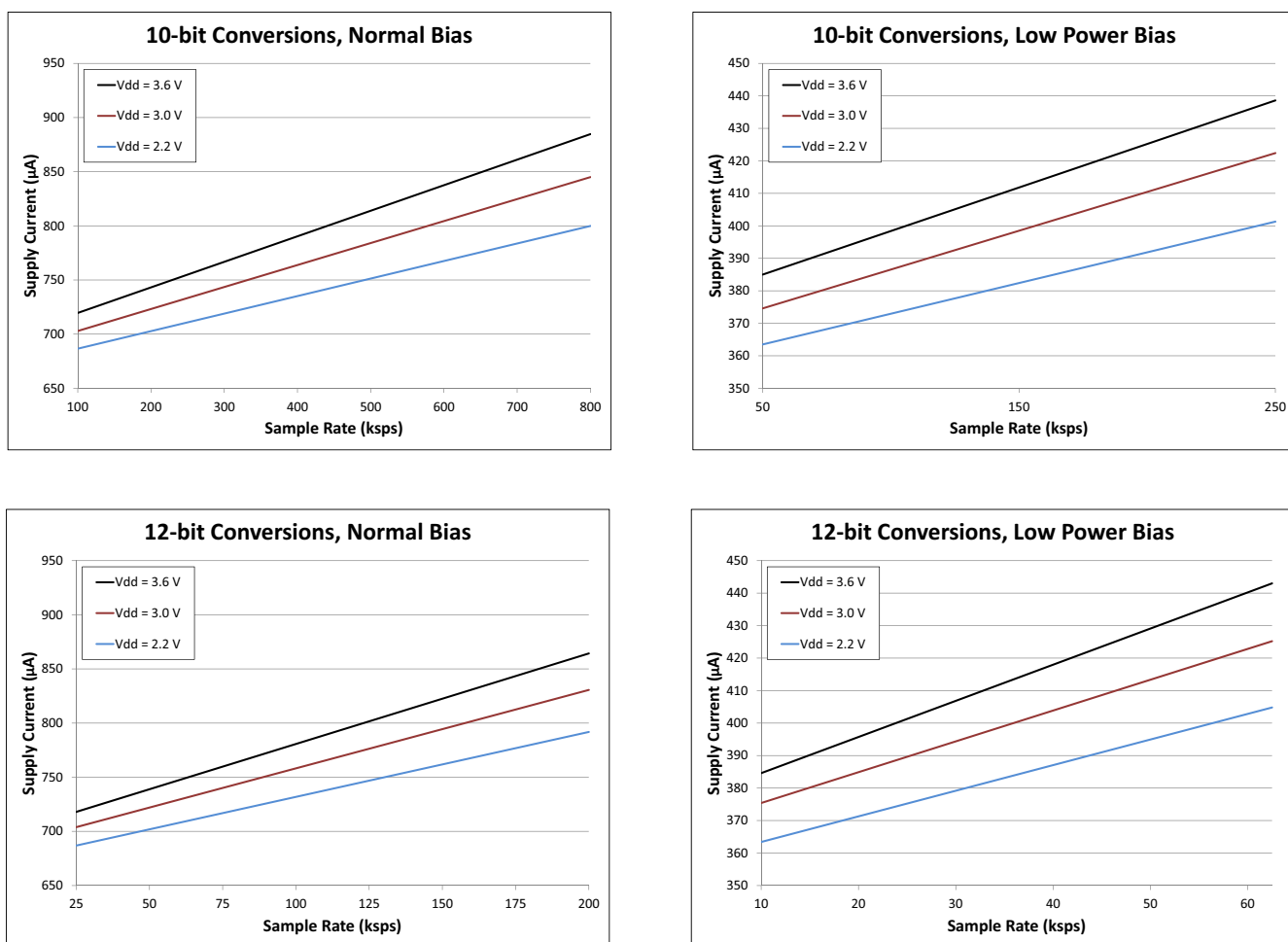


Figure 1.4. Typical ADC Power Consumption in Normal (Always-On) Mode

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
N/C	No Connection	1 13 24			

3.2. C8051F850/1/2/3/4/5 QFN20 Pin Definitions

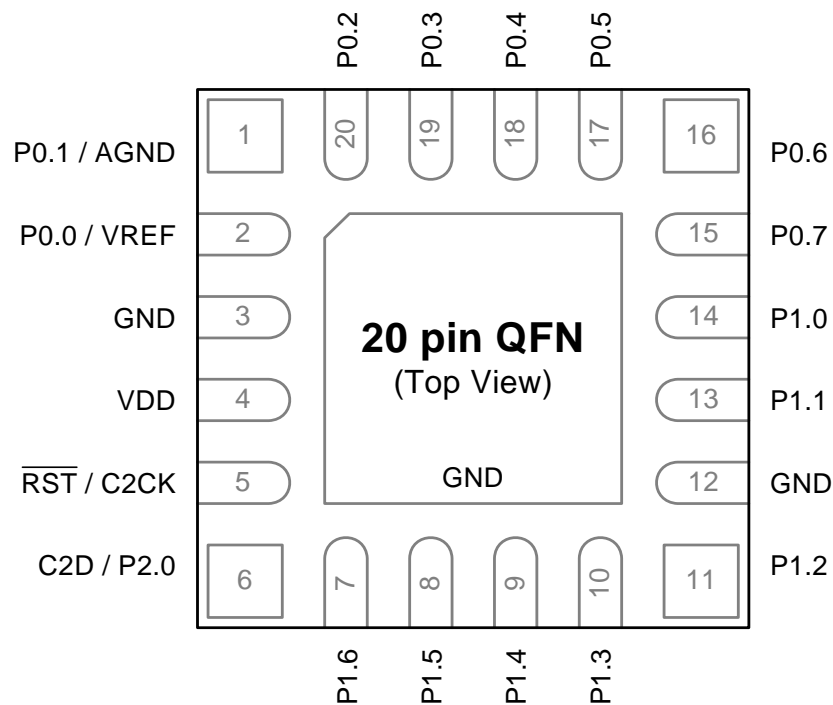


Figure 3.2. C8051F850/1/2/3/4/5-GM and C8051F850/1/2/3/4/5-IM Pinout

Table 3.2. Pin Definitions for C8051F850/1/2/3/4/5-GM and C8051F850/1/2/3/4/5-IM

Pin Name	Type	Pin Numbers	Crossbar Capability	Additional Digital Functions	Analog Functions
GND	Ground	Center 3 12			
VDD	Power	4			
RST / C2CK	Active-low Reset / C2 Debug Clock	5			

Table 3.3. Pin Definitions for C8051F860/1/2/3/4/5-GS and C8051F860/1/2/3/4/5-IS

Pin Name	Type	Pin Numbers	Crossbar Capability	Additional Digital Functions	Analog Functions
P0.1	Standard I/O	2	Yes	P0MAT.1 INT0.1 INT1.1	ADC0.1 CP0P.1 CP0N.1
P0.2	Standard I/O	1	Yes	P0MAT.2 INT0.2 INT1.2	ADC0.2 CP0P.2 CP0N.2
P0.3 / EXTCLK	Standard I/O / External CMOS Clock Input	16	Yes	P0MAT.3 EXTCLK INT0.3 INT1.3	ADC0.3 CP0P.3 CP0N.3
P0.4	Standard I/O	15	Yes	P0MAT.4 INT0.4 INT1.4	ADC0.4 CP0P.4 CP0N.4
P0.5	Standard I/O	14	Yes	P0MAT.5 INT0.5 INT1.5	ADC0.5 CP0P.5 CP0N.5
P0.6	Standard I/O	13	Yes	P0MAT.6 CNVSTR INT0.6 INT1.6	ADC0.6 CP1P.0 CP1N.0
P0.7	Standard I/O	12	Yes	P0MAT.7 INT0.7 INT1.7	ADC0.7 CP1P.1 CP1N.1
P1.0	Standard I/O	11	Yes	P1MAT.0	ADC0.8 CP1P.2 CP1N.2

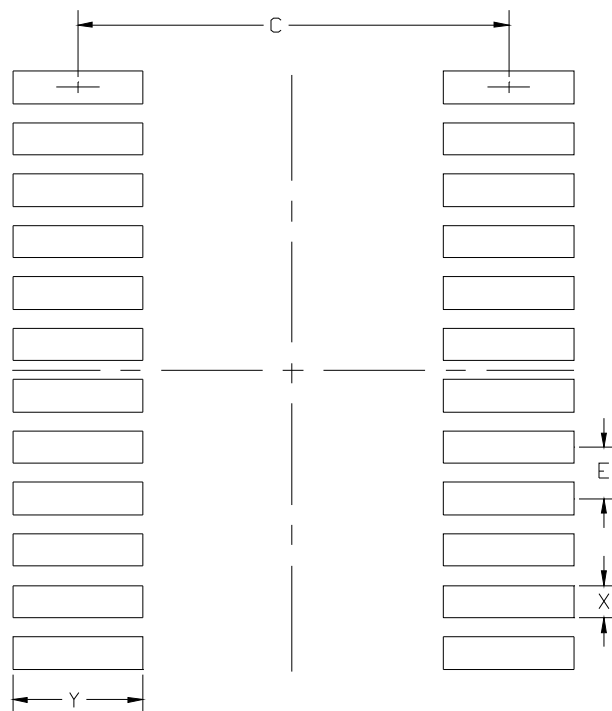


Figure 5.2. QSOP-24 PCB Land Pattern

Table 5.2. QSOP-24 PCB Land Pattern Dimensions

Dimension	Min	Max
C	5.20	5.30
E	0.635 BSC	
X	0.30	0.40
Y	1.50	1.60

Notes:

General

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. This land pattern design is based on the IPC-7351 guidelines.

Solder Mask Design

3. All metal pads are to be non-solder mask defined (NSMD). Clearance between the solder mask and the metal pad is to be 60 μm minimum, all the way around the pad.

Stencil Design

4. A stainless steel, laser-cut and electro-polished stencil with trapezoidal walls should be used to assure good solder paste release.
5. The stencil thickness should be 0.125 mm (5 mils).
6. The ratio of stencil aperture to land pad size should be 1:1 for all perimeter pads.

Card Assembly

7. A No-Clean, Type-3 solder paste is recommended.
8. The recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

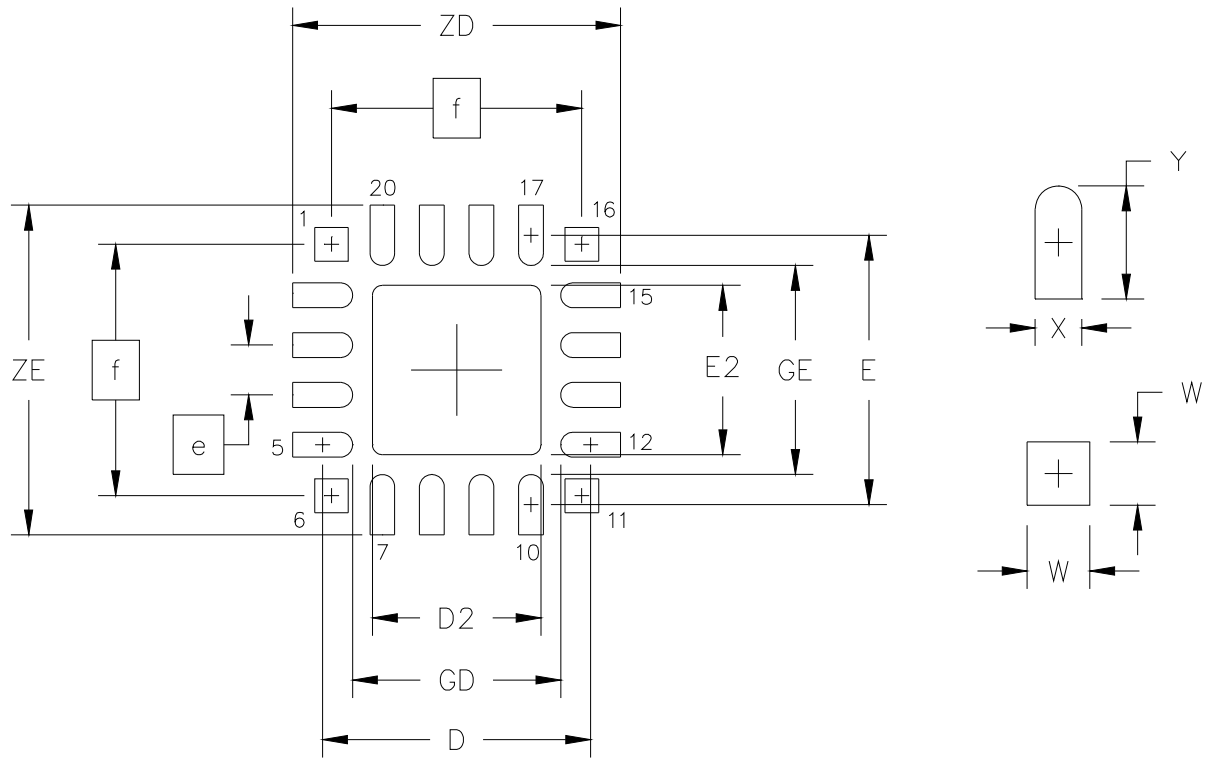


Figure 6.2. QFN-20 Landing Diagram

12. Interrupts

The C8051F85x/86x includes an extended interrupt system supporting multiple interrupt sources with two priority levels. The allocation of interrupt sources between on-chip peripherals and external input pins varies according to the specific version of the device. Each interrupt source has one or more associated interrupt-pending flag(s) located in an SFR. When a peripheral or external source meets a valid interrupt condition, the associated interrupt-pending flag is set to logic 1.

If interrupts are enabled for the source, an interrupt request is generated when the interrupt-pending flag is set. As soon as execution of the current instruction is complete, the CPU generates an LCALL to a predetermined address to begin execution of an interrupt service routine (ISR). Each ISR must end with an RETI instruction, which returns program execution to the next instruction that would have been executed if the interrupt request had not occurred. If interrupts are not enabled, the interrupt-pending flag is ignored by the hardware and program execution continues as normal. The interrupt-pending flag is set to logic 1 regardless of the interrupt's enable/disable state.

Each interrupt source can be individually enabled or disabled through the use of an associated interrupt enable bit in an SFR (IE and EIE1). However, interrupts must first be globally enabled by setting the EA bit in the IE register to logic 1 before the individual interrupt enables are recognized. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings.

Some interrupt-pending flags are automatically cleared by the hardware when the CPU vectors to the ISR. However, most are not cleared by the hardware and must be cleared by software before returning from the ISR. If an interrupt-pending flag remains set after the CPU completes the return-from-interrupt (RETI) instruction, a new interrupt request will be generated immediately and the CPU will re-enter the ISR after the completion of the next instruction.

12.1. MCU Interrupt Sources and Vectors

The C8051F85x/86x MCUs support interrupt sources for each peripheral on the device. Software can simulate an interrupt by setting any interrupt-pending flag to logic 1. If interrupts are enabled for the flag, an interrupt request will be generated and the CPU will vector to the ISR address associated with the interrupt-pending flag. MCU interrupt sources, associated vector addresses, priority order and control bits are summarized in Table 12.1. Refer to the datasheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

12.1.1. Interrupt Priorities

Each interrupt source can be individually programmed to one of two priority levels: low or high. A low priority interrupt service routine can be preempted by a high priority interrupt. A high priority interrupt cannot be preempted. Each interrupt has an associated interrupt priority bit in an SFR (IP or EIP1) used to configure its priority level. Low priority is the default. If two interrupts are recognized simultaneously, the interrupt with the higher priority is serviced first. If both interrupts have the same priority level, a fixed priority order is used to arbitrate, given in Table 12.1.

12.1.2. Interrupt Latency

Interrupt response time depends on the state of the CPU when the interrupt occurs. Pending interrupts are sampled and priority decoded each system clock cycle. Therefore, the fastest possible response time is 5 system clock cycles: 1 clock cycle to detect the interrupt and 4 clock cycles to complete the LCALL to the ISR. If an interrupt is pending when a RETI is executed, a single instruction is executed before an LCALL is made to service the pending interrupt. Therefore, the maximum response time for an interrupt (when no other interrupt is currently being serviced or the new interrupt is of greater priority) occurs when the CPU is performing an RETI instruction followed by a DIV as the next instruction. In this case, the response time is 18 system clock cycles: 1 clock cycle to detect the interrupt, 5 clock cycles to execute the RETI, 8 clock

13. Power Management and Internal Regulator

All internal circuitry on the C8051F85x/86x devices draws power from the VDD supply pin. Circuits with external connections (I/O pins, analog muxes) are powered directly from the VDD supply voltage, while most of the internal circuitry is supplied by an on-chip LDO regulator. The regulator output is fully internal to the device, and is available also as an ADC input or reference source for the comparators and ADC.

The devices support the standard 8051 power modes: idle and stop. For further power savings in stop mode, the internal LDO regulator may be disabled, shutting down the majority of the power nets on the device.

Although the C8051F85x/86x has idle and stop modes available, more control over the device power can be achieved by enabling/disabling individual peripherals as needed. Each analog peripheral can be disabled when not in use and placed in low power mode. Digital peripherals, such as timers and serial buses, have their clocks gated off and draw little power when they are not in use.

13.1. Power Modes

Idle mode halts the CPU while leaving the peripherals and clocks active. In stop mode, the CPU is halted, all interrupts and timers are inactive, and the internal oscillator is stopped (analog peripherals remain in their selected states; the external oscillator is not affected). 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 because the majority of the device is shut down with no clocks active. The Power Control Register (PCON) is used to control the C8051F85x/86x's Stop and Idle power management modes.

13.1.1. Idle Mode

Setting the Idle Mode Select bit (PCON.0) causes the hardware to halt the CPU and enter idle mode as soon as the instruction that sets the bit completes execution. 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 is asserted or a reset occurs. 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.

Note: If the instruction following the write of the IDLE bit is a single-byte instruction and an interrupt occurs during the execution phase of the instruction that sets the IDLE bit, the CPU may not wake from Idle mode when a future interrupt occurs. Therefore, instructions that set the IDLE bit should be followed by an instruction that has two or more opcode bytes, for example:

```
// in 'C':
PCON |= 0x01;           // set IDLE bit
PCON = PCON;            // ... followed by a 3-cycle dummy instruction

; in assembly:
ORL PCON, #01h          ; set IDLE bit
MOV PCON, PCON           ; ... followed by a 3-cycle dummy instruction
```

If enabled, the Watchdog Timer (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.

14.2. ADC Operation

The ADC is clocked by an adjustable conversion clock (SARCLK). SARCLK is a divided version of the selected system clock when burst mode is disabled (ADBMEN = 0), or a divided version of the high-frequency oscillator when burst mode is enabled (ADBMEN = 1). The clock divide value is determined by the ADSC bits in the ADC0CF register. In most applications, SARCLK should be adjusted to operate as fast as possible, without exceeding the maximum electrical specifications. The SARCLK does not directly determine sampling times or sampling rates.

14.2.1. Starting a Conversion

A conversion can be initiated in many ways, depending on the programmed states of the ADC0 Start of Conversion Mode field (ADCM) in register ADC0CN0. Conversions may be initiated by one of the following:

1. Writing a 1 to the ADBUSY bit of register ADC0CN0 (software-triggered)
2. A timer overflow (see the ADC0CN0 register and the timer section for timer options)
3. A rising edge on the CNVSTR input signal (external pin-triggered)

Writing a 1 to ADBUSY provides software control of ADC0 whereby conversions are performed "on-demand". All other trigger sources occur autonomous to code execution. When the conversion is complete, the ADC posts the result to its output register and sets the ADC interrupt flag (ADINT). ADINT may be used to trigger a system interrupts, if enabled, or polled by firmware.

During conversion, the ADBUSY bit is set to logic 1 and reset to logic 0 when the conversion is complete. However, when polling for ADC conversion completions, the ADC0 interrupt flag (ADINT) should be used instead of the ADBUSY bit. Converted data is available in the ADC0 data registers, ADC0H:ADC0L, when the conversion is complete.

Important Note About Using CNVSTR: When the CNVSTR input is used as the ADC0 conversion source, the associated port pin should be skipped in the crossbar settings.

14.2.2. Tracking Modes

Each ADC0 conversion must be preceded by a minimum tracking time in order for the converted result to be accurate. The minimum tracking time is given in the electrical specifications tables. The ADTM bit in register ADC0CN0 controls the ADC0 track-and-hold mode. In its default state when Burst Mode is disabled, the ADC0 input is continuously tracked, except when a conversion is in progress. A conversion will begin immediately when the start-of-conversion trigger occurs.

When the ADTM bit is logic 1, each conversion is preceded by a tracking period of 4 SAR clocks (after the start-of-conversion signal) for any internal (non-CNVSTR) conversion trigger source. When the CNVSTR signal is used to initiate conversions with ADTM set to 1, ADC0 tracks only when CNVSTR is low; conversion begins on the rising edge of CNVSTR (see Figure 14.2). Setting ADTM to 1 is primarily useful when AMUX settings are frequently changed and conversions are started using the ADBUSY bit.

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.

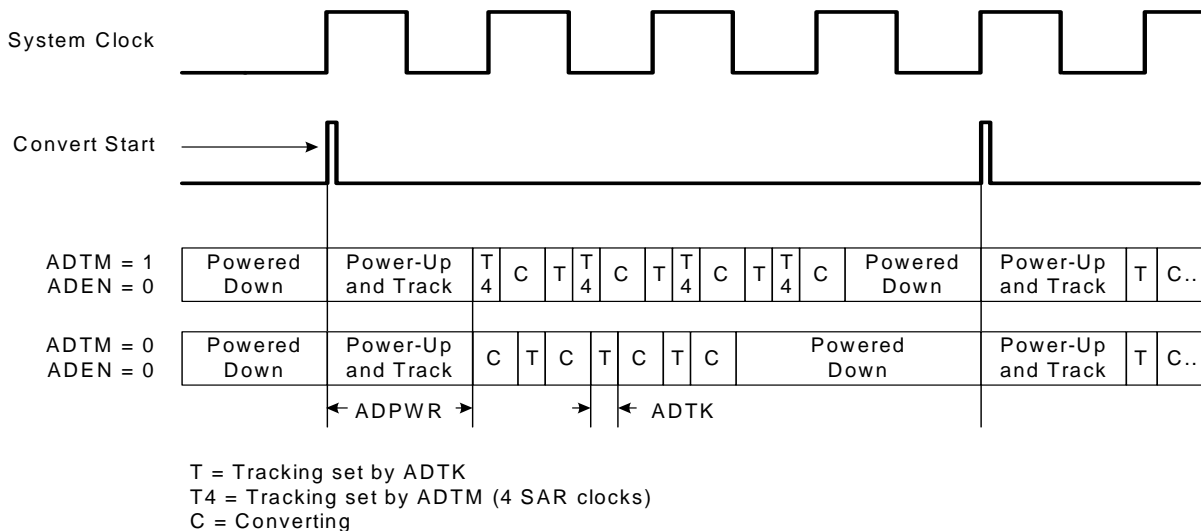


Figure 14.3. Burst Mode Tracking Example with Repeat Count Set to 4

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.

18.6. CRC Control Registers

Register 18.1. CRC0CN: CRC0 Control

Bit	7	6	5	4	3	2	1	0
Name	Reserved				CRCINIT	CRCVAL	Reserved	CRCPNT
Type	R				RW	RW	R	RW
Reset	0	0	0	1	0	0	0	0

SFR Address: 0xCE

Table 18.2. CRC0CN Register Bit Descriptions

Bit	Name	Function
7:4	Reserved	Must write reset value.
3	CRCINIT	CRC Result Initialization Bit. Writing a 1 to this bit initializes the entire CRC result based on CRCVAL.
2	CRCVAL	CRC Set Value Initialization Bit. This bit selects the set value of the CRC result. 0: CRC result is set to 0x0000 on write of 1 to CRCINIT. 1: CRC result is set to 0xFFFF on write of 1 to CRCINIT.
1	Reserved	Must write reset value.
0	CRCPNT	CRC Result Pointer. Specifies the byte of the CRC result to be read/written on the next access to CRC0DAT. This bit will automatically toggle upon each read or write. 0: CRC0DAT accesses bits 7-0 of the 16-bit CRC result. 1: CRC0DAT accesses bits 15-8 of the 16-bit CRC result.

Note: Upon initiation of an automatic CRC calculation, the three cycles following a write to CRC0CN that initiate a CRC operation must only contain instructions which execute in the same number of cycles as the number of bytes in the instruction. An example of such an instruction is a 3-byte MOV that targets the CRC0FLIP register. When programming in C, the dummy value written to CRC0FLIP should be a non-zero value to prevent the compiler from generating a 2-byte MOV instruction.

20. Programmable Counter Array (PCA0)

The Programmable Counter Array (PCA0) provides three channels of enhanced timer and PWM functionality while requiring less CPU intervention than standard counter/timers. The PCA consists of a dedicated 16-bit counter/timer and three 16-bit capture/compare modules. The counter/timer is driven by a programmable timebase that can select between seven sources: system clock, system clock divided by four, system clock divided by twelve, the external oscillator clock source divided by 8, low frequency oscillator divided by 8, Timer 0 overflows, or an external clock signal on the ECI input pin. Each capture/compare module may be configured to operate independently in one of six modes: Edge-Triggered Capture, Software Timer, High-Speed Output, Frequency Output, 8 to 11-Bit PWM, or 16-Bit PWM. Additionally, all PWM modes support both center and edge-aligned operation. The external oscillator and LFO oscillator clock options allow the PCA to be clocked by an external oscillator or the LFO while the internal oscillator drives the system clock. Each capture/compare module has its own associated I/O line (CEXn) which is routed through the crossbar to port I/O when enabled. The I/O signals have programmable polarity and Comparator 0 can optionally be used to perform a cycle-by-cycle kill operation on the PCA outputs. A PCA block diagram is shown in Figure 20.1

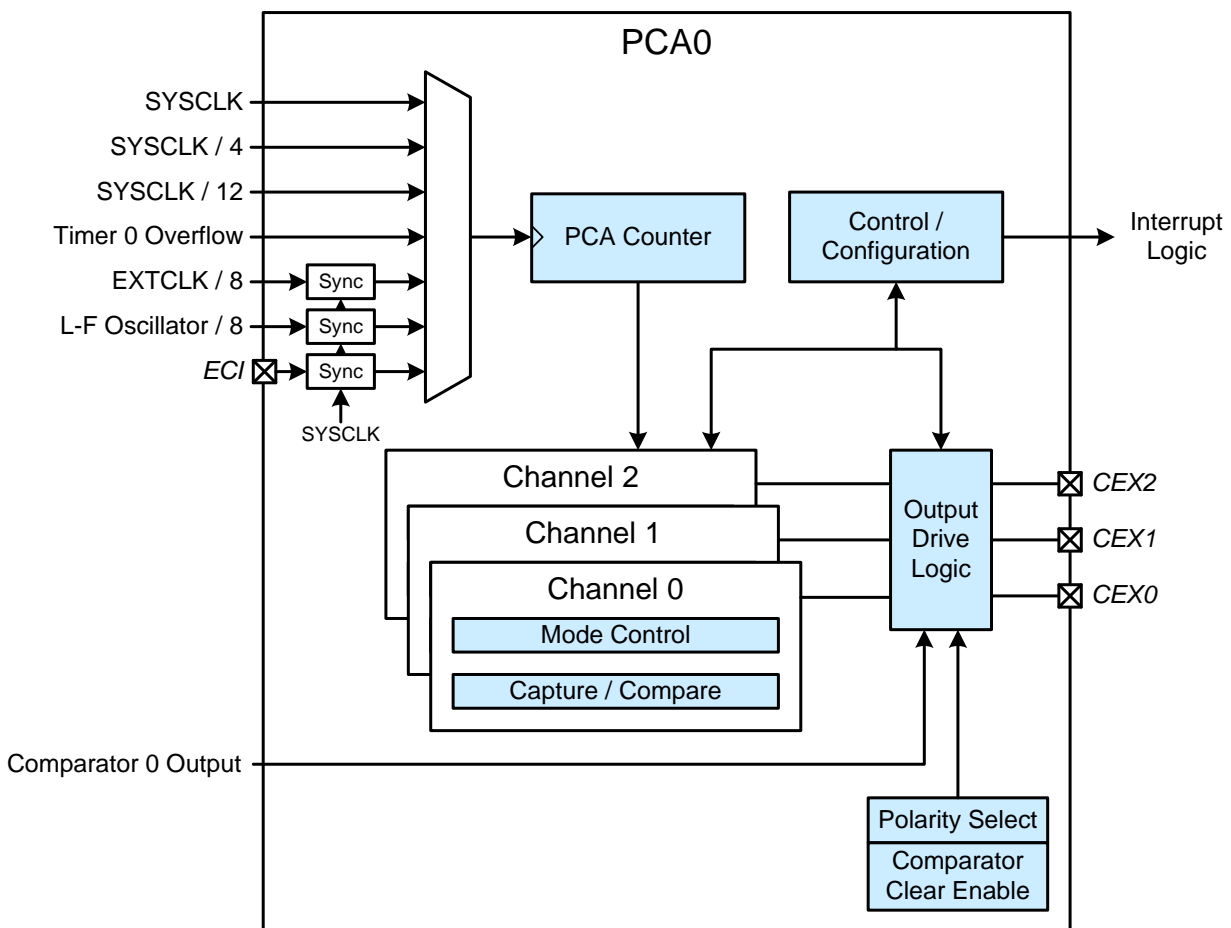


Figure 20.1. PCA0 Block Diagram

Register 20.7. PCA0H: PCA Counter/Timer High Byte

Bit	7	6	5	4	3	2	1	0
Name	PCA0H							
Type	RW							
Reset	0	0	0	0	0	0	0	0
SFR Address: 0xFA								

Table 20.9. PCA0H Register Bit Descriptions

Bit	Name	Function
7:0	PCA0H	PCA Counter/Timer High Byte. The PCA0H register holds the high byte (MSB) of the 16-bit PCA Counter/Timer. Reads of this register will read the contents of a snapshot register, whose contents are updated only when the contents of PCA0L are read.

Register 21.2. XBR1: Port I/O Crossbar 1

Bit	7	6	5	4	3	2	1	0
Name	Reserved		T2E	T1E	T0E	ECIE	PCA0ME	
Type	R	RW	RW	RW	RW	RW	RW	
Reset	0	0	0	0	0	0	0	0

SFR Address: 0xE2**Table 21.5. XBR1 Register Bit Descriptions**

Bit	Name	Function
7:6	Reserved	Must write reset value.
5	T2E	T2 Enable. 0: T2 unavailable at Port pin. 1: T2 routed to Port pin.
4	T1E	T1 Enable. 0: T1 unavailable at Port pin. 1: T1 routed to Port pin.
3	T0E	T0 Enable. 0: T0 unavailable at Port pin. 1: T0 routed to Port pin.
2	ECIE	PCA0 External Counter Input Enable. 0: ECI unavailable at Port pin. 1: ECI routed to Port pin.
1:0	PCA0ME	PCA Module I/O Enable Bits. 00: All PCA I/O unavailable at Port pins. 01: CEX0 routed to Port pin. 10: CEX0, CEX1 routed to Port pins. 11: CEX0, CEX1, CEX2 routed to Port pins.

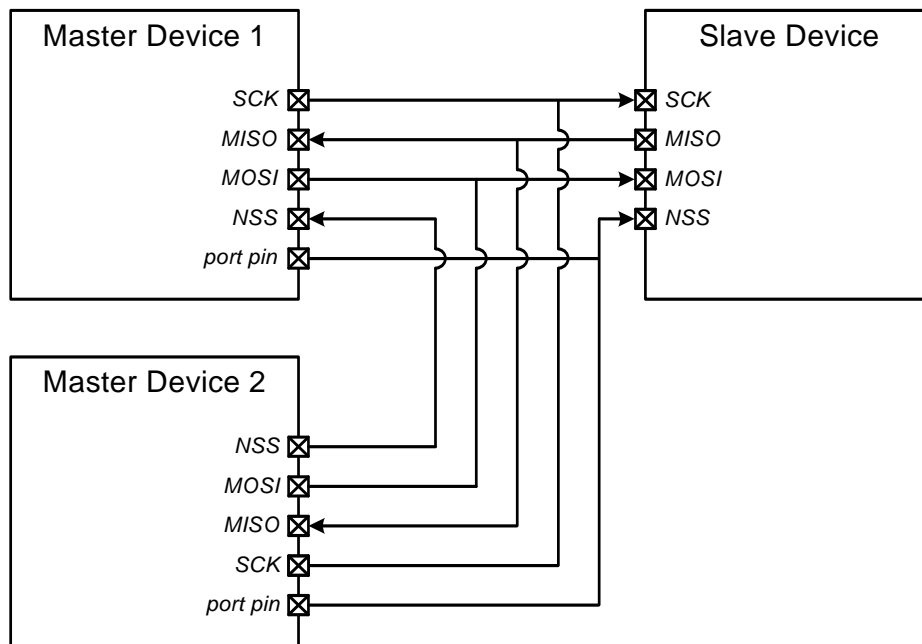


Figure 23.2. Multiple-Master Mode Connection Diagram

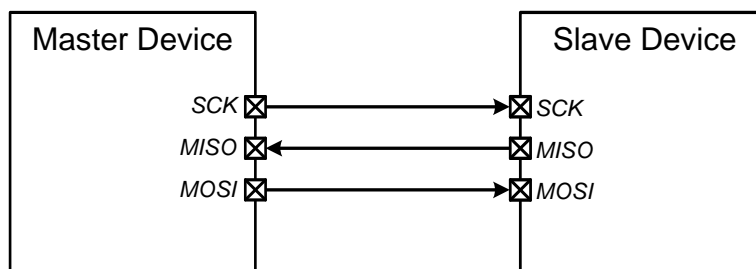


Figure 23.3. 3-Wire Single Master and 3-Wire Single Slave Mode Connection Diagram

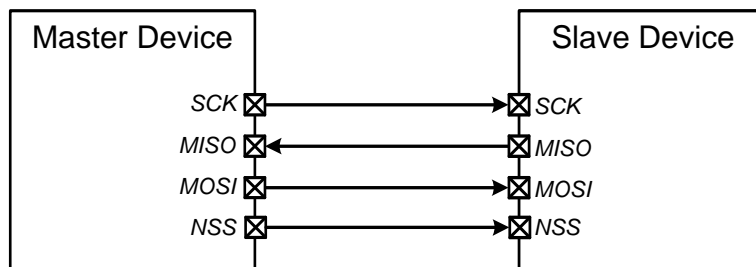


Figure 23.4. 4-Wire Single Master Mode and 4-Wire Slave Mode Connection Diagram

24. System Management Bus / I²C (SMBus0)

The SMBus I/O interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the I²C serial bus.

Reads and writes to the SMBus by the system controller are byte oriented with the SMBus interface autonomously controlling the serial transfer of the data. Data can be transferred at up to 1/20th of the system clock as a master or slave (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.

The SMBus may operate as a master and/or slave, and may function on a bus with multiple masters. The SMBus provides control of SDA (serial data), SCL (serial clock) generation and synchronization, arbitration logic, and START/STOP control and generation. The SMBus peripherals can be fully driven by software (i.e., software accepts/rejects slave addresses, and generates ACKs), or hardware slave address recognition and automatic ACK generation can be enabled to minimize software overhead. A block diagram of the SMBus0 peripheral is shown in Figure 24.1.

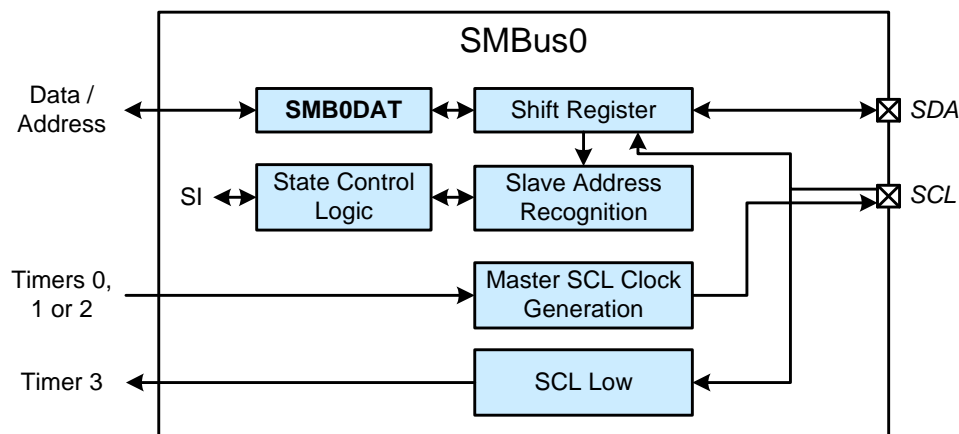


Figure 24.1. SMBus0 Block Diagram

24.5.2. Read Sequence (Master)

During a read sequence, an SMBus master reads data from a slave device. The master in this transfer will be a transmitter during the address byte, and a receiver during all data bytes. The SMBus interface generates the START condition and transmits the first 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 (READ). Serial data is then received from the slave on SDA while the SMBus outputs the serial clock. The slave transmits one or more bytes of serial data.

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.**

Writing a 1 to the ACK bit generates an ACK; writing a 0 generates a NACK. Software should write a 0 to the ACK bit for the last data transfer, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. The interface will switch to Master Transmitter Mode if SMB0DAT is written while an active Master Receiver. Figure 24.6 shows a typical master read 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.

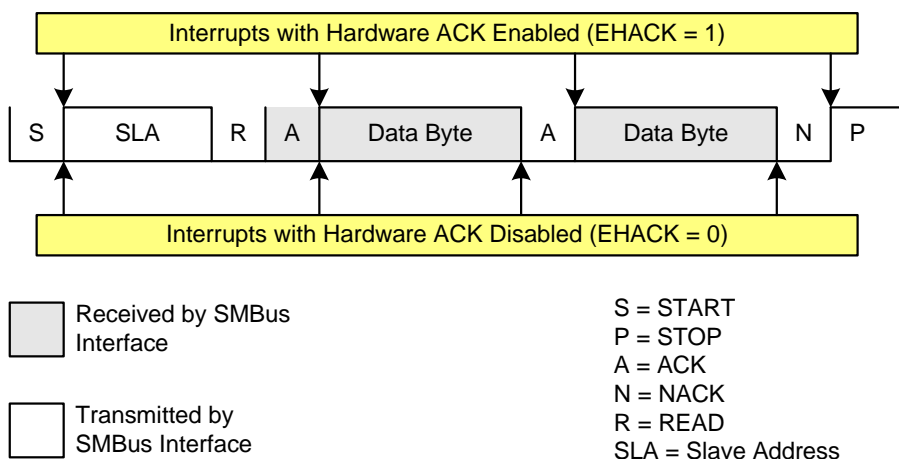


Figure 24.6. Typical Master Read Sequence

25.2. Timer 2 and Timer 3

Timer 2 and Timer 3 are functionally equivalent, with the only differences being the top-level connections to other parts of the system, as detailed in Table 25.1 and Table 25.2.

The timers are 16 bits wide, formed by two 8-bit SFRs: TMRnL (low byte) and TMRnH (high byte). Each timer may operate in 16-bit auto-reload mode or (split) 8-bit auto-reload mode. The TnSPLIT bit in TMRnCN defines the timer operation mode.

The timers may be clocked by the system clock, the system clock divided by 12, or the external oscillator source divided by 8. Note that the external oscillator source divided by 8 is synchronized with the system clock.

25.2.1. 16-bit Timer with Auto-Reload

When TnSPLIT is zero, the timer operates as a 16-bit timer with auto-reload. In this mode, the timer may be configured to clock from SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the timer reload registers (TMRnRLH and TMRnRLL) is loaded into the main timer count register as shown in Figure 25.4, and the High Byte Overflow Flag (TFnH) is set. If the timer interrupts are enabled, an interrupt will be generated on each timer overflow. Additionally, if the timer interrupts are enabled and the TFnLEN bit is set, an interrupt will be generated each time the lower 8 bits (TMRnL) overflow from 0xFF to 0x00.

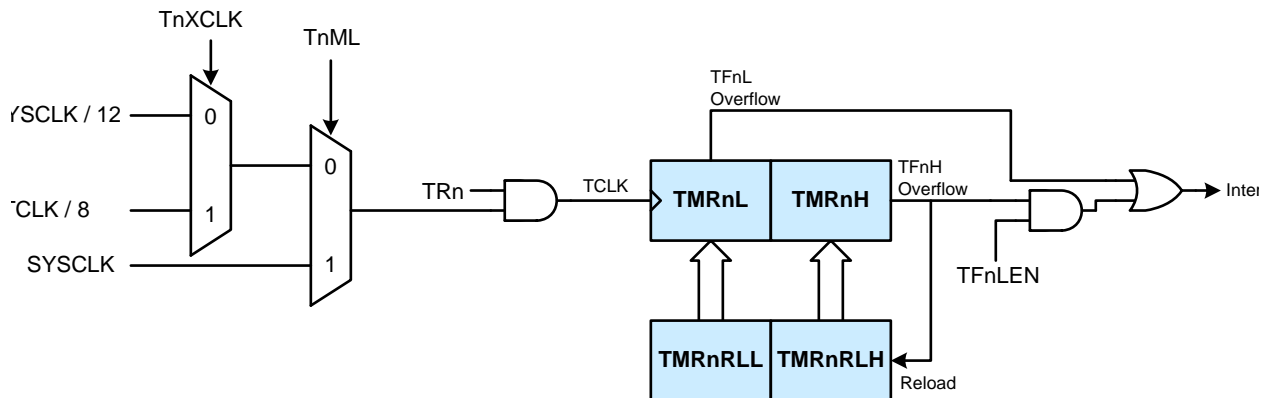


Figure 25.4. 16-Bit Mode Block Diagram

26.2. Operational Modes

UART0 provides standard asynchronous, full duplex communication. The UART mode (8-bit or 9-bit) is selected by the S0MODE bit in register SCON.

26.2.1. 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 TX pin and received at the RX pin. On receive, the eight data bits are stored in SBUF0 and the stop bit goes into RB8 in the SCON register.

Data transmission begins when software writes a data byte to the SBUF0 register. The TI Transmit Interrupt Flag is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN Receive Enable bit is set to logic 1. After the stop bit is received, the data byte will be loaded into the SBUF0 receive register if the following conditions are met: RI must be logic 0, and if MCE 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 SBUF0 receive register and the following overrun data bits are lost.

If these conditions are met, the eight bits of data is stored in SBUF0, the stop bit is stored in RB8 and the RI flag is set. If these conditions are not met, SBUF0 and RB8 will not be loaded and the RI flag will not be set. An interrupt will occur if enabled when either TI or RI is set.

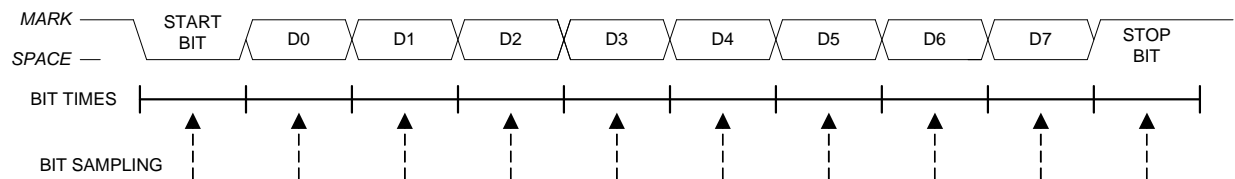


Figure 26.3. 8-Bit UART Timing Diagram