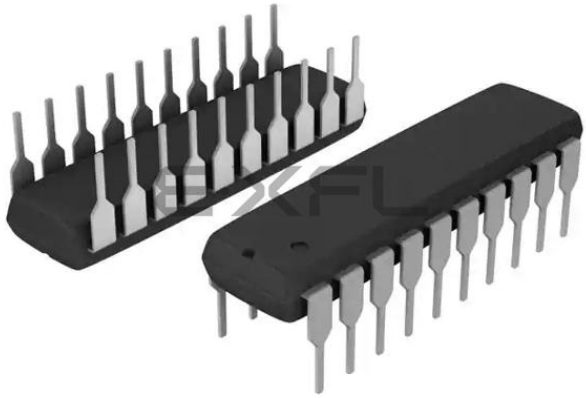


Welcome to E-XFL.COM

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	SMBus (2-Wire/I²C), SPI, UART/USART
Peripherals	POR, PWM, Temp Sensor, WDT
Number of I/O	17
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	768 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 16x10b; D/A 1x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	20-DIP
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f330d

Table of Contents

1. System Overview

C8051F330/1, C8051F330D devices are fully integrated mixed-signal System-on-a-Chip MCUs. Highlighted features are listed below. Refer to Table 1.1 for specific product feature selection.

- High-speed pipelined 8051-compatible microcontroller core (up to 25 MIPS)
- In-system, full-speed, non-intrusive debug interface (on-chip)
- True 10-bit 200 ksps 16-channel single-ended/differential ADC with analog multiplexer
- 10-bit Current Output DAC
- Precision programmable 25 MHz internal oscillator
- 8k bytes of on-chip Flash memory—512 bytes are reserved
- 768 bytes of on-chip RAM
- SMBus/I2C, Enhanced UART, and Enhanced SPI serial interfaces implemented in hardware
- Four general-purpose 16-bit timers
- Programmable Counter/Timer Array (PCA) with three capture/compare modules and Watchdog Timer function
- On-chip Power-On Reset, V_{DD} Monitor, and Temperature Sensor
- On-chip Voltage Comparator
- 17 Port I/O (5 V tolerant)

With on-chip Power-On Reset, V_{DD} monitor, Watchdog Timer, and clock oscillator, the C8051F330/1, C8051F330D devices are truly stand-alone System-on-a-Chip solutions. The Flash memory can be reprogrammed even in-circuit, providing non-volatile data storage, and also allowing field upgrades of the 8051 firmware. User software has complete control of all peripherals, and may individually shut down any or all peripherals for power savings.

The on-chip Silicon Labs 2-Wire (C2) Development Interface allows non-intrusive (uses no on-chip resources), full speed, in-circuit debugging using the production MCU installed in the final application. This debug logic supports inspection and modification of memory and registers, setting breakpoints, single stepping, run and halt commands. All analog and digital peripherals are fully functional while debugging using C2. The two C2 interface pins can be shared with user functions, allowing in-system debugging without occupying package pins.

Each device is specified for 2.7 V-to-3.6 V operation over the industrial temperature range (–40 to +85 °C). The Port I/O and /RST pins are tolerant of input signals up to 5 V. The C8051F330/1 are available in a 20-pin MLP package and the C8051F330D is available in a 20-pin DIP package. Block diagrams are included in Figure 1.1 and Figure 1.2.

5.4. Programmable Window Detector

The ADC Programmable Window Detector continuously compares the ADC0 output registers to user-programmed limits, and notifies the system when a desired condition is detected. This is especially effective in an interrupt-driven system, saving code space and CPU bandwidth while delivering faster system response times. The window detector interrupt flag (AD0WINT in register ADC0CN) can also be used in polled mode. The ADC0 Greater-Than (ADC0GTH, ADC0GTL) and Less-Than (ADC0LTH, ADC0LTL) registers hold the comparison values. The window detector flag can be programmed to indicate when measured data is inside or outside of the user-programmed limits, depending on the contents of the ADC0 Less-Than and ADC0 Greater-Than registers.

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xC4

Bits7-0: High byte of ADC0 Greater-Than Data Word

Figure 5.11. ADC0GTH: ADC0 Greater-Than Data High Byte Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xC3

Bits7-0: Low byte of ADC0 Greater-Than Data Word

Figure 5.12. ADC0GTL: ADC0 Greater-Than Data Low Byte Register

6.1.2. Update Output Based on Timer Overflow

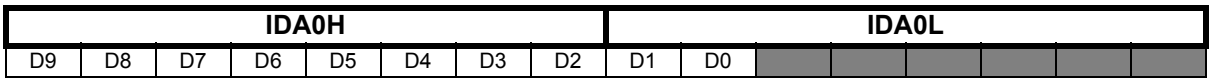
Similar to the ADC operation, in which an ADC conversion can be initiated by a timer overflow independently of the processor, the IDAC outputs can use a Timer overflow to schedule an output update event. This feature is useful in systems where the IDAC is used to generate a waveform of a defined sampling rate by eliminating the effects of variable interrupt latency and instruction execution on the timing of the IDAC output. When the IDA0CM bits (IDA0CN.[6:4]) are set to ‘000’, ‘001’, ‘010’ or ‘011’, writes to both IDAC data registers (IDA0L and IDA0H) are held until an associated Timer overflow event (Timer 0, Timer 1, Timer 2 or Timer 3, respectively) occurs, at which time the IDA0H:IDA0L contents are copied to the IDAC input latches, allowing the IDAC output to change to the new value.

6.1.3. Update Output Based on CNVSTR Edge

The IDAC output can also be configured to update on a rising edge, falling edge, or both edges of the external CNVSTR signal. When the IDA0CM bits (IDA0CN.[6:4]) are set to ‘100’, ‘101’, or ‘110’, writes to both IDAC data registers (IDA0L and IDA0H) are held until an edge occurs on the CNVSTR input pin. The particular setting of the IDA0CM bits determines whether IDAC outputs are updated on rising, falling, or both edges of CNVSTR. When a corresponding edge occurs, the IDA0H:IDA0L contents are copied to the IDAC input latches, allowing the IDAC output to change to the new value.

6.2. IDAC Output Mapping

The IDAC data registers (IDA0H and IDA0L) are left-justified, meaning that the eight MSBs of the IDAC output word are mapped to bits 7-0 of the IDA0H register, and the two LSBs of the IDAC output word are mapped to bits 7 and 6 of the IDA0L register. The data word mapping for the IDAC is shown in Figure 6.2.



Input Data Word (D9 - D0)	Output Current IDA0OMD[1:0] = ‘1x’	Output Current IDA0OMD[1:0] = ‘01’	Output Current IDA0OMD[1:0] = ‘00’
0x000	0 mA	0 mA	0 mA
0x001	1/1024 x 2 mA	1/1024 x 1 mA	1/1024 x 0.5 mA
0x200	512/1024 x 2 mA	512/1024 x 1 mA	512/1024 x 0.5 mA
0x3FF	1023/1024 x 2 mA	1023/1024 x 1 mA	1023/1024 x 0.5 mA

Figure 6.2. IDA0 Data Word Mapping

The full-scale output current of the IDAC is selected using the IDA0OMD bits (IDA0CN[1:0]). By default, the IDAC is set to a full-scale output current of 2 mA. The IDA0OMD bits can also be configured to provide full-scale output currents of 1 mA or 0.5 mA, as shown in Figure 6.3.

The Comparator output can be polled in software, used as an interrupt source, and/or routed to a Port pin. When routed to a Port pin, the Comparator output is available asynchronous or synchronous to the system clock; the asynchronous output is available even in STOP mode (with no system clock active). When disabled, the Comparator output (if assigned to a Port I/O pin via the Crossbar) defaults to the logic low state, and its supply current falls to less than 100 nA. See **Section "14.1. Priority Crossbar Decoder" on page 115** for details on configuring Comparator outputs via the digital Crossbar. Comparator inputs can be externally driven from –0.25 V to (V_{DD}) + 0.25 V without damage or upset. The complete Comparator electrical specifications are given in Table 8.1.

The Comparator response time may be configured in software via the CPT0MD register (see Figure 8.5). Selecting a longer response time reduces the Comparator supply current. See Table 8.1 for complete timing and power consumption specifications.

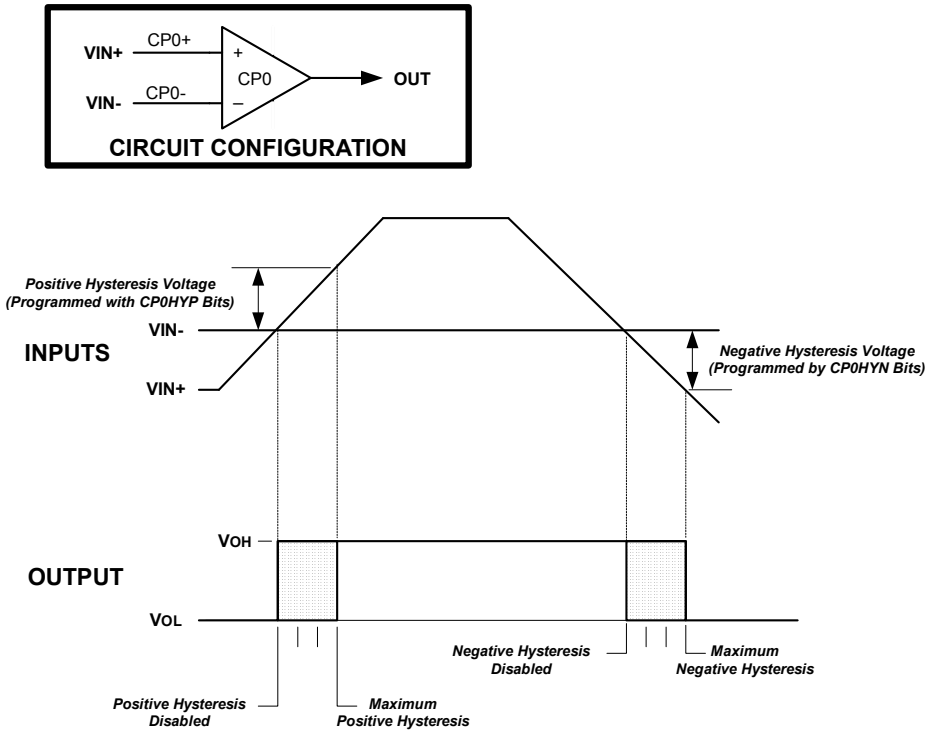


Figure 8.2. Comparator Hysteresis Plot

The Comparator hysteresis is software-programmable via its Comparator Control register CPT0CN. The user can program both the amount of hysteresis voltage (referred to the input voltage) and the positive and negative-going symmetry of this hysteresis around the threshold voltage.

The Comparator hysteresis is programmed using Bits3-0 in the Comparator Control Register CPT0CN (shown in Figure 8.3). The amount of negative hysteresis voltage is determined by the settings of the CP0HYN bits. As shown in Figure 8.2, settings of 20, 10 or 5 mV of negative hysteresis can be programmed, or negative hysteresis can be disabled. In a similar way, the amount of positive hysteresis is determined by the setting the CP0HYP bits.

Comparator interrupts can be generated on both rising-edge and falling-edge output transitions. (For Interrupt enable and priority control, see **Section "8.3. Interrupt Handler" on page 58**). The CP0FIF flag is set

9.3.5. Interrupt Register Descriptions

The SFRs used to enable the interrupt sources and set their priority level are described below. Refer to the data sheet section associated with a particular on-chip peripheral for information regarding valid interrupt conditions for the peripheral and the behavior of its interrupt-pending flag(s).

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
EA	ESPI0	ET2	ES0	ET1	EX1	ET0	EX0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address:
						(bit addressable)		0xA8

Bit7: EA: Enable All Interrupts.
This bit globally enables/disables all interrupts. It overrides the individual interrupt mask settings.
0: Disable all interrupt sources.
1: Enable each interrupt according to its individual mask setting.

Bit6: ESPI0: Enable Serial Peripheral Interface (SPI0) Interrupt.
This bit sets the masking of the SPI0 interrupts.
0: Disable all SPI0 interrupts.
1: Enable interrupt requests generated by SPI0.

Bit5: ET2: Enable Timer 2 Interrupt.
This bit sets the masking of the Timer 2 interrupt.
0: Disable Timer 2 interrupt.
1: Enable interrupt requests generated by the TF2L or TF2H flags.

Bit4: ES0: Enable UART0 Interrupt.
This bit sets the masking of the UART0 interrupt.
0: Disable UART0 interrupt.
1: Enable UART0 interrupt.

Bit3: ET1: Enable Timer 1 Interrupt.
This bit sets the masking of the Timer 1 interrupt.
0: Disable all Timer 1 interrupt.
1: Enable interrupt requests generated by the TF1 flag.

Bit2: EX1: Enable External Interrupt 1.
This bit sets the masking of External Interrupt 1.
0: Disable external interrupt 1.
1: Enable interrupt requests generated by the /INT1 input.

Bit1: ET0: Enable Timer 0 Interrupt.
This bit sets the masking of the Timer 0 interrupt.
0: Disable all Timer 0 interrupt.
1: Enable interrupt requests generated by the TF0 flag.

Bit0: EX0: Enable External Interrupt 0.
This bit sets the masking of External Interrupt 0.
0: Disable external interrupt 0.
1: Enable interrupt requests generated by the /INT0 input.

Figure 9.9. IE: Interrupt Enable

11. Flash Memory

On-chip, re-programmable Flash memory is included for program code and non-volatile data storage. The Flash memory can be programmed in-system, a single byte at a time, through the C2 interface or by software using the MOVX instruction. Once cleared to logic 0, a Flash bit must be erased to set it back to logic 1. Flash bytes would typically be erased (set to 0xFF) before being reprogrammed. The write and erase operations are automatically timed by hardware for proper execution; data polling to determine the end of the write/erase operation is not required. Code execution is stalled during a Flash write/erase operation. Refer to Table 11.1 for complete Flash memory electrical characteristics.

11.1. Programming The Flash Memory

The simplest means of programming the Flash memory is through the C2 interface using programming tools provided by Silicon Labs or a third party vendor. This is the only means for programming a non-initialized device. For details on the C2 commands to program Flash memory, see **Section "20. C2 Interface" on page 205**.

To ensure the integrity of Flash contents, it is strongly recommended that the on-chip V_{DD} Monitor be enabled in any system that includes code that writes and/or erases Flash memory from software.

11.1.1. Flash Lock and Key Functions

Flash writes and erases by user software are protected with a lock and key function. The Flash Lock and Key Register (FLKEY) must be written with the correct key codes, in sequence, before Flash operations may be performed. The key codes are: 0xA5, 0xF1. The timing does not matter, but the codes must be written in order. If the key codes are written out of order, or the wrong codes are written, Flash writes and erases will be disabled until the next system reset. Flash writes and erases will also be disabled if a Flash write or erase is attempted before the key codes have been written properly. The Flash lock resets after each write or erase; the key codes must be written again before a following Flash operation can be performed. The FLKEY register is detailed in Figure 11.4.

11.1.2. Flash Erase Procedure

The Flash memory can be programmed by software using the MOVX write instruction with the address and data byte to be programmed provided as normal operands. Before writing to Flash memory using MOVX, Flash write operations must be enabled by: (1) setting the PSWE Program Store Write Enable bit (PSCTL.0) to logic 1 (this directs the MOVX writes to target Flash memory); and (2) Writing the Flash key codes in sequence to the Flash Lock register (FLKEY). The PSWE bit remains set until cleared by software.

A write to Flash memory can clear bits to logic 0 but cannot set them; only an erase operation can set bits to logic 1 in Flash. **A byte location to be programmed should be erased before a new value is written.** The Flash memory is organized in 512-byte pages. The erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire 512-byte page, perform the following steps:

Step 1. Disable interrupts (recommended).
Step 2. Set thePSEE bit (register PSCTL).
Step 3. Set the PSWE bit (register PSCTL).
Step 4. Write the first key code to FLKEY: 0xA5.
Step 5. Write the second key code to FLKEY: 0xF1.
Step 6. Using the MOVX instruction, write a data byte to any location within the 512-byte page to be erased.
Step 7. Clear the PSWE and PSEE bits.

11.2. Non-volatile Data Storage

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

11.3. Security Options

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

A Security Lock Byte located at the last byte of Flash user space offers protection of the Flash program memory from access (reads, writes, or erases) by unprotected code or the C2 interface. The Flash security mechanism allows the user to lock *n* 512-byte Flash pages, starting at page 0 (addresses 0x0000 to 0x01FF), where *n* is the 1's complement number represented by the Security Lock Byte. **Note that the page containing the Flash Security Lock Byte is unlocked when no other Flash pages are locked (all bits of the Lock Byte are '1') and locked when any other Flash pages are locked (any bit of the Lock Byte is '0').** See example below.

Security Lock Byte:	11111101b
1s Complement:	00000010b
Flash pages locked:	3 (First two Flash pages + Lock Byte Page)
Addresses locked:	0x0000 to 0x03FF (first two Flash pages) and 0x1C00 to 0x1DFF (Lock Byte Page)

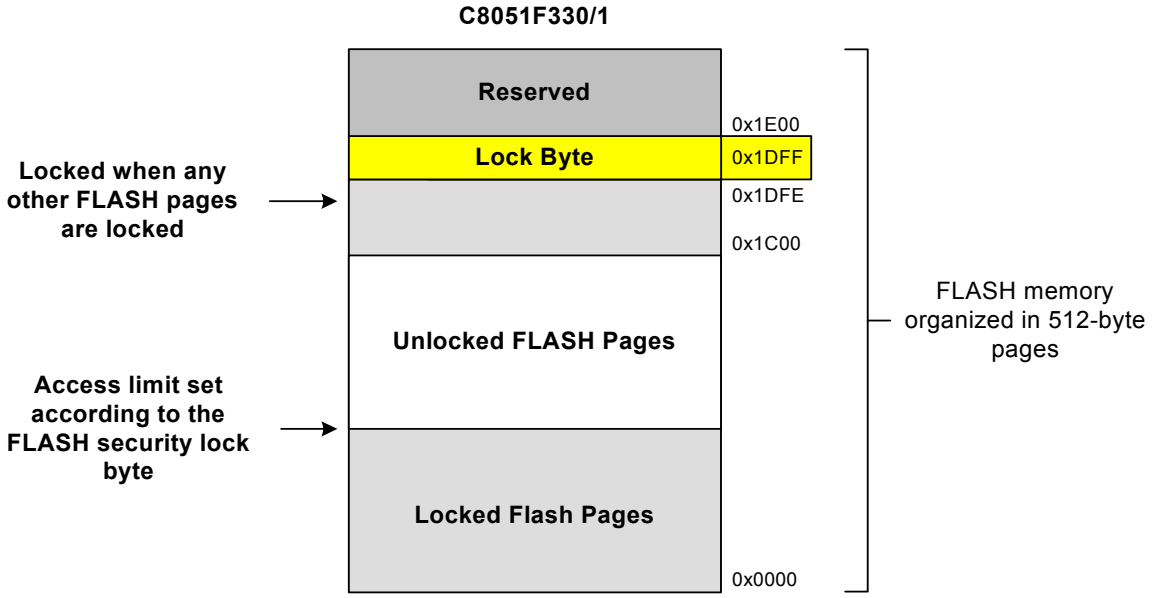


Figure 11.2. Flash Program Memory Map

R	R/W	R/W	R/W	R	R/W	R/W	R/W	Reset Value
XTLVLD	XOSCMD2	XOSCMD1	XOSCMD0	-	XFCN2	XFCN1	XFCN0	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xB1

Bit7: XTLVLD: Crystal Oscillator Valid Flag.
(Read only when XOSCMD = 11x.)
0: Crystal Oscillator is unused or not yet stable.
1: Crystal Oscillator is running and stable.

Bits6-4: XOSCMD2-0: External Oscillator Mode Bits.
00x: External Oscillator circuit off.
010: External CMOS Clock Mode.
011: External CMOS Clock Mode with divide by 2 stage.
100: RC Oscillator Mode.
101: Capacitor Oscillator Mode.
110: Crystal Oscillator Mode.
111: Crystal Oscillator Mode with divide by 2 stage.

Bit3: RESERVED. Read = 0, Write = don't care.

Bits2-0: XFCN2-0: External Oscillator Frequency Control Bits.
000-111: See table below:

XFCN	Crystal (XOSCMD = 11x)	RC (XOSCMD = 10x)	C (XOSCMD = 10x)
000	f ≤ 32 kHz	f ≤ 25 kHz	K Factor = 0.87
001	32 kHz < f ≤ 84 kHz	25 kHz < f ≤ 50 kHz	K Factor = 2.6
010	84 kHz < f ≤ 225 kHz	50 kHz < f ≤ 100 kHz	K Factor = 7.7
011	225 kHz < f ≤ 590 kHz	100 kHz < f ≤ 200 kHz	K Factor = 22
100	590 kHz < f ≤ 1.5 MHz	200 kHz < f ≤ 400 kHz	K Factor = 65
101	1.5 MHz < f ≤ 4 MHz	400 kHz < f ≤ 800 kHz	K Factor = 180
110	4 MHz < f ≤ 10 MHz	800 kHz < f ≤ 1.6 MHz	K Factor = 664
111	10 MHz < f ≤ 30 MHz	1.6 MHz < f ≤ 3.2 MHz	K Factor = 1590

CRYSTAL MODE (Circuit from Figure 13.1, Option 1; XOSCMD = 11x)
Choose XFCN value to match crystal frequency.

RC MODE (Circuit from Figure 13.1, Option 2; XOSCMD = 10x)
Choose XFCN value to match frequency range:
$f = 1.23(10^3) / (R * C)$, where
f = frequency of clock in MHz
C = capacitor value in pF
R = Pull-up resistor value in kΩ

C MODE (Circuit from Figure 13.1, Option 3; XOSCMD = 10x)
Choose K Factor (KF) for the oscillation frequency desired:
$f = KF / (C * V_{DD})$, where
f = frequency of clock in MHz
C = capacitor value the XTAL2 pin in pF
V_{DD} = Power Supply on MCU in volts

Figure 13.5. OSCXCN: External Oscillator Control Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								11111111
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xF2

Bits7-0: Analog Input Configuration Bits for P1.7-P1.0 (respectively).
Port pins configured as analog inputs have their weak pull-up, digital driver, and digital receiver disabled.
0: Corresponding P1.n pin is configured as an analog input.
1: Corresponding P1.n pin is not configured as an analog input.

Figure 14.12. P1MDIN: Port1 Input Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xA5

Bits7-0: Output Configuration Bits for P1.7-P1.0 (respectively): ignored if corresponding bit in register P1MDIN is logic 0.
0: Corresponding P1.n Output is open-drain.
1: Corresponding P1.n Output is push-pull.

Figure 14.13. P1MDOUT: Port1 Output Mode Register

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
								00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SFR Address: 0xD5

Bits7-0: P1SKIP[7:0]: Port1 Crossbar Skip Enable Bits.
These bits select Port pins to be skipped by the Crossbar Decoder. Port pins used as analog inputs (for ADC or Comparator) or used as special functions (VREF input, external oscillator circuit, CNVSTR input) should be skipped by the Crossbar.
0: Corresponding P1.n pin is not skipped by the Crossbar.
1: Corresponding P1.n pin is skipped by the Crossbar.

Figure 14.14. P1SKIP: Port1 Skip Register

15.3. SMBus Operation

Two types of data transfers are possible: data transfers from a master transmitter to an addressed slave receiver (WRITE), and data transfers from an addressed slave transmitter to a master receiver (READ). The master device initiates both types of data transfers and provides the serial clock pulses on SCL. The SMBus interface may operate as a master or a slave, and multiple master devices on the same bus are supported. If two or more masters attempt to initiate a data transfer simultaneously, an arbitration scheme is employed with a single master always winning the arbitration. Note that it is not necessary to specify one device as the Master in a system; any device who transmits a START and a slave address becomes the master for the duration of that transfer.

A typical SMBus transaction consists of a START condition followed by an address byte (Bits7-1: 7-bit slave address; Bit0: R/W direction bit), one or more bytes of data, and a STOP condition. Each byte that is received (by a master or slave) must be acknowledged (ACK) with a low SDA during a high SCL (see Figure 15.3). If the receiving device does not ACK, the transmitting device will read a NACK (not acknowledge), which is a high SDA during a high SCL.

The direction bit (R/W) occupies the least-significant bit position of the address byte. The direction bit is set to logic 1 to indicate a "READ" operation and cleared to logic 0 to indicate a "WRITE" operation.

All transactions are initiated by a master, with one or more addressed slave devices as the target. The master generates the START condition and then transmits the slave address and direction bit. If the transaction is a WRITE operation from the master to the slave, the master transmits the data a byte at a time waiting for an ACK from the slave at the end of each byte. For READ operations, the slave transmits the data waiting for an ACK from the master at the end of each byte. At the end of the data transfer, the master generates a STOP condition to terminate the transaction and free the bus. Figure 15.3 illustrates a typical SMBus transaction.

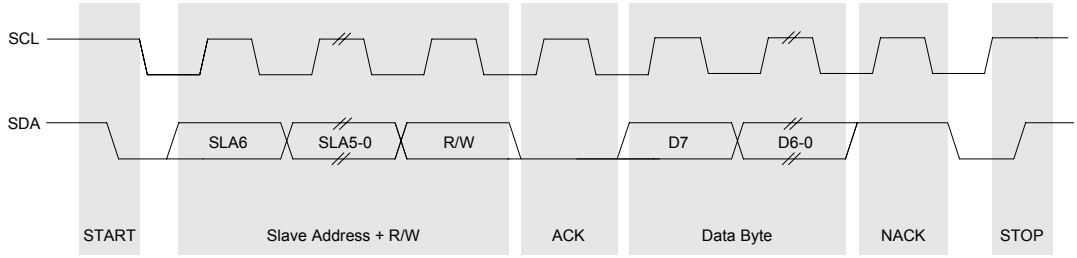


Figure 15.3. SMBus Transaction

15.3.1. Arbitration

A master may start a transfer only if the bus is free. The bus is free after a STOP condition or after the SCL and SDA lines remain high for a specified time (see **Section "15.3.4. SCL High (SMBus Free) Timeout" on page 128**). In the event that two or more devices attempt to begin a transfer at the same time, an arbitration scheme is employed to force one master to give up the bus. The master devices continue transmitting until one attempts a HIGH while the other transmits a LOW. Since the bus is open-drain, the bus will be pulled LOW. The master attempting the HIGH will detect a LOW SDA and lose the arbitration. The winning master continues its transmission without interruption; the losing master becomes a slave and receives the rest of the transfer if addressed. This arbitration scheme is non-destructive: one device always wins, and no data is lost.

17.2. SPI0 Master Mode Operation

A SPI master device initiates all data transfers on a SPI bus. SPI0 is placed in master mode by setting the Master Enable flag (MSTEN, SPI0CN.6). Writing a byte of data to the SPI0 data register (SPI0DAT) when in master mode writes to the transmit buffer. If the SPI shift register is empty, the byte in the transmit buffer is moved to the shift register, and a data transfer begins. The SPI0 master immediately shifts out the data serially on the MOSI line while providing the serial clock on SCK. The SPIF (SPI0CN.7) flag is set to logic 1 at the end of the transfer. If interrupts are enabled, an interrupt request is generated when the SPIF flag 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 17.2 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 17.3 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 17.4 shows a connection diagram for a master device in 4-wire master mode and two slave devices.

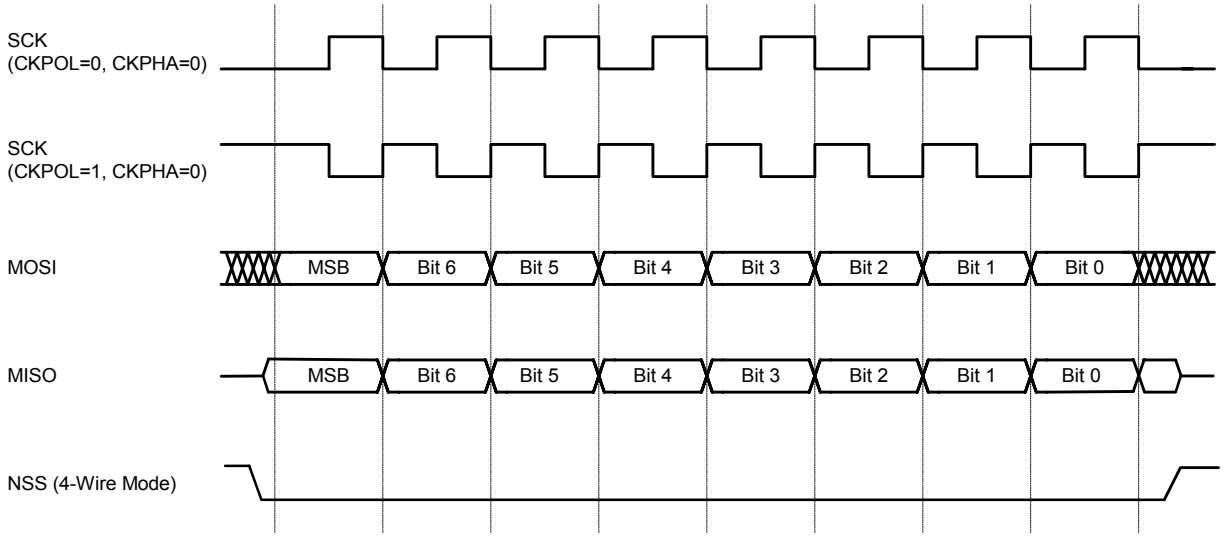


Figure 17.6. Slave Mode Data/Clock Timing (CKPHA = 0)

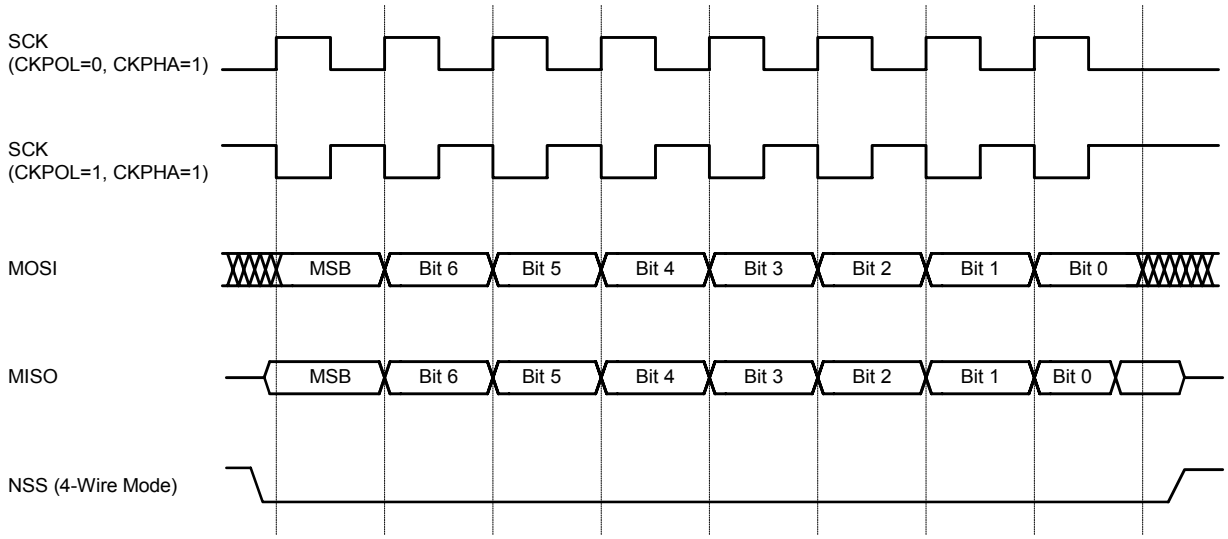
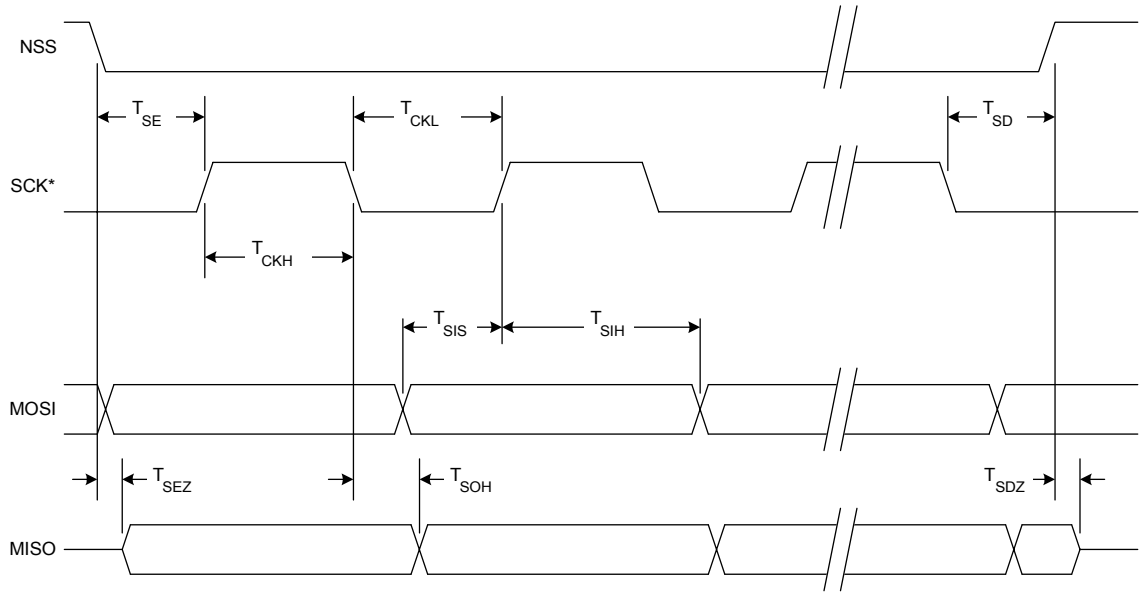
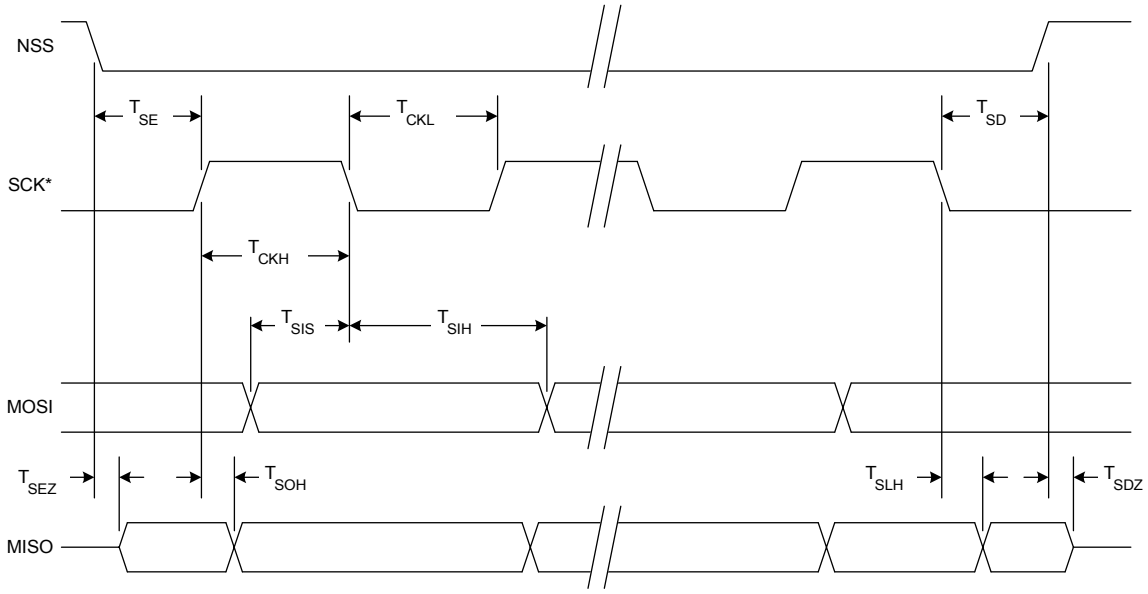


Figure 17.7. Slave Mode Data/Clock Timing (CKPHA = 1)



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

Figure 17.14. SPI Slave Timing (CKPHA = 0)



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

Figure 17.15. SPI Slave Timing (CKPHA = 1)

18.2. Timer 2

Timer 2 is a 16-bit timer formed by two 8-bit SFRs: TMR2L (low byte) and TMR2H (high byte). Timer 2 may operate in 16-bit auto-reload mode or (split) 8-bit auto-reload mode. The T2SPLIT bit (TMR2CN.3) defines the Timer 2 operation mode.

Timer 2 may be clocked by the system clock, the system clock divided by 12, or the external oscillator source divided by 8. The external clock mode is ideal for real-time clock (RTC) functionality, where the internal oscillator drives the system clock while Timer 2 (and/or the PCA) is clocked by an external precision oscillator. Note that the external oscillator source divided by 8 is synchronized with the system clock.

18.2.1. 16-bit Timer with Auto-Reload

When T2SPLIT (TMR2CN.3) is zero, Timer 2 operates as a 16-bit timer with auto-reload. Timer 2 can be clocked by SYSCLK, SYSCLK divided by 12, or the external oscillator clock source divided by 8. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the Timer 2 reload registers (TMR2RLH and TMR2RLL) is loaded into the Timer 2 register as shown in Figure 18.11, and the Timer 2 High Byte Overflow Flag (TMR2CN.7) is set. If Timer 2 interrupts are enabled (if IE.5 is set), an interrupt will be generated on each Timer 2 overflow. Additionally, if Timer 2 interrupts are enabled and the TF2LEN bit is set (TMR2CN.5), an interrupt will be generated each time the lower 8 bits (TMR2L) overflow from 0xFF to 0x00.

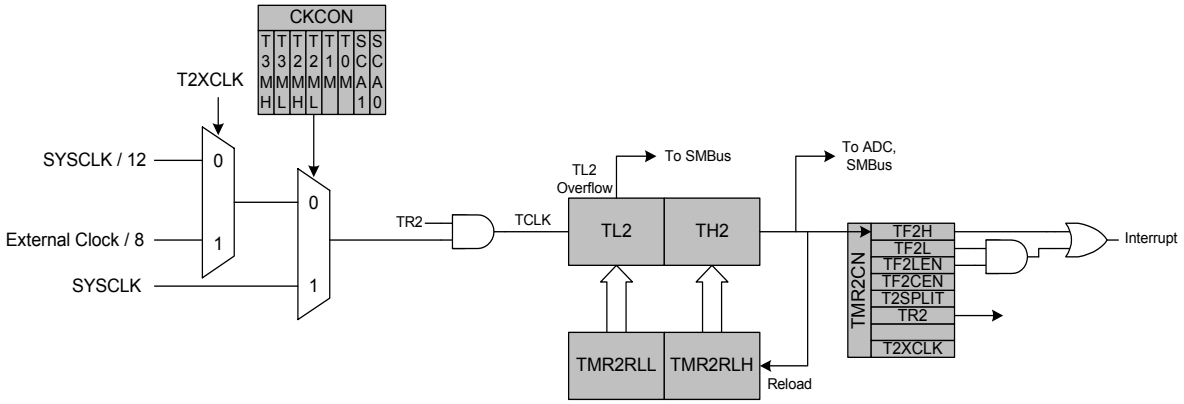


Figure 18.11. Timer 2 16-Bit Mode Block Diagram

19. Programmable Counter Array

The Programmable Counter Array (PCA0) provides enhanced timer functionality while requiring less CPU intervention than the standard 8051 counter/timers. The PCA consists of a dedicated 16-bit counter/timer and three 16-bit capture/compare modules. Each capture/compare module has its own associated I/O line (CEXn) which is routed through the Crossbar to Port I/O when enabled (See **Section "14.1. Priority Crossbar Decoder" on page 115** for details on configuring the Crossbar). The counter/timer is driven by a programmable timebase that can select between six sources: system clock, system clock divided by four, system clock divided by twelve, the external oscillator clock source divided by 8, Timer 0 overflow, or an external clock signal on the ECI input pin. Each capture/compare module may be configured to operate independently in one of six modes: Edge-Triggered Capture, Software Timer, High-Speed Output, Frequency Output, 8-Bit PWM, or 16-Bit PWM (each mode is described in **Section "19.2. Capture/Compare Modules" on page 189**). The external oscillator clock option is ideal for real-time clock (RTC) functionality, allowing the PCA to be clocked by a precision external oscillator while the internal oscillator drives the system clock. The PCA is configured and controlled through the system controller's Special Function Registers. The PCA block diagram is shown in Figure 19.1

Important Note: The PCA Module 2 may be used as a watchdog timer (WDT), and is enabled in this mode following a system reset. **Access to certain PCA registers is restricted while WDT mode is enabled**. See **Section 19.3** for details.

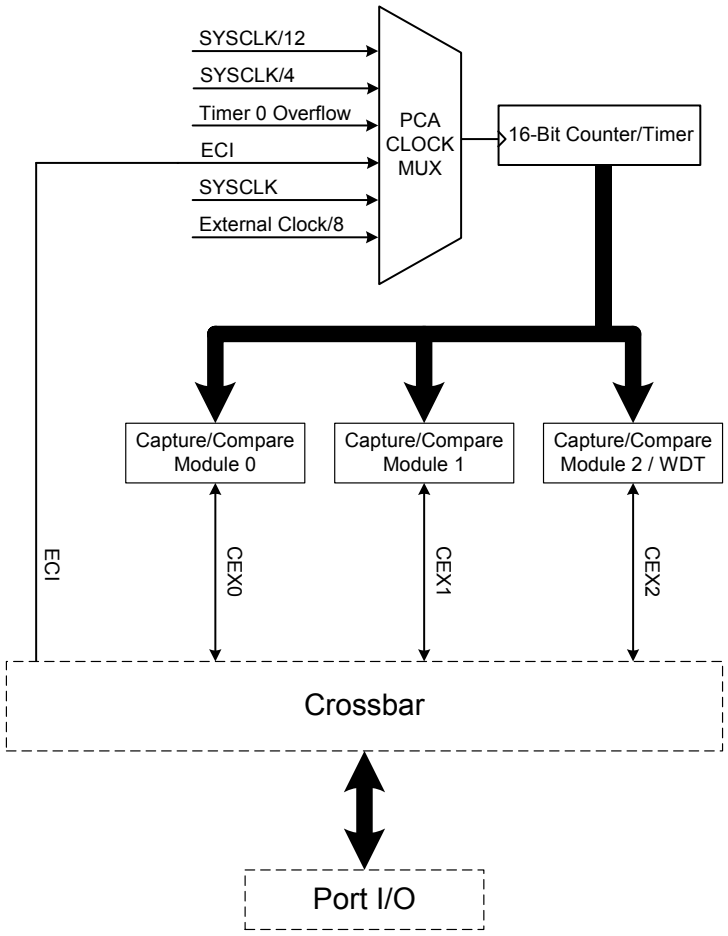


Figure 19.1. PCA Block Diagram

R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	Reset Value
PWM16n	ECOMn	CAPPn	CAPNn	MATn	TOGn	PWMn	ECCFn	00000000
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	

SFR Address: PCA0CPM0: 0xDA, PCA0CPM1: 0xDB, PCA0CPM2: 0xDC

Bit7: PWM16n: 16-bit Pulse Width Modulation Enable.
This bit selects 16-bit mode when Pulse Width Modulation mode is enabled (PWMn = 1).
0: 8-bit PWM selected.
1: 16-bit PWM selected.

Bit6: ECOMn: Comparator Function Enable.
This bit enables/disables the comparator function for PCA module n.
0: Disabled.
1: Enabled.

Bit5: CAPPn: Capture Positive Function Enable.
This bit enables/disables the positive edge capture for PCA module n.
0: Disabled.
1: Enabled.

Bit4: CAPNn: Capture Negative Function Enable.
This bit enables/disables the negative edge capture for PCA module n.
0: Disabled.
1: Enabled.

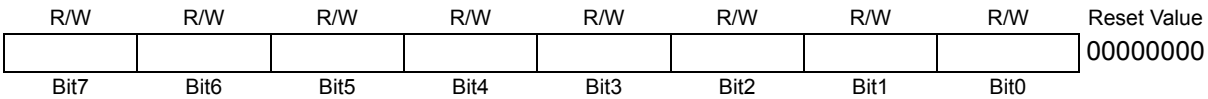
Bit3: MATn: Match Function Enable.
This bit enables/disables the match function for PCA module n. When enabled, matches of the PCA counter with a module's capture/compare register cause the CCFn bit in PCA0MD register to be set to logic 1.
0: Disabled.
1: Enabled.

Bit2: TOGn: Toggle Function Enable.
This bit enables/disables the toggle function for PCA module n. When enabled, matches of the PCA counter with a module's capture/compare register cause the logic level on the CEXn pin to toggle. If the PWMn bit is also set to logic 1, the module operates in Frequency Output Mode.
0: Disabled.
1: Enabled.

Bit1: PWMn: Pulse Width Modulation Mode Enable.
This bit enables/disables the PWM function for PCA module n. When enabled, a pulse width modulated signal is output on the CEXn pin. 8-bit PWM is used if PWM16n is cleared; 16-bit mode is used if PWM16n is set to logic 1. If the TOGn bit is also set, the module operates in Frequency Output Mode.
0: Disabled.
1: Enabled.

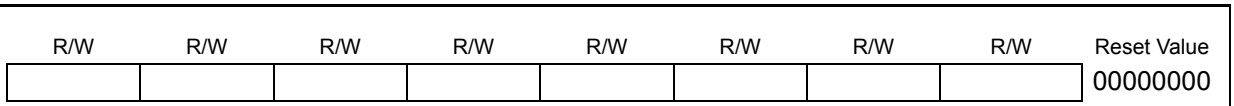
Bit0: ECCFn: Capture/Compare Flag Interrupt Enable.
This bit sets the masking of the Capture/Compare Flag (CCFn) interrupt.
0: Disable CCFn interrupts.
1: Enable a Capture/Compare Flag interrupt request when CCFn is set.

Figure 19.13. PCA0CPMn: PCA Capture/Compare Mode Registers



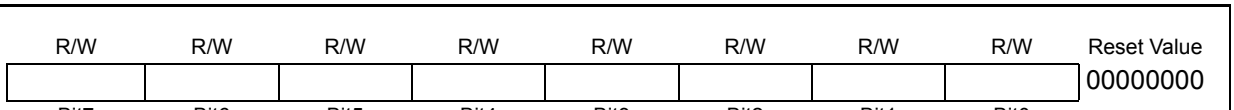
Bits 7-0: PCA0L: PCA Counter/Timer Low Byte.
The PCA0L register holds the low byte (LSB) of the 16-bit PCA Counter/Timer.

Figure 19.14. PCA0L: PCA Counter/Timer Low Byte



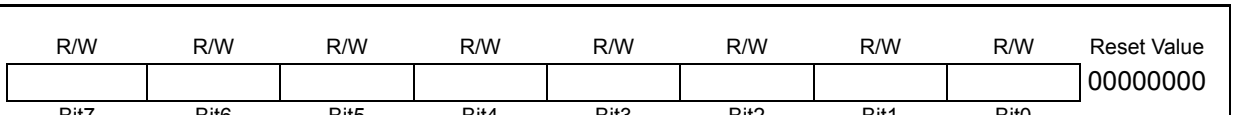
Bits 7-0: PCA0H: PCA Counter/Timer High Byte.
The PCA0H register holds the high byte (MSB) of the 16-bit PCA Counter/Timer.

Figure 19.15. PCA0H: PCA Counter/Timer High Byte



Bits7-0: PCA0CPLn: PCA Capture Module Low Byte.
The PCA0CPLn register holds the low byte (LSB) of the 16-bit capture module n.

Figure 19.16. PCA0CPLn: PCA Capture Module Low Byte



Bits7-0: PCA0CPHn: PCA Capture Module High Byte.
The PCA0CPHn register holds the high byte (MSB) of the 16-bit capture module n.

Figure 19.17. PCA0CPHn: PCA Capture Module High Byte