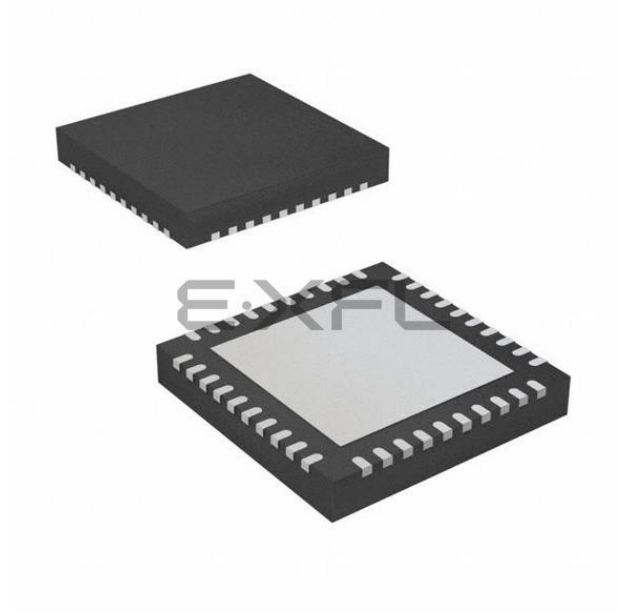


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	8051
Core Size	8-Bit
Speed	25MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	DMA, LCD, POR, PWM, WDT
Number of I/O	34
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4.25K 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	40-VFQFN Exposed Pad
Supplier Device Package	40-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f969-b-gm

Table 4.18. LCD0 Electrical Characteristics

V_{BAT} = 1.8 to 3.8 V; T_A = –40 to +85 °C unless otherwise specified; Using factory-calibrated settings.

Parameter	Conditions	Min	Typ	Max	Units
Charge Pump Output Voltage Error		—	±30	—	mV
LCD Clock Frequency		16	—	33	kHz

Table 4.19. PC0 Electrical Characteristics

V_{BAT} = 1.8 to 3.8 V; T_A = –40 to +85 °C unless otherwise specified; Using factory-calibrated settings.

Parameter	Conditions	Min	Typ	Max	Units
Supply Current (25 °C, 2 ms sample rate)	1.8 V	—	145	—	nA
	2.2 V	—	175	—	
	3.0 V	—	235	—	
	3.8 V	—	285	—	

7. Comparators

C8051F96x devices include two on-chip programmable voltage comparators: Comparator 0 (CPT0) is shown in Figure 7.1; Comparator 1 (CPT1) is shown in Figure 7.2. The two comparators operate identically, but may differ in their ability to be used as reset or wake-up sources. See the Reset Sources chapter and the Power Management chapter for details on reset sources and low power mode wake-up sources, respectively.

The Comparator offers programmable response time and hysteresis, an analog input multiplexer, and two outputs that are optionally available at the Port pins: a synchronous “latched” output (CP0, CP1), or an asynchronous “raw” output (CP0A, CP1A). The asynchronous CP0A signal is available even when the system clock is not active. This allows the Comparator to operate and generate an output when the device is in some low power modes.

7.1. Comparator Inputs

Each Comparator performs an analog comparison of the voltage levels at its positive (CP0+ or CP1+) and negative (CP0- or CP1-) input. Both comparators support multiple port pin inputs multiplexed to their positive and negative comparator inputs using analog input multiplexers. The analog input multiplexers are completely under software control and configured using SFR registers. See Section “7.6. Comparator0 and Comparator1 Analog Multiplexers” on page 112 for details on how to select and configure Comparator inputs.

Important Note About Comparator Inputs: The Port pins selected as Comparator inputs should be configured as analog inputs and skipped by the Crossbar. See the Port I/O chapter for more details on how to configure Port I/O pins as Analog Inputs. The Comparator may also be used to compare the logic level of digital signals, however, Port I/O pins configured as digital inputs must be driven to a valid logic state (HIGH or LOW) to avoid increased power consumption.

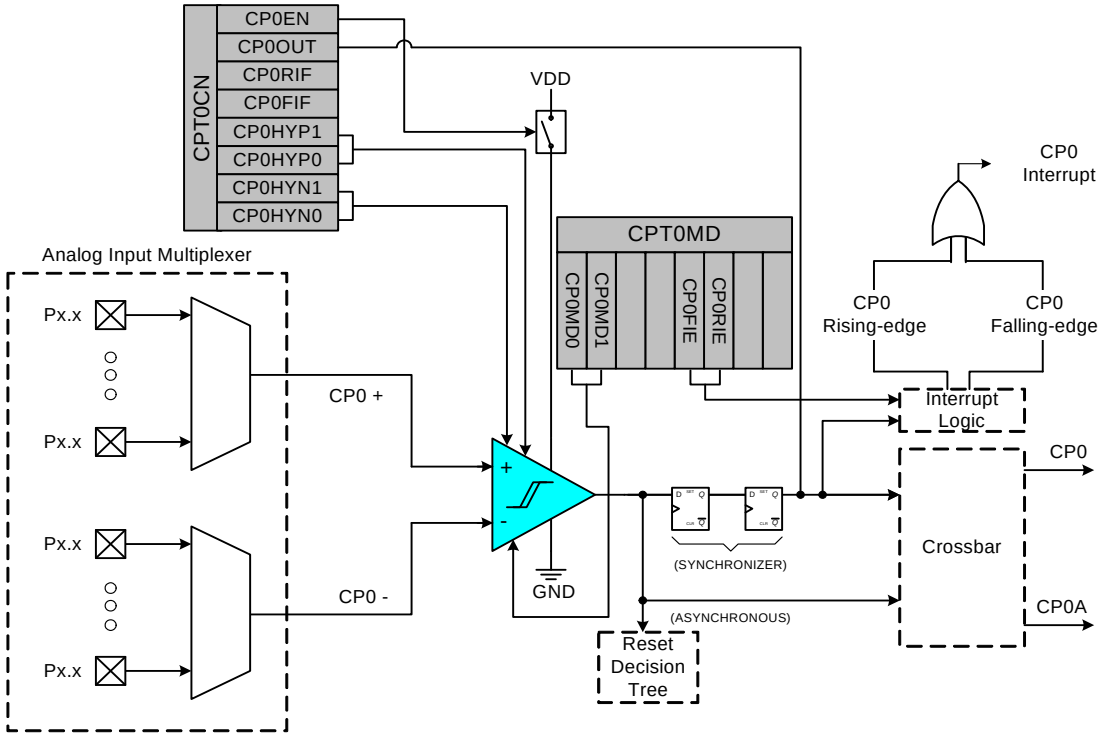


Figure 7.1. Comparator 0 Functional Block Diagram

12. Cyclic Redundancy Check Unit (CRC0)

C8051F96x devices include a cyclic redundancy check unit (CRC0) that can perform a CRC using a 16-bit or 32-bit polynomial. CRC0 accepts a stream of 8-bit data written to the CRC0IN register. CRC0 posts the 16-bit or 32-bit result to an internal register. The internal result register may be accessed indirectly using the CRC0PNT bits and CRC0DAT register, as shown in Figure 12.1. CRC0 also has a bit reverse register for quick data manipulation.

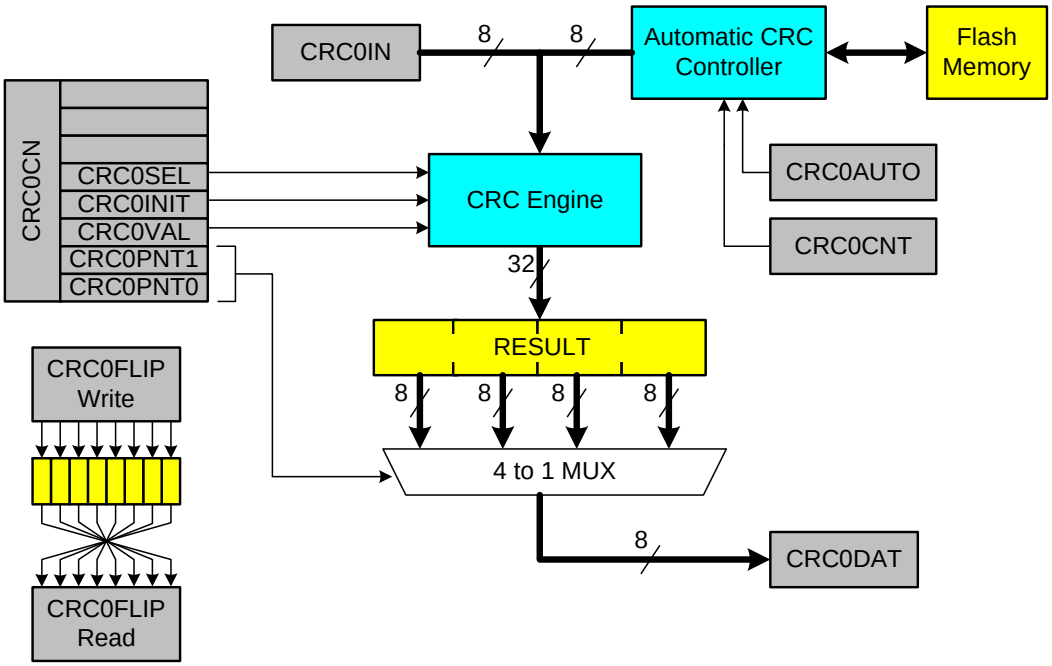


Figure 12.1. CRC0 Block Diagram

12.1. 16-bit CRC Algorithm

The C8051F96x CRC unit calculates the 16-bit CRC MSB-first, using a poly of 0x1021. The following describes the 16-bit CRC algorithm performed by the hardware:

1. XOR the most-significant byte of the current CRC result with the input byte. If this is the first iteration of the CRC unit, the current CRC result will be the set initial value (0x0000 or 0xFFFF).
   a. If the MSB of the CRC result is set, left-shift the CRC result, and then XOR the CRC result with the polynomial (0x1021).
   b. If the MSB of the CRC result is not set, left-shift the CRC result.
2. Repeat at Step 2a for the number of input bits (8).

Table 12.2. Example 32-bit CRC Outputs

Input	Output
0x63	0xF9462090
0xAA, 0xBB, 0xCC	0x41B207B3
0x00, 0x00, 0xAA, 0xBB, 0xCC	0x78D129BC

12.3. Preparing for a CRC Calculation

To prepare CRC0 for a CRC calculation, software should select the desired polynomial and set the initial value of the result. Two polynomials are available: 0x1021 (16-bit) and 0x04C11DB7 (32-bit). The CRC0 result may be initialized to one of two values: 0x00000000 or 0xFFFFFFFF. The following steps can be used to initialize CRC0.

1. Select a polynomial (Set CRC0SEL to 0 for 32-bit or 1 for 16-bit).
2. Select the initial result value (Set CRC0VAL to 0 for 0x00000000 or 1 for 0xFFFFFFFF).
3. Set the result to its initial value (Write 1 to CRC0INIT).

12.4. Performing a CRC Calculation

Once CRC0 is initialized, the input data stream is sequentially written to CRC0IN, one byte at a time. The CRC0 result is automatically updated after each byte is written. The CRC engine may also be configured to automatically perform a CRC on one or more flash sectors. The following steps can be used to automatically perform a CRC on flash memory.

1. Prepare CRC0 for a CRC calculation as shown above.
2. If necessary, set the IFBANK bits in the PSBANK for the desired code bank.
3. Write the index of the starting page to CRC0AUTO.
4. Set the AUTOEN bit in CRC0AUTO.
5. Write the number of flash sectors to perform in the CRC calculation to CRC0CNT.
   Note: Each flash sector is 1024 bytes.
6. Write any value to CRC0CN (or OR its contents with 0x00) to initiate the CRC calculation. The CPU will not execute code any additional code until the CRC operation completes.
7. Clear the AUTOEN bit in CRC0AUTO.
8. Read the CRC result using the procedure below.

Setting the IFBANK bits in the PSBANK SFR is only necessary when accessing the upper banks on 128 kB code bank devices (‘F960/1/2/3). Multiple CRCs are required to cover the entire 128 kB Flash array. When writing to the PSBANK SFR, the code initiating the auto CRC of flash must be executing from the common area.

12.5. Accessing the CRC0 Result

The internal CRC0 result is 32-bits (CRC0SEL = 0b) or 16-bits (CRC0SEL = 1b). The CRC0PNT bits select the byte that is targeted by read and write operations on CRC0DAT and increment after each read or write. The calculation result will remain in the internal CR0 result register until it is set, overwritten, or additional data is written to CRC0IN.

The key and data to be encrypted should be stored as an array with the first byte to be encrypted at the lowest address. The value of the big endian bit of the DMACF0 sfr does not matter. The AES block uses only one byte transfers, so there is no particular endianness associated with a one byte transfer.

The dummy data can be zeros or any value. The encrypted data is discarded, so the value of the dummy data does not mater.

It is not strictly required to use DMA channels 0, 1, and 2. Any three DMA channels may be used. The internal state machine of the AES module will send the peripheral requests in the required order.

If the other DMA channels are going to be used concurrently with encryption, then only the bits corresponding to the encryption channels should be manipulated in DM0AEN and DMA0NT sfrs.

14.2.2. Key Inversion using SFRs

Normally, the AES block is used with the DMA. This provides the best performance and lowest power consumption. However, it is also possible to use the DMA with direct SFR access. The steps are documented in the datasheet for completeness.

Steps to generate the Decryption Key from Encryption Key using SFR.

- First configure the AES block for Key inversion:
  - Reset AES module by writing 0x00 to AES0BCFG.
  - Configure the AES Module data flow for inverse key generation by writing 0x04 to the AES0DCFG sfr.
  - Write key size to bits 1 and 0 of the AES0BCFG.
  - Configure the AES core for encryption by setting bit 2 of AES0BCFG.
  - Enable the AES core by setting bit 3 of AES0BCFG.
- Write the dummy data alternating with Key data:
  - Write the first dummy byte to AES0BIN
  - Write the first key byte to AES0KIN
  - Repeat until all dummy data bytes are written
- If using 192-bit and 256-bit key, write remaining key bytes to AES0KIN:
- Wait on AES done interrupt or poll bit 5 of AES0BCFG
- Read first byte of the decryption key from the AES0YOUT sfr

14.6.2. CBC Encryption Initialization Vector Location

The first block to be encrypted uses the initialization vector for the AES0XIN data. Subsequent blocks will use the encrypted ciphertext from the previous block. The DMA is capable of encrypting multiple blocks. If the initialization is located at an arbitrary location in xram, the DMA base address location will need to be changed to the start of the encrypted ciphertext after encrypting the first block. However, if the initialization vector explicitly located in xram immediately before the encrypted ciphertext, the pointer will be advanced to the start of the encrypted ciphertext naturally and multiple blocks can be encrypted autonomously.

14.6.3. CBC Encryption 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 encryption Key, initialization vector, and data to be encrypted in xram.

(The initialization vector should be located immediately before the data to be encrypted to encrypt 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 encryption 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 encrypted to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be encrypted 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 encrypted in multiples of 16 bytes to the DMA0NSZH and DMA0NSZL sfrs.
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- * Configure the fourth DMA channel for the AES0YOUT sfr
  - Select the fourth channel by writing 0x03 to the DMA0SEL sfr
  - Configure the fourth 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 encrypted data to the DMA0NBAH and DMA0NBAL sfrs.
  - Write the number of bytes to be encrypted 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.

SFR Definition 14.6. AES0YOUT: AES Y Output

Bit	7	6	5	4	3	2	1	0
Name	AES0YOUT[7:0]							
Type	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xF5; SFR page = 0x2; Not bit-Addressable

Bit	Name	Function
7:0	AES0YOUT[7:0]	**AES Y Output.** Upon completion of an encryption/decryption operation The output data may be read, one byte at a time, from the AES0YOUT SFR. When used with the DMA, the DMA will read directly from this SFR. The AES0YOUT SFR may be used in conjunction with the AES0XIN SFR for some cipher block modes. When used without the DMA, the firmware should wait on the DONE flag before reading from the AES0YOUT SFR. When used without the DMA and using XOR on the output, one byte should be written to AES0XIN before reading each byte from AES0YOUT. Reading this register over the C2 interface will not increment the output data.

SFR Definition 17.6. EIP2: Extended Interrupt Priority 2

Bit	7	6	5	4	3	2	1	0
Name	PAES0	PENC0	PDMA0	PPC0	PSPI1	PRTC0F	PMAT	PWARN
Type	R	R	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Page = All Pages; SFR Address = 0xF7

Bit	Name	Function
7	PAES0	**AES0 Interrupt Priority Control.** This bit sets the priority of the AES0 interrupt. 0: AES0 interrupt set to low priority level. 1: AES0 interrupt set to high priority level.
6	PENC0	**Encoder (ENC0) Interrupt Priority Control.** This bit sets the priority of the ENC0 interrupt. 0: ENC0 interrupt set to low priority level. 1: SPI0 interrupt set to high priority level.
5	PDMA0	**DMA0 Interrupt Priority Control.** This bit sets the priority of the DMA0 interrupt. 0: DMA0 interrupt set to low priority level. 1: DMA0 interrupt set to high priority level.
4	PPC0	**Pulse Counter (PC0) Interrupt Priority Control.** This bit sets the priority of the PC0 interrupt. 0: PC0 interrupt set to low priority level. 1: PC0 interrupt set to high priority level.
3	PSPI1	**Serial Peripheral Interface (SPI1) Interrupt Priority Control.** This bit sets the priority of the SPI0 interrupt. 0: SPI1 interrupt set to low priority level. 1: SPI1 interrupt set to high priority level.
2	PRTC0F	**SmaRTClock Oscillator Fail Interrupt Priority Control.** This bit sets the priority of the SmaRTClock Alarm interrupt. 0: SmaRTClock Alarm interrupt set to low priority level. 1: SmaRTClock Alarm interrupt set to high priority level.
1	PMAT	**Port Match Interrupt Priority Control.** This bit sets the priority of the Port Match Event interrupt. 0: Port Match interrupt set to low priority level. 1: Port Match interrupt set to high priority level.
0	PWARN	**VDD/DC+ Supply Monitor Early Warning Interrupt Priority Control.** This bit sets the priority of the VDD/DC+ Supply Monitor Early Warning interrupt. 0: VDD/DC+ Supply Monitor Early Warning interrupt set to low priority level. 1: VDD/DC+ Supply Monitor Early Warning interrupt set to high priority level.

running (RTC0TR = 1) in order to set or capture the main timer. The transfer can take up to 2 smaRTClock cycles to complete.

24.3.2. Setting a SmaRTClock Alarm

The SmaRTClock alarm function compares the 32-bit value of SmaRTClock Timer to the value of the ALARMnBn registers. An alarm event is triggered if the SmaRTClock timer is **equal to** the ALARMnBn registers. If Auto Reset is enabled, the 32-bit timer will be cleared to zero one SmaRTClock cycle after the alarm 0 event.

The SmaRTClock alarm event can be configured to reset the MCU, wake it up from a low power mode, or generate an interrupt. See Section “17. Interrupt Handler” on page 232, Section “19. Power Management” on page 257, and Section “22. Reset Sources” on page 278 for more information.

The following steps can be used to set up a SmaRTClock Alarm:

1. Disable SmaRTClock Alarm Events (RTC0AEN = 0).
2. Set the ALARMn registers to the desired value.
3. Enable SmaRTClock Alarm Events (RTC0AEN = 1).

Notes:

1. The ALRM bit, which is used as the SmaRTClock Alarm Event flag, is cleared by disabling SmaRTClock Alarm Events (RTC0AEN = 0).
2. If AutoReset is disabled, disabling (RTC0AEN = 0) then Re-enabling Alarm Events (RTC0AEN = 1) after a SmaRTClock Alarm without modifying ALARMn registers will automatically schedule the next alarm after 2^32 SmaRTClock cycles (approximately 36 hours using a 32.768 kHz crystal).

24.3.3. Software Considerations for using the SmaRTClock Timer and Alarm

The SmaRTClock timer and alarm have two operating modes to suit varying applications. The two modes are described below:

Mode 1:
The first mode uses the SmaRTClock timer as a perpetual timebase which is never reset to zero. Every 36 hours, the timer is allowed to overflow without being stopped or disrupted. The alarm interval is software managed and is added to the ALRMnBn registers by software after each alarm. This allows the alarm match value to always stay ahead of the timer by one software managed interval. If software uses 32-bit unsigned addition to increment the alarm match value, then it does not need to handle overflows since both the timer and the alarm match value will overflow in the same manner.

This mode is ideal for applications which have a long alarm interval (e.g., 24 or 36 hours) and/or have a need for a perpetual timebase. An example of an application that needs a perpetual timebase is one whose wake-up interval is constantly changing. For these applications, software can keep track of the number of timer overflows in a 16-bit variable, extending the 32-bit (36 hour) timer to a 48-bit (272 year) perpetual timebase.

Mode 2:
The second mode uses the SmaRTClock timer as a general purpose up counter which is auto reset to zero by hardware after each alarm 0 event. The alarm interval is managed by hardware and stored in the ALRM0Bn registers. Software only needs to set the alarm interval once during device initialization. After each alarm 0 event, software should keep a count of the number of alarms that have occurred in order to keep track of time. Alarm 1 and alarm 2 events do not trigger the auto reset.

This mode is ideal for applications that require minimal software intervention and/or have a fixed alarm interval. This mode is the most power efficient since it requires less CPU time per alarm.

25. Low-Power Pulse Counter

The C8051F96x family of microcontrollers contains a low-power Pulse Counter module with advanced features, such as ultra low power input comparators, a wide range of pull up values with a self calibration engine, asymmetrical integrators for low pass filtering and switch debounce, single, dual, and quadrature modes of operation, two 24-bit counters, threshold comparators, and a variety of interrupt and sleep wake up capabilities. This combination of features provides water, gas, and heat metering system designers with an optimal tool for saving power while collecting meter usage data.

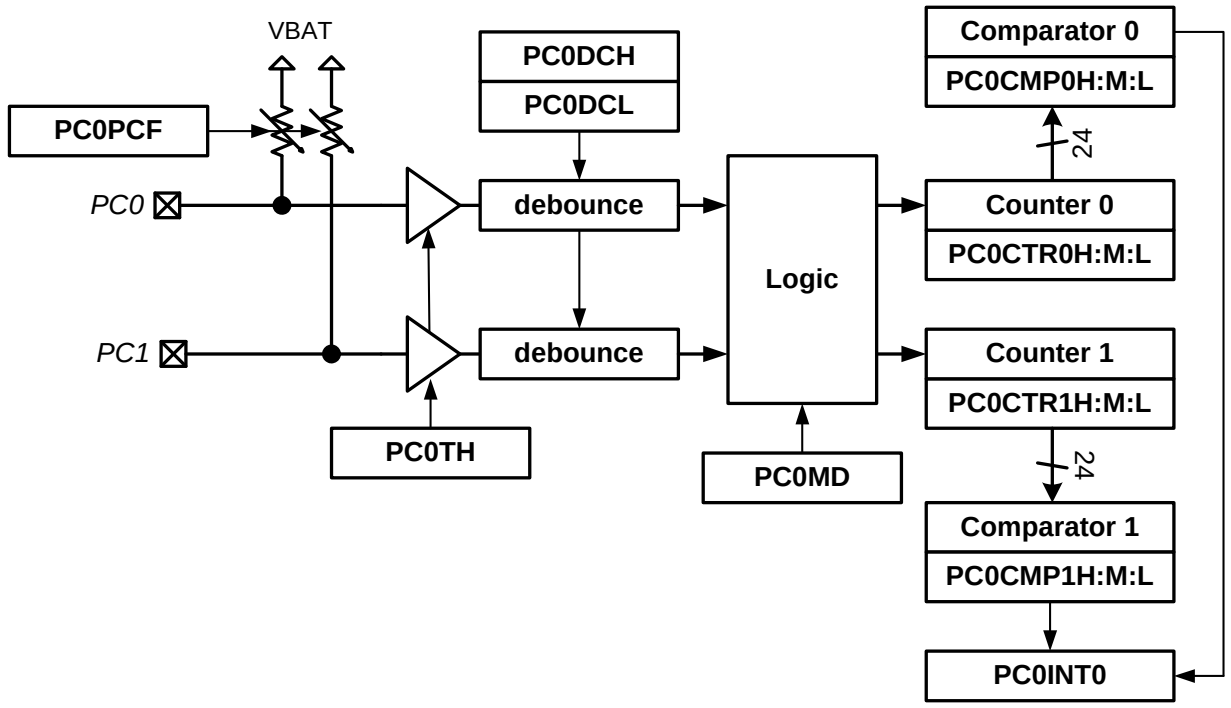


Figure 25.1. Pulse Counter Block Diagram

The low-power Pulse Counter is a low-power sleep-mode peripheral designed primarily to work meters using reed switches, including water and gas meters. The Pulse Counter is very flexible and can count pulses from many different types of sources.

The Pulse Counter operates in sleep mode to enable ultra-low power metering systems. The MCU does not have to wake up on every edge or transition and can remain in sleep mode while the Pulse Counter counts pulses for an extended period of time. The Pulse Counter includes two 24-bit counters. These counters can count up to 16,777,215 (2^{24}-1) transitions in sleep mode before overflowing. The Pulse Counter can wake up the MCU when one of the counters overflows. The Pulse Counter also has two 24-bit comparators. The comparators have the ability to wake up the MCU when the one of the counters reaches a predetermined threshold.

The Pulse Counter uses the RTC clock for sampling, de-bouncing, and managing the low-power pull-up resistors. The RTC must be enabled when counting pulses. The RTC alarms can wake up the MCU periodically to read the pulse counters, instead of using the Pulse Counter comparators. For example, the RTC can wake up the MCU every five minutes. The MCU can then read the Pulse Counter and transmit the information using the UART or a wireless transceiver.

SFR Definition 27.1. XBR0: Port I/O Crossbar Register 0

Bit	7	6	5	4	3	2	1	0
Name	CP1AE	CP1E	CP0AE	CP0E	SYSCKE	SMB0E	SPI0E	URT0E
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 Page = 0x0; SFR Address = 0xE1

Bit	Name	Function
7	CP1AE	**Comparator1 Asynchronous Output Enable.** 0: Asynchronous CP1 output unavailable at Port pin. 1: Asynchronous CP1 output routed to Port pin.
6	CP1E	**Comparator1 Output Enable.** 0: CP1 output unavailable at Port pin. 1: CP1 output routed to Port pin.
5	CP0AE	**Comparator0 Asynchronous Output Enable.** 0: Asynchronous CP0 output unavailable at Port pin. 1: Asynchronous CP0 output routed to Port pin.
4	CP0E	**Comparator0 Output Enable.** 0: CP1 output unavailable at Port pin. 1: CP1 output routed to Port pin.
3	SYSCKE	**$\overline{\text{SYSCLK}}$ Output Enable.** 0: $\overline{\text{SYSCLK}}$ output unavailable at Port pin. 1: $\overline{\text{SYSCLK}}$ output routed to Port pin.
2	SMB0E	**SMBus I/O Enable.** 0: SMBus I/O unavailable at Port pin. 1: SDA and SCL routed to Port pins.
1	SPI0E	**SPI0 I/O Enable** 0: SPI0 I/O unavailable at Port pin. 1: SCK, MISO, and MOSI (for SPI0) routed to Port pins. NSS (for SPI0) routed to Port pin only if SPI0 is configured to 4-wire mode.
0	URT0E	**UART0 Output Enable.** 0: UART I/O unavailable at Port pin. 1: TX0 and RX0 routed to Port pins P0.4 and P0.5.
Note: SPI0 can be assigned either 3 or 4 Port I/O pins.		

SFR Definition 27.16. P1MDOUT: Port1 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P1MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xA5

Bit	Name	Function
7:0	P1MDOUT[7:0]	**Output Configuration Bits for P1.7–P1.0 (respectively).** These bits control the digital driver even when the corresponding bit in register P1MDIN is logic 0. 0: Corresponding P1.n Output is open-drain. 1: Corresponding P1.n Output is push-pull.

SFR Definition 27.17. P1DRV: Port1 Drive Strength

Bit	7	6	5	4	3	2	1	0
Name	P1DRV[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0xF; SFR Address = 0xA5

Bit	Name	Function
7:0	P1DRV[7:0]	**Drive Strength Configuration Bits for P1.7–P1.0 (respectively).** Configures digital I/O Port cells to high or low output drive strength. 0: Corresponding P1.n Output has low output drive strength. 1: Corresponding P1.n Output has high output drive strength.

SFR Definition 27.20. P2MDIN: Port2 Input Mode

Bit	7	6	5	4	3	2	1	0
Name		P2MDIN[6:0]						
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0x0; SFR Address = 0xF3

Bit	Name	Function
7	Reserved	Read = 1b; Must Write 1b.
6:0	P2MDIN[3:0]	**Analog Configuration Bits for P2.6–P2.0 (respectively).** Port pins configured for analog mode have their weak pullup and digital receiver disabled. The digital driver is not explicitly disabled. 0: Corresponding P2.n pin is configured for analog mode. 1: Corresponding P2.n pin is not configured for analog mode.

SFR Definition 27.21. P2MDOUT: Port2 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P2MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0xA6

Bit	Name	Function
7:0	P2MDOUT[7:0]	**Output Configuration Bits for P2.7–P2.0 (respectively).** These bits control the digital driver even when the corresponding bit in register P2MDIN is logic 0. 0: Corresponding P2.n Output is open-drain. 1: Corresponding P2.n Output is push-pull.

SFR Definition 27.28. P4MDIN: Port4 Input Mode

Bit	7	6	5	4	3	2	1	0
Name	P4MDIN[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0xF; SFR Address = 0xF2

Bit	Name	Function
7:0	P4MDIN[3:0]	**Analog Configuration Bits for P4.7–P4.0 (respectively).** Port pins configured for analog mode have their weak pullup and digital receiver disabled. The digital driver is not explicitly disabled. 0: Corresponding P4.n pin is configured for analog mode. 1: Corresponding P4.n pin is not configured for analog mode.

SFR Definition 27.29. P4MDOUT: Port4 Output Mode

Bit	7	6	5	4	3	2	1	0
Name	P4MDOUT[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0xF; SFR Address = 0xF9

Bit	Name	Function
7:0	P4MDOUT[7:0]	**Output Configuration Bits for P4.7–P4.0 (respectively).** These bits control the digital driver even when the corresponding bit in register P4MDIN is logic 0. 0: Corresponding P4.n Output is open-drain. 1: Corresponding P4.n Output is push-pull.

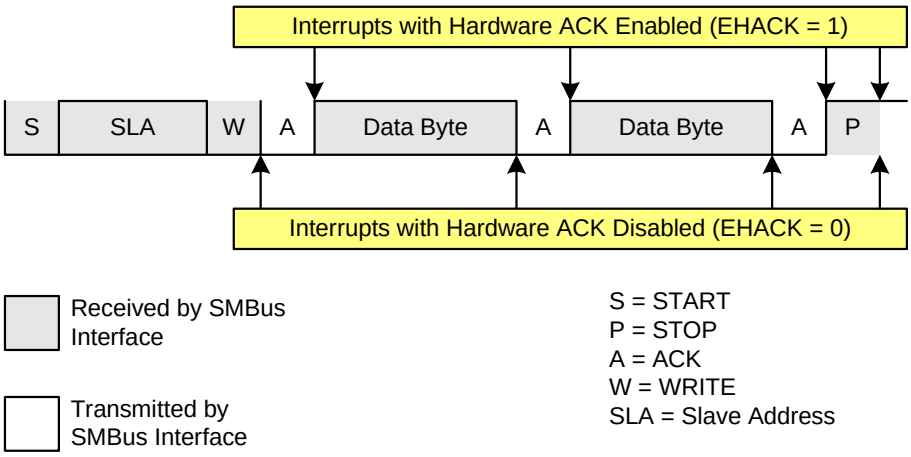


Figure 28.7. Typical Slave Write Sequence

28.5.4. Read Sequence (Slave)

During a read sequence, an SMBus master reads data from a slave device. The slave in this transfer will be a receiver during the address byte, and a transmitter during all data bytes. When slave events are enabled (INH = 0), the interface enters Slave Receiver Mode (to receive the slave address) when a START followed by a slave address and direction bit (READ in this case) is received. If hardware ACK generation is disabled, upon entering Slave Receiver Mode, an interrupt is generated and the ACKRQ bit is set. The software must respond to the received slave address with an ACK, or ignore the received slave address with a NACK. If hardware ACK generation is enabled, the hardware will apply the ACK for a slave address which matches the criteria set up by SMB0ADR and SMB0ADM. The interrupt will occur after the ACK cycle.

If the received slave address is ignored (by software or hardware), slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are transmitted. If the received slave address is acknowledged, data should be written to SMB0DAT to be transmitted. The interface enters Slave Transmitter Mode, and transmits one or more bytes of data. After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (an error condition may be generated if SMB0DAT is written following a received NACK while in Slave Transmitter Mode). The interface exits Slave Transmitter Mode after receiving a STOP. Note that the interface will switch to Slave Receiver Mode if SMB0DAT is not written following a Slave Transmitter interrupt. Figure 28.8 shows a typical slave read sequence. Two transmitted data bytes are shown, though any number of bytes may be transmitted. All of the “data byte transferred” interrupts occur **after** the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

SFR Definition 32.4. TL0: Timer 0 Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TL0[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0x8A

Bit	Name	Function
7:0	TL0[7:0]	**Timer 0 Low Byte.** The TL0 register is the low byte of the 16-bit Timer 0.

SFR Definition 32.5. TL1: Timer 1 Low Byte

Bit	7	6	5	4	3	2	1	0
Name	TL1[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0x0; SFR Address = 0x8B

Bit	Name	Function
7:0	TL1[7:0]	**Timer 1 Low Byte.** The TL1 register is the low byte of the 16-bit Timer 1.

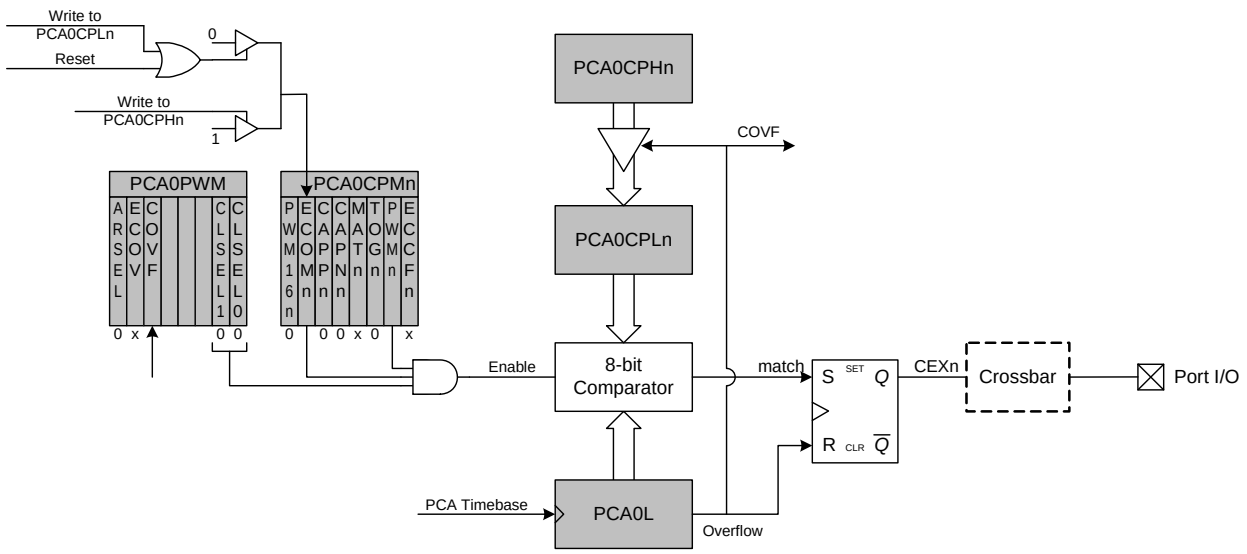


Figure 33.8. PCA 8-Bit PWM Mode Diagram

33.3.5.2. 9/10/11-bit Pulse Width Modulator Mode

The duty cycle of the PWM output signal in 9/10/11-bit PWM mode should be varied by writing to an "Auto-Reload" Register, which is dual-mapped into the PCA0CPHn and PCA0CPLn register locations. The data written to define the duty cycle should be right-justified in the registers. The auto-reload registers are accessed (read or written) when the bit ARSEL in PCA0PWM is set to 1. The capture/compare registers are accessed when ARSEL is set to 0.

When the least-significant N bits of the PCA0 counter match the value in the associated module's capture/compare register (PCA0CPn), the output on CEXn is asserted high. When the counter overflows from the Nth bit, CEXn is asserted low (see Figure 33.9). Upon an overflow from the Nth bit, the COVF flag is set, and the value stored in the module's auto-reload register is loaded into the capture/compare register. The value of N is determined by the CLSEL bits in register PCA0PWM.

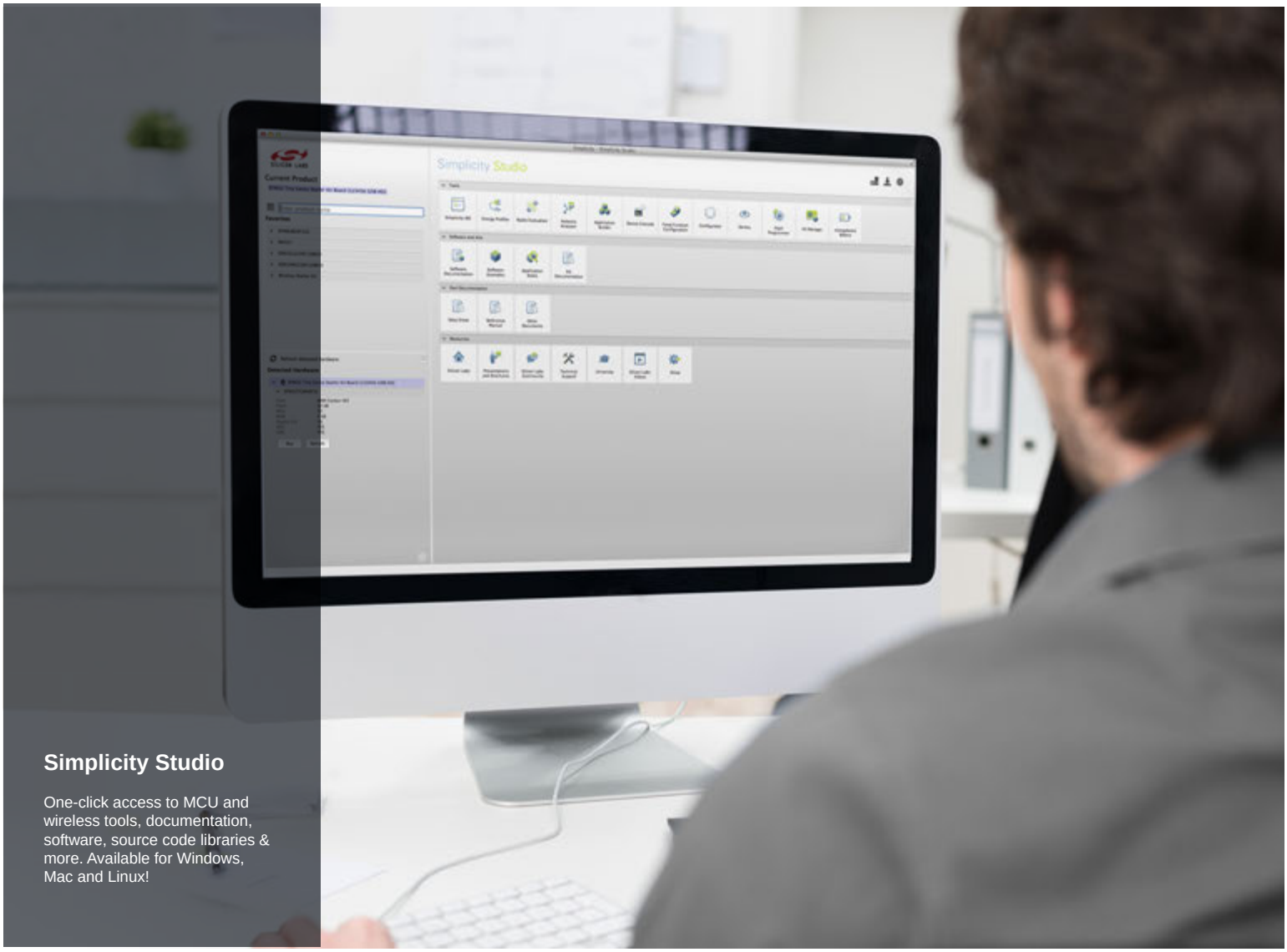
The 9, 10 or 11-bit PWM mode is selected by setting the ECOMn and PWMn bits in the PCA0CPMn register, and setting the CLSEL bits in register PCA0PWM to the desired cycle length (other than 8-bits). If the MATn bit is set to 1, the CCFn flag for the module will be set each time a comparator match (rising edge) occurs. The COVF flag in PCA0PWM can be used to detect the overflow (falling edge), which will occur every 512 (9-bit), 1024 (10-bit) or 2048 (11-bit) PCA clock cycles. The duty cycle for 9/10/11-Bit PWM Mode is given in Equation 33.3, where N is the number of bits in the PWM cycle.

Important Note About PCA0CPHn and PCA0CPLn Registers: When writing a 16-bit value to the PCA0CPn registers, the low byte should always be written first. Writing to PCA0CPLn clears the ECOMn bit to 0; writing to PCA0CPHn sets ECOMn to 1.

$$\text{Duty Cycle} = \frac{(2^N - PCA0CPn)}{2^N}$$

Equation 33.3. 9, 10, and 11-Bit PWM Duty Cycle

A 0% duty cycle may be generated by clearing the ECOMn bit to 0.



Simplicity Studio

One-click access to MCU and wireless tools, documentation, software, source code libraries & more. Available for Windows, Mac and Linux!

IoT Portfolio
www.silabs.com/IoT

SW/HW
www.silabs.com/simplicity

Quality
www.silabs.com/quality

Support and Community
community.silabs.com

SILICON LABS®

Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

http://www.silabs.com