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	I²C, SPI, UART/USART
Peripherals	DMA, LCD, POR, PWM, WDT
Number of I/O	57
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 ~ 3.8V
Data Converters	A/D 16x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f960-a-gq

Table 8.1. CIP-51 Instruction Set Summary (Continued)

Mnemonic	Description	Bytes	Clock Cycles
ANL C, /bit	AND complement of direct bit to Carry	2	2
ORL C, bit	OR direct bit to carry	2	2
ORL C, /bit	OR complement of direct bit to Carry	2	2
MOV C, bit	Move direct bit to Carry	2	2
MOV bit, C	Move Carry to direct bit	2	2
JC rel	Jump if Carry is set	2	2/3
JNC rel	Jump if Carry is not set	2	2/3
JB bit, rel	Jump if direct bit is set	3	3/4
JNB bit, rel	Jump if direct bit is not set	3	3/4
JBC bit, rel	Jump if direct bit is set and clear bit	3	3/4
Program Branching			
ACALL addr11	Absolute subroutine call	2	3
LCALL addr16	Long subroutine call	3	4
RET	Return from subroutine	1	5
RETI	Return from interrupt	1	5
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (relative address)	2	3
JMP @A+DPTR	Jump indirect relative to DPTR	1	3
JZ rel	Jump if A equals zero	2	2/3
JNZ rel	Jump if A does not equal zero	2	2/3
CJNE A, direct, rel	Compare direct byte to A and jump if not equal	3	3/4
CJNE A, #data, rel	Compare immediate to A and jump if not equal	3	3/4
CJNE Rn, #data, rel	Compare immediate to Register and jump if not equal	3	3/4
CJNE @Ri, #data, rel	Compare immediate to indirect and jump if not equal	3	4/5
DJNZ Rn, rel	Decrement Register and jump if not zero	2	2/3
DJNZ direct, rel	Decrement direct byte and jump if not zero	3	3/4
NOP	No operation	1	1

10.6.2.2. 8-bit MOVX without Bank Select: EMI0CF[4:2] = 001 or 011

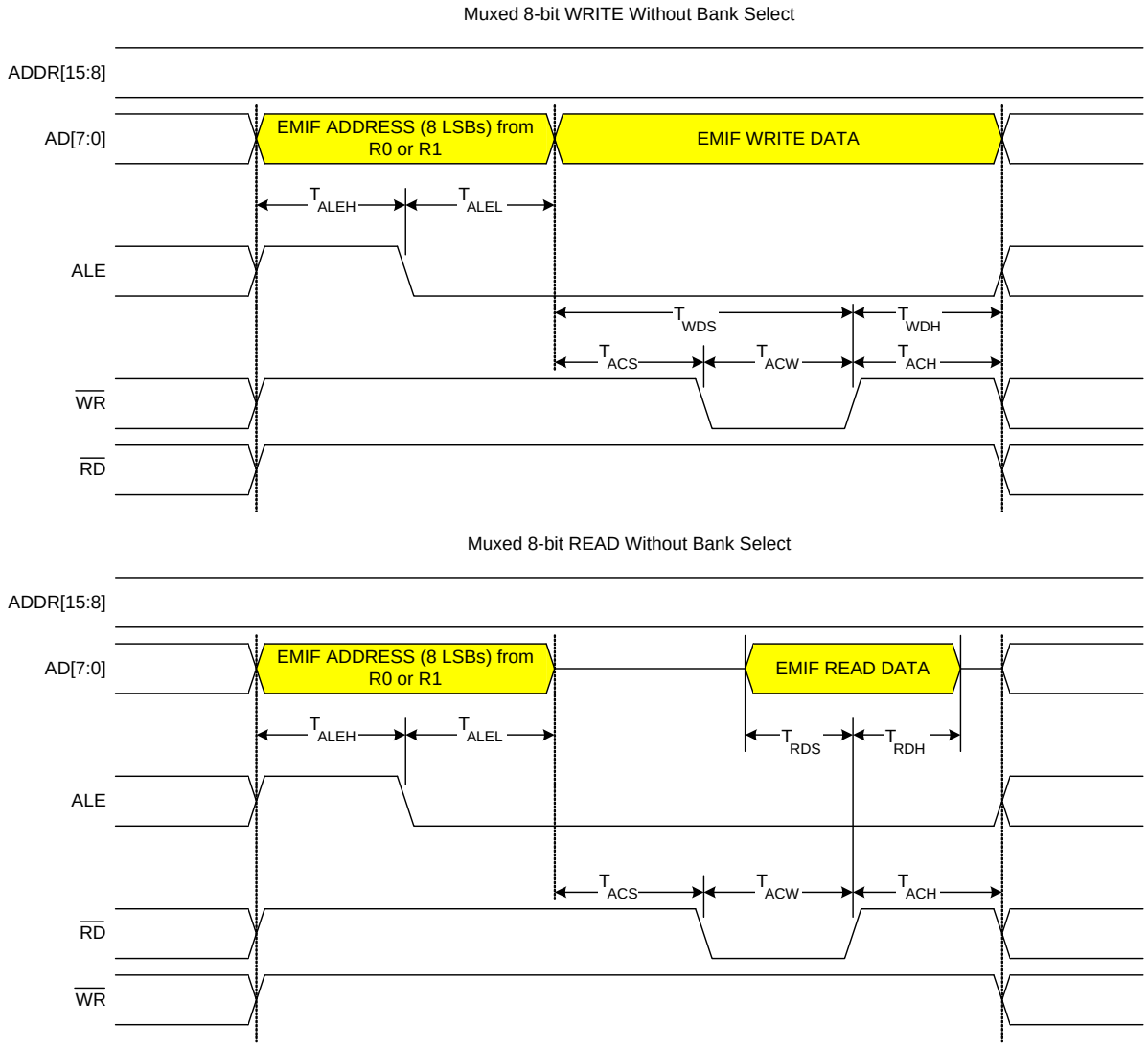

Figure 10.8. Multiplexed 8-bit MOVX without Bank Select Timing

14.6.4.1. CBC Decryption using DMA

Normally, the AES block is used with the DMA. This provides the best performance and lowest power consumption. Code examples are provided in 8051 compiler independent C code using the DMA. It is highly recommended to use with the code examples. The steps are documented in the datasheet for completeness.

- Prepare decryption Key, initialization vector, and data to be decrypted in xram.
- The initialization vector should be located immediately before the data to be decrypted to decrypt multiple blocks.
- Reset AES module by clearing bit 2 of AES0BCFG.
- Disable the first four DMA channels by clearing bits 0 to 3 in the DMA0EN sfr.
- Configure the first DMA channel for the AES0KIN sfr
  - Select the first DMA channel by writing 0x00 to the DMA0SEL sfr
  - Configure the first DMA channel to move xram to AES0KIN sfr by writing 0x05 to the DMA0NCF sfr
  - Write 0x01 to DMA0NMD to enable wrapping
  - Write the xram location of decryption key to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the key length in bytes to DMA0NSZL sfr
  - Clear the DMA0NSZH sfr
  - Clear the DMA0NAOH and DMA0NAOL sfrs
- Configure the second DMA channel for the AES0BIN sfr.
  - Select the second DMA channel by writing 0x01 to the DMA0SEL sfr.
  - Configure the second DMA channel to move xram to AES0BIN sfr by writing 0x06 to the DMA0NCF sfr.
  - Clear DMA0NMD to disable wrapping.
  - Write the xram address of the data to be decrypted to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be decrypted in multiples of 16 bytes to the DMA0NSZH and DMA0NSZL sfrs.
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- Configure the third DMA channel for the AES0XIN sfr.
  - Select the third DMA channel by writing 0x02 to the DMA0SEL sfr.
  - Configure the third DMA channel to move xram to AES0XIN sfr by writing 0x07 to the DMA0NCF sfr.
  - Clear DMA0NMD to disable wrapping.
  - Write the xram address of initialization vector to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be decrypted in multiples of 16 bytes to the DMA0NSZH and DMA0NSZL sfrs.
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- Configure the forth DMA channel for the AES0YOUT sfr
  - Select the forth channel by writing 0x03 to the DMA0SEL sfr
  - Configure the forth DMA channel to move the contents of the AES0YOUT sfr to xram by writing 0x08 to the DMA0NCF sfr
  - Enable transfer complete interrupt by setting bit 7 of DMA0NCF sfr
  - Clear DMA0NMD to disable wrapping
  - Write the xram address for decrypted data to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be decrypted in multiples of 16 bytes to the DMA0NSZH and DMA0NSZL sfrs.
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- Clear first four DMA interrupts by clearing bits 0 to 2 in the DMA0INT sfr.
- Enable first four DMA channels setting bits 0 to 2 in the DMA0EN sfr
- Configure the AES Module data flow for XOR on output data by writing 0x02 to the AES0DCFG sfr.
- Write key size to bits 1 and 0 of the AES0BCFG
- Configure the AES core for decryption by clearing bit 2 of AES0BCFG
- Initiate the decryption operation be setting bit 3 of AES0BCFG
- Wait on the DMA interrupt from DMA channel 3
- Disable the AES Module by clearing bit 2 of AES0BCFG
- Disable the DMA by writing 0x00 to DMA0EN

18.5. Flash Write and Erase Guidelines

Any system which contains routines which write or erase flash memory from software involves some risk that the write or erase routines will execute unintentionally if the CPU is operating outside its specified operating range of VDD, system clock frequency, or temperature. This accidental execution of flash modifying code can result in alteration of flash memory contents causing a system failure that is only recoverable by re-Flashing the code in the device.

To help prevent the accidental modification of flash by firmware, the VDD Monitor must be enabled and enabled as a reset source on C8051F96x devices for the flash to be successfully modified. **If either the VDD Monitor or the VDD Monitor reset source is not enabled, a Flash Error Device Reset will be generated when the firmware attempts to modify the flash.**

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

18.5.1. 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 $\overline{\text{RST}}$ pin of the device that holds the device in reset until VDD reaches the minimum device operating voltage and re-asserts $\overline{\text{RST}}$ 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.

Notes:
On C8051F96x 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 C8051F96x 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.

19.5. Suspend Mode

Setting the Suspend Mode Select bit (PMU0CF.6) causes the system clock to be gated off and all internal oscillators disabled. The system clock source must be set to the low power internal oscillator or the precision oscillator prior to entering Suspend Mode. All digital logic (timers, communication peripherals, interrupts, CPU, etc.) stops functioning until one of the enabled wake-up sources occurs.

The following wake-up sources can be configured to wake the device from Suspend Mode:

- Pulse Counter Count Reached Event
- VBAT Monitor (part of LCD logic)
- SmaRTClock Oscillator Fail
- SmaRTClock Alarm
- Port Match Event
- Comparator0 Rising Edge

Note: Upon wake-up from suspend mode, PMU0 requires two system clocks in order to update the PMU0CF wake-up flags. All flags will read back a value of '0' during the first two system clocks following a wake-up from suspend mode.

In addition, a noise glitch on $\overline{\text{RST}}$ that is not long enough to reset the device will cause the device to exit suspend. In order for the MCU to respond to the pin reset event, software must not place the device back into suspend mode for a period of 15 µs. The PMU0CF register may be checked to determine if the wake-up was due to a falling edge on the /RST pin. If the wake-up source is not due to a falling edge on $\overline{\text{RST}}$, there is no time restriction on how soon software may place the device back into suspend mode. A 4.7 kW pullup resistor to VDD is recommend for RST to prevent noise glitches from waking the device.

19.6. Sleep Mode

Setting the Sleep Mode Select bit (PMU0CF.7) turns off the internal 1.8 V regulator (VREG0) and switches the power supply of all on-chip RAM to the VBAT pin (see Figure 19.1). Power to most digital logic on the chip is disconnected; only PMU0, LCD, Power Select Switch, Pulse Counter, and the SmaRTClock remain powered. Analog peripherals remain powered; however, only the Comparators remain functional when the device enters Sleep Mode. All other analog peripherals (ADC0, IREF0, External Oscillator, etc.) should be disabled prior to entering Sleep Mode. The system clock source must be set to the low power internal oscillator or the precision oscillator prior to entering Sleep Mode.

GPIO pins configured as digital outputs will retain their output state during sleep mode. In two-cell mode, they will maintain the same current drive capability in sleep mode as they have in normal mode.

GPIO pins configured as digital inputs can be used during sleep mode as wakeup sources using the port match feature. In two-cell mode, they will maintain the same input level specs in sleep mode as they have in normal mode.

‘C8051F96x devices support a wakeup request for external devices. Upon exit from sleep mode, the wake-up request signal is driven low, allowing other devices in the system to wake up from their low power modes.

RAM and SFR register contents are preserved in sleep mode as long as the voltage on VBAT does not fall below V_{POR}. The PC counter and all other volatile state information is preserved allowing the device to resume code execution upon waking up from Sleep mode.

25.3. Programmable Pull-Up Resistors

The Pulse Counter features low-power pull-up resistors with a programmable resistance and duty-cycle. The average pull-up current will depend on the selected resistor, sample rate, and pull-up duty-cycle multiplier. Example code is available that will calculate the values for the Pull-Up configuration SFR (PC0PCF).

Table 25.1through Table 25.3 are used with Equation 25.1 to calculate the average pull-up resistor current. Table 25.4through Table 25.7 give the average current for all combinations.

$$I_{pull\text{-}up} = I_R \times D_{SR} \times D_{PU}$$

Equation 25.1. Average Pull-Up Current

Where:

I_R = Pull-up Resistor current selected by PC0PCF[4:2].

D_{SR} = Sample Rate Duty Cycle Multiplier selected by PC0MD[5:4].

D_{PU} = Pull-Up Duty Cycle Multiplier selected by PC0PCF[4:2].

Table 25.1. Pull-Up Resistor Current

PC0PCF[4:2]	I_R
000	0
001	1 μA
010	4 μA
011	16 μA
100	64 μA
101	256 μA
110	1 mA
111	4 mA

Table 25.2. Sample Rate Duty-Cycle Multiplier

PC0MD[5:4]	D_{SR}
000	1
001	1/2
010	1/4
011	1/8

Table 25.3. Pull-Up Duty-Cycle Multiplier

PC0PCF[4:2]	D_{PU}
000	1/4
001	3/8
010	1/2
011	3/4

To enable the Pulse Counter as a wake up source, enable the source in the PC0INT0/1 SFRs and enable the Pulse Counter as a wake-up source by setting bit 0 (PC0WK) to 1 in the PMU0FL SFR. Upon waking, firmware should read the PMCU0CF and PMU0FL SFRs to determine the wake-up source. If the PC0WK bit is set indicating that the Pulse Counter has woke the MCU, firmware should read the flag bits PC0INT0/1 SFRs to determine the Pulse Counter wake-up source and clear the flag bits before going back to sleep.

PC0INT0 includes the more common interrupt and wake-up sources. These include comparator match, counter overflow, and quadrature direction change. PC0INT1 includes interrupt and wake-up sources for the advanced features, including flutter detection and quadrature error.

25.9. Real-Time Register Access

Several of the Pulse Counter registers values change in real-time synchronous to the RTC clock. Hardware synchronization between the RTC clock domain and the system clock domain hardware would result in long delays when reading real-time registers. Instead, real-time register values are available instantaneously, but the read must be qualified using the read valid bit (PC0TH bit 0). If the register value does not change during the read access, the read valid bit will be set indicating the last was valid. If the value of the real-time register changes during the read access, the read valid bit is 0, indicating the read was invalid. After an invalid read, firmware must read the register and check the read valid bit again.

These 8-bit counter registers need to be qualified using the read valid bit:

- PC0STAT
- PC0HIST
- PC0INT0
- PC0INT1
- PC0CTR0L
- PC0CTC1L

The 24-bit counters are three-byte real-time read-only registers that require a special access method for reading. Firmware must read the low-byte (PC0CTR0L and PC0CTR1L) first and qualify using the read valid bit. Reading the low-byte latches the middle and high bytes. If the read valid bit is 0, the read is invalid and firmware must read the low-byte and check the read valid bit again. If the read valid bit is set, the read is valid and the middle and high bytes are also safe to read. Firmware should read the middle and high bytes only after reading the low byte and qualifying with the read valid bit.

The 24-bit compators are three-byte real-time read-write registers that require a special access method for writing. Firmware must write the low-byte last. After writing the low-byte, it might take up to two RTC clock cycles for the new comparator value to take effect. System designers should consider the synchronization delay when setting the comparator value. The counter may be incremented before new comparator value takes effect. Setting the comparator to at least 2 counts above the current count will eliminate the chance of missing the comparator match during synchronization.

Example code is provided with accessor functions for all the real-time Pulse Counter registers.

25.10. Advanced Features

25.10.1. Quadrature Error

The quadrature encoder must only send valid quadrature codes. A valid quadrature sequence consists of four valid states. The quadrature codes are only permitted to transition to one of the adjacent states, and an invalid transition will result in a quadrature error. Note that a quadrature error is likely to occur when first enabling the quadrature counter mode, since the Pulse Counter state machine starts at the LL state and the initial state of the quadrature is arbitrary. It is safe to ignore the first quadrature error immediately after initialization.

SFR Definition 25.13. PC0CMP0H: PC0 Comparator 0 High (MSB)

Bit	7	6	5	4	3	2	1	0
Name	PC0CMP0H[23:16]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xE3; SFR Page = 0x2

Bit	Name	Function
7:0	PC0CMP0H[23:16]	**PC0 Comparator 0 High Byte** Bits 23:16 of Counter 0.

SFR Definition 25.14. PC0CMP0M: PC0 Comparator 0 Middle

Bit	7	6	5	4	3	2	1	0
Name	PC0CMP0M[15:8]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xE2; SFR Page = 0x2

Bit	Name	Function
7:0	PC0CMP0M[15:8]	**PC0 Comparator 0 Middle Byte** Bits 15:8 of Counter 0.

SFR Definition 25.15. PC0CMP0L: PC0 Comparator 0 Low (LSB)

Bit	7	6	5	4	3	2	1	0
Name	PC0CMP0L[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xE1; SFR Page = 0x2

Bit	Name	Function
7:0	PC0CMP0L[7:0]	**PC0 Comparator 0 Low Byte** Bits 7:0 of Counter 0.

Note: PC0CMP0L must be written last after writing PC0CMP0M and PC0CMP0H. After writing PC0CMP0L, the synchronization into the PC clock domain can take 2 RTC clock cycles.

SFR Definition 26.5. LCD0MSCF: LCD0 Master Configuration

Bit	7	6	5	4	3	2	1	0
Name							DCENSLP	CHPBYP
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	0

SFR Page = 0x2; SFR Address = 0xAC

Bit	Name	Function
7:2	Reserved	Read = 111111b. Must write 111111b.
1	DCENSLP	**DCDC Converter Enable in Sleep Mode** 0: DCDC is disabled in Sleep Mode. 1: DCDC is enabled in Sleep Mode.
0	CHPBYP	**LCD0 Charge Pump Bypass** This bit should be set to 1b in Contrast Control Mode 1 and Mode 2. 0: LCD0 Charge Pump is not bypassed. 1: LCD0 Charge Pump is bypassed.

SFR Definition 26.6. LCD0PWR: LCD0 Power

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

SFR Page = 0x2; SFR Address = 0xA4

Bit	Name	Function
7:4	Unused	Read = 0000b. Write = don’t care.
3	MODE	**LCD0 Contrast Control Mode Selection.** 0: LCD0 Contrast Control Mode 1 or Mode 4 is selected. 1: LCD0 Contrast Control Mode 2 or Mode 3 is selected.
2:0	Reserved	Read = 001b. Must write 001b.

27.1. Port I/O Modes of Operation

Port pins P0.0–P6.7 use the Port I/O cell shown in Figure 27.2. The supply pin for P1.4 - P2.3 is VIORF and the supply for all other GPIOs is VIO. Each Port I/O cell can be configured by software for analog I/O or digital I/O using the PnMDIN registers. P7.0 can only be used for digital functtons and is shared with the C2D signal. On reset, all Port I/O cells default to a digital high impedance state with weak pull-ups enabled.

27.1.1. Port Pins Configured for Analog I/O

Any pins to be used as Comparator or ADC input, external oscillator input/output, or AGND, VREF, or Current Reference output should be configured for analog I/O (PnMDIN.n = 0). When a pin is configured for analog I/O, its weak pullup and digital receiver are disabled. In most cases, software should also disable the digital output drivers. Port pins configured for analog I/O will always read back a value of 0 regardless of the actual voltage on the pin.

Configuring pins as analog I/O saves power and isolates the Port pin from digital interference. Port pins configured as digital inputs may still be used by analog peripherals; however, this practice is not recommended and may result in measurement errors.

27.1.2. Port Pins Configured For Digital I/O

Any pins to be used by digital peripherals (UART, SPI, SMBus, etc.), external digital event capture functions, or as GPIO should be configured as digital I/O (PnMDIN.n = 1). For digital I/O pins, one of two output modes (push-pull or open-drain) must be selected using the PnMDOUT registers.

Push-pull outputs (PnMDOUT.n = 1) drive the Port pad to the supply or GND rails based on the output logic value of the Port pin. Open-drain outputs have the high side driver disabled; therefore, they only drive the Port pad to GND when the output logic value is 0 and become high impedance inputs (both high and low drivers turned off) when the output logic value is 1.

When a digital I/O cell is placed in the high impedance state, a weak pull-up transistor pulls the Port pad to the supply voltage to ensure the digital input is at a defined logic state. Weak pull-ups are disabled when the I/O cell is driven to GND to minimize power consumption and may be globally disabled by setting WEAKPUD to 1. The user must ensure that digital I/O are always internally or externally pulled or driven to a valid logic state. Port pins configured for digital I/O always read back the logic state of the Port pad, regardless of the output logic value of the Port pin.

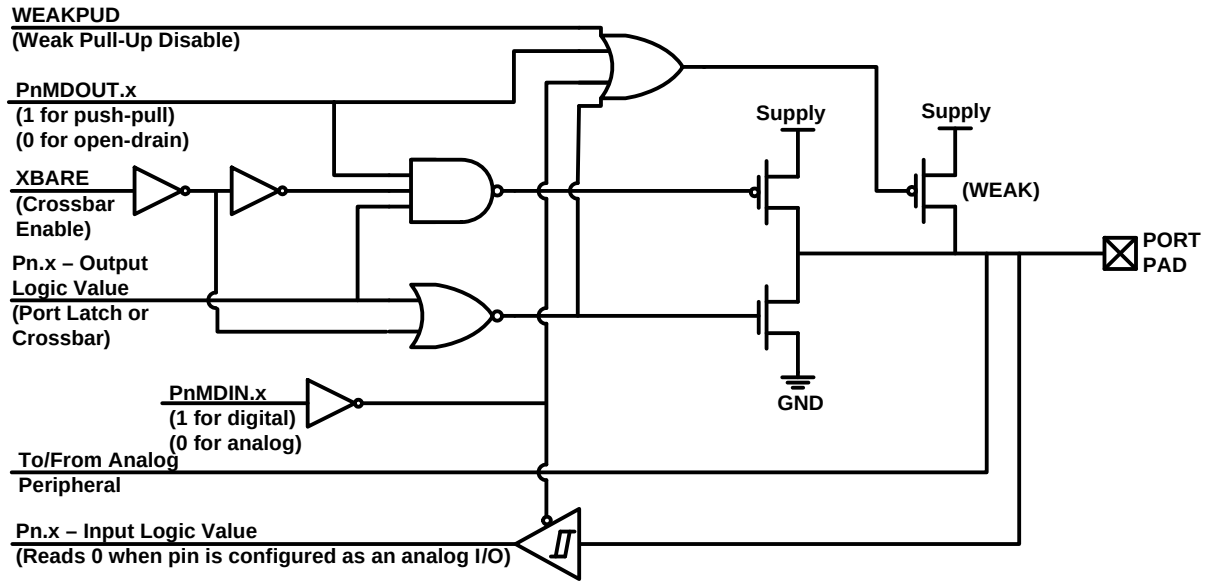

Figure 27.2. Port I/O Cell Block Diagram

SFR Definition 27.18. P2: Port2

Bit	7	6	5	4	3	2	1	0
Name	P2[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = All Pages; SFR Address = 0xA0; Bit-Addressable

Bit	Name	Description	Read	Write
7:0	P2[7:0]	**Port 2 Data.** Sets the Port latch logic value or reads the Port pin logic state in Port cells configured for digital I/O.	0: Set output latch to logic LOW. 1: Set output latch to logic HIGH.	0: P2.n Port pin is logic LOW. 1: P2.n Port pin is logic HIGH.

SFR Definition 27.19. P2SKIP: Port2 Skip

Bit	7	6	5	4	3	2	1	0
Name	P2SKIP[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xD6

Bit	Name	Description	Read	Write
7:0	P2SKIP[7:0]	**Port 1 Crossbar Skip Enable Bits.** These bits select Port 2 pins to be skipped by the Crossbar Decoder. Port pins used for analog, special functions or GPIO should be skipped by the Crossbar. 0: Corresponding P2.n pin is not skipped by the Crossbar. 1: Corresponding P2.n pin is skipped by the Crossbar.		

SFR Definition 27.36. P6MDIN: Port6 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	P6MDIN[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0xF; SFR Address = 0xF4

Bit	Name	Function
7:0	P6MDIN[3:0]	**Analog Configuration Bits for P6.7–P6.0 (respectively).** Port pins configured for analog mode have their weak pullup and digital receiver disabled. The digital driver is not explicitly disabled. 0: Corresponding P6.n pin is configured for analog mode. 1: Corresponding P6.n pin is not configured for analog mode.

SFR Definition 27.37. P6MDOUT: Port6 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P6MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0xF; SFR Address = 0xFB

Bit	Name	Function
7:0	P6MDOUT[7:0]	**Output Configuration Bits for P6.7–P6.0 (respectively).** These bits control the digital driver even when the corresponding bit in register P6MDIN is logic 0. 0: Corresponding P6.n Output is open-drain. 1: Corresponding P6.n Output is push-pull.

Figure 28.5 shows a typical master write sequence. Two transmit data bytes are shown, though any number of bytes may be transmitted. All “data byte transferred” interrupts occur **after** the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

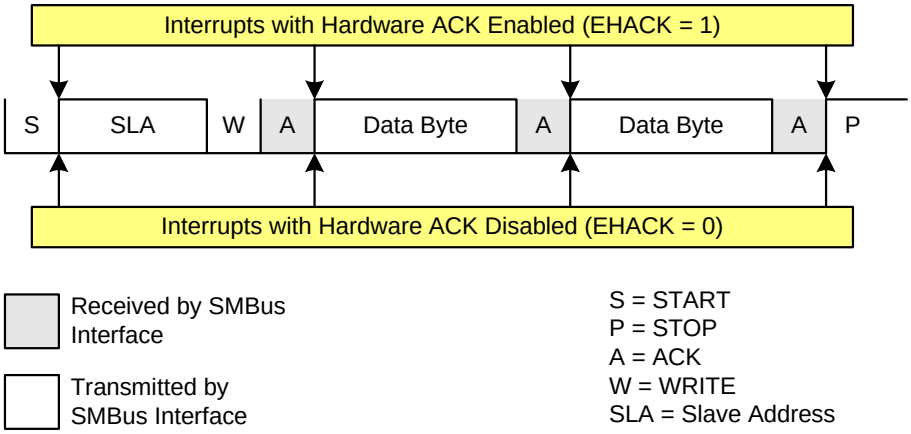

Figure 28.5. Typical Master Write Sequence

28.5.2. Read Sequence (Master)

During a read sequence, an SMBus master reads data from a slave device. The master in this transfer will be a transmitter during the address byte, and a receiver during all data bytes. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 1 (READ). Serial data is then received from the slave on SDA while the SMBus outputs the serial clock. The slave transmits one or more bytes of serial data.

If hardware ACK generation is disabled, the ACKRQ is set to 1 and an interrupt is generated after each received byte. Software must write the ACK bit at that time to ACK or NACK the received byte.

With hardware ACK generation enabled, the SMBus hardware will automatically generate the ACK/NACK, and then post the interrupt. **It is important to note that the appropriate ACK or NACK value should be set up by the software prior to receiving the byte when hardware ACK generation is enabled.**

Writing a 1 to the ACK bit generates an ACK; writing a 0 generates a NACK. Software should write a 0 to the ACK bit for the last data transfer, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. The interface will switch to Master Transmitter Mode if SMB0DAT is written while an active Master Receiver. Figure 28.6 shows a typical master read sequence. Two received data bytes are shown, though any number of bytes may be received. The “data byte transferred” interrupts occur at different places in the sequence, depending on whether hardware ACK generation is enabled. The interrupt occurs **before** the ACK with hardware ACK generation disabled, and **after** the ACK when hardware ACK generation is enabled.

is set. While the SPI0 master transfers data to a slave on the MOSI line, the addressed SPI slave device simultaneously transfers the contents of its shift register to the SPI master on the MISO line in a full-duplex operation. Therefore, the SPIF flag serves as both a transmit-complete and receive-data-ready flag. The data byte received from the slave is transferred MSB-first into the master's shift register. When a byte is fully shifted into the register, it is moved to the receive buffer where it can be read by the processor by reading SPI0DAT.

When configured as a master, SPI0 can operate in one of three different modes: multi-master mode, 3-wire single-master mode, and 4-wire single-master mode. The default, multi-master mode is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 1. In this mode, NSS is an input to the device, and is used to disable the master SPI0 when another master is accessing the bus. When NSS is pulled low in this mode, MSTEN (SPI0CN.6) and SPIEN (SPI0CN.0) are set to 0 to disable the SPI master device, and a Mode Fault is generated (MODF, SPI0CN.5 = 1). Mode Fault will generate an interrupt if enabled. SPI0 must be manually re-enabled in software under these circumstances. In multi-master systems, devices will typically default to being slave devices while they are not acting as the system master device. In multi-master mode, slave devices can be addressed individually (if needed) using general-purpose I/O pins. Figure 30.2 shows a connection diagram between two master devices in multiple-master mode.

3-wire single-master mode is active when NSSMD1 (SPI0CN.3) = 0 and NSSMD0 (SPI0CN.2) = 0. In this mode, NSS is not used, and is not mapped to an external port pin through the crossbar. Any slave devices that must be addressed in this mode should be selected using general-purpose I/O pins. Figure 30.3 shows a connection diagram between a master device in 3-wire master mode and a slave device.

4-wire single-master mode is active when NSSMD1 (SPI0CN.3) = 1. In this mode, NSS is configured as an output pin, and can be used as a slave-select signal for a single SPI device. In this mode, the output value of NSS is controlled (in software) with the bit NSSMD0 (SPI0CN.2). Additional slave devices can be addressed using general-purpose I/O pins. Figure 30.4 shows a connection diagram for a master device in 4-wire master mode and two slave devices.

SFR Definition 31.3. SPI1CKR: SPI1 Clock Rate

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

SFR Page = 0x0; SFR Address = 0xA2

Bit	Name	Function
7:0	SCR[7:0]	**SPI1 Clock Rate.** These bits determine the frequency of the SCK output when the SPI1 module is configured for master mode operation. The SCK clock frequency is a divided version of the system clock, and is given in the following equation, where *SYSCLK* is the system clock frequency and *SPI1CKR* is the 8-bit value held in the SPI1CKR register. $f_{SCK} = \frac{SYSCLK}{2 \times (SPI1CKR[7:0] + 1)}$ for 0 <= SPI1CKR <= 255 Example: If SYSCLK = 2 MHz and SPI1CKR = 0x04, $f_{SCK} = \frac{2000000}{2 \times (4 + 1)}$ $f_{SCK} = 200kHz$

SFR Definition 31.4. SPI1DAT: SPI1 Data

Bit	7	6	5	4	3	2	1	0
Name	SPI1DAT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xA3

Bit	Name	Function
7:0	SPI1DAT[7:0]	**SPI1 Transmit and Receive Data.** The SPI1DAT register is used to transmit and receive SPI1 data. Writing data to SPI1DAT places the data into the transmit buffer and initiates a transfer when in Master Mode. A read of SPI1DAT returns the contents of the receive buffer.

32. Timers

Each MCU includes four counter/timers: two are 16-bit counter/timers compatible with those found in the standard 8051, and two are 16-bit auto-reload timer for use with the ADC, SMBus, or for general purpose use. These timers can be used to measure time intervals, count external events and generate periodic interrupt requests. Timer 0 and Timer 1 are nearly identical and have four primary modes of operation. Timer 2 and Timer 3 offer 16-bit and split 8-bit timer functionality with auto-reload. Additionally, Timer 2 and Timer 3 have a Capture Mode that can be used to measure the SmaRTClock, Comparator, or external clock period with respect to another oscillator. The ability to measure the Comparator period with respect to another oscillator is particularly useful when interfacing to capacitive sensors.

Timer 0 and Timer 1 Modes:	Timer 2 Modes:	Timer 3 Modes:
13-bit counter/timer	16-bit timer with auto-reload	16-bit timer with auto-reload
16-bit counter/timer		
8-bit counter/timer with auto-reload	Two 8-bit timers with auto-reload	Two 8-bit timers with auto-reload
Two 8-bit counter/timers (Timer 0 only)		

Timers 0 and 1 may be clocked by one of five sources, determined by the Timer Mode Select bits (T1M–T0M) and the Clock Scale bits (SCA1–SCA0). The Clock Scale bits define a pre-scaled clock from which Timer 0 and/or Timer 1 may be clocked (See SFR Definition 32.1 for pre-scaled clock selection).

Timer 0/1 may then be configured to use this pre-scaled clock signal or the system clock. Timer 2 and Timer 3 may be clocked by the system clock, the system clock divided by 12. Timer 2 may additionally be clocked by the SmaRTClock divided by 8 or the Comparator0 output. Timer 3 may additionally be clocked by the external oscillator clock source divided by 8 or the Comparator1 output.

Timer 0 and Timer 1 may also be operated as counters. When functioning as a counter, a counter/timer register is incremented on each high-to-low transition at the selected input pin (T0 or T1). Events with a frequency of up to one-fourth the system clock frequency can be counted. The input signal need not be periodic, but it should be held at a given level for at least two full system clock cycles to ensure the level is properly sampled.

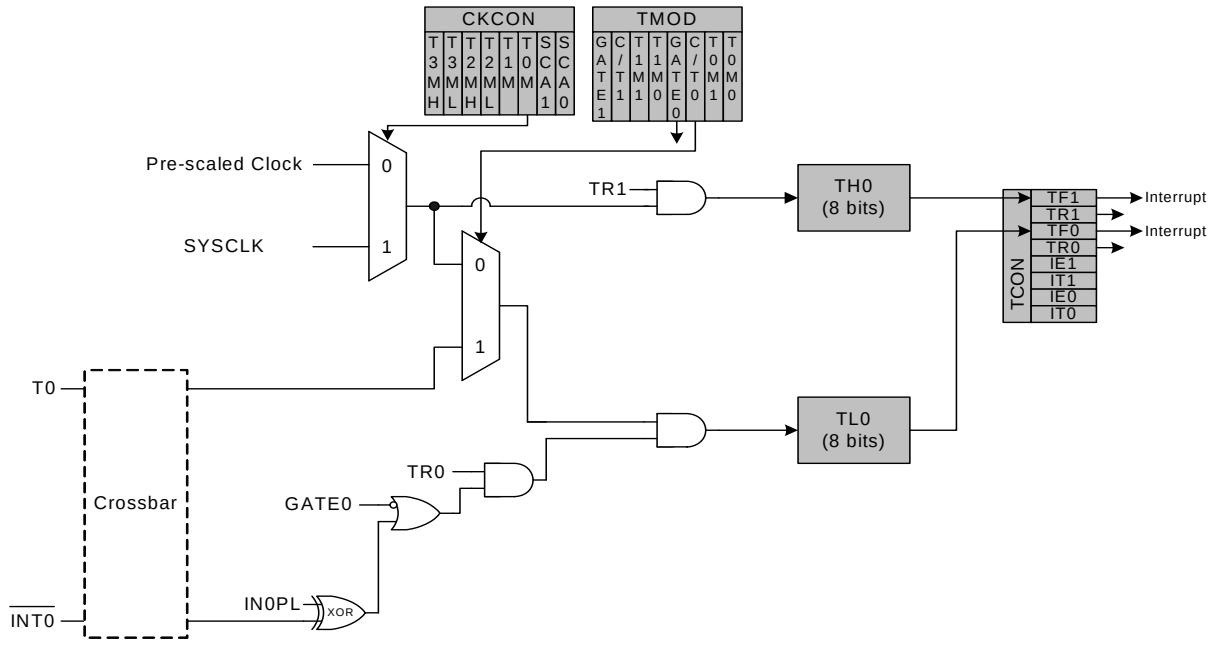

Figure 32.3. T0 Mode 3 Block Diagram

SFR Definition 32.9. TMR2RLL: Timer 2 Reload Register Low Byte

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

SFR Page = 0x0; SFR Address = 0xCA

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

SFR Definition 32.10. TMR2RLH: Timer 2 Reload Register High Byte

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

SFR Page = 0x0; SFR Address = 0xCB

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

Setting TF3CEN to 1 enables the SmaRTClock/External Oscillator Capture Mode for Timer 3. In this mode, T3SPLIT should be set to 0, as the full 16-bit timer is used.

When Capture Mode is enabled, a capture event will be generated either every SmaRTClock rising edge or every 8 external clock cycles, depending on the T3XCLK1 setting. When the capture event occurs, the contents of Timer 3 (TMR3H:TMR3L) are loaded into the Timer 3 reload registers (TMR3RLH:TMR3RLL) and the TF3H flag is set (triggering an interrupt if Timer 3 interrupts are enabled). By recording the difference between two successive timer capture values, the SmaRTClock or external clock period can be determined with respect to the Timer 3 clock. The Timer 3 clock should be much faster than the capture clock to achieve an accurate reading.

For example, if T3ML = 1b, T3XCLK1 = 0b, and TF3CEN = 1b, Timer 3 will clock every SYSCLK and capture every SmaRTClock rising edge. If SYSCLK is 24.5 MHz and the difference between two successive captures is 350 counts, then the SmaRTClock period is as follows:

350 x (1 / 24.5 MHz) = 14.2 µs.

This mode allows software to determine the exact frequency of the external oscillator in C and RC mode or the time between consecutive SmaRTClock rising edges, which is useful for determining the SmaRTClock frequency.

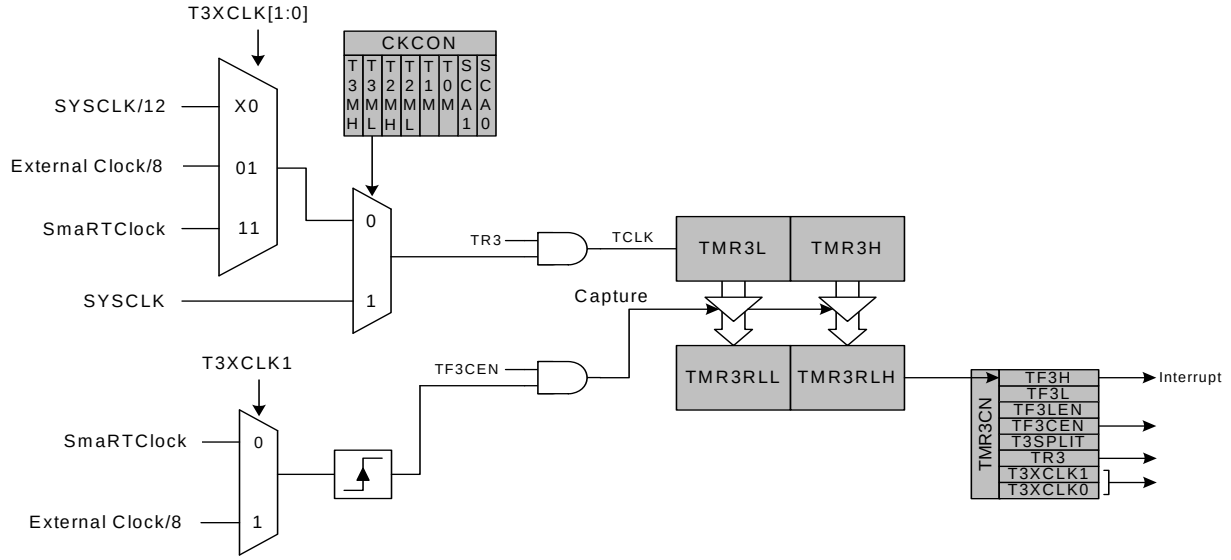

Figure 32.9. Timer 3 Capture Mode Block Diagram

SFR Definition 33.7. PCA0CPLn: PCA Capture Module Low Byte

Bit	7	6	5	4	3	2	1	0
Name	PCA0CPn[7:0]							
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Addresses: PCA0CPL0 = 0xFB, PCA0CPL1 = 0xE9, PCA0CPL2 = 0xEB,
PCA0CPL3 = 0xED, PCA0CPL4 = 0xFD, PCA0CPL5 = 0xD2

SFR Pages: PCA0CPL0 = 0x0, PCA0CPL1 = 0x0, PCA0CPL2 = 0x0,
PCA0CPL3 = 0x0, PCA0CPL4 = 0x0, PCA0CPL5 = 0x0

Bit	Name	Function
7:0	PCA0CPn[7:0]	**PCA Capture Module Low Byte.** The PCA0CPLn register holds the low byte (LSB) of the 16-bit capture module n. This register address also allows access to the low byte of the corresponding PCA channel's auto-reload value for 9, 10, or 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed.
Note: A write to this register will clear the module's ECOMn bit to a 0.		

SFR Definition 33.8. PCA0CPHn: PCA Capture Module High Byte

Bit	7	6	5	4	3	2	1	0
Name	PCA0CPn[15:8]							
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Addresses: PCA0CPH0 = 0xFC, PCA0CPH1 = 0xEA, PCA0CPH2 = 0xEC,
PCA0CPH3 = 0xEE, PCA0CPH4 = 0xFE, PCA0CPH5 = 0xD3

SFR Pages: PCA0CPH0 = 0x0, PCA0CPH1 = 0x0, PCA0CPH2 = 0x0,
PCA0CPH3 = 0x0, PCA0CPH4 = 0x0, PCA0CPH5 = 0x0

Bit	Name	Function
7:0	PCA0CPn[15:8]	**PCA Capture Module High Byte.** The PCA0CPHn register holds the high byte (MSB) of the 16-bit capture module n. This register address also allows access to the high byte of the corresponding PCA channel's auto-reload value for 9, 10, or 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed.
Note: A write to this register will set the module's ECOMn bit to a 1.		