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	57
Program Memory Size	64KB (64K 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/c8051f964-b-gqr

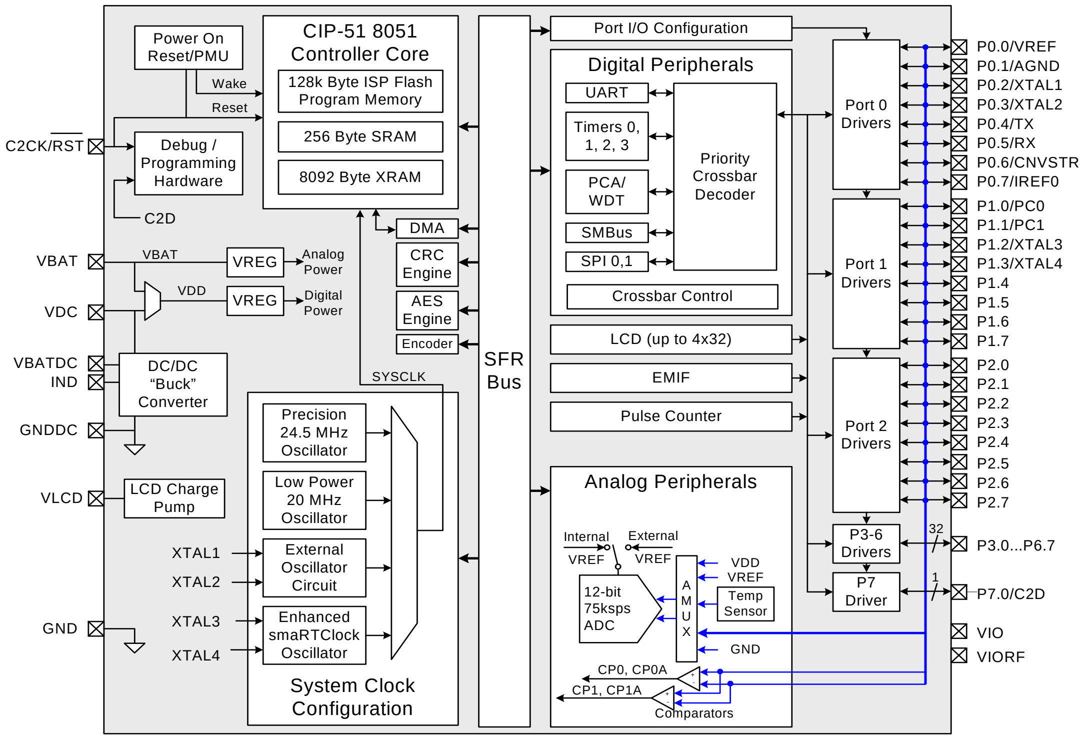


Figure 1.1. C8051F960 Block Diagram

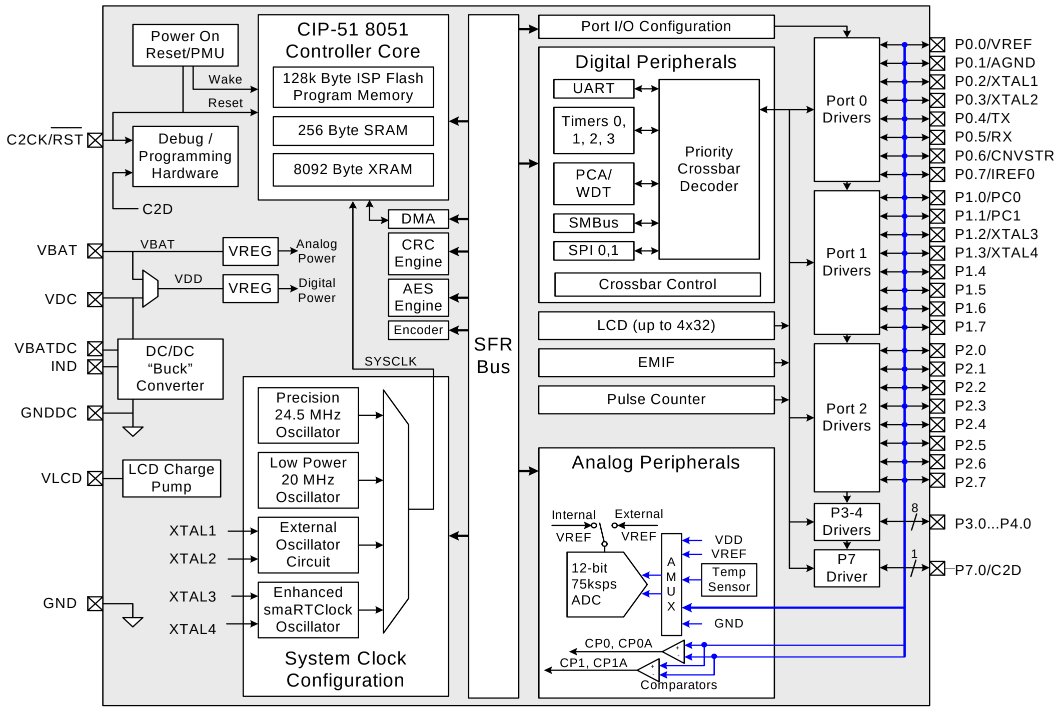


Figure 1.2. C8051F961 Block Diagram

1.5. SAR ADC with 16-bit Auto-Averaging Accumulator and Autonomous Low Power Burst Mode

The ADC0 on C8051F96x devices is a 300 ksps, 10-bit or 75 ksps, 12-bit successive-approximation-register (SAR) ADC with integrated track-and-hold and programmable window detector. ADC0 also has an autonomous low power Burst Mode which can automatically enable ADC0, capture and accumulate samples, then place ADC0 in a low power shutdown mode without CPU intervention. It also has a 16-bit accumulator that can automatically oversample and average the ADC results. See Section “5.4. 12-Bit Mode” on page 84 for more details on using the ADC in 12-bit mode.

The ADC is fully configurable under software control via Special Function Registers. The ADC0 operates in single-ended mode and may be configured to measure various different signals using the analog multiplexer described in Section “5.7. ADC0 Analog Multiplexer” on page 95. The voltage reference for the ADC is selected as described in Section “5.9. Voltage and Ground Reference Options” on page 100.

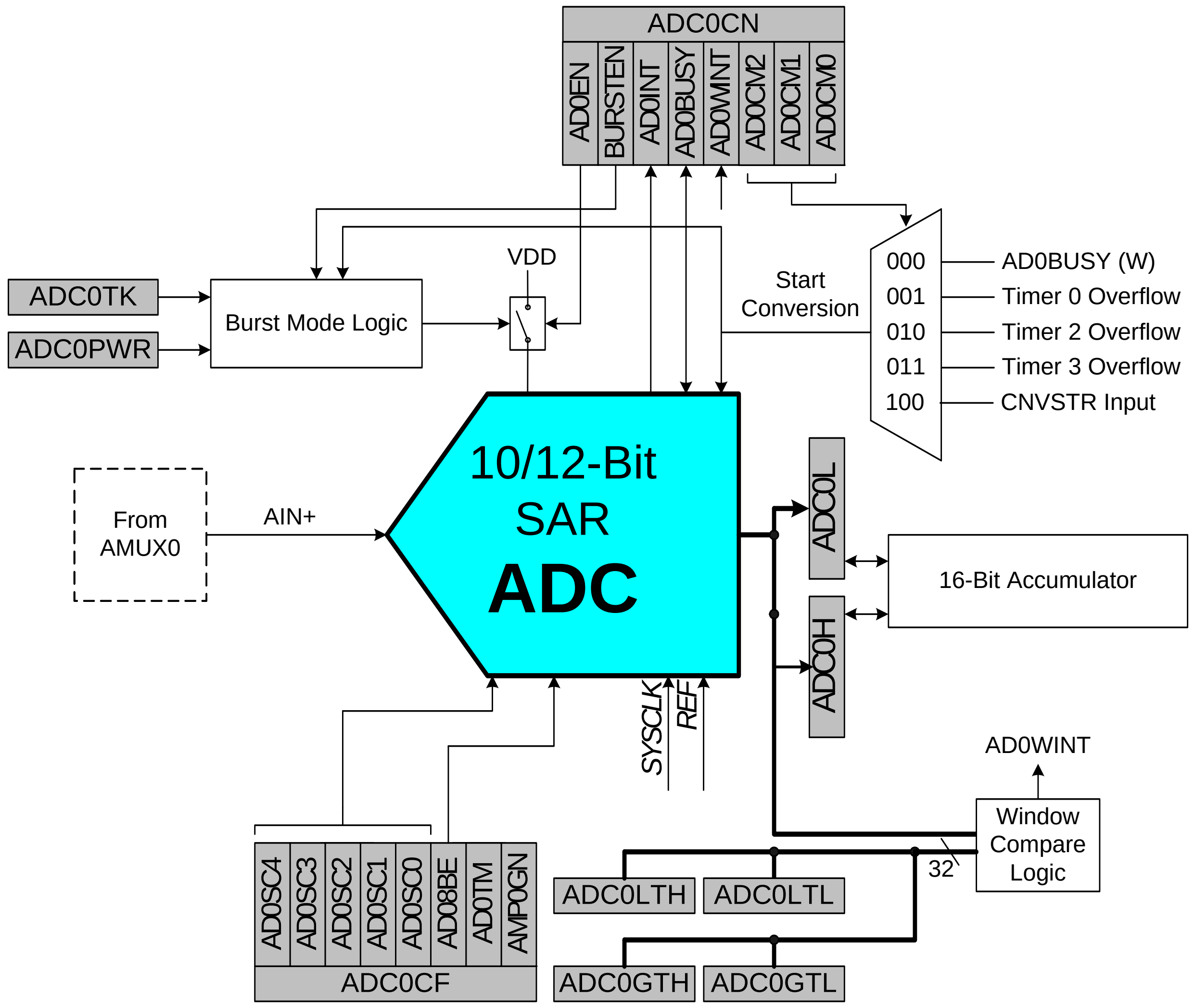
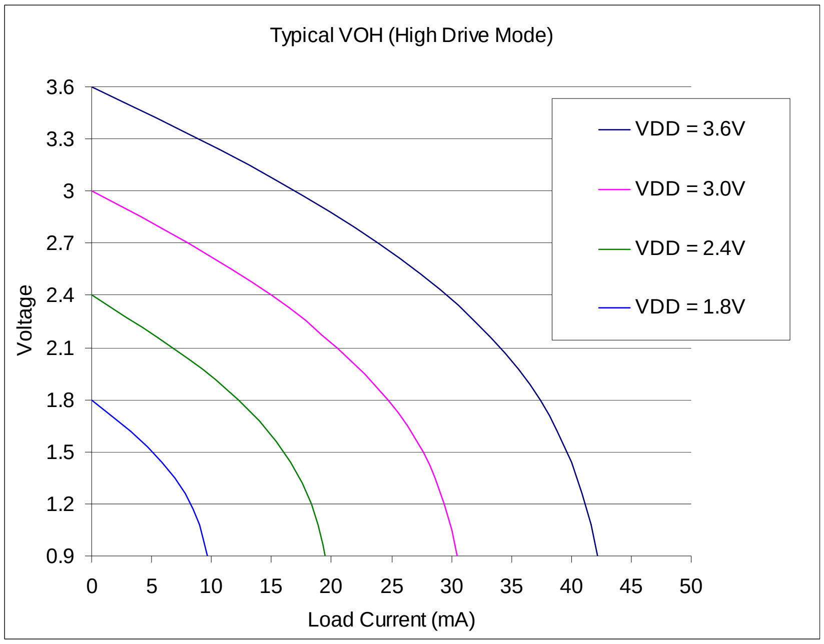


Figure 1.13. ADC0 Functional Block Diagram



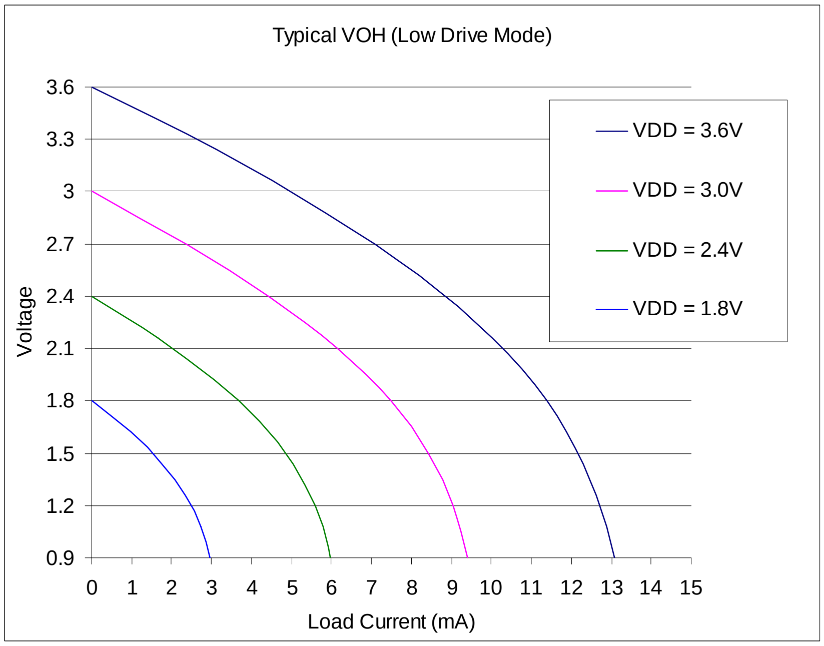


Figure 4.2. Typical VOH Curves, 1.8–3.6 V

SFR Definition 7.5. CPT0MX: Comparator0 Input Channel Select

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	CMX0N[3:0]				CMX0P[3:0]			
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	1

SFR Page = 0x0; SFR Address = 0x9F

Bit	**Name**	**Function**			
7:4	CMX0N	**Comparator0 Negative Input Selection.** Selects the negative input channel for Comparator0.			
		0000:	P0.1	1000:	P2.1
		0001:	P0.3	1001:	P2.3
		0010:	P0.5	1010:	Reserved
		0011:	Reserved	1011:	Reserved
		0100:	Reserved	1100:	Compare
		0101:	Reserved	1101:	VBAT divided by 2
		0110:	P1.5	1110:	Digital Supply Voltage
		0111:	P1.7	1111:	Ground
3:0	CMX0P	**Comparator0 Positive Input Selection.** Selects the positive input channel for Comparator0.			
		0000:	P0.0	1000:	P2.0
		0001:	P0.2	1001:	P2.2
		0010:	P0.4	1010:	Reserved
		0011:	P0.6	1011:	Reserved
		0100:	Reserved	1100:	Compare
		0101:	Reserved	1101:	VBAT divided by 2
		0110:	P1.4	1110:	VBAT Supply Voltage
		0111:	P1.6	1111:	VBAT Supply Voltage

Table 10.1. EMIF Pinout (C8051F960/2/4/6/8)

Multiplexed Mode		
Signal Name	**Port Pin**	
	8-Bit Mode[1]	**16-Bit Mode[2]**
$\overline{RD}$	P3.6	P3.6
$\overline{WR}$	P3.7	P3.7
ALE	P3.5	P3.5
AD0	P6.0	P6.0
AD1	P6.1	P6.1
AD2	P6.2	P6.2
AD3	P6.3	P6.3
AD4	P6.4	P6.4
AD5	P6.5	P6.5
AD6	P6.6	P6.6
AD7	P6.7	P6.7
A8	—	P5.0
A9	—	P5.1
A10	—	P5.2
A11	—	P5.3
A12	—	P5.4
A13	—	P5.5
A14	—	P5.6
A15	—	P5.7
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
—	—	—
Required I/O:	**11**	**19**

Non Multiplexed Mode		
Signal Name	**Port Pin**	
	8-Bit Mode [1]	**16-Bit Mode[2]**
$\overline{RD}$	P3.6	P3.6
$\overline{WR}$	P3.7	P3.7
D0	P6.0	P6.0
D1	P6.1	P6.1
D2	P6.2	P6.2
D3	P6.3	P6.3
D4	P6.4	P6.4
D5	P6.5	P6.5
D6	P6.6	P6.6
D7	P6.7	P6.7
A0	P5.0	P5.0
A1	P5.1	P5.1
A2	P5.2	P5.2
A3	P5.3	P5.3
A4	P5.4	P5.4
A5	P5.5	P5.5
A6	P5.6	P5.6
A7	P5.7	P5.7
A8	—	P4.0
A9	—	P4.1
A10	—	P4.2
A11	—	P4.3
A12	—	P4.4
A13	—	P4.5
A14	—	P4.6
A15	—	P4.7
Required I/O:	**18**	**26**

Notes:

1. Using 8-bit movx instruction without bank select.
2. Using 16-bit movx instruction.

11. Direct Memory Access (DMA0)

An on-chip direct memory access (DMA0) is included on the C8051F96x devices. The DMA0 subsystem allows autonomous variable-length data transfers between XRAM and peripheral SFR registers without CPU intervention. During DMA0 operation, the CPU is free to perform some other tasks. In order to save total system power consumption, the CPU and flash can be powered down. DMA0 improves the system performance and efficiency with high data throughput peripherals.

DMA0 contains seven independent channels, common control registers, and a DMA0 Engine (see Figure 11.1). Each channel includes a register that assigns a peripheral to the channel, a channel control register, and a set of SFRs that include XRAM address information and SFR address information used by the channel during a data transfer. The DMA0 architecture is described in detail in Section 11.1.

The DMA0 in C8051F96x devices supports four peripherals: AES0, ENC0, CRC1, and SPI1. Peripherals with DMA0 capability should be configured to work with the DMA0 through their own registers. The DMA0 provides up to seven channels, and each channel can be configured for one of nine possible data transfer functions:

- XRAM to ENC0L/M/H
- ENC0L/M/H sfrs to XRAM
- XRAM to CRC1IN sfr
- XRAM to SPI1DAT sfr
- SPI1DAT sfr to XRAM
- XRAM to AES0KIN sfr
- XRAM to AES0BIN sfr
- XRAM to AES0XIN sfr
- AES0YOUT sfr to XRAM

The DMA0 subsystem signals the MCU through a set of interrupt service routine flags. Interrupts can be generated when the DMA0 transfers half of the data length or full data length on any channel.

13.3. CRC Seed Value

Normally, the initial value or the CRC results is cleared to 0x0000. However, a CRC might be specified with an initial value preset to all ones (0xFFFF).

The steps to preset the CRC with all ones is as follows:

1. Set the SEED bit to 1.
2. Reset the CRC1 module by setting the CLR bit to 1 in CRC1CN.
3. Clear the SEED bit to 0.

The CRC1 module is now ready to calculate a CRC using a CRC seed value of 0xFFFF.

13.4. Inverting the Final Value

Sometimes it is necessary to invert the final value. This will take the ones complement of the final result.

The steps to flip the final CRC results are as follows:

1. Clear the CRC module by setting the CLR bit in CRC1CN SFR.
2. Write the polynomial to CRC1POLH:L.
3. Write all data bytes to CRC1IN.
4. Set the INV bit in the CRC1CN SFR to invert the final results.
5. Read the final CRC results from CRC1OUTH:L.

Clear the FLIP bit in the CRC1CN SFR.

13.5. Flipping the Final Value

The steps to flip the final CRC results are as follows:

1. Clear the CRC module by setting the CLR bit in CRC1CN SFR.
2. Write the polynomial to CRC1POLH:L.
3. Write all data bytes to CRC1IN.
4. Set the FLIP bit in the CRC1CN SFR to flip the final results.
5. Read the final CRC results from CRC1OUTH:L.
6. Clear the FLIP bit in the CRC1CN SFR.

The flip operation will exchange bit 15 with bit 0, bit 14 with bit 1, bit 13 with bit 2, and so on.

14.5. AES Block Cipher Decryption

14.5.1. AES Block Cipher 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 and data to be decryption in xram.
- Reset AES module by clearing bit 2 of AES0BCFG.
- sable the first three DMA channels by clearing bits 0 to 2 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 AES0YOUT sfr
  - Select the third DMA channel by writing 0x02 to the DMA0SEL sfr
  - Configure the third 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 DMA0NSZH sfr
  - Clear the DMA0NAOH and DMA0NAOL sfrs.
- Clear first three DMA interrupts by clearing bits 0 to 2 in the DMA0INT sfr.
- Enable first three DMA channels setting bits 0 to 2 in the DMA0EN sfr
- Configure the AES Module data flow for AES Block Cipher by writing 0x00 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 encryption operation be setting bit 3 of AES0BCFG
- Wait on the DMA interrupt from DMA channel 2
- Disable the AES Module by clearing bit 2 of AES0BCFG
- Disable the DMA by writing 0x00 to DMA0EN

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 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 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

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 18.3. PSCTL: Program Store R/W Control

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name							PSEE	PSWE
Type	R	R	R	R	R	R	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Page =0x0; SFR Address = 0x8F

Bit	**Name**	**Function**
7:2	Unused	Read = 000000b, Write = don't care.
1	PSEE	**Program Store Erase Enable.** Setting this bit (in combination with PSWE) allows an entire page of flash program memory to be erased. If this bit is logic 1 and flash writes are enabled (PSWE is logic 1), a write to flash memory using the MOVX instruction will erase the entire page that contains the location addressed by the MOVX instruction. The value of the data byte written does not matter. 0: Flash program memory erasure disabled. 1: Flash program memory erasure enabled.
0	PSWE	**Program Store Write Enable.** Setting this bit allows writing a byte of data to the flash program memory using the MOVX write instruction. The flash location should be erased before writing data. 0: Writes to flash program memory disabled. 1: Writes to flash program memory enabled; the MOVX write instruction targets flash memory.

22.7. Flash Error Reset

If a Flash read/write/erase or program read targets an illegal address, a system reset is generated. This may occur due to any of the following:

- A Flash write or erase is attempted above user code space. This occurs when PSWE is set to 1 and a MOVX write operation targets an address above the Lock Byte address.
- A Flash read is attempted above user code space. This occurs when a MOVC operation targets an address above the Lock Byte address.
- A Program read is attempted above user code space. This occurs when user code attempts to branch to an address above the Lock Byte address.
- A Flash read, write or erase attempt is restricted due to a Flash security setting (see Section “18.3. Security Options” on page 247).
- A Flash write or erase is attempted while the V_{DD} Monitor is disabled.

The FERROR bit (RSTSRC.6) is set following a Flash error reset. The state of the $\overline{\text{RST}}$ pin is unaffected by this reset.

22.8. SmaRTClock (Real Time Clock) Reset

The SmaRTClock can generate a system reset on two events: SmaRTClock Oscillator Fail or SmaRTClock Alarm. The SmaRTClock Oscillator Fail event occurs when the SmaRTClock Missing Clock Detector is enabled and the SmaRTClock clock is below approximately 20 kHz. A SmaRTClock alarm event occurs when the SmaRTClock Alarm is enabled and the SmaRTClock timer value matches the ALARMn registers. The SmaRTClock can be configured as a reset source by writing a 1 to the RTC0RE flag (RSTSRC.7). The SmaRTClock reset remains functional even when the device is in the low power Suspend or Sleep mode. The state of the $\overline{\text{RST}}$ pin is unaffected by this reset.

22.9. Software Reset

Software may force a reset by writing a 1 to the SWRSF bit (RSTSRC.4). The SWRSF bit will read 1 following a software forced reset. The state of the $\overline{\text{RST}}$ pin is unaffected by this reset.

23.4. Special Function Registers for Selecting and Configuring the System Clock

The clocking sources on C8051F96x devices are enabled and configured using the OSCICN, OSCICL, OSCXCN and the SmaRTClock internal registers. See Section “24. SmaRTClock (Real Time Clock)” on page 295 for SmaRTClock register descriptions. The system clock source for the MCU can be selected using the CLKSEL register. To minimize active mode current, the oneshot timer which sets Flash read time should by bypassed when the system clock is greater than 10 MHz. See the FLSCL register description for details.

The clock selected as the system clock can be divided by 1, 2, 4, 8, 16, 32, 64, or 128. When switching between two clock divide values, the transition may take up to 128 cycles of the undivided clock source. The CLKRDY flag can be polled to determine when the new clock divide value has been applied. The clock divider must be set to "divide by 1" when entering Suspend or Sleep Mode.

The system clock source may also be switched on-the-fly. The switchover takes effect after one clock period of the slower oscillator.

SFR Definition 23.1. CLKSEL: Clock Select

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

SFR Page = All Pages; SFR Address = 0xA9

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

27.2.2. Assigning Port I/O Pins to Digital Functions

Any Port pins not assigned to analog functions may be assigned to digital functions or used as GPIO. Most digital functions rely on the Crossbar for pin assignment; however, some digital functions bypass the Crossbar in a manner similar to the analog functions listed above. **Port pins used by these digital functions and any Port pins selected for use as GPIO should have their corresponding bit in PnSKIP set to 1.** Table 27.2 shows all available digital functions and the potential mapping of Port I/O to each digital function.

Table 27.2. Port I/O Assignment for Digital Functions

Digital Function	Potentially Assignable Port Pins	SFR(s) used for Assignment
UART0, SPI0, SPI1, SMBus, CP0 and CP1 Outputs, System Clock Output, PCA0, Timer0 and Timer1 External Inputs.	Any Port pin available for assignment by the Crossbar. This includes P0.0–P2.7 pins which have their PnSKIP bit set to 0. **Note:** The Crossbar will always assign UART0 and SPI1 pins to fixed locations.	XBR0, XBR1, XBR2
Any pin used for GPIO	P0.0–P7.0	P0SKIP, P1SKIP, P2SKIP
External Memory Interface	P3.6–P6.7	EMI0CF

27.2.3. Assigning Port I/O Pins to External Digital Event Capture Functions

External digital event capture functions can be used to trigger an interrupt or wake the device from a low power mode when a transition occurs on a digital I/O pin. The digital event capture functions do not require dedicated pins and will function on both GPIO pins (PnSKIP = 1) and pins in use by the Crossbar (PnSKIP = 0). External digital even capture functions cannot be used on pins configured for analog I/O. Table 27.3 shows all available external digital event capture functions.

Table 27.3. Port I/O Assignment for External Digital Event Capture Functions

Digital Function	Potentially Assignable Port Pins	SFR(s) used for Assignment
External Interrupt 0	P0.0–P0.5, P1.6, P1.7	IT01CF
External Interrupt 1	P0.0–P0.4, P1.6, P1.7	IT01CF
Port Match	P0.0–P1.7	P0MASK, P0MAT P1MASK, P1MAT

SFR Definition 27.26. P3DRV: Port3 Drive Strength

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	P3DRV[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0xF; SFR Address = 0xA1

Bit	Name	Function
7:0	P3DRV[7:0]	**Drive Strength Configuration Bits for P3.7–P3.0 (respectively).** Configures digital I/O Port cells to high or low output drive strength. 0: Corresponding P3.n Output has low output drive strength. 1: Corresponding P3.n Output has high output drive strength.

SFR Definition 27.27. P4: Port4

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Name	P4[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0xF; SFR Address = 0xD9

Bit	Name	Description	Write	Read
7:0	P4[7:0]	**Port 4 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: P4.n Port pin is logic LOW. 1: P4.n Port pin is logic HIGH.

SFR Definition 27.34. P5DRV: Port5 Drive Strength

Bit	7	6	5	4	3	2	1	0
Name	P5DRV[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Page = 0xF; SFR Address = 0xA3

Bit	Name	Function
7:0	P5DRV[7:0]	**Drive Strength Configuration Bits for P5.7–P5.0 (respectively).** Configures digital I/O Port cells to high or low output drive strength. 0: Corresponding P5.n Output has low output drive strength. 1: Corresponding P5.n Output has high output drive strength.

SFR Definition 27.35. P6: Port6

Bit	7	6	5	4	3	2	1	0
Name	P6[7:0]							
Type	R/W							
Reset	1	1	1	1	1	1	1	1

SFR Page = 0xF; SFR Address = 0xDB

Bit	Name	Description	Write	Read
7:0	P6[7:0]	**Port 6 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: P6.n Port pin is logic LOW. 1: P6.n Port pin is logic HIGH.

28.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see SFR Definition 28.2). The higher four bits of SMB0CN (MASTER, TXMODE, STA, and STO) form a status vector that can be used to jump to service routines. MASTER indicates whether a device is the master or slave during the current transfer. TXMODE indicates whether the device is transmitting or receiving data for the current byte.

STA and STO indicate that a START and/or STOP has been detected or generated since the last SMBus interrupt. STA and STO are also used to generate START and STOP conditions when operating as a master. Writing a 1 to STA will cause the SMBus interface to enter Master Mode and generate a START when the bus becomes free (STA is not cleared by hardware after the START is generated). Writing a 1 to STO while in Master Mode will cause the interface to generate a STOP and end the current transfer after the next ACK cycle. If STO and STA are both set (while in Master Mode), a STOP followed by a START will be generated.

The ARBLOST bit indicates that the interface has lost an arbitration. This may occur anytime the interface is transmitting (master or slave). A lost arbitration while operating as a slave indicates a bus error condition. ARBLOST is cleared by hardware each time SI is cleared.

The SI bit (SMBus Interrupt Flag) is set at the beginning and end of each transfer, after each byte frame, or when an arbitration is lost; see Table 28.3 for more details.
Important Note About the SI Bit: The SMBus interface is stalled while SI is set; thus SCL is held low, and the bus is stalled until software clears SI.

28.4.2.1. Software ACK Generation

When the EHACK bit in register SMB0ADM is cleared to 0, the firmware on the device must detect incoming slave addresses and ACK or NACK the slave address and incoming data bytes. As a receiver, writing the ACK bit defines the outgoing ACK value; as a transmitter, reading the ACK bit indicates the value received during the last ACK cycle. ACKRQ is set each time a byte is received, indicating that an outgoing ACK value is needed. When ACKRQ is set, software should write the desired outgoing value to the ACK bit before clearing SI. A NACK will be generated if software does not write the ACK bit before clearing SI. SDA will reflect the defined ACK value immediately following a write to the ACK bit; however SCL will remain low until SI is cleared. If a received slave address is not acknowledged, further slave events will be ignored until the next START is detected.

28.4.2.2. Hardware ACK Generation

When the EHACK bit in register SMB0ADM is set to 1, automatic slave address recognition and ACK generation is enabled. More detail about automatic slave address recognition can be found in Section 28.4.3. As a receiver, the value currently specified by the ACK bit will be automatically sent on the bus during the ACK cycle of an incoming data byte. As a transmitter, reading the ACK bit indicates the value received on the last ACK cycle. The ACKRQ bit is not used when hardware ACK generation is enabled. If a received slave address is NACKed by hardware, further slave events will be ignored until the next START is detected, and no interrupt will be generated.
Table 28.3 lists all sources for hardware changes to the SMB0CN bits. Refer to Table 28.5 for SMBus status decoding using the SMB0CN register.

32.2.2. 8-bit Timers with Auto-Reload

When T2SPLIT is set, Timer 2 operates as two 8-bit timers (TMR2H and TMR2L). Both 8-bit timers operate in auto-reload mode as shown in Figure 32.5. TMR2RLL holds the reload value for TMR2L; TMR2RLH holds the reload value for TMR2H. The TR2 bit in TMR2CN handles the run control for TMR2H. TMR2L is always running when configured for 8-bit Mode.

Each 8-bit timer may be configured to use SYSCLK, SYSCLK divided by 12, SmaRTClock divided by 8 or Comparator 0 output. The Timer 2 Clock Select bits (T2MH and T2ML in CKCON) select either SYSCLK or the clock defined by the Timer 2 External Clock Select bits (T2XCLK[1:0] in TMR2CN), as follows:

T2MH	T2XCLK[1:0]	TMR2H Clock Source
0	00	SYSCLK / 12
0	01	SmaRTClock / 8
0	10	Reserved
0	11	Comparator 0
1	X	SYSCLK

T2ML	T2XCLK[1:0]	TMR2L Clock Source
0	00	SYSCLK / 12
0	01	SmaRTClock / 8
0	10	Reserved
0	11	Comparator 0
1	X	SYSCLK

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

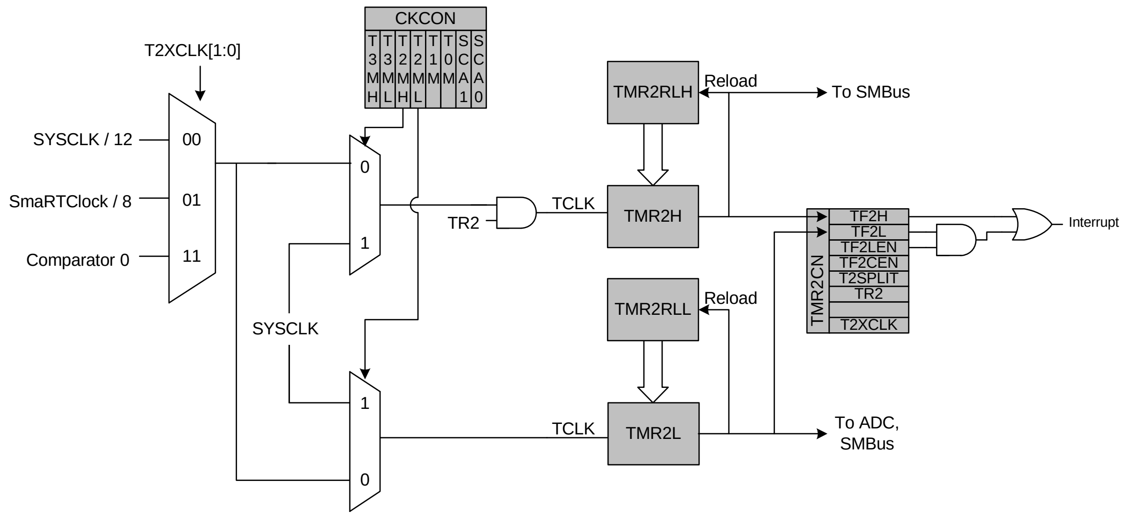


Figure 32.5. Timer 2 8-Bit Mode Block Diagram

32.2.3. Comparator 0/SmaRTClock Capture Mode

The Capture Mode in Timer 2 allows either Comparator 0 or the SmaRTClock period to be measured against the system clock or the system clock divided by 12. Comparator 0 and the SmaRTClock period can also be compared against each other. Timer 2 Capture Mode is enabled by setting TF2CEN to 1. Timer 2 should be in 16-bit auto-reload mode when using Capture Mode.

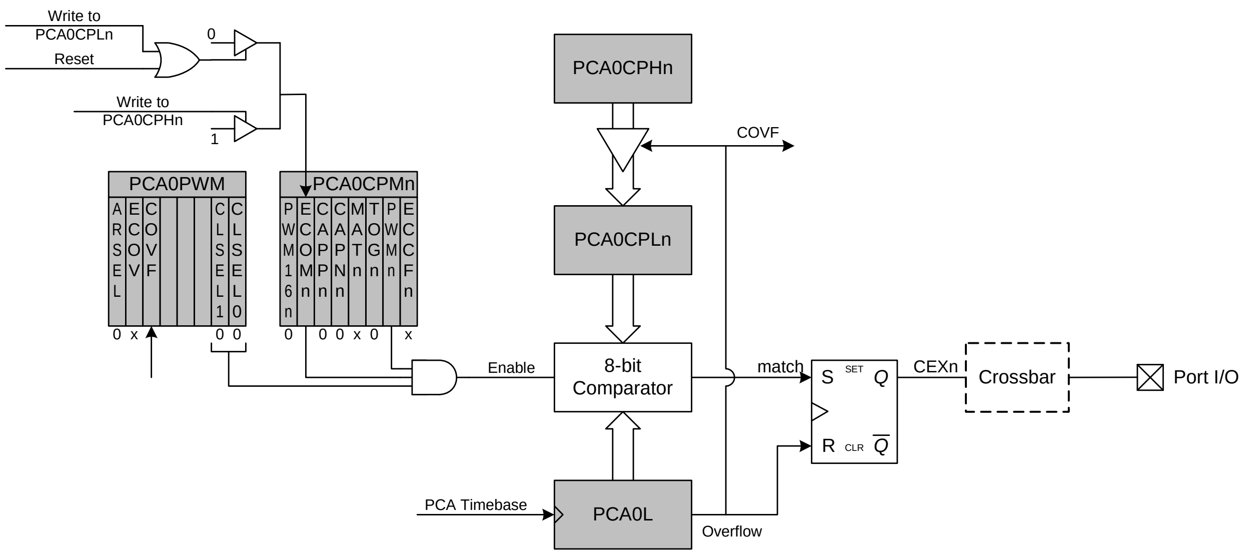


Figure 33.8. PCA 8-Bit PWM Mode Diagram

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.