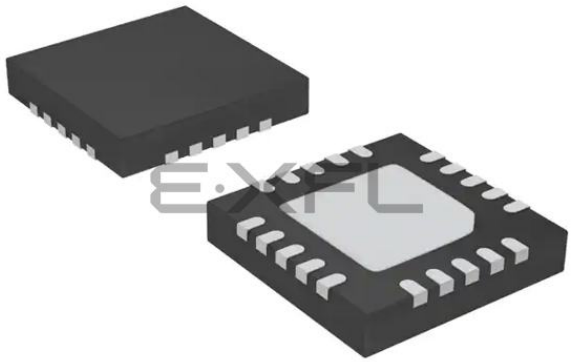

Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Not For New Designs
Core Processor	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	512 x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 16x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-VFQFN Exposed Pad
Supplier Device Package	20-QFN (4x4)
Purchase URL	https://www.e-xfl.com/product-detail/silicon-labs/c8051f814-gmr

List of Figures

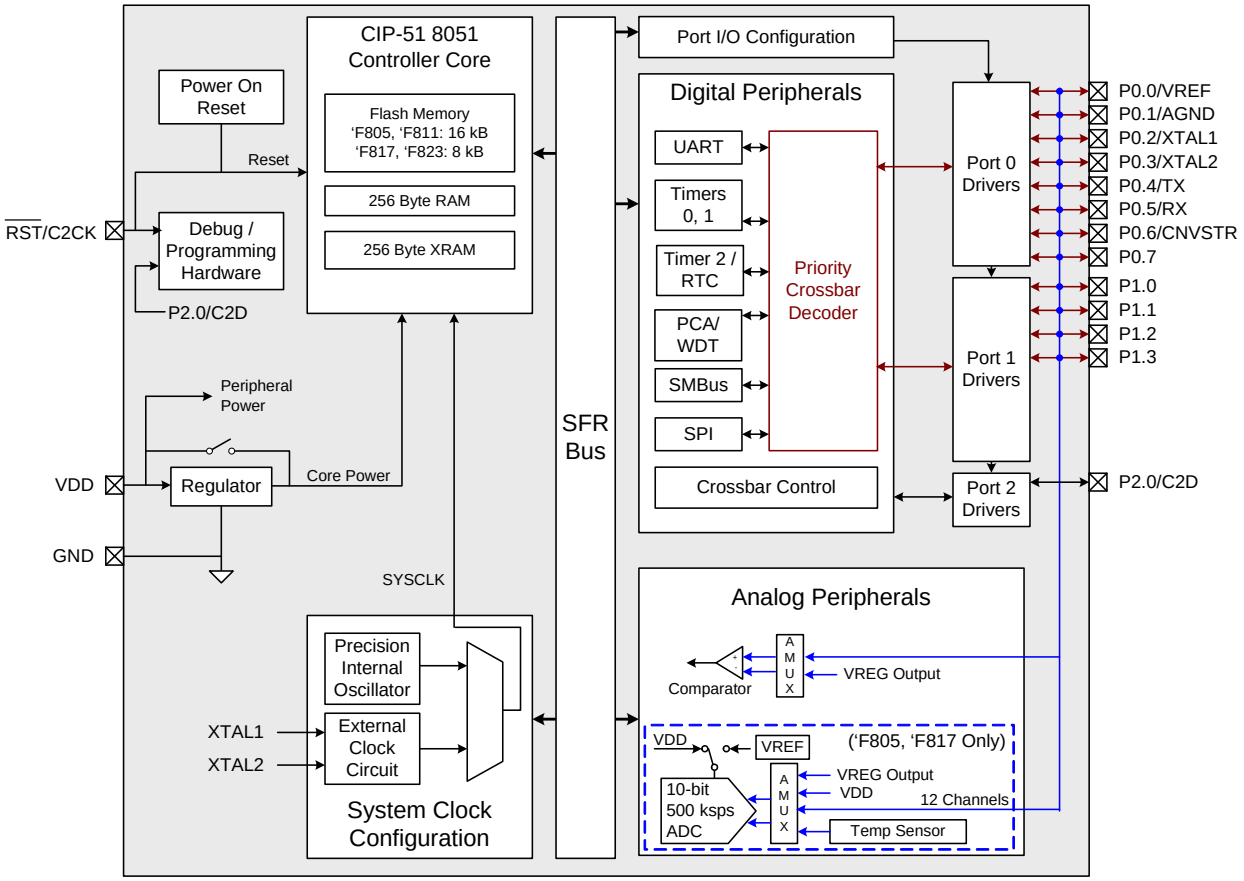

Figure 1.6. C8051F805, C8051F811, C8051F817, C8051F823 Block Diagram

8.1. Output Code Formatting

The ADC measures the input voltage with reference to GND. The registers ADC0H and ADC0L contain the high and low bytes of the output conversion code from the ADC at the completion of each conversion. Data can be right-justified or left-justified, depending on the setting of the AD0LJST bit. Conversion codes are represented as 10-bit unsigned integers. Inputs are measured from 0 to VREF x 1023/1024. Example codes are shown below for both right-justified and left-justified data. Unused bits in the ADC0H and ADC0L registers are set to 0.

Input Voltage	Right-Justified ADC0H:ADC0L (AD0LJST = 0)	Left-Justified ADC0H:ADC0L (AD0LJST = 1)
VREF x 1023/1024	0x03FF	0xFFC0
VREF x 512/1024	0x0200	0x8000
VREF x 256/1024	0x0100	0x4000
0	0x0000	0x0000

8.2. 8-Bit Mode

Setting the ADC08BE bit in register ADC0CF to 1 will put the ADC in 8-bit mode. In 8-bit mode, only the 8 MSBs of data are converted, and the ADC0H register holds the results. The AD0LJST bit is ignored for 8-bit mode. 8-bit conversions take two fewer SAR clock cycles than 10-bit conversions, so the conversion is completed faster, and a 500 ksps sampling rate can be achieved with a slower SAR clock.

8.3. Modes of Operation

ADC0 has a maximum conversion speed of 500 ksps. The ADC0 conversion clock is a divided version of the system clock, determined by the AD0SC bits in the ADC0CF register.

8.3.1. Starting a Conversion

A conversion can be initiated in one of six ways, depending on the programmed states of the ADC0 Start of Conversion Mode bits (AD0CM2–0) in register ADC0CN. Conversions may be initiated by one of the following:

1. Writing a 1 to the AD0BUSY bit of register ADC0CN
2. A Timer 0 overflow (i.e., timed continuous conversions)
3. A Timer 2 overflow
4. A Timer 1 overflow
5. A rising edge on the CNVSTR input signal

Writing a 1 to AD0BUSY provides software control of ADC0 whereby conversions are performed "on-demand". During conversion, the AD0BUSY bit is set to logic 1 and reset to logic 0 when the conversion is complete. The falling edge of AD0BUSY triggers an interrupt (when enabled) and sets the ADC0 interrupt flag (AD0INT). When polling for ADC conversion completions, the ADC0 interrupt flag (AD0INT) should be used. Converted data is available in the ADC0 data registers, ADC0H:ADC0L, when bit AD0INT is logic 1. When Timer 2 overflows are used as the conversion source, Low Byte overflows are used if Timer 2/3 is in 8-bit mode; High byte overflows are used if Timer 2 is in 16-bit mode. See Section “28. Timers” on page 209 for timer configuration.

Important Note About Using CNVSTR: The CNVSTR input pin also functions as a Port I/O pin. When the CNVSTR input is used as the ADC0 conversion source, the associated pin should be skipped by the Digital Crossbar. See Section “23. Port Input/Output” on page 138 for details on Port I/O configuration.

12.1. Comparator Multiplexer

C8051F80x-83x devices include an analog input multiplexer to connect Port I/O pins to the comparator inputs. The Comparator0 inputs are selected in the CPT0MX register (SFR Definition 12.3). The CMX0P3–CMX0P0 bits select the Comparator0 positive input; the CMX0N3–CMX0N0 bits select the Comparator0 negative input. **Important Note About Comparator Inputs:** The Port pins selected as comparator inputs should be configured as analog inputs in their associated Port configuration register, and configured to be skipped by the Crossbar (for details on Port configuration, see Section “23.6. Special Function Registers for Accessing and Configuring Port I/O” on page 152).

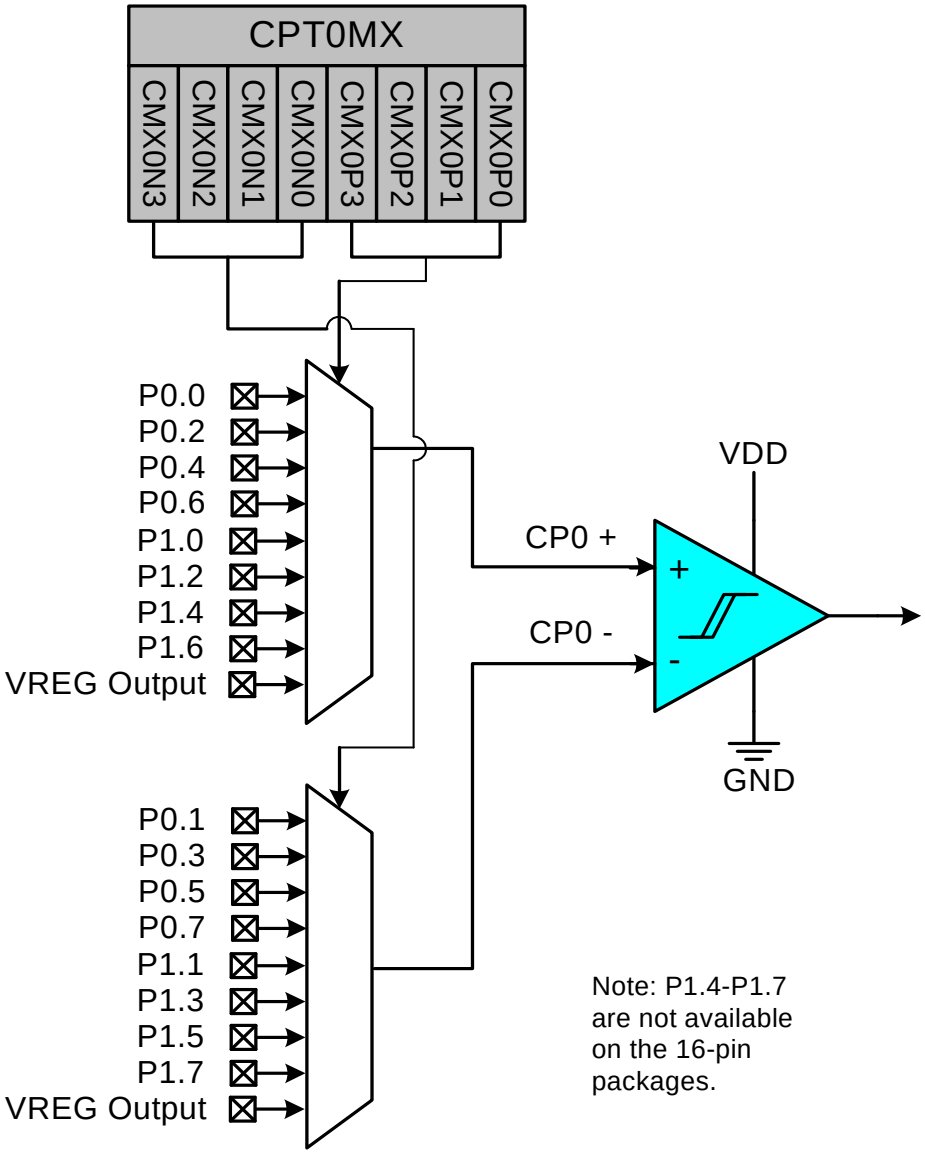

Figure 12.3. Comparator Input Multiplexer Block Diagram

SFR Definition 13.3. CS0DH: Capacitive Sense Data High Byte

Bit	7	6	5	4	3	2	1	0
Name	CS0DH[7:0]							
Type	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xAC

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

SFR Definition 13.4. CS0DL: Capacitive Sense Data Low Byte

Bit	7	6	5	4	3	2	1	0
Name	CS0DL[7:0]							
Type	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xAB

Bit	Name	Description
7:0	CS0DL	**CS0 Data Low Byte.** Stores the low byte of the last completed 16-bit Capacitive Sense conversion.

14. CIP-51 Microcontroller

The MCU system controller core is the CIP-51 microcontroller. The CIP-51 is fully compatible with the MCS-51™ instruction set; standard 803x/805x assemblers and compilers can be used to develop software. The MCU family has a superset of all the peripherals included with a standard 8051. The CIP-51 also includes on-chip debug hardware (see description in Section 30), and interfaces directly with the analog and digital subsystems providing a complete data acquisition or control-system solution in a single integrated circuit.

The CIP-51 Microcontroller core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability (see Figure 14.1 for a block diagram). The CIP-51 includes the following features:

- Fully Compatible with MCS-51 Instruction Set
- 25 MIPS Peak Throughput with 25 MHz Clock
- 0 to 25 MHz Clock Frequency
- Extended Interrupt Handler
- Reset Input
- Power Management Modes
- On-chip Debug Logic
- Program and Data Memory Security

Performance

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. In a standard 8051, all instructions except for MUL and DIV take 12 or 24 system clock cycles to execute, and usually have a maximum system clock of 12 MHz. By contrast, the CIP-51 core executes 70% of its instructions in one or two system clock cycles, with no instructions taking more than eight system clock cycles.

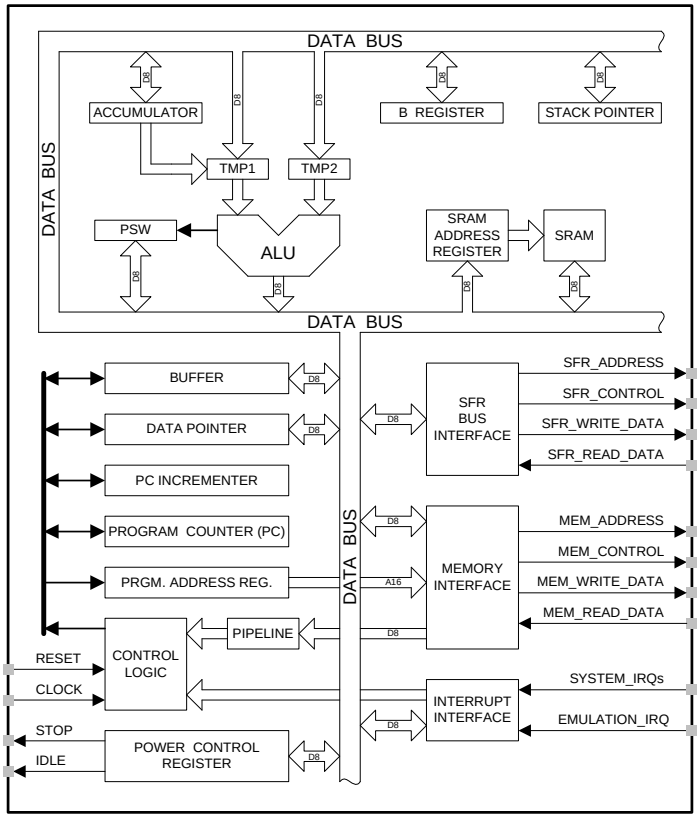

Figure 14.1. CIP-51 Block Diagram

With the CIP-51's maximum system clock at 25 MHz, it has a peak throughput of 25 MIPS. The CIP-51 has a total of 109 instructions. The table below shows the total number of instructions that require each execution time.

Clocks to Execute	1	2	2/3	3	3/4	4	4/5	5	8
Number of Instructions	26	50	5	14	6	3	2	2	1

14.1. Instruction Set

The instruction set of the CIP-51 System Controller is fully compatible with the standard MCS-51™ instruction set. Standard 8051 development tools can be used to develop software for the CIP-51. All CIP-51 instructions are the binary and functional equivalent of their MCS-51™ counterparts, including opcodes, addressing modes and effect on PSW flags. However, instruction timing is different than that of the standard 8051.

14.1.1. Instruction and CPU Timing

In many 8051 implementations, a distinction is made between machine cycles and clock cycles, with machine cycles varying from 2 to 12 clock cycles in length. However, the CIP-51 implementation is based solely on clock cycle timing. All instruction timings are specified in terms of clock cycles.

Due to the pipelined architecture of the CIP-51, most instructions execute in the same number of clock cycles as there are program bytes in the instruction. Conditional branch instructions take one less clock cycle to complete when the branch is not taken as opposed to when the branch is taken. Table 14.1 is the CIP-51 Instruction Set Summary, which includes the mnemonic, number of bytes, and number of clock cycles for each instruction.

SFR Definition 18.1. IE: Interrupt Enable

Bit	7	6	5	4	3	2	1	0
Name	EA	ESPI0	ET2	ES0	ET1	EX1	ET0	EX0
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 Address = 0xA8; Bit-Addressable

Bit	Name	Function
7	EA	**Enable All Interrupts.** Globally enables/disables all interrupts. It overrides individual interrupt mask settings. 0: Disable all interrupt sources. 1: Enable each interrupt according to its individual mask setting.
6	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.
5	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.
4	ES0	**Enable UART0 Interrupt.** This bit sets the masking of the UART0 interrupt. 0: Disable UART0 interrupt. 1: Enable UART0 interrupt.
3	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.
2	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 $\overline{\text{INT1}}$ input.
1	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.
0	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 $\overline{\text{INT0}}$ input.

features of the C8051F80x-83x devices.

Table 19.1. Flash Security Summary

Action	C2 Debug Interface	User Firmware executing from:	
		an unlocked page	a locked page
Read, Write or Erase unlocked pages (except page with Lock Byte)	Permitted	Permitted	Permitted
Read, Write or Erase locked pages (except page with Lock Byte)	Not Permitted	FEDR	Permitted
Read or Write page containing Lock Byte (if no pages are locked)	Permitted	Permitted	Permitted
Read or Write page containing Lock Byte (if any page is locked)	Not Permitted	FEDR	Permitted
Read contents of Lock Byte (if no pages are locked)	Permitted	Permitted	Permitted
Read contents of Lock Byte (if any page is locked)	Not Permitted	FEDR	Permitted
Erase page containing Lock Byte (if no pages are locked)	Permitted	FEDR	FEDR
Erase page containing Lock Byte—Unlock all pages (if any page is locked)	Only by C2DE	FEDR	FEDR
Lock additional pages (change 1s to 0s in the Lock Byte)	Not Permitted	FEDR	FEDR
Unlock individual pages (change 0s to 1s in the Lock Byte)	Not Permitted	FEDR	FEDR
Read, Write or Erase Reserved Area	Not Permitted	FEDR	FEDR

C2DE—C2 Device Erase (Erases all Flash pages including the page containing the Lock Byte)
FEDR—Not permitted; Causes Flash Error Device Reset (FERROR bit in RSTSRC is 1 after reset)

- All prohibited operations that are performed via the C2 interface are ignored (do not cause device reset).
- Locking any Flash page also locks the page containing the Lock Byte.
- Once written to, the Lock Byte cannot be modified except by performing a C2 Device Erase.
- If user code writes to the Lock Byte, the Lock does not take effect until the next device reset.

19.4. Flash Write and Erase Guidelines

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

To help prevent the accidental modification of Flash by firmware, the VDD Monitor must be enabled and enabled as a reset source on C8051F80x-83x devices for the Flash to be successfully modified. **If either the VDD Monitor or the VDD Monitor reset source is not enabled, a Flash Error Device Reset will be generated when the firmware attempts to modify the Flash.**

20.2. Stop Mode

Setting the Stop Mode Select bit (PCON.1) causes the controller core to enter Stop mode as soon as the instruction that sets the bit completes execution. In Stop mode the internal oscillator, CPU, and all digital peripherals are stopped; the state of the external oscillator circuit is not affected. Each analog peripheral (including the external oscillator circuit) may be shut down individually prior to entering Stop Mode. Stop mode can only be terminated by an internal or external reset. On reset, the device performs the normal reset sequence and begins program execution at address 0x0000.

If enabled, the Missing Clock Detector will cause an internal reset and thereby terminate the Stop mode. The Missing Clock Detector should be disabled if the CPU is to be put to in STOP mode for longer than the MCD timeout of 100 µs.

20.3. Suspend Mode

Suspend mode allows a system running from the internal oscillator to go to a very low power state similar to Stop mode, but the processor can be awakened by certain events without requiring a reset of the device. Setting the SUSPEND bit (OSCICN.5) causes the hardware to halt the CPU and the high-frequency internal oscillator, and go into Suspend mode as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data. Most digital peripherals are not active in Suspend mode. The exception to this is the Port Match feature and Timer 3, when it is run from an external oscillator source.

The clock divider bits CLKDIV[2:0] in register CLKSEL must be set to "divide by 1" when entering suspend mode.

Suspend mode can be terminated by five types of events, a port match (described in Section “23.5. Port Match” on page 150), a Timer 2 overflow (described in Section “28.2. Timer 2” on page 219), a comparator low output (if enabled), a capacitive sense greater-than comparator event, or a device reset event. In order to run Timer 3 in suspend mode, the timer must be configured to clock from the external clock source. When suspend mode is terminated, the device will continue execution on the instruction following the one that set the SUSPEND bit. If the wake event (port match or Timer 2 overflow) was configured to generate an interrupt, the interrupt will be serviced upon waking the device. If suspend mode is terminated by an internal or external reset, the CIP-51 performs a normal reset sequence and begins program execution at address 0x0000.

Note: The device will still enter suspend mode if a wake source is "pending", and the device will not wake on such pending sources. It is important to ensure that the intended wake source will trigger after the device enters suspend mode. For example, if a CS0 conversion completes and the interrupt fires before the device is in suspend mode, that interrupt cannot trigger the wake event. Because port match events are level-sensitive, pre-existing port match events will trigger a wake, as long as the match condition is still present when the device enters suspend.

21.1. Power-On Reset

During power-up, the device is held in a reset state and the $\overline{\text{RST}}$ pin is driven low until V_{DD} settles above V_{RST}. A delay occurs before the device is released from reset; the delay decreases as the V_{DD} ramp time increases (V_{DD} ramp time is defined as how fast V_{DD} ramps from 0 V to V_{RST}). Figure 21.2. plots the power-on and V_{DD} monitor reset timing. The maximum V_{DD} ramp time is 1 ms; slower ramp times may cause the device to be released from reset before V_{DD} reaches the V_{RST} level. For ramp times less than 1 ms, the power-on reset delay ($T_{PORDelay}$) is typically less than 10 ms.

On exit from a power-on reset, the PORSF flag (RSTSRC.1) is set by hardware to logic 1. When PORSF is set, all of the other reset flags in the RSTSRC Register are indeterminate (PORSF is cleared by all other resets). Since all resets cause program execution to begin at the same location (0x0000) software can read the PORSF flag to determine if a power-up was the cause of reset. The content of internal data memory should be assumed to be undefined after a power-on reset. The V_{DD} monitor is enabled and selected as a reset source following a power-on reset.

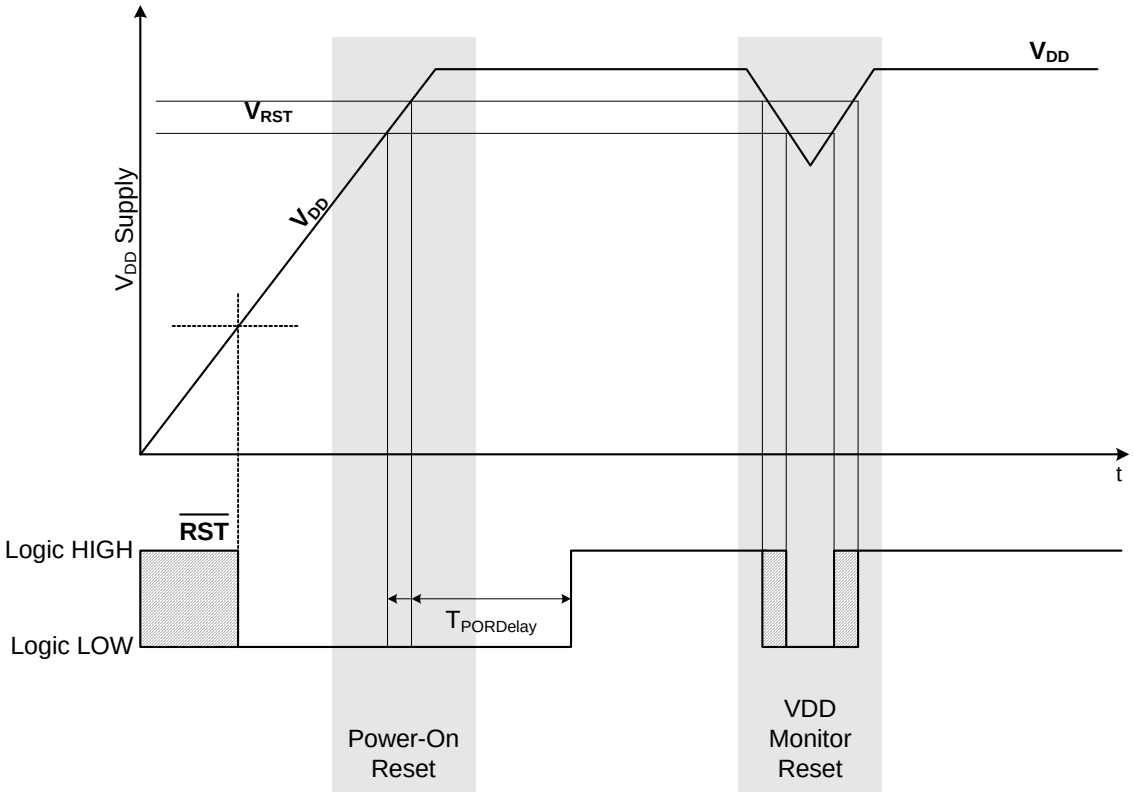

Figure 21.2. Power-On and V_{DD} Monitor Reset Timing

24.1. 16-bit CRC Algorithm

The C8051F80x-83x 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).
2. If the MSB of the CRC result is set, left-shift the CRC result, and then XOR the CRC result with the polynomial (0x1021).
3. If the MSB of the CRC result is not set, left-shift the CRC result.
4. Repeat at Step 2 for the number of input bits (8).

For example, the 16-bit C8051F80x-83x CRC algorithm can be described by the following code:

```
unsigned short UpdateCRC (unsigned short CRC_acc, unsigned char CRC_input){
   unsigned char i;                    // loop counter
   #define POLY 0x1021
   // Create the CRC "dividend" for polynomial arithmetic (binary arithmetic
   // with no carries)
   CRC_acc = CRC_acc ^ (CRC_input << 8);
   // "Divide" the poly into the dividend using CRC XOR subtraction
   // CRC_acc holds the "remainder" of each divide
   // Only complete this division for 8 bits since input is 1 byte
   for (i = 0; i < 8; i++)
   {
      // Check if the MSB is set (if MSB is 1, then the POLY can "divide"
      // into the "dividend")
      if ((CRC_acc & 0x8000) == 0x8000)
      {
         // if so, shift the CRC value, and XOR "subtract" the poly
         CRC_acc = CRC_acc << 1;
         CRC_acc ^= POLY;
      }
      else
      {
         // if not, just shift the CRC value
         CRC_acc = CRC_acc << 1;
      }
   }
   return CRC_acc; // Return the final remainder (CRC value)
}
```

Table 24.1 lists example input values and the associated outputs using the 16-bit C8051F80x-83x CRC algorithm (an initial value of 0xFFFF is used):

Table 24.1. Example 16-bit CRC Outputs

Input	Output
0x63	0xBD35
0xAA, 0xBB, 0xCC	0x6CF6
0x00, 0x00, 0xAA, 0xBB, 0xCC	0xB166

24.6. CRC0 Bit Reverse Feature

CRC0 includes hardware to reverse the bit order of each bit in a byte as shown in Figure 24.1. Each byte of data written to CRC0FLIP is read back bit reversed. For example, if 0xC0 is written to CRC0FLIP, the data read back is 0x03. Bit reversal is a useful mathematical function used in algorithms such as the FFT.

SFR Definition 24.6. CRC0FLIP: CRC Bit Flip

Bit	7	6	5	4	3	2	1	0
Name	CRC0FLIP[7:0]							
Type	R/W							
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xCF

Bit	Name	Function
7:0	CRC0FLIP[7:0]	**CRC0 Bit Flip.** Any byte written to CRC0FLIP is read back in a bit-reversed order, i.e. the written LSB becomes the MSB. For example: If 0xC0 is written to CRC0FLIP, the data read back will be 0x03. If 0x05 is written to CRC0FLIP, the data read back will be 0xA0.

26.4.2. SMB0CN Control Register

SMB0CN is used to control the interface and to provide status information (see SFR Definition 26.2). 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 26.3 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.

26.4.2.1. 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.

26.4.2.2. 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 26.4.3. 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 26.3 lists all sources for hardware changes to the SMB0CN bits. Refer to Table 26.5 for SMBus status decoding using the SMB0CN register.

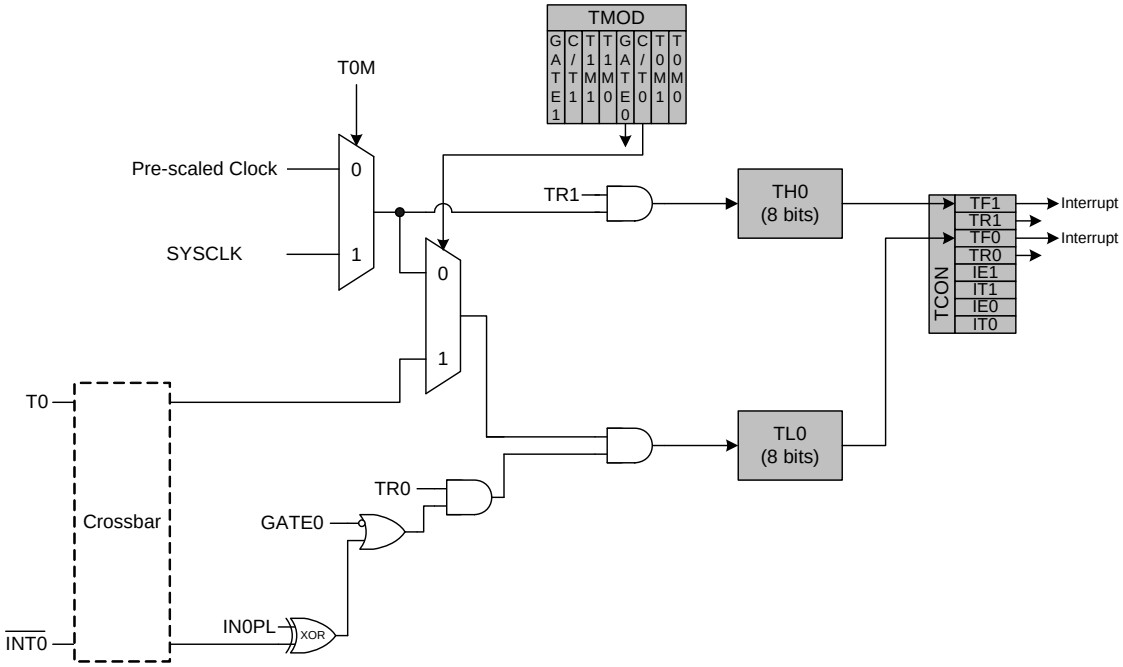

Figure 28.3. T0 Mode 3 Block Diagram

29. 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. 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 overflows, 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 to 15-Bit PWM, or 16-Bit PWM (each mode is described in Section “29.3. Capture/Compare Modules” on page 228). 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 29.1

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 29.4 for details.

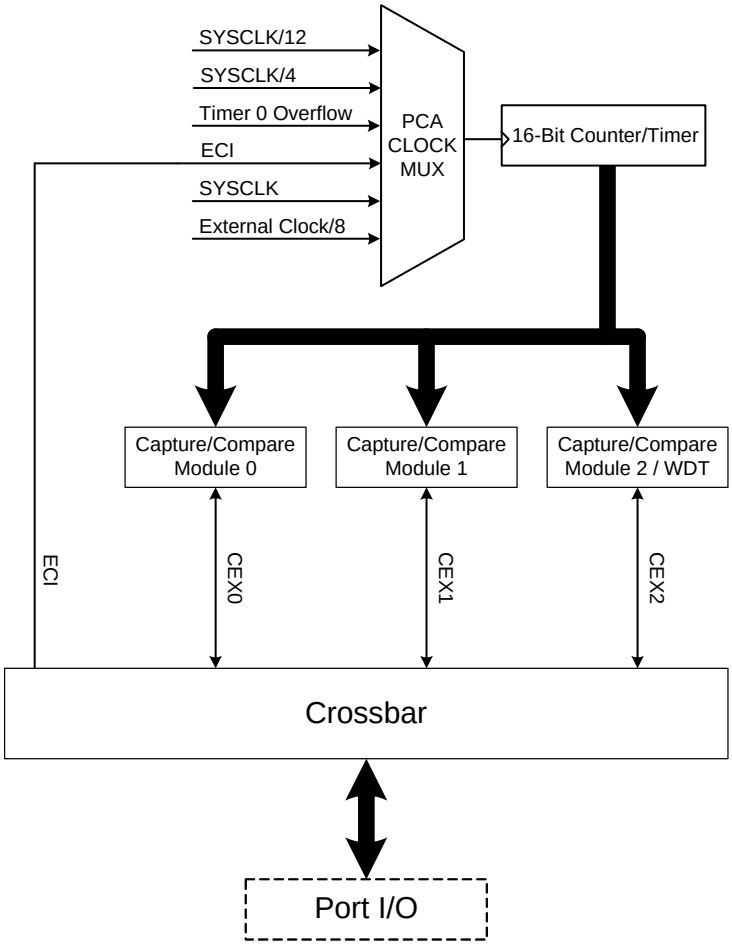

Figure 29.1. PCA Block Diagram

29.1. PCA Counter/Timer

The 16-bit PCA counter/timer consists of two 8-bit SFRs: PCA0L and PCA0H. PCA0H is the high byte (MSB) of the 16-bit counter/timer and PCA0L is the low byte (LSB). Reading PCA0L automatically latches the value of PCA0H into a “snapshot” register; the following PCA0H read accesses this “snapshot” register. **Reading the PCA0L Register first guarantees an accurate reading of the entire 16-bit PCA0 counter.** Reading PCA0H or PCA0L does not disturb the counter operation. The CPS2–CPS0 bits in the PCA0MD register select the timebase for the counter/timer as shown in Table 29.1.

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

Table 29.1. PCA Timebase Input Options

CPS2	CPS1	CPS0	Timebase
0	0	0	System clock divided by 12
0	0	1	System clock divided by 4
0	1	0	Timer 0 overflow
0	1	1	High-to-low transitions on ECI (max rate = system clock divided by 4)
1	0	0	System clock
1	0	1	External oscillator source divided by 8 (Note)
1	1	x	Reserved
Note: External oscillator source divided by 8 is synchronized with the system clock.			

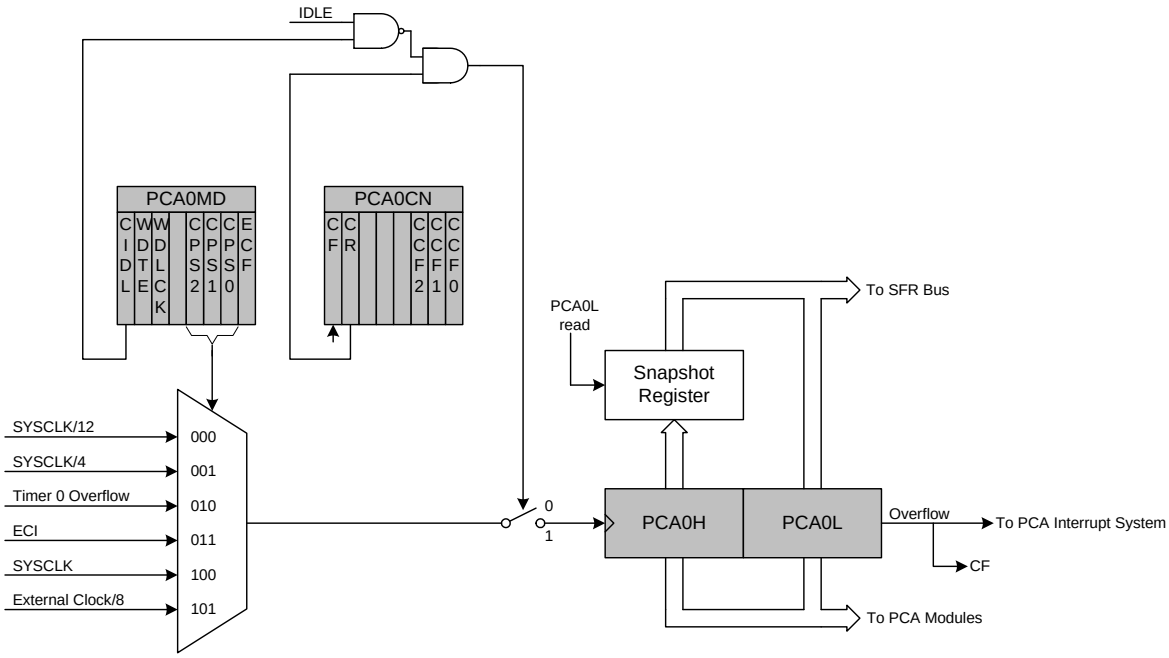

Figure 29.2. PCA Counter/Timer Block Diagram

29.3.2. Software Timer (Compare) Mode

In Software Timer mode, the PCA counter/timer value is compared to the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match occurs, the Capture/Compare Flag (CCFn) in PCA0CN is set to logic 1. An interrupt request is generated if the CCFn interrupt for that module is enabled. The CCFn bit is not automatically cleared by hardware when the CPU vectors to the interrupt service routine, and must be cleared by software. Setting the ECOMn and MATn bits in the PCA0CPMn register enables Software Timer mode.

Important Note About Capture/Compare Registers: When writing a 16-bit value to the PCA0 Capture/Compare 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.

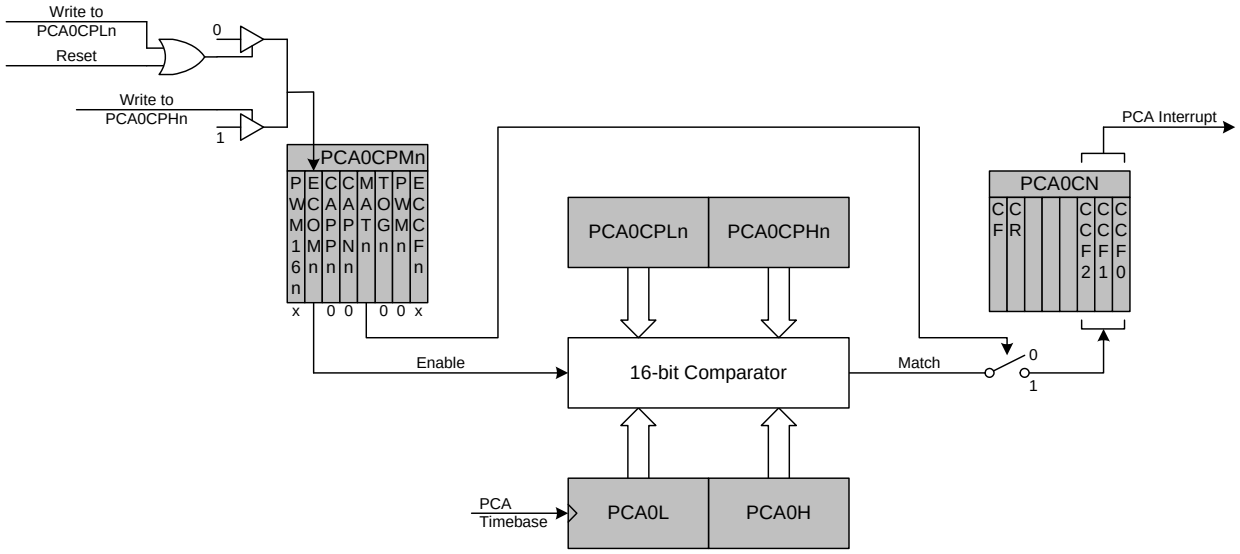

Figure 29.5. PCA Software Timer Mode Diagram

SFR Definition 29.5. PCA0L: PCA0 Counter/Timer Low Byte

Bit	7	6	5	4	3	2	1	0
Name	PCA0[7:0]							
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xF9

Bit	Name	Function
7:0	PCA0[7:0]	**PCA Counter/Timer Low Byte.** The PCA0L register holds the low byte (LSB) of the 16-bit PCA Counter/Timer.

Note: When the WDTE bit is set to 1, the PCA0L register cannot be modified by software. To change the contents of the PCA0L register, the Watchdog Timer must first be disabled.

SFR Definition 29.6. PCA0H: PCA0 Counter/Timer High Byte

Bit	7	6	5	4	3	2	1	0
Name	PCA0[15:8]							
Type	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

SFR Address = 0xFA

Bit	Name	Function
7:0	PCA0[15:8]	**PCA Counter/Timer High Byte.** The PCA0H register holds the high byte (MSB) of the 16-bit PCA Counter/Timer. Reads of this register will read the contents of a “snapshot” register, whose contents are updated only when the contents of PCA0L are read (see Section 29.1).

Note: When the WDTE bit is set to 1, the PCA0H register cannot be modified by software. To change the contents of the PCA0H register, the Watchdog Timer must first be disabled.