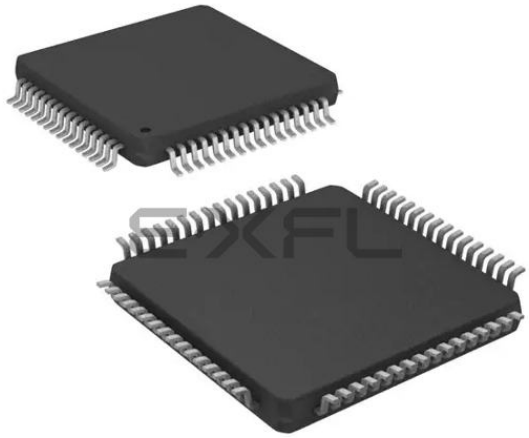


Welcome to E-XFL.COM

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	Obsolete
Core Processor	8051
Core Size	8-Bit
Speed	25MHz
Connectivity	EBI/EMI, SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	Cap Sense, POR, PWM, WDT
Number of I/O	54
Program Memory Size	15KB (15K x 8)
Program Memory Type	FLASH
EEPROM Size	32 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f701-gq

SFR Definition 28.29. P4DRV: Port 4 Drive Strength 206
SFR Definition 28.30. P5: Port 5 206
SFR Definition 28.31. P5MDIN: Port 5 Input Mode 207
SFR Definition 28.32. P5MDOUT: Port 5 Output Mode 207
SFR Definition 28.33. P5DRV: Port 5 Drive Strength 208
SFR Definition 28.34. P6: Port 6 208
SFR Definition 28.35. P6MDIN: Port 6 Input Mode 209
SFR Definition 28.36. P6MDOUT: Port 6 Output Mode 209
SFR Definition 28.37. P6DRV: Port 6 Drive Strength 210
SFR Definition 29.1. CRC0CN: CRC0 Control 215
SFR Definition 29.2. CRC0IN: CRC Data Input 216
SFR Definition 29.3. CRC0DATA: CRC Data Output 216
SFR Definition 29.4. CRC0AUTO: CRC Automatic Control 217
SFR Definition 29.5. CRC0CNT: CRC Automatic Flash Sector Count 217
SFR Definition 29.6. CRC0FLIP: CRC Bit Flip 218
SFR Definition 30.1. SMB0CF: SMBus Clock/Configuration 225
SFR Definition 30.2. SMB0CN: SMBus Control 227
SFR Definition 30.3. SMB0ADR: SMBus Slave Address 229
SFR Definition 30.4. SMB0ADM: SMBus Slave Address Mask 230
SFR Definition 30.5. SMB0DAT: SMBus Data 231
SFR Definition 31.1. SPI0CFG: SPI0 Configuration 248
SFR Definition 31.2. SPI0CN: SPI0 Control 249
SFR Definition 31.3. SPI0CKR: SPI0 Clock Rate 250
SFR Definition 31.4. SPI0DAT: SPI0 Data 250
SFR Definition 32.1. SCON0: Serial Port 0 Control 259
SFR Definition 32.2. SBUF0: Serial (UART0) Port Data Buffer 260
SFR Definition 33.1. CKCON: Clock Control 263
SFR Definition 33.2. TCON: Timer Control 268
SFR Definition 33.3. TMOD: Timer Mode 269
SFR Definition 33.4. TL0: Timer 0 Low Byte 270
SFR Definition 33.5. TL1: Timer 1 Low Byte 270
SFR Definition 33.6. TH0: Timer 0 High Byte 271
SFR Definition 33.7. TH1: Timer 1 High Byte 271
SFR Definition 33.8. TMR2CN: Timer 2 Control 275
SFR Definition 33.9. TMR2RLL: Timer 2 Reload Register Low Byte 276
SFR Definition 33.10. TMR2RLH: Timer 2 Reload Register High Byte 276
SFR Definition 33.11. TMR2L: Timer 2 Low Byte 277
SFR Definition 33.12. TMR2H Timer 2 High Byte 277
SFR Definition 33.13. TMR3CN: Timer 3 Control 281
SFR Definition 33.14. TMR3RLL: Timer 3 Reload Register Low Byte 282
SFR Definition 33.15. TMR3RLH: Timer 3 Reload Register High Byte 282
SFR Definition 33.16. TMR3L: Timer 3 Low Byte 283
SFR Definition 33.17. TMR3H Timer 3 High Byte 283
SFR Definition 34.1. PCA0CN: PCA Control 295
SFR Definition 34.2. PCA0MD: PCA Mode 296

Table 3.1. Pin Definitions for the C8051F70x/71x (Continued)

Name	TQFP64	TQFP48 QFN48	QFN32	QFN24	Type	Description
P0.7	48	35	—	—	D I/O or A In	Port 0.7. ADC0 Input.
P1.0	47	34	—	—	D I/O or A In	Port 1.0. ADC0 Input.
P1.1	46	33	—	—	D I/O or A In	Port 1.1. ADC0 Input.
P1.2	45	32	—	—	D I/O or A In	Port 1.2. ADC0 Input.
P1.3	44	31	—	—	D I/O or A In	Port 1.3. ADC0 Input.
P1.4	43	—	—	—	D I/O or A In	Port 1.4. ADC0 Input.
P1.5	42	—	—	—	D I/O or A In	Port 1.5. ADC0 Input.
P1.6	39	—	—	—	D I/O or A In	Port 1.6. ADC0 Input.
P1.7	38	—	—	—	D I/O or A In	Port 1.7. ADC0 Input.
P2.0	37	29	23	17	D I/O or A In	Port 2.0. CS0 input pin 1.
P2.1	36	28	22	16	D I/O or A In	Port 2.1. CS0 input pin 2.
P2.2	35	27	21	15	D I/O or A In	Port 2.2. CS0 input pin 3.
P2.3	34	26	20	14	D I/O or A In	Port 2.3. CS0 input pin 4.
P2.4	33	25	19	13	D I/O or A In	Port 2.4. CS0 input pin 5.
P2.5	32	24	18	12	D I/O or A In	Port 2.5. CS0 input pin 6.
P2.6	31	23	17	11	D I/O or A In	Port 2.6. CS0 input pin 7.
P2.7	30	22	16	10	D I/O or A In	Port 2.7. CS0 input pin 8.

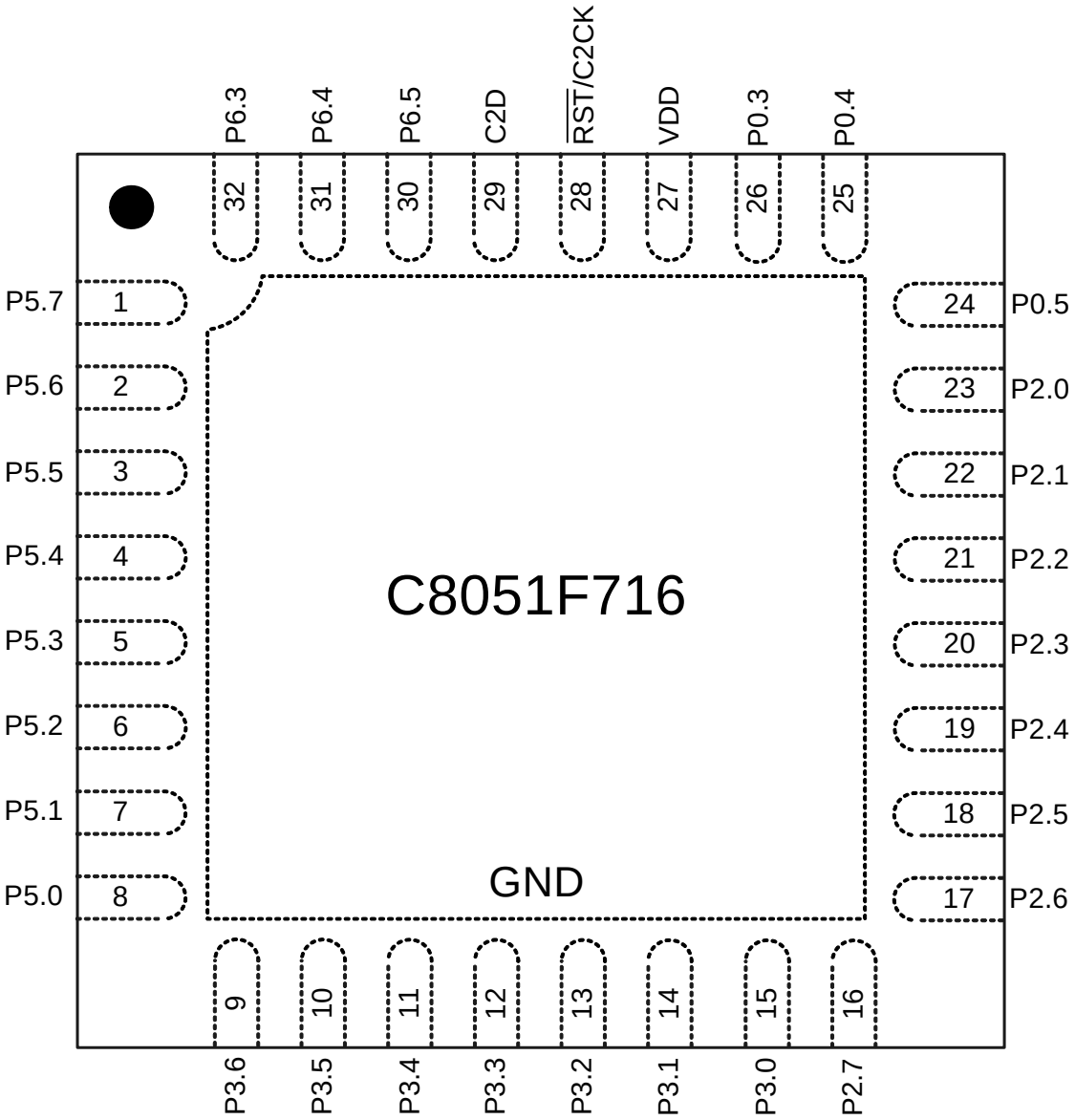


Figure 3.4. C8051F716-GM QFN32 Pinout Diagram (Top View)

7. QFN-32 Package Specifications

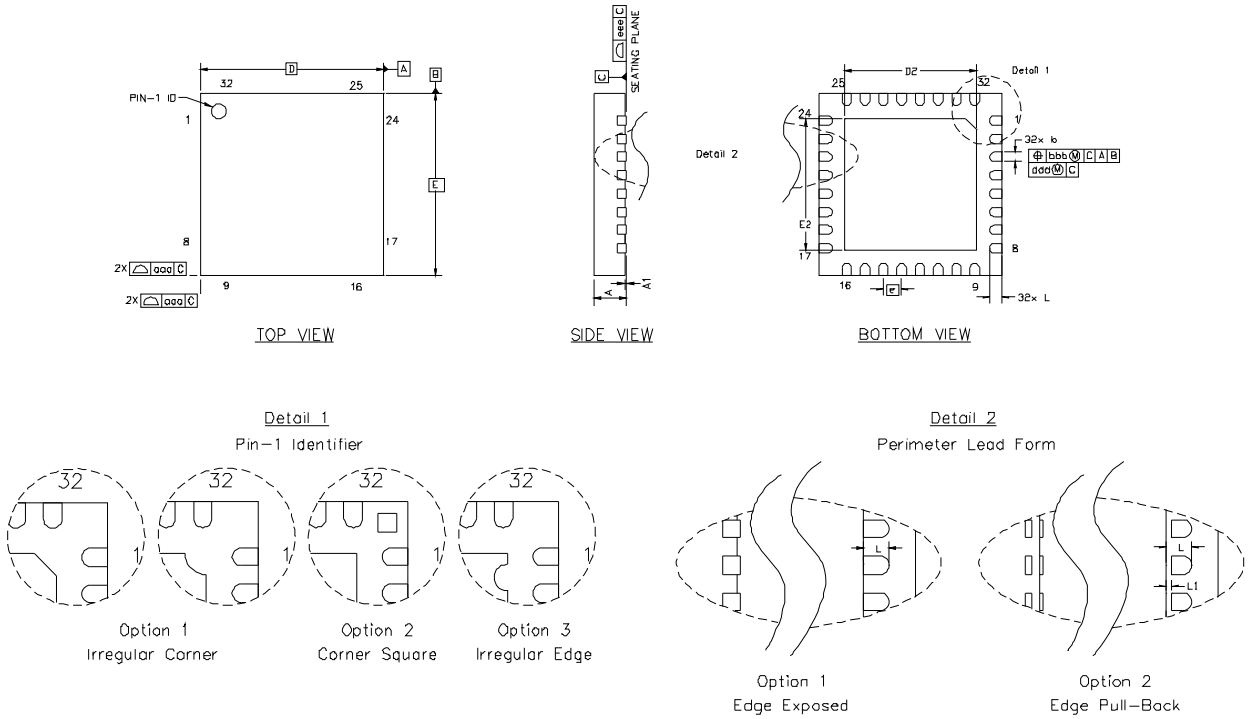


Figure 7.1. QFN-32 Package Drawing

Table 7.1. QFN-32 Package Dimensions

Dimension	Min	Typ	Max
A	0.80	0.90	1.00
A1	0.00	0.02	0.05
b	0.18	0.25	0.30
D		5.00 BSC.	
D2	3.50	3.60	3.70
e		0.50 BSC.	
E		5.00 BSC.	

Dimension	Min	Typ	Max
E2	3.50	3.60	3.70
L	0.30	0.35	0.40
L1	0.00	—	0.10
aaa		0.15	
bbb		0.10	
ddd		0.05	
eee		0.08	

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 Solid State Outline MO-220, variation VHHD except for custom features D2, E2, L and L1 which are toleranced per supplier designation.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

11. Temperature Sensor

An on-chip temperature sensor is included on the C8051F700/2/4/6/8 and C8051F710/2/4/6 which can be directly accessed via the ADC multiplexer in single-ended configuration. To use the ADC to measure the temperature sensor, the ADC mux channel should be configured to connect to the temperature sensor. The temperature sensor transfer function is shown in Figure 11.1. The output voltage (V_{TEMP}) is the positive ADC input when the ADC multiplexer is set correctly. The TEMPE bit in register REF0CN enables/disables the temperature sensor, as described in SFR Definition 12.1. While disabled, the temperature sensor defaults to a high impedance state and any ADC measurements performed on the sensor will result in meaningless data. Refer to Table 9.12 for the slope and offset parameters of the temperature sensor.

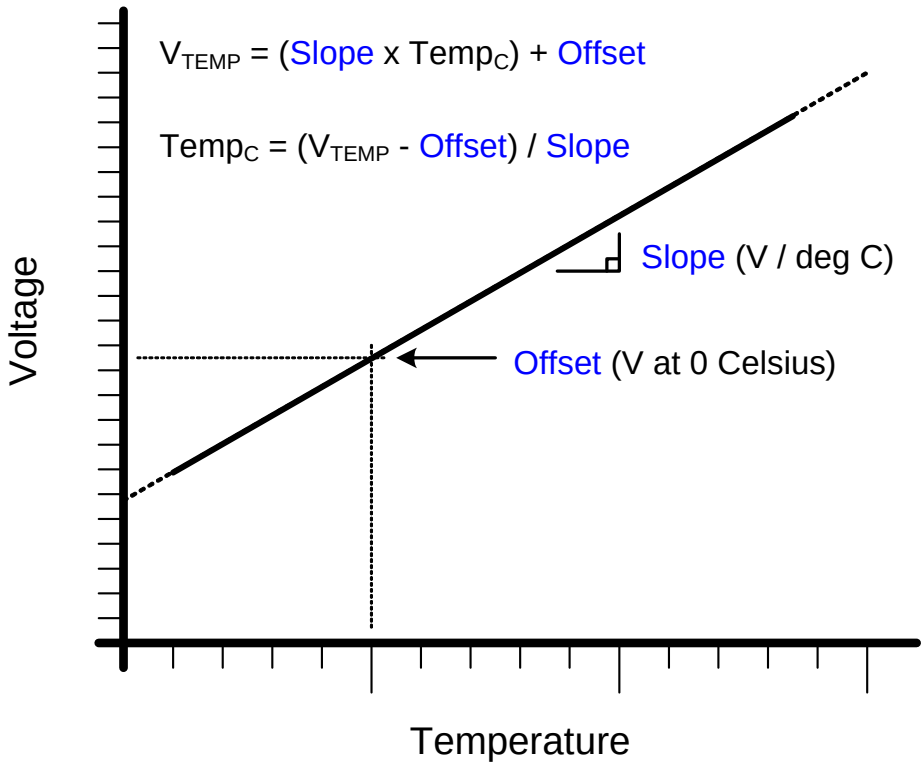


Figure 11.1. Temperature Sensor Transfer Function

11.1. Calibration

The uncalibrated temperature sensor output is extremely linear and suitable for relative temperature measurements (see Table 5.1 for linearity specifications). For absolute temperature measurements, offset and/or gain calibration is recommended. Typically a 1-point (offset) calibration includes the following steps:

1. Control/measure the ambient temperature (this temperature must be known).
2. Power the device, and delay for a few seconds to allow for self-heating.
3. Perform an ADC conversion with the temperature sensor selected as the ADC's input.
4. Calculate the offset characteristics, and store this value in non-volatile memory for use with subsequent temperature sensor measurements.

Figure 5.3 shows the typical temperature sensor error assuming a 1-point calibration at 0 °C.

SFR Definition 18.3. EMI0TC: External Memory Timing Control

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

SFR Address = 0xEE; SFR Page = F

Bit	Name	Function
7:6	EAS[1:0]	**EMIF Address Setup Time Bits.** 00: Address setup time = 0 SYSCLK cycles. 01: Address setup time = 1 SYSCLK cycle. 10: Address setup time = 2 SYSCLK cycles. 11: Address setup time = 3 SYSCLK cycles.
5:2	EWR[3:0]	**EMIF $\overline{WR}$ and $\overline{RD}$ Pulse-Width Control Bits.** 0000: $\overline{WR}$ and $\overline{RD}$ pulse width = 1 SYSCLK cycle. 0001: $\overline{WR}$ and $\overline{RD}$ pulse width = 2 SYSCLK cycles. 0010: $\overline{WR}$ and $\overline{RD}$ pulse width = 3 SYSCLK cycles. 0011: $\overline{WR}$ and $\overline{RD}$ pulse width = 4 SYSCLK cycles. 0100: $\overline{WR}$ and $\overline{RD}$ pulse width = 5 SYSCLK cycles. 0101: $\overline{WR}$ and $\overline{RD}$ pulse width = 6 SYSCLK cycles. 0110: $\overline{WR}$ and $\overline{RD}$ pulse width = 7 SYSCLK cycles. 0111: $\overline{WR}$ and $\overline{RD}$ pulse width = 8 SYSCLK cycles. 1000: $\overline{WR}$ and $\overline{RD}$ pulse width = 9 SYSCLK cycles. 1001: $\overline{WR}$ and $\overline{RD}$ pulse width = 10 SYSCLK cycles. 1010: $\overline{WR}$ and $\overline{RD}$ pulse width = 11 SYSCLK cycles. 1011: $\overline{WR}$ and $\overline{RD}$ pulse width = 12 SYSCLK cycles. 1100: $\overline{WR}$ and $\overline{RD}$ pulse width = 13 SYSCLK cycles. 1101: $\overline{WR}$ and $\overline{RD}$ pulse width = 14 SYSCLK cycles. 1110: $\overline{WR}$ and $\overline{RD}$ pulse width = 15 SYSCLK cycles. 1111: $\overline{WR}$ and $\overline{RD}$ pulse width = 16 SYSCLK cycles.
1:0	EAH[1:0]	**EMIF Address Hold Time Bits.** 00: Address hold time = 0 SYSCLK cycles. 01: Address hold time = 1 SYSCLK cycle. 10: Address hold time = 2 SYSCLK cycles. 11: Address hold time = 3 SYSCLK cycles.

Table 20.1. Special Function Register (SFR) Memory Map

Addr	SFR Page	0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)
F8	0 F	**SPI0CN**	PCA0L P0DRV	PCA0H P1DRV	PCA0CPL0 P2DRV	PCA0CPH0 P3DRV	P4DRV	P5DRV	**VDM0CN**
F0	0 F	**B**	P0MDIN	P1MDIN	P0MAT P2MDIN	P0MASK P3MDIN	P4MDIN	P5MDIN	P6MDIN
E8	0 F	**ADC0CN**	PCA0CPL1	PCA0CPH1	PCA0CPL2	PCA0CPH2 DERIVID	PCA0MD	EMI0TC	**RSTSRC**
E0	0 F	**ACC**	P1MAT XBR0	P1MASK XBR1	**WDTCN**	IT01CF		**EIE1**	**EIE2**
D8	0 F	**PCA0CN**	CRC0DATA	PCA0CPM0	PCA0CPM1	PCA0CPM2			
D0	0 F	**PSW**	**EEDATA**	REF0CN		P0SKIP	P1SKIP	P2SKIP	
C8	0 F	**TMR2CN**		TMR2RLL	TMR2RLH	TMR2L	TMR2H	EIP1	EIP2
C0	0 F	**SMB0CN**	SMB0CF P6DRV	SMB0DAT	ADC0GTL	ADC0GTH HWID	ADC0LTL EECNTL	ADC0LTH EEKEY	EMI0CF
B8	0 F	**IP**	REG0CN	SMB0ADR	ADC0MX SMB0ADM	ADC0CF	ADC0L CLKSEL	ADC0H CS0MD2	OSCICL
B0	0 F	**P3**		**P6**	**P5**		OSCXCN	**EEADDR**	**FLKEY**
A8	0 F	**IE**	CS0DL OSCICN	CS0DH EMI0CN		**P4**	CS0MD1 REVID		P3MDOUT
A0	0 F	**P2**	SPI0CFG PCA0PWM	SPI0CKR	SPI0DAT	P0MDOUT	P1MDOUT	P2MDOUT	**SFRPAGE**
98	0 F	**SCON0**	**SBUF0**	CS0CN P4MDOUT	CPT0CN P5MDOUT	CS0MX P6MDOUT	CPT0MD	CS0CF	CPT0MX CS0PM
90	0 F	**P1**	TMR3CN CRC0CN	TMR3RLL CS0SS	TMR3RLH CS0SE	TMR3L CRC0IN	TMR3H CRC0FLIP	CS0THL CRC0AUTO	CS0THH CRC0CNT
88	0 F	**TCON**	**TMOD**	**TL0**	**TL1**	**TH0**	**TH1**	**CKCON**	**PSCTL**
80	0 F	**P0**	**SP**	**DPL**	**DPH**				**PCON**
		0(8)	1(9)	2(A)	3(B)	4(C)	5(D)	6(E)	7(F)

Notes:

1. SFR addresses ending in 0x0 or 0x8 (leftmost column) are bit-addressable.
2. SFRs indicated with bold lettering and shaded cells are available on both SFR Page 0 and F.

6. Using the MOVX instruction, write a data byte to any location within the 512-byte page to be erased.
7. Clear the PSWE and PSEE bits.
8. Restore previous interrupt state.

Steps 4–6 must be repeated for each 512-byte page to be erased.

Note: Flash security settings may prevent erasure of some Flash pages, such as the reserved area and the page containing the lock bytes. For a summary of Flash security settings and restrictions affecting Flash erase operations, please see Section “22.3. Security Options” on page 149.

22.1.3. Flash Write Procedure

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 recommended procedure for writing a single byte in Flash is as follows:

1. Save current interrupt state and disable interrupts.
2. Ensure that the Flash byte has been erased (has a value of 0xFF).
3. Set the PSWE bit (register PSCTL).
4. Clear the PSEE bit (register PSCTL).
5. Write the first key code to FLKEY: 0xA5.
6. Write the second key code to FLKEY: 0xF1.
7. Using the MOVX instruction, write a single data byte to the desired location within the 512-byte sector.
8. Clear the PSWE bit.
9. Restore previous interrupt state.

Steps 5–7 must be repeated for each byte to be written.

Note: Flash security settings may prevent writes to some areas of Flash, such as the reserved area. For a summary of Flash security settings and restrictions affecting Flash write operations, please see Section “22.3. Security Options” on page 149.

22.2. Non-volatile Data Storage

The Flash memory can be used for non-volatile data storage as well as program code. This allows data such as calibration coefficients to be calculated and stored at run time. Data is written using the MOVX write instruction and read using the MOVC instruction.

Note: MOVX read instructions always target XRAM.

22.3. Security Options

The CIP-51 provides security options to protect the Flash memory from inadvertent modification by software as well as to prevent the viewing of proprietary program code and constants. The Program Store Write Enable (bit PSWE in register PSCTL) and the Program Store Erase Enable (bit PSEE in register PSCTL) bits protect the Flash memory from accidental modification by software. PSWE must be explicitly set to 1 before software can modify the Flash memory; both PSWE and PSEE must be set to 1 before software can erase Flash memory. Additional security features prevent proprietary program code and data constants from being read or altered across the C2 interface.

A Security Lock Byte located at the last byte of Flash user space offers protection of the Flash program memory from access (reads, writes, and erases) by unprotected code or the C2 interface. The Flash security mechanism allows the user to lock all Flash pages, starting at page 0, by writing a non-0xFF value to the lock byte. **Note that writing a non-0xFF value to the lock byte will lock all pages of FLASH from reads, writes, and erases, including the page containing the lock byte.**

The level of Flash security depends on the Flash access method. The three Flash access methods that can be restricted are reads, writes, and erases from the C2 debug interface, user firmware executing on

The following guidelines are recommended for any system that contains routines which write or erase Flash from code.

22.4.1. VDD Maintenance and the VDD Monitor

1. If the system power supply is subject to voltage or current "spikes," add sufficient transient protection devices to the power supply to ensure that the supply voltages listed in the Absolute Maximum Ratings table are not exceeded.
2. Make certain that the minimum VDD rise time specification of 1 ms is met. If the system cannot meet this rise time specification, then add an external VDD brownout circuit to the /RST pin of the device that holds the device in reset until VDD reaches the minimum device operating voltage and re-asserts /RST if VDD drops below the minimum device operating voltage.
3. Keep the on-chip VDD Monitor enabled and enable the VDD Monitor as a reset source as early in code as possible. This should be the first set of instructions executed after the Reset Vector. For C-based systems, this will involve modifying the startup code added by the C compiler. See your compiler documentation for more details. Make certain that there are no delays in software between enabling the VDD Monitor and enabling the VDD Monitor as a reset source. Code examples showing this can be found in “AN201: Writing to Flash from Firmware," available from the Silicon Laboratories web site.

Note: On C8051F70x/71x devices, both the VDD Monitor and the VDD Monitor reset source must be enabled to write or erase Flash without generating a Flash Error Device Reset.

On C8051F70x/71x devices, both the VDD Monitor and the VDD Monitor reset source are enabled by hardware after a power-on reset.

4. As an added precaution, explicitly enable the VDD Monitor and enable the VDD Monitor as a reset source inside the functions that write and erase Flash memory. The VDD Monitor enable instructions should be placed just after the instruction to set PSWE to a 1, but before the Flash write or erase operation instruction.
5. Make certain that all writes to the RSTSRC (Reset Sources) register use direct assignment operators and explicitly DO NOT use the bit-wise operators (such as AND or OR). For example, "RSTSRC = 0x02" is correct, but "RSTSRC |= 0x02" is incorrect.
6. Make certain that all writes to the RSTSRC register explicitly set the PORSF bit to a 1. Areas to check are initialization code which enables other reset sources, such as the Missing Clock Detector or Comparator, for example, and instructions which force a Software Reset. A global search on "RSTSRC" can quickly verify this.

22.4.2. PSWE Maintenance

7. Reduce the number of places in code where the PSWE bit (b0 in PSCTL) is set to a 1. There should be exactly one routine in code that sets PSWE to a 1 to write Flash bytes and one routine in code that sets both PSWE and PSEE both to a 1 to erase Flash pages.
8. Minimize the number of variable accesses while PSWE is set to a 1. Handle pointer address updates and loop maintenance outside the "PSWE = 1;... PSWE = 0;" area. Code examples showing this can be found in “AN201: Writing to Flash from Firmware," available from the Silicon Laboratories web site.
9. Disable interrupts prior to setting PSWE to a 1 and leave them disabled until after PSWE has been reset to 0. Any interrupts posted during the Flash write or erase operation will be serviced in priority order after the Flash operation has been completed and interrupts have been re-enabled by software.
10. Make certain that the Flash write and erase pointer variables are not located in XRAM. See your compiler documentation for instructions regarding how to explicitly locate variables in different memory areas.
11. Add address bounds checking to the routines that write or erase Flash memory to ensure that a routine called with an illegal address does not result in modification of the Flash.

23. EEPROM

C8051F700/1/4/5/8/9 and C8051F712/3 devices have hardware which emulates 32 bytes of non-volatile, byte-programmable EEPROM data space. The module mirrors each non-volatile byte through 32 bytes of volatile data space. This data space can be accessed indirectly through EEADDR and EEDATA. Users can copy the complete 32-byte image between EEPROM space and volatile space using controls in the EECNTL SFR.

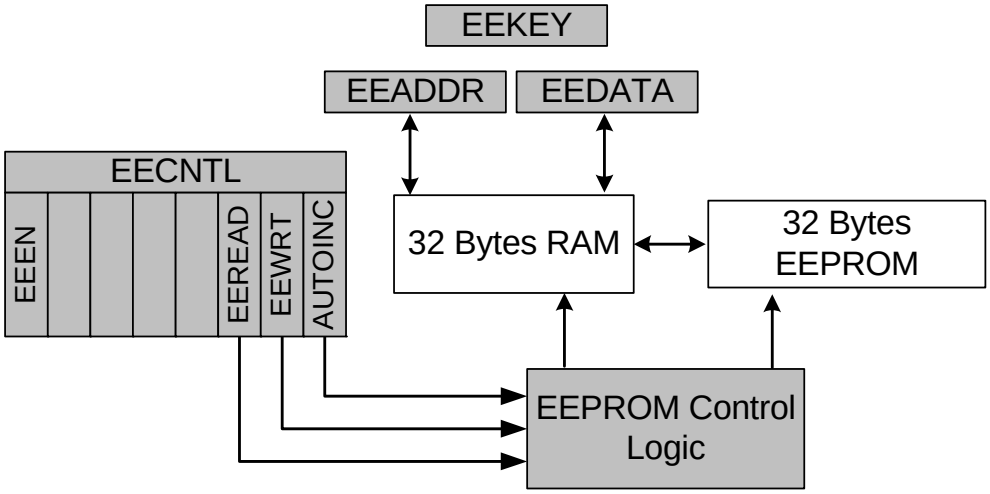


Figure 23.1. EEPROM Block Diagram

23.1. RAM Reads and Writes

In order to perform EEPROM reads and writes, the EEPROM control logic must be enabled by setting EEEN (EECNTL.7).

32 bytes of RAM can be accessed indirectly through EEADDR and EEDATA. To write to a byte of RAM, write address of byte to EEADDR and then write the value to be written to EEDATA. To read a byte from RAM, write address of byte to be read to EEADDR. The value stored at that address can then be read from EEDATA.

23.2. Auto Increment

When AUTOINC (EECNTL.0) is set, EEADDR will increment by one after each write to EEDATA and each read from EEDATA. When Auto Increment is enabled and EEADDR reaches the top address of dedicated RAM space, the next write to or read from EEDATA will cause EEADDR to wrap along the address boundary, which will set the address to 0.

23.3. Interfacing with the EEPROM

The EEPROM is accessed through the dedicated 32 bytes of RAM. Writes to EEPROM are allowed only after writes have been enabled (see “23.4. EEPROM Security”). The contents of the EEPROM can be uploaded to the RAM by setting EEREAD (EECNTL.2). Contents of RAM can be downloaded to EEPROM by setting EEWRT (EENTL.1).

Note: A minimum SYSCLK frequency is required for writing EEPROM memory, as detailed in Section “Table 9.9. EEPROM Electrical Characteristics” on page 52.

SFR Definition 27.1. CLKSEL: Clock Select

Bit	7	6	5	4	3	2	1	0
Name	CLKRDY	CLKDIV[2:0]			Reserved	CLKSEL[2:0]		
Type	R	R/W	R/W	R/W	R	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xBD; SFR Page= F

Bit	Name	Function
7	CLKRDY	**System Clock Divider Clock Ready Flag.** 0: The selected clock divide setting has not been applied to the system clock. 1: The selected clock divide setting has been applied to the system clock.
6:4	CLKDIV	**System Clock Divider Bits.** Selects the clock division to be applied to the selected source (internal or external). 000: Selected clock is divided by 1. 001: Selected clock is divided by 2. 010: Selected clock is divided by 4. 011: Selected clock is divided by 8. 100: Selected clock is divided by 16. 101: Selected clock is divided by 32. 110: Selected clock is divided by 64. 111: Selected clock is divided by 128.
3	Reserved	Read = 0b. Must write 0b.
2:0	CLKSEL[2:0]	**System Clock Select.** Selects the oscillator to be used as the undivided system clock source. 000: Internal Oscillator 001: External Oscillator All other values reserved.

28. Port Input/Output

Digital and analog resources are available through 64 I/O pins. Each of the Port pins P0.0–P2.7 can be defined as general-purpose I/O (GPIO), assigned to one of the internal digital resources, or assigned to an analog function as shown in Figure 28.4. The designer has complete control over which functions are assigned, limited only by the number of physical I/O pins. This resource assignment flexibility is achieved through the use of a Priority Crossbar Decoder. The state of a Port I/O pin can always be read in the corresponding Port latch, regardless of the Crossbar settings.

The Crossbar assigns the selected internal digital resources to the I/O pins based on the Priority Decoder. The registers XBR0 and XBR1, defined in SFR Definition 28.1 and SFR Definition 28.2, are used to select internal digital functions.

All Port I/Os except P0.3 are tolerant of voltages up to 2 V above the V_{DD} supply (refer to Figure 28.2 for the Port cell circuit). The Port I/O cells are configured as either push-pull or open-drain in the Port Output Mode registers (PnMDOUT, where n = 0,1). Complete Electrical Specifications for Port I/O are given in Section “9. Electrical Characteristics” on page 47.

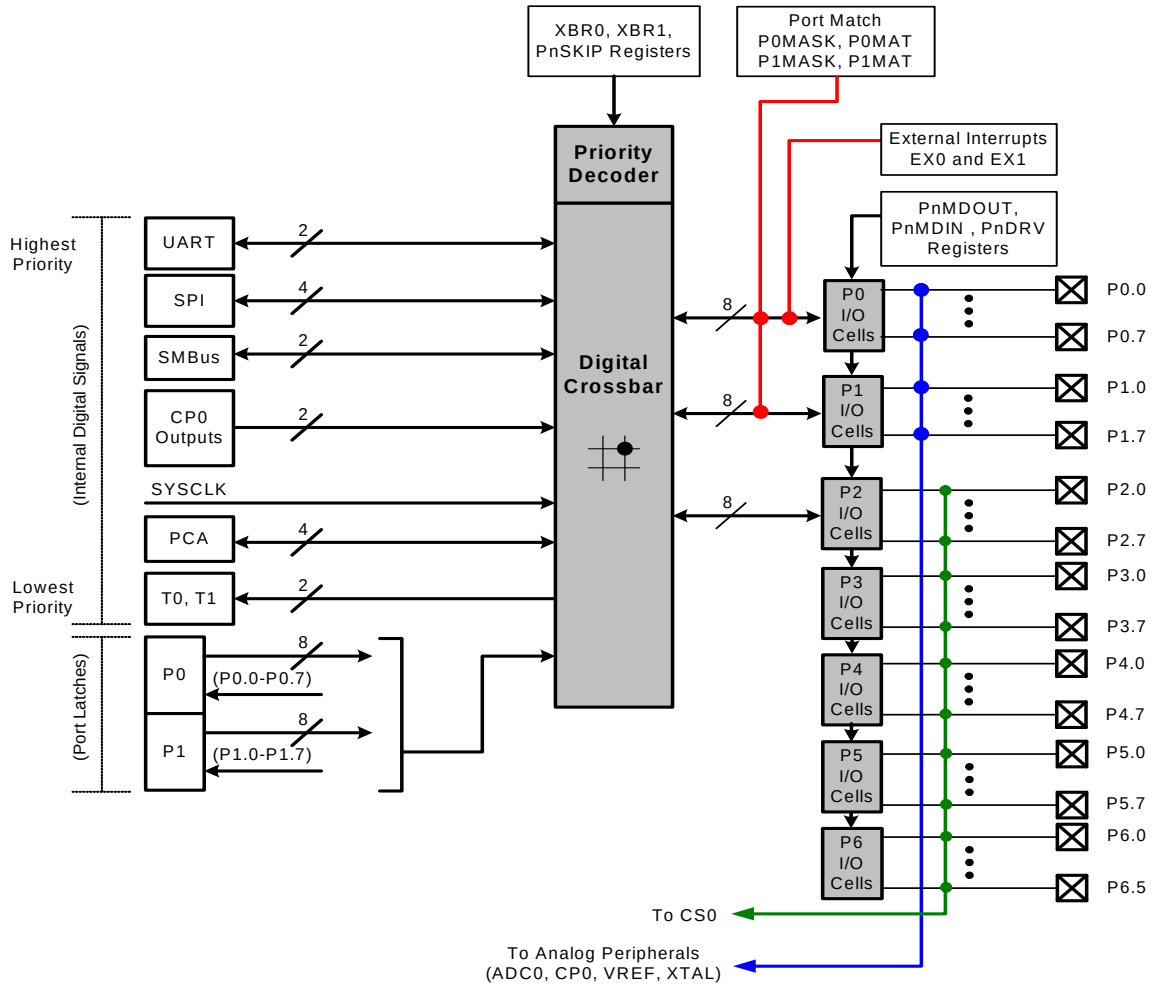
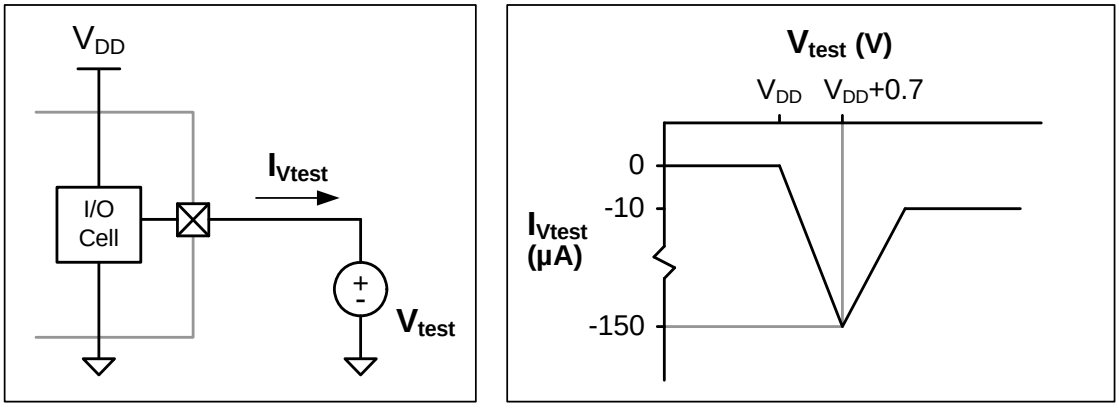


Figure 28.1. Port I/O Functional Block Diagram

28.1.3. Interfacing Port I/O to 5 V Logic

All Port I/O configured for digital, open-drain operation are capable of interfacing to digital logic operating at a supply voltage up to 2 V higher than VDD and less than 5.25 V. An external pull-up resistor to the higher supply voltage is typically required for most systems.

Important Note: In a multi-voltage interface, the external pull-up resistor should be sized to allow a current of at least 150 µA to flow into the Port pin when the supply voltage is between (VDD + 0. 6V) and (VDD + 1.0V). Once the Port pin voltage increases beyond this range, the current flowing into the Port pin is minimal. Figure 28.3 shows the input current characteristics of port pins driven above VDD. The port pin requires 150 µA peak overdrive current when its voltage reaches approximately (VDD + 0.7 V).



Port I/O Overdrive Test Circuit **Port I/O Overdrive Current vs. Voltage**

Figure 28.3. Port I/O Overdrive Current

28.1.4. Increasing Port I/O Drive Strength

Port I/O output drivers support a high and low drive strength; the default is low drive strength. The drive strength of a Port I/O can be configured using the PnDRV registers. See Section “9. Electrical Characteristics” on page 47 for the difference in output drive strength between the two modes.

28.2. Assigning Port I/O Pins to Analog and Digital Functions

Port I/O pins P0.0–P2.7 can be assigned to various analog, digital, and external interrupt functions. The Port pins assigned to analog functions should be configured for analog I/O, and Port pins assigned to digital or external interrupt functions should be configured for digital I/O.

28.2.1. Assigning Port I/O Pins to Analog Functions

Table 28.1 shows all available analog functions that require Port I/O assignments. **Port pins selected for these analog functions should have their corresponding bit in PnSKIP set to 1.** This reserves the pin for use by the analog function and does not allow it to be claimed by the Crossbar. Table 28.1 shows the potential mapping of Port I/O to each analog function.

28.5. Port Match

Port match functionality allows system events to be triggered by a logic value change on P0 or P1. A software controlled value stored in the PnMATCH registers specifies the expected or normal logic values of P0 and P1. A Port mismatch event occurs if the logic levels of the Port's input pins no longer match the software controlled value. This allows Software to be notified if a certain change or pattern occurs on P0 or P1 input pins regardless of the XBRn settings.

The PnMASK registers can be used to individually select which P0 and P1 pins should be compared against the PnMATCH registers. A Port mismatch event is generated if (P0 & P0MASK) does not equal (P0MATCH & P0MASK) or if (P1 & P1MASK) does not equal (P1MATCH & P1MASK).

A Port mismatch event may be used to generate an interrupt or wake the device from a low power mode, such as IDLE or SUSPEND. See the Interrupts and Power Options chapters for more details on interrupt and wake-up sources.

SFR Definition 28.3. P0MASK: Port 0 Mask Register

Bit	7	6	5	4	3	2	1	0
Name	P0MASK[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xF4; SFR Page = 0

Bit	Name	Function
7:0	P0MASK[7:0]	**Port 0 Mask Value.** Selects P0 pins to be compared to the corresponding bits in P0MAT. 0: P0.n pin logic value is ignored and cannot cause a Port Mismatch event. 1: P0.n pin logic value is compared to P0MAT.n.

30. 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 I^2C 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. The SMBus peripheral can be fully driven by software (i.e., software accepts/rejects slave addresses, and generates ACKs), or hardware slave address recognition and automatic ACK generation can be enabled to minimize software overhead. A block diagram of the SMBus peripheral and the associated SFRs is shown in Figure 30.1.

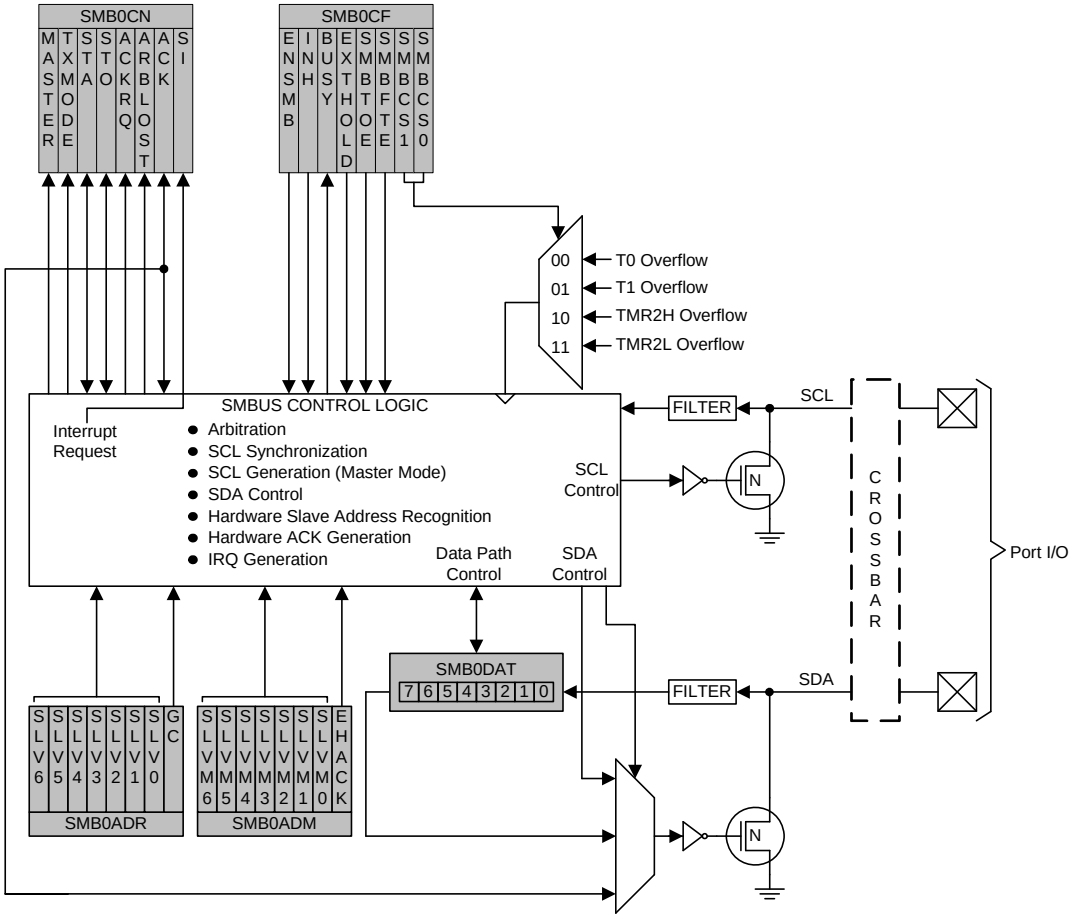


Figure 30.1. SMBus Block Diagram

31. Enhanced Serial Peripheral Interface (SPI0)

The Enhanced Serial Peripheral Interface (SPI0) provides access to a flexible, full-duplex synchronous serial bus. SPI0 can operate as a master or slave device in both 3-wire or 4-wire modes, and supports multiple masters and slaves on a single SPI bus. The slave-select (NSS) signal can be configured as an input to select SPI0 in slave mode, or to disable Master Mode operation in a multi-master environment, avoiding contention on the SPI bus when more than one master attempts simultaneous data transfers. NSS can also be configured as a chip-select output in master mode, or disabled for 3-wire operation. Additional general purpose port I/O pins can be used to select multiple slave devices in master mode.

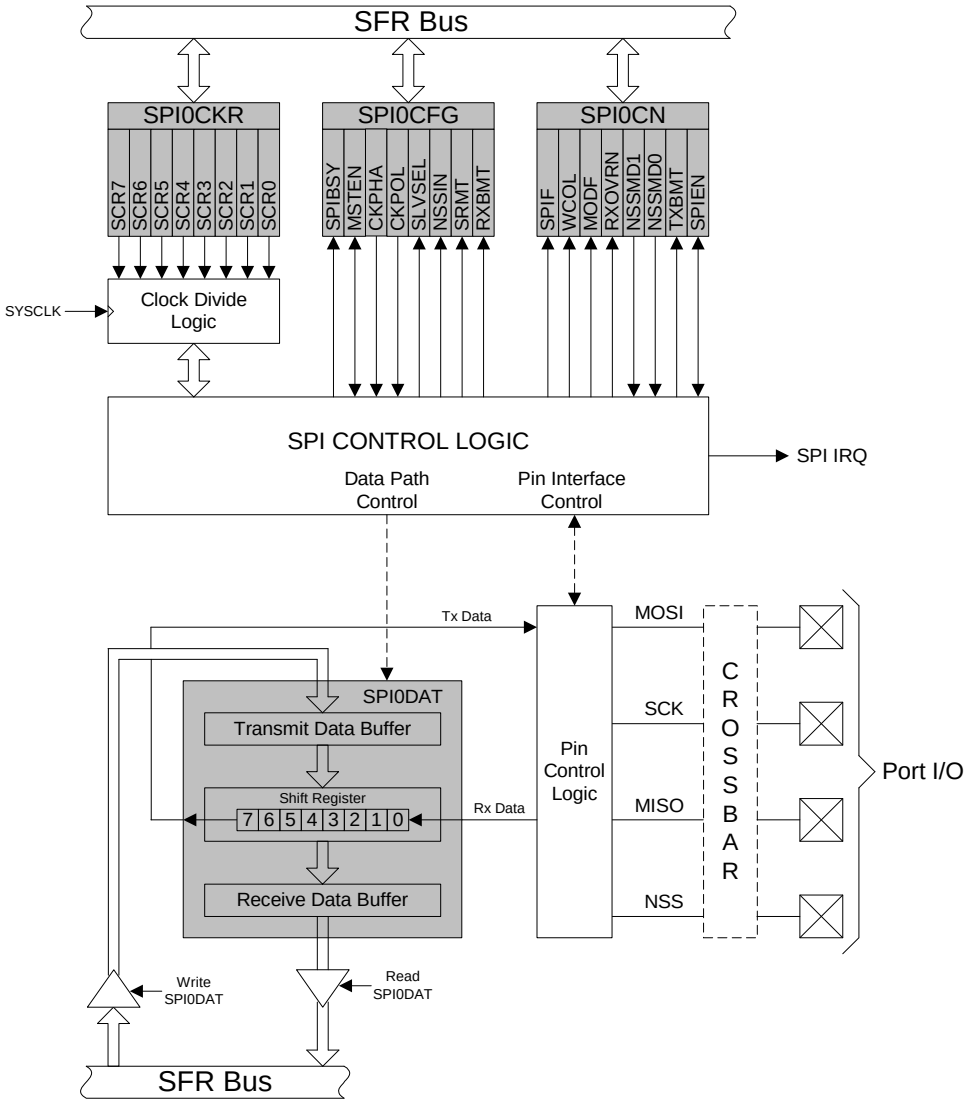


Figure 31.1. SPI Block Diagram

31.4. SPI0 Interrupt Sources

When SPI0 interrupts are enabled, the following four flags will generate an interrupt when they are set to logic 1:

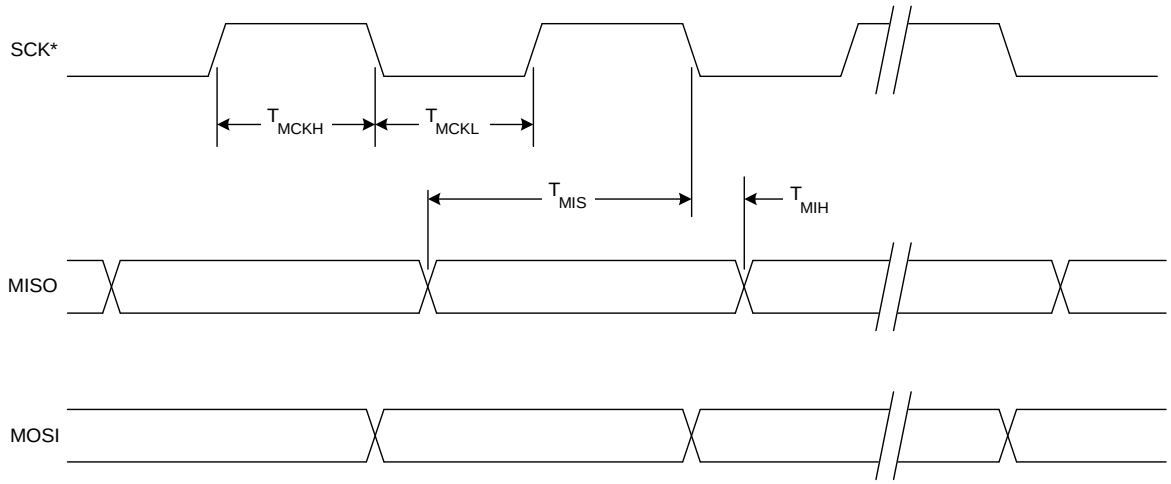
All of the following bits must be cleared by software.

- The SPI Interrupt Flag, SPIF (SPI0CN.7) is set to logic 1 at the end of each byte transfer. This flag can occur in all SPI0 modes.
- The Write Collision Flag, WCOL (SPI0CN.6) is set to logic 1 if a write to SPI0DAT is attempted when the transmit buffer has not been emptied to the SPI shift register. When this occurs, the write to SPI0DAT will be ignored, and the transmit buffer will not be written.This flag can occur in all SPI0 modes.
- The Mode Fault Flag MODF (SPI0CN.5) is set to logic 1 when SPI0 is configured as a master, and for multi-master mode and the NSS pin is pulled low. When a Mode Fault occurs, the MSTEN and SPIEN bits in SPI0CN are set to logic 0 to disable SPI0 and allow another master device to access the bus.
- The Receive Overrun Flag RXOVRN (SPI0CN.4) is set to logic 1 when configured as a slave, and a transfer is completed and the receive buffer still holds an unread byte from a previous transfer. The new byte is not transferred to the receive buffer, allowing the previously received data byte to be read. The data byte which caused the overrun is lost.

31.5. Serial Clock Phase and Polarity

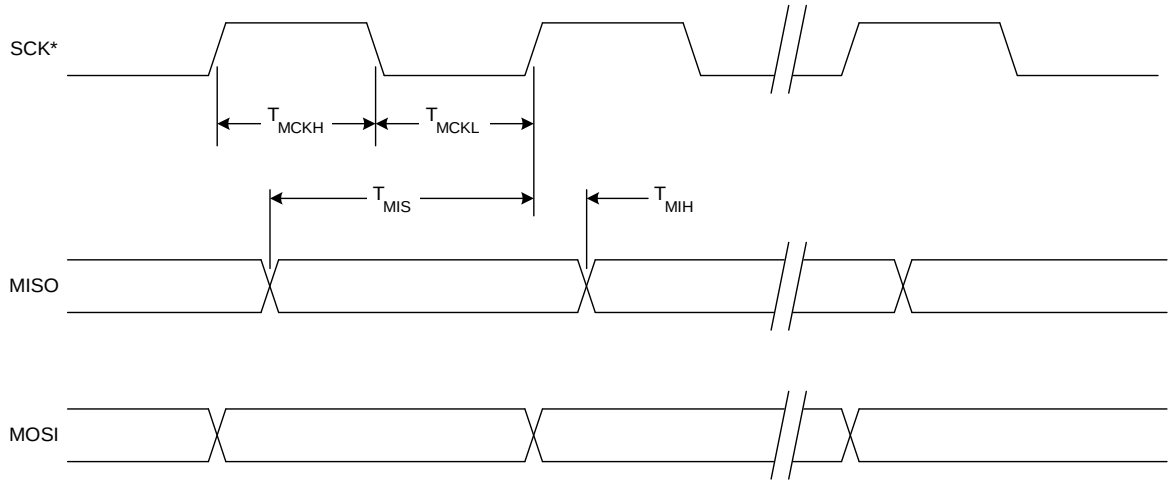
Four combinations of serial clock phase and polarity can be selected using the clock control bits in the SPI0 Configuration Register (SPI0CFG). The CKPHA bit (SPI0CFG.5) selects one of two clock phases (edge used to latch the data). The CKPOL bit (SPI0CFG.4) selects between an active-high or active-low clock. Both master and slave devices must be configured to use the same clock phase and polarity. SPI0 should be disabled (by clearing the SPIEN bit, SPI0CN.0) when changing the clock phase or polarity. The clock and data line relationships for master mode are shown in Figure 31.5. For slave mode, the clock and data relationships are shown in Figure 31.6 and Figure 31.7. CKPHA should be set to 0 on both the master and slave SPI when communicating between two Silicon Labs C8051 devices.

The SPI0 Clock Rate Register (SPI0CKR) as shown in SFR Definition 31.3 controls the master mode serial clock frequency. This register is ignored when operating in slave mode. When the SPI is configured as a master, the maximum data transfer rate (bits/sec) is one-half the system clock frequency or 12.5 MHz, whichever is slower. When the SPI is configured as a slave, the maximum data transfer rate (bits/sec) for full-duplex operation is 1/10 the system clock frequency, provided that the master issues SCK, NSS (in 4-wire slave mode), and the serial input data synchronously with the slave's system clock. If the master issues SCK, NSS, and the serial input data asynchronously, the maximum data transfer rate (bits/sec) must be less than 1/10 the system clock frequency. In the special case where the master only wants to transmit data to the slave and does not need to receive data from the slave (i.e., half-duplex operation), the SPI slave can receive data at a maximum data transfer rate (bits/sec) of 1/4 the system clock frequency. This is provided that the master issues SCK, NSS, and the serial input data synchronously with the slave's system clock.



* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

Figure 31.8. SPI Master Timing (CKPHA = 0)



* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

Figure 31.9. SPI Master Timing (CKPHA = 1)

32.2.2. 9-Bit UART

9-bit UART mode uses a total of eleven bits per data byte: a start bit, 8 data bits (LSB first), a programmable ninth data bit, and a stop bit. The state of the ninth transmit data bit is determined by the value in TB80 (SCON0.3), which is assigned by user software. It can be assigned the value of the parity flag (bit P in register PSW) for error detection, or used in multiprocessor communications. On receive, the ninth data bit goes into RB80 (SCON0.2) and the stop bit is ignored.

Data transmission begins when an instruction writes a data byte to the SBUF0 register. The TI0 Transmit Interrupt Flag (SCON0.1) is set at the end of the transmission (the beginning of the stop-bit time). Data reception can begin any time after the REN0 Receive Enable bit (SCON0.4) is set to 1. After the stop bit is received, the data byte will be loaded into the SBUF0 receive register if the following conditions are met: (1) RI0 must be logic 0, and (2) if MCE0 is logic 1, the 9th bit must be logic 1 (when MCE0 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 SBUF0, the ninth bit is stored in RB80, and the RI0 flag is set to 1. If the above conditions are not met, SBUF0 and RB80 will not be loaded and the RI0 flag will not be set to 1. A UART0 interrupt will occur if enabled when either TI0 or RI0 is set to 1.

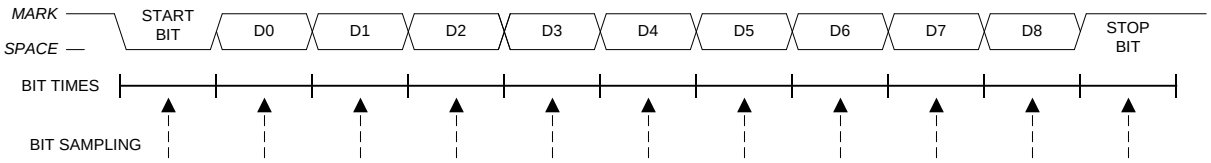


Figure 32.5. 9-Bit UART Timing Diagram

33.2.2. 8-bit Timers with Auto-Reload

When T2SPLIT is set, Timer 2 operates as two 8-bit timers (TMR2H and TMR2L). Both 8-bit timers operate in auto-reload mode as shown in Figure 33.5. TMR2RLL holds the reload value for TMR2L; TMR2RLH holds the reload value for TMR2H. The TR2 bit in TMR2CN handles the run control for TMR2H. TMR2L is always running when configured for 8-bit Mode.

Each 8-bit timer may be configured to use SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. The Timer 2 Clock Select bits (T2MH and T2ML in CKCON) select either SYSCLK or the clock defined by the Timer 2 External Clock Select bit (T2XCLK in TMR2CN), as follows:

T2MH	T2XCLK	TMR2H Clock Source
0	0	SYSCLK / 12
0	1	External Clock / 8
1	X	SYSCLK

T2ML	T2XCLK	TMR2L Clock Source
0	0	SYSCLK / 12
0	1	External Clock / 8
1	X	SYSCLK

The TF2H bit is set when TMR2H overflows from 0xFF to 0x00; the TF2L bit is set when TMR2L overflows from 0xFF to 0x00. When Timer 2 interrupts are enabled (IE.5), an interrupt is generated each time TMR2H overflows. If Timer 2 interrupts are enabled and TF2LEN (TMR2CN.5) is set, an interrupt is generated each time either TMR2L or TMR2H overflows. When TF2LEN is enabled, software must check the TF2H and TF2L flags to determine the source of the Timer 2 interrupt. The TF2H and TF2L interrupt flags are not cleared by hardware and must be manually cleared by software.

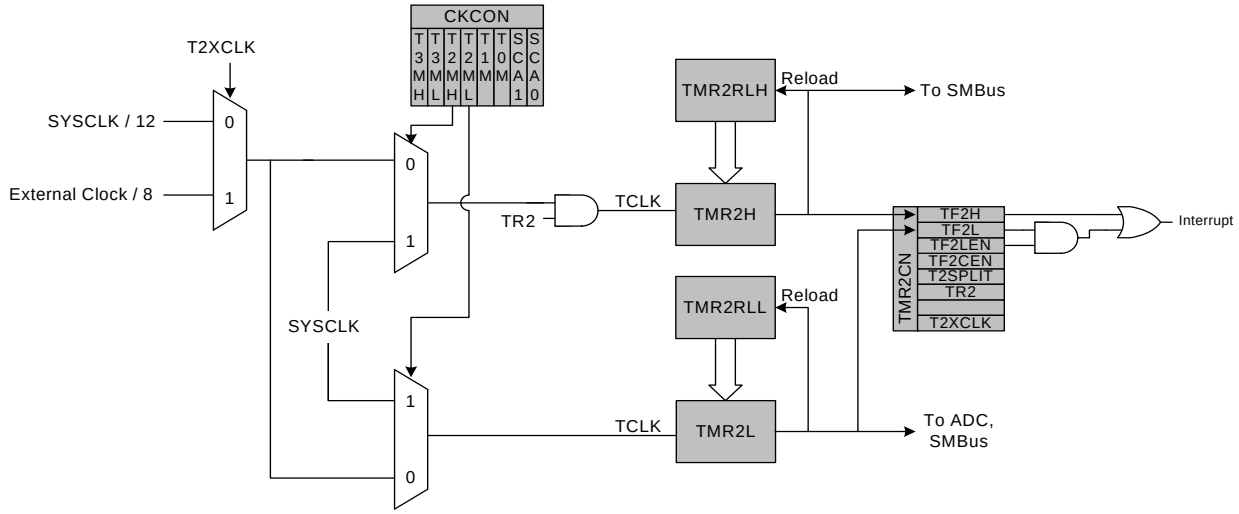


Figure 33.5. Timer 2 8-Bit Mode Block Diagram