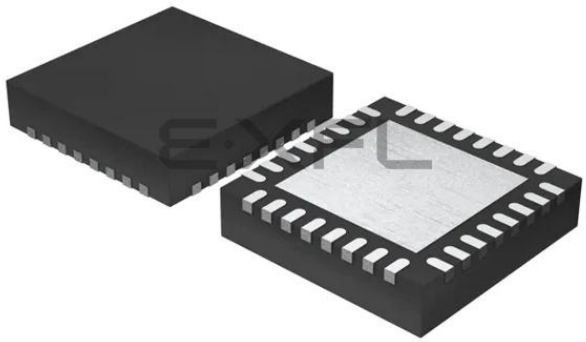

Welcome to E-XFL.COM

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	128KB (128K 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/c8051f582-imr

List of Tables

3. Pin Definitions

Table 3.1. Pin Definitions for the C8051F58x/F59x

Name	Pin F580/1/4/5 (48-pin)	Pin F588/9-F590/1 (40-pin)	Pin F582/3/6/7 (32-pin)	Type	Description
VDD	4	4	4		Digital Supply Voltage. Must be connected.
GND	6	6	6		Digital Ground. Must be connected.
VDDA	5	5	5		Analog Supply Voltage. Must be connected.
GNDA	7	7	7		Analog Ground. Must be connected.
VREGIN	3	3	3		Voltage Regulator Input
VIO	2	2	2		Port I/O Supply Voltage. Must be connected.
$\overline{\text{RST}}$/	12	10	10	D I/O	Device Reset. Open-drain output of internal POR or V_{DD} Monitor. An external source can initiate a system reset by driving this pin low.
C2CK				D I/O	Clock signal for the C2 Debug Interface.
C2D	11	—	—	D I/O	Bi-directional data signal for the C2 Debug Interface.
P4.0/	—	9	—	D I/O or A In	Port 4.0. See SFR Definition 20.29 for a description.
C2D				D I/O	Bi-directional data signal for the C2 Debug Interface.
P3.0/	—		9	D I/O or A In	Port 3.0. See SFR Definition 20.25 for a description.
C2D				D I/O	Bi-directional data signal for the C2 Debug Interface.
P0.0	8	8	8	D I/O or A In	Port 0.0. See SFR Definition 20.13 for a description.
P0.1	1	1	1	D I/O or A In	Port 0.1
P0.2	48	40	32	D I/O or A In	Port 0.2
P0.3	47	39	31	D I/O or A In	Port 0.3
P0.4	46	38	30	D I/O or A In	Port 0.4
P0.5	45	37	29	D I/O or A In	Port 0.5

Table 3.1. Pin Definitions for the C8051F58x/F59x (Continued)

Name	Pin F580/1/4/5 (48-pin)	Pin F588/9-F590/1 (40-pin)	Pin F582/3/6/7 (32-pin)	Type	Description
P3.7	19	11	—	D I/O or A In	Port 3.7.
P4.0	18	—	—	D I/O	Port 4.0. See SFR Definition 20.29 for a description.
P4.1	17	—	—	D I/O	Port 4.1.
P4.2	16	—	—	D I/O	Port 4.2.
P4.3	15	—	—	D I/O	Port 4.3.
P4.4	14	—	—	D I/O	Port 4.4.
P4.5	13	—	—	D I/O	Port 4.5.
P4.6	10	—	—	D I/O	Port 4.6.
P4.7	9	—	—	D I/O	Port 4.7.

SFR Definition 12.1. PSBANK: Program Space Bank Select

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

SFR Address = 0xF5; SFR Page = All Pages

Bit	Name	Function
7:6	Reserved	Read = 00b, Must Write = 00b.
5:4	COBANK[1:0]	**Constant Operations Bank Select.** These bits select which Flash bank is targeted during constant operations (MOVC and Flash MOVX) involving address 0x8000 to 0xFFFF. 00: Constant Operations Target Bank 0 (note that Bank 0 is also mapped between 0x0000 to 0x7FFF). 01: Constant operations target Bank 1. 10: Constant operations target Bank 2. 11: Constant operations target Bank 3.
3:2	Reserved	Read = 00b, Must Write = 00b.
1:0	IFBANK[1:0]	**Instruction Fetch Operations Bank Select.** These bits select which Flash bank is used for instruction fetches involving address 0x8000 to 0xFFFF. These bits can only be changed from code in Bank 0. 00: Instructions fetch from Bank 0 (note that Bank 0 is also mapped between 0x0000 to 0x7FFF). 01: Instructions fetch from Bank 1. 10: Instructions fetch from Bank 2. 11: Instructions fetch from Bank 3.
Note: COBANK[1:0] and IFBANK[1:0] should not be set to select Bank 3 (11b) on the C8051F584/5/6/7-F590/1 devices.		

12.1.1. MOVX Instruction and Program Memory

The MOVX instruction in an 8051 device is typically used to access external data memory. On the C8051F58x/F59x devices, the MOVX instruction is normally used to read and write on-chip XRAM, but can be re-configured to write and erase on-chip Flash memory space. MOVC instructions are always used to read Flash memory, while MOVX write instructions are used to erase and write Flash. This Flash access feature provides a mechanism for the C8051F58x/F59x to update program code and use the program memory space for non-volatile data storage. Refer to Section “15. Flash Memory” on page 138 for further details.

12.2. Data Memory

The C8051F58x/F59x devices include 8448 bytes of RAM data memory. 256 bytes of this memory is mapped into the internal RAM space of the 8051. The other 8192 bytes of this memory is on-chip “external” memory. The data memory map is shown in Figure 12.1 for reference.

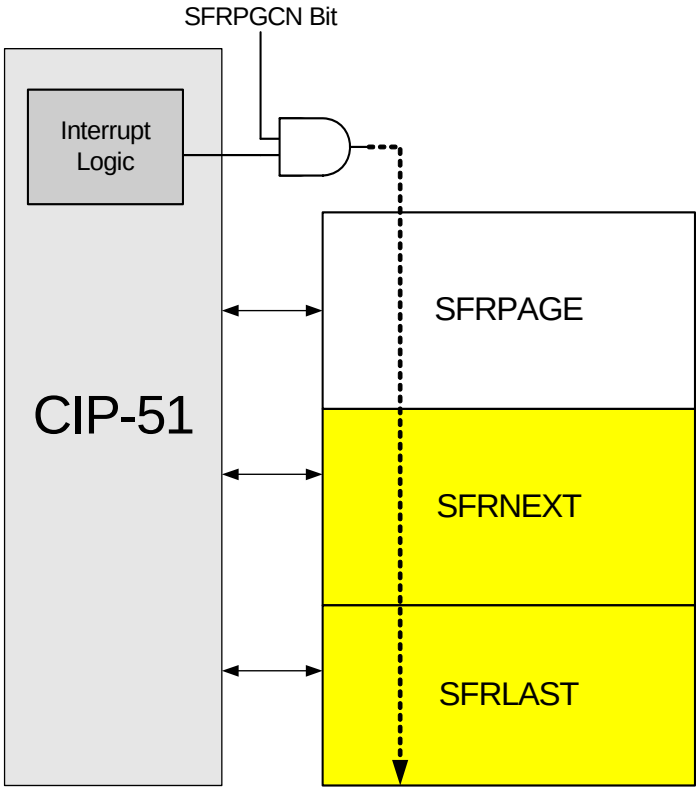

Figure 13.1. SFR Page Stack

Automatic hardware switching of the SFR Page on interrupts may be enabled or disabled as desired using the SFR Automatic Page Control Enable Bit located in the SFR Page Control Register (SFR0CN). This function defaults to “enabled” upon reset. In this way, the autoswitching function will be enabled unless disabled in software.

A summary of the SFR locations (address and SFR page) are provided in Table 13.3 in the form of an SFR memory map. Each memory location in the map has an SFR page row, denoting the page in which that SFR resides. Certain SFRs are accessible from ALL SFR pages, and are denoted by the “(ALL PAGES)” designation. For example, the Port I/O registers P0, P1, P2, and P3 all have the “(ALL PAGES)” designation, indicating these SFRs are accessible from all SFR pages regardless of the SFRPAGE register value.

13.3. SFR Page Stack Example

The following is an example that shows the operation of the SFR Page Stack during interrupts. In this example, the SFR Control register is left in the default enabled state (i.e., SFRPGEN = 1), and the CIP-51 is executing in-line code that is writing values to SPI Data Register (SFR “SPI0DAT”, located at address 0xA3 on SFR Page 0x00). The device is also using the CAN peripheral (CAN0) and the Programmable Counter Array (PCA0) peripheral to generate a PWM output. The PCA is timing a critical control function in its interrupt service round so its associated ISR that is set to low priority. At this point, the SFR page is set to access the SPI0DAT SFR (SFRPAGE = 0x00). See Figure 13.2.

SFR Definition 13.4. SFRLAST: SFR Last

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

SFR Address = 0xA7; SFR Page = All Pages

Bit	Name	Function
7:0	SFRLAST[7:0]	**SFR Page Stack Bits.** This is the value that will go to the SFRNEXT register upon a return from interrupt. Write: Sets the SFR Page in the last entry of the SFR Stack. This will cause the SFRNEXT SFR to have this SFR page value upon a return from interrupt. Read: Returns the value of the SFR page contained in the last entry of the SFR stack. SFR page context is retained upon interrupts/return from interrupts in a 3 byte SFR Page Stack: SFRPAGE is the first entry, SFRNEXT is the second, and SFRLAST is the third entry. The SFR stack bytes may be used alter the context in the SFR Page Stack, and will not cause the stack to “push” or “pop”. Only interrupts and return from interrupts cause pushes and pops of the SFR Page Stack.

15.1.2. Flash Lock and Key Functions

Flash writes and erases by user software are protected with a lock and key function. The Flash Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before Flash operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, Flash writes and erases will be disabled until the next system reset. Flash writes and erases will also be disabled if a Flash write or erase is attempted before the key codes have been written properly. The Flash lock resets after each write or erase; the key codes must be written again before a following Flash operation can be performed. The FLKEY register is detailed in SFR Definition 15.2.

15.1.3. Flash Erase Procedure

The Flash memory can be programmed by software using the MOVX write instruction with the address and data byte to be programmed provided as normal operands. Before writing to Flash memory using MOVX, Flash write operations must be enabled by doing the following: (1) setting the PSWE Program Store Write Enable bit (PSCTL.0) to logic 1 (this directs the MOVX writes to target Flash memory); and (2) Writing the Flash key codes in sequence to the Flash Lock register (FLKEY). The PSWE bit remains set until cleared by software.

A write to Flash memory can clear bits to logic 0 but cannot set them; only an erase operation can set bits to logic 1 in Flash. **A byte location to be programmed should be erased before a new value is written.** The Flash memory is organized in 512-byte pages. The erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire 512-byte page, perform the following steps:

1. Disable interrupts (recommended).
2. If erasing a page in Banks 1, 2, or 3, set the COBANK[1:0] bits (register PSBANK) for the appropriate bank.
3. Set the FLEWT bit (register FLSCL).
4. Set the PSEE bit (register PSCTL).
5. Set the PSWE bit (register PSCTL).
6. Write the first key code to FLKEY: 0xA5.
7. Write the second key code to FLKEY: 0xF1.
8. Using the MOVX instruction, write a data byte to any location within the 512-byte page to be erased.
9. Clear the PSWE and PSEE bits.

15.1.4. Flash Write Procedure

Flash bytes are programmed by software with the following sequence:

1. Disable interrupts (recommended).
2. If writing to an address in Banks 1, 2, or 3, set the COBANK[1:0] (register PSBANK) for the appropriate bank.
3. Erase the 512-byte Flash page containing the target location, as described in Section 15.1.3.
4. Set the FLEWT bit (register FLSCL).
5. Set the PSWE bit (register PSCTL).
6. Clear the PSEE bit (register PSCTL).
7. Write the first key code to FLKEY: 0xA5.
8. Write the second key code to FLKEY: 0xF1.
9. Using the MOVX instruction, write a single data byte to the desired location within the 512-byte sector.
10. Clear the PSWE bit.

Table 18.2. EMIF Pinout (C8051F588/9-F590/1)

Multiplexed Mode	
Signal Name	**Port Pin**
$\overline{RD}$	P1.6
$\overline{WR}$	P1.7
ALE	P1.5
D0/A0	P3.0
D1/A1	P3.1
D2/A2	P3.2
D3/A3	P3.3
D4/A4	P3.4
D5/A5	P3.5
D6/A6	P3.6
D7/A7	P3.7
A8	P2.0
A9	P2.1
A10	P2.2
A11	P2.3
A12	P2.4
A13	P2.5
A14	P2.6
A15	P2.7

19.4.1. External Crystal Example

If a crystal or ceramic resonator is used as an external oscillator source for the MCU, the circuit should be configured as shown in Figure 19.1, Option 1. The External Oscillator Frequency Control value (XFCN) should be chosen from the Crystal column of the table in SFR Definition 19.6 (OSCXCN register). For example, an 11.0592 MHz crystal requires an XFCN setting of 111b and a 32.768 kHz Watch Crystal requires an XFCN setting of 001b. After an external 32.768 kHz oscillator is stabilized, the XFCN setting can be switched to 000 to save power. It is recommended to enable the missing clock detector before switching the system clock to any external oscillator source.

Note: Small surface mount crystals can have maximum drive level specifications that are exceeded by the above XFCN recommendations. In these cases, a software-controlled startup sequence may be used to reliably start the crystal using a higher XFCN setting, and then lowering the XFCN setting once the oscillator has started to reduce the drive level and prevent damage or premature aging of the crystal. In all cases, the drive level should be measured to ensure that the crystal is being driven within its operational guidelines as part of robust oscillator system design. Contact technical support for additional details and recommendations if using surface mount crystals with these devices.

When the crystal oscillator is first enabled, the oscillator amplitude detection circuit requires a settling time to achieve proper bias. Introducing a delay of 1 ms between enabling the oscillator and checking the XTLVLD bit will prevent a premature switch to the external oscillator as the system clock. Switching to the external oscillator before the crystal oscillator has stabilized can result in unpredictable behavior. The recommended procedure is:

1. Force XTAL1 and XTAL2 to a high state. This involves enabling the Crossbar and writing 1 to the port pins associated with XTAL1 and XTAL2.
2. Configure XTAL1 and XTAL2 as analog inputs using.
3. Enable the external oscillator.
4. Wait at least 1 ms.
5. Poll for XTLVLD => 1.
6. Enable the Missing Clock Detector.
7. Switch the system clock to the external oscillator.

Important Note on External Crystals: Crystal oscillator circuits are quite sensitive to PCB layout. The crystal should be placed as close as possible to the XTAL pins on the device. The traces should be as short as possible and shielded with ground plane from any other traces which could introduce noise or interference.

The capacitors shown in the external crystal configuration provide the load capacitance required by the crystal for correct oscillation. These capacitors are "in series" as seen by the crystal and "in parallel" with the stray capacitance of the XTAL1 and XTAL2 pins.

Note: The desired load capacitance depends upon the crystal and the manufacturer. Refer to the crystal data sheet when completing these calculations.

For example, a tuning-fork crystal of 32.768 kHz with a recommended load capacitance of 12.5 pF should use the configuration shown in Figure 19.1, Option 1. The total value of the capacitors and the stray capacitance of the XTAL pins should equal 25 pF. With a stray capacitance of 3 pF per pin, the 22 pF capacitors yield an equivalent capacitance of 12.5 pF across the crystal, as shown in Figure 19.3.

$$f = (KF)/(R \times V_{DD})$$

Equation 19.2. C Mode Oscillator Frequency

For example: Assume V_{DD} = 2.1 V and f = 75 kHz:

f = KF / (C x VDD)

0.075 MHz = KF / (C x 2.1)

Since the frequency of roughly 75 kHz is desired, select the K Factor from the table in SFR Definition 19.6 (OSCXCN) as KF = 7.7:

0.075 MHz = 7.7 / (C x 2.1)

C x 2.1 = 7.7 / 0.075 MHz

C = 102.6 / 2.0 pF = 51.3 pF

Therefore, the XFCN value to use in this example is 010b.

SFR Definition 20.3. XBR2: Port I/O Crossbar Register 2

Bit	7	6	5	4	3	2	1	0
Name	WEAKPUD	XBARE	Reserved		CP2AE	CP2E	URT1E	LIN0E
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 = 0xC7; SFR Page = 0x0F

Bit	Name	Function
7	WEAKPUD	**Port I/O Weak Pullup Disable.** 0: Weak Pullups enabled (except for Ports whose I/O are configured for analog mode). 1: Weak Pullups disabled.
6	XBARE	**Crossbar Enable.** 0: Crossbar disabled. 1: Crossbar enabled.
5:4	Reserved	Always Write to 00b.
3	CP2AE	**Comparator2 Asynchronous Output Enable.** 0: Asynchronous CP2 unavailable at Port pin. 1: Asynchronous CP2 routed to Port pin.
2	CP2E	**Comparator2 Output Enable.** 0: CP2 unavailable at Port pin. 1: CP2 routed to Port pin.
1	URT1E	**UART1 I/O Output Enable.** 0: UART1 I/O unavailable at Port pin. 1: UART1 TX0, RX0 routed to Port pins.
0	LIN0E	**LIN I/O Output Enable.** 0: LIN I/O unavailable at Port pin. 1: LIN_TX, LIN_RX routed to Port pins.

SFR Definition 20.14. P0MDIN: Port 0 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	P0MDIN[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Address = 0xF1; SFR Page = 0x0F

Bit	Name	Function
7:0	P0MDIN[7:0]	**Analog Configuration Bits for P0.7–P0.0 (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 P0MDOUT register. 0: Corresponding P0.n pin is configured for analog mode. 1: Corresponding P0.n pin is not configured for analog mode.

SFR Definition 20.15. P0MDOUT: Port 0 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P0MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xA4; SFR Page = 0x0F

Bit	Name	Function
7:0	P0MDOUT[7:0]	**Output Configuration Bits for P0.7–P0.0 (respectively).** These bits are ignored if the corresponding bit in register P0MDIN is logic 0. 0: Corresponding P0.n Output is open-drain. 1: Corresponding P0.n Output is push-pull.

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.

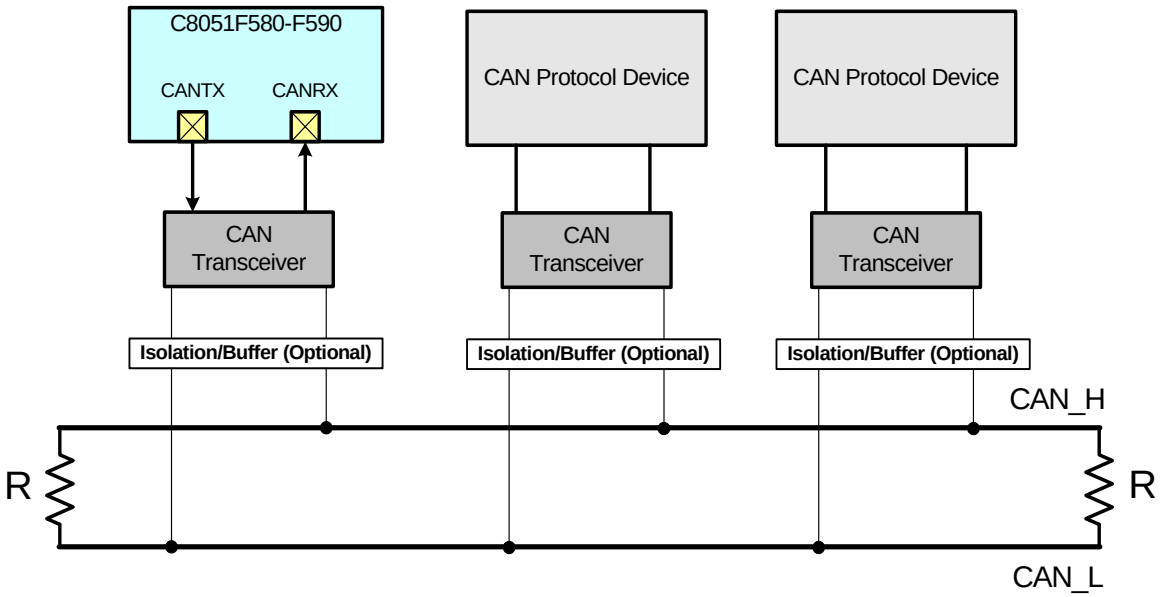

Figure 22.1. Typical CAN Bus Configuration

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.

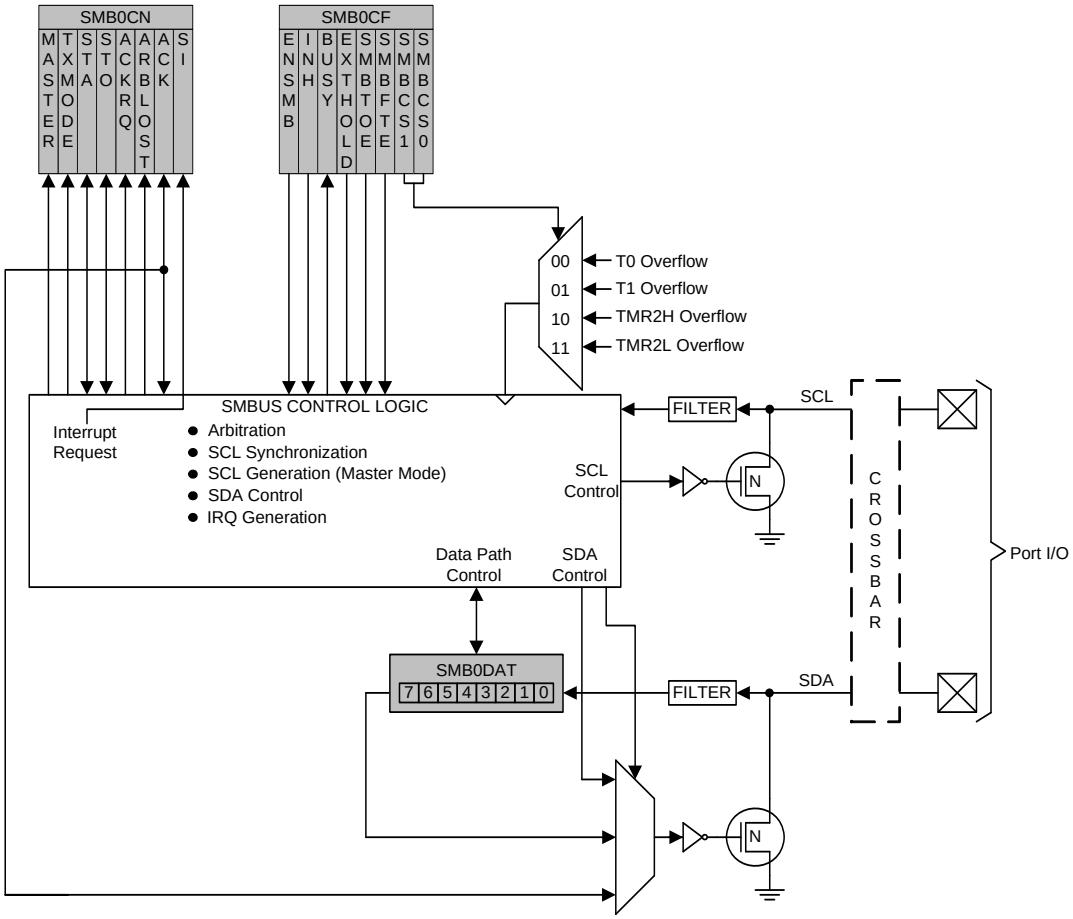

Figure 23.1. SMBus Block Diagram

23.4.3. Data Register

The SMBus Data register SMB0DAT holds a byte of serial data to be transmitted or one that has just been received. Software may safely read or write to the data register when the SI flag is set. Software should not attempt to access the SMB0DAT register when the SMBus is enabled and the SI flag is cleared to logic 0, as the interface may be in the process of shifting a byte of data into or out of the register.

Data in SMB0DAT is always shifted out MSB first. After a byte has been received, the first bit of received data is located at the MSB of SMB0DAT. While data is being shifted out, data on the bus is simultaneously being shifted in. SMB0DAT always contains the last data byte present on the bus. In the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data or address in SMB0DAT.

SFR Definition 23.3. SMB0DAT: SMBus Data

Bit	7	6	5	4	3	2	1	0
Name	SMB0DAT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xC2; SMB0DAT = 0x00

Bit	Name	Function
7:0	SMB0DAT[7:0]	**SMBus Data.** The SMB0DAT register contains a byte of data to be transmitted on the SMBus serial interface or a byte that has just been received on the SMBus serial interface. The CPU can read from or write to this register whenever the SI serial interrupt flag (SMB0CN.0) is set to logic 1. The serial data in the register remains stable as long as the SI flag is set. When the SI flag is not set, the system may be in the process of shifting data in/out and the CPU should not attempt to access this register.

23.5. SMBus Transfer Modes

The SMBus interface may be configured to operate as master and/or slave. At any particular time, it will be operating in one of the following four modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. The SMBus interface enters Master Mode any time a START is generated, and remains in Master Mode until it loses an arbitration or generates a STOP. An SMBus interrupt is generated at the end of all SMBus byte frames. As a receiver, the interrupt for an ACK occurs **before** the ACK. As a transmitter, interrupts occur **after** the ACK.

23.5.3. Write Sequence (Slave)

During a write sequence, an SMBus master writes data to a slave device. The slave in this transfer will be a receiver during the address byte, and a receiver during all data bytes. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode when a START followed by a slave address and direction bit (WRITE in this case) is received. Upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. The software must respond to the received slave address with an ACK, or ignore the received slave address with a NACK.

If 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 received. Software must write the ACK bit at that time to ACK or NACK the received byte.

The interface exits Slave Receiver Mode after receiving a STOP. Note that the interface will switch to Slave Transmitter Mode if SMB0DAT is written while an active Slave Receiver. Figure 23.7 shows a typical slave write sequence. Two received data bytes are shown, though any number of bytes may be received. Notice that the 'data byte transferred' interrupts occur **before** the ACK in this mode.

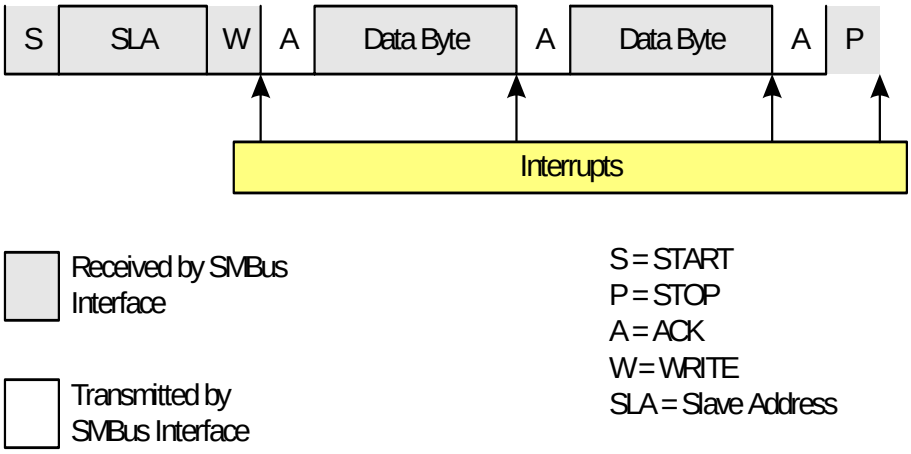

Figure 23.7. Typical Slave Write Sequence

27.1. 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 129); Timer 1 interrupts can be enabled by setting the ET1 bit in the IE register (Section “14.2. Interrupt Register Descriptions” on page 129). 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.

27.1.1. 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 192 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 27.1).

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 129), 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).

SFR Definition 27.20. TMRnCAPL: Timer 4 and 5 Capture Register Low Byte

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

TMR4CAPL SFR Address = 0xCA; TMR5CAPL SFR Address = 0x92; SFR Page = 0x10

Bit	Name	Function
7:0	TMRnCAPL[7:0]	**Timer n Reload Register Low Byte.** TMRnCAPL captures the low byte of Timer 4 and 5 when Timer 4 and 5 are configured in capture mode. When Timer 4 and 5 are configured in auto-reload mode, it holds the low byte of the reload value.

SFR Definition 27.21. TMRnCAPH: Timer 4 and 5 Capture Register High Byte

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

TMR4CAPH SFR Address = 0xCB; TMR5CAPH SFR Address = 0x93; SFR Page = 0x10

Bit	Name	Function
7:0	TMRnCAPH[7:0]	**Timer n Reload Register High Byte.** TMRnCAPH captures the high byte of Timer 4 and 5 when Timer 4 and 5 are configured in capture mode. When Timer 4 and 5 are configured in auto-reload mode, it holds the high byte of the reload value.

28. Programmable Counter Array 0 (PCA0)

The Programmable Counter Array (PCA0) provides enhanced timer functionality while requiring less CPU intervention than the standard 8051 counter/timers. PCA0 consists of a dedicated 16-bit counter/timer and six 16-bit capture/compare modules. 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 counter/timer is driven by a programmable timebase that can select between eight sources: system clock, system clock divided by four, system clock divided by twelve, the external oscillator clock source divided by 8, Timer 0, 4, or 5 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 (each mode is described in Section “28.3. Capture/Compare Modules” on page 317). The external oscillator clock option is ideal for real-time clock (RTC) functionality, allowing PCA0 to be clocked by a precision external oscillator while the internal oscillator drives the system clock. PCA0 is configured and controlled through the system controller's Special Function Registers. The PCA0 block diagram is shown in Figure 28.1

Important Note: PCA0 Module 5 may be used as a watchdog timer (WDT), and is enabled in this mode following a system reset. **Access to certain PCA0 registers is restricted while WDT mode is enabled**. See Section 28.4 for details.

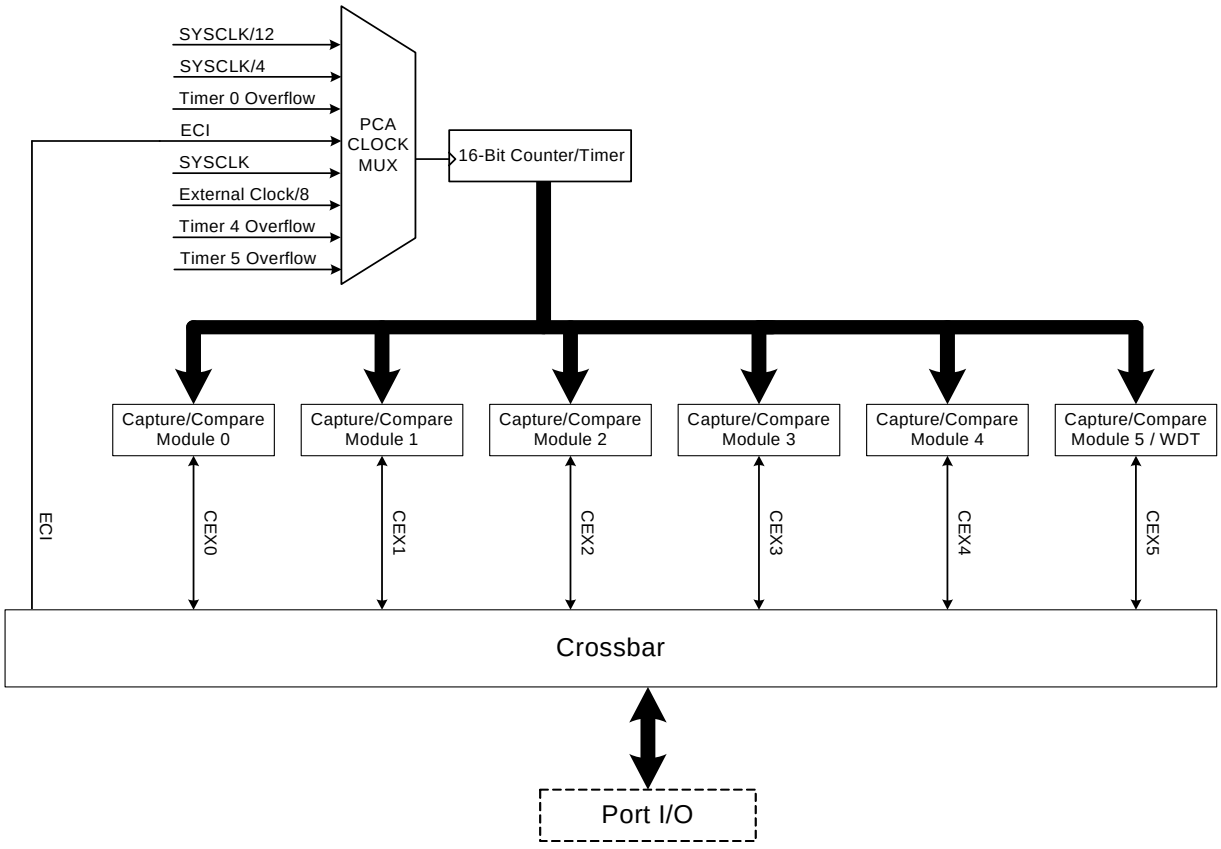

Figure 28.1. PCA0 Block Diagram