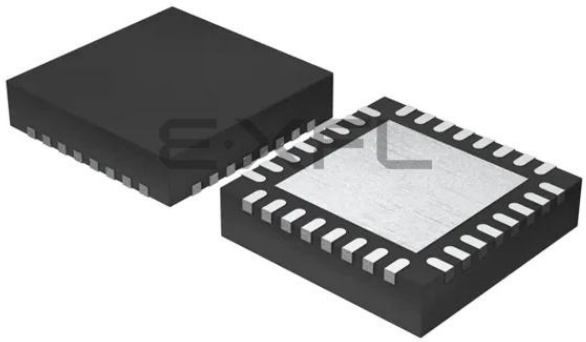


Welcome to E-XFL.COM

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	Not For New Designs
Core Processor	CIP-51™
Core Size	8-Bit
Speed	25MHz
Connectivity	I²C/SMBus, I²C Slave, SPI, UART/USART
Peripherals	CapSense, DMA, POR, PWM, WDT
Number of I/O	28
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 28x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°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/c8051f971-a-gm

5. QFN-48 Package Specifications

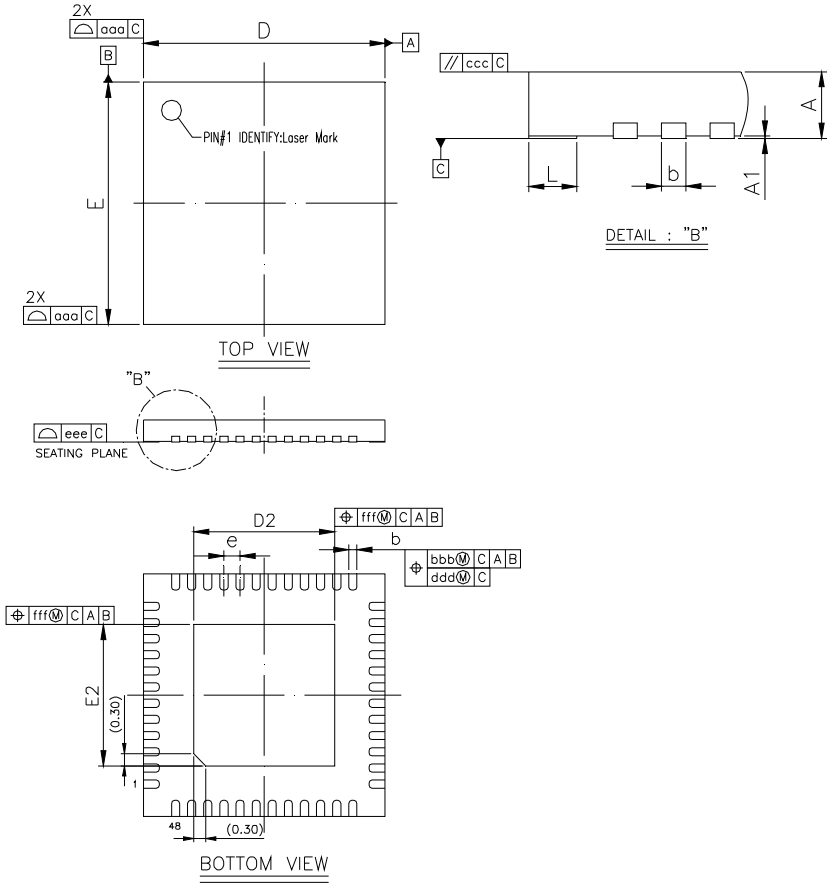


Figure 5.1. QFN-48 Package Drawing (6x6 mm)

Table 5.1. QFN-48 Package Dimensions

Dimension	Min	Typ	Max	Dimension	Min	Typ	Max
A	0.50	0.55	0.60	D2	3.35	3.50	3.65
A1	0.00	0.02	0.05	L	0.30	0.40	0.50
b	0.15	0.20	0.25	aaa	0.10		
D	6.00 BSC			bbb	0.07		
D2	3.35	3.50	3.65	ccc	0.10		
e	0.40 BSC			ddd	0.05		
E	6.00 BSC			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 JEDEC outline MO-220.
4. Recommended card reflow profile is per the JEDEC/IPC J-STD-020 specification for Small Body Components.

8.1. Program Memory

The CIP-51 core has a 64 kB program memory space. The C8051F97x family implements 32 kB, or 16 kB of this program memory space as in-system, re-programmable flash memory. The last address in the flash block (0x7FFF on 32 kB devices and 0x3FFF on 16 kB devices) serves as a security lock byte for the device, and provides read, write and erase protection. Addresses above the lock byte within the 64 kB address space are reserved.

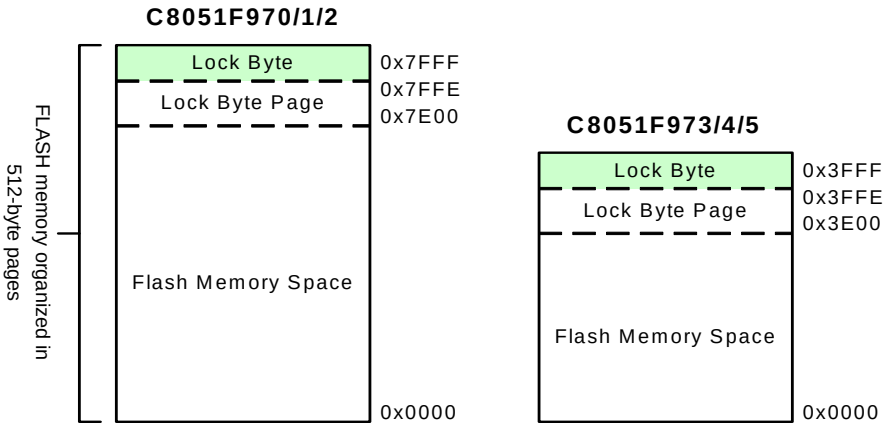


Figure 8.2. Flash Program Memory Map

8.1.1. MOVX Instruction and Program Memory

The MOVX instruction in an 8051 device is typically used to access external data memory. On the C8051F97x 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 C8051F97x to update program code and use the program memory space for non-volatile data storage. Refer to Section “10. Flash Memory” on page 65 for further details.

8.2. Data Memory

The C8051F97x device family includes up to 512 bytes of RAM data memory. 256 bytes of this memory is mapped into the internal RAM space of the 8051. On devices with 512 bytes total RAM, 256 additional bytes of memory are available as on-chip “external” memory. The data memory map is shown in Figure 8.1 for reference.

8.2.1. Internal RAM

There are 256 bytes of internal RAM mapped into the data memory space from 0x00 through 0xFF. The lower 128 bytes of data memory are used for general purpose registers and scratch pad memory. Either direct or indirect addressing may be used to access the lower 128 bytes of data memory. Locations 0x00 through 0x1F are addressable as four banks of general purpose registers, each bank consisting of eight byte-wide registers. The next 16 bytes, locations 0x20 through 0x2F, may either be addressed as bytes or as 128 bit locations accessible with the direct addressing mode.

The upper 128 bytes of data memory are accessible only by indirect addressing. This region occupies the same address space as the Special Function Registers (SFR) but is physically separate from the SFR space. The addressing mode used by an instruction when accessing locations above 0x7F determines whether the CPU accesses the upper 128 bytes of data memory space or the SFRs. Instructions that use direct addressing will access the SFR space. Instructions using indirect addressing above 0x7F access the upper 128 bytes of data memory. Figure 8.1 illustrates the data memory organization of the C8051F97x.

12. Device Identification and Unique Identifier

The C8051F97x has SFRs that identify the device family, derivative, and revision. These SFRs can be read by firmware at runtime to determine the capabilities of the MCU that is executing code. This allows the same firmware image to run on MCUs with different memory sizes and peripherals, and dynamically change functionality to suit the capabilities of that MCU.

In addition to the device identification registers, a 128-bit unique identifier (UID) is preprogrammed into all devices. The UID resides in the last sixteen bytes of XRAM. The UID can be read by firmware using MOVX instructions and through the debug port.

Firmware can overwrite the UID during normal operation, and the bytes in memory will be automatically reinitialized with the UID value after any device reset. Firmware using this area of memory should always initialize the memory to a known value, as any previous data stored at these locations will be overwritten and not retained through a reset.

Table 12.1. UID Implementation Information

Device	External Memory (XRAM) Addresses
C8051F970 C8051F971 C8051F972	**(MSB)** 0x1FFF, 0x1FFE, 0x1FFD, 0x1FFC, 0x1FFB, 0x1FFA, 0x1FF9, 0x1FF8, 0x1FF7, 0x1FF6, 0x1FF5, 0x1FF4, 0x1FF3, 0x1FF2, 0x1FF1, 0x1FF0 **(LSB)**
C8051F973 C8051F974 C8051F975	**(MSB)** 0x0FFF, 0x0FFE, 0x0FFD, 0x0FFC, 0x0FFB, 0x0FFA, 0x0FF9, 0x0FF8, 0x0FF7, 0x0FF6, 0x0FF5, 0x0FF4, 0x0FF3, 0x0FF2, 0x0FF1, 0x0FF0 **(LSB)**

16.10. Power Control Registers

Register 16.1. PCON: Power Control

Bit	7	6	5	4	3	2	1	0
Name	GF5	GF4	GF3	GF2	GF1	GF0	STOP	IDLE
Type	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0
SFR Page = ALL; SFR Address: 0x87								

Bit	Name	Function
7	GF5	**General Purpose Flag 5.** This flag is a general purpose flag for use under firmware control.
6	GF4	**General Purpose Flag 4.** This flag is a general purpose flag for use under firmware control.
5	GF3	**General Purpose Flag 3.** This flag is a general purpose flag for use under firmware control.
4	GF2	**General Purpose Flag 2.** This flag is a general purpose flag for use under firmware control.
3	GF1	**General Purpose Flag 1.** This flag is a general purpose flag for use under firmware control.
2	GF0	**General Purpose Flag 0.** This flag is a general purpose flag for use under firmware control.
1	STOP	**Stop Mode Select.** Setting this bit will place the CIP-51 in Stop mode. This bit will always be read as 0.
0	IDLE	**Idle Mode Select.** Setting this bit will place the CIP-51 in Idle mode. This bit will always be read as 0.

18.11. CS0 Conversion Accumulator

CS0 can be configured to accumulate multiple conversions on an input channel. The number of samples to be accumulated is configured using the CS0ACU2:0 bits (CS0CF2:0). The accumulator can accumulate 1, 4, 8, 16, 32, or 64 samples. After the defined number of samples have been accumulated, the result is divided by either 1, 4, 8, 16, 32, or 64 (depending on the CS0ACU[2:0] setting) and copied to the CS0DH:CS0DL SFRs.

Table 18.1. Operation with Auto-Scan and Accumulate

Auto-Scan Enabled	Accumulator Enabled	CS0 Conversion Complete Interrupt Behavior	CS0 Greater Than Interrupt Behavior	CS0MX Behavior
N	N	CS0INT Interrupt serviced after 1 conversion completes	Interrupt serviced after 1 conversion completes if value in CS0DH:CS0DL is greater than CS0THH:CS0THL	CS0MX unchanged.
N	Y	CS0INT Interrupt serviced after *M* conversions complete	Interrupt serviced after *M* conversions complete if value in CS0DH:CS0DL (post accumulate and divide) is greater than CS0THH:CS0THL	CS0MX unchanged.
Y	N	CS0INT Interrupt serviced after 1 conversion completes	Interrupt serviced after conversion completes if value in CS0DH:CS0DL is greater than CS0THH:CS0THL; Auto-Scan stopped	If greater-than comparator detects conversion value is greater than CS0THH:CS0THL, CS0MX is left unchanged; otherwise, CS0MX updates to the next channel (CS0MX + 1) and wraps back to CS0SS after passing CS0SE.
Y	Y	CS0INT Interrupt serviced after *M* conversions complete	Interrupt serviced after *M* conversions complete if value in CS0DH:CS0DL (post accumulate and divide) is greater than CS0THH:CS0THL; Auto-Scan stopped	If greater-than comparator detects conversion value is greater than CS0THH:CS0THL, CS0MX is left unchanged; otherwise, CS0MX updates to the next channel (CS0MX + 1) and wraps back to CS0SS after passing CS0SE.

Note: M = Accumulator setting (1x, 4x, 8x, 16x, 32x, 64x).

Table 18.2. CS0 Input Multiplexer Channels (Continued)

ADC0MX Setting	Signal Name	QFN-48 Pin Name	QFN-32 Pin Name	QFN-24 Pin Name
010110	CS0.22	P2.6	P2.6	Reserved
010111	CS0.23	P2.7	P2.7	Reserved
011000	CS0.24	P3.0	P3.0	Reserved
011001	CS0.25	P3.1	P3.1	Reserved
011010	CS0.26	P3.2	P3.2	Reserved
011011	CS0.27	P3.3	Reserved	Reserved
011100	CS0.28	P3.4	Reserved	Reserved
011101	CS0.29	P3.5	Reserved	Reserved
011110	CS0.30	P3.6	Reserved	Reserved
011111	CS0.31	P3.7	Reserved	Reserved
100000	CS0.32	P4.0	Reserved	Reserved
100001	CS0.33	P4.1	Reserved	Reserved
100010	CS0.34	P4.2	Reserved	Reserved
100011	CS0.35	P4.3	Reserved	Reserved
100100	CS0.36	P4.4	Reserved	Reserved
100101	CS0.37	P4.5	Reserved	Reserved
100110	CS0.38	P4.6	Reserved	Reserved
100111	CS0.39	P4.7	Reserved	Reserved
101000	CS0.40	P5.0	Reserved	Reserved
101001	CS0.41	P5.1	Reserved	Reserved
101010	CS0.42	P5.2	P5.2	P5.2
101011-111111	Reserved	Reserved		

Register 18.3. CS0DH: Capacitive Sense 0 Data High Byte

Bit	7	6	5	4	3	2	1	0
Name	CS0DH							
Type	R							
Reset	0	0	0	0	0	0	0	0
SFR Page = 0x0; SFR Address: 0xEE								

Bit	Name	Function
7:0	CS0DH	**CS0 Data High Byte.** Stores the high byte of the last completed 16-bit Capacitive Sense conversion.

Register 21.8. DMA0NBAL: Memory Base Address Low

Bit	7	6	5	4	3	2	1	0
Name	NBAL							
Type	RW							
Reset	0	0	0	0	0	0	0	0
SFR Page = 0xF; SFR Address: 0xC9								

Bit	Name	Function
7:0	NBAL	**Memory Base Address Low.** This field sets low byte of the channel memory base address. This base address is the starting channel XRAM address if the channel's address offset DMA0NAO is reset to 0.
Note: This register is a DMA channel indirect register. Select the desired channel first using the DMA0SEL register.		

22.5. MCU Mode Operation

MCU mode operation for the MAC is enabled when there is no DMA channel enabled for transferring data to or from any MAC0 register. When MCU mode operation is active, the MAC inputs and results are transferred to or from XRAM via software access to the SFRs. During MCU mode operation, a MAC operation is triggered by software writing 1 to the BUSY bit in the MAC0STA register. When the MAC operation has completed, hardware clears the BUSY bit to 0. The typical sequence of MCU mode operations is as follows:

1. Firmware disables all MAC-specific DMA channels.
2. Firmware initializes the control registers (MAC0CF0, MAC0CF1, MAC0CF2, MAC0ITER) appropriately.
3. Firmware writes all the operand values:
   - Update MAC0A, MAC0B
   - Setup accumulator either by clearing it (via setting the CLRACC bit) or writing directly to MAC0ACC0-3 and MAC0OVR
4. Firmware writes 1 to the BUSY bit.
5. MAC0 completes the MAC operation in 1 SYSCLK cycle, clears BUSY bit to 0, and sets both the MACINT and ACCRDY bits to 1.

22.6. DMA Mode Operation

DMA mode operation is a powerful mode that can be used process large array of data using the MAC0 module. Alternatively, it can be used to implement digital filters efficiently. DMA mode operation for the MAC0 is enabled when there is at least one DMA channel enabled for transferring data to or from any MAC0 register.

During DMA mode operation, the BUSY bit must be set to 1 to generate DMA requests. The complete flowchart of DMA mode operation is given in Figure 22.4.

22.11.3. Initializing Memory Block Using DMA0 and MAC0

This example demonstrates a sophisticated example of initializing a block of memory in XRAM with value 0x55.

```
// Memory block to initialize
volatile SEGMENT_VARIABLE(memblock[500], U8, SEG_XDATA);

SFRPAGE = DMA0_PAGE;    // Change Page register to DMA0 Page
DMA0SEL = 0;            // Select DMA channel 0
DMA0EN &= ~0x01;        // Disable DMA channel 0
DMA0INT &= ~0x01;       // Clear interrupt bit for channel 0
DMA0NCF = 0x05;         // Select MAC0 Accumulator to XRAM transfer
DMA0NCF |= (LSB << 4);  // Use LSB to spec endian bit for compiler independence
DMA0NBA = (U16)&memblock[0];  // XRAM base address
DMA0NAO = 0;                  // XRAM offset
DMA0NSZ = 500;                // Transfer 500 bytes
DMA0EN |= 0x01;         // Enable DMA channel 0

// No need to change page register as MAC0 and DMA0 registers are in same page.
MAC0CF0 = 0x28;         // Clear accumulator, Multiply Only mode
MAC0CF1 = 0x0A;         // Select unsigned mode, constant A and constant B
MAC0CF2 = 0x04;         // Disable rounding, saturation and alignment logic
                        // 2-byte DMA transaction from MAC0 Accumulator to XRAM
MAC0A = 1;              // Load MAC0A with 1 decimal
MAC0B = 0x5555;         // Load MAC0B with 0x5555 hexadecimal
MAC0ITER = 1;           // Set to 1 iteration
MAC0STA = 1;            // Set BUSY to start MAC operation

while (!(DMA0INT & 1));// Poll for DMA Channel 0 completion interrupt

// All 500 bytes in memblock[] will be initialized to 0x55
```

Register 22.8. MAC0BH: Operand B High Byte

Bit	7	6	5	4	3	2	1	0
Name	MAC0BH							
Type	RW							
Reset	0	0	0	0	0	0	0	0
SFR Page = 0xF; SFR Address: 0xAF								

Bit	Name	Function
7:0	MAC0BH	**MAC0 B High Byte.** This field is the upper 8 bits of the MAC0 B input.

Register 26.7. P0MDOUT: Port 0 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	B7	B6	B5	B4	B3	B2	B1	B0
Type	RW	RW	RW	RW	RW	RW	RW	RW
Reset	0	0	0	0	0	0	0	0
SFR Page = 0xF; SFR Address: 0xD9								

Table 26.10. P0MDOUT Register Bit Descriptions

Bit	Name	Function
7	B7	**Port 0 Bit 7 Output Mode.** 0: P0.7 output is open-drain. 1: P0.7 output is push-pull.
6	B6	**Port 0 Bit 6 Output Mode.** 0: P0.6 output is open-drain. 1: P0.6 output is push-pull.
5	B5	**Port 0 Bit 5 Output Mode.** 0: P0.5 output is open-drain. 1: P0.5 output is push-pull.
4	B4	**Port 0 Bit 4 Output Mode.** 0: P0.4 output is open-drain. 1: P0.4 output is push-pull.
3	B3	**Port 0 Bit 3 Output Mode.** 0: P0.3 output is open-drain. 1: P0.3 output is push-pull.
2	B2	**Port 0 Bit 2 Output Mode.** 0: P0.2 output is open-drain. 1: P0.2 output is push-pull.
1	B1	**Port 0 Bit 1 Output Mode.** 0: P0.1 output is open-drain. 1: P0.1 output is push-pull.
0	B0	**Port 0 Bit 0 Output Mode.** 0: P0.0 output is open-drain. 1: P0.0 output is push-pull.

Register 26.13. P1MDIN: Port 1 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	B7	B6	B5	B4	B3	B2	B1	B0
Type	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1
SFR Page = 0xF; SFR Address: 0xED								

Table 26.16. P1MDIN Register Bit Descriptions

Bit	Name	Function
7	B7	**Port 1 Bit 7 Input Mode.** 0: P1.7 pin is configured for analog mode. 1: P1.7 pin is configured for digital mode.
6	B6	**Port 1 Bit 6 Input Mode.** 0: P1.6 pin is configured for analog mode. 1: P1.6 pin is configured for digital mode.
5	B5	**Port 1 Bit 5 Input Mode.** 0: P1.5 pin is configured for analog mode. 1: P1.5 pin is configured for digital mode.
4	B4	**Port 1 Bit 4 Input Mode.** 0: P1.4 pin is configured for analog mode. 1: P1.4 pin is configured for digital mode.
3	B3	**Port 1 Bit 3 Input Mode.** 0: P1.3 pin is configured for analog mode. 1: P1.3 pin is configured for digital mode.
2	B2	**Port 1 Bit 2 Input Mode.** 0: P1.2 pin is configured for analog mode. 1: P1.2 pin is configured for digital mode.
1	B1	**Port 1 Bit 1 Input Mode.** 0: P1.1 pin is configured for analog mode. 1: P1.1 pin is configured for digital mode.
0	B0	**Port 1 Bit 0 Input Mode.** 0: P1.0 pin is configured for analog mode. 1: P1.0 pin is configured for digital mode.
Note: Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled.		

Register 26.28. P4: Port 4 Pin Latch

Bit	7	6	5	4	3	2	1	0
Name	B7	B6	B5	B4	B3	B2	B1	B0
Type	RW	RW	RW	RW	RW	RW	RW	RW
Reset	1	1	1	1	1	1	1	1
SFR Page = 0x0; SFR Address: 0xE2								

Table 26.31. P4 Register Bit Descriptions

Bit	Name	Function
7	B7	**Port 4 Bit 7 Latch.** 0: P4.7 is low. Set P4.7 to drive low. 1: P4.7 is high. Set P4.7 to drive or float high.
6	B6	**Port 4 Bit 6 Latch.** 0: P4.6 is low. Set P4.6 to drive low. 1: P4.6 is high. Set P4.6 to drive or float high.
5	B5	**Port 4 Bit 5 Latch.** 0: P4.5 is low. Set P4.5 to drive low. 1: P4.5 is high. Set P4.5 to drive or float high.
4	B4	**Port 4 Bit 4 Latch.** 0: P4.4 is low. Set P4.4 to drive low. 1: P4.4 is high. Set P4.4 to drive or float high.
3	B3	**Port 4 Bit 3 Latch.** 0: P4.3 is low. Set P4.3 to drive low. 1: P4.3 is high. Set P4.3 to drive or float high.
2	B2	**Port 4 Bit 2 Latch.** 0: P4.2 is low. Set P4.2 to drive low. 1: P4.2 is high. Set P4.2 to drive or float high.
1	B1	**Port 4 Bit 1 Latch.** 0: P4.1 is low. Set P4.1 to drive low. 1: P4.1 is high. Set P4.1 to drive or float high.
0	B0	**Port 4 Bit 0 Latch.** 0: P4.0 is low. Set P4.0 to drive low. 1: P4.0 is high. Set P4.0 to drive or float high.

Notes:
1. Writing this register sets the port latch logic value for the associated I/O pins configured as digital I/O.
2. Reading this register returns the logic value at the pin, regardless if it is configured as output or input.

29.3.5. SCL High (SMBus Free) Timeout

The SMBus specification stipulates that if the SCL and SDA lines remain high for more that 50 µs, the bus is designated as free. When the SMB0FTE bit in SMB0CF is set, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods (as defined by the timer configured for the SMBus clock source). If the SMBus is waiting to generate a Master START, the START will be generated following this timeout. A clock source is required for free timeout detection, even in a slave-only implementation.

29.4. Using the SMBus

The SMBus can operate in both Master and Slave modes. The interface provides timing and shifting control for serial transfers; higher level protocol is determined by user software. The SMBus interface provides the following application-independent features:

- Byte-wise serial data transfers
- Clock signal generation on SCL (Master Mode only) and SDA data synchronization
- Timeout/bus error recognition, as defined by the SMB0CF configuration register
- START/STOP timing, detection, and generation
- Bus arbitration
- Interrupt generation
- Status information
- Optional hardware recognition of slave address and automatic acknowledgement of address/data

SMBus interrupts are generated for each data byte or slave address that is transferred. When hardware acknowledgement is disabled, the point at which the interrupt is generated depends on whether the hardware is acting as a data transmitter or receiver. When a transmitter (i.e., sending address/data, receiving an ACK), this interrupt is generated after the ACK cycle so that software may read the received ACK value; when receiving data (i.e., receiving address/data, sending an ACK), this interrupt is generated before the ACK cycle so that software may define the outgoing ACK value. If hardware acknowledgement is enabled, these interrupts are always generated after the ACK cycle. See Section 29.5 for more details on transmission sequences.

Interrupts are also generated to indicate the beginning of a transfer when a master (START generated), or the end of a transfer when a slave (STOP detected). Software should read the SMB0CN (SMBus Control register) to find the cause of the SMBus interrupt. Table 29.5 provides a quick SMB0CN decoding reference.

29.4.1. SMBus Configuration Register

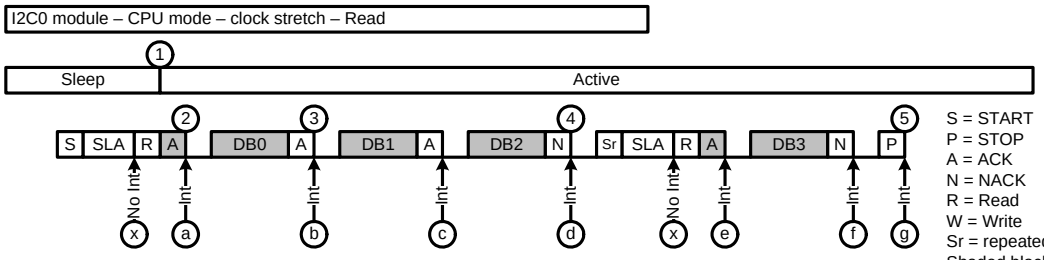
The SMBus Configuration register (SMB0CF) is used to enable the SMBus Master and/or Slave modes, select the SMBus clock source, and select the SMBus timing and timeout options. When the ENSMB bit is set, the SMBus is enabled for all master and slave events. Slave events may be disabled by setting the INH bit. With slave events inhibited, the SMBus interface will still monitor the SCL and SDA pins; however, the interface will NACK all received addresses and will not generate any slave interrupts. When the INH bit is set, all slave events will be inhibited following the next START (interrupts will continue for the duration of the current transfer).

Table 29.1. SMBus Clock Source Selection

SMBCS	SMBus0 Clock Source
00	Timer 0 Overflow
01	Timer 1 Overflow
10	Timer 2 High Byte Overflow
11	Timer 2 Low Byte Overflow

30.5.2. I²C Read Sequence (CPU mode)

Figure 30.6 shows the details of how the I2C0STAT status bits change during an I²C Read data transfer.



1. SLA+R wakes CPU from sleep. SLA is ACK if BUSY was 0.
2. INT generated. CPU writes DB0 into I2C0DOUT. SCL is held until CPU clears I2C0INT. CPU clears I2C0INT, releasing SCL
3. SCL held low; NACK bit reads '0'; CPU writes DB1 into I2C0DOUT and clears I2C0INT
4. NACK switches I2C slave to IDLE state and generates INT. Any following DB are ignored. If RpStart follows after NACK, the START sticky bit will be set, but no interrupt.
5. STOP generates interrupt. No Clock Stretch. CPU clears I2C0INT

No int (x) I2C0STAT = x1001000 at 8th SCL rising edge

I2C0 int (a) I2C0STAT = x1101001; CPU clears START. CPU writes DB0 into I2C0DOUT and clears I2C0INT

I2C0 int (b) I2C0STAT = x1100001; CPU writes DB1 into I2C0DOUT and clears I2C0INT

I2C0 int (c) I2C0STAT = x1100001; CPU writes DB2 into I2C0DOUT and clears I2C0INT

I2C0 int (d) I2C0STAT = x0110001; CPU prepares for STOP or RpStart and clears I2C0INT and clears NACK

I2C0 int (e) I2C0STAT = x1101001; CPU clears START. CPU writes DB3 into I2C0DOUT and clears I2C0INT

I2C0 int (f) I2C0STAT = x0110001; CPU prepares for STOP or RpStart and clears I2C0INT and clears NACK

I2C0 int (g) I2C0STAT = x0100100; CPU clears STOP and clears I2C0INT

* At a, b, c, d, e, f : Bits are set/cleared at 9th SCL falling edge. CPU clears I2C0INT to release SCL

Figure 30.6. Typical I2C Read Sequence in CPU Mode

Note that the I²C Master MUST always generate a NACK before it can generate a repeated START bit or a STOP bit. This is because the NACK will cause the I²C Slave to release the SDA line for the I²C Master to generate the START or STOP bit.

30.5.3. I²C Write Sequence (DMA mode)

Figure 30.7 shows the details of how the I2C0STAT status bits change during an I²C Write data transfer.

Register 32.11. TMR2L: Timer 2 Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TMR2L							
Type	RW							
Reset	0	0	0	0	0	0	0	0
SFR Page = 0x0; SFR Address: 0xCC								

Table 32.13. TMR2L Register Bit Descriptions

Bit	Name	Function
7:0	TMR2L	**Timer 2 Low Byte.** In 16-bit mode, the TMR2L register contains the low byte of the 16-bit Timer 2. In 8-bit mode, TMR2L contains the 8-bit low byte timer value.

33.1. PCA Counter/Timer

The 16-bit PCA counter/timer consists of two 8-bit SFRs: PCA0L and PCA0H. PCA0H is the high byte (MSB) of the 16-bit counter/timer and PCA0L is the low byte (LSB). Reading PCA0L automatically latches the value of PCA0H into a “snapshot” register; the following PCA0H read accesses this “snapshot” register. **Reading the PCA0L Register first guarantees an accurate reading of the entire 16-bit PCA0 counter.** Reading PCA0H or PCA0L does not disturb the counter operation. The CPS2–CPS0 bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 33.1.

When the counter/timer overflows from 0xFFFF to 0x0000, the Counter Overflow Flag (CF) in PCA0MD is set to logic 1 and an interrupt request is generated if CF interrupts are enabled. Setting the ECF bit in PCA0MD to logic 1 enables the CF flag to generate an interrupt request. The CF bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Clearing the CIDL bit in the PCA0MD register allows the PCA to continue normal operation while the CPU is in Idle mode.

Table 33.1. PCA Timebase Input Options

CPS2	CPS1	CPS0	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 ECI (max rate = system clock divided by 4)
1	0	0	System clock
1	0	1	External oscillator source divided by 8[1]
1	1	0	SmaRTClock oscillator source divided by 8[2]
1	1	1	Reserved

Notes:
1. External oscillator source divided by 8 is synchronized with the system clock.
2. SmaRTClock oscillator source divided by 8 is synchronized with the system clock.

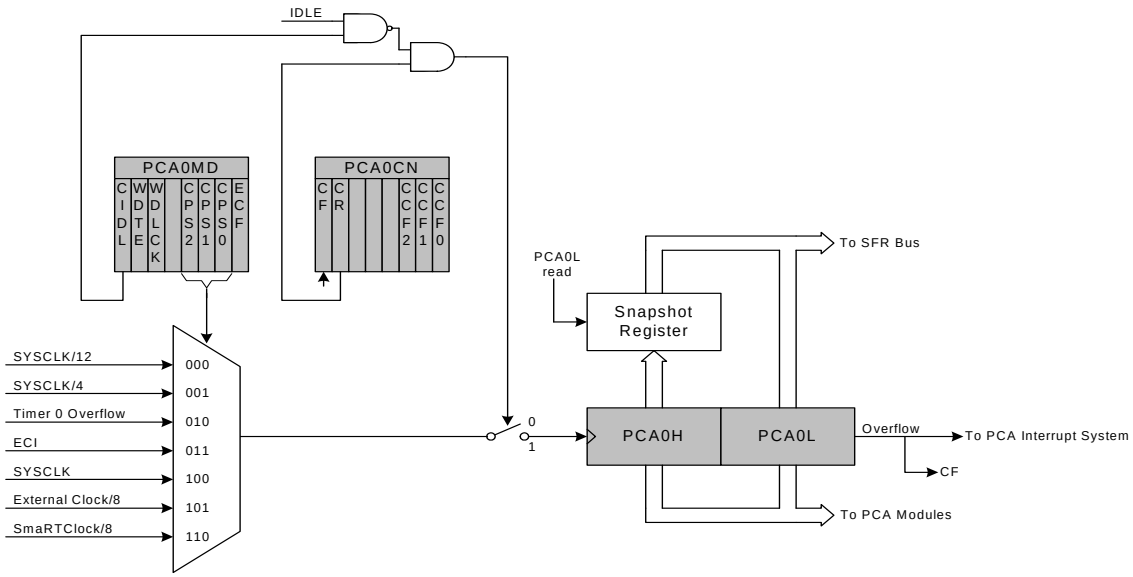


Figure 33.2. PCA Counter/Timer Block Diagram

33.2. PCA0 Interrupt Sources

Figure 33.3 shows a diagram of the PCA interrupt tree. There are five independent event flags that can be used to generate a PCA0 interrupt. They are: the main PCA counter overflow flag (CF), which is set upon a 16-bit overflow of the PCA0 counter, an intermediate overflow flag (COVF), which can be set on an overflow from the 8th, 9th, 10th, or 11th bit of the PCA0 counter, and the individual flags for each PCA channel (CCF0, CCF1, and CCF2), which are set according to the operation mode of that module. These event flags are always set when the trigger condition occurs. Each of these flags can be individually selected to generate a PCA0 interrupt, using the corresponding interrupt enable flag (ECF for CF, ECOV for COVF, and ECCFn for each CCFn). PCA0 interrupts must be globally enabled before any individual interrupt sources are recognized by the processor. PCA0 interrupts are globally enabled by setting the EA bit and the EPCA0 bit to logic 1.

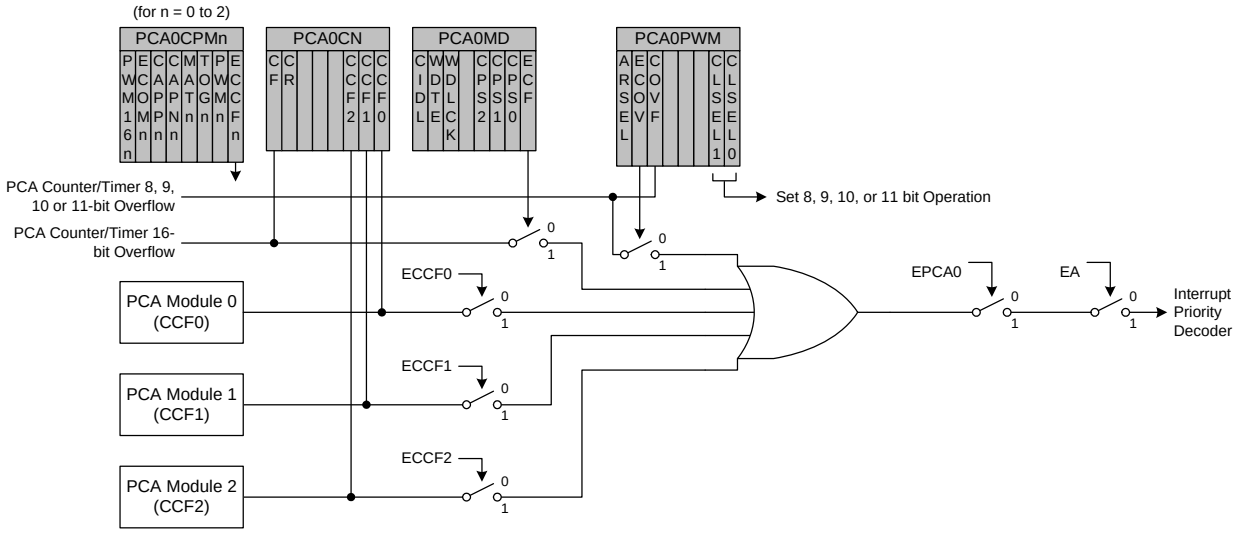


Figure 33.3. PCA Interrupt Block Diagram

$$Offset = (256 \times PCA0CPL2) + (256 - PCA0L)$$

Equation 33.5. Watchdog Timer Offset in PCA Clocks

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH2 and PCA0H. Software may force a WDT reset by writing a 1 to the CCF2 flag (PCA0CN.2) while the WDT is enabled.

33.4.2. Watchdog Timer Usage

To configure the WDT, perform the following tasks:

1. Disable the WDT by writing a 0 to the WDTE bit.
2. Select the desired PCA clock source (with the CPS2–CPS0 bits).
3. Load PCA0CPL2 with the desired WDT update offset value.
4. Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
5. Enable the WDT by setting the WDTE bit to 1.
6. Reset the WDT timer by writing to PCA0CPH2.

The PCA clock source and Idle mode select cannot be changed while the WDT is enabled. The watchdog timer is enabled by setting the WDTE or WDLCK bits in the PCA0MD register. When WDLCK is set, the WDT cannot be disabled until the next system reset. If WDLCK is not set, the WDT is disabled by clearing the WDTE bit.

The WDT is enabled following any reset. The PCA0 counter clock defaults to the system clock divided by 12, PCA0L defaults to 0x00, and PCA0CPL2 defaults to 0x00. Using Equation 33.5, this results in a WDT timeout interval of 256 PCA clock cycles, or 3072 system clock cycles. Table 33.3 lists some example timeout intervals for typical system clocks.

Table 33.3. Watchdog Timer Timeout Intervals[1]

System Clock (Hz)	PCA0CPL2	Timeout Interval (ms)
24,500,000	255	32.1
24,500,000	128	16.2
24,500,000	32	4.1
3,062,500[2]	255	257
3,062,500[2]	128	129.5
3,062,500[2]	32	33.1
32,000	255	24576
32,000	128	12384
32,000	32	3168

Notes:

1. Assumes SYSCLK/12 as the PCA clock source, and a PCA0L value of 0x00 at the update time.
2. Internal SYSCLK reset frequency = Internal Oscillator divided by 8.