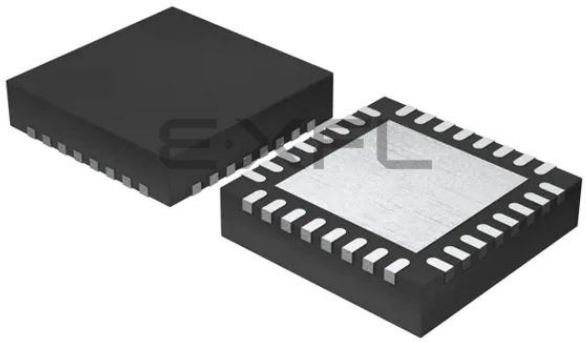

Welcome to E-XFL.COM

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), SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	25
Program Memory Size	96KB (96K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8.25K 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/c8051f587-im

Table 2.1. Product Selection Guide

Ordering Part Number	Flash Memory (kB)	CAN2.0B	LIN2.1	Digital Port I/Os	External Memory Interface	Package	Ordering Part Number	Flash Memory (kB)	CAN2.0B	LIN2.1	Digital Port I/Os	External Memory Interface	Package
C8051F580-IQ	128	✓	✓	40	✓	QFP-48	C8051F585-IQ	96	—	—	40	✓	QFP-48
C8051F580-IM	128	✓	✓	40	✓	QFN-48	C8051F585-IM	96	—	—	40	✓	QFN-48
C8051F581-IQ	128	—	—	40	✓	QFP-48	C8051F586-IQ	96	✓	✓	25	—	QFP-32
C8051F581-IM	128	—	—	40	✓	QFN-48	C8051F586-IM	96	✓	✓	25	—	QFN-32
C8051F582-IQ	128	✓	✓	25	—	QFP-32	C8051F587-IQ	96	—	—	25	—	QFP-32
C8051F582-IM	128	✓	✓	25	—	QFN-32	C8051F587-IM	96	—	—	25	—	QFN-32
C8051F583-IQ	128	—	—	25	—	QFP-32	C8051F588-IM	128	✓	✓	33	✓	QFN-40
C8051F583-IM	128	—	—	25	—	QFN-32	C8051F589-IM	128	—	—	33	✓	QFN-40
C8051F584-IQ	96	✓	✓	40	✓	QFP-48	C8051F590-IM	96	✓	✓	33	✓	QFN-40
C8051F584-IM	96	✓	✓	40	✓	QFN-48	C8051F591-IM	96	—	—	33	✓	QFN-40

Note: The suffix of the part number indicates the device rating and the package. All devices are RoHS compliant.

All of these devices are also available in an automotive version. For the automotive version, the -I in the ordering part number is replaced with -A. For example, the automotive version of the C8051F580-IM is the C8051F580-AM.

The -AM and -AQ devices receive full automotive quality production status, including AEC-Q100 qualification, registration with International Material Data System (IMDS) and Part Production Approval Process (PPAP) documentation. PPAP documentation is available at www.silabs.com with a registered and NDA approved user account. The -AM and -AQ devices enable high volume automotive OEM applications with their enhanced testing and processing. Please contact Silicon Labs sales for more information regarding –AM and -AQ devices for your automotive project.

4. Package Specifications

4.1. QFP-48 Package Specifications

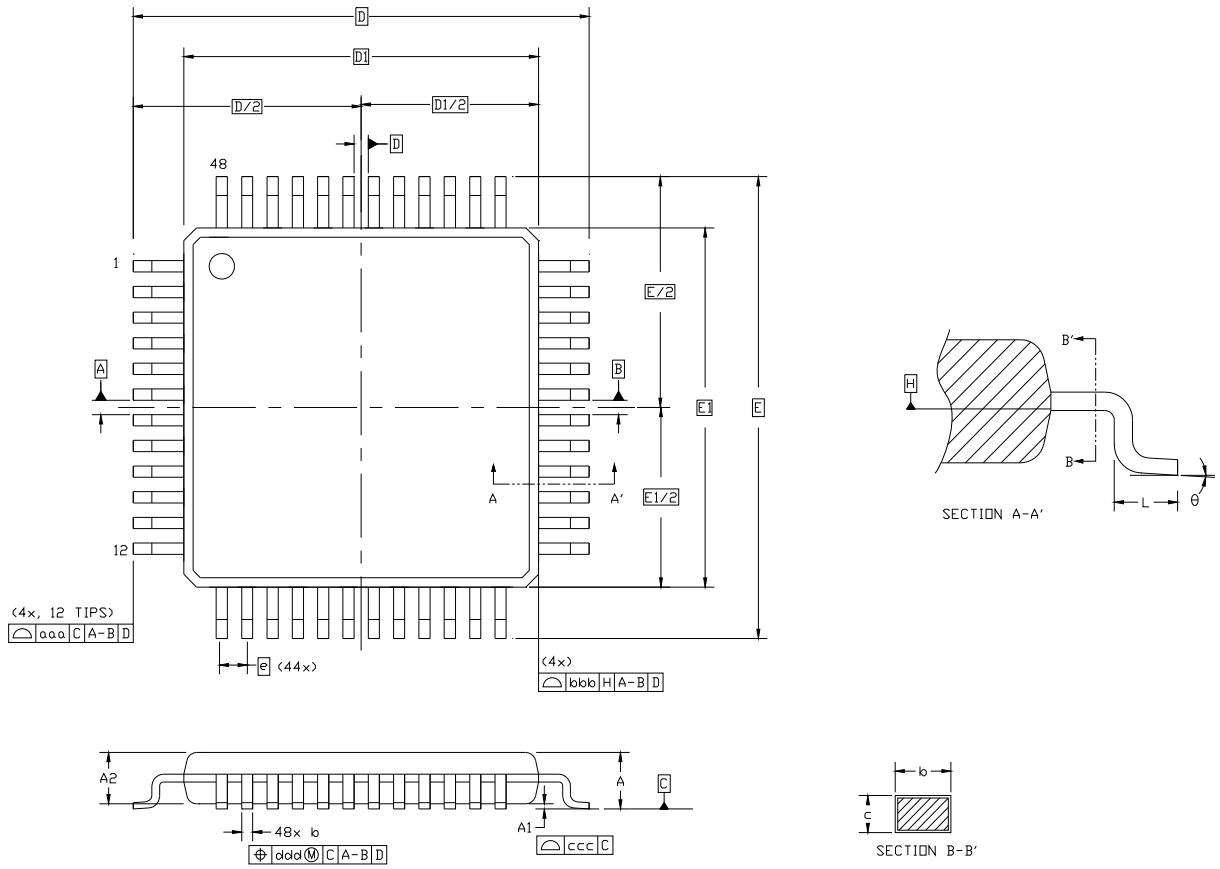

Figure 4.1. QFP-48 Package Drawing

Table 4.1. QFP-48 Package Dimensions

Dimension	Min	Typ	Max
A	—	—	1.20
A1	0.05	—	0.15
A2	0.95	1.00	1.05
b	0.17	0.22	0.27
c	0.09	—	0.20
D	9.00 BSC.		
D1	7.00 BSC.		
e	0.50 BSC.		

Dimension	Min	Typ	Max
E	9.00 BSC.		
E1	7.00 BSC.		
L	0.45	0.60	0.75
aaa	0.20		
bbb	0.20		
ccc	0.08		
ddd	0.08		
θ	0°	3.5°	7°

Notes:

1. All dimensions shown are in millimeters (mm) unless otherwise noted.
2. Dimensioning and Tolerancing per ANSI Y14.5M-1994.
3. This drawing conforms to the JEDEC outline MS-026, variation ABC.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

Table 5.2. Global Electrical Characteristics (Continued)
–40 to +125 °C, 24 MHz system clock unless otherwise specified.

Parameter	Conditions	Min	Typ	Max	Units
Digital Supply Current—CPU Inactive (Idle Mode, not fetching instructions from Flash)					
I_{DD}[4]	V_{DD} = 2.1 V, F = 200 kHz	—	130	—	µA
	V_{DD} = 2.1 V, F = 1.5 MHz	—	440	—	µA
	V_{DD} = 2.1 V, F = 25 MHz	—	5.8	8.0	mA
	V_{DD} = 2.1 V, F = 50 MHz	—	11	16	mA
I_{DD}[4]	V_{DD} = 2.6 V, F = 200 kHz	—	170	—	µA
	V_{DD} = 2.6 V, F = 1.5 MHz	—	570	—	µA
	V_{DD} = 2.6 V, F = 25 MHz	—	7.3	15	mA
	V_{DD} = 2.6 V, F = 50 MHz	—	15	25	mA
I_{DD} Supply Sensitivity[4]	F = 25 MHz	—	53	—	%/V
	F = 1 MHz	—	60	—	
I_{DD} Frequency Sensitivity [4.6]	V_{DD} = 2.1V, F ≤ 12.5 MHz, T = 25 °C	—	0.28	—	mA/MHz
	V_{DD} = 2.1V, F > 12.5 MHz, T = 25 °C	—	0.28	—	
	V_{DD} = 2.6V, F ≤ 12.5 MHz, T = 25 °C	—	0.35	—	
	V_{DD} = 2.6V, F > 12.5 MHz, T = 25 °C	—	0.35	—	
Digital Supply Current[4] (Stop or Suspend Mode)	Oscillator not running, V_{DD} Monitor Disabled				µA
	Temp = 25 °C	—	230	—	
	Temp = 60 °C	—	230	—	
	Temp= 125 °C	—	330	—	

Notes:
1. Given in Table 5.4 on page 48.
2. V_{IO} should not be lower than the V_{DD} voltage.
3. SYSCLK must be at least 32 kHz to enable debugging.
4. Based on device characterization data; Not production tested. Does not include oscillator supply current.
5. IDD can be estimated for frequencies ≤ 15 MHz by simply multiplying the frequency of interest by the frequency sensitivity number for that range. When using these numbers to estimate I_{DD} for >15 MHz, the estimate should be the current at 50 MHz minus the difference in current indicated by the frequency sensitivity number. For example: V_{DD} = 2.6 V; F = 20 MHz, I_{DD} = 21 mA - (50 MHz - 20 MHz) * 0.46 mA/MHz = 7.2 mA.
6. Idle IDD can be estimated for frequencies ≤ 1 MHz by simply multiplying the frequency of interest by the frequency sensitivity number for that range. When using these numbers to estimate Idle I_{DD} for >1 MHz, the estimate should be the current at 50 MHz minus the difference in current indicated by the frequency sensitivity number.
   For example: V_{DD} = 2.6 V; F = 5 MHz, Idle I_{DD} = 19 mA – (50 MHz – 5 MHz) x 0.38 mA/MHz = 1.9 mA.

Post-Tracking Mode is selected when AD0TM is set to 01b. A programmable tracking time based on AD0TK is started immediately following the convert start signal. Conversions are started after the programmed tracking time ends. After a conversion is complete, ADC0 does not track the input. Rather, the sampling capacitor remains disconnected from the input making the input pin high-impedance until the next convert start signal.

Dual-Tracking Mode is selected when AD0TM is set to 11b. A programmable tracking time based on AD0TK is started immediately following the convert start signal. Conversions are started after the programmed tracking time ends. After a conversion is complete, ADC0 tracks continuously until the next conversion is started.

Depending on the output connected to the ADC input, additional tracking time, more than is specified in Table 5.10, may be required after changing MUX settings. See the settling time requirements described in Section “6.2.1. Settling Time Requirements” on page 59.

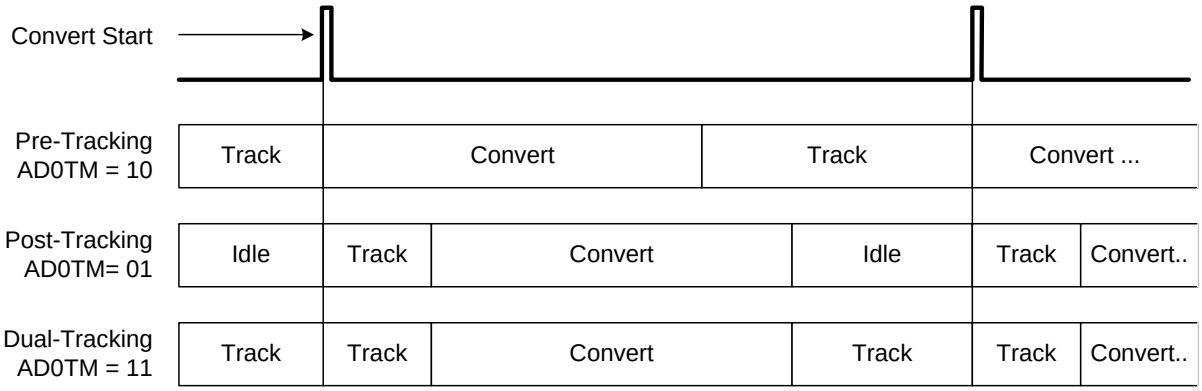

Figure 6.2. ADC0 Tracking Modes

6.1.3. Timing

ADC0 has a maximum conversion speed specified in Table 5.10. ADC0 is clocked from the ADC0 Subsystem Clock (FCLK). The source of FCLK is selected based on the BURSTEN bit. When BURSTEN is logic 0, FCLK is derived from the current system clock. When BURSTEN is logic 1, FCLK is derived from the Burst Mode Oscillator, an independent clock source with a maximum frequency of 25 MHz.

When ADC0 is performing a conversion, it requires a clock source that is typically slower than FCLK. The ADC0 SAR conversion clock (SAR clock) is a divided version of FCLK. The divide ratio can be configured using the AD0SC bits in the ADC0CF register. The maximum SAR clock frequency is listed in Table 5.10.

ADC0 can be in one of three states at any given time: tracking, converting, or idle. Tracking time depends on the tracking mode selected. For Pre-Tracking Mode, tracking is managed by software and ADC0 starts conversions immediately following the convert start signal. For Post-Tracking and Dual-Tracking Modes, the tracking time after the convert start signal is equal to the value determined by the AD0TK bits plus 2 FCLK cycles. Tracking is immediately followed by a conversion. The ADC0 conversion time is always 13 SAR clock cycles plus an additional 2 FCLK cycles to start and complete a conversion. Figure 6.4 shows timing diagrams for a conversion in Pre-Tracking Mode and tracking plus conversion in Post-Tracking or Dual-Tracking Mode. In this example, repeat count is set to one.

6.2. Output Code Formatting

The registers ADC0H and ADC0L contain the high and low bytes of the output conversion code. When the repeat count is set to 1, conversion codes are represented in 12-bit unsigned integer format and the output conversion code is updated after each conversion. Inputs are measured from 0 to V_{REF} x 4095/4096. Data can be right-justified or left-justified, depending on the setting of the AD0LJST bit (ADC0CN.2). Unused bits in the ADC0H and ADC0L registers are set to 0. Example codes are shown below for both right-justified and left-justified data.

Input Voltage	Right-Justified ADC0H:ADC0L (AD0LJST = 0)	Left-Justified ADC0H:ADC0L (AD0LJST = 1)
VREF x 4095/4096	0x0FFF	0xFFF0
VREF x 2048/4096	0x0800	0x8000
VREF x 2047/4096	0x07FF	0x7FF0
0	0x0000	0x0000

When the ADC0 Repeat Count is greater than 1, the output conversion code represents the accumulated result of the conversions performed and is updated after the last conversion in the series is finished. Sets of 4, 8, or 16 consecutive samples can be accumulated and represented in unsigned integer format. The repeat count can be selected using the AD0RPT bits in the ADC0CF register. The value must be right-justified (AD0LJST = 0), and unused bits in the ADC0H and ADC0L registers are set to 0. The following example shows right-justified codes for repeat counts greater than 1. Notice that accumulating 2^n samples is equivalent to left-shifting by *n* bit positions when all samples returned from the ADC have the same value.

Input Voltage	Repeat Count = 4	Repeat Count = 8	Repeat Count = 16
V_{REF} x 4095/4096	0x3FFC	0x7FF8	0xFFF0
V_{REF} x 2048/4096	0x2000	0x4000	0x8000
V_{REF} x 2047/4096	0x1FFC	0x3FF8	0x7FF0
0	0x0000	0x0000	0x0000

6.2.1. Settling Time Requirements

A minimum tracking time is required before an accurate conversion is performed. This tracking time is determined by any series impedance, including the AMUX0 resistance, the ADC0 sampling capacitance, and the accuracy required for the conversion.

Figure 6.5 shows the equivalent ADC0 input circuit. The required ADC0 settling time for a given settling accuracy (SA) may be approximated by Equation 6.1. When measuring the Temperature Sensor output, use the tracking time specified in Table 5.11 on page 52. When measuring V_{DD} with respect to GND, R_{TOTAL} reduces to R_{MUX}. See Table 5.10 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 6.1. 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.
n is the ADC resolution in bits (10).

SFR Definition 6.4. ADC0CF: ADC0 Configuration

Bit	7	6	5	4	3	2	1	0
Name	AD0SC[4:0]					AD0RPT[1:0]		GAINEN
Type	R/W					R/W	R/W	R/W
Reset	1	1	1	1	1	0	0	0

SFR Address = 0xBC; SFR Page = 0x00

Bit	Name	Function
7:3	AD0SC[4:0]	**ADC0 SAR Conversion Clock Period Bits.** SAR Conversion clock is derived from system clock by the following equation, where *AD0SC* refers to the 5-bit value held in bits AD0SC4–0. SAR Conversion clock requirements are given in the ADC specification table BURSTEN = 0: FCLK is the current system clock BURSTEN = 1: FLCLK is a maximum of 30 Mhz, independent of the current system clock.. $AD0SC = \frac{FCLK}{CLK_{SAR}} - 1$ **Note:** Round up the result of the calculation for AD0SC
2:1	A0RPT[1:0]	**ADC0 Repeat Count.** Controls the number of conversions taken and accumulated between ADC0 End of Conversion (ADCINT) and ADC0 Window Comparator (ADCWINT) interrupts. A convert start is required for each conversion unless Burst Mode is enabled. In Burst Mode, a single convert start can initiate multiple self-timed conversions. Results in both modes are accumulated in the ADC0H:ADC0L register. **When AD0RPT1–0 are set to a value other than '00', the AD0LJST bit in the ADC0CN register must be set to '0' (right justified).** 00: 1 conversion is performed. 01: 4 conversions are performed and accumulated. 10: 8 conversions are performed and accumulated. 11: 16 conversions are performed and accumulated.
0	GAINEN	**Gain Enable Bit.** Controls the gain programming. Refer to Section “6.3. Selectable Gain” on page 60 for information about using this bit.

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

Mnemonic	Description	Bytes	Clock Cycles
XRL A, #data	Exclusive-OR immediate to A	2	2
XRL direct, A	Exclusive-OR A to direct byte	2	2
XRL direct, #data	Exclusive-OR immediate to direct byte	3	3
CLR A	Clear A	1	1
CPL A	Complement A	1	2
RL A	Rotate A left	1	1
RLC A	Rotate A left through Carry	1	1
RR A	Rotate A right	1	1
RRC A	Rotate A right through Carry	1	1
SWAP A	Swap nibbles of A	1	1
Data Transfer			
MOV A, Rn	Move Register to A	1	1
MOV A, direct	Move direct byte to A	2	2
MOV A, @Ri	Move indirect RAM to A	1	2
MOV A, #data	Move immediate to A	2	2
MOV Rn, A	Move A to Register	1	1
MOV Rn, direct	Move direct byte to Register	2	2
MOV Rn, #data	Move immediate to Register	2	2
MOV direct, A	Move A to direct byte	2	2
MOV direct, Rn	Move Register to direct byte	2	2
MOV direct, direct	Move direct byte to direct byte	3	3
MOV direct, @Ri	Move indirect RAM to direct byte	2	2
MOV direct, #data	Move immediate to direct byte	3	3
MOV @Ri, A	Move A to indirect RAM	1	2
MOV @Ri, direct	Move direct byte to indirect RAM	2	2
MOV @Ri, #data	Move immediate to indirect RAM	2	2
MOV DPTR, #data16	Load DPTR with 16-bit constant	3	3
MOVC A, @A+DPTR	Move code byte relative DPTR to A	1	4-7*
MOVC A, @A+PC	Move code byte relative PC to A	1	3
MOVX A, @Ri	Move external data (8-bit address) to A	1	3
MOVX @Ri, A	Move A to external data (8-bit address)	1	3
MOVX A, @DPTR	Move external data (16-bit address) to A	1	3
MOVX @DPTR, A	Move A to external data (16-bit address)	1	3
PUSH direct	Push direct byte onto stack	2	2
POP direct	Pop direct byte from stack	2	2
XCH A, Rn	Exchange Register with A	1	1
XCH A, direct	Exchange direct byte with A	2	2
XCH A, @Ri	Exchange indirect RAM with A	1	2
XCHD A, @Ri	Exchange low nibble of indirect RAM with A	1	2
Boolean Manipulation			
CLR C	Clear Carry	1	1
CLR bit	Clear direct bit	2	2

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

11.4. Serial Number Special Function Registers (SFRs)

The C8051F58x/F59x devices include four SFRs, SN0 through SN3, that are pre-programmed during production with a unique, 32-bit serial number. The serial number provides a unique identification number for each device and can be read from the application firmware. If the serial number is not used in the application, these four registers can be used as general purpose SFRs.

SFR Definition 11.7. SNn: Serial Number n

Bit	7	6	5	4	3	2	1	0
Name	SERNUMn[7:0]							
Type	R/W							
Reset	Varies—Unique 32-bit value							

SFR Addresses: SN0 = 0xF9; SN1 = 0xFA; SN2 = 0xFB; SN3 = 0xFC; SFR Page = 0x0F;

Bit	Name	Function
7:0	SERNUMn[7:0]	**Serial Number Bits.** The four serial number registers form a 32-bit serial number, with SN3 as the most significant byte and SN0 as the least significant byte.

SFR Definition 14.2. IP: Interrupt Priority

Bit	7	6	5	4	3	2	1	0
Name		PSPI0	PT2	PS0	PT1	PX1	PT0	PX0
Type	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0

SFR Address = 0xB8; Bit-Addressable; SFR Page = All Pages

Bit	Name	Function
7	Unused	Read = 1b, Write = Don't Care.
6	PSPI0	**Serial Peripheral Interface (SPI0) Interrupt Priority Control.** This bit sets the priority of the SPI0 interrupt. 0: SPI0 interrupt set to low priority level. 1: SPI0 interrupt set to high priority level.
5	PT2	**Timer 2 Interrupt Priority Control.** This bit sets the priority of the Timer 2 interrupt. 0: Timer 2 interrupt set to low priority level. 1: Timer 2 interrupt set to high priority level.
4	PS0	**UART0 Interrupt Priority Control.** This bit sets the priority of the UART0 interrupt. 0: UART0 interrupt set to low priority level. 1: UART0 interrupt set to high priority level.
3	PT1	**Timer 1 Interrupt Priority Control.** This bit sets the priority of the Timer 1 interrupt. 0: Timer 1 interrupt set to low priority level. 1: Timer 1 interrupt set to high priority level.
2	PX1	**External Interrupt 1 Priority Control.** This bit sets the priority of the External Interrupt 1 interrupt. 0: External Interrupt 1 set to low priority level. 1: External Interrupt 1 set to high priority level.
1	PT0	**Timer 0 Interrupt Priority Control.** This bit sets the priority of the Timer 0 interrupt. 0: Timer 0 interrupt set to low priority level. 1: Timer 0 interrupt set to high priority level.
0	PX0	**External Interrupt 0 Priority Control.** This bit sets the priority of the External Interrupt 0 interrupt. 0: External Interrupt 0 set to low priority level. 1: External Interrupt 0 set to high priority level.

SFR Definition 15.4. CCH0CN: Cache Control

Bit	7	6	5	4	3	2	1	0
Name	Reserved	Reserved	CHPFEN	Reserved	Reserved	Reserved	Reserved	CHBLKW
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	1	0	0	0	0	0

SFR Address = 0xE3; SFR Page = 0x0F

Bit	Name	Function
7:6	Reserved	Must Write 00b
5	CHPFEN	**Cache Prefect Enable Bit.** 0: Prefetch engine is disabled. 1: Prefetch engine is enabled.
4:1	Reserved	Must Write 0000b.
0	CHBLKW	**Block Write Enable Bit.** This bit allows block writes to Flash memory from firmware. 0: Each byte of a software Flash write is written individually. 1: Flash bytes are written in groups of two.

SFR Definition 15.5. ONESHOT: Flash Oneshot Period

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

SFR Address = 0xBE; SFR Page = 0x0F

Bit	Name	Function
7:4	Unused	Read = 0000b. Write = don't care.
3:0	PERIOD[3:0]	**Oneshot Period Control Bits.** These bits limit the internal Flash read strobe width as follows. When the Flash read strobe is de-asserted, the Flash memory enters a low-power state for the remainder of the system clock cycle. These bits have no effect when the system clocks is greater than 12.5 MHz and FLRT = 0. $FLASH_{RDMAX} = 5ns + (PERIOD \times 5ns)$

SFR Definition 20.11. P3MASK: Port 3 Mask Register

Bit	7	6	5	4	3	2	1	0
Name	P3MASK[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xAF; SFR Page = 0x00

Bit	Name	Function
7:0	P3MASK[7:0]	**Port 1 Mask Value.** Selects P3 pins to be compared to the corresponding bits in P3MAT. 0: P3.n pin logic value is ignored and cannot cause a Port Mismatch event. 1: P3.n pin logic value is compared to P3MAT.n.
Note: P3.1–P3.6 are only available on the 48-pin and 40-pin packages		

SFR Definition 20.12. P3MAT: Port 3 Match Register

Bit	7	6	5	4	3	2	1	0
Name	P3MAT[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xAE; SFR Page = 0x00

Bit	Name	Function
7:0	P3MAT[7:0]	**Port 3 Match Value.** Match comparison value used on Port 3 for bits in P3MAT which are set to 1. 0: P3.n pin logic value is compared with logic LOW. 1: P3.n pin logic value is compared with logic HIGH.
Note: P3.1–P3.6 are only available on the 48-pin and 40-pin packages		

SFR Definition 21.3. LIN0CF: LIN0 Control Mode Register

Bit	7	6	5	4	3	2	1	0
Name	LINEN	MODE	ABAUD					
Type	R/W	R/W	R/W	R	R	R	R	R
Reset	0	1	1	0	0	0	0	0

SFR Address = 0xC9; SFR Page = 0x0F

Bit	Name	Function
7	LINEN	**LIN Interface Enable Bit.** 0: LIN0 is disabled. 1: LIN0 is enabled.
6	MODE	**LIN Mode Selection Bit.** 0: LIN0 operates in slave mode. 1: LIN0 operates in master mode.
5	ABAUD	**LIN Mode Automatic Baud Rate Selection.** This bit only has an effect when the MODE bit is configured for slave mode. 0: Manual baud rate selection is enabled. 1: Automatic baud rate selection is enabled.
4:0	Unused	Read = 00000b; Write = Don't Care

22. Controller Area Network (CAN0)

Important Documentation Note: The Bosch CAN Controller is integrated in the C8051F580/2/4/6/8-F590 devices. This section of the data sheet gives a description of the CAN controller as an overview and offers a description of how the Silicon Labs CIP-51 MCU interfaces with the on-chip Bosch CAN controller. In order to use the CAN controller, refer to Bosch's C_CAN User's Manual as an accompanying manual to the Silicon Labs' data sheet.

The C8051F580/2/4/6/8-F590 devices feature a Control Area Network (CAN) controller that enables serial communication using the CAN protocol. Silicon Labs CAN facilitates communication on a CAN network in accordance with the Bosch specification 2.0A (basic CAN) and 2.0B (full CAN). The CAN controller consists of a CAN Core, Message RAM (separate from the CIP-51 RAM), a message handler state machine, and control registers. Silicon Labs CAN is a protocol controller and does not provide physical layer drivers (i.e., transceivers). Figure 22.1 shows an example typical configuration on a CAN bus.

Silicon Labs CAN operates at bit rates of up to 1 Mbit/second, though this can be limited by the physical layer chosen to transmit data on the CAN bus. The CAN processor has 32 Message Objects that can be configured to transmit or receive data. Incoming data, message objects and their identifier masks are stored in the CAN message RAM. All protocol functions for transmission of data and acceptance filtering is performed by the CAN controller and not by the CIP-51 MCU. In this way, minimal CPU bandwidth is needed to use CAN communication. The CIP-51 configures the CAN controller, accesses received data, and passes data for transmission via Special Function Registers (SFRs) in the CIP-51.

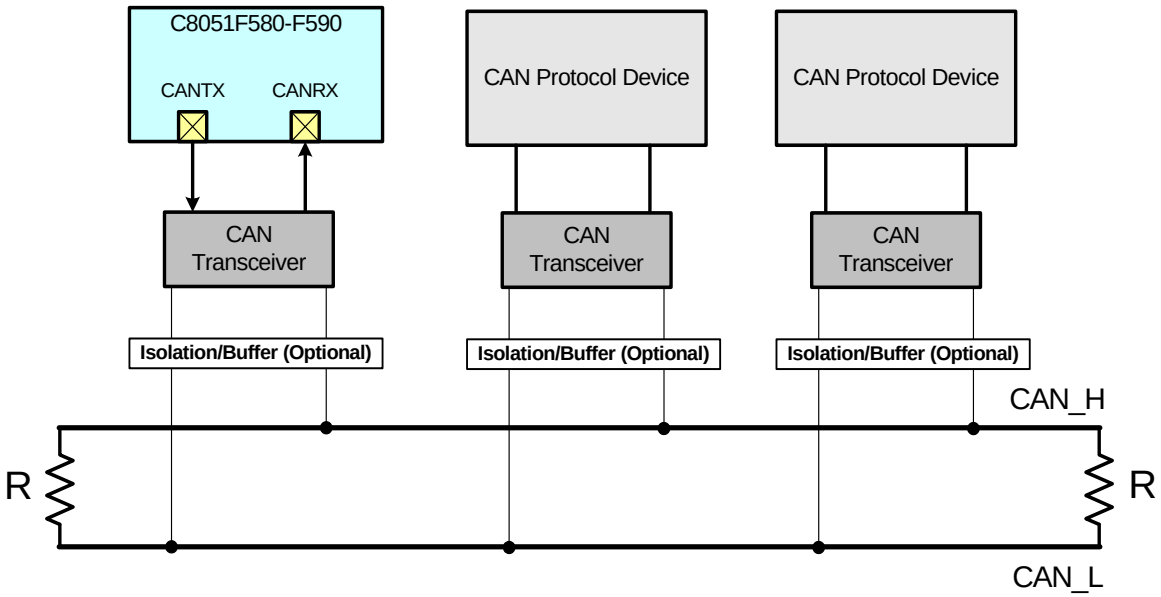

Figure 22.1. Typical CAN Bus Configuration

22.1. Bosch CAN Controller Operation

The CAN Controller featured in the C8051F580/2/4/6/8-F590 devices is a full implementation of Bosch's full CAN module and fully complies with CAN specification 2.0B. A block diagram of the CAN controller is shown in Figure 22.2. The CAN Core provides shifting (CANTX and CANRX), serial/parallel conversion of messages, and other protocol related tasks such as transmission of data and acceptance filtering. The message RAM stores 32 message objects which can be received or transmitted on a CAN network. The CAN registers and message handler provide an interface for data transfer and notification between the CAN controller and the CIP-51.

The function and use of the CAN Controller is detailed in the Bosch CAN User's Guide. The User's Guide should be used as a reference to configure and use the CAN controller. This data sheet describes how to access the CAN controller.

All of the CAN controller registers are located on SFR Page 0x0C. Before accessing any of the CAN registers, the SFRPAGE register must be set to 0x0C.

The CAN Controller is typically initialized using the following steps:

1. Set the SFRPAGE register to the CAN registers page (page 0x0C).
2. Set the INIT and the CCE bits to 1 in CAN0CN. See the CAN User's Guide for bit definitions.
3. Set timing parameters in the Bit Timing Register and the BRP Extension Register.
4. Initialize each message object or set its MsgVal bit to NOT VALID.
5. Reset the INIT bit to 0.

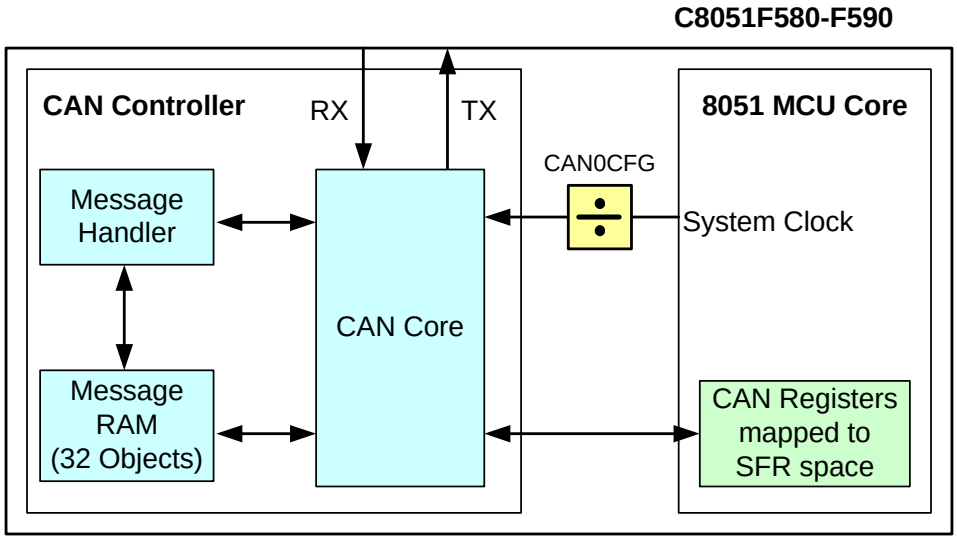

Figure 22.2. CAN Controller Diagram

22.1.1. CAN Controller Timing

The CAN controller's clock (f_{sys}) is derived from the CIP-51 system clock (SYSCLK). The internal oscillator is accurate to within 0.5% of 24 MHz across the entire temperature range and for VDD voltages greater than or equal to the minimum output of the on-chip voltage regulator, so an external oscillator is not required for CAN communication for most systems. Refer to Section "4.10.4 Oscillator Tolerance Range" in the Bosch CAN User's Guide for further information regarding this topic.

23. SMBus

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 interface 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 interface 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. A block diagram of the SMBus peripheral and the associated SFRs is shown in Figure 23.1.

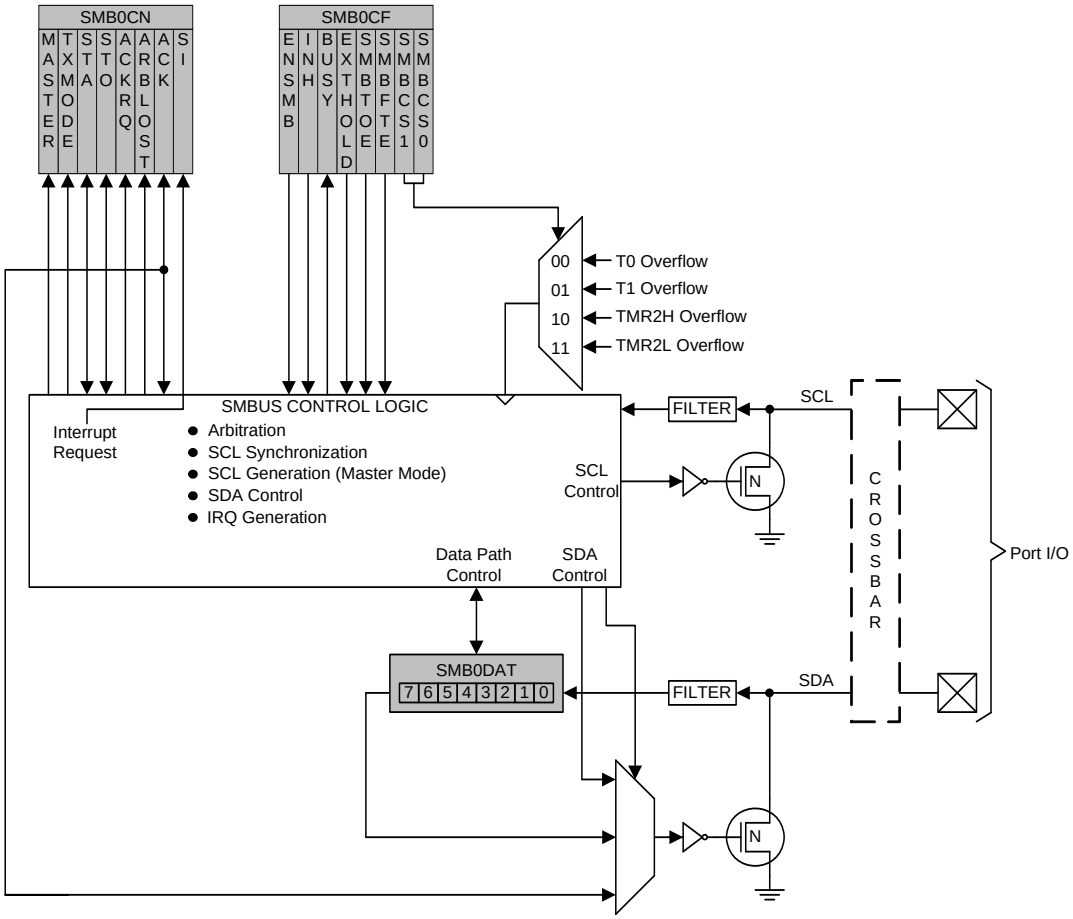

Figure 23.1. SMBus Block Diagram

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 “27. Timers” on page 285.

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

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.

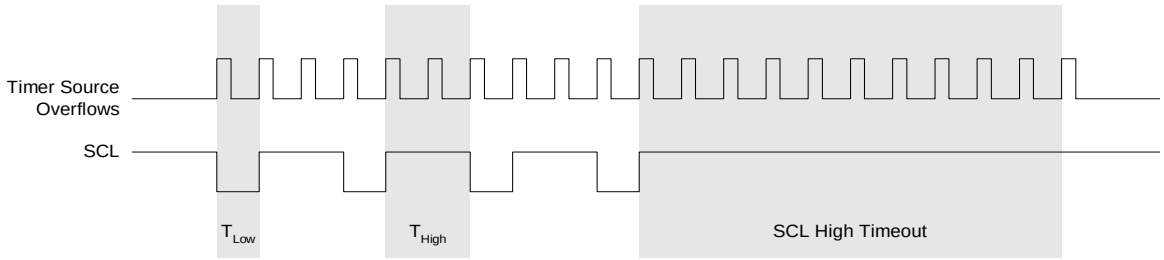

Figure 23.4. Typical SMBus SCL Generation

Data transmission begins when an instruction writes a data byte to the SBUF1 register. The TI1 Transmit Interrupt Flag (SCON1.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN1 Receive Enable bit (SCON1.4) is set to 1. After the stop bit is received, the data byte will be loaded into the SBUF1 receive register if the following conditions are met: (1) RI1 must be logic 0, and (2) if MCE1 is logic 1, the 9th bit must be logic 1 (when MCE1 is logic 0, the state of the ninth data bit is unimportant). If these conditions are met, the eight bits of data are stored in SBUF1, the ninth bit is stored in RB81, and the RI1 flag is set to 1. If the above conditions are not met, SBUF1 and RB81 will not be loaded and the RI1 flag will not be set to 1. A UART1 interrupt will occur if enabled when either TI1 or RI1 is set to '1'.

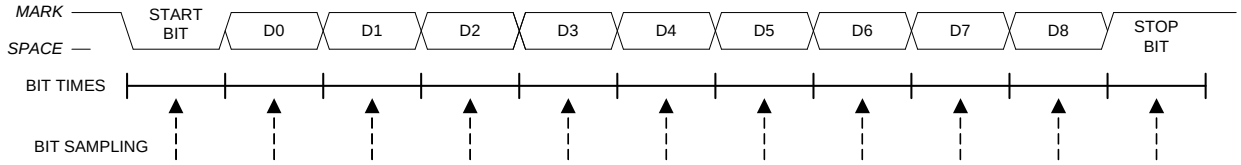

Figure 25.5. 9-Bit UART Timing Diagram

25.3. Multiprocessor Communications

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

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

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

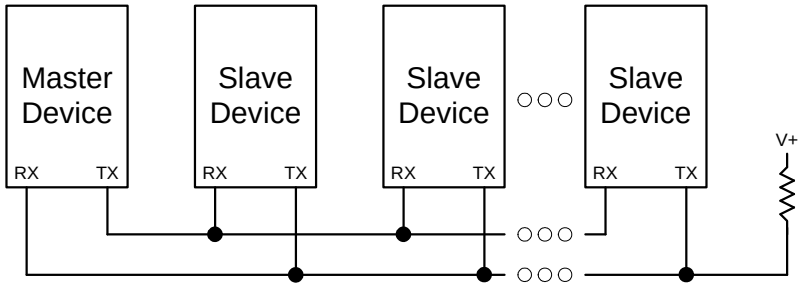

Figure 25.6. UART Multi-Processor Mode Interconnect Diagram

SFR Definition 27.9. TMR2RLL: Timer 2 Reload Register Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLL[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCA; SFR Page = 0x00

Bit	Name	Function
7:0	TMR2RLL[7:0]	**Timer 2 Reload Register Low Byte.** TMR2RLL holds the low byte of the reload value for Timer 2.

SFR Definition 27.10. TMR2RLH: Timer 2 Reload Register High Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2RLH[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCB; SFR Page = 0x00

Bit	Name	Function
7:0	TMR2RLH[7:0]	**Timer 2 Reload Register High Byte.** TMR2RLH holds the high byte of the reload value for Timer 2.

SFR Definition 28.5. PCA0L: PCA0 Counter/Timer Low Byte

Bit	7	6	5	4	3	2	1	0
Name	PCA0[7:0]							
Type	R/W	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 = 0xF9; SFR Page = 0x00

Bit	Name	Function
7:0	PCA0[7:0]	**PCA0 Counter/Timer Low Byte.** The PCA0L register holds the low byte (LSB) of the 16-bit PCA0 Counter/Timer.
Note: When the WDTE bit is set to 1, the PCA0L register cannot be modified by software. To change the contents of the PCA0L register, the Watchdog Timer must first be disabled.		

SFR Definition 28.6. PCA0H: PCA0 Counter/Timer High Byte

Bit	7	6	5	4	3	2	1	0
Name	PCA0[15:8]							
Type	R/W	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 = 0xFA; SFR Page = 0x00

Bit	Name	Function
7:0	PCA0[15:8]	**PCA0 Counter/Timer High Byte.** The PCA0H register holds the high byte (MSB) of the 16-bit PCA0 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 (see Section 28.1).
Note: When the WDTE bit is set to 1, the PCA0H register cannot be modified by software. To change the contents of the PCA0H register, the Watchdog Timer must first be disabled.		

29.1. PCA1 Counter/Timer

The 16-bit PCA1 counter/timer consists of two 8-bit SFRs: PCA1L and PCA1H. PCA1H is the high byte (MSB) of the 16-bit counter/timer and PCA1L is the low byte (LSB). Reading PCA1L automatically latches the value of PCA1H into a “snapshot” register; the following PCA1H read accesses this “snapshot” register. **Reading the PCA1L Register first guarantees an accurate reading of the entire 16-bit PCA1 counter.** Reading PCA1H or PCA1L does not disturb the counter operation. The CPS12–CPS10 bits in the PCA1MD register select the timebase for the counter/timer as shown in Table 29.1.

When the counter/timer overflows from 0xFFFF to 0x0000, the Counter Overflow Flag (CF1) in PCA1MD is set to logic 1 and an interrupt request is generated if CF1 interrupts are enabled. Setting the ECF1 bit in PCA1MD to logic 1 enables the CF1 flag to generate an interrupt request. The CF1 bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Clearing the CIDL1 bit in the PCA1MD register allows the PCA1 to continue normal operation while the CPU is in Idle mode.

Table 29.1. PCA1 Timebase Input Options

CPS12	CPS11	CPS10	Timebase
0	0	0	System clock divided by 12.
0	0	1	System clock divided by 4.
0	1	0	Timer 0 overflow.
0	1	1	High-to-low transitions on ECI1 (max rate = system clock divided by 4).
1	0	0	System clock.
1	0	1	External oscillator source divided by 8.*
1	1	0	Timer 4 Overflow.
1	1	1	Timer 5 Overflow.
***Note:** External oscillator source divided by 8 is synchronized with the system clock.			

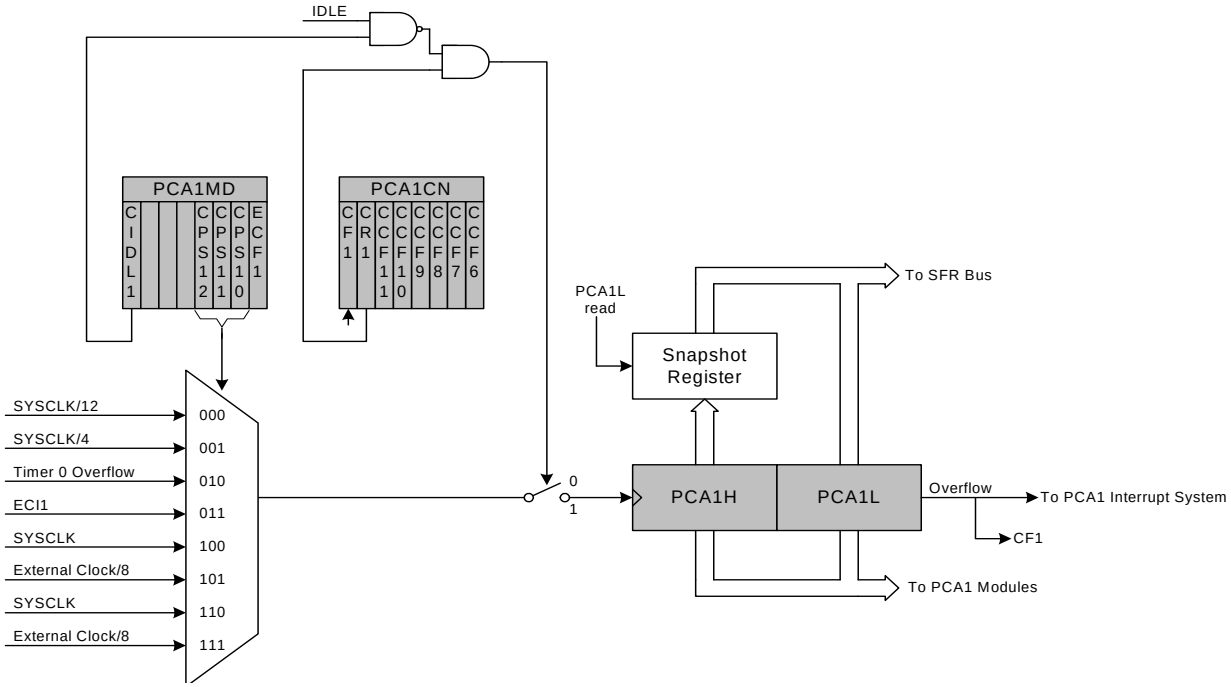

Figure 29.2. PCA1 Counter/Timer Block Diagram