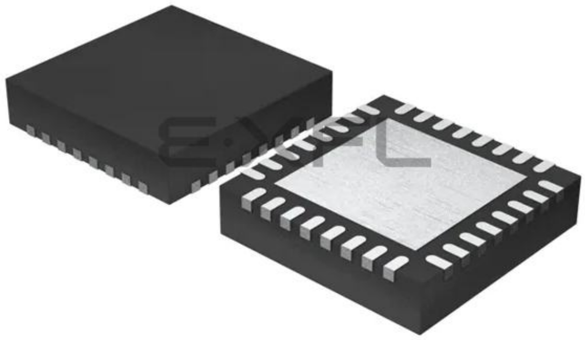


Welcome to E-XFL.COM

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	50MHz
Connectivity	SMBus (2-Wire/I²C), CANbus, LINbus, SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	25
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4.25K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.25V
Data Converters	A/D 25x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	32-VFQFN Exposed Pad
Supplier Device Package	32-QFN (5x5)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f506-im

Table 5.12. Comparator 0 and Comparator 1 Electrical Characteristics
VIO = 1.8 to 5.125 V, –40 to +125 °C unless otherwise noted.

Parameter	Conditions	Min	Typ	Max	Units
Response Time: Mode 0, Vcm* = 1.5 V	CPn+ – CPn– = 100 mV	—	310	—	ns
	CPn+ – CPn– = –100 mV	—	340	—	ns
Response Time: Mode 1, Vcm* = 1.5 V	CPn+ – CPn– = 100 mV	—	410	—	ns
	CPn+ – CPn– = –100 mV	—	510	—	ns
Response Time: Mode 2, Vcm* = 1.5 V	CPn+ – CPn– = 100 mV	—	480	—	ns
	CP0+ – CP0– = –100 mV	—	620	—	ns
Response Time: Mode 3, Vcm* = 1.5 V	CPn+ – CPn– = 100 mV	—	1600	—	ns
	CPn+ – CPn– = –100 mV	—	2600	—	ns
Common-Mode Rejection Ratio		—	1.7	8.9	mV/V
Positive Hysteresis 1	CPnHYP1–0 = 00	-2	0	2	mV
Positive Hysteresis 2	CPnHYP1–0 = 01	2	6	10	mV
Positive Hysteresis 3	CPnHYP1–0 = 10	5	11	20	mV
Positive Hysteresis 4	CPnHYP1–0 = 11	13	21	40	mV
Negative Hysteresis 1	CPnHYN1–0 = 00	-2	0	2	mV
Negative Hysteresis 2	CPnHYN1–0 = 01	2	6	10	mV
Negative Hysteresis 3	CPnHYN1–0 = 10	5	11	20	mV
Negative Hysteresis 4	CPnHYN1–0 = 11	13	21	40	mV
Inverting or Non-Inverting Input Voltage Range		–0.25	—	V_{IO} + 0.25	V
Input Capacitance		—	8	—	pF
Input Offset Voltage		–10	—	+10	mV
Power Supply					
Power Supply Rejection		—	0.33	—	mV/V
Power-Up Time		—	3	—	µs
Supply Current at DC	Mode 0	—	6.2	20	µA
	Mode 1	—	3.8	10	µA
	Mode 2	—	2.6	7.5	µA
	Mode 3	—	0.6	3	µA
***Note:** Vcm is the common-mode voltage on CP0+ and CP0–.					

SILICON LABS

6.1. Modes of Operation

In a typical system, ADC0 is configured using the following steps:

1. If a gain adjustment is required, refer to Section "6.3. Selectable Gain" on page 58.
2. Choose the start of conversion source.
3. Choose Normal Mode or Burst Mode operation.
4. If Burst Mode, choose the ADC0 Idle Power State and set the Power-Up Time.
5. Choose the tracking mode. Note that Pre-Tracking Mode can only be used with Normal Mode.
6. Calculate the required settling time and set the post convert-start tracking time using the AD0TK bits.
7. Choose the repeat count.
8. Choose the output word justification (Right-Justified or Left-Justified).
9. Enable or disable the End of Conversion and Window Comparator Interrupts.

6.1.1. Starting a Conversion

A conversion can be initiated in one of four ways, depending on the programmed states of the ADC0 Start of Conversion Mode bits (AD0CM1–0) in register ADC0CN. Conversions may be initiated by one of the following:

- Writing a 1 to the AD0BUSY bit of register ADC0CN
- A rising edge on the CNVSTR input signal (pin P0.1)
- A Timer 1 overflow (i.e., timed continuous conversions)
- A Timer 2 overflow (i.e., timed continuous conversions)

Writing a 1 to AD0BUSY provides software control of ADC0 whereby conversions are performed "on-demand." During conversion, the AD0BUSY bit is set to logic 1 and reset to logic 0 when the conversion is complete. The falling edge of AD0BUSY triggers an interrupt (when enabled) and sets the ADC0 interrupt flag (AD0INT). Note: When polling for ADC conversion completions, the ADC0 interrupt flag (AD0INT) should be used. Converted data is available in the ADC0 data registers, ADC0H:ADC0L, when bit AD0INT is logic 1. Note that when Timer 2 overflows are used as the conversion source, Low Byte overflows are used if Timer2 is in 8-bit mode; High byte overflows are used if Timer 2 is in 16-bit mode. See Section "26. Timers" on page 265 for timer configuration.

Important Note About Using CNVSTR: The CNVSTR input pin also functions as Port pin P0.1. When the CNVSTR input is used as the ADC0 conversion source, Port pin P0.1 should be skipped by the Digital Crossbar. To configure the Crossbar to skip P0.1, set to 1 Bit1 in register P0SKIP. See Section "20. Port Input/Output" on page 177 for details on Port I/O configuration.

6.1.2. Tracking Modes

Each ADC0 conversion must be preceded by a minimum tracking time for the converted result to be accurate. ADC0 has three tracking modes: Pre-Tracking, Post-Tracking, and Dual-Tracking. Pre-Tracking Mode provides the minimum delay between the convert start signal and end of conversion by tracking continuously before the convert start signal. This mode requires software management in order to meet minimum tracking requirements. In Post-Tracking Mode, a programmable tracking time starts after the convert start signal and is managed by hardware. Dual-Tracking Mode maximizes tracking time by tracking before and after the convert start signal. Figure 6.2 shows examples of the three tracking modes.

Pre-Tracking Mode is selected when AD0TM is set to 10b. Conversions are started immediately following the convert start signal. ADC0 is tracking continuously when not performing a conversion. Software must allow at least the minimum tracking time between each end of conversion and the next convert start signal. The minimum tracking time must also be met prior to the first convert start signal after ADC0 is enabled.

Table 11.1. CIP-51 Instruction Set Summary (Prefetch-Enabled)

Mnemonic	Description	Bytes	Clock Cycles
Arithmetic Operations			
ADD A, Rn	Add register to A	1	1
ADD A, direct	Add direct byte to A	2	2
ADD A, @Ri	Add indirect RAM to A	1	2
ADD A, #data	Add immediate to A	2	2
ADDC A, Rn	Add register to A with carry	1	1
ADDC A, direct	Add direct byte to A with carry	2	2
ADDC A, @Ri	Add indirect RAM to A with carry	1	2
ADDC A, #data	Add immediate to A with carry	2	2
SUBB A, Rn	Subtract register from A with borrow	1	1
SUBB A, direct	Subtract direct byte from A with borrow	2	2
SUBB A, @Ri	Subtract indirect RAM from A with borrow	1	2
SUBB A, #data	Subtract immediate from A with borrow	2	2
INC A	Increment A	1	1
INC Rn	Increment register	1	1
INC direct	Increment direct byte	2	2
INC @Ri	Increment indirect RAM	1	2
DEC A	Decrement A	1	1
DEC Rn	Decrement register	1	1
DEC direct	Decrement direct byte	2	2
DEC @Ri	Decrement indirect RAM	1	2
INC DPTR	Increment Data Pointer	1	1
MUL AB	Multiply A and B	1	4
DIV AB	Divide A by B	1	8
DA A	Decimal adjust A	1	1
Logical Operations			
ANL A, Rn	AND Register to A	1	1
ANL A, direct	AND direct byte to A	2	2
ANL A, @Ri	AND indirect RAM to A	1	2
ANL A, #data	AND immediate to A	2	2
ANL direct, A	AND A to direct byte	2	2
ANL direct, #data	AND immediate to direct byte	3	3
ORL A, Rn	OR Register to A	1	1
ORL A, direct	OR direct byte to A	2	2
ORL A, @Ri	OR indirect RAM to A	1	2
ORL A, #data	OR immediate to A	2	2
ORL direct, A	OR A to direct byte	2	2
ORL direct, #data	OR immediate to direct byte	3	3
XRL A, Rn	Exclusive-OR Register to A	1	1
XRL A, direct	Exclusive-OR direct byte to A	2	2
XRL A, @Ri	Exclusive-OR indirect RAM to A	1	2

Note: Certain instructions take a variable number of clock cycles to execute depending on instruction alignment and the FLRT setting (SFR Definition 15.3).

14. Interrupts

The C8051F50x/F51x devices include an extended interrupt system supporting a total of 18 interrupt sources with two priority levels. The allocation of interrupt sources between on-chip peripherals and external inputs 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, EIE1, or EIE2). However, interrupts must first be globally enabled by setting the EA bit (IE.7) 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.

Note: Any instruction that clears a bit to disable an interrupt should be immediately followed by an instruction that has two or more opcode bytes. Using EA (global interrupt enable) as an example:

```
// in 'C':
EA = 0; // clear EA bit.
EA = 0; // this is a dummy instruction with two-byte opcode.

; in assembly:
CLR EA ; clear EA bit.
CLR EA ; this is a dummy instruction with two-byte opcode.
```

For example, if an interrupt is posted during the execution phase of a "CLR EA" opcode (or any instruction which clears a bit to disable an interrupt source), and the instruction is followed by a single-cycle instruction, the interrupt may be taken. However, a read of the enable bit will return a 0 inside the interrupt service routine. When the bit-clearing opcode is followed by a multi-cycle instruction, the interrupt will not be taken.

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.

14.1. MCU Interrupt Sources and Vectors

The C8051F50x/F51x MCUs support 18 interrupt sources. 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 14.1. 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).

IE0 (TCON.1) and IE1 (TCON.3) serve as the interrupt-pending flags for the $\overline{INT0}$ and $\overline{INT1}$ external interrupts, respectively. If an $\overline{INT0}$ or $\overline{INT1}$ external interrupt is configured as edge-sensitive, the corresponding interrupt-pending flag is automatically cleared by the hardware when the CPU vectors to the ISR. When configured as level sensitive, the interrupt-pending flag remains logic 1 while the input is active as defined by the corresponding polarity bit (IN0PL or IN1PL); the flag remains logic 0 while the input is inactive. The external interrupt source must hold the input active until the interrupt request is recognized. It must then deactivate the interrupt request before execution of the ISR completes or another interrupt request will be generated.

SFR Definition 20.6. P1MASK: Port 1 Mask Register

Bit	7	6	5	4	3	2	1	0
Name	P1MASK[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xF4; SFR Page = 0x00

Bit	Name	Function
7:0	P1MASK[7:0]	**Port 1 Mask Value.** Selects P1 pins to be compared to the corresponding bits in P1MAT. 0: P1.n pin logic value is ignored and cannot cause a Port Mismatch event. 1: P1.n pin logic value is compared to P1MAT.n.

SFR Definition 20.7. P1MAT: Port 1 Match Register

Bit	7	6	5	4	3	2	1	0
Name	P1MAT[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xF3; SFR Page = 0x00

Bit	Name	Function
7:0	P1MAT[7:0]	**Port 1 Match Value.** Match comparison value used on Port 1 for bits in P1MAT which are set to 1. 0: P1.n pin logic value is compared with logic LOW. 1: P1.n pin logic value is compared with logic HIGH.

SFR Definition 20.21. P2MDIN: Port 2 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	P2MDIN[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xF3; SFR Page = 0x0F

Bit	Name	Function
7:0	P2MDIN[7:0]	**Analog Configuration Bits for P2.7–P2.0 (respectively).** Port pins configured for analog mode have their weak pull-up and digital receiver disabled. For analog mode, the pin also needs to be configured for open-drain mode in the P2MDOUT register. 0: Corresponding P2.n pin is configured for analog mode. 1: Corresponding P2.n pin is not configured for analog mode.

SFR Definition 20.22. P2MDOUT: Port 2 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P2MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xA6; SFR Page = 0x0F

Bit	Name	Function
7:0	P2MDOUT[7:0]	**Output Configuration Bits for P2.7–P2.0 (respectively).** These bits are ignored if the corresponding bit in register P2MDIN is logic 0. 0: Corresponding P2.n Output is open-drain. 1: Corresponding P2.n Output is push-pull.

Table 21.3. Autobaud Parameters Examples

System Clock (MHz)	Prescaler	Divider
25	1	312
24.5	1	306
24	1	300
22.1184	1	276
16	1	200
12.25	0	306
12	0	300
11.0592	0	276
8	0	200

21.3. LIN Master Mode Operation

The master node is responsible for the scheduling of messages and sends the header of each frame containing the SYNCH BREAK FIELD, SYNCH FIELD, and IDENTIFIER FIELD. The steps to schedule a message transmission or reception are listed below.

1. Load the 6-bit Identifier into the LIN0ID register.
2. Load the data length into the LIN0SIZE register. Set the value to the number of data bytes or "1111b" if the data length should be decoded from the identifier. Also, set the checksum type, classic or enhanced, in the same LIN0SIZE register.
3. Set the data direction by setting the TXRX bit (LIN0CTRL.5). Set the bit to 1 to perform a master transmit operation, or set the bit to 0 to perform a master receive operation.
4. If performing a master transmit operation, load the data bytes to transmit into the data buffer (LIN0DT1 to LIN0DT8).
5. Set the STREQ bit (LIN0CTRL.0) to start the message transfer. The LIN controller will schedule the message frame and request an interrupt if the message transfer is successfully completed or if an error has occurred.

This code segment shows the procedure to schedule a message in a transmission operation:

```
LIN0ADR   = 0x08;              // Point to LIN0CTRL
LIN0DAT  |= 0x20;              // Select to transmit data
LIN0ADR   = 0x0E;              // Point to LIN0ID
LIN0DAT   = 0x11;              // Load the ID, in this example 0x11
LIN0ADR   = 0x0B;              // Point to LIN0SIZE
LIN0DAT  = ( LIN0DAT & 0xF0 ) | 0x08;      // Load the size with 8

LIN0ADR  = 0x00;              // Point to Data buffer first byte
for (i=0; i<8; i++)
{
   LIN0DAT = i + 0x41;         // Load the buffer with ‘A’, ‘B’, ...
   LIN0ADR++;                  // Increment the address to the next buffer
}
LIN0ADR  = 0x08;              // Point to LIN0CTRL
LIN0DAT  = 0x01;              // Start Request
```

The application should perform the following steps when an interrupt is requested.

LIN Register Definition 21.9. LIN0DIV: LIN0 Divider Register

Bit	7	6	5	4	3	2	1	0
Name	DIVLSB[3:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

Indirect Address = 0x0C

Bit	Name	Function
7:0	DIVLSB	**LIN Baud Rate Divider Least Significant Bits.** The 8 least significant bits for the baud rate divider. The 9th and most significant bit is the DIV9 bit (LIN0MUL.0). The valid range for the divider is 200 to 511.

LIN Register Definition 21.10. LIN0MUL: LIN0 Multiplier Register

Bit	7	6	5	4	3	2	1	0
Name	PRESCL[1:0]		LINMUL[4:0]					DIV9
Type	R/W		R/W					R/W
Reset	1	1	1	1	1	1	1	1

Indirect Address = 0x0D

Bit	Name	Function
7:6	PRESCL[1:0]	**LIN Baud Rate Prescaler Bits.** These bits are the baud rate prescaler bits.
5:1	LINMUL[4:0]	**LIN Baud Rate Multiplier Bits.** These bits are the baud rate multiplier bits. These bits are not used in slave mode.
0	DIV9	**LIN Baud Rate Divider Most Significant Bit.** The most significant bit of the baud rate divider. The 8 least significant bits are in LIN0DIV. The valid range for the divider is 200 to 511.

Table 23.1. SMBus Clock Source Selection

SMBCS1	SMBCS0	SMBus Clock Source
0	0	Timer 0 Overflow
0	1	Timer 1 Overflow
1	0	Timer 2 High Byte Overflow
1	1	Timer 2 Low Byte Overflow

The SMBCS1–0 bits select the SMBus clock source, which is used only when operating as a master or when the Free Timeout detection is enabled. When operating as a master, overflows from the selected source determine the absolute minimum SCL low and high times as defined in Equation 23.1. Note that the selected clock source may be shared by other peripherals so long as the timer is left running at all times. For example, Timer 1 overflows may generate the SMBus and UART baud rates simultaneously. Timer configuration is covered in Section “26. Timers” on page 265.

$$T_{HighMin} = T_{LowMin} = \frac{1}{f_{ClockSourceOverflow}}$$

Equation 23.1. Minimum SCL High and Low Times

The selected clock source should be configured to establish the minimum SCL High and Low times as per Equation 23.1. When the interface is operating as a master (and SCL is not driven or extended by any other devices on the bus), the typical SMBus bit rate is approximated by Equation 23.2.

$$BitRate = \frac{f_{ClockSourceOverflow}}{3}$$

Equation 23.2. Typical SMBus Bit Rate

Figure 23.4 shows the typical SCL generation described by Equation 23.2. Notice that T_{HIGH} is typically twice as large as T_{LOW}. The actual SCL output may vary due to other devices on the bus (SCL may be extended low by slower slave devices, or driven low by contending master devices). The bit rate when operating as a master will never exceed the limits defined by equation Equation 23.1.

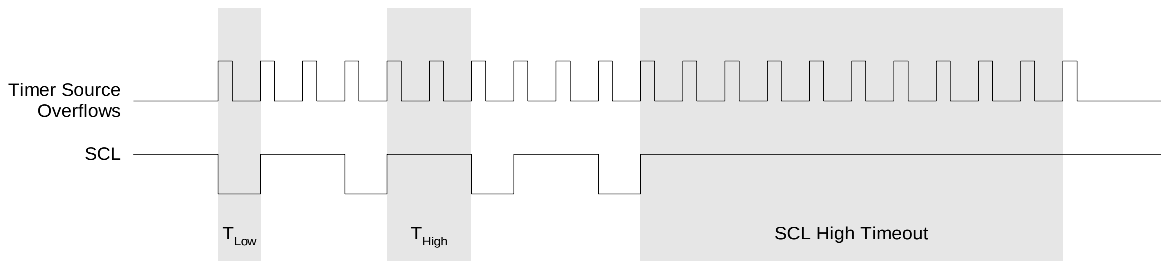


Figure 23.4. Typical SMBus SCL Generation

23.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see SFR Definition 23.2). The higher four bits of SMB0CN (MASTER, TXMODE, STA, and STO) form a status vector that can be used to jump to service routines. MASTER indicates whether a device is the master or slave during the current transfer. TXMODE indicates whether the device is transmitting or receiving data for the current byte.

STA and STO indicate that a START and/or STOP has been detected or generated since the last SMBus interrupt. STA and STO are also used to generate START and STOP conditions when operating as a master. Writing a 1 to STA will cause the SMBus interface to enter Master Mode and generate a START when the bus becomes free (STA is not cleared by hardware after the START is generated). Writing a 1 to STO while in Master Mode will cause the interface to generate a STOP and end the current transfer after the next ACK cycle. If STO and STA are both set (while in Master Mode), a STOP followed by a START will be generated.

As a receiver, writing the ACK bit defines the outgoing ACK value; as a transmitter, reading the ACK bit indicates the value received during the last ACK cycle. ACKRQ is set each time a byte is received, indicating that an outgoing ACK value is needed. When ACKRQ is set, software should write the desired outgoing value to the ACK bit before clearing SI. A NACK will be generated if software does not write the ACK bit before clearing SI. SDA will reflect the defined ACK value immediately following a write to the ACK bit; however SCL will remain low until SI is cleared. If a received slave address is not acknowledged, further slave events will be ignored until the next START is detected.

The ARBLOST bit indicates that the interface has lost an arbitration. This may occur anytime the interface is transmitting (master or slave). A lost arbitration while operating as a slave indicates a bus error condition. ARBLOST is cleared by hardware each time SI is cleared.

The SI bit (SMBus Interrupt Flag) is set at the beginning and end of each transfer, after each byte frame, or when an arbitration is lost; see Table 23.3 for more details.

Important Note About the SI Bit: The SMBus interface is stalled while SI is set; thus SCL is held low, and the bus is stalled until software clears SI.

23.5.4. Read Sequence (Slave)

During a read sequence, an SMBus master reads data from a slave device. The slave in this transfer will be a receiver during the address byte, and a transmitter during all data bytes. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode (to receive the slave address) when a START followed by a slave address and direction bit (READ in this case) is received. 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. The interrupt will occur after the ACK cycle.

If the received slave address is ignored, slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are transmitted. If the received slave address is acknowledged, data should be written to SMB0DAT to be transmitted. The interface enters Slave Transmitter Mode, and transmits one or more bytes of data. After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (Note: an error condition may be generated if SMB0DAT is written following a received NACK while in Slave Transmitter Mode). The interface exits Slave Transmitter Mode after receiving a STOP. Note that the interface will switch to Slave Receiver Mode if SMB0DAT is not written following a Slave Transmitter interrupt. Figure 23.8 shows a typical slave read sequence. Two transmitted data bytes are shown, though any number of bytes may be transmitted. Notice that all of the ‘data byte transferred’ interrupts occur **after** the ACK cycle in this mode.

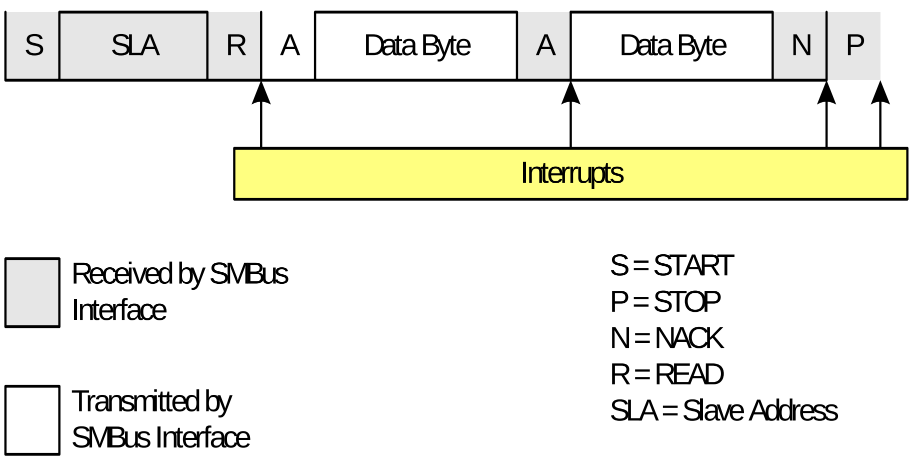


Figure 23.8. Typical Slave Read Sequence

23.6. SMBus Status Decoding

The current SMBus status can be easily decoded using the SMB0CN register. In the tables, STATUS VECTOR refers to the four upper bits of SMB0CN: MASTER, TXMODE, STA, and STO. The shown response options are only the typical responses; application-specific procedures are allowed as long as they conform to the SMBus specification. Highlighted responses are allowed by hardware but do not conform to the SMBus specification.

24.2. Data Format

UART0 has a number of available options for data formatting. Data transfers begin with a start bit (logic low), followed by the data bits (sent LSB-first), a parity or extra bit (if selected), and end with one or two stop bits (logic high). The data length is variable between 5 and 8 bits. A parity bit can be appended to the data, and automatically generated and detected by hardware for even, odd, mark, or space parity. The stop bit length is selectable between 1 and 2 bit times, and a multi-processor communication mode is available for implementing networked UART buses. All of the data formatting options can be configured using the SMOD0 register, shown in SFR Definition 24.2. Figure 24.2 shows the timing for a UART0 transaction without parity or an extra bit enabled. Figure 24.3 shows the timing for a UART0 transaction with parity enabled (PE0 = 1). Figure 24.4 is an example of a UART0 transaction when the extra bit is enabled (XBE0 = 1). Note that the extra bit feature is not available when parity is enabled, and the second stop bit is only an option for data lengths of 6, 7, or 8 bits.

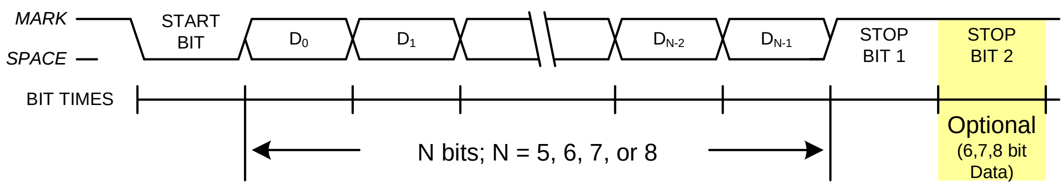


Figure 24.2. UART0 Timing Without Parity or Extra Bit

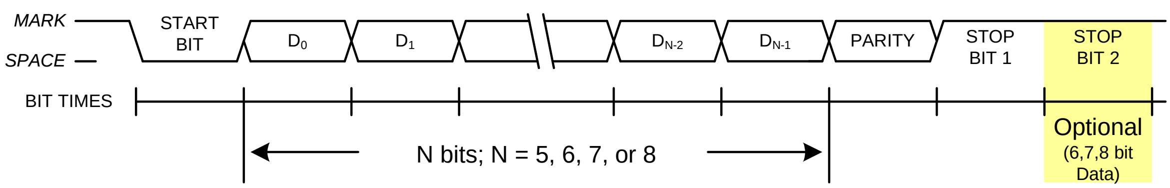


Figure 24.3. UART0 Timing With Parity

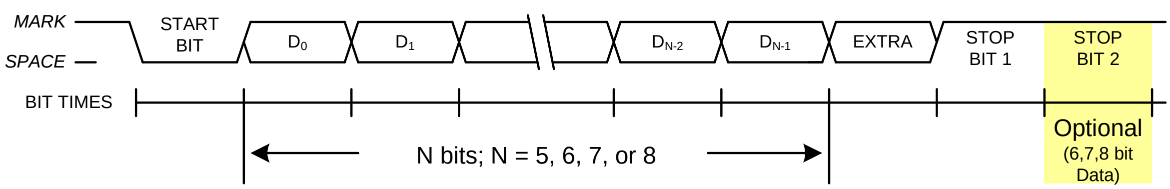


Figure 24.4. UART0 Timing With Extra Bit

SFR Definition 26.1. CKCON: Clock Control

Bit	7	6	5	4	3	2	1	0
Name	T3MH	T3ML	T2MH	T2ML	T1M	T0M	SCA[1:0]	
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0

SFR Address = 0x8E; SFR Page = All Pages

Bit	Name	Function
7	T3MH	**Timer 3 High Byte Clock Select.** Selects the clock supplied to the Timer 3 high byte (split 8-bit timer mode only). 0: Timer 3 high byte uses the clock defined by the T3XCLK bit in TMR3CN. 1: Timer 3 high byte uses the system clock.
6	T3ML	**Timer 3 Low Byte Clock Select.** Selects the clock supplied to Timer 3. Selects the clock supplied to the lower 8-bit timer in split 8-bit timer mode. 0: Timer 3 low byte uses the clock defined by the T3XCLK bit in TMR3CN. 1: Timer 3 low byte uses the system clock.
5	T2MH	**Timer 2 High Byte Clock Select.** Selects the clock supplied to the Timer 2 high byte (split 8-bit timer mode only). 0: Timer 2 high byte uses the clock defined by the T2XCLK bit in TMR2CN. 1: Timer 2 high byte uses the system clock.
4	T2ML	**Timer 2 Low Byte Clock Select.** Selects the clock supplied to Timer 2. If Timer 2 is configured in split 8-bit timer mode, this bit selects the clock supplied to the lower 8-bit timer. 0: Timer 2 low byte uses the clock defined by the T2XCLK bit in TMR2CN. 1: Timer 2 low byte uses the system clock.
3	T1	**Timer 1 Clock Select.** Selects the clock source supplied to Timer 1. Ignored when C/T1 is set to 1. 0: Timer 1 uses the clock defined by the prescale bits SCA[1:0]. 1: Timer 1 uses the system clock.
2	T0	**Timer 0 Clock Select.** Selects the clock source supplied to Timer 0. Ignored when C/T0 is set to 1. 0: Counter/Timer 0 uses the clock defined by the prescale bits SCA[1:0]. 1: Counter/Timer 0 uses the system clock.
1:0	SCA[1:0]	**Timer 0/1 Prescale Bits.** These bits control the Timer 0/1 Clock Prescaler: 00: System clock divided by 12 01: System clock divided by 4 10: System clock divided by 48 11: External clock divided by 8 (synchronized with the system clock)

26.1. Timer 0 and Timer 1

Each timer is implemented as a 16-bit register accessed as two separate bytes: a low byte (TL0 or TL1) and a high byte (TH0 or TH1). The Counter/Timer Control register (TCON) is used to enable Timer 0 and Timer 1 as well as indicate status. Timer 0 interrupts can be enabled by setting the ET0 bit in the IE register (Section “14.2. Interrupt Register Descriptions” on page 120); Timer 1 interrupts can be enabled by setting the ET1 bit in the IE register (Section “14.2. Interrupt Register Descriptions” on page 120). Both counter/timers operate in one of four primary modes selected by setting the Mode Select bits T1M1–T0M0 in the Counter/Timer Mode register (TMOD). Each timer can be configured independently. Each operating mode is described below.

26.1.1. Mode 0: 13-bit Counter/Timer

Timer 0 and Timer 1 operate as 13-bit counter/timers in Mode 0. The following describes the configuration and operation of Timer 0. However, both timers operate identically, and Timer 1 is configured in the same manner as described for Timer 0.

The TH0 register holds the eight MSBs of the 13-bit counter/timer. TL0 holds the five LSBs in bit positions TL0.4–TL0.0. The three upper bits of TL0 (TL0.7–TL0.5) are indeterminate and should be masked out or ignored when reading. As the 13-bit timer register increments and overflows from 0x1FFF (all ones) to 0x0000, the timer overflow flag TF0 (TCON.5) is set and an interrupt will occur if Timer 0 interrupts are enabled.

The C/T0 bit (TMOD.2) selects the counter/timer's clock source. When C/T0 is set to logic 1, high-to-low transitions at the selected Timer 0 input pin (T0) increment the timer register (Refer to Section “20.3. Priority Crossbar Decoder” on page 180 for information on selecting and configuring external I/O pins). Clearing C/T selects the clock defined by the T0M bit (CKCON.3). When T0M is set, Timer 0 is clocked by the system clock. When T0M is cleared, Timer 0 is clocked by the source selected by the Clock Scale bits in CKCON (see SFR Definition 26.1).

Setting the TR0 bit (TCON.4) enables the timer when either GATE0 (TMOD.3) is logic 0 or the input signal $\overline{\text{INT0}}$ is active as defined by bit IN0PL in register IT01CF (see SFR Definition 14.7). Setting GATE0 to 1 allows the timer to be controlled by the external input signal $\overline{\text{INT0}}$ (see Section “14.2. Interrupt Register Descriptions” on page 120), facilitating pulse width measurements.

TR0	**GATE0**	**$\overline{\text{INT0}}$**	**Counter/Timer**
0	X	X	Disabled
1	0	X	Enabled
1	1	0	Disabled
1	1	1	Enabled
Note: X = Don't Care			

Setting TR0 does not force the timer to reset. The timer registers should be loaded with the desired initial value before the timer is enabled.

TL1 and TH1 form the 13-bit register for Timer 1 in the same manner as described above for TL0 and TH0. Timer 1 is configured and controlled using the relevant TCON and TMOD bits just as with Timer 0. The input signal $\overline{\text{INT1}}$ is used with Timer 1; the $\overline{\text{INT1}}$ polarity is defined by bit IN1PL in register IT01CF (see SFR Definition 14.7).

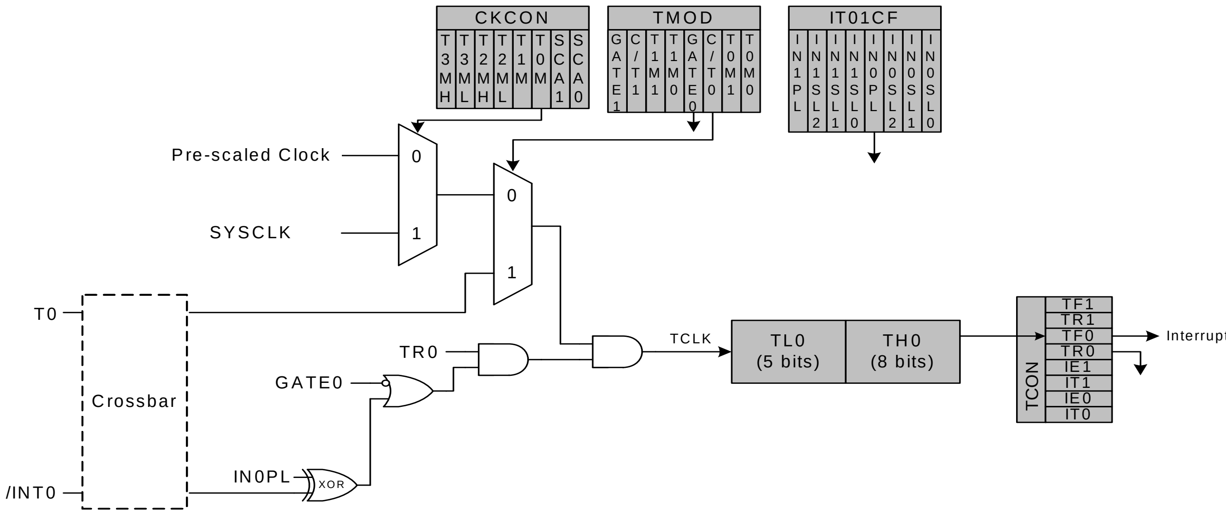


Figure 26.1. T0 Mode 0 Block Diagram

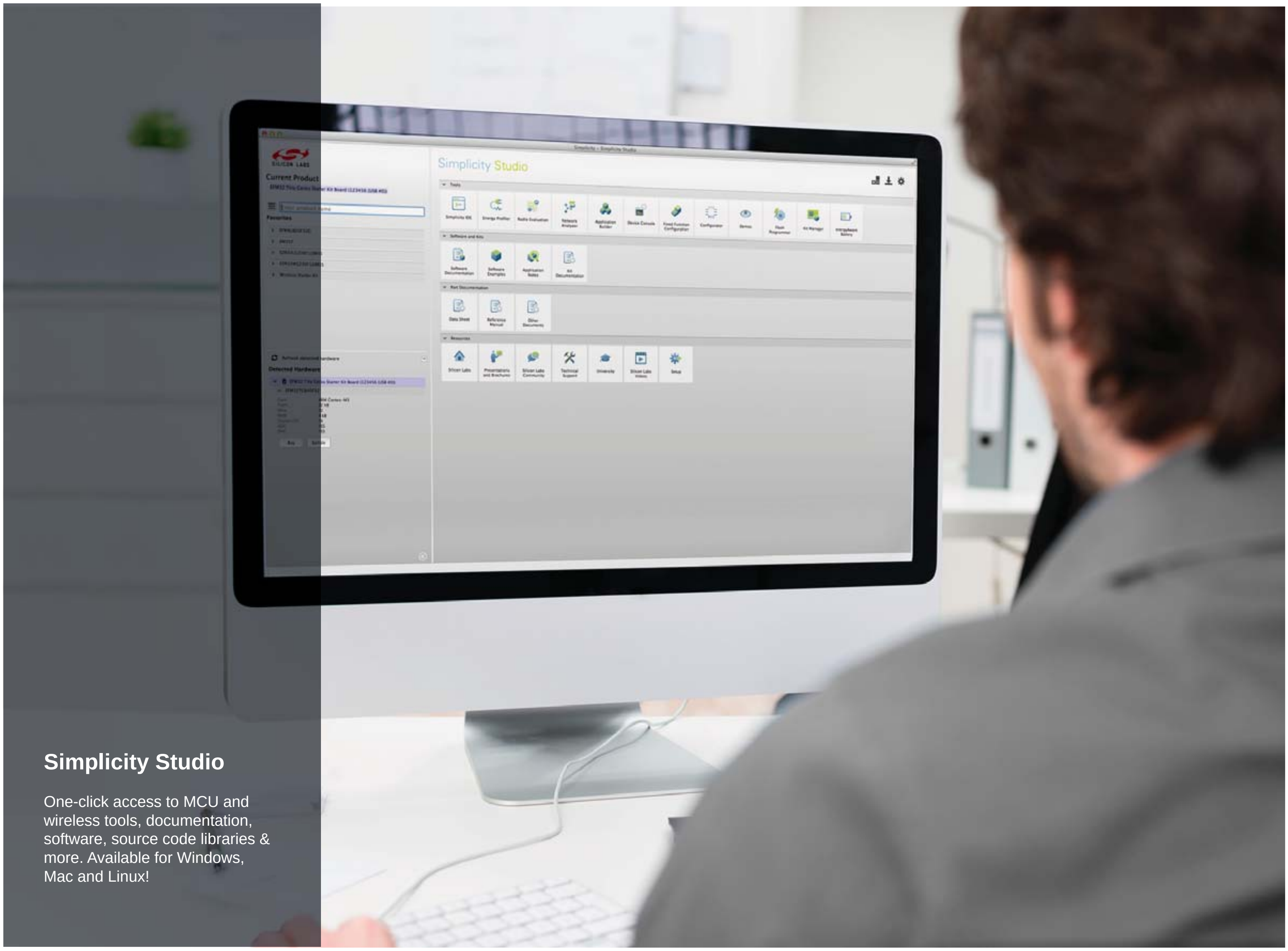
26.1.2. Mode 1: 16-bit Counter/Timer

Mode 1 operation is the same as Mode 0, except that the counter/timer registers use all 16 bits. The counter/timers are enabled and configured in Mode 1 in the same manner as for Mode 0.

26.1.3. Mode 2: 8-bit Counter/Timer with Auto-Reload

Mode 2 configures Timer 0 and Timer 1 to operate as 8-bit counter/timers with automatic reload of the start value. TL0 holds the count and TH0 holds the reload value. When the counter in TL0 overflows from all ones to 0x00, the timer overflow flag TF0 (TCON.5) is set and the counter in TL0 is reloaded from TH0. If Timer 0 interrupts are enabled, an interrupt will occur when the TF0 flag is set. The reload value in TH0 is not changed. TL0 must be initialized to the desired value before enabling the timer for the first count to be correct. When in Mode 2, Timer 1 operates identically to Timer 0.

Both counter/timers are enabled and configured in Mode 2 in the same manner as Mode 0. Setting the TR0 bit (TCON.4) enables the timer when either GATE0 (TMOD.3) is logic 0 or when the input signal $\overline{\text{INT0}}$ is active as defined by bit IN0PL in register IT01CF (see Section “14.3. External Interrupts INT0 and INT1” on page 126 for details on the external input signals $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$).



IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

Disclaimer
Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information
Silicon Laboratories Inc., Silicon Laboratories, Silicon Labs, SiLabs and the Silicon Labs logo, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem ®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com