# EFM8 Sleepy Bee Family
# EFM8SB1 Reference Manual

The EFM8SB1, part of the Sleepy Bee family of MCUs, is the world's most energy friendly 8-bit microcontrollers with a comprehensive feature set in small packages.

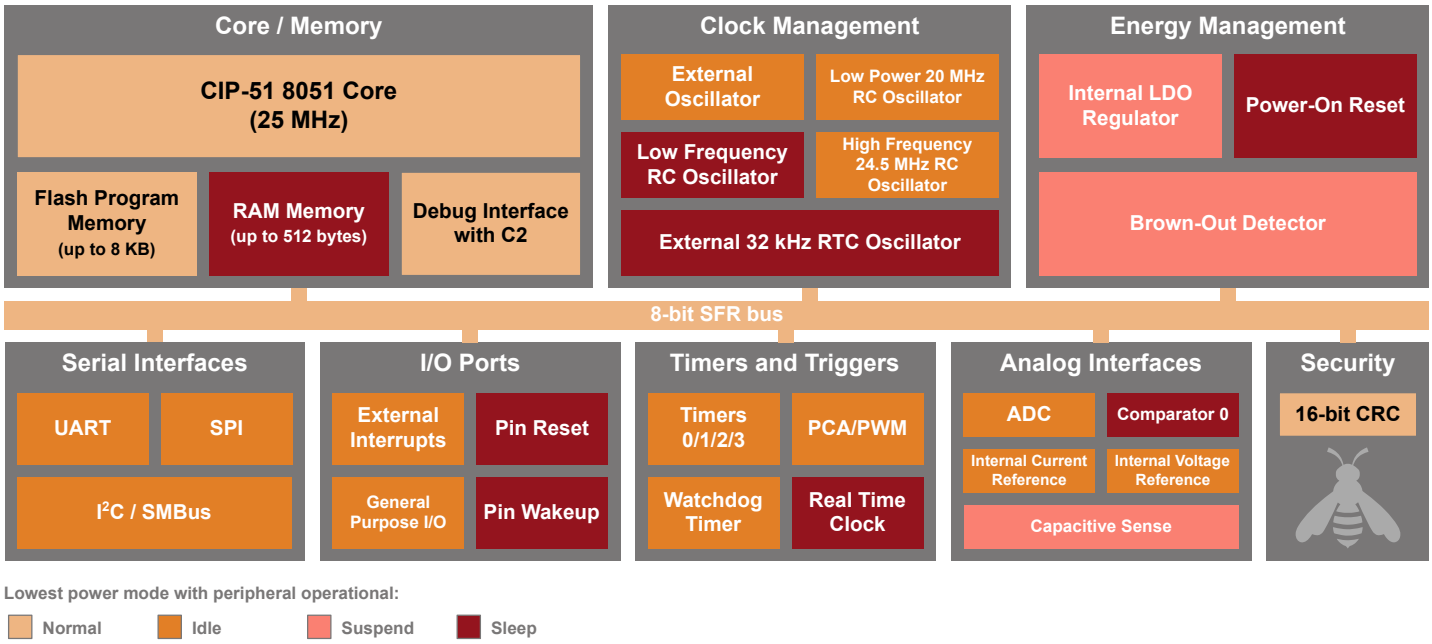
These devices offer lowest power consumption by combining innovative low energy techniques and short wakeup times from energy saving modes into small packages, making them well-suited for any battery operated applications. With an efficient 8051 core, 14 high-quality capacitive sense channels, and precision analog, the EFM8SB1 family is also optimal for embedded applications.

EFM8SB1 applications include the following:

- Touch pads / key pads
- Wearables
- Instrumentation panels
- Battery-operated consumer electronics

**ENERGY FRIENDLY FEATURES**

- Lowest MCU sleep current with supply brownout (50 nA)
- Lowest MCU active current (150 µA / MHz at 24.5 MHz)
- Lowest MCU wake on touch average current (< 1 µA)
- Lowest sleep current using internal RTC and supply brownout (< 300 nA)
- Ultra-fast wake up for digital and analog peripherals (< 2 µs)
- Integrated LDO to maintain ultra-low active current at all voltages



# 1. System Overview

## 1.1 Introduction

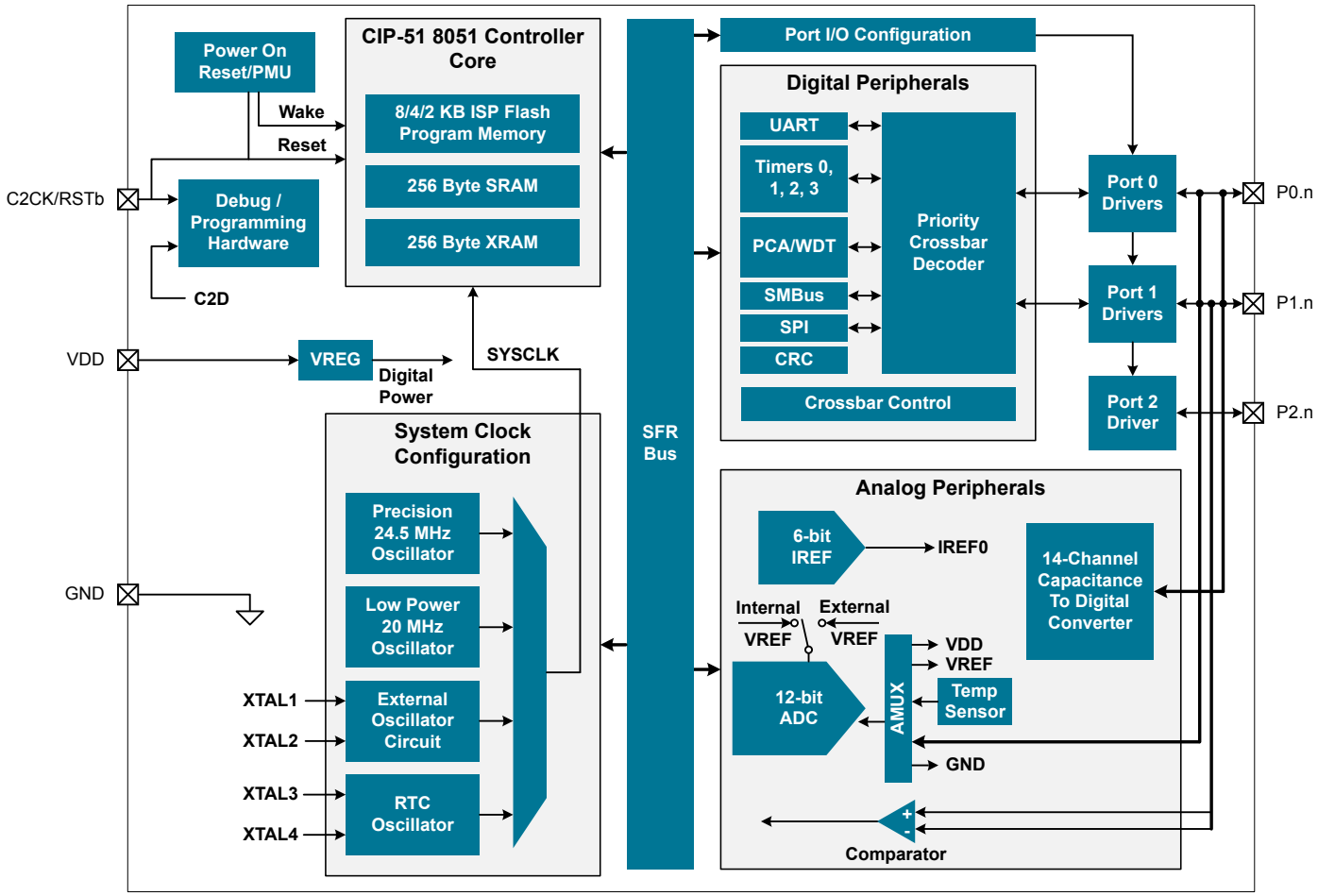



**Figure 1.1. Detailed EFM8SB1 Block Diagram**

### 1.2 Power

All internal circuitry draws power from the VDD supply pin. External I/O pins are powered from the VIO supply voltage (or VDD on devices without a separate VIO connection), while most of the internal circuitry is supplied by an on-chip LDO regulator. Control over the device power can be achieved by enabling/disabling individual peripherals as needed. Each analog peripheral can be disabled when not in use and placed in low power mode. Digital peripherals, such as timers and serial buses, have their clocks gated off and draw little power when they are not in use.

**Table 1.1. Power Modes**

| Power Mode | Details | Mode Entry | Wake-Up Sources |
|---|---|---|---|
| Normal | Core and all peripherals clocked and fully operational | — | — |
| Idle | • Core halted<br>• All peripherals clocked and fully operational<br>• Code resumes execution on wake event | Set IDLE bit in PCON0 | Any interrupt |
| Suspend | • Core and digital peripherals halted<br>• Internal oscillators disabled<br>• Code resumes execution on wake event | 1. Switch SYSCLK to HFOSC0 or LPOSC0<br>2. Set SUSPEND bit in PMU0CF | • RTC0 Alarm Event<br>• RTC0 Fail Event<br>• CS0 Interrupt<br>• Port Match Event<br>• Comparator 0 Rising Edge |
| Sleep | • Most internal power nets shut down<br>• Select circuits remain powered<br>• Pins retain state<br>• All RAM and SFRs retain state<br>• Code resumes execution on wake event | 1. Disable unused analog peripherals<br>2. Set SLEEP bit in PMU0CF | • RTC0 Alarm Event<br>• RTC0 Fail Event<br>• Port Match Event<br>• Comparator 0 Rising Edge |

### 1.3 I/O

Digital and analog resources are externally available on the device's multi-purpose I/O pins. Port pins P0.0-P1.7 can be defined as general-purpose I/O (GPIO), assigned to one of the internal digital resources through the crossbar or dedicated channels, or assigned to an analog function. Port pin P2.7 can be used as GPIO. Additionally, the C2 Interface Data signal (C2D) is shared with P2.7.

- Up to 17 multi-functions I/O pins, supporting digital and analog functions.
- Flexible priority crossbar decoder for digital peripheral assignment.
- Two drive strength settings for each pin.
- Two direct-pin interrupt sources with dedicated interrupt vectors (INT0 and INT1).
- Up to 16 direct-pin interrupt sources with shared interrupt vector (Port Match).

### 1.4 Clocking

The CPU core and peripheral subsystem may be clocked by both internal and external oscillator resources. By default, the system clock comes up running from the 20 MHz low power oscillator divided by 8.

- Provides clock to core and peripherals.
- 20 MHz low power oscillator (LPOSC0), accurate to ±10% over supply and temperature corners.
- 24.5 MHz internal oscillator (HFOSC0), accurate to ±2% over supply and temperature corners.
- 16.4 kHz low-frequency oscillator (LFOSC0) or external RTC 32 kHz crystal.
- External RC, C, CMOS, and high-frequency crystal clock options (EXTCLK).
- Clock divider with eight settings for flexible clock scaling: Divide the selected clock source by 1, 2, 4, 8, 16, 32, 64, or 128.

## 1.5 Counters/Timers and PWM

### Real Time Clock (RTC0)

The RTC is an ultra low power, 36 hour 32-bit independent time-keeping Real Time Clock with alarm. The RTC has a dedicated 32 kHz oscillator. No external resistor or loading capacitors are required, and a missing clock detector features alerts the system if the external crystal fails. The on-chip loading capacitors are programmable to 16 discrete levels allowing compatibility with a wide range of crystals.

The RTC module includes the following features:

- Up to 36 hours (32-bit) of independent time keeping.
- Support for internal 16.4 kHz low frequency oscillator (LFOSC0) or external 32 kHz crystal.
- Internal crystal loading capacitors with 16 levels.
- Operation in the lowest power mode and across the full supported voltage range.
- Alarm and oscillator failure events to wake from the lowest power mode or reset the device.
- Buffered clock output available for other system devices even in the lowest power mode.

### Programmable Counter Array (PCA0)

The programmable counter array (PCA) provides multiple channels of enhanced timer and PWM functionality while requiring less CPU intervention than standard counter/timers. The PCA consists of a dedicated 16-bit counter/timer and one 16-bit capture/compare module for each channel. The counter/timer is driven by a programmable timebase that has flexible external and internal clocking options. Each capture/compare module may be configured to operate independently in one of five modes: Edge-Triggered Capture, Software Timer, High-Speed Output, Frequency Output, or Pulse-Width Modulated (PWM) Output. Each capture/compare module has its own associated I/O line (CEXn) which is routed through the crossbar to port I/O when enabled.

- 16-bit time base.
- Programmable clock divisor and clock source selection.
- Up to three independently-configurable channels
- 8, 9, 10, 11 and 16-bit PWM modes (edge-aligned operation).
- Frequency output mode.
- Capture on rising, falling or any edge.
- Compare function for arbitrary waveform generation.
- Software timer (internal compare) mode.
- Integrated watchdog timer.

### Timers (Timer 0, Timer 1, Timer 2, and Timer 3)

Several counter/timers are included in the device: two are 16-bit counter/timers compatible with those found in the standard 8051, and the rest are 16-bit auto-reload timers for timing peripherals or for general purpose use. These timers can be used to measure time intervals, count external events and generate periodic interrupt requests. Timer 0 and Timer 1 are nearly identical and have four primary modes of operation. The other timers offer both 16-bit and split 8-bit timer functionality with auto-reload and capture capabilities.

Timer 0 and Timer 1 include the following features:

- Standard 8051 timers, supporting backwards-compatibility with firmware and hardware.
- Clock sources include SYSCLK, SYSCLK divided by 12, 4, or 48, the External Clock divided by 8, or an external pin.
- 8-bit auto-reload counter/timer mode
- 13-bit counter/timer mode
- 16-bit counter/timer mode
- Dual 8-bit counter/timer mode (Timer 0)

Timer 2 and Timer 3 are 16-bit timers including the following features:

- Clock sources include SYSCLK, SYSCLK divided by 12, or the External Clock divided by 8.
- 16-bit auto-reload timer mode
- Dual 8-bit auto-reload timer mode
- Comparator 0 or RTC0 capture (Timer 2)
- RTC0 or EXTCLK/8 capture (Timer 3)

### Watchdog Timer (WDT0)

The device includes a programmable watchdog timer (WDT) integrated within the PCA0 peripheral. A WDT overflow forces the MCU into the reset state. To prevent the reset, the WDT must be restarted by application software before overflow. If the system experiences a software or hardware malfunction preventing the software from restarting the WDT, the WDT overflows and causes a reset. Following a reset, the WDT is automatically enabled and running with the default maximum time interval. If needed, the WDT can be disabled by system software. The state of the RSTb pin is unaffected by this reset.

The Watchdog Timer integrated in the PCA0 peripheral has the following features:

- Programmable timeout interval
- Runs from the selected PCA clock source
- Automatically enabled after any system reset

## 1.6 Communications and Other Digital Peripherals

### Universal Asynchronous Receiver/Transmitter (UART0)

UART0 is an asynchronous, full duplex serial port offering modes 1 and 3 of the standard 8051 UART. Enhanced baud rate support allows a wide range of clock sources to generate standard baud rates. Received data buffering allows UART0 to start reception of a second incoming data byte before software has finished reading the previous data byte.

The UART module provides the following features:

- Asynchronous transmissions and receptions
- Baud rates up to SYSCLK/2 (transmit) or SYSCLK/8 (receive)
- 8- or 9-bit data
- Automatic start and stop generation

### Serial Peripheral Interface (SPI0)

The serial peripheral interface (SPI) module provides access to a flexible, full-duplex synchronous serial bus. The SPI can operate as a master or slave device in both 3-wire or 4-wire modes, and supports multiple masters and slaves on a single SPI bus. The slave-select (NSS) signal can be configured as an input to select the SPI in slave mode, or to disable master mode operation in a multi-master environment, avoiding contention on the SPI bus when more than one master attempts simultaneous data transfers. NSS can also be configured as a firmware-controlled chip-select output in master mode, or disabled to reduce the number of pins required. Additional general purpose port I/O pins can be used to select multiple slave devices in master mode.

The SPI module includes the following features:

- Supports 3- or 4-wire operation in master or slave modes.
- Supports external clock frequencies up to SYSCLK / 2 in master mode and SYSCLK / 10 in slave mode.
- Support for four clock phase and polarity options.
- 8-bit dedicated clock clock rate generator.
- Support for multiple masters on the same data lines.

### System Management Bus / I2C (SMB0)

The SMBus I/O interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the $I^2C$ serial bus.

The SMBus module includes the following features:

- Standard (up to 100 kbps) and Fast (400 kbps) transfer speeds.
- Support for master, slave, and multi-master modes.
- Hardware synchronization and arbitration for multi-master mode.
- Clock low extending (clock stretching) to interface with faster masters.
- Hardware support for 7-bit slave and general call address recognition.
- Firmware support for 10-bit slave address decoding.
- Ability to inhibit all slave states.
- Programmable data setup/hold times.

### 16-bit CRC (CRC0)

The cyclic redundancy check (CRC) module performs a CRC using a 16-bit polynomial. CRC0 accepts a stream of 8-bit data and posts the 16-bit result to an internal register. In addition to using the CRC block for data manipulation, hardware can automatically CRC the flash contents of the device.

The CRC module is designed to provide hardware calculations for flash memory verification and communications protocols. The CRC module supports the standard CCITT-16 16-bit polynomial (0x1021), and includes the following features:

- Support for CCITT-16 polynomial
- Byte-level bit reversal
- Automatic CRC of flash contents on one or more 256-byte blocks
- Initial seed selection of 0x0000 or 0xFFFF

## 1.7 Analog

### Capacitive Sense (CS0)

The Capacitive Sense subsystem uses a capacitance-to-digital circuit to determine the capacitance on a port pin. The module can take measurements from different port pins using the module's analog multiplexer. The module can be configured to take measurements on one port pin, a group of port pins one-by-one using auto-scan, or the total capacitance on multiple channels together. A selectable gain circuit allows the designer to adjust the maximum allowable capacitance. An accumulator is also included, which can be configured to average multiple conversions on an input channel. Interrupts can be generated when the CS0 peripheral completes a conversion or when the measured value crosses a configurable threshold.

The Capacitive Sense module includes the following features:

- Measure multiple pins one-by-one using auto-scan or total capacitance on multiple channels together.
- Configurable input gain.
- Hardware auto-accumulate and average.
- Multiple internal start-of-conversion sources.
- Operational in Suspend when all other clocks are disabled.
- Interrupts available at the end of a conversion or when the measured value crosses a configurable threshold.

### Programmable Current Reference (IREF0)

The programmable current reference (IREF0) module enables current source or sink with two output current settings: Low Power Mode and High Current Mode. The maximum current output in Low Power Mode is 63 µA (1 µA steps) and the maximum current output in High Current Mode is 504 µA (8 µA steps).

The IREF module includes the following features:

- Capable of sourcing or sinking current in programmable steps.
- Two operational modes: Low Power Mode and High Current Mode.
- Fine-tuning mode for higher output precision available in conjunction with the PCA0 module.

### 12-Bit Analog-to-Digital Converter (ADC0)

The ADC is a successive-approximation-register (SAR) ADC with 12-, 10-, and 8-bit modes, integrated track-and hold and a programmable window detector. The ADC is fully configurable under software control via several registers. The ADC may be configured to measure different signals using the analog multiplexer. The voltage reference for the ADC is selectable between internal and external reference sources.

- Up to 10 external inputs.
- Single-ended 12-bit and 10-bit modes.
- Supports an output update rate of 75 ksps samples per second in 12-bit mode or 300 ksps samples per second in 10-bit mode.
- Operation in low power modes at lower conversion speeds.
- Asynchronous hardware conversion trigger, selectable between software, external I/O and internal timer sources.
- Output data window comparator allows automatic range checking.
- Support for burst mode, which produces one set of accumulated data per conversion-start trigger with programmable power-on settling and tracking time.
- Conversion complete and window compare interrupts supported.
- Flexible output data formatting.
- Includes an internal 1.65 V fast-settling reference and support for external reference.
- Integrated temperature sensor.

### Low Current Comparator (CMP0)

An analog comparator is used to compare the voltage of two analog inputs, with a digital output indicating which input voltage is higher. External input connections to device I/O pins and internal connections are available through separate multiplexers on the positive and negative inputs. Hysteresis, response time, and current consumption may be programmed to suit the specific needs of the application.

The comparator module includes the following features:

- Input options in addition to the pins:
  - Capacitive Sense Comparator output.
  - VDD.
  - VDD divided by 2.
  - Internal connection to LDO output.
  - Direct connection to GND.
- Synchronous and asynchronous outputs can be routed to pins via crossbar.
- Programmable hysteresis between 0 and ±20 mV.
- Programmable response time.
- Interrupts generated on rising, falling, or both edges.

### 1.8 Reset Sources

Reset circuitry allows the controller to be easily placed in a predefined default condition. On entry to this reset state, the following occur:

- The core halts program execution.
- Module registers are initialized to their defined reset values unless the bits reset only with a power-on reset.
- External port pins are forced to a known state.
- Interrupts and timers are disabled.

All registers are reset to the predefined values noted in the register descriptions unless the bits only reset with a power-on reset. The contents of RAM are unaffected during a reset; any previously stored data is preserved as long as power is not lost. The Port I/O latches are reset to 1 in open-drain mode. Weak pullups are enabled during and after the reset. For Supply Monitor and power-on resets, the RSTb pin is driven low until the device exits the reset state. On exit from the reset state, the program counter (PC) is reset, and the system clock defaults to an internal oscillator. The Watchdog Timer is enabled, and program execution begins at location 0x0000.

Reset sources on the device include the following:

- Power-on reset
- External reset pin
- Comparator reset
- Software-triggered reset
- Supply monitor reset (monitors VDD supply)
- Watchdog timer reset
- Missing clock detector reset
- Flash error reset
- RTC0 alarm or oscillator failure

### 1.9 Debugging

The EFM8SB1 devices include an on-chip Silicon Labs 2-Wire (C2) debug interface to allow flash programming and in-system debugging with the production part installed in the end application. The C2 interface uses a clock signal (C2CK) and a bi-directional C2 data signal (C2D) to transfer information between the device and a host system. See the C2 Interface Specification for details on the C2 protocol.

### 1.10 Bootloader

All devices come pre-programmed with a UART bootloader. This bootloader resides in flash and can be erased if it is not needed.

# 2. Memory Organization

## 2.1 Memory Organization

The memory organization of the CIP-51 System Controller is similar to that of a standard 8051. There are two separate memory spaces: program memory and data memory. Program and data memory share the same address space but are accessed via different instruction types. Program memory consists of a non-volatile storage area that may be used for either program code or non-volatile data storage. The data memory, consisting of "internal" and "external" data space, is implemented as RAM, and may be used only for data storage. Program execution is not supported from the data memory space.

## 2.2 Program Memory

The CIP-51 core has a 64 KB program memory space. The product family implements some of this program memory space as in-system, re-programmable flash memory. Flash security is implemented by a user-programmable location in the flash block and provides read, write, and erase protection. All addresses not specified in the device memory map are reserved and may not be used for code or data storage.

### MOVX Instruction and Program Memory

The MOVX instruction in an 8051 device is typically used to access external data memory. On the devices, the MOVX instruction is normally used to read and write on-chip XRAM, but can be re-configured to write and erase on-chip flash memory space. MOVC instructions are always used to read flash memory, while MOVX write instructions are used to erase and write flash. This flash access feature provides a mechanism for the product to update program code and use the program memory space for non-volatile data storage.

## 2.3 Data Memory

The RAM space on the chip includes both an "internal" RAM area which is accessed with MOV instructions, and an on-chip "external" RAM area which is accessed using MOVX instructions. Total RAM varies, based on the specific device. The device memory map has more details about the specific amount of RAM available in each area for the different device variants.

### Internal RAM

There are 256 bytes of internal RAM mapped into the data memory space from 0x00 through 0xFF. The lower 128 bytes of data memory are used for general purpose registers and scratch pad memory. Either direct or indirect addressing may be used to access the lower 128 bytes of data memory. Locations 0x00 through 0x1F are addressable as four banks of general purpose registers, each bank consisting of eight byte-wide registers. The next 16 bytes, locations 0x20 through 0x2F, may either be addressed as bytes or as 128 bit locations accessible with the direct addressing mode.

The upper 128 bytes of data memory are accessible only by indirect addressing. This region occupies the same address space as the Special Function Registers (SFR) but is physically separate from the SFR space. The addressing mode used by an instruction when accessing locations above 0x7F determines whether the CPU accesses the upper 128 bytes of data memory space or the SFRs. Instructions that use direct addressing will access the SFR space. Instructions using indirect addressing above 0x7F access the upper 128 bytes of data memory.

### General Purpose Registers

The lower 32 bytes of data memory, locations 0x00 through 0x1F, may be addressed as four banks of general-purpose registers. Each bank consists of eight byte-wide registers designated R0 through R7. Only one of these banks may be enabled at a time. Two bits in the program status word (PSW) register, RS0 and RS1, select the active register bank. This allows fast context switching when entering subroutines and interrupt service routines. Indirect addressing modes use registers R0 and R1 as index registers.

### Bit Addressable Locations

In addition to direct access to data memory organized as bytes, the sixteen data memory locations at 0x20 through 0x2F are also accessible as 128 individually addressable bits. Each bit has a bit address from 0x00 to 0x7F. Bit 0 of the byte at 0x20 has bit address 0x00 while bit 7 of the byte at 0x20 has bit address 0x07. Bit 7 of the byte at 0x2F has bit address 0x7F. A bit access is distinguished from a full byte access by the type of instruction used (bit source or destination operands as opposed to a byte source or destination).

The MCS-51™ assembly language allows an alternate notation for bit addressing of the form XX.B where XX is the byte address and B is the bit position within the byte. For example, the instruction:

```
Mov  C, 22.3h
```

moves the Boolean value at 0x13 (bit 3 of the byte at location 0x22) into the Carry flag.

### Stack

A programmer's stack can be located anywhere in the 256-byte data memory. The stack area is designated using the Stack Pointer (SP) SFR. The SP will point to the last location used. The next value pushed on the stack is placed at SP+1 and then SP is incremented. A reset initializes the stack pointer to location 0x07. Therefore, the first value pushed on the stack is placed at location 0x08, which is also the first register (R0) of register bank 1. Thus, if more than one register bank is to be used, the SP should be initialized to a location in the data memory not being used for data storage. The stack depth can extend up to 256 bytes.

### External RAM

On devices with more than 256 bytes of on-chip RAM, the additional RAM is mapped into the external data memory space (XRAM). Addresses in XRAM area accessed using the external move (MOVX) instructions.

**Note:** The 16-bit MOVX write instruction is also used for writing and erasing the flash memory. More details may be found in the flash memory section.

**Note:** On device reset or upon waking up from Sleep mode, address 0x0000 of external memory (XRAM) may be overwritten by an indeterminate value. The indeterminate value is 0x00 in most situations. A dummy variable should be placed at address 0x0000 in external memory to ensure that the application firmware does not store any data that needs to be retained through reset or Sleep at this memory location.

## 2.4 Memory Map

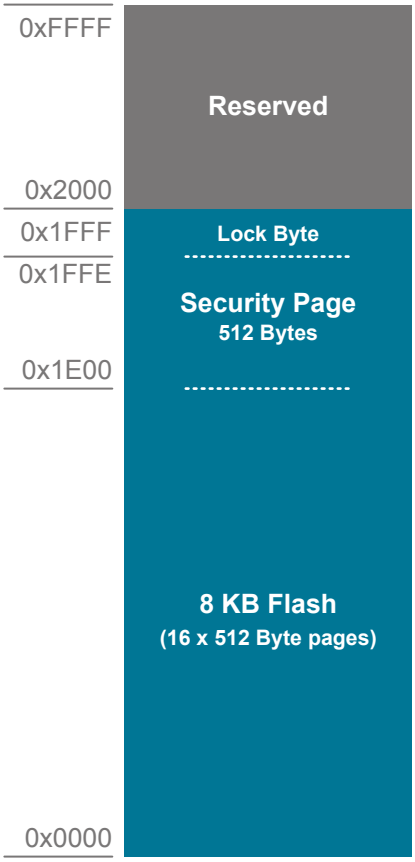



**Figure 2.1. Flash Memory Map — 8 KB Devices**

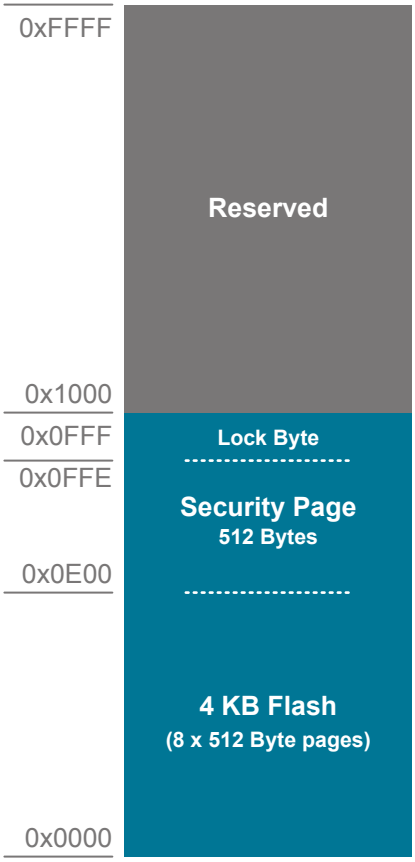



**Figure 2.2. Flash Memory Map — 4 KB Devices**

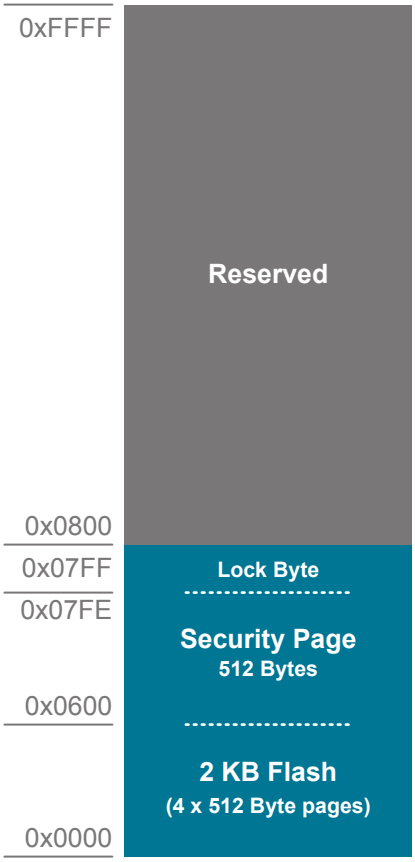



**Figure 2.3. Flash Memory Map — 2 KB Devices**

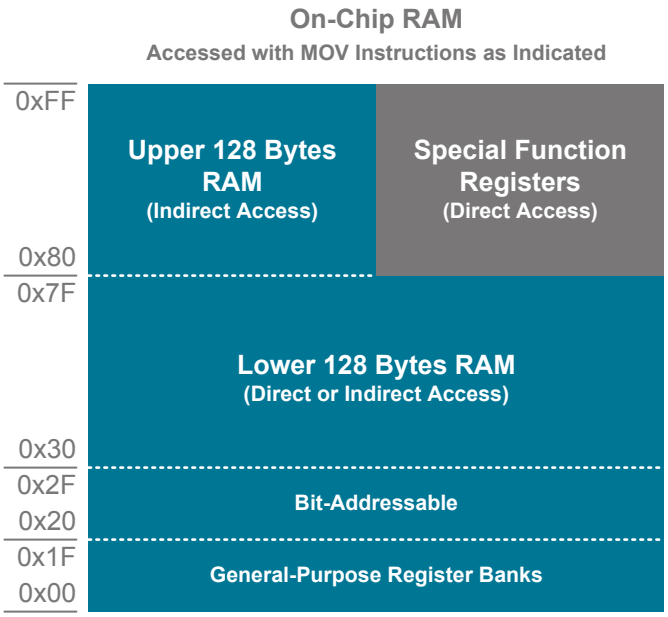



**Figure 2.4. Direct / Indirect RAM Memory**

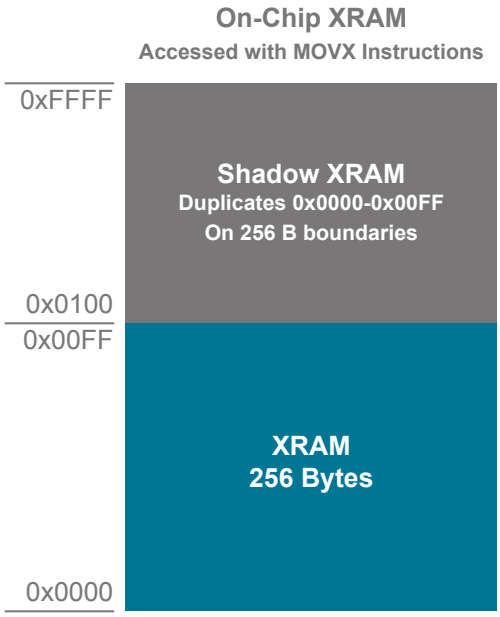



**Figure 2.5. XRAM Memory**

# 3. Special Function Registers

## 3.1 Special Function Register Access

The direct-access data memory locations from 0x80 to 0xFF constitute the special function registers (SFRs). The SFRs provide control and data exchange with the CIP-51's resources and peripherals. The CIP-51 duplicates the SFRs found in a typical 8051 implementation as well as implementing additional SFRs used to configure and access the sub-systems unique to the MCU. This allows the addition of new functionality while retaining compatibility with the MCS-51 ™ instruction set.

The SFR registers are accessed anytime the direct addressing mode is used to access memory locations from 0x80 to 0xFF. SFRs with addresses ending in 0x0 or 0x8 (e.g., P0, TCON, SCON0, IE, etc.) are bit-addressable as well as byte-addressable. All other SFRs are byte-addressable only. Unoccupied addresses in the SFR space are reserved for future use. Accessing these areas will have an indeterminate effect and should be avoided.

### SFR Paging

The CIP-51 features SFR paging, allowing the device to map many SFRs into the 0x80 to 0xFF memory address space. The SFR memory space has 256 pages. In this way, each memory location from 0x80 to 0xFF can access up to 256 SFRs. The EFM8SB1x devices utilize multiple SFR pages. All of the common 8051 SFRs are available on all pages. Certain SFRs are only available on a subset of pages. SFR pages are selected using the SFRPAGE register. The procedure for reading and writing an SFR is as follows:

1. Select the appropriate SFR page using the SFRPAGE register.
2. Use direct accessing mode to read or write the special function register (MOV instruction).

The SFRPAGE register only needs to be changed in the case that the SFR to be accessed does not exist on the currently-selected page. See the SFR memory map for details on the locations of each SFR. It is good practice inside of interrupt service routines to save the current SFRPAGE at the beginning of the ISR and restore this value at the end.

### Interrupts and SFR Paging

In any system which changes the SFRPAGE while interrupts are active, it is good practice to save the current SFRPAGE value upon ISR entry, and then restore the SFRPAGE before exiting the ISR. This ensures that SFRPAGE will remain at the desired setting when returning from the ISR.

## 3.2 Special Function Register Memory Map

**Table 3.1. Special Function Registers by Address**

| Address | SFR Page | | Address | SFR Page | |
|---|---|---|---|---|---|
| (*bit-addressable) | 0x00 | 0x0F | (*bit-addressable) | 0x00 | 0x0F |
| 0x80* | P0 | | 0xC0* | SMB0CN0 | - |
| 0x81 | SP | | 0xC1 | SMB0CF | - |
| 0x82 | DPL | | 0xC2 | SMB0DAT | - |
| 0x83 | DPH | | 0xC3 | ADC0GTL | - |
| 0x84 | CRC0CN0 | | 0xC4 | ADC0GTH | - |
| 0x85 | CRC0IN | | 0xC5 | ADC0LTL | - |
| 0x86 | CRC0DAT | | 0xC6 | ADC0LTH | - |
| 0x87 | PCON0 | | 0xC7 | P0MASK | - |
| 0x88* | TCON | - | 0xC8* | TMR2CN0 | - |
| 0x89 | TMOD | - | 0xC9 | REG0CN | - |
| 0x8A | TL0 | - | 0xCA | TMR2RLL | - |
| 0x8B | TL1 | - | 0xCB | TMR2RLH | - |
| 0x8C | TH0 | - | 0xCC | TMR2L | - |
| 0x8D | TH1 | TOFFL | 0xCD | TMR2H | - |
| 0x8E | CKCON0 | TOFFH | 0xCE | PMU0FL | - |
| 0x8F | PSCTL | - | 0xCF | P1MAT | - |
| 0x90* | P1 | | 0xD0* | PSW | |
| 0x91 | TMR3CN0 | - | 0xD1 | REF0CN | - |
| 0x92 | TMR3RLL | - | 0xD2 | CS0SCAN0 | - |
| 0x93 | TMR3RLH | - | 0xD3 | CS0SCAN1 | - |
| 0x94 | TMR3L | - | 0xD4 | P0SKIP | - |
| 0x95 | TMR3H | - | 0xD5 | P1SKIP | - |
| 0x96 | ADC0MX | - | 0xD6 | IREF0CN0 | - |
| 0x97 | ADC0CF | - | 0xD7 | P0MAT | - |
| 0x98* | SCON0 | - | 0xD8* | PCA0CN0 | - |
| 0x99 | SBUF0 | P0DRV | 0xD9 | PCA0MD | - |
| 0x9A | CRC0CNT | | 0xDA | PCA0CPM0 | - |
| 0x9B | CMP0CN0 | P1DRV | 0xDB | PCA0CPM1 | - |
| 0x9C | CRC0FLIP | | 0xDC | PCA0CPM2 | - |
| 0x9D | CMP0MD | P2DRV | 0xDD | CS0SS | - |
| 0x9E | CRC0AUTO | | 0xDE | CS0SE | CS0PM |
| 0x9F | CMP0MX | - | 0xDF | PCA0PWM | - |
| 0xA0* | P2 | | 0xE0* | ACC | |

| Address | SFR Page | | Address | SFR Page | |
|---|---|---|---|---|---|
| (*bit-addressable) | 0x00 | 0x0F | (*bit-addressable) | 0x00 | 0x0F |
| 0xA1 | SPI0CFG | - | 0xE1 | XBR0 | - |
| 0xA2 | SPI0CKR | - | 0xE2 | XBR1 | REVID |
| 0xA3 | SPI0DAT | - | 0xE3 | XBR2 | DERIVID |
| 0xA4 | P0MDOUT | - | 0xE4 | IT01CF | - |
| 0xA5 | P1MDOUT | - | 0xE5 | - | |
| 0xA6 | P2MDOUT | - | 0xE6 | EIE1 | |
| 0xA7 | SFRPAGE | | 0xE7 | EIE2 | |
| 0xA8* | IE | | 0xE8* | ADC0CN0 | - |
| 0xA9 | CLKSEL | | 0xE9 | PCA0CPL1 | - |
| 0xAA | CS0CF | - | 0xEA | PCA0CPH1 | - |
| 0xAB | CS0MX | - | 0xEB | PCA0CPL2 | - |
| 0xAC | RTC0ADR | - | 0xEC | PCA0CPH2 | - |
| 0xAD | RTC0DAT | - | 0xED | CS0DL | - |
| 0xAE | RTC0KEY | - | 0xEE | CS0DH | - |
| 0xAF | CS0MD1 | - | 0xEF | RSTSRC | - |
| 0xB0* | CS0CN0 | - | 0xF0* | B | |
| 0xB1 | XOSC0CN | - | 0xF1 | P0MDIN | - |
| 0xB2 | HFO0CN | - | 0xF2 | P1MDIN | - |
| 0xB3 | HFO0CAL | - | 0xF3 | CS0MD2 | CS0MD3 |
| 0xB4 | - | | 0xF4 | SMB0ADR | - |
| 0xB5 | PMU0CF | PMU0MD | 0xF5 | SMB0ADM | - |
| 0xB6 | FLSCL | - | 0xF6 | EIP1 | |
| 0xB7 | FLKEY | - | 0xF7 | EIP2 | |
| 0xB8* | IP | | 0xF8* | SPI0CN0 | - |
| 0xB9 | IREF0CF | | 0xF9 | PCA0L | - |
| 0xBA | ADC0AC | - | 0xFA | PCA0H | - |
| 0xBB | ADC0PWR | | 0xFB | PCA0CPL0 | - |
| 0xBC | ADC0TK | | 0xFC | PCA0CPH0 | - |
| 0xBD | ADC0L | - | 0xFD | CS0THL | - |
| 0xBE | ADC0H | - | 0xFE | CS0THH | - |
| 0xBF | P1MASK | - | 0xFF | VDM0CN | - |

**Table 3.2. Special Function Registers by Name**

| Register | Address | SFR Pages | Description |
|---|---|---|---|
| ACC | 0xE0 | ALL | Accumulator |
| ADC0AC | 0xBA | 0x00 | ADC0 Accumulator Configuration |

| Register | Address | SFR Pages | Description |
|---|---|---|---|
| ADC0CF | 0x97 | 0x00 | ADC0 Configuration |
| ADC0CN0 | 0xE8 | 0x00 | ADC0 Control 0 |
| ADC0GTH | 0xC4 | 0x00 | ADC0 Greater-Than High Byte |
| ADC0GTL | 0xC3 | 0x00 | ADC0 Greater-Than Low Byte |
| ADC0H | 0xBE | 0x00 | ADC0 Data Word High Byte |
| ADC0L | 0xBD | 0x00 | ADC0 Data Word Low Byte |
| ADC0LTH | 0xC6 | 0x00 | ADC0 Less-Than High Byte |
| ADC0LTL | 0xC5 | 0x00 | ADC0 Less-Than Low Byte |
| ADC0MX | 0x96 | 0x00 | ADC0 Multiplexer Selection |
| ADC0PWR | 0xBB | ALL | ADC0 Power Control |
| ADC0TK | 0xBC | ALL | ADC0 Burst Mode Track Time |
| B | 0xF0 | ALL | B Register |
| CKCON0 | 0x8E | 0x00 | Clock Control 0 |
| CLKSEL | 0xA9 | ALL | Clock Select |
| CMP0CN0 | 0x9B | 0x00 | Comparator 0 Control 0 |
| CMP0MD | 0x9D | 0x00 | Comparator 0 Mode |
| CMP0MX | 0x9F | 0x00 | Comparator 0 Multiplexer Selection |
| CRC0AUTO | 0x9E | ALL | CRC0 Automatic Control |
| CRC0CN0 | 0x84 | ALL | CRC0 Control 0 |
| CRC0CNT | 0x9A | ALL | CRC0 Automatic Flash Sector Count |
| CRC0DAT | 0x86 | ALL | CRC0 Data Output |
| CRC0FLIP | 0x9C | ALL | CRC0 Bit Flip |
| CRC0IN | 0x85 | ALL | CRC0 Data Input |
| CS0CF | 0xAA | 0x00 | Capacitive Sense 0 Configuration |
| CS0CN0 | 0xB0 | 0x00 | Capacitive Sense 0 Control |
| CS0DH | 0xEE | 0x00 | Capacitive Sense 0 Data High Byte |
| CS0DL | 0xED | 0x00 | Capacitive Sense 0 Data Low Byte |
| CS0MD1 | 0xAF | 0x00 | Capacitive Sense 0 Mode 1 |
| CS0MD2 | 0xF3 | 0x00 | Capacitive Sense 0 Mode 2 |
| CS0MD3 | 0xF3 | 0x0F | Capacitive Sense 0 Mode 3 |
| CS0MX | 0xAB | 0x00 | Capacitive Sense 0 Mux Channel Select |
| CS0PM | 0xDE | 0x0F | Capacitive Sense 0 Pin Monitor |
| CS0SCAN0 | 0xD2 | 0x00 | Capacitive Sense 0 Channel Scan Mask 0 |
| CS0SCAN1 | 0xD3 | 0x00 | Capacitive Sense 0 Channel Scan Mask 1 |
| CS0SE | 0xDE | 0x00 | Capacitive Sense 0 Auto Scan End Channel |
| CS0SS | 0xDD | 0x00 | Capacitive Sense 0 Auto Scan Start Channel |
| CS0THH | 0xFE | 0x00 | Capacitive Sense 0 Comparator Threshold High Byte |

| Register | Address | SFR Pages | Description |
|---|---|---|---|
| CS0THL | 0xFD | 0x00 | Capacitive Sense 0 Comparator Threshold Low Byte |
| DERIVID | 0xE3 | 0x0F | Device Identification |
| DPH | 0x83 | ALL | Data Pointer High |
| DPL | 0x82 | ALL | Data Pointer Low |
| EIE1 | 0xE6 | ALL | Extended Interrupt Enable 1 |
| EIE2 | 0xE7 | ALL | Extended Interrupt Enable 2 |
| EIP1 | 0xF6 | ALL | Extended Interrupt Priority 1 |
| EIP2 | 0xF7 | ALL | Extended Interrupt Priority 2 |
| FLKEY | 0xB7 | 0x00 | Flash Lock and Key |
| FLSCL | 0xB6 | 0x00 | Flash Scale |
| HFO0CAL | 0xB3 | 0x00 | High Frequency Oscillator Calibration |
| HFO0CN | 0xB2 | 0x00 | High Frequency Oscillator Control |
| IE | 0xA8 | ALL | Interrupt Enable |
| IP | 0xB8 | ALL | Interrupt Priority |
| IREF0CF | 0xB9 | ALL | Current Reference Configuration |
| IREF0CN0 | 0xD6 | 0x00 | Current Reference Control 0 |
| IT01CF | 0xE4 | 0x00 | INT0/INT1 Configuration |
| P0 | 0x80 | ALL | Port 0 Pin Latch |
| P0DRV | 0x99 | 0x0F | Port 0 Drive Strength |
| P0MASK | 0xC7 | 0x00 | Port 0 Mask |
| P0MAT | 0xD7 | 0x00 | Port 0 Match |
| P0MDIN | 0xF1 | 0x00 | Port 0 Input Mode |
| P0MDOUT | 0xA4 | 0x00 | Port 0 Output Mode |
| P0SKIP | 0xD4 | 0x00 | Port 0 Skip |
| P1 | 0x90 | ALL | Port 1 Pin Latch |
| P1DRV | 0x9B | 0x0F | Port 1 Drive Strength |
| P1MASK | 0xBF | 0x00 | Port 1 Mask |
| P1MAT | 0xCF | 0x00 | Port 1 Match |
| P1MDIN | 0xF2 | 0x00 | Port 1 Input Mode |
| P1MDOUT | 0xA5 | 0x00 | Port 1 Output Mode |
| P1SKIP | 0xD5 | 0x00 | Port 1 Skip |
| P2 | 0xA0 | ALL | Port 2 Pin Latch |
| P2DRV | 0x9D | 0x0F | Port 2 Drive Strength |
| P2MDOUT | 0xA6 | 0x00 | Port 2 Output Mode |
| PCA0CN0 | 0xD8 | 0x00 | PCA Control 0 |
| PCA0CPH0 | 0xFC | 0x00 | PCA Channel 0 Capture Module High Byte |
| PCA0CPH1 | 0xEA | 0x00 | PCA Channel 1 Capture Module High Byte |

| Register | Address | SFR Pages | Description |
|---|---|---|---|
| PCA0CPH2 | 0xEC | 0x00 | PCA Channel 2 Capture Module High Byte |
| PCA0CPL0 | 0xFB | 0x00 | PCA Channel 0 Capture Module Low Byte |
| PCA0CPL1 | 0xE9 | 0x00 | PCA Channel 1 Capture Module Low Byte |
| PCA0CPL2 | 0xEB | 0x00 | PCA Channel 2 Capture Module Low Byte |
| PCA0CPM0 | 0xDA | 0x00 | PCA Channel 0 Capture/Compare Mode |
| PCA0CPM1 | 0xDB | 0x00 | PCA Channel 1 Capture/Compare Mode |
| PCA0CPM2 | 0xDC | 0x00 | PCA Channel 2 Capture/Compare Mode |
| PCA0H | 0xFA | 0x00 | PCA Counter/Timer High Byte |
| PCA0L | 0xF9 | 0x00 | PCA Counter/Timer Low Byte |
| PCA0MD | 0xD9 | 0x00 | PCA Mode |
| PCA0PWM | 0xDF | 0x00 | PCA PWM Configuration |
| PCON0 | 0x87 | ALL | Power Control 0 |
| PMU0CF | 0xB5 | 0x00 | Power Management Unit Configuration |
| PMU0FL | 0xCE | 0x00 | Power Management Unit Flag |
| PMU0MD | 0xB5 | 0x0F | Power Management Unit Mode |
| PSCTL | 0x8F | 0x00 | Program Store Control |
| PSW | 0xD0 | ALL | Program Status Word |
| REF0CN | 0xD1 | 0x00 | Voltage Reference Control |
| REG0CN | 0xC9 | 0x00 | Voltage Regulator Control |
| REVID | 0xE2 | 0x0F | Revision Identifcation |
| RSTSRC | 0xEF | 0x00 | Reset Source |
| RTC0ADR | 0xAC | 0x00 | RTC Address |
| RTC0DAT | 0xAD | 0x00 | RTC Data |
| RTC0KEY | 0xAE | 0x00 | RTC Lock and Key |
| SBUF0 | 0x99 | 0x00 | UART0 Serial Port Data Buffer |
| SCON0 | 0x98 | 0x00 | UART0 Serial Port Control |
| SFRPAGE | 0xA7 | ALL | SFR Page |
| SMB0ADM | 0xF5 | 0x00 | SMBus 0 Slave Address Mask |
| SMB0ADR | 0xF4 | 0x00 | SMBus 0 Slave Address |
| SMB0CF | 0xC1 | 0x00 | SMBus 0 Configuration |
| SMB0CN0 | 0xC0 | 0x00 | SMBus 0 Control |
| SMB0DAT | 0xC2 | 0x00 | SMBus 0 Data |
| SP | 0x81 | ALL | Stack Pointer |
| SPI0CFG | 0xA1 | 0x00 | SPI0 Configuration |
| SPI0CKR | 0xA2 | 0x00 | SPI0 Clock Rate |
| SPI0CN0 | 0xF8 | 0x00 | SPI0 Control |
| SPI0DAT | 0xA3 | 0x00 | SPI0 Data |

| Register | Address | SFR Pages | Description |
|---|---|---|---|
| TCON | 0x88 | 0x00 | Timer 0/1 Control |
| TH0 | 0x8C | 0x00 | Timer 0 High Byte |
| TH1 | 0x8D | 0x00 | Timer 1 High Byte |
| TL0 | 0x8A | 0x00 | Timer 0 Low Byte |
| TL1 | 0x8B | 0x00 | Timer 1 Low Byte |
| TMOD | 0x89 | 0x00 | Timer 0/1 Mode |
| TMR2CN0 | 0xC8 | 0x00 | Timer 2 Control 0 |
| TMR2H | 0xCD | 0x00 | Timer 2 High Byte |
| TMR2L | 0xCC | 0x00 | Timer 2 Low Byte |
| TMR2RLH | 0xCB | 0x00 | Timer 2 Reload High Byte |
| TMR2RLL | 0xCA | 0x00 | Timer 2 Reload Low Byte |
| TMR3CN0 | 0x91 | 0x00 | Timer 3 Control 0 |
| TMR3H | 0x95 | 0x00 | Timer 3 High Byte |
| TMR3L | 0x94 | 0x00 | Timer 3 Low Byte |
| TMR3RLH | 0x93 | 0x00 | Timer 3 Reload High Byte |
| TMR3RLL | 0x92 | 0x00 | Timer 3 Reload Low Byte |
| TOFFH | 0x8E | 0x0F | Temperature Sensor Offset High |
| TOFFL | 0x8D | 0x0F | Temperature Sensor Offset Low |
| VDM0CN | 0xFF | 0x00 | VDD Supply Monitor Control |
| XBR0 | 0xE1 | 0x00 | Port I/O Crossbar 0 |
| XBR1 | 0xE2 | 0x00 | Port I/O Crossbar 1 |
| XBR2 | 0xE3 | 0x00 | Port I/O Crossbar 2 |
| XOSC0CN | 0xB1 | 0x00 | External Oscillator Control |

## 3.3 SFR Access Control Registers

### 3.3.1 SFRPAGE: SFR Page

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SFRPAGE | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0xA7 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | SFRPAGE | 0x00 | RW | **SFR Page.**<br>Specifies the SFR Page used when reading, writing, or modifying special function registers. |

# 4. Flash Memory

## 4.1 Introduction

On-chip, re-programmable flash memory is included for program code and non-volatile data storage. The flash memory is organized in 512-byte pages. It can be erased and written through the C2 interface or from firmware by overloading the MOVX instruction. Any individual byte in flash memory must only be written once between page erase operations.

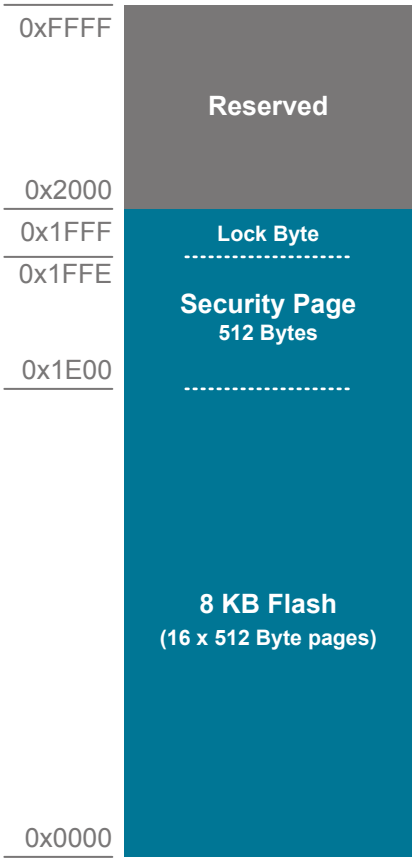



**Figure 4.1. Flash Memory Map — 8 KB Devices**

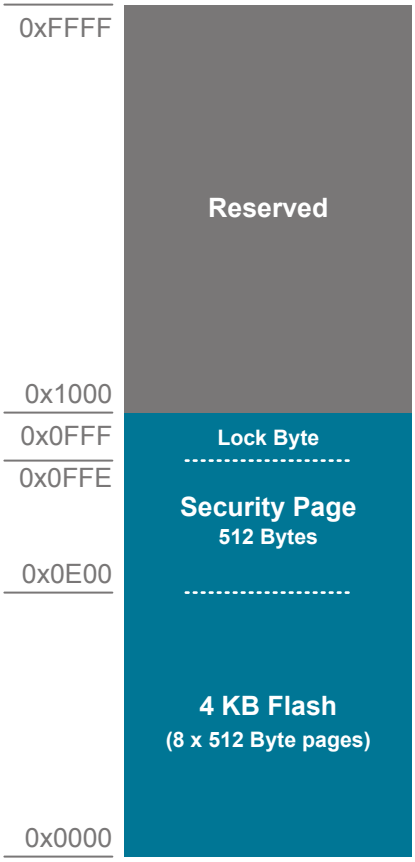



**Figure 4.2. Flash Memory Map — 4 KB Devices**

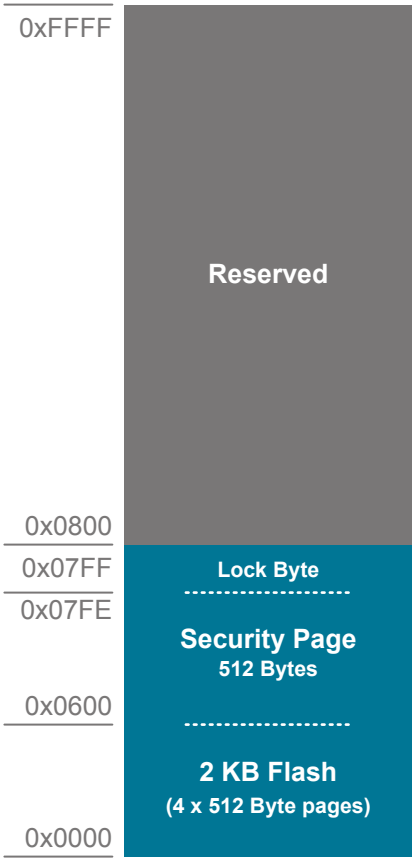



**Figure 4.3. Flash Memory Map — 2 KB Devices**

### 4.2 Features

The flash memory has the following features:

- Up to 8 KB organized in 512-byte sectors.
- In-system programmable from user firmware.
- Security lock to prevent unwanted read/write/erase access.

## 4.3 Functional Description

### 4.3.1 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 in flash user space offers protection of the flash program memory from access (reads, writes, or erases) by unprotected code or the C2 interface. See the specific device memory map for the location of the security byte. The flash security mechanism allows the user to lock "n" flash pages, starting at page 0, where "n" is the 1s complement number represented by the Security Lock Byte.
**Note:** 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).

**Table 4.1. Security Byte Decoding**

| | |
|---|---|
| Security Lock Byte | 111111101b |
| 1s Complement | 00000010b |
| Flash Pages Locked | 3 (First two flash pages + Lock Byte Page) |

The level of flash security depends on the flash access method. The three flash access methods that can be restricted are reads, writes, and erases from the C2 debug interface, user firmware executing on unlocked pages, and user firmware executing on locked pages.

**Table 4.2. Flash Security Summary—Firmware Permissions**

| | Permissions according to the area firmware is executing from: | | | |
|---|---|---|---|---|
| **Target Area for Read / Write / Erase** | **Unlocked User Page** | **Locked User Page** | **Unlocked Data Page** | **Locked Data Page** |
| Any Unlocked Page | [R] [W] [E] | [R] [W] [E] | [R] [W] [E] | [R] [W] [E] |
| Locked Page (except security page) | reset | [R] [W] [E] | reset | [R] [W] [E] |
| Locked Security Page | reset | [R] [W] | reset | [R] [W] |
| Reserved Area | reset | reset | reset | reset |
| [R] = Read permitted<br>[W] = Write permitted<br>[E] = Erase permitted<br>reset = Flash error reset triggered<br>n/a = Not applicable | | | | |

**Table 4.3. Flash Security Summary—C2 Permissions**

| Target Area for Read / Write / Erase | Permissions from C2 interface |
|---|---|
| Any Unlocked Page | [R] [W] [E] |
| Any Locked Page | Device Erase Only |
| Reserved Area | None |

| Target Area for Read / Write / Erase | Permissions from C2 interface |
|---|---|
| [R] = Read permitted<br>[W] = Write permitted<br>[E] = Erase permitted<br>Device Erase Only = No read, write, or individual page erase is allowed. Must erase entire flash space.<br>None = Read, write and erase are not permitted | |

### 4.3.2 Programming the Flash Memory

Writes to flash memory clear bits from logic 1 to logic 0 and can be performed on single byte locations. Flash erasures set bits back to logic 1 and occur only on full pages. 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.

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. Firmware may also be loaded into the device to implement code-loader functions or allow non-volatile data storage. To ensure the integrity of flash contents, it is strongly recommended that the on-chip supply monitor be enabled in any system that includes code that writes and/or erases flash memory from software.

#### 4.3.2.1 Flash Lock and Key Functions

Flash writes and erases by user software are protected with a lock and key function. The FLKEY register must be written with the correct key codes, in sequence, before flash operations may be performed. The key codes are 0xA5 and 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 another flash write or erase operation can be performed.

#### 4.3.2.2 Flash Page Erase Procedure

The flash memory is erased one page at a time by firmware using the MOVX write instruction with the address targeted to any byte within the page. Before erasing a page of flash memory, flash write and erase operations must be enabled by setting the PSWE and PSEE bits in the PSCTL register to logic 1 (this directs the MOVX writes to target flash memory and enables page erasure) and writing the flash key codes in sequence to the FLKEY register. The PSWE and PSEE bits remain set until cleared by firmware.

Erase operation applies to an entire page (setting all bytes in the page to 0xFF). To erase an entire page, perform the following steps:

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

#### 4.3.2.3 Flash Byte Write Procedure

The flash memory is written by firmware using the MOVX write instruction with the address and data byte to be programmed provided as normal operands in DPTR and A. Before writing to flash memory using MOVX, flash write operations must be enabled by setting the PSWE bit in the PSCTL register to logic 1 (this directs the MOVX writes to target flash memory) and writing the flash key codes in sequence to the FLKEY register. The PSWE bit remains set until cleared by firmware. A write to flash memory can clear bits to logic 0 but cannot set them. A byte location to be programmed should be erased (already set to 0xFF) before a new value is written.

To write a byte of flash, perform the following steps:

1. Disable interrupts (recommended).
2. Write the first key code to FLKEY: 0xA5.
3. Write the second key code to FLKEY: 0xF1.
4. Set the PSWE bit (register PSCTL).
5. Clear the PSEE bit (register PSCTL).
6. Using the MOVX instruction, write a single data byte to the desired location within the desired page.

7. Clear the PSWE bit.

### 4.3.3 Flash Write and Erase Precautions

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 supply voltage, 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, hardware restricts flash writes and erasures when the supply monitor is not active and selected as a reset source. As the monitor is enabled and selected as a reset source by default, it is recommended that systems writing or erasing flash simply maintain the default state.

The following sections provide general guidelines for any system which contains routines which write or erase flash from code. Additional flash recommendations and example code can be found in *AN201: Writing to Flash From Firmware*, available from the Silicon Laboratories website.

#### Voltage Supply Maintenance and the Supply Monitor

- If the system power supply is subject to voltage or current "spikes," add sufficient transient protection devices to the power supply to ensure that the supply voltages listed in the Absolute Maximum Ratings table are not exceeded.
- Make certain that the minimum supply rise time specification is met. If the system cannot meet this rise time specification, then add an external supply brownout circuit to the RSTb pin of the device that holds the device in reset until the voltage supply reaches the lower limit, and re-asserts RSTb if the supply drops below the low supply limit.
- Do not disable the supply monitor. If the supply monitor must be disabled in the system, firmware should be added to the startup routine to enable the on-chip supply monitor and enable the supply monitor as a reset source as early in code as possible. This should be the first set of instructions executed after the reset vector. For C-based systems, this may involve modifying the startup code added by the C compiler. See your compiler documentation for more details. Make certain that there are no delays in software between enabling the supply monitor and enabling the supply monitor as a reset source.
  **Note:** The supply monitor must be enabled and enabled as a reset source when writing or erasing flash memory. A flash error reset will occur if either condition is not met.

- As an added precaution if the supply monitor is ever disabled, explicitly enable the supply monitor and enable the supply monitor as a reset source inside the functions that write and erase flash memory. The supply monitor enable instructions should be placed just after the instruction to set PSWE to a 1, but before the flash write or erase operation instruction.
- Make certain that all writes to the RSTSRC (Reset Sources) register use direct assignment operators and explicitly do not use the bit-wise operators (such as AND or OR). For example, "RSTSRC = 0x02" is correct. "RSTSRC |= 0x02" is incorrect.
- Make certain that all writes to the RSTSRC register explicitly set the PORSF bit to a 1. Areas to check are initialization code which enables other reset sources, such as the Missing Clock Detector or Comparator, for example, and instructions which force a Software Reset. A global search on "RSTSRC" can quickly verify this.

#### PSWE Maintenance

- Reduce the number of places in code where the PSWE bit (in register PSCTL) is set to a 1. There should be exactly one routine in code that sets PSWE to a 1 to write flash bytes and one routine in code that sets PSWE and PSEE both to a 1 to erase flash pages.
- Minimize the number of variable accesses while PSWE is set to a 1. Handle pointer address updates and loop variable maintenance outside the "PSWE = 1;... PSWE = 0;" area.
- Disable interrupts prior to setting PSWE to a 1 and leave them disabled until after PSWE has been reset to 0. Any interrupts posted during the flash write or erase operation will be serviced in priority order after the flash operation has been completed and interrupts have been re-enabled by software.
- Make certain that the flash write and erase pointer variables are not located in XRAM. See your compiler documentation for instructions regarding how to explicitly locate variables in different memory areas.
- Add address bounds checking to the routines that write or erase flash memory to ensure that a routine called with an illegal address does not result in modification of the flash.

#### System Clock

- If operating from an external crystal-based source, be advised that crystal performance is susceptible to electrical interference and is sensitive to layout and to changes in temperature. If the system is operating in an electrically noisy environment, use the internal oscillator or use an external CMOS clock.
- If operating from the external oscillator, switch to the internal oscillator during flash write or erase operations. The external oscillator can continue to run, and the CPU can switch back to the external oscillator after the flash operation has completed.

### 4.3.4 Minimizing Flash Read Current

The flash memory is responsible for a substantial portion of the total digital supply current when the device is executing code. Below are suggestions to minimize flash read current.

1. Use low power modes while waiting for an interrupt, rather than polling the interrupt flag.
2. Disable the one-shot timer.
3. Reduce the number of toggling address lines for short code loops.

#### Using Low Power Modes

To reduce flash read current, use idle, suspend, or sleep modes while waiting for an interrupt, rather than polling the interrupt flag. Idle mode is particularly well-suited for use in implementing short pauses, since the wake-up time is no more than three system clock cycles. See the Power Management chapter for details on the various low-power operating modes.

#### Disabling the One-Shot Timer

The flash has a one-shot timer that saves power when operating at system clock frequencies of 14 MHz or less. The one-shot timer generates a minimum-duration enable signal for the flash sense amps on each clock cycle in which the flash memory is accessed. This allows the flash to remain in a low power state for the remainder of the long clock cycle.

At clock frequencies above 14 MHz, the system clock cycle becomes short enough that the one-shot timer no longer provides a power benefit. Disabling the one-shot timer at higher frequencies reduces power consumption. The one-shot is enabled by default, and it can be disabled (bypassed) by setting the BYPASS bit in the FLSCL register. To reenable the one-shot, clear the BYPASS bit to logic 0.

#### Reduce Toggling Lines in Loops

Flash read current depends on the number of address lines that toggle between sequential flash read operations. In most cases, the difference in power is relatively small (on the order of 5%).

The flash memory is organized in rows of 64 bytes. A substantial current increase can be detected when the read address jumps from one row in the flash memory to another. Consider a 3-cycle loop (e.g., SJMP $, or while(1);) which straddles a flash row boundary. The flash address jumps from one row to another on two of every three clock cycles. This can result in a current increase of up 30% when compared to the same 3-cycle loop contained entirely within a single row.

To minimize the power consumption of small loops, it is best to locate them within a single row, if possible. To check if a loop is contained within a flash row, divide the starting address of the first instruction in the loop by 64. If the remainder (result of modulo operation) plus the length of the loop is less than 63, then the loop fits inside a single flash row. Otherwise, the loop will be straddling two adjacent flash rows. If a loop executes in 20 or more clock cycles, then the transitions from one row to another will occur on relatively few clock cycles, and any resulting increase in operating current will be negligible.

## 4.4 Flash Control Registers

### 4.4.1 PSCTL: Program Store Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | | | PSEE | PSWE |
| Access | R | | | | | | RW | RW |
| Reset | 0x00 | | | | | | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0x8F | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:2* | *Reserved* | *Must write reset value.* | | |
| 1 | PSEE | 0 | RW | **Program Store Erase Enable.**<br>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 | PSWE | 0 | RW | **Program Store Write Enable.**<br>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. |

Bit 1 — PSEE

| Value | Name | Description |
|---|---|---|
| 0 | ERASE_DISABLED | Flash program memory erasure disabled. |
| 1 | ERASE_ENABLED | Flash program memory erasure enabled. |

Bit 0 — PSWE

| Value | Name | Description |
|---|---|---|
| 0 | WRITE_DISABLED | Writes to flash program memory disabled. |
| 1 | WRITE_ENABLED | Writes to flash program memory enabled; the MOVX write instruction targets flash memory. |

### 4.4.2 FLKEY: Flash Lock and Key

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FLKEY | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xB7 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | FLKEY | 0x00 | RW | **Flash Lock and Key Register.**<br>Write:<br>This register provides a lock and key function for flash erasures and writes. Flash writes and erases are enabled by writing 0xA5 followed by 0xF1 to the FLKEY register. Flash writes and erases are automatically disabled after the next write or erase is complete. If any writes to FLKEY are performed incorrectly, or if a flash write or erase operation is attempted while these operations are disabled, the flash will be permanently locked from writes or erasures until the next device reset. If an application never writes to flash, it can intentionally lock the flash by writing a non-0xA5 value to FLKEY from firmware.<br>Read:<br>When read, bits 1-0 indicate the current flash lock state.<br>00: Flash is write/erase locked.<br>01: The first key code has been written (0xA5).<br>10: Flash is unlocked (writes/erases allowed).<br>11: Flash writes/erases are disabled until the next reset. |

### 4.4.3 FLSCL: Flash Scale

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | BYPASS | Reserved | | | | | |
| Access | R | RW | R | | | | | |
| Reset | 0 | 0 | 0x00 | | | | | |
| SFR Page = 0x0; SFR Address: 0xB6 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7* | *Reserved* | *Must write reset value.* | | |
| 6 | BYPASS | 0 | RW | **Flash Read Timing One-Shot Bypass.** |
| | Value | Name | | Description |
| | 0 | ONE_SHOT | | The one-shot determines the flash read time. This setting should be used for operating frequencies less than 14 MHz. |
| | 1 | SYSCLK | | The system clock determines the flash read time. This setting should be used for frequencies greater than 14 MHz. |
| *5:0* | *Reserved* | *Must write reset value.* | | |

Operations which clear the BYPASS bit do not need to be immediately followed by a benign 3-byte instruction. For code compatibility with EFM8SB2 devices, a benign 3-byte instruction whose third byte is a don't care should follow the clear operation. See the EFM8SB2 Reference Manual for more details.

# 5. Device Identification

## 5.1 Device Identification

The SFR map includes registers that may be used to identify the device family (DEVICEID), derivative (DERIVID), and revision (REVID). These SFRs can be read by firmware at runtime to determine the capabilities of the MCU that is executing code. This allows the same firmware image to run on MCUs with different memory sizes and peripherals, and dynamically change functionality to suit the capabilities of that MCU.

## 5.2 Unique Identifier

A 32-bit unique identifier (UID) is pre-loaded upon device reset into the RAM area on all devices. The UID resides in the last four bytes of XRAM on devices which include XRAM, or the last four bytes of the RAM space for devices without XRAM. For devices with the UID in RAM, the UID can be read by firmware using indirect data accesses. For devices with the UID in XRAM, the UID can be read by firmware using MOVX instructions. The UID can also be read through the debug port for all devices.

As the UID appears in RAM, firmware can overwrite the UID during normal operation. The bytes in memory will be automatically reinitialized with the UID value after any device reset. Firmware using this area of memory should always initialize the memory to a known value, as any previous data stored at these locations will be overwritten and not retained through a reset.

**Table 5.1. UID Location in Memory**

| Total Device RAM | Memory Segment | Addresses |
|---|---|---|
| 512 Bytes | XRAM | (MSB) 0x00FF, 0x00FE, 0x00FD, 0x00FC (LSB) |
| 256 Bytes | RAM | (MSB) 0xFF, 0xFE, 0xFD, 0xFC (LSB) |

## 5.3 Device Identification Registers

### 5.3.1 DERIVID: Device Identification

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DERIVID | | | | | | | |
| Access | R | | | | | | | |
| Reset | Varies | | | | | | | |
| SFR Page = 0xF; SFR Address: 0xE3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | DERIVID | Varies | R | **Derivative ID.**<br>This read-only register returns the 8-bit derivative ID, which can be used by firmware to identify which device in the product family the code is executing on. The '{R}' tag in the part numbers indicates the device revision letter in the ordering code. The revision letter may be determined by decoding the REVID register. |

| Value | Name | Description |
|---|---|---|
| 0x01 | EFM8SB10F8G_QFN24 | EFM8SB10F8G-{R}-QFN24. |
| 0x02 | EFM8SB10F8G_QSOP24 | EFM8SB10F8G-{R}-QSOP24. |
| 0x03 | EFM8SB10F8G_QFN20 | EFM8SB10F8G-{R}-QFN20. |
| 0x06 | EFM8SB10F4G_QFN20 | EFM8SB10F4G-{R}-QFN20. |
| 0x09 | EFM8SB10F2G_QFN20 | EFM8SB10F2G-{R}-QFN20. |

### 5.3.2 REVID: Revision Identifcation

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | REVID | | | | | | | |
| Access | R | | | | | | | |
| Reset | Varies | | | | | | | |
| SFR Page = 0xF; SFR Address: 0xE2 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | REVID | Varies | R | **Revision ID.** |

This read-only register returns the 8-bit revision ID.

| Value | Name | Description |
|---|---|---|
| 0x00 | REV_A | Revision A |

# 6. Interrupts

## 6.1 Introduction

The MCU core includes an extended interrupt system supporting multiple interrupt sources and priority levels. The allocation of interrupt sources between on-chip peripherals and external input pins varies according to the specific version of the device.

Interrupt sources may have one or more associated interrupt-pending flag(s) located in an SFR local to the associated peripheral. When a peripheral or external source meets a valid interrupt condition, the associated interrupt-pending flag is set to logic 1.

If interrupts are enabled for the source, an interrupt request is generated when the interrupt-pending flag is set. As soon as execution of the current instruction is complete, the CPU generates an LCALL to a predetermined address to begin execution of an interrupt service routine (ISR). Each ISR must end with an RETI instruction, which returns program execution to the next instruction that would have been executed if the interrupt request had not occurred. If interrupts are not enabled, the interrupt-pending flag is ignored by the hardware and program execution continues as normal. The interrupt-pending flag is set to logic 1 regardless of whether the interrupt is enabled.

Each interrupt source can be individually enabled or disabled through the use of an associated interrupt enable bit in the IE and EIEn registers. However, interrupts must first be globally enabled by setting the EA bit to logic 1 before the individual interrupt enables are recognized. Setting the EA bit to logic 0 disables all interrupt sources regardless of the individual interrupt-enable settings.

Some interrupt-pending flags are automatically cleared by the hardware when the CPU vectors to the ISR or by other hardware conditions. However, most are not cleared by the hardware and must be cleared by software before returning from the ISR. If an interrupt-pending flag remains set after the CPU completes the return-from-interrupt (RETI) instruction, a new interrupt request will be generated immediately and the CPU will re-enter the ISR after the completion of the next instruction.

## 6.2 Interrupt Sources and Vectors

The CIP51 core supports interrupt sources for each peripheral on the device. Software can simulate an interrupt for many peripherals by setting any interrupt-pending flag to logic 1. If interrupts are enabled for the flag, an interrupt request will be generated and the CPU will vector to the ISR address associated with the interrupt-pending flag. 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).

### 6.2.1 Interrupt Priorities

Each interrupt source can be individually programmed to one of two priority levels: low or high. A low priority interrupt service routine can be preempted by a high priority interrupt. A high priority interrupt cannot be preempted. Each interrupt has an associated interrupt priority bit in the IP and EIPn registers, which are used to configure its priority level. Low priority is the default. If two interrupts are recognized simultaneously, the interrupt with the higher priority is serviced first. If both interrupts have the same priority level, a fixed order is used to arbitrate, based on the interrupt source's location in the interrupt vector table. Interrupts with a lower number in the vector table have priority.

### 6.2.2 Interrupt Latency

Interrupt response time depends on the state of the CPU when the interrupt occurs. Pending interrupts are sampled and priority decoded on every system clock cycle. Therefore, the fastest possible response time is 5 system clock cycles: 1 clock cycle to detect the interrupt and 4 clock cycles to complete the LCALL to the ISR. If an interrupt is pending when a RETI is executed, a single instruction is executed before an LCALL is made to service the pending interrupt. Therefore, the maximum response time for an interrupt (when no other interrupt is currently being serviced or the new interrupt is of greater priority) occurs when the CPU is performing an RETI instruction followed by a DIV as the next instruction. In this case, the response time is 18 system clock cycles: 1 clock cycle to detect the interrupt, 5 clock cycles to execute the RETI, 8 clock cycles to complete the DIV instruction and 4 clock cycles to execute the LCALL to the ISR. If the CPU is executing an ISR for an interrupt with equal or higher priority, the new interrupt will not be serviced until the current ISR completes, including the RETI and following instruction. If more than one interrupt is pending when the CPU exits an ISR, the CPU will service the next highest priority interrupt that is pending.

### 6.2.3 Interrupt Summary

**Table 6.1. Interrupt Priority Table**

| Interrupt Source | Vector | Priority | Primary Enable | Auxiliary Enable(s) | Pending Flag(s) |
|---|---|---|---|---|---|
| Reset | 0x0000 | Top | - | - | - |
| External Interrupt 0 | 0x0003 | 0 | IE_EX0 | - | TCON_IE0 |
| Timer 0 Overflow | 0x000B | 1 | IE_ET0 | - | TCON_TF0 |
| External Interrupt 1 | 0x0013 | 2 | IE_EX1 | - | TCON_IE1 |
| Timer 1 Overflow | 0x001B | 3 | IE_ET1 | - | TCON_TF1 |
| UART 0 | 0x0023 | 4 | IE_ES0 | - | SCON0_RI<br>SCON0_TI |
| Timer 2 Overflow | 0x002B | 5 | IE_ET2 | TMR2CN0_TF2CEN<br>TMR2CN0_TF2LEN | TMR2CN0_TF2H<br>TMR2CN0_TF2L |
| SPI0 | 0x0033 | 6 | IE_ESPI0 | - | SPI0CN0_MODF<br>SPI0CN0_RXOVRN<br>SPI0CN0_SPIF<br>SPI0CN0_WCOL |
| SMBus 0 | 0x003B | 7 | EIE1_ESMB0 | - | SMB0CN0_SI |
| RTC0 Alarm | 0x0043 | 8 | EIE1_ERTC0A | - | RTC0CN0_ALRM |
| ADC0 Window Compare | 0x004B | 9 | EIE1_EWADC0 | - | ADC0CN0_ADWINT |
| ADC0 End of Conversion | 0x0053 | 10 | EIE1_EADC0 | - | ADC0CN0_ADINT |
| PCA0 | 0x005B | 11 | EIE1_EPCA0 | PCA0CPM0_ECCF<br>PCA0CPM1_ECCF<br>PCA0CPM2_ECCF<br>PCA0PWM_ECOV | PCA0CN0_CCF0<br>PCA0CN0_CCF1<br>PCA0CN0_CCF2<br>PCA0CN0_CF |
| Comparator 0 | 0x0063 | 12 | EIE1_ECP0 | CMP0MD_CPRIE<br>CMP0MD_CPFIE | CMP0CN0_CPFIF<br>CMP0CN0_CPRIF |
| Reserved | 0x006B | 13 | - | - | - |
| Timer 3 Overflow | 0x0073 | 14 | EIE1_ET3 | TMR3CN0_TF3CEN<br>TMR3CN0_TF3LEN | TMR3CN0_TF3H<br>TMR3CN0_TF3L |
| Supply Monitor Early Warning | 0x007B | 15 | EIE2_EWARN | VDM0CN_VDDOKIE | VDM0CN_VDDOK |
| Port Match | 0x0083 | 16 | EIE2_EMAT | - | - |
| RTC0 Oscillator Fail | 0x008B | 17 | EIE2_ERTC0F | - | RTC0CN0_OSCFAIL |
| Reserved | 0x0093 | 18 | - | - | - |
| CS0 End of Conversion | 0x009B | 19 | EIE2_ECSCPT | - | CS0CN0_CSINT |
| CS0 Digital Comparator | 0x00A3 | 20 | EIE2_ECSDC | - | CS0CN0_CSCMPF |
| CS0 End of Scan | 0x00AB | 21 | EIE2_ECSEOS | - | CS0CN0_CSEOS |

## 6.3 Interrupt Control Registers

### 6.3.1 IE: Interrupt Enable

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | EA | ESPI0 | ET2 | ES0 | ET1 | EX1 | ET0 | EX0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0xA8 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | EA | 0 | RW | **All Interrupts Enable.** |
| | Globally enables/disables all interrupts and overrides individual interrupt mask settings. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable all interrupt sources. |
| | 1 | ENABLED | | Enable each interrupt according to its individual mask setting. |
| 6 | ESPI0 | 0 | RW | **SPI0 Interrupt Enable.** |
| | This bit sets the masking of the SPI0 interrupts. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable all SPI0 interrupts. |
| | 1 | ENABLED | | Enable interrupt requests generated by SPI0. |
| 5 | ET2 | 0 | RW | **Timer 2 Interrupt Enable.** |
| | This bit sets the masking of the Timer 2 interrupt. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable Timer 2 interrupt. |
| | 1 | ENABLED | | Enable interrupt requests generated by the TF2L or TF2H flags. |
| 4 | ES0 | 0 | RW | **UART0 Interrupt Enable.** |
| | This bit sets the masking of the UART0 interrupt. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable UART0 interrupt. |
| | 1 | ENABLED | | Enable UART0 interrupt. |
| 3 | ET1 | 0 | RW | **Timer 1 Interrupt Enable.** |
| | This bit sets the masking of the Timer 1 interrupt. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable all Timer 1 interrupt. |
| | 1 | ENABLED | | Enable interrupt requests generated by the TF1 flag. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2 | EX1 | 0 | RW | **External Interrupt 1 Enable.** |

This bit sets the masking of External Interrupt 1.

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable external interrupt 1. |
| 1 | ENABLED | Enable interrupt requests generated by the INT1 input. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 1 | ET0 | 0 | RW | **Timer 0 Interrupt Enable.** |

This bit sets the masking of the Timer 0 interrupt.

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable all Timer 0 interrupt. |
| 1 | ENABLED | Enable interrupt requests generated by the TF0 flag. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 0 | EX0 | 0 | RW | **External Interrupt 0 Enable.** |

This bit sets the masking of External Interrupt 0.

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable external interrupt 0. |
| 1 | ENABLED | Enable interrupt requests generated by the INT0 input. |

### 6.3.2 IP: Interrupt Priority

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | PSPI0 | PT2 | PS0 | PT1 | PX1 | PT0 | PX0 |
| Access | R | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0xB8 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7* | *Reserved* | *Must write reset value.* | | |
| 6 | PSPI0 | 0 | RW | **Serial Peripheral Interface (SPI0) Interrupt Priority Control.**<br>This bit sets the priority of the SPI0 interrupt.<br>0: LOW — SPI0 interrupt set to low priority level.<br>1: HIGH — SPI0 interrupt set to high priority level. |
| 5 | PT2 | 0 | RW | **Timer 2 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 2 interrupt.<br>0: LOW — Timer 2 interrupt set to low priority level.<br>1: HIGH — Timer 2 interrupt set to high priority level. |
| 4 | PS0 | 0 | RW | **UART0 Interrupt Priority Control.**<br>This bit sets the priority of the UART0 interrupt.<br>0: LOW — UART0 interrupt set to low priority level.<br>1: HIGH — UART0 interrupt set to high priority level. |
| 3 | PT1 | 0 | RW | **Timer 1 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 1 interrupt.<br>0: LOW — Timer 1 interrupt set to low priority level.<br>1: HIGH — Timer 1 interrupt set to high priority level. |
| 2 | PX1 | 0 | RW | **External Interrupt 1 Priority Control.**<br>This bit sets the priority of the External Interrupt 1 interrupt.<br>0: LOW — External Interrupt 1 set to low priority level.<br>1: HIGH — External Interrupt 1 set to high priority level. |
| 1 | PT0 | 0 | RW | **Timer 0 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 0 interrupt. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | Value | Name | | Description |
| | 0 | LOW | | Timer 0 interrupt set to low priority level. |
| | 1 | HIGH | | Timer 0 interrupt set to high priority level. |
| 0 | PX0 | 0 | RW | **External Interrupt 0 Priority Control.** |
| | This bit sets the priority of the External Interrupt 0 interrupt. | | | |
| | Value | Name | | Description |
| | 0 | LOW | | External Interrupt 0 set to low priority level. |
| | 1 | HIGH | | External Interrupt 0 set to high priority level. |

### 6.3.3 EIE1: Extended Interrupt Enable 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ET3 | Reserved | ECP0 | EPCA0 | EADC0 | EWADC0 | ERTC0A | ESMB0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0xE6 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | ET3 | 0 | RW | **Timer 3 Interrupt Enable.**<br>This bit sets the masking of the Timer 3 interrupt. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable Timer 3 interrupts. |
| | 1 | ENABLED | | Enable interrupt requests generated by the TF3L or TF3H flags. |
| *6* | *Reserved* | *Must write reset value.* | | |
| 5 | ECP0 | 0 | RW | **Comparator0 (CP0) Interrupt Enable.**<br>This bit sets the masking of the CP0 interrupt. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable CP0 interrupts. |
| | 1 | ENABLED | | Enable interrupt requests generated by the comparator 0 CPRIF or CPFIF flags. |
| 4 | EPCA0 | 0 | RW | **Programmable Counter Array (PCA0) Interrupt Enable.**<br>This bit sets the masking of the PCA0 interrupts. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable all PCA0 interrupts. |
| | 1 | ENABLED | | Enable interrupt requests generated by PCA0. |
| 3 | EADC0 | 0 | RW | **ADC0 Conversion Complete Interrupt Enable.**<br>This bit sets the masking of the ADC0 Conversion Complete interrupt. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable ADC0 Conversion Complete interrupt. |
| | 1 | ENABLED | | Enable interrupt requests generated by the ADINT flag. |
| 2 | EWADC0 | 0 | RW | **ADC0 Window Comparison Interrupt Enable.**<br>This bit sets the masking of ADC0 Window Comparison interrupt. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable ADC0 Window Comparison interrupt. |
| | 1 | ENABLED | | Enable interrupt requests generated by ADC0 Window Compare flag (ADWINT). |
| 1 | ERTC0A | 0 | RW | **RTC Alarm Interrupt Enable.**<br>This bit sets the masking of the RTC Alarm interrupt. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable RTC Alarm interrupts. |
| | 1 | ENABLED | | Enable interrupt requests generated by a RTC Alarm. |
| 0 | ESMB0 | 0 | RW | **SMBus (SMB0) Interrupt Enable.** |
| | This bit sets the masking of the SMB0 interrupt. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable all SMB0 interrupts. |
| | 1 | ENABLED | | Enable interrupt requests generated by SMB0. |

### 6.3.4 EIP1: Extended Interrupt Priority 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PT3 | Reserved | PCP0 | PPCA0 | PADC0 | PWADC0 | PRTC0A | PSMB0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0xF6 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | PT3 | 0 | RW | **Timer 3 Interrupt Priority Control.**<br>This bit sets the priority of the Timer 3 interrupt. |
| | Value | Name | | Description |
| | 0 | LOW | | Timer 3 interrupts set to low priority level. |
| | 1 | HIGH | | Timer 3 interrupts set to high priority level. |
| *6* | *Reserved* | *Must write reset value.* | | |
| 5 | PCP0 | 0 | RW | **Comparator0 (CP0) Interrupt Priority Control.**<br>This bit sets the priority of the CP0 interrupt. |
| | Value | Name | | Description |
| | 0 | LOW | | CP0 interrupt set to low priority level. |
| | 1 | HIGH | | CP0 interrupt set to high priority level. |
| 4 | PPCA0 | 0 | RW | **Programmable Counter Array (PCA0) Interrupt Priority Control.**<br>This bit sets the priority of the PCA0 interrupt. |
| | Value | Name | | Description |
| | 0 | LOW | | PCA0 interrupt set to low priority level. |
| | 1 | HIGH | | PCA0 interrupt set to high priority level. |
| 3 | PADC0 | 0 | RW | **ADC0 Conversion Complete Interrupt Priority Control.**<br>This bit sets the priority of the ADC0 Conversion Complete interrupt. |
| | Value | Name | | Description |
| | 0 | LOW | | ADC0 Conversion Complete interrupt set to low priority level. |
| | 1 | HIGH | | ADC0 Conversion Complete interrupt set to high priority level. |
| 2 | PWADC0 | 0 | RW | **ADC0 Window Comparator Interrupt Priority Control.**<br>This bit sets the priority of the ADC0 Window interrupt. |
| | Value | Name | | Description |
| | 0 | LOW | | ADC0 Window interrupt set to low priority level. |
| | 1 | HIGH | | ADC0 Window interrupt set to high priority level. |
| 1 | PRTC0A | 0 | RW | **RTC Alarm Interrupt Priority Control.**<br>This bit sets the priority of the RTC Alarm interrupt. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | Value | Name | | Description |
| | 0 | LOW | | RTC Alarm interrupt set to low priority level. |
| | 1 | HIGH | | RTC Alarm interrupt set to high priority level. |
| 0 | PSMB0 | 0 | RW | **SMBus (SMB0) Interrupt Priority Control.** |
| | This bit sets the priority of the SMB0 interrupt. | | | |
| | Value | Name | | Description |
| | 0 | LOW | | SMB0 interrupt set to low priority level. |
| | 1 | HIGH | | SMB0 interrupt set to high priority level. |

### 6.3.5 EIE2: Extended Interrupt Enable 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | ECSEOS | ECSDC | ECSCPT | Reserved | ERTC0F | EMAT | EWARN |
| Access | RW | RW | RW | RW | R | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0xE7 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7* | *Reserved* | *Must write reset value.* | | |
| 6 | ECSEOS | 0 | RW | **Capacitive Sense End of Scan Interrupt Enable.**<br>This bit sets the masking of the Capacitive Sense End of Scan interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable Capacitive Sense End of Scan interrupt. |
| 1 | ENABLED | Enable interrupt requests generated by the Capacitive Sense End of Scan. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5 | ECSDC | 0 | RW | **Capacitive Sense Digital Comparator Interrupt Enable.**<br>This bit sets the masking of the Capacitive Sense Digital Comparator interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable Capacitive Sense Digital Comparator interrupt. |
| 1 | ENABLED | Enable interrupt requests generated by the Capacitive Sense Digital Comparator. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 4 | ECSCPT | 0 | RW | **Capacitive Sense Conversion Complete Interrupt Enable.**<br>This bit sets the masking of the Capacitive Sense Conversion Complete interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable Capacitive Sense Conversion Complete interrupt. |
| 1 | ENABLED | Enable interrupt requests generated by CS0INT. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *3* | *Reserved* | *Must write reset value.* | | |
| 2 | ERTC0F | 0 | RW | **RTC Oscillator Fail Interrupt Enable.**<br>This bit sets the masking of the RTC Oscillator Fail interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable RTC Oscillator Fail interrupts. |
| 1 | ENABLED | Enable interrupt requests generated by the RTC Oscillator Fail event. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 1 | EMAT | 0 | RW | **Port Match Interrupts Enable.**<br>This bit sets the masking of the Port Match Event interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable all Port Match interrupts. |
| 1 | ENABLED | Enable interrupt requests generated by a Port Match. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 0 | EWARN | 0 | RW | **Supply Monitor Early Warning Interrupt Enable.**<br>This bit sets the masking of the Supply Monitor Early Warning interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable the Supply Monitor Early Warning interrupt. |
| 1 | ENABLED | Enable interrupt requests generated by the Supply Monitor. |

### 6.3.6 EIP2: Extended Interrupt Priority 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | PCSEOS | PCSDC | PCSCPT | Reserved | PRTC0F | PMAT | PWARN |
| Access | R | RW | RW | RW | R | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0xF7 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7* | *Reserved* | *Must write reset value.* | | |
| 6 | PCSEOS | 0 | RW | **Capacitive Sense End of Scan Interrupt Priority Control.**<br>This bit sets the priority of the Capacitive Sense End of Scan interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | LOW | Capacitive Sense End of Scan interrupt set to low priority level. |
| 1 | HIGH | Capacitive Sense End of Scan interrupt set to high priority level. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5 | PCSDC | 0 | RW | **Capacitive Sense Digital Comparator Interrupt Priority Control.**<br>This bit sets the priority of the Capacitive Sense Digital Comparator interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | LOW | Capacitive Sense Digital Comparator interrupt set to low priority level. |
| 1 | HIGH | Capacitive Sense Digital Comparator interrupt set to high priority level. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 4 | PCSCPT | 0 | RW | **Capacitive Sense Conversion Complete Interrupt Priority Control.**<br>This bit sets the priority of the Capacitive Sense Conversion Complete interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | LOW | Capacitive Sense Conversion Complete interrupt set to low priority level. |
| 1 | HIGH | Capacitive Sense Conversion Complete interrupt set to high priority level. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *3* | *Reserved* | *Must write reset value.* | | |
| 2 | PRTC0F | 0 | RW | **RTC Oscillator Fail Interrupt Priority Control.**<br>This bit sets the priority of the RTC Oscillator Fail interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | LOW | RTC Oscillator Fail interrupt set to low priority level. |
| 1 | HIGH | RTC Oscillator Fail interrupt set to high priority level. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 1 | PMAT | 0 | RW | **Port Match Interrupt Priority Control.**<br>This bit sets the priority of the Port Match Event interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | LOW | Port Match interrupt set to low priority level. |
| 1 | HIGH | Port Match interrupt set to high priority level. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 0 | PWARN | 0 | RW | **Supply Monitor Early Warning Interrupt Priority Control.** |

This bit sets the priority of the Supply Monitor Early Warning interrupt.

| Value | Name | Description |
|---|---|---|
| 0 | LOW | Supply Monitor Early Warning interrupt set to low priority level. |
| 1 | HIGH | Supply Monitor Early Warning interrupt set to high priority level. |

# 7. Power Management and Internal Regulators

## 7.1 Introduction

All internal circuitry draws power from the VDD supply pin. External I/O pins are powered from the VIO supply voltage (or VDD on devices without a separate VIO connection), while most of the internal circuitry is supplied by an on-chip LDO regulator. Control over the device power can be achieved by enabling/disabling individual peripherals as needed. Each analog peripheral can be disabled when not in use and placed in low power mode. Digital peripherals, such as timers and serial buses, have their clocks gated off and draw little power when they are not in use.

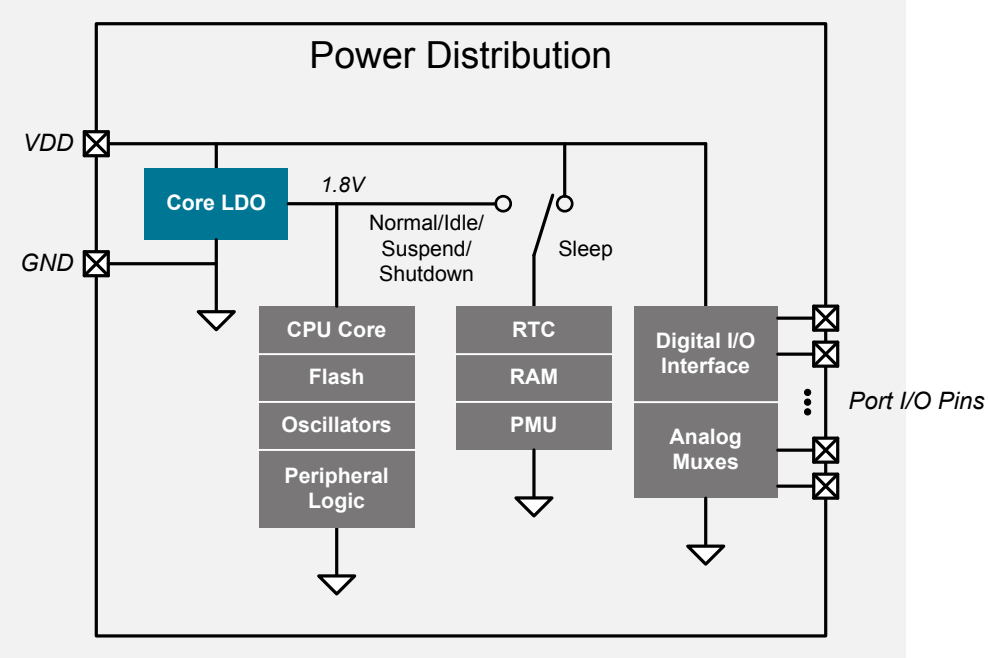



**Figure 7.1. Power System Block Diagram**

**Table 7.1. Power Modes**

| Power Mode | Details | Mode Entry | Wake-Up Sources |
|---|---|---|---|
| Normal | Core and all peripherals clocked and fully operational | — | — |
| Idle | • Core halted<br>• All peripherals clocked and fully operational<br>• Code resumes execution on wake event | Set IDLE bit in PCON0 | Any interrupt |
| Suspend | • Core and digital peripherals halted<br>• Internal oscillators disabled<br>• Code resumes execution on wake event | 1. Switch SYSCLK to HFOSC0 or LPOSC0<br>2. Set SUSPEND bit in PMU0CF | • RTC0 Alarm Event<br>• RTC0 Fail Event<br>• CS0 Interrupt<br>• Port Match Event<br>• Comparator 0 Rising Edge |

| Power Mode | Details | Mode Entry | Wake-Up Sources |
|---|---|---|---|
| Sleep | • Most internal power nets shut down<br>• Select circuits remain powered<br>• Pins retain state<br>• All RAM and SFRs retain state<br>• Code resumes execution on wake event | 1. Disable unused analog peripherals<br>2. Set SLEEP bit in PMU0CF | • RTC0 Alarm Event<br>• RTC0 Fail Event<br>• Port Match Event<br>• Comparator 0 Rising Edge |

### 7.2 Features

- Supports four power modes:
  - Normal mode: Core and all peripherals fully operational.
  - Idle mode: Core halted, peripherals fully operational, core waiting for interrupt to continue.
  - Suspend mode: Similar to Sleep mode, with faster wake-up times, but higher current consumption. Code resumes execution at the next instruction.
  - Sleep mode: Ultra low power mode with flexible wake-up sources. Code resumes execution at the next instruction.

  Note: Legacy 8051 Stop mode is also supported, but Suspend and Sleep offer more functionality with better power consumption.
- Fully internal core LDO supplies power to majority of blocks.

### 7.3 Idle Mode

In idle mode, CPU core execution is halted while any enabled peripherals and clocks remain active. Power consumption in idle mode is dependent upon the system clock frequency and any active peripherals.

Setting the IDLE bit in the PCON0 register causes the hardware to halt the CPU and enter idle mode as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data. All analog and digital peripherals can remain active during idle mode.

Idle mode is terminated when an enabled interrupt is asserted or a reset occurs. The assertion of an enabled interrupt will cause the IDLE bit to be cleared and the CPU to resume operation. The pending interrupt will be serviced and the next instruction to be executed after the return from interrupt (RETI) will be the instruction immediately following the one that set the IDLE bit. If idle 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:** If the instruction following the write of the IDLE bit is a single-byte instruction and an interrupt occurs during the execution phase of the instruction that sets the IDLE bit, the CPU may not wake from idle mode when a future interrupt occurs. Therefore, instructions that set the IDLE bit should be followed by an instruction that has two or more opcode bytes. For example:

```
 // in ‘C’:
PCON0 |= 0x01; // set IDLE bit
PCON0 = PCON0; // ... followed by a 3-cycle dummy instruction

; in assembly:
ORL PCON0, #01h ; set IDLE bit
MOV PCON0, PCON0 ; ... followed by a 3-cycle dummy instruction
```

If enabled, the Watchdog Timer (WDT) will eventually cause an internal watchdog reset and thereby terminate the Idle mode. This feature protects the system from an unintended permanent shutdown in the event of an inadvertent write to the PCON0 register. If this behavior is not desired, the WDT may be disabled by software prior to entering the idle mode if the WDT was initially configured to allow this operation. This provides the opportunity for additional power savings, allowing the system to remain in the idle mode indefinitely, waiting for an external stimulus to wake up the system.

**Note:** To ensure the MCU enters a low power state upon entry into Idle mode, the one-shot circuit should be enabled by clearing the BYPASS bit in the FLSCL register.

### 7.4 Stop Mode

In stop mode, the CPU is halted and peripheral clocks are stopped. Analog peripherals remain in their selected states.

Setting the STOP bit in the PCON0 register causes the controller core to enter stop mode as soon as the instruction that sets the bit completes execution. Before entering stop mode, the system clock must be sourced by HFOSC0. In stop mode, the CPU and internal clocks are stopped. Analog peripherals may remain enabled, but will not be provided a clock. Each analog peripheral may be shut down individually by firmware 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 as a reset source, the missing clock detector will cause an internal reset and thereby terminate the stop mode. If this reset is undesirable in the system, and the CPU is to be placed in stop mode for longer than the missing clock detector timeout, the missing clock detector should be disabled in firmware prior to setting the STOP bit.

**Note:** To ensure the MCU enters a low power state upon entry into Stop mode, the one-shot circuit should be enabled by clearing the BYPASS bit in the FLSCL register.

### 7.5 Suspend Mode

Suspend mode is entered by setting the SUSPEND bit while operating from the internal 24.5 MHz oscillator (HFOSC0) or the internal 20 MHz oscillator (LPOSC0). Upon entry into suspend mode, the hardware halts all of the internal oscillators and goes into a low power state as soon as the instruction that sets the bit completes execution. All internal registers and memory maintain their original data.
**Note:** When entering Suspend mode, the global clock divider must be set to "divide by 1" using the CLKDIV field in the CLKSEL register.

**Note:** The one-shot circuit should be enabled by clearing the BYPASS bit in the FLSCL register to logic 0.

**Note:** Upon wake-up from Suspend, the power management unit requires two system clocks in order to update the PMU0CF wake-up flags. All flags will read back a value of 0 during the first two system clocks following a wake-up from Suspend.

**Note:** The instruction placing the device in Suspend mode should be immediately followed by four NOP instructions. This will ensure the PMU resynchronizes with the core.

Suspend mode is terminated by any enabled wake or reset 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 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.

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

### 7.6 Sleep Mode

Setting the sleep mode select bit in the PMU0CF register turns off the internal 1.8 V core LDO regulator and switches the power supply of all on-chip RAM to the VDD pin. Power to most digital logic on the chip is disconnected; only the power management unit and RTC remain powered. Only the comparators remain functional when the device enters Sleep mode. All other analog peripherals (ADC0, IREF0, External Oscillator, etc.) should be disabled prior to entering Sleep mode.
**Note:** The system clock source must be set to the low power internal oscillator (LPOSC0) with the clock divider set to 1 prior to entering Sleep mode.

**Note:** The instruction placing the device in Sleep mode should be immediately followed by four NOP instructions. This will ensure the PMU resynchronizes with the core.

The precision internal oscillator may potentially lock up after exiting Sleep mode. Systems using Sleep mode and the precision oscillator (HPOSC0) should switch to the low power oscillator prior to entering Sleep:

1. Switch the system clock to the low power oscillator.
2. Turn off the precision oscillator.
3. Enter Sleep.
4. Exit Sleep.
5. Wait 4 NOP instructions.
6. Turn on the precision oscillator.
7. Switch the system clock to the precision oscillator.

GPIO pins configured as digital outputs will retain their output state during sleep mode and maintain the same current drive capability in sleep mode as they have in normal mode. GPIO pins configured as digital inputs can be used during sleep mode as wakeup sources using the port match feature and will maintain the same input level specs in Sleep mode as they have in normal mode.

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

RAM and SFR register contents are preserved in Sleep as long as the voltage on VDD does not fall below $V_{POR}$. The PC counter and all other volatile state information is preserved allowing the device to resume code execution upon waking up from sleep mode.
**Note:** On device reset or upon waking up from Sleep mode, address 0x0000 of external memory (XRAM) may be overwritten by an indeterminate value. The indeterminate value is 0x00 in most situations. A dummy variable should be placed at address 0x0000 in external memory to ensure that the application firmware does not store any data that needs to be retained through reset or Sleep at this memory location.

The following wake-up sources can be configured to wake the device from sleep mode:

- RTC oscillator fail
- RTC alarm
- Port match event
- Comparator 0 rising edge

The comparator requires a supply voltage of at least 1.8 V to operate properly. In addition, any falling edge on RSTb (due to a pin reset or a noise glitch) will cause the device to exit Sleep In order for the MCU to respond to the pin reset event, software must not place the device back into Sleep for a period of 15 µs. The PMU0CF register may be checked to determine if the wake-up was due to a falling edge on the RSTb pin. If the wake-up source is not due to a falling edge on RSTb, there is no time restriction on how soon software may place the device back into sleep mode. A 4.7 kΩ pullup resistor to VDD is recommend for RSTb to prevent noise glitches from waking the device.

#### 7.6.1 Configuring Wakeup Sources

Before placing the device in a low power mode, firmware should enable one or more wakeup sources so that the device does not remain in the low power mode indefinitely. For Idle mode, this includes enabling any interrupt. For Stop mode, this includes enabling any reset source or relying on the RSTb pin to reset the device.

Wake-up sources for Suspend and Sleep modes are configured through the PMU0CF register. Wake-up sources are enabled by writing 1 to the corresponding wake-up source enable bit. Wake-up sources must be re-enabled each time the device is placed in Suspend or Sleep mode in the same write that places the device in the low power mode.

The reset pin is always enabled as a wake-up source. The device will awaken from Sleep mode on the falling edge of RSTb. The device must remain awake for more than 15 µs in order for the reset to take place.

### 7.6.2 Determining the Event that Caused the Last Wakeup

When waking from Idle mode, the CPU will vector to the interrupt which caused it to wake up. When waking from Stop mode, the RSTSRC register may be read to determine the cause of the last reset.

Upon exit from Suspend or Sleep mode, the wake-up flags in the power management registers can be read to determine the event which caused the device to wake up. After waking up, the wake-up flags will continue to be updated if any of the wake-up events occur. Wake-up flags are always updated, even if they are not enabled as wake-up sources.

All wake-up flags enabled as wake-up sources in the power management registers must be cleared before the device can enter Suspend or Sleep mode. After clearing the wake-up flags, each of the enabled wake-up events should be checked in the individual peripherals to ensure that a wake-up event did not occur while the wake-up flags were being cleared.

## 7.7 Power Management Control Registers

### 7.7.1 PCON0: Power Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | GF5 | GF4 | GF3 | GF2 | GF1 | GF0 | STOP | IDLE |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0x87 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | GF5 | 0 | RW | **General Purpose Flag 5.**<br>This flag is a general purpose flag for use under firmware control. |
| 6 | GF4 | 0 | RW | **General Purpose Flag 4.**<br>This flag is a general purpose flag for use under firmware control. |
| 5 | GF3 | 0 | RW | **General Purpose Flag 3.**<br>This flag is a general purpose flag for use under firmware control. |
| 4 | GF2 | 0 | RW | **General Purpose Flag 2.**<br>This flag is a general purpose flag for use under firmware control. |
| 3 | GF1 | 0 | RW | **General Purpose Flag 1.**<br>This flag is a general purpose flag for use under firmware control. |
| 2 | GF0 | 0 | RW | **General Purpose Flag 0.**<br>This flag is a general purpose flag for use under firmware control. |
| 1 | STOP | 0 | RW | **Stop Mode Select.**<br>Setting this bit will place the CIP-51 in Stop mode. This bit will always be read as 0. |
| 0 | IDLE | 0 | RW | **Idle Mode Select.**<br>Setting this bit will place the CIP-51 in Idle mode. This bit will always be read as 0. |
| To ensure the MCU enters a low power state upon entry into Idle or Stop mode, the one-shot circuit should be enabled by clearing the BYPASS bit in the FLSCL register. | | | | |

### 7.7.2 PMU0CF: Power Management Unit Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SLEEP | SUSPEND | CLEAR | RSTWK | RTCFWK | RTCAWK | PMATWK | CPT0WK |
| Access | W | W | W | R | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | Varies | Varies | Varies | Varies | Varies |
| SFR Page = 0x0; SFR Address: 0xB5 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | SLEEP | 0 | W | **Sleep Mode Select.**<br>Writing a 1 to this bit places the device in Sleep mode. |
| 6 | SUSPEND | 0 | W | **Suspend Mode Select.**<br>Writing a 1 to this bit places the device in Suspend mode. |
| 5 | CLEAR | 0 | W | **Wake-up Flag Clear.**<br>Writing a 1 to this bit clears all wake-up flags. |
| 4 | RSTWK | Varies | R | **Reset Pin Wake-up Flag.**<br>This bit is set to 1 if a glitch has been detected on RSTb. |
| 3 | RTCFWK | Varies | RW | **RTC Oscillator Fail Wake-up Source Enable and Flag.**<br>Read: Hardware sets this bit to 1 if the RTC oscillator failed.<br>Write: Write this bit to 1 to enable wake-up on an RTC oscillator failure. |
| 2 | RTCAWK | Varies | RW | **RTC Alarm Wake-up Source Enable and Flag.**<br>Read: Hardware sets this bit to 1 if the RTC Alarm occured.<br>Write: Write this bit to 1 to enable wake-up on an RTC Alarm. |
| 1 | PMATWK | Varies | RW | **Port Match Wake-up Source Enable and Flag.**<br>Read: Hardware sets this bit to 1 if Port Match event occured.<br>Write: Write this bit to 1 to enable wake-up on a Port Match event. |
| 0 | CPT0WK | Varies | RW | **Comparator0 Wake-up Source Enable and Flag.**<br>Read: Hardware sets this bit to 1 if a Comparator 0 rising edge caused the last wake-up.<br>Write: Write this bit to 1 to enable wake-up on a Comparator 0 rising edge. |

Read-modify-write operations (ORL, ANL, etc.) should not be used on this register. Wake-up sources must be re-enabled each time the SLEEP or SUSPEND bits are written to 1.

The Low Power Internal Oscillator cannot be disabled and the MCU cannot be placed in Suspend or Sleep Mode if any wake-up flags are set to 1. Software should clear all wake-up sources after each reset and after each wake-up from Suspend or Sleep Modes.

PMU0 requires two system clocks to update the wake-up source flags after waking from Suspend mode. The wake-up source flags will read 0 during the first two system clocks following the wake from Suspend mode.

### 7.7.3 PMU0FL: Power Management Unit Flag

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | | | | CS0WK |
| Access | R | | | | | | | RW |
| Reset | 0x00 | | | | | | | Varies |
| SFR Page = 0x0; SFR Address: 0xCE | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:1* | *Reserved* | *Must write reset value.* | | |
| 0 | CS0WK | Varies | RW | **CS0 Wake-up Source Enable and Flag.**<br>Read: Hardware sets this bit to 1 if a Capacitive Sensing event occured.<br>Write: Write this bit to 1 to enable wake-up on a Capacitive Sensing event. |

The Low Power Internal Oscillator cannot be disabled and the MCU cannot be placed in Suspend or Sleep mode if any wake-up flags are set to 1. Software should clear all wake-up sources after each reset and after each wake-up from Suspend or Sleep modes.

PMU0 requires two system clocks to update the wake-up source flags after waking from Suspend mode. The wake-up source flags will read 0 during the first two system clocks following the wake from Suspend mode.

### 7.7.4 PMU0MD: Power Management Unit Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RTCOE | WAKEOE | Reserved | | | | | |
| Access | RW | RW | RW | | | | | |
| Reset | 0 | 0 | 0x00 | | | | | |
| SFR Page = 0xF; SFR Address: 0xB5 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | RTCOE | 0 | RW | **Buffered RTC Output Enable.**<br>Enables the buffered RTC oscillator output on P0.2. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable the buffered RTC output. |
| 1 | ENABLED | Enable the buffered RTC output. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | WAKEOE | 0 | RW | **Wakeup Request Output Enable.**<br>Enables the Sleep Mode wake-up request signal on P0.3. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable the wake-up request signal. |
| 1 | ENABLED | Enable the wake-up request signal. |

| | | | | |
|---|---|---|---|---|
| *5:0* | *Reserved* | *Must write reset value.* | | |

### 7.7.5 REG0CN: Voltage Regulator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | OSCBIAS | Reserved | | | |
| Access | R | | | RW | R | | | |
| Reset | 0x0 | | | 0 | 0x0 | | | |
| SFR Page = 0x0; SFR Address: 0xC9 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:5* | *Reserved* | *Must write reset value.* | | |
| 4 | OSCBIAS | 0 | RW | **High Frequency Oscillator Bias.**<br>When set to 1, the bias used by the precision High Frequency Oscillator is enabled. If the precision oscillator is not being used, this bit may be cleared to 0 to reduce supply current in all non-Sleep power modes. This bit must be set to 1 prior to enabling the precision High Frequency Oscillator. |
| *3:0* | *Reserved* | *Must write reset value.* | | |

# 8. Clocking and Oscillators

## 8.1 Introduction

The CPU core and peripheral subsystem may be clocked by both internal and external oscillator resources. By default, the system clock comes up running from the 20 MHz low power oscillator divided by 8.

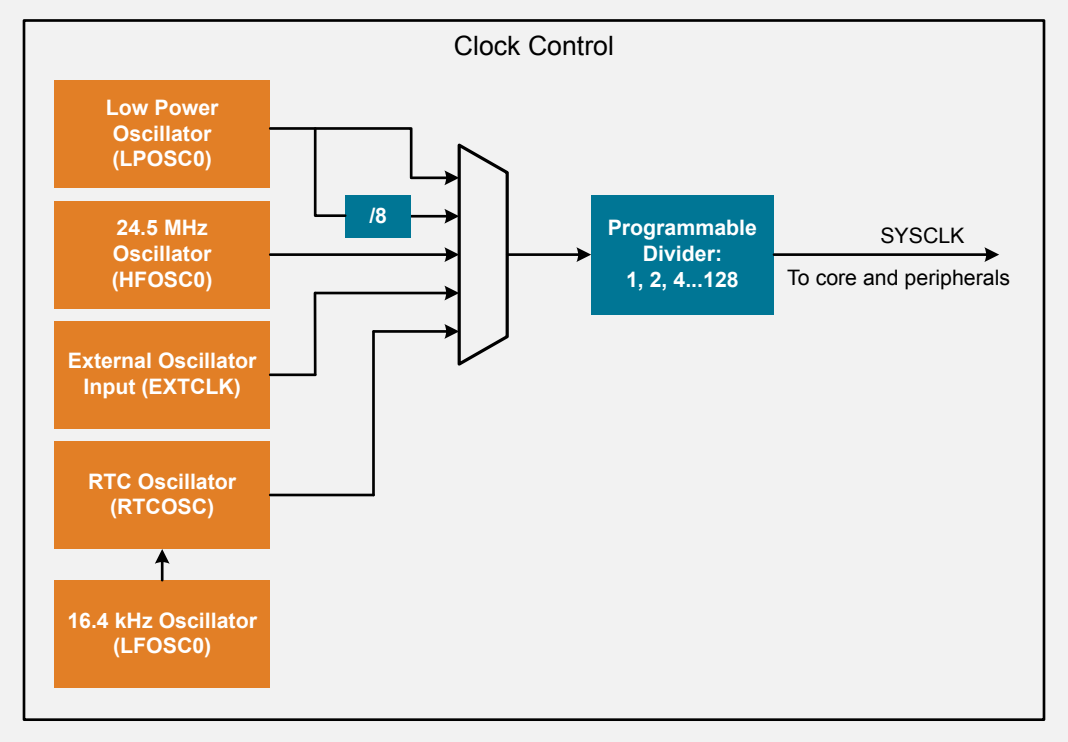



**Figure 8.1. Clock Control Block Diagram**

## 8.2 Features

- Provides clock to core and peripherals.
- 20 MHz low power oscillator (LPOSC0), accurate to ±10% over supply and temperature corners.
- 24.5 MHz internal oscillator (HFOSC0), accurate to ±2% over supply and temperature corners.
- 16.4 kHz low-frequency oscillator (LFOSC0) or external RTC 32 kHz crystal.
- External RC, C, CMOS, and high-frequency crystal clock options (EXTCLK).
- Clock divider with eight settings for flexible clock scaling: Divide the selected clock source by 1, 2, 4, 8, 16, 32, 64, or 128.

## 8.3 Functional Description

### 8.3.1 Clock Selection

The CLKSEL register is used to select the clock source for the system (SYSCLK). The CLKSL field selects which oscillator source is used as the system clock, while CLKDIV controls the programmable divider. When an internal oscillator source is selected as the SYSCLK, the external oscillator may still clock certain peripherals. In these cases, the external oscillator source is synchronized to the SYSCLK source. The system clock may be switched on-the-fly between any of the oscillator sources so long as the selected clock source is enabled and has settled, and CLKDIV may be changed at any time.
**Note:** Some device families do place restrictions on the difference in operating frequency when switching clock sources. Please see the CLKSEL register description for details.

### 8.3.2 LPOSC0 20 MHz Internal Oscillator

LPOSC0 is a programmable internal low power oscillator that is factory-calibrated to 20 MHz. The oscillator is automatically enabled when selected as the system clock and disabled when not in use. This oscillator tolerance is ±10%.

### 8.3.3 HFOSC0 24.5 MHz Internal Oscillator

HFOSC0 is a programmable internal high-frequency oscillator that is factory-calibrated to 24.5 MHz. The oscillator is automatically enabled when it is requested. The oscillator period can be adjusted via the HFO0CAL register to obtain other frequencies.

### 8.3.4 RTC0 Oscillator

The system clock can be derived from the RTC0 oscillator, which can run from either an external 32 kHz crystal or an internal 16.4 kHz ±20% low frequency oscillator (LFOSC0). No loading capacitors are required for the crystal, and it can be connected directly to the XTAL3 and XTAL4 pins.

### 8.3.5 External Crystal

If a crystal or ceramic resonator is used as the external oscillator, the crystal/resonator and a 10 MΩ resistor must be wired across the XTAL1 and XTAL2 pins. Appropriate loading capacitors should be added to XTAL1 and XTAL2, and both pins should be configured for analog I/O with the digital output drivers disabled.

The capacitors shown in the external crystal configuration provide the load capacitance required by the crystal for correct oscillation. These capacitors are “in series” as seen by the crystal and “in parallel” with the stray capacitance of the XTAL1 and XTAL2 pins.
**Note:** The recommended load capacitance depends upon the crystal and the manufacturer. Refer to the crystal data sheet when completing these calculations.

The equation for determining the load capacitance for two capacitors is as follows:

$$C_L = \frac{C_A \times C_B}{C_A + C_B} + C_S$$

**Figure 8.2. External Oscillator Load Capacitance**

Where:

- $C_A$ and $C_B$ are the capacitors connected to the crystal leads.
- $C_S$ is the total stray capacitance of the PCB.
- The stray capacitance for a typical layout where the crystal is as close as possible to the pins is 2-5 pF per pin.

If $C_A$ and $C_B$ are the same (C), then the equation becomes the following:

$$C_L = \frac{C}{2} + C_S$$

**Figure 8.3. External Oscillator Load Capacitance with Equal Capacitors**

For example, a tuning-fork crystal of 25 MHz has a recommended load capacitance of 12.5 pF. With a stray capacitance of 3 pF per pin (6 pF total), the 13 pF capacitors yield an equivalent capacitance of 12.5 pF across the crystal.

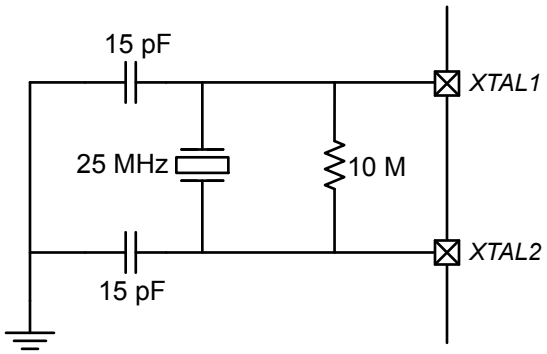



**Figure 8.4. 25 MHz External Crystal Example**

Crystal oscillator circuits are quite sensitive to PCB layout. The crystal should be placed as close as possible to the XTAL pins on the device. The traces should be as short as possible and shielded with ground plane from any other traces which could introduce noise or interference. When using an external crystal, the external oscillator drive circuit must be configured by firmware for Crystal Oscillator Mode or Crystal Oscillator Mode with divide by 2 stage. The divide by 2 stage ensures that the clock derived from the external oscillator has a duty cycle of 50%. The External Oscillator Frequency Control value (XFCN) must also be specified based on the crystal frequency. For example, a 25 MHz crystal requires an XFCN setting of 111b.

**Table 8.1. Recommended XFCN Settings for Crystal Mode**

| XFCN Field Setting | Crystal Frequency | Approximate Bias Current |
|---|---|---|
| 000 | f ≤ 20 kHz | 0.5 µA |
| 001 | 20 kHz < f ≤ 58 kHz | 1.5 µA |
| 010 | 58 kHz < f ≤ 155 kHz | 4.8 µA |
| 011 | 155 kHz < f ≤ 415 kHz | 14 µA |
| 100 | 415 kHz < f ≤ 1.1 MHz | 40 µA |
| 101 | 1.1 MHz < f ≤ 3.1 MHz | 120 µA |
| 110 | 3.1 MHz < f ≤ 8.2 MHz | 550 µA |
| 111 | 8.2 MHz < f ≤ 25 MHz | 2.6 mA |

When the crystal oscillator is first enabled, the external oscillator valid detector allows software to determine when the external system clock has stabilized. Switching to the external oscillator before the crystal oscillator has stabilized can result in unpredictable behavior. The recommended procedure for starting the crystal is as follows:

1. Configure XTAL1 and XTAL2 for analog I/O and disable the digital output drivers.
2. Disable the XTAL1 and XTAL2 digital output drivers by writing 1's to the appropriate bits in the port latch register.
3. Configure and enable the external oscillator.
4. Wait at least 1 ms
5. Poll for XCLKVLD set to 1.
6. Switch the system clock to the external oscillator.

### 8.3.6 External RC and C Modes

#### External RC Example

An RC network connected to the XTAL2 pin can be used as a basic oscillator. XTAL1 is not affected in RC mode.

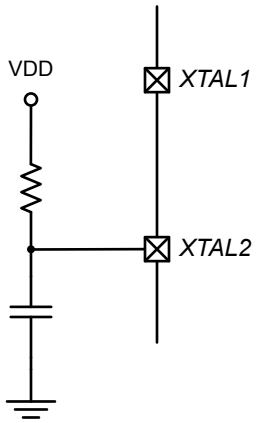



**Figure 8.5. External RC Oscillator Configuration**

The capacitor should be no greater than 100 pF; however, for very small capacitors, the total capacitance may be dominated by parasitic capacitance in the PCB layout. To determine the required XFCN field value, first select the RC network value to produce the desired frequency of oscillation, according to , where f = the frequency of oscillation in MHz, C = the capacitor value in pF, and R = the pull-up resistor value in kΩ.

$$f = \frac{1.23 \times 10^3}{R \times C}$$

**Figure 8.6. RC Mode Oscillator Frequency**

For example, if the frequency desired is 100 kHz, let R = 246 kΩ and C = 50 pF:

$$f = \frac{1.23 \times 10^3}{R \times C} = \frac{1.23 \times 10^3}{246 \times 50} = 100 \text{ kHz}$$

**Figure 8.7. RC Mode Oscillator Example**

Referencing , the recommended XFCN setting for 100 kHz is 010.

When the RC oscillator is first enabled, the external oscillator valid detector allows firmware to determine when oscillation has stabilized. The recommended procedure for starting the RC oscillator is as follows:

1. Configure XTAL2 for analog I/O and disable the digital output drivers.
2. Configure and enable the external oscillator.
3. Poll for XCLKVLD = 1.
4. Switch the system clock to the external oscillator.

### External Capacitor Example

If a capacitor is used as the external oscillator, the circuit should be configured as shown in . The capacitor should be added to XTAL2, and XTAL2 should be configured for analog I/O with the digital output drivers disabled. XTAL1 is not affected in C mode.

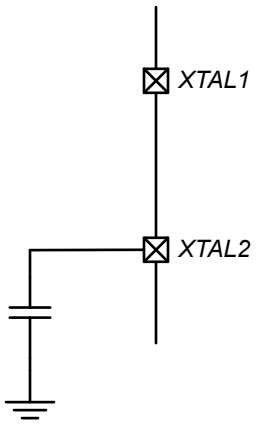



**Figure 8.8. External Capacitor Oscillator Configuration**

The capacitor should be no greater than 100 pF; however, for very small capacitors, the total capacitance may be dominated by parasitic capacitance in the PCB layout. The oscillation frequency and the required XFCN field value determined by the following equation, where f is the frequency in MHz, C is the capacitor value on XTAL2 in pF, and $V_{DD}$ is the power supply voltage in Volts:

$$f = \frac{KF}{C \times V_{DD}}$$

**Figure 8.9. C Mode Oscillator Frequency**

For example, assume VDD = 3.0 V and f = 150 kHz. Since a frequency of roughly 150 kHz is desired, select the K Factor from as KF = 22:

$$f = \frac{KF}{C \times V_{DD}}$$

$$0.150\ \text{MHz} = \frac{22}{C \times 3.0}$$

$$C = \frac{22}{0.150\ \text{MHz} \times 3.0}$$

$$C = 48.8\ \text{pF}$$

**Figure 8.10. C Mode Oscillator Example**

Therefore, the XFCN value to use in this example is 011 and C is approximately 50 pF. The recommended startup procedure for C mode is the same as RC mode.

**Recommended XFCN Settings for RC and C Modes**

**Table 8.2. Recommended XFCN Settings for RC and C Modes**

| XFCN Field Setting | Approximate Frequency Range | K Factor (C Mode) | Actual Measured Frequency (C Mode) |
|---|---|---|---|
| 000 | f ≤ 25 kHz | K Factor = 0.87 | f = 11 kHz, C = 33 pF |
| 001 | 25 kHz < f ≤ 50 kHz | K Factor = 2.6 | f = 33 kHz, C = 33 pF |
| 010 | 50 kHz < f ≤ 100 kHz | K Factor = 7.7 | f = 98 kHz, C = 33 pF |
| 011 | 100 kHz < f ≤ 200 kHz | K Factor = 22 | f = 270 kHz, C = 33 pF |
| 100 | 200 kHz < f ≤ 400 kHz | K Factor = 65 | f = 310 kHz, C = 46 pF |
| 101 | 400 kHz < f ≤ 800 kHz | K Factor = 180 | f = 890 kHz, C = 46 pF |
| 110 | 800 kHz < f ≤ 1.6 MHz | K Factor = 664 | f = 2.0 MHz, C = 46 pF |
| 111 | 1.6 MHz < f ≤ 3.2 MHz | K Factor = 1590 | f = 6.8 MHz, C = 46 pF |

### 8.3.7 External CMOS

An external CMOS clock source is also supported as a core clock source. The XTAL2/EXTCLK pin on the device serves as the external clock input when running in this mode. When not selected as the SYSCLK source, the EXTCLK input is always re-synchronized to SYSCLK. XTAL1 is not used in external CMOS clock mode.
**Note:** When selecting the EXTCLK pin as a clock input source, the pin should be skipped in the crossbar and configured as a digital input. Firmware should ensure that the external clock source is present or enable the missing clock detector before switching the CLKSL field.

The external oscillator valid detector will always return zero when the external oscillator is configured to External CMOS Clock mode.

## 8.4 Clocking and Oscillator Control Registers

### 8.4.1 CLKSEL: Clock Select

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CLKRDY | CLKDIV | | | Reserved | CLKSL | | |
| Access | R | RW | | | R | RW | | |
| Reset | 1 | 0x0 | | | 0 | 0x2 | | |
| SFR Page = ALL; SFR Address: 0xA9 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | CLKRDY | 1 | R | **System Clock Divider Clock Ready Flag.** |
| | Value | Name | | Description |
| | 0 | NOT_SET | | The selected clock divide setting has not been applied to the system clock. |
| | 1 | SET | | The selected clock divide setting has been applied to the system clock. |
| 6:4 | CLKDIV | 0x0 | RW | **Clock Source Divider.** |
| | This field controls the divider applied to the clock source selected by CLKSL. The output of this divider is the system clock (SYSCLK). | | | |
| | Value | Name | | Description |
| | 0x0 | SYSCLK_DIV_1 | | SYSCLK is equal to selected clock source divided by 1. |
| | 0x1 | SYSCLK_DIV_2 | | SYSCLK is equal to selected clock source divided by 2. |
| | 0x2 | SYSCLK_DIV_4 | | SYSCLK is equal to selected clock source divided by 4. |
| | 0x3 | SYSCLK_DIV_8 | | SYSCLK is equal to selected clock source divided by 8. |
| | 0x4 | SYSCLK_DIV_16 | | SYSCLK is equal to selected clock source divided by 16. |
| | 0x5 | SYSCLK_DIV_32 | | SYSCLK is equal to selected clock source divided by 32. |
| | 0x6 | SYSCLK_DIV_64 | | SYSCLK is equal to selected clock source divided by 64. |
| | 0x7 | SYSCLK_DIV_128 | | SYSCLK is equal to selected clock source divided by 128. |
| *3* | *Reserved* | *Must write reset value.* | | |
| 2:0 | CLKSL | 0x2 | RW | **Clock Source Select.** |
| | Selects the oscillator to be used as the undivided system clock source. | | | |
| | Value | Name | | Description |
| | 0x0 | HFOSC | | Clock derived from the internal precision High-Frequency Oscillator. |
| | 0x1 | EXTOSC | | Clock derived from the External Oscillator circuit. |
| | 0x2 | LPO_DIV_8 | | Clock derived from the Internal Low Power Oscillator divided by 8. |
| | 0x3 | RTC | | Clock derived from the RTC. |
| | 0x4 | LPOSC | | Clock derived from the Internal Low Power Oscillator. |
| There are no restrictions when switching between clock sources or divider values for this family. | | | | |

### 8.4.2 HFO0CAL: High Frequency Oscillator Calibration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SSE | HFO0CAL | | | | | | |
| Access | RW | RW | | | | | | |
| Reset | 0 | Varies | | | | | | |
| SFR Page = 0x0; SFR Address: 0xB3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | SSE | 0 | RW | **Spread Spectrum Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Spread Spectrum clock dithering disabled. |
| | 1 | ENABLED | | Spread Spectrum clock dithering enabled. |
| 6:0 | HFO0CAL | Varies | RW | **Oscillator Calibration.**<br>These bits determine the internal oscillator period. When set to 00000000b, the oscillator operates at its fastest setting. When set to 11111111b, the oscillator operates at its slowest setting. The reset value is factory calibrated to generate an internal oscillator frequency of 24.5 MHz. |

### 8.4.3 HFO0CN: High Frequency Oscillator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IOSCEN | IFRDY | Reserved | | | | | |
| Access | RW | R | RW | | | | | |
| Reset | 0 | 0 | Varies | | | | | |
| SFR Page = 0x0; SFR Address: 0xB2 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | IOSCEN | 0 | RW | **High Frequency Oscillator Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | High Frequency Oscillator disabled. |
| | 1 | ENABLED | | High Frequency Oscillator enabled. |
| 6 | IFRDY | 0 | R | **Internal Oscillator Frequency Ready Flag.** |
| | Value | Name | | Description |
| | 0 | NOT_SET | | High Frequency Oscillator is not running at its programmed frequency. |
| | 1 | SET | | High Frequency Oscillator is running at its programmed frequency. |
| *5:0* | *Reserved* | *Must write reset value.* | | |

Read-modify-write operations such as ORL and ANL must be used to set or clear the enable bit of this register to avoid modifing the reserved field.

REG0CN_OSCBIAS must be set to 1 before enabling the High Frequency Oscillator.

### 8.4.4 XOSC0CN: External Oscillator Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | XCLKVLD | XOSCMD | | | Reserved | XFCN | | |
| Access | R | RW | | | RW | RW | | |
| Reset | 0 | 0x0 | | | 0 | 0x0 | | |
| SFR Page = 0x0; SFR Address: 0xB1 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | XCLKVLD | 0 | R | **External Oscillator Valid Flag.** |
| | Provides External Oscillator status and is valid at all times for all modes of operation except External CMOS Clock Mode and External CMOS Clock Mode with divide by 2. In these modes, XCLKVLD always returns 0. | | | |
| | Value | Name | | Description |
| | 0 | NOT_SET | | External Oscillator is unused or not yet stable. |
| | 1 | SET | | External Oscillator is running and stable. |
| 6:4 | XOSCMD | 0x0 | RW | **External Oscillator Mode.** |
| | Value | Name | | Description |
| | 0x0 | DISABLED | | External Oscillator circuit disabled. |
| | 0x2 | CMOS | | External CMOS Clock Mode. |
| | 0x3 | CMOS_DIV_2 | | External CMOS Clock Mode with divide by 2 stage. |
| | 0x4 | RC | | RC Oscillator Mode. |
| | 0x5 | C | | Capacitor Oscillator Mode. |
| | 0x6 | CRYSTAL | | Crystal Oscillator Mode. |
| | 0x7 | CRYSTAL_DIV_2 | | Crystal Oscillator Mode with divide by 2 stage. |
| *3* | *Reserved* | *Must write reset value.* | | |
| 2:0 | XFCN | 0x0 | RW | **External Oscillator Frequency Control.** |
| | Controls the external oscillator bias current. The value selected for this field depends on the frequency range of the external oscillator. | | | |

# 9. Real Time Clock (RTC0)

## 9.1 Introduction

The RTC is an ultra low power, 36 hour 32-bit independent time-keeping Real Time Clock with alarm. The RTC has a dedicated 32 kHz oscillator. No external resistor or loading capacitors are required, and a missing clock detector features alerts the system if the external crystal fails. The on-chip loading capacitors are programmable to 16 discrete levels allowing compatibility with a wide range of crystals.

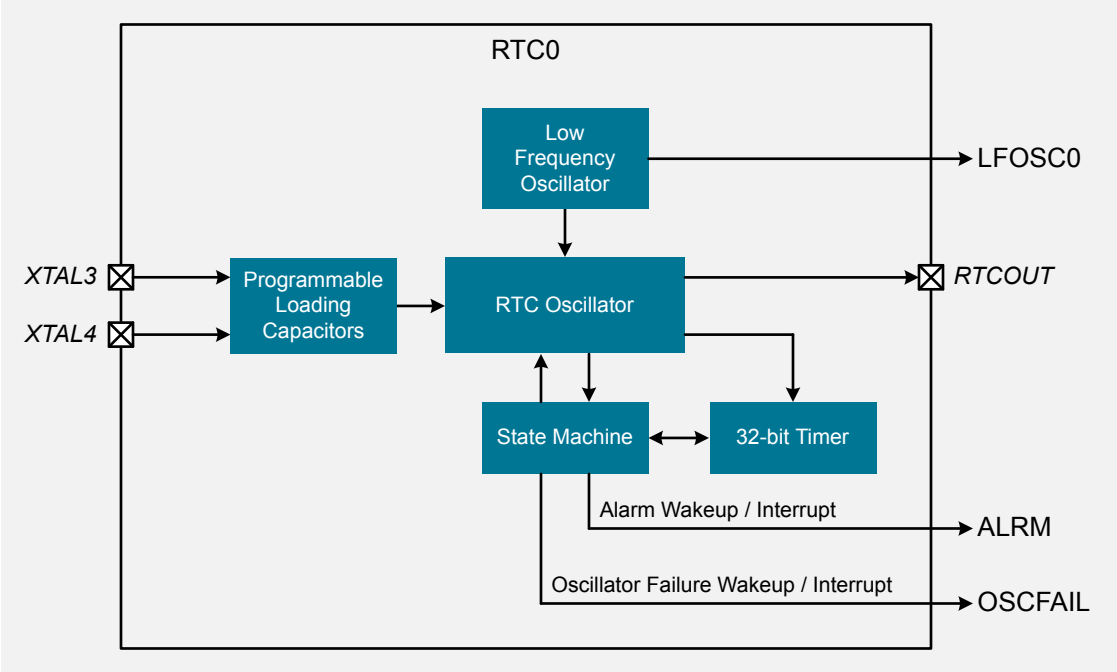



**Figure 9.1. RTC Block Diagram**

## 9.2 Features

The RTC module includes the following features:

- Up to 36 hours (32-bit) of independent time keeping.
- Support for internal 16.4 kHz low frequency oscillator (LFOSC0) or external 32 kHz crystal.
- Internal crystal loading capacitors with 16 levels.
- Operation in the lowest power mode and across the full supported voltage range.
- Alarm and oscillator failure events to wake from the lowest power mode or reset the device.
- Buffered clock output available for other system devices even in the lowest power mode.

## 9.3 Functional Description

### 9.3.1 Interface

The RTC Interface consists of three registers: RTC0KEY, RTC0ADR, and RTC0DAT. These interface registers are located on the SFR map and provide access to the RTC internal registers. The RTC internal registers can only be accessed indirectly through the RTC interface.

The RTC interface has an RTC0KEY register to be compatible across the device family. All writes to this register are ignored on this device, and the RTC interface is always unlocked.

### Accessing Internal RTC Registers

The RTC internal registers can be read and written using RTC0ADR and RTC0DAT. The RTC0ADR register selects the RTC internal register that will be targeted by subsequent reads or writes. Recommended instruction timing is provided in this section. If the recommended instruction timing is not followed, then firmware should check the BUSY bit prior to each read or write operation to make sure the RTC interface is not busy performing the previous read or write operation. An RTC write operation is initiated by writing to the RTC0DAT register:

1. Poll BUSY until it returns 0 or follow the recommended instruction timing.
2. Write the desired register address to RTC0ADR.
3. Write the desired value to RTC0DAT. This will transfer the data to the selected internal register.

An RTC read operation is initiated by setting the BUSY bit, which transfers the contents of the internal register selected by RTC0ADR to RTC0DAT. The transferred data will remain in RTC0DAT until the next read or write operation. To read an RTC register:

1. Poll BUSY until it returns 0 or follow the recommended instruction timing.
2. Write the desired register address to RTC0ADR.
3. Write 1 to BUSY. This initiates the transfer of data from the selected register to RTC0DAT.
4. Poll BUSY until it returns 0 or follow the recommend instruction timing.
5. Read the data from RTC0DAT.

**Note:** The RTC0ADR and RTC0DAT registers will retain their state upon a device reset.

### Short Strobe Feature

Reads and writes to indirect RTC registers normally take 7 system clock cycles. To minimize the indirect register access time, the short strobe feature decreases the read and write access time to 6 system clocks. The short strobe feature is automatically enabled on reset and can be manually enabled/disabled using the SHORT control bit in the RTC0ADR register. The recommended instruction timing for a single register read with short strobe enabled is as follows:

```
mov RTC0ADR, #095h
nop
nop
nop
mov A, RTC0DAT
```

The recommended instruction timing for a single register write with short strobe enabled is as follows:

```
mov RTC0ADR, #015h
mov RTC0DAT, #000h
nop
```

### Autoread Feature

When autoread is enabled, each read from RTC0DAT initiates the next indirect read operation on the RTC internal register selected by RTC0ADR. Firmware should set the BUSY bit once at the beginning of each series of consecutive reads. Firmware should follow recommended instruction timing or check if the RTC interface is busy prior to reading RTC0DAT. Autoread is enabled by setting AUTORD to 1 in the RTC0ADR register.

#### Autoincrement Feature

For ease of reading and writing the 32-bit CAPTURE and ALARM values, RTC0ADR automatically increments after each read or write to a CAPTUREn or ALARMn register. This speeds up the process of setting an alarm or reading the current RTC timer value. Auto-increment is always enabled. The recommended instruction timing for a multi-byte register read with short strobe and auto read enabled is as follows:

```
mov RTC0ADR, #0d0h
nop
nop
nop
mov A, RTC0DAT
nop
nop
mov A, RTC0DAT
nop
nop
mov A, RTC0DAT
nop
nop
mov A, RTC0DAT
```

The recommended instruction timing for a multi-byte register write with short strobe enabled is as follows:

```
mov RTC0ADR, #010h
mov RTC0DAT, #05h
nop
mov RTC0DAT, #06h
nop
mov RTC0DAT, #07h
nop
mov RTC0DAT, #08h
nop
```

### 9.3.2 Clocking Options

#### Using an External Crystal or CMOS Clock

When using crystal mode, a 32.768 kHz crystal should be connected between XTAL3 and XTAL4. No other external components are required. The following steps show how to start the RTC crystal oscillator in software:

1. If XTAL3 and XTAL4 are shared with standard GPIO functionality, set these pins to analog mode. If they XTAL3 and XTAL4 are dedicated pins, skip this step.
2. Set RTC to crystal mode (XMODE = 1).
3. Disable automatic gain control (AGCEN) and enable bias doubling (BIASX2) for fast crystal startup.
4. Set the desired loading capacitance (RTC0XCF).
5. Enable power to the RTC oscillator circuit (RTC0EN = 1).
6. Wait 20 ms.
7. Poll the RTC clock valid flag (CLKVLD) until the crystal oscillator stabilizes.
8. Poll the RTC load capacitance ready flag (LOADRDY) until the load capacitance reaches its programmed value.
9. Enable automatic gain control (AGCEN) and disable bias doubling (BIASX2) for maximum power savings.
10. Enable the RTC missing clock detector.
11. Wait 2 ms.
12. Clear the PMU0CF wake-up source flags.

While configured for crystal mode, the RTC oscillator may be driven by an external CMOS clock. The CMOS clock should be applied to XTAL3, while XTAL4 should be left floating. The RTC oscillator should be configured to its lowest bias setting with AGC disabled. The CLKVLD bit is indeterminate when using a CMOS clock, but the OSCFAIL flag may be checked 2 ms after the RTC oscillator is powered on to ensure that there is a valid clock on XTAL3.

For devices with a dedicated XTAL3 pin, the input low voltage (VIL) and input high voltage (VIH) for XTAL3 when used with an external CMOS clock are 0.1 and 0.8 V, respectively.

For devices where XTAL3 is shared with standard GPIO functionality, bias levels closer to VDD will result in lower I/O power consumption because the XTAL3 pin has a built-in weak pull-up. In this mode, the external CMOS clock is ac coupled into the RTC and should have a minimum voltage swing of 400 mV. The CMOS clock signal voltage should not exceed VDD or drop below GND.

### Using the Low Frequency Oscillator (LFO)

The low frequency oscillator provides an ultra low power, on-chip clock source to the RTC. The typical frequency of oscillation is 16.4 kHz ±20%. No external components are required to use the LFO, and the XTAL3 and XTAL4 pins do not need to be shorted together. The following steps show how to configure RTC for use with the LFO:

1. Enable and select the Low Frequency Oscillator (LFOEN = 1).
2. The LFO starts oscillating instantaneously.

When the LFO is enabled, the RTC oscillator increments bit 1 of the 32-bit timer instead of bit 0. This effectively multiplies the LFO frequency by 2, making the RTC timebase behave as if a 32.768 kHz crystal is connected at the output.

### Using Self-Oscillate Mode

When using self-oscillate mode, the XTAL3 and XTAL4 pins are internally shorted together. To configure the RTC for self-oscillate mode:

1. Set RTC to self-oscillate mode (XMODE = 0).
2. Set the desired oscillation frequency:
   - For oscillation at about 20 kHz, set BIASX2 = 0.
   - For oscillation at about 40 kHz, set BIASX2 = 1.
3. The oscillator starts oscillating instantaneously.
4. Fine tune the oscillation frequency by adjusting the load capacitance (RTC0XCF).

### Programmable Load Capacitance

The programmable load capacitance has 16 values to support crystal oscillators with a wide range of recommended load capacitance. If automatic load capacitance stepping is enabled, the crystal load capacitors start at the smallest setting to allow a fast startup time, then slowly increase the capacitance until the final programmed value is reached. The final programmed loading capacitor value is specified using the LOADCAP field in the RTC0XCF register. The LOADCAP setting specifies the amount of on-chip load capacitance and does not include any stray PCB capacitance. Once the final programmed loading capacitor value is reached, hardware will set the LOADRDY flag to 1.

When using the RTC oscillator in self-oscillate mode, the programmable load capacitance can be used to fine tune the oscillation frequency. In most cases, increasing the load capacitor value will result in a decrease in oscillation frequency.

**Table 9.1. RTC Load Capacitance Settings**

| LOADCAP Field | Crystal Load Capacitance | Equivalent Capacitance seen on XTAL3 and XTAL4 |
|---|---|---|
| 0000 | 4.0 pF | 8.0 pF |
| 0001 | 4.5 pF | 9.0 pF |
| 0010 | 5.0 pF | 10.0 pF |
| 0011 | 5.5 pF | 11.0 pF |
| 0100 | 6.0 pF | 12.0 pF |
| 0101 | 6.5 pF | 13.0 pF |
| 0110 | 7.0 pF | 14.0 pF |
| 0111 | 7.5 pF | 15.0 pF |
| 1000 | 8.0 pF | 16.0 pF |
| 1001 | 8.5 pF | 17.0 pF |
| 1010 | 9.0 pF | 18.0 pF |
| 1011 | 9.5 pF | 19.0 pF |
| 1100 | 10.5 pF | 21.0 pF |
| 1101 | 11.5 pF | 23.0 pF |
| 1110 | 12.5 pF | 25.0 pF |
| 1111 | 13.5 pF | 27.0 pF |

### Automatic Gain Control (Crystal Mode Only) and Bias Doubling

Automatic gain control (AGC) allows the RTC oscillator to trim the oscillation amplitude of a crystal in order to achieve the lowest possible power consumption. Automatic gain control automatically detects when the oscillation amplitude has reached a point where it safe to reduce the drive current, so it may be enabled during crystal startup. It is recommended to enable AGC in most systems which use the RTC oscillator in crystal mode. The following are recommended crystal specifications and operating conditions when AGC is enabled:

- ESR < 50 kΩ
- Load Capacitance < 10 pF
- Supply Voltage < 3.0 V
- Temperature > –20 °C

When using AGC, it is recommended to perform an oscillation robustness test to ensure that the chosen crystal will oscillate under the worst case condition to which the system will be exposed. The worst case condition that should result in the least robust oscillation is at the following system conditions: lowest temperature, highest supply voltage, highest ESR, highest load capacitance, and lowest bias current (AGC enabled, bias doubling disabled).

To perform the oscillation robustness test, the RTC oscillator should be enabled and selected as the system clock source. Next, the SYSCLK signal should be routed to a port pin configured as a push-pull digital output. The positive duty cycle of the output clock can be used as an indicator of oscillation robustness. Duty cycles less than 55% indicate a robust oscillation. As the duty cycle approaches 60%, oscillation becomes less reliable and the risk of clock failure increases. Increasing the bias current (by disabling AGC) will always improve oscillation robustness and will reduce the output clock's duty cycle. This test should be performed at the worst case system conditions, as results at very low temperatures or high supply voltage will vary from results taken at room temperature or low supply voltage.

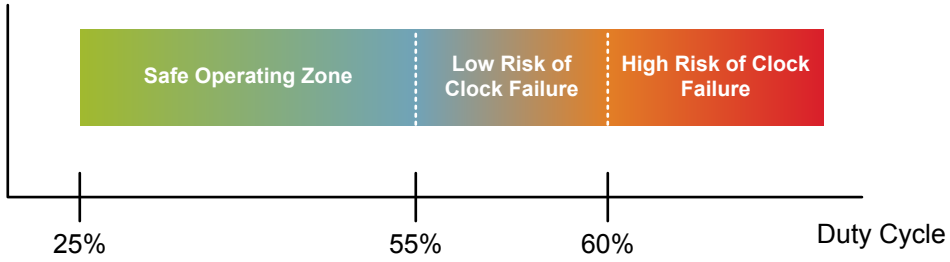



**Figure 9.2. Interpreting Oscillation Robustness (Duty Cycle) Test Results**

As an alternative to performing the oscillation robustness test, AGC may be disabled at the cost of increased power consumption (approximately 200 nA). Disabling AGC will provide the crystal oscillator with higher immunity against external factors which may lead to clock failure. AGC must be disabled if using the RTC oscillator in self-oscillate mode. The RTC bias doubling feature allows the self-oscillation frequency to be increased (almost doubled) and allows a higher crystal drive strength in crystal mode. High crystal drive strength is recommended when the crystal is exposed to poor environmental conditions such as excessive moisture. RTC bias doubling is enabled by setting BIASX2 to 1.

**Table 9.2. RTC Load Capacitance Settings**

| Mode | Setting | Power Consumption |
|---|---|---|
| Crystal | Bias double off, AGC on | Lowest<br>600 nA |
| | Bias double off, AGC off | Low<br>800 nA |
| | Bias double on, AGC on | High |
| | Bias double on, AGC off | Highest |
| Self-Oscillate | Bias double off | Low |
| | Bias double on | High |

#### Missing Clock Detector

The missing RTC detector is a one-shot circuit enabled by setting MCLKEN to 1. When the RTC missing clock detector is enabled, OSCFAIL is set by hardware if the RTC oscillator remains high or low for more than 100 µs. An RTC missing clock detector timeout can trigger an interrupt, wake the device from a low power mode, or reset the device.
**Note:** The RTC missing clock detector should be disabled when making changes to the oscillator settings in RTC0XCN0.

#### Oscillator Crystal Valid Detector

The RTC oscillator crystal valid detector is an oscillation amplitude detector circuit used during crystal startup to determine when oscillation has started and is nearly stable. The output of this detector can be read from the CLKVLD bit.
**Note:** The CLKVLD bit has a blanking interval of 2 ms. During the first 2 ms after turning on the crystal oscillator, the output of CLKVLD is not valid.

**Note:** This RTC crystal valid detector (CLKVLD) is not intended for detecting an oscillator failure. The missing RTC detector (OSCFAIL) should be used for this purpose.

### 9.3.3 Timer and Alarm

The RTC timer is a 32-bit counter that, when running (RTC0TR = 1), is incremented every RTC oscillator cycle. The timer has an alarm function that can be set to generate an interrupt, wake the device from a low power mode, or reset the device at a specific time.

The RTC timer includes an auto reset feature, which automatically resets the timer to zero one RTC cycle after the alarm signal is deasserted. When using auto reset, the Alarm match value should always be set to 2 counts less than the desired match value. When using the LFO in combination with auto reset, the right-justified Alarm match value should be set to 4 counts less than the desired match value. Auto reset can be enabled by writing a 1 to ALRM.

#### Setting and Reading the RTC Timer

The 32-bit RTC timer can be set or read using the CAPTUREn internal registers. Note that the timer does not need to be stopped before reading or setting its value. The following steps can be used to set the timer value:

1. Write the desired 32-bit set value to the CAPTUREn registers.
2. Write 1 to RTC0SET. This will transfer the contents of the CAPTUREn registers to the RTC timer.
3. The operation is complete when RTC0SET is cleared to 0 by hardware.

To read the current timer value:

1. Write 1 to RTC0CAP. This will transfer the contents of the timer to the CAPTUREn registers.
2. Poll RTC0CAP until it is cleared to 0 by hardware.
3. A snapshot of the timer value can be read from the CAPTUREn registers

#### Setting an RTC Alarm

The RTC alarm function compares the 32-bit value of the RTC timer to the value of the ALARMn registers. An alarm event is triggered if the RTC timer is equal to the ALARMn registers. If auto reset is enabled, the 32-bit timer will be cleared to zero one RTC cycle after the alarm event. The RTC alarm event can be configured to reset the MCU, wake it up from a low power mode, or generate an interrupt. To set up an RTC alarm:

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

**Note:** The ALRM bit, which is used as the RTC Alarm event flag, is cleared by disabling RTC Alarm events (RTC0AEN = 0).

**Note:** If auto reset is disabled, disabling (RTC0AEN = 0) then re-enabling alarm events (RTC0AEN = 1) after an RTC Alarm without modifying ALARMn registers will automatically schedule the next alarm after $2^{32}$ RTC cycles (approximately 36 hours using a 32.768 kHz crystal).

**Note:** The RTC Alarm event flag will remain asserted for a maximum of one RTC cycle. When using the RTC in conjunction with low power modes, the PMU must be used to determine the cause of the last wake event.

### Software Considerations

The RTC timer and alarm have two operating modes to suit varying applications:

#### Mode 1

The first mode uses the RTC timer as a perpetual timebase which is never reset to zero. Every 36 hours, the timer is allowed to overflow without being stopped or disrupted. The alarm interval is software managed and is added to the ALRMn registers by software after each alarm. This allows the alarm match value to always stay ahead of the timer by one software managed interval. If software uses 32-bit unsigned addition to increment the alarm match value, then it does not need to handle overflows since both the timer and the alarm match value will overflow in the same manner. This mode is ideal for applications which have a long alarm interval (e.g., 24 or 36 hours) and/or have a need for a perpetual timebase. An example of an application that needs a perpetual timebase is one whose wake-up interval is constantly changing. For these applications, software can keep track of the number of timer overflows in a 16-bit variable, extending the 32-bit (36 hour) timer to a 48-bit (272 year) perpetual timebase.

#### Mode 2

The second mode uses the RTC timer as a general purpose up counter which is auto reset to zero by hardware after each alarm. The alarm interval is managed by hardware and stored in the ALRMn registers. Software only needs to set the alarm interval once during device initialization. After each alarm, software should keep a count of the number of alarms that have occurred in order to keep track of time. This mode is ideal for applications that require minimal software intervention and/or have a fixed alarm interval. This mode is the most power efficient since it requires less CPU time per alarm.

## 9.4 Clocking and Oscillator Control Registers

### 9.4.1 RTC0KEY: RTC Lock and Key

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RTC0ST | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xAE | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | RTC0ST | 0x00 | RW | **RTC Interface Status.**<br>Provides lock status when read.<br>Writes to RTC0KEY have no effect. |

| Value | Name | Description |
|---|---|---|
| 0x02 | UNLOCKED | RTC Interface is unlocked. |

### 9.4.2 RTC0ADR: RTC Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | BUSY | AUTORD | Reserved | SHORT | ADDR | | | |
| Access | RW | RW | R | RW | RW | | | |
| Reset | 0 | 0 | 0 | 0 | 0x0 | | | |
| SFR Page = 0x0; SFR Address: 0xAC | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | BUSY | 0 | RW | **RTC Interface Busy Indicator.**<br>This bit indicates the RTC interface status. Writing a 1 to this bit initiates an indirect read. |
| 6 | AUTORD | 0 | RW | **RTC Interface Autoread Enable.**<br>When autoread is enabled, firmware should set the BUSY bit once at the beginning of each series of consecutive reads. Firmware must check if the RTC Interface is busy prior to reading RTC0DAT. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable autoread. Firmware must write the BUSY bit for each RTC indirect read operation. |
| | 1 | ENABLED | | Enable autoread. The next RTC indirect read operation is initiated when firmware reads the RTC0DAT register. |
| *5* | *Reserved* | *Must write reset value.* | | |
| 4 | SHORT | 0 | RW | **Short Strobe Enable.**<br>Enables/disables the Short Strobe feature. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable short strobe. |
| | 1 | ENABLED | | Enable short strobe. |
| 3:0 | ADDR | 0x0 | RW | **RTC Indirect Register Address.**<br>Sets the currently-selected RTC internal register. |
| The ADDR bits increment after each indirect read/write operation that targets a CAPTUREn or ALARMn internal RTC register. | | | | |

### 9.4.3 RTC0DAT: RTC Data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RTC0DAT | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xAD | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | RTC0DAT | 0x00 | RW | **RTC Data.**<br>Holds data transferred to/from the internal RTC register selected by RTC0ADR. |
| Read-modify-write instructions (orl, anl, etc.) should not be used on this register. | | | | |

### 9.4.4 RTC0CN0: RTC Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RTC0EN | MCLKEN | OSCFAIL | RTC0TR | RTC0AEN | ALRM | RTC0SET | RTC0CAP |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | Varies | 0 | 0 | 0 | 0 | 0 |
| Indirect Address: 0x04 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | RTC0EN | 0 | RW | **RTC Enable.**<br>Enables/disables the RTC oscillator and associated bias currents. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable RTC oscillator. |
| | 1 | ENABLED | | Enable RTC oscillator. |
| 6 | MCLKEN | 0 | RW | **Missing RTC Detector Enable.**<br>Enables/disables the missing RTC detector. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable missing RTC detector. |
| | 1 | ENABLED | | Enable missing RTC detector. |
| 5 | OSCFAIL | Varies | RW | **RTC Oscillator Fail Event Flag.**<br>Set by hardware when a missing RTC detector timeout occurs. Must be cleared by firmware. The value of this bit is not defined when the RTC oscillator is disabled. |
| 4 | RTC0TR | 0 | RW | **RTC Timer Run Control.**<br>Controls if the RTC timer is running or stopped (holds current value). |
| | Value | Name | | Description |
| | 0 | STOP | | RTC timer is stopped. |
| | 1 | RUN | | RTC timer is running. |
| 3 | RTC0AEN | 0 | RW | **RTC Alarm Enable.**<br>Enables/disables the RTC alarm function. Also clears the ALRM flag. |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable RTC alarm. |
| | 1 | ENABLED | | Enable RTC alarm. |
| 2 | ALRM | 0 | RW | **RTC Alarm Event Flag and Auto Reset Enable.**<br>Reads return the state of the alarm event flag.<br>Writes enable/disable the Auto Reset function. |
| | Value | Name | | Description |
| | 0 | NOT_SET | | Alarm event flag is not set or disable the auto reset function. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | 1 | SET | | Alarm event flag is set or enable the auto reset function. |
| 1 | RTC0SET | 0 | RW | **RTC Timer Set.**<br>Writing 1 initiates a RTC timer set operation. This bit is cleared to 0 by hardware to indicate that the timer set operation is complete. |
| 0 | RTC0CAP | 0 | RW | **RTC Timer Capture.**<br>Writing 1 initiates a RTC timer capture operation. This bit is cleared to 0 by hardware to indicate that the timer capture operation is complete. |

The ALRM flag will remain asserted for a maximum of one RTC cycle.

This register is accessed indirectly using the RTC0ADR and RTC0DAT registers.

### 9.4.5 RTC0XCN0: RTC Oscillator Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AGCEN | XMODE | BIASX2 | CLKVLD | LFOEN | Reserved | | |
| Access | RW | RW | RW | R | RW | R | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0x0 | | |
| Indirect Address: 0x05 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | AGCEN | 0 | RW | **RTC Oscillator Automatic Gain Control (AGC) Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable AGC. |
| | 1 | ENABLED | | Enable AGC. |
| 6 | XMODE | 0 | RW | **RTC Oscillator Mode.** |
| | Selects Crystal or Self Oscillate Mode. | | | |
| | Value | Name | | Description |
| | 0 | SELF_OSCILLATE | | Self-Oscillate Mode selected. |
| | 1 | CRYSTAL | | Crystal Mode selected. |
| 5 | BIASX2 | 0 | RW | **RTC Oscillator Bias Double Enable.** |
| | Enables/disables the Bias Double feature. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable the Bias Double feature. |
| | 1 | ENABLED | | Enable the Bias Double feature. |
| 4 | CLKVLD | 0 | R | **RTC Oscillator Crystal Valid Indicator.** |
| | Indicates if oscillation amplitude is sufficient for maintaining oscillation. | | | |
| | Value | Name | | Description |
| | 0 | NOT_SET | | Oscillation has not started or oscillation amplitude is too low to maintain oscillation. |
| | 1 | SET | | Sufficient oscillation amplitude detected. |
| 3 | LFOEN | 0 | RW | **Low Frequency Oscillator Enable and Select.** |
| | Overrides XMODE and selects the internal low frequency oscillator (LFOSC0) as the RTC oscillator source. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | XMODE determines RTC oscillator source. |
| | 1 | ENABLED | | LFOSC0 enabled and selected as RTC oscillator source. |
| *2:0* | *Reserved* | *Must write reset value.* | | |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.6 RTC0XCF: RTC Oscillator Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AUTOSTP | LOADRDY | Reserved | | LOADCAP | | | |
| Access | RW | R | R | | RW | | | |
| Reset | 0 | 0 | 0x0 | | Varies | | | |
| Indirect Address: 0x06 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | AUTOSTP | 0 | RW | **Automatic Load Capacitance Stepping Enable.**<br>Enables/disables automatic load capacitance stepping.<br>Value: 0 – DISABLED – Disable load capacitance stepping.<br>Value: 1 – ENABLED – Enable load capacitance stepping. |
| 6 | LOADRDY | 0 | R | **Load Capacitance Ready Indicator.**<br>Set by hardware when the load capacitance matches the programmed value.<br>Value: 0 – NOT_SET – Load capacitance is currently stepping.<br>Value: 1 – SET – Load capacitance has reached it programmed value. |
| *5:4* | *Reserved* | *Must write reset value.* | | |
| 3:0 | LOADCAP | Varies | RW | **Load Capacitance Programmed Value.**<br>Holds the desired load capacitance value. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.7 CAPTURE0: RTC Timer Capture 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CAPTURE0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x00 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CAPTURE0 | 0x00 | RW | **RTC Timer Capture 0.**<br>The CAPTURE3-CAPTURE0 registers are used to read or set the 32-bit RTC timer. Data is transferred to or from the RTC timer when the RTC0SET or RTC0CAP bits are set. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.8 CAPTURE1: RTC Timer Capture 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CAPTURE1 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x01 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CAPTURE1 | 0x00 | RW | **RTC Timer Capture 1.**<br>The CAPTURE3-CAPTURE0 registers are used to read or set the 32-bit RTC timer. Data is transferred to or from the RTC timer when the RTC0SET or RTC0CAP bits are set. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.9 CAPTURE2: RTC Timer Capture 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CAPTURE2 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x02 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CAPTURE2 | 0x00 | RW | **RTC Timer Capture 2.**<br>The CAPTURE3-CAPTURE0 registers are used to read or set the 32-bit RTC timer. Data is transferred to or from the RTC timer when the RTC0SET or RTC0CAP bits are set. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.10 CAPTURE3: RTC Timer Capture 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CAPTURE3 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x03 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CAPTURE3 | 0x00 | RW | **RTC Timer Capture 3.**<br>The CAPTURE3-CAPTURE0 registers are used to read or set the 32-bit RTC timer. Data is transferred to or from the RTC timer when the RTC0SET or RTC0CAP bits are set. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.11 ALARM0: RTC Alarm Programmed Value 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ALARM0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x08 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ALARM0 | 0x00 | RW | **RTC Alarm Programmed Value 0.**<br>The ALARM3-ALARM0 registers are used to set an alarm event for the RTC timer. The RTC alarm should be disabled (RTC0AEN=0) when updating these registers. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.12 ALARM1: RTC Alarm Programmed Value 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ALARM1 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x09 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ALARM1 | 0x00 | RW | **RTC Alarm Programmed Value 1.**<br>The ALARM3-ALARM0 registers are used to set an alarm event for the RTC timer. The RTC alarm should be disabled (RTC0AEN=0) when updating these registers. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.13 ALARM2: RTC Alarm Programmed Value 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ALARM2 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x0A | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ALARM2 | 0x00 | RW | **RTC Alarm Programmed Value 2.**<br>The ALARM3-ALARM0 registers are used to set an alarm event for the RTC timer. The RTC alarm should be disabled (RTC0AEN=0) when updating these registers. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

### 9.4.14 ALARM3: RTC Alarm Programmed Value 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ALARM3 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| Indirect Address: 0x0B | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ALARM3 | 0x00 | RW | **RTC Alarm Programmed Value 3.**<br>The ALARM3-ALARM0 registers are used to set an alarm event for the RTC timer. The RTC alarm should be disabled (RTC0AEN=0) when updating these registers. |
| This register is accessed indirectly using the RTC0ADR and RTC0DAT registers. | | | | |

# 10. Reset Sources and Power Supply Monitor

## 10.1 Introduction

Reset circuitry allows the controller to be easily placed in a predefined default condition. On entry to this reset state, the following occur:

- The core halts program execution.
- Module registers are initialized to their defined reset values unless the bits reset only with a power-on reset.
- External port pins are forced to a known state.
- Interrupts and timers are disabled.

All registers are reset to the predefined values noted in the register descriptions unless the bits only reset with a power-on reset. The contents of RAM are unaffected during a reset; any previously stored data is preserved as long as power is not lost. The Port I/O latches are reset to 1 in open-drain mode. Weak pullups are enabled during and after the reset. For Supply Monitor and power-on resets, the RSTb pin is driven low until the device exits the reset state. On exit from the reset state, the program counter (PC) is reset, and the system clock defaults to an internal oscillator. The Watchdog Timer is enabled, and program execution begins at location 0x0000.

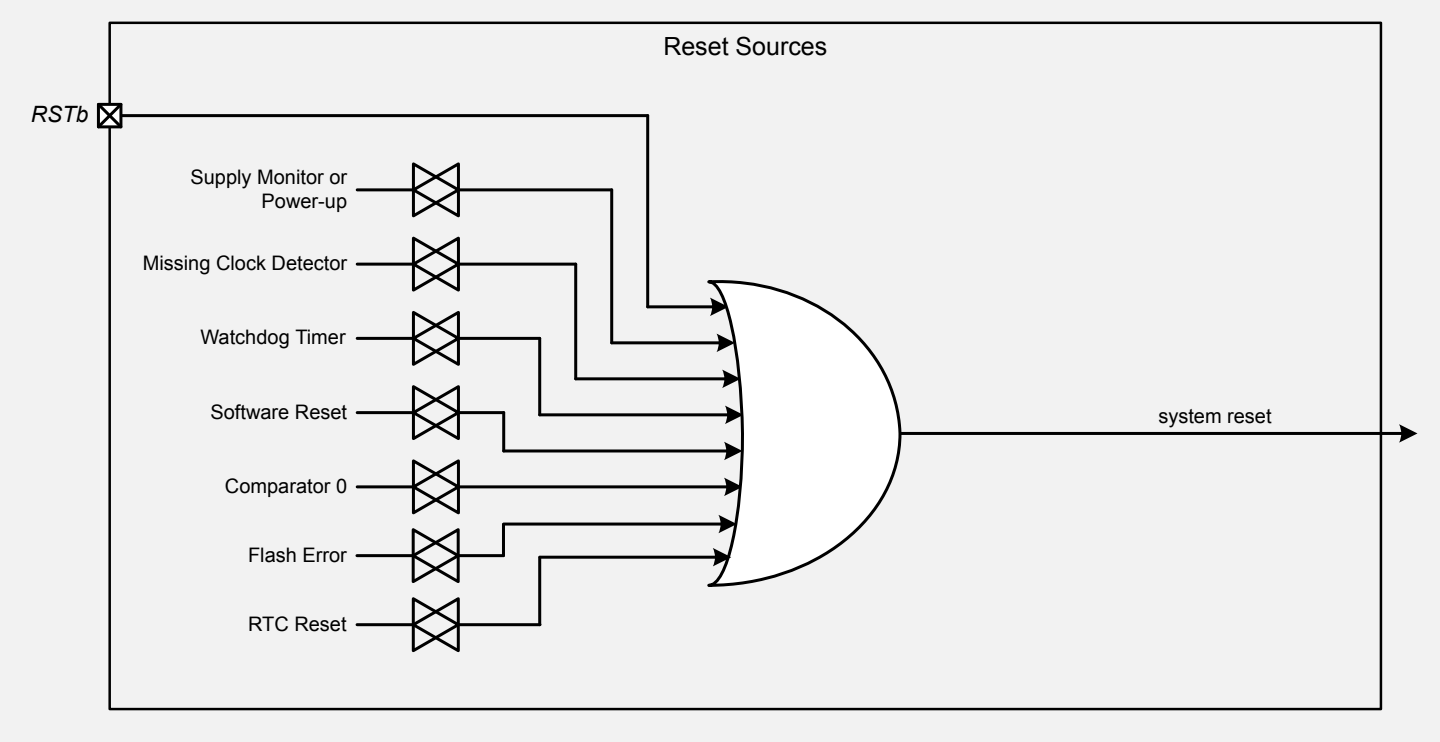



**Figure 10.1. Reset Sources Block Diagram**

## 10.2 Features

Reset sources on the device include the following:

- Power-on reset
- External reset pin
- Comparator reset
- Software-triggered reset
- Supply monitor reset (monitors VDD supply)
- Watchdog timer reset
- Missing clock detector reset
- Flash error reset
- RTC0 alarm or oscillator failure

## 10.3 Functional Description

### 10.3.1 Device Reset

Upon entering a reset state from any source, the following events occur:

- The processor core halts program execution.
- Special Function Registers (SFRs) are initialized to their defined reset values.
- External port pins are placed in a known state.
- Interrupts and timers are disabled.

SFRs are reset to the predefined reset values noted in the detailed register descriptions. The contents of internal data memory are unaffected during a reset; any previously stored data is preserved. However, since the stack pointer SFR is reset, the stack is effectively lost, even though the data on the stack is not altered.

The port I/O latches are reset to 0xFF (all logic ones) in open-drain mode. Weak pullups are enabled during and after the reset. For Supply Monitor and power-on resets, the RSTb pin is driven low until the device exits the reset state.
**Note:** During a power-on event, there may be a short delay before the POR circuitry fires and the RSTb pin is driven low. During that time, the RSTb pin will be weakly pulled to the supply pin.

On exit from the reset state, the program counter (PC) is reset, the watchdog timer is enabled, and the system clock defaults to an internal oscillator. Program execution begins at location 0x0000.

**Note:** On device reset or upon waking up from Sleep mode, address 0x0000 of external memory (XRAM) may be overwritten by an indeterminate value. The indeterminate value is 0x00 in most situations. A dummy variable should be placed at address 0x0000 in external memory to ensure that the application firmware does not store any data that needs to be retained through reset or Sleep at this memory location.

### 10.3.2 Power-On Reset

During power-up, the POR circuit fires. When POR fires, the device is held in a reset state and the RSTb pin is high-impedance with the weak pull-up off until the supply voltage settles above $V_{RST}$. Two delays are present during the supply ramp time. First, a delay occurs before the POR circuitry fires and pulls the RSTb pin low. A second delay occurs before the device is released from reset; the delay decreases as the supply ramp time increases (supply ramp time is defined as how fast the supply pin ramps from 0 V to $V_{RST}$). For ramp times less than 1 ms, the power-on reset time ($T_{POR}$) is typically less than 0.3 ms. Additionally, the power supply must reach $V_{RST}$ before the POR circuit releases the device from reset.

On exit from a power-on reset, the PORSF flag 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 supply monitor is enabled following a power-on reset.

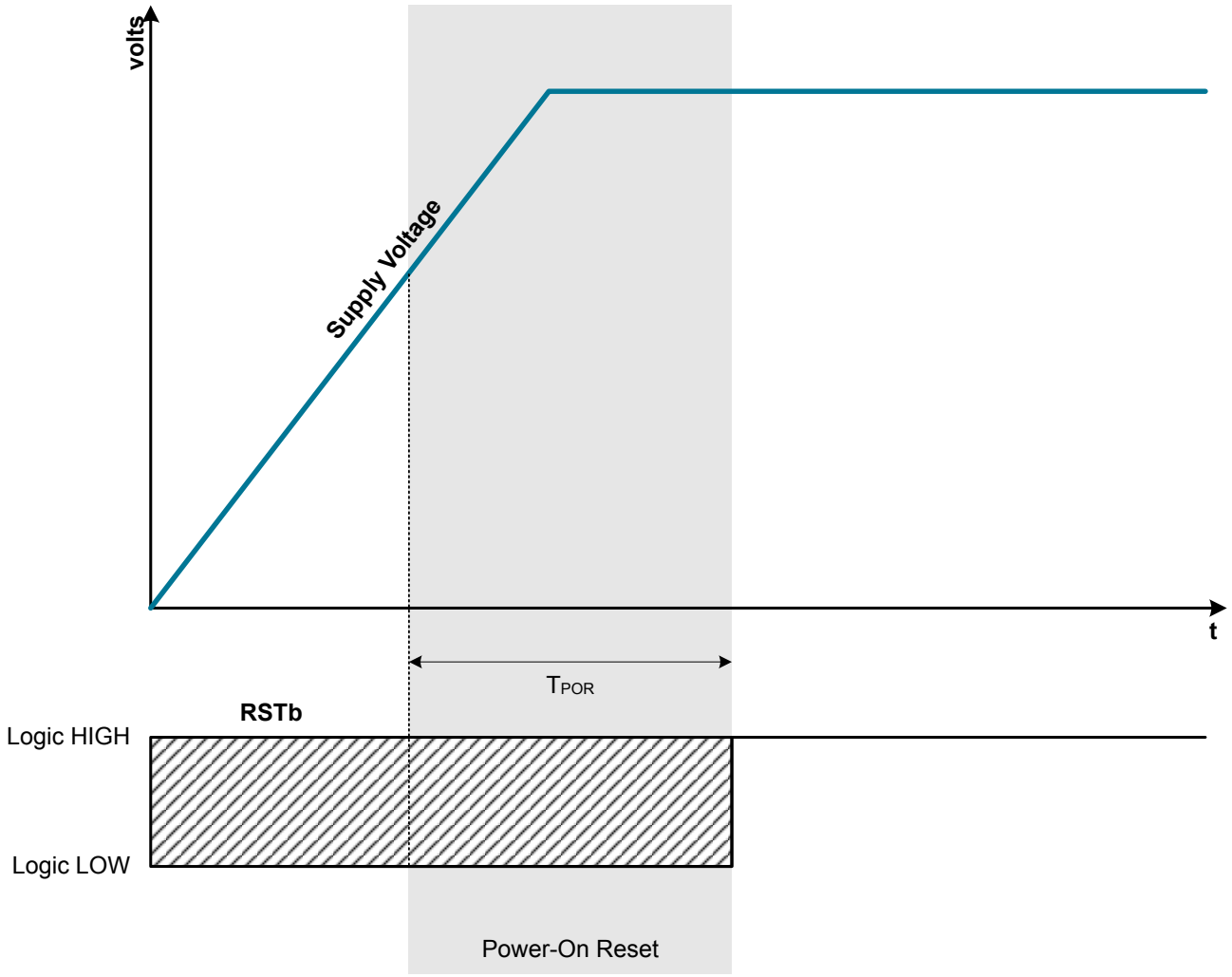



**Figure 10.2. Power-On Reset Timing**

### 10.3.3 Supply Monitor Reset

The supply monitor senses the voltage on the device's supply pin and can generate a reset if the supply drops below the corresponding threshold. This monitor is enabled and enabled as a reset source after initial power-on to protect the device until the supply is an adequate and stable voltage. When enabled and selected as a reset source, any power down transition or power irregularity that causes the supply to drop below the reset threshold will drive the RSTb pin low and hold the core in a reset state. When the supply returns to a level above the reset threshold, the monitor will release the core from the reset state. The reset status can then be read using the device reset sources module. After a power-fail reset, the PORF flag reads 1 and all of the other reset flags in the RSTSRC register are indeterminate. The power-on reset delay ($t_{POR}$) is not incurred after a supply monitor reset. The contents of RAM should be presumed invalid after a supply monitor reset. The enable state of the supply monitor and its selection as a reset source is not altered by device resets. For example, if the supply monitor is de-selected as a reset source and disabled by software using the VDMEN bit in the VDM0CN register, and then firmware performs a software reset, the supply monitor will remain disabled and de-selected after the reset. To protect the integrity of flash contents, the supply monitor must be enabled and selected as a reset source if software contains routines that erase or write flash memory. If the supply monitor is not enabled, any erase or write performed on flash memory will be ignored.

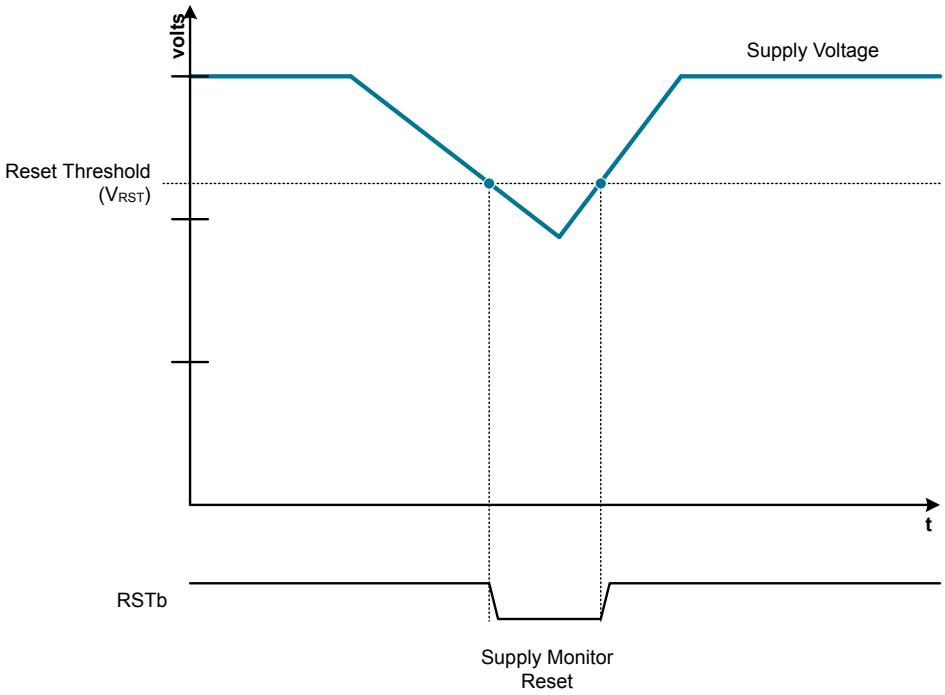



**Figure 10.3. Reset Sources**

### 10.3.4 External Reset

The external RSTb pin provides a means for external circuitry to force the device into a reset state. Asserting an active-low signal on the RSTb pin generates a reset; an external pullup and/or decoupling of the RSTb pin may be necessary to avoid erroneous noise-induced resets. The PINRSF flag is set on exit from an external reset.

### 10.3.5 Missing Clock Detector Reset

The Missing Clock Detector (MCD) is a one-shot circuit that is triggered by the system clock. If the system clock remains high or low for more than the MCD time window, the one-shot will time out and generate a reset. After a MCD reset, the MCDRSF flag will read 1, signifying the MCD as the reset source; otherwise, this bit reads 0. Writing a 1 to the MCDRSF bit enables the Missing Clock Detector; writing a 0 disables it. The state of the RSTb pin is unaffected by this reset.

### 10.3.6 Comparator (CMP0) Reset

Comparator0 can be configured as a reset source by writing a 1 to the C0RSEF flag. Comparator0 should be enabled and allowed to settle prior to writing to C0RSEF to prevent any turn-on chatter on the output from generating an unwanted reset. The Comparator0 reset is active-low: if the non-inverting input voltage (on CP0+) is less than the inverting input voltage (on CP0–), the device is put into the reset state. After a Comparator0 reset, the C0RSEF flag will read 1 signifying Comparator0 as the reset source; otherwise, this bit reads 0. The state of the RSTb pin is unaffected by this reset.

### 10.3.7 PCA Watchdog Timer Reset

The programmable watchdog timer (WDT) function of the programmable counter array (PCA) can be used to prevent software from running out of control during a system malfunction. The PCA WDT function can be enabled or disabled by software as described in the PCA documentation. The WDT is enabled and clocked by SYSCLK/12 following any reset. If a system malfunction prevents user software from updating the WDT, a reset is generated and the WDTRSF bit in RSTSRC is set to 1. The state of the RSTb pin is unaffected by this reset.

### 10.3.8 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.
- A flash read is attempted above user code space.
- A program read is attempted above user code space (i.e., a branch instruction to the reserved area).
- A flash read, write or erase attempt is restricted due to a flash security setting.

The FERROR bit is set following a flash error reset. The state of the RSTb pin is unaffected by this reset.

### 10.3.9 Software Reset

Software may force a reset by writing a 1 to the SWRSF bit. The SWRSF bit will read 1 following a software forced reset. The state of the RSTb pin is unaffected by this reset.

### 10.3.10 RTC Reset

The RTC can generate a system reset on two events: RTC oscillator fail or RTC alarm. The RTC oscillator fail event occurs when the RTC missing clock detector is enabled and the RTC clock is below approximately 20 kHz. A RTC alarm event occurs when the RTC alarm is enabled and the RTC timer value matches the ALARMn registers. The RTC can be configured as a reset source by writing a 1 to the RTC0RE flag in the RSTSRC register. The RTC reset remains functional even when the device is in the low power Suspend or Sleep mode. The state of the RSTb pin is unaffected by this reset.

## 10.4 Reset Sources and Supply Monitor Control Registers

### 10.4.1 RSTSRC: Reset Source

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RTC0RE | FERROR | C0RSEF | SWRSF | WDTRSF | MCDRSF | PORSF | PINRSF |
| Access | RW | R | RW | RW | R | RW | RW | R |
| Reset | Varies | Varies | Varies | Varies | Varies | Varies | Varies | Varies |
| SFR Page = 0x0; SFR Address: 0xEF | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | RTC0RE | Varies | RW | **RTC Reset Enable and Flag.**<br>Read: This bit reads 1 if a RTC alarm or oscillator fail caused the last reset.<br>Write: Writing a 1 to this bit enables the RTC as a reset source. |
| 6 | FERROR | Varies | R | **Flash Error Reset Flag.**<br>This read-only bit is set to '1' if a flash read/write/erase error caused the last reset. |
| 5 | C0RSEF | Varies | RW | **Comparator0 Reset Enable and Flag.**<br>Read: This bit reads 1 if Comparator 0 caused the last reset.<br>Write: Writing a 1 to this bit enables Comparator 0 (active-low) as a reset source. |
| 4 | SWRSF | Varies | RW | **Software Reset Force and Flag.**<br>Read: This bit reads 1 if last reset was caused by a write to SWRSF.<br>Write: Writing a 1 to this bit forces a system reset. |
| 3 | WDTRSF | Varies | R | **Watchdog Timer Reset Flag.**<br>This read-only bit is set to '1' if a watchdog timer overflow caused the last reset. |
| 2 | MCDRSF | Varies | RW | **Missing Clock Detector Enable and Flag.**<br>Read: This bit reads 1 if a missing clock detector timeout caused the last reset.<br>Write: Writing a 1 to this bit enables the missing clock detector. The MCD triggers a reset if a missing clock condition is detected. |
| 1 | PORSF | Varies | RW | **Power-On / Supply Monitor Reset Flag, and Supply Monitor Reset Enable.**<br>Read: This bit reads 1 anytime a power-on or supply monitor reset has occurred.<br>Write: Writing a 1 to this bit enables the supply monitor as a reset source. |
| 0 | PINRSF | Varies | R | **HW Pin Reset Flag.**<br>This read-only bit is set to '1' if the RSTb pin caused the last reset. |

Reads and writes of the RSTSRC register access different logic in the device. Reading the register always returns status information to indicate the source of the most recent reset. Writing to the register activates certain options as reset sources. It is recommended to not use any kind of read-modify-write operation on this register.

When the PORSF bit reads back '1' all other RSTSRC flags are indeterminate.

Writing '1' to the PORSF bit when the supply monitor is not enabled and stabilized may cause a system reset.

### 10.4.2 VDM0CN: VDD Supply Monitor Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | VDMEN | VDDSTAT | VDDOK | Reserved | VDDOKIE | Reserved | | |
| Access | RW | R | R | R | RW | R | | |
| Reset | 1 | Varies | Varies | 0 | 1 | 0x0 | | |
| SFR Page = 0x0; SFR Address: 0xFF | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | VDMEN | 1 | RW | **V<subscript>DD</subscript> Supply Monitor Enable.**<br>This bit turns the $V_{DD}$ supply monitor circuit on/off. The $V_{DD}$ Supply Monitor cannot generate system resets until it is also selected as a reset source in register RSTSRC.<br>0: DISABLED — Disable the $V_{DD}$ supply monitor.<br>1: ENABLED — Enable the $V_{DD}$ supply monitor. |
| 6 | VDDSTAT | Varies | R | **V<subscript>DD</subscript> Supply Status.**<br>This bit indicates the current power supply status.<br>0: VDD_BELOW_VRST — $V_{DD}$ is at or below the VRST threshold.<br>1: VDD_ABOVE_VRST — $V_{DD}$ is above the VRST threshold. |
| 5 | VDDOK | Varies | R | **V<subscript>DD</subscript> Supply Status (Early Warning).**<br>This bit indicates the current VDD power supply status.<br>0: VDD_BELOW_VDDWARN — $V_{DD}$ is at or below the VDDWARN threshold.<br>1: VDD_ABOVE_VDDWARN — $V_{DD}$ is above the VDDWARN threshold. |
| *4* | *Reserved* | *Must write reset value.* | | |
| 3 | VDDOKIE | 1 | RW | **V<subscript>DD</subscript> Early Warning Interrupt Enable.**<br>Enables the $V_{DD}$ Early Warning interrupt.<br>0: DISABLED — Disable the $V_{DD}$ Early Warning interrupt.<br>1: ENABLED — Enable the $V_{DD}$ Early Warning interrupt. |
| *2:0* | *Reserved* | *Must write reset value.* | | |

# 11. CIP-51 Microcontroller Core

## 11.1 Introduction

The CIP-51 microcontroller core is a high-speed, pipelined, 8-bit core utilizing the standard MCS-51™ instruction set. Any 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 includes on-chip debug hardware and interfaces directly with the analog and digital subsystems providing a complete data acquisition or control system solution.

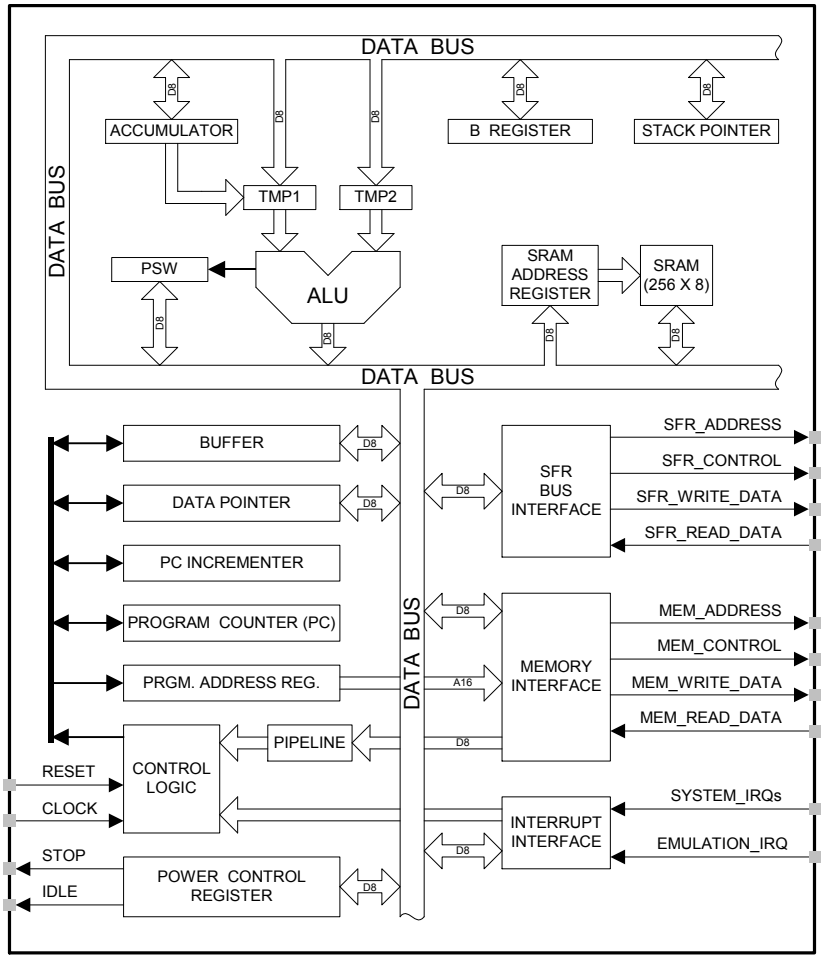



**Figure 11.1. CIP-51 Block Diagram**

**Performance**

The CIP-51 employs a pipelined architecture that greatly increases its instruction throughput over the standard 8051 architecture. The CIP-51 core executes 76 of its 109 instructions in one or two clock cycles, with no instructions taking more than eight clock cycles. The table below shows the distribution of instructions vs. the number of clock cycles required for execution.

**Table 11.1. Instruction Execution Timing**

| Clocks to Execute | 1 | 2 | 2 or 3 | 3 | 3 or 4 | 4 | 4 or 5 | 5 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| Number of Instructions | 26 | 50 | 5 | 14 | 7 | 3 | 1 | 2 | 1 |

Notes:
1. Conditional branch instructions (indicated by "2 or 3", "3 or 4" and "4 or 5") require an extra clock cycle if the branch is taken.

## 11.2 Features

The CIP-51 Microcontroller core implements the standard 8051 organization and peripherals as well as additional custom peripherals and functions to extend its capability. The CIP-51 includes the following features:

- Fast, efficient, pipelined architecture.
- Fully compatible with MCS-51 instruction set.
- 0 to 25 MHz operating clock frequency.
- 25 MIPS peak throughput with 25 MHz clock.
- Extended interrupt handler.
- Power management modes.
- On-chip debug logic.
- Program and data memory security.

## 11.3 Functional Description

### 11.3.1 Programming and Debugging Support

In-system programming of the flash program memory and communication with on-chip debug support logic is accomplished via the Silicon Labs 2-Wire development interface (C2).

The on-chip debug support logic facilitates full speed in-circuit debugging, allowing the setting of hardware breakpoints, starting, stopping and single stepping through program execution (including interrupt service routines), examination of the program's call stack, and reading/writing the contents of registers and memory. This method of on-chip debugging is completely non-intrusive, requiring no RAM, stack, timers, or other on-chip resources.

The CIP-51 is supported by development tools from Silicon Labs and third party vendors. Silicon Labs provides an integrated development environment (IDE) including editor, debugger and programmer. The IDE's debugger and programmer interface to the CIP-51 via the C2 interface to provide fast and efficient in-system device programming and debugging. Third party macro assemblers and C compilers are also available.

### 11.3.2 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 much faster than that of the standard 8051.

All instruction timing on the CIP-51 controller is based directly on the core clock timing. This is in contrast to many other 8-bit architectures, where a distinction is made between machine cycles and clock cycles, with machine cycles taking multiple core 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. The following table summarizes the instruction set, including the mnemonic, number of bytes, and number of clock cycles for each instruction.

**Table 11.2. CIP-51 Instruction Set Summary**

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| Arithmetic Operations | | | |
| ADD A, Rn | Add register to A | 1 | 1 |
| ADD A, direct | Add direct byte to A | 2 | 2 |
| ADD A, @Ri | Add indirect RAM to A | 1 | 2 |
| ADD A, #data | Add immediate to A | 2 | 2 |
| ADDC A, Rn | Add register to A with carry | 1 | 1 |
| ADDC A, direct | Add direct byte to A with carry | 2 | 2 |
| ADDC A, @Ri | Add indirect RAM to A with carry | 1 | 2 |
| ADDC A, #data | Add immediate to A with carry | 2 | 2 |
| SUBB A, Rn | Subtract register from A with borrow | 1 | 1 |
| SUBB A, direct | Subtract direct byte from A with borrow | 2 | 2 |
| SUBB A, @Ri | Subtract indirect RAM from A with borrow | 1 | 2 |
| SUBB A, #data | Subtract immediate from A with borrow | 2 | 2 |
| INC A | Increment A | 1 | 1 |
| INC Rn | Increment register | 1 | 1 |
| INC direct | Increment direct byte | 2 | 2 |
| INC @Ri | Increment indirect RAM | 1 | 2 |
| DEC A | Decrement A | 1 | 1 |
| DEC Rn | Decrement register | 1 | 1 |
| DEC direct | Decrement direct byte | 2 | 2 |
| DEC @Ri | Decrement indirect RAM | 1 | 2 |
| INC DPTR | Increment Data Pointer | 1 | 1 |
| MUL AB | Multiply A and B | 1 | 4 |
| DIV AB | Divide A by B | 1 | 8 |
| DA A | Decimal adjust A | 1 | 1 |
| Logical Operations | | | |
| ANL A, Rn | AND Register to A | 1 | 1 |
| ANL A, direct | AND direct byte to A | 2 | 2 |

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| ANL A, @Ri | AND indirect RAM to A | 1 | 2 |
| ANL A, #data | AND immediate to A | 2 | 2 |
| ANL direct, A | AND A to direct byte | 2 | 2 |
| ANL direct, #data | AND immediate to direct byte | 3 | 3 |
| ORL A, Rn | OR Register to A | 1 | 1 |
| ORL A, direct | OR direct byte to A | 2 | 2 |
| ORL A, @Ri | OR indirect RAM to A | 1 | 2 |
| ORL A, #data | OR immediate to A | 2 | 2 |
| ORL direct, A | OR A to direct byte | 2 | 2 |
| ORL direct, #data | OR immediate to direct byte | 3 | 3 |
| XRL A, Rn | Exclusive-OR Register to A | 1 | 1 |
| XRL A, direct | Exclusive-OR direct byte to A | 2 | 2 |
| XRL A, @Ri | Exclusive-OR indirect RAM to A | 1 | 2 |
| XRL A, #data | Exclusive-OR immediate to A | 2 | 2 |
| XRL direct, A | Exclusive-OR A to direct byte | 2 | 2 |
| XRL direct, #data | Exclusive-OR immediate to direct byte | 3 | 3 |
| CLR A | Clear A | 1 | 1 |
| CPL A | Complement A | 1 | 1 |
| RL A | Rotate A left | 1 | 1 |
| RLC A | Rotate A left through Carry | 1 | 1 |
| RR A | Rotate A right | 1 | 1 |
| RRC A | Rotate A right through Carry | 1 | 1 |
| SWAP A | Swap nibbles of A | 1 | 1 |
| Data Transfer | | | |
| MOV A, Rn | Move Register to A | 1 | 1 |
| MOV A, direct | Move direct byte to A | 2 | 2 |
| MOV A, @Ri | Move indirect RAM to A | 1 | 2 |
| MOV A, #data | Move immediate to A | 2 | 2 |
| MOV Rn, A | Move A to Register | 1 | 1 |
| MOV Rn, direct | Move direct byte to Register | 2 | 2 |
| MOV Rn, #data | Move immediate to Register | 2 | 2 |
| MOV direct, A | Move A to direct byte | 2 | 2 |
| MOV direct, Rn | Move Register to direct byte | 2 | 2 |
| MOV direct, direct | Move direct byte to direct byte | 3 | 3 |
| MOV direct, @Ri | Move indirect RAM to direct byte | 2 | 2 |
| MOV direct, #data | Move immediate to direct byte | 3 | 3 |
| MOV @Ri, A | Move A to indirect RAM | 1 | 2 |

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| MOV @Ri, direct | Move direct byte to indirect RAM | 2 | 2 |
| MOV @Ri, #data | Move immediate to indirect RAM | 2 | 2 |
| MOV DPTR, #data16 | Load DPTR with 16-bit constant | 3 | 3 |
| MOVC A, @A+DPTR | Move code byte relative DPTR to A | 1 | 3 |
| MOVC A, @A+PC | Move code byte relative PC to A | 1 | 3 |
| MOVX A, @Ri | Move external data (8-bit address) to A | 1 | 3 |
| MOVX @Ri, A | Move A to external data (8-bit address) | 1 | 3 |
| MOVX A, @DPTR | Move external data (16-bit address) to A | 1 | 3 |
| MOVX @DPTR, A | Move A to external data (16-bit address) | 1 | 3 |
| PUSH direct | Push direct byte onto stack | 2 | 2 |
| POP direct | Pop direct byte from stack | 2 | 2 |
| XCH A, Rn | Exchange Register with A | 1 | 1 |
| XCH A, direct | Exchange direct byte with A | 2 | 2 |
| XCH A, @Ri | Exchange indirect RAM with A | 1 | 2 |
| XCHD A, @Ri | Exchange low nibble of indirect RAM with A | 1 | 2 |
| Boolean Manipulation | | | |
| CLR C | Clear Carry | 1 | 1 |
| CLR bit | Clear direct bit | 2 | 2 |
| SETB C | Set Carry | 1 | 1 |
| SETB bit | Set direct bit | 2 | 2 |
| CPL C | Complement Carry | 1 | 1 |
| CPL bit | Complement direct bit | 2 | 2 |
| ANL C, bit | AND direct bit to Carry | 2 | 2 |
| ANL C, /bit | AND complement of direct bit to Carry | 2 | 2 |
| ORL C, bit | OR direct bit to carry | 2 | 2 |
| ORL C, /bit | OR complement of direct bit to Carry | 2 | 2 |
| MOV C, bit | Move direct bit to Carry | 2 | 2 |
| MOV bit, C | Move Carry to direct bit | 2 | 2 |
| JC rel | Jump if Carry is set | 2 | 2 or 3 |
| JNC rel | Jump if Carry is not set | 2 | 2 or 3 |
| JB bit, rel | Jump if direct bit is set | 3 | 3 or 4 |
| JNB bit, rel | Jump if direct bit is not set | 3 | 3 or 4 |
| JBC bit, rel | Jump if direct bit is set and clear bit | 3 | 3 or 4 |
| Program Branching | | | |
| ACALL addr11 | Absolute subroutine call | 2 | 3 |
| LCALL addr16 | Long subroutine call | 3 | 4 |
| RET | Return from subroutine | 1 | 5 |

| Mnemonic | Description | Bytes | Clock Cycles |
|---|---|---|---|
| RETI | Return from interrupt | 1 | 5 |
| AJMP addr11 | Absolute jump | 2 | 3 |
| LJMP addr16 | Long jump | 3 | 4 |
| SJMP rel | Short jump (relative address) | 2 | 3 |
| JMP @A+DPTR | Jump indirect relative to DPTR | 1 | 3 |
| JZ rel | Jump if A equals zero | 2 | 2 or 3 |
| JNZ rel | Jump if A does not equal zero | 2 | 2 or 3 |
| CJNE A, direct, rel | Compare direct byte to A and jump if not equal | 3 | 3 or 4 |
| CJNE A, #data, rel | Compare immediate to A and jump if not equal | 3 | 3 or 4 |
| CJNE Rn, #data, rel | Compare immediate to Register and jump if not equal | 3 | 3 or 4 |
| CJNE @Ri, #data, rel | Compare immediate to indirect and jump if not equal | 3 | 4 or 5 |
| DJNZ Rn, rel | Decrement Register and jump if not zero | 2 | 2 or 3 |
| DJNZ direct, rel | Decrement direct byte and jump if not zero | 3 | 3 or 4 |
| NOP | No operation | 1 | 1 |

Notes:

- **Rn**: Register R0–R7 of the currently selected register bank.
- **@Ri**: Data RAM location addressed indirectly through R0 or R1.
- **rel**: 8-bit, signed (twos complement) offset relative to the first byte of the following instruction. Used by SJMP and all conditional jumps.
- **direct**: 8-bit internal data location's address. This could be a direct-access Data RAM location (0x00–0x7F) or an SFR (0x80–0xFF).
- **#data**: 8-bit constant.
- **#data16**: 16-bit constant.
- **bit**: Direct-accessed bit in Data RAM or SFR.
- **addr11**: 11-bit destination address used by ACALL and AJMP. The destination must be within the same 2 KB page of program memory as the first byte of the following instruction.
- **addr16**: 16-bit destination address used by LCALL and LJMP. The destination may be anywhere within the 8 KB program memory space.
- There is one unused opcode (0xA5) that performs the same function as NOP. All mnemonics copyrighted © Intel Corporation 1980.

## 11.4 CPU Core Registers

### 11.4.1 DPL: Data Pointer Low

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DPL | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0x82 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | DPL | 0x00 | RW | **Data Pointer Low.**<br>The DPL register is the low byte of the 16-bit DPTR. DPTR is used to access indirectly addressed flash memory or XRAM. |

### 11.4.2 DPH: Data Pointer High

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DPH | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0x83 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | DPH | 0x00 | RW | **Data Pointer High.**<br>The DPH register is the high byte of the 16-bit DPTR. DPTR is used to access indirectly addressed flash memory or XRAM. |

### 11.4.3 SP: Stack Pointer

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SP | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x07 | | | | | | | |
| SFR Page = ALL; SFR Address: 0x81 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | SP | 0x07 | RW | **Stack Pointer.**<br>The Stack Pointer holds the location of the top of the stack. The stack pointer is incremented before every PUSH operation. The SP register defaults to 0x07 after reset. |

### 11.4.4 ACC: Accumulator

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ACC | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0xE0 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ACC | 0x00 | RW | **Accumulator.**<br>This register is the accumulator for arithmetic operations. |

### 11.4.5 B: B Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0xF0 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | B | 0x00 | RW | **B Register.**<br>This register serves as a second accumulator for certain arithmetic operations. |

### 11.4.6 PSW: Program Status Word

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CY | AC | F0 | RS | | OV | F1 | PARITY |
| Access | RW | RW | RW | RW | | RW | RW | R |
| Reset | 0 | 0 | 0 | 0x0 | | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0xD0 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | CY | 0 | RW | **Carry Flag.**<br>This bit is set when the last arithmetic operation resulted in a carry (addition) or a borrow (subtraction). It is cleared to logic 0 by all other arithmetic operations. |
| 6 | AC | 0 | RW | **Auxiliary Carry Flag.**<br>This bit is set when the last arithmetic operation resulted in a carry into (addition) or a borrow from (subtraction) the high order nibble. It is cleared to logic 0 by all other arithmetic operations. |
| 5 | F0 | 0 | RW | **User Flag 0.**<br>This is a bit-addressable, general purpose flag for use under firmware control. |
| 4:3 | RS | 0x0 | RW | **Register Bank Select.**<br>These bits select which register bank is used during register accesses. |

| Value | Name | Description |
|---|---|---|
| 0x0 | BANK0 | Bank 0, Addresses 0x00-0x07 |
| 0x1 | BANK1 | Bank 1, Addresses 0x08-0x0F |
| 0x2 | BANK2 | Bank 2, Addresses 0x10-0x17 |
| 0x3 | BANK3 | Bank 3, Addresses 0x18-0x1F |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2 | OV | 0 | RW | **Overflow Flag.**<br>This bit is set to 1 under the following circumstances:<br>1. An ADD, ADDC, or SUBB instruction causes a sign-change overflow.<br>2. A MUL instruction results in an overflow (result is greater than 255).<br>3. A DIV instruction causes a divide-by-zero condition.<br>The OV bit is cleared to 0 by the ADD, ADDC, SUBB, MUL, and DIV instructions in all other cases. |
| 1 | F1 | 0 | RW | **User Flag 1.**<br>This is a bit-addressable, general purpose flag for use under firmware control. |
| 0 | PARITY | 0 | R | **Parity Flag.**<br>This bit is set to logic 1 if the sum of the eight bits in the accumulator is odd and cleared if the sum is even. |

# 12. Port I/O, Crossbar, External Interrupts, and Port Match

## 12.1 Introduction

Digital and analog resources are externally available on the device's multi-purpose I/O pins. Port pins P0.0-P1.7 can be defined as general-purpose I/O (GPIO), assigned to one of the internal digital resources through the crossbar or dedicated channels, or assigned to an analog function. Port pin P2.7 can be used as GPIO. Additionally, the C2 Interface Data signal (C2D) is shared with P2.7.

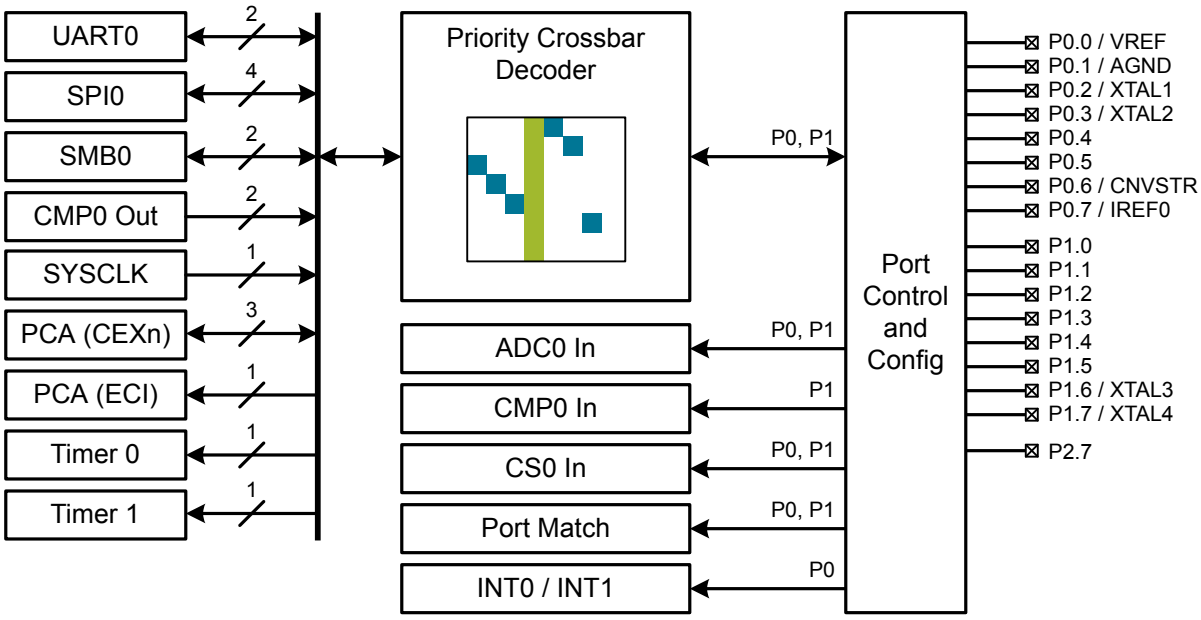



**Figure 12.1. Port I/O Block Diagram**

## 12.2 Features

- Up to 17 multi-functions I/O pins, supporting digital and analog functions.
- Flexible priority crossbar decoder for digital peripheral assignment.
- Two drive strength settings for each pin.
- Two direct-pin interrupt sources with dedicated interrupt vectors (INT0 and INT1).
- Up to 16 direct-pin interrupt sources with shared interrupt vector (Port Match).

## 12.3 Functional Description

### 12.3.1 Port I/O Modes of Operation

Port pins are configured by firmware as digital or analog I/O using the special function registers. Port I/O initialization consists of the following general steps:

1. Select the input mode (analog or digital) for all port pins, using the Port Input Mode register (PnMDIN).
2. Select the output mode (open-drain or push-pull) for all port pins, using the Port Output Mode register (PnMDOUT).
3. Select any pins to be skipped by the I/O crossbar using the Port Skip registers (PnSKIP).
4. Assign port pins to desired peripherals.
5. Enable the crossbar (XBARE = 1).

A diagram of the port I/O cell is shown in the following figure.

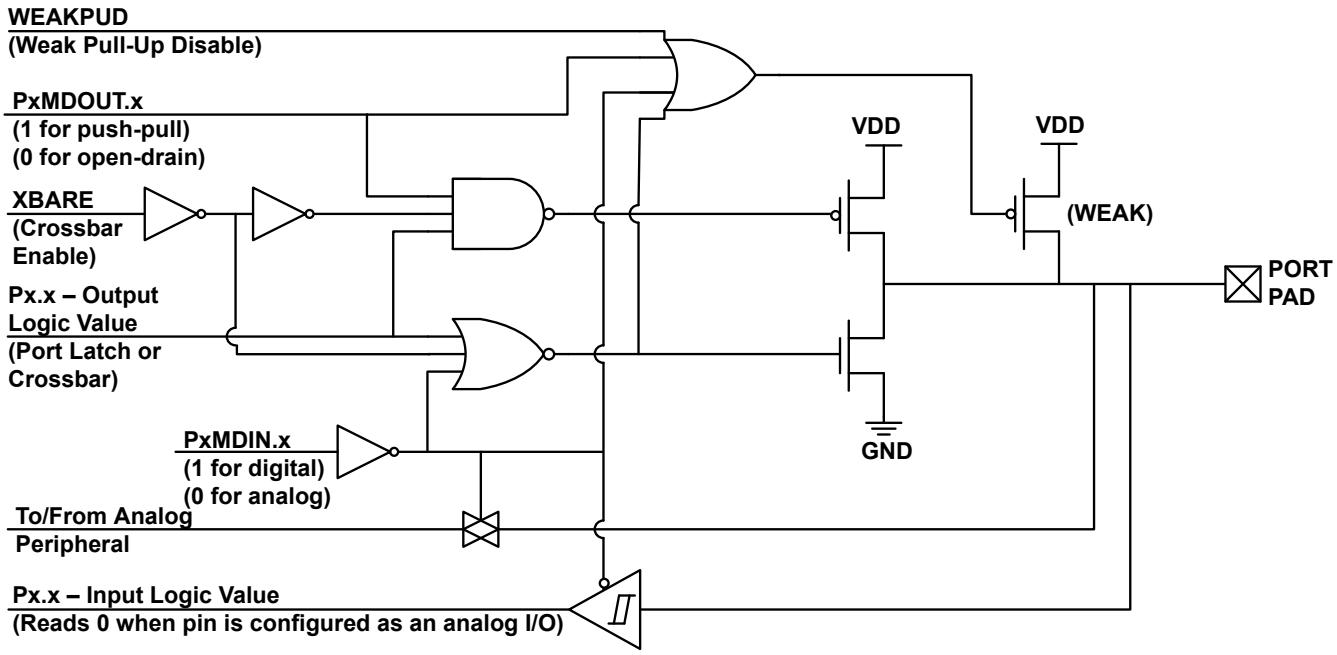



**Figure 12.2. Port I/O Cell Block Diagram**

#### Configuring Port Pins For Analog Modes

Any pins to be used for analog functions should be configured for analog mode. When a pin is configured for analog I/O, its weak pull-up, digital driver, and digital receiver are disabled. This saves power by eliminating crowbar current, and reduces noise on the analog input. Pins configured as digital inputs may still be used by analog peripherals; however this practice is not recommended. Port pins configured for analog functions will always read back a value of 0 in the corresponding Pn Port Latch register. To configure a pin as analog, the following steps should be taken:

1. Clear the bit associated with the pin in the PnMDIN register to 0. This selects analog mode for the pin.
2. Set the bit associated with the pin in the Pn register to 1.
3. Skip the bit associated with the pin in the PnSKIP register to ensure the crossbar does not attempt to assign a function to the pin.

**Configuring Port Pins For Digital Modes**

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

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

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

To configure a pin as a digital input:

1. Set the bit associated with the pin in the PnMDIN register to 1. This selects digital mode for the pin.
2. lear the bit associated with the pin in the PnMDOUT register to 0. This configures the pin as open-drain.
3. Set the bit associated with the pin in the Pn register to 1. This tells the output driver to “drive” logic high. Because the pin is configured as open-drain, the high-side driver is disabled, and the pin may be used as an input.

Open-drain outputs are configured exactly as digital inputs. The pin may be driven low by an assigned peripheral, or by writing 0 to the associated bit in the Pn register if the signal is a GPIO.

To configure a pin as a digital, push-pull output:

1. Set the bit associated with the pin in the PnMDIN register to 1. This selects digital mode for the pin.
2. Set the bit associated with the pin in the PnMDOUT register to 1. This configures the pin as push-pull.

If a digital pin is to be used as a general-purpose I/O, or with a digital function that is not part of the crossbar, the bit associated with the pin in the PnSKIP register can be set to 1 to ensure the crossbar does not attempt to assign a function to the pin. The crossbar must be enabled to use port pins as standard port I/O in output mode. Port output drivers of all I/O pins are disabled whenever the crossbar is disabled.

#### 12.3.1.1 Pin Drive Strength

Pin drive strength can be controlled on a pin-by-pin basis using the PnDRV registers. Each pin has a bit in the corresponding PnDRV register to select the high or low drive strengh setting. By default, all port pins are configured for low drive strength.

### 12.3.2 Analog and Digital Functions

#### 12.3.2.1 Port I/O Analog Assignments

The following table displays the potential mapping of port I/O to each analog function.

**Table 12.1. Port I/O Assignment for Analog Functions**

| Analog Function | Potentially Assignable Port Pins | SFR(s) Used For Assignment |
|---|---|---|
| ADC Input | P0.0 – P0.7, P1.2 – P1.4 | ADC0MX, PnSKIP, PnMDIN |
| Comparator 0 Input | P1.0, P0.1 | CMP0MX, PnSKIP, PnMDIN |
| Voltage Reference (VREF) | P0.0 | REF0CN, PnSKIP, PnMDIN |
| Reference Ground (AGND) | P0.1 | REF0CN, PnSKIP, PnMDIN |
| Current Refernence (IREF0) | P0.7 | IREF0CN0, PnSKIP, PnMDIN |
| External Oscillator Input (XTAL2) | P0.2 | HFO0CN, PnSKIP, PnMDIN |
| External Oscillator Output (XTAL1) | P0.3 | HFO0CN, PnSKIP, PnMDIN |
| RTC Oscillator Input (XTAL3) | P1.6 | RTC0CN0, PnSKIP, PnMDIN |
| RTC Oscillator Output (XTAL4) | P1.7 | RTC0CN0, PnSKIP, PnMDIN |

#### 12.3.2.2 Port I/O Digital Assignments

The following table displays the potential mapping of port I/O to each digital function.

**Table 12.2. Port I/O Assignment for Digital Functions**

| Digital Function | Potentially Assignable Port Pins | SFR(s) Used For Assignment |
|---|---|---|
| UART0, SPI0, SMB0, CP0, CP0A, SYSCLK, PCA0 (CEX0-2 and ECI), T0, T1 | Any port pin available for assignment by the crossbar. This includes P0.0–P1.7 pins which have their PnSKIP bit set to 0. The crossbar will always assign UART0 pins to P0.4 and P0.5. | XBR0, XBR1, XBR2 |
| External Interrupt 0, External Interrupt 1 | P0.0–P0.7 | IT01CF |
| Conversion Start (CNVSTR) | P0.6 | ADC0CN0 |
| Port Match | P0.0–P1.7 | P0MASK, P0MAT, P1MASK, P1MAT |
| Any pin used for GPIO | P0.0–P1.7, P2.7 | P0SKIP, P1SKIP |

### 12.3.3 Priority Crossbar Decoder

The priority crossbar decoder assigns a priority to each I/O function, starting at the top with UART0. The XBRn registers are used to control which crossbar resources are assigned to physical I/O port pins.

When a digital resource is selected, the least-significant unassigned port pin is assigned to that resource (excluding UART0, which is always assigned to dedicated pins). If a port pin is assigned, the crossbar skips that pin when assigning the next selected resource. Additionally, the the PnSKIP registers allow software to skip port pins that are to be used for analog functions, dedicated digital functions, or GPIO. If a port pin is to be used by a function which is not assigned through the crossbar, its corresponding PnSKIP bit should be set to 1 in most cases. The crossbar skips these pins as if they were already assigned, and moves to the next unassigned pin.

It is possible for crossbar-assigned peripherals and dedicated functions to coexist on the same pin. For example, the port match function could be configured to watch for a falling edge on a UART RX line and generate an interrupt or wake up the device from a low-power state. However, if two functions share the same pin, the crossbar will have control over the output characteristics of that pin and the dedicated function will only have input access. Likewise, it is possible for firmware to read the logic state of any digital I/O pin assigned to a crossbar peripheral, but the output state cannot be directly modified.

Figure 12.3 Crossbar Priority Decoder Example Assignments on page 100 shows an example of the resulting pin assignments of the device with UART0 and SPI0 enabled and P0.3 skipped (P0SKIP = 0x08). UART0 is the highest priority and it will be assigned first. The UART0 pins can only appear at fixed locations (in this example, P0.4 and P0.5), so it occupies those pins. The next-highest enabled peripheral is SPI0. P0.0, P0.1 and P0.2 are free, so SPI0 takes these three pins. The fourth pin, NSS, is routed to P0.6 because P0.3 is skipped and P0.4 and P0.5 are already occupied by the UART. Any other pins on the device are available for use as general-purpose digital I/O or analog functions.

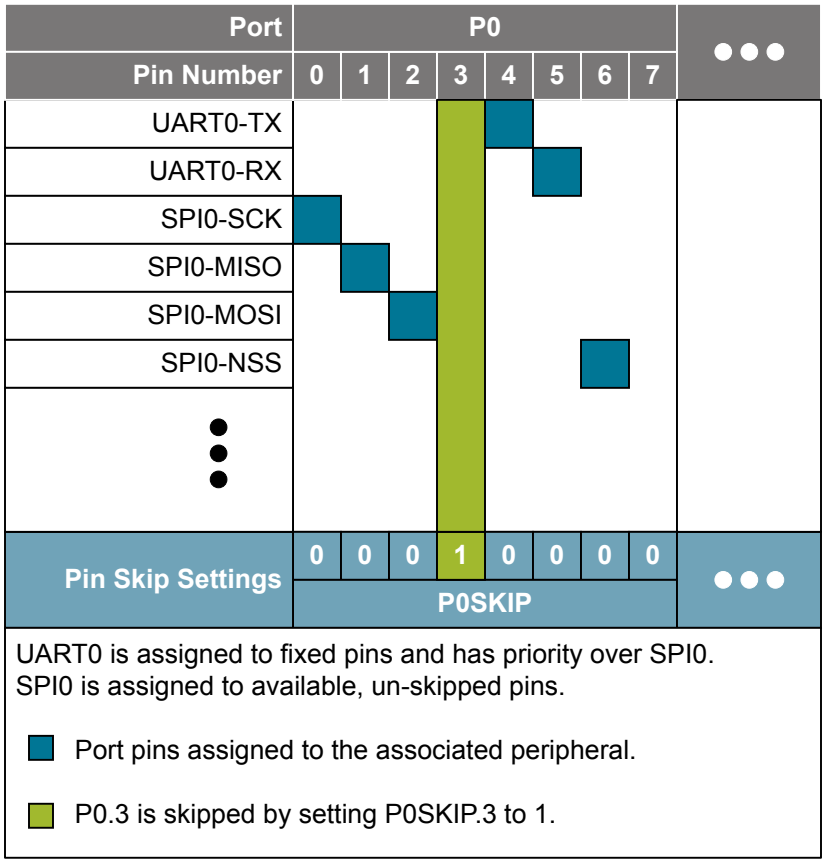



**Figure 12.3. Crossbar Priority Decoder Example Assignments**

#### 12.3.3.1 Crossbar Functional Map

Figure 12.4 Full Crossbar Map on page 101 shows all of the potential peripheral-to-pin assignments available to the crossbar. Note that this does not mean any peripheral can always be assigned to the highlighted pins. The actual pin assignments are determined by the priority of the enabled peripherals.

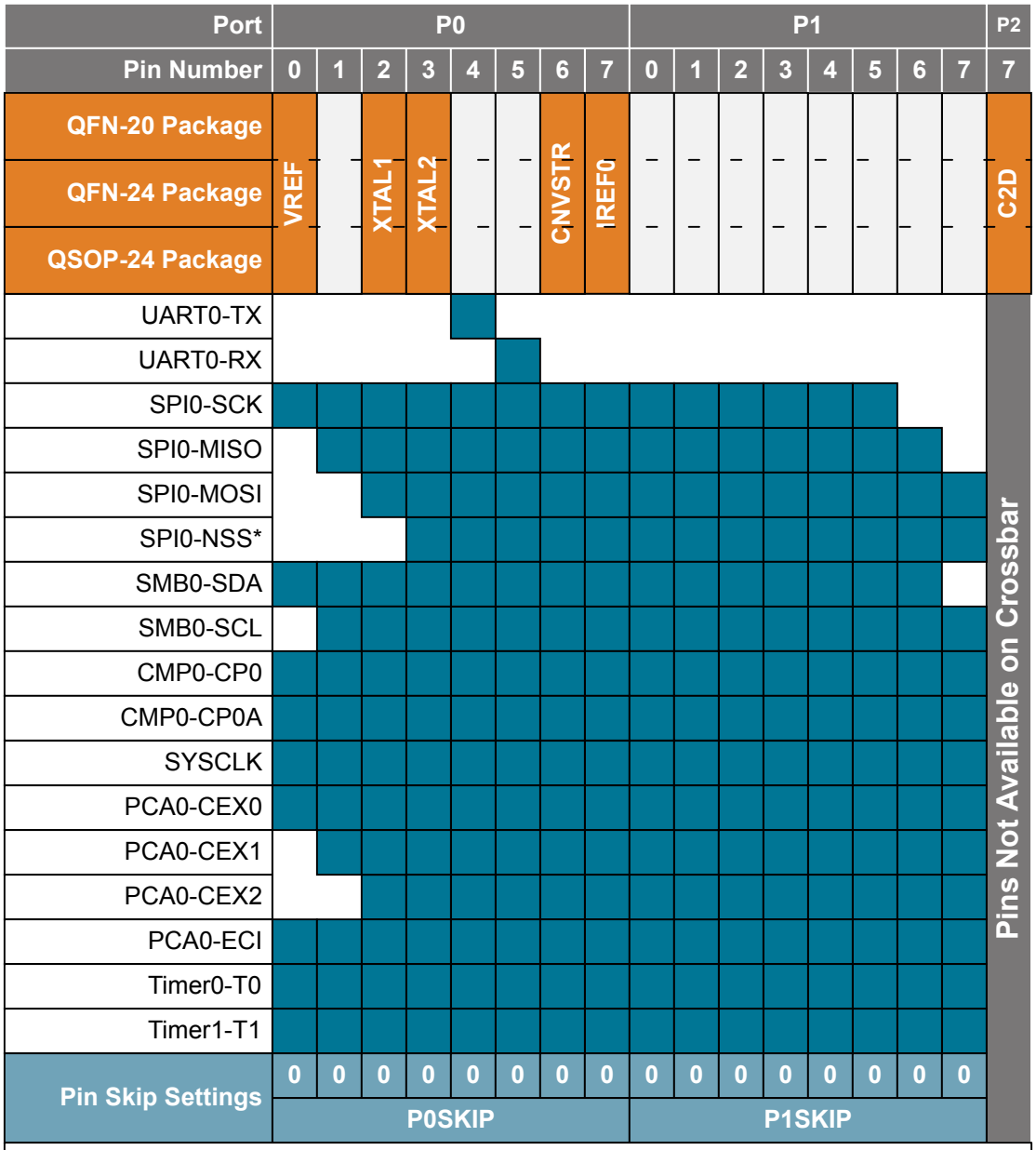



**Figure 12.4. Full Crossbar Map**

### 12.3.4 INT0 and INT1

Two direct-pin digital interrupt sources (INT0 and INT1) are included, which can be routed to port 0 pins. Additional I/O interrupts are available through the port match function. As is the case on a standard 8051 architecture, certain controls for these two interrupt sources are available in the Timer0/1 registers. Extensions to these controls which provide additional functionality are available in the IT01CF register. INT0 and INT1 are configurable as active high or low, edge- or level-sensitive. The IN0PL and IN1PL bits in the IT01CF register select active high or active low; the IT0 and IT1 bits in TCON select level- or edge-sensitive. The table below lists the possible configurations.

**Table 12.3. INT0/INT1 configuration**

| IT0 or IT1 | IN0PL or IN1PL | INT0 or INT1 Interrupt |
|---|---|---|
| 1 | 0 | Interrupt on falling edge |
| 1 | 1 | Interrupt on rising edge |
| 0 | 0 | Interrupt on low level |
| 0 | 1 | Interrupt on high level |

INT0 and INT1 are assigned to port pins as defined in the IT01CF register. INT0 and INT1 port pin assignments are independent of any crossbar assignments, and may be assigned to pins used by crossbar peripherals. INT0 and INT1 will monitor their assigned port pins without disturbing the peripheral that was assigned the port pin via the crossbar. To assign a port pin only to INT0 and/or INT1, configure the crossbar to skip the selected pin(s).

IE0 and IE1 in the TCON register serve as the interrupt-pending flags for the INT0 and INT1 external interrupts, respectively. If an INT0 or INT1 external interrupt is configured as edge-sensitive, the corresponding interrupt pending flag is automatically cleared by the hardware when the CPU vectors to the ISR. When configured as level sensitive, the interrupt-pending flag remains logic 1 while the input is active as defined by the corresponding polarity bit (IN0PL or IN1PL); the flag remains logic 0 while the input is inactive. The external interrupt source must hold the input active until the interrupt request is recognized. It must then deactivate the interrupt request before execution of the ISR completes or another interrupt request will be generated.

### 12.3.5 Port Match

Port match functionality allows system events to be triggered by a logic value change on one or more port I/O pins. A software controlled value stored in the PnMATCH registers specifies the expected or normal logic values of the associated port pins (for example, P0MATCH.0 would correspond to P0.0). A port mismatch event occurs if the logic levels of the port's input pins no longer match the software controlled value. This allows software to be notified if a certain change or pattern occurs on the input pins regardless of the XBRn settings.

The PnMASK registers can be used to individually select which pins should be compared against the PnMATCH registers. A port mismatch event is generated if (Pn & PnMASK) does not equal (PnMATCH & PnMASK) for all ports with a PnMAT and PnMASK register.

A port mismatch event may be used to generate an interrupt or wake the device from low power modes. See the interrupts and power options chapters for more details on interrupt and wake-up sources.

### 12.3.6 Direct Port I/O Access (Read/Write)

All port I/O are accessed through corresponding special function registers. When writing to a port, the value written to the SFR is latched to maintain the output data value at each pin. When reading, the logic levels of the port's input pins are returned regardless of the XBRn settings (i.e., even when the pin is assigned to another signal by the crossbar, the port register can always read its corresponding port I/O pin). The exception to this is the execution of the read-modify-write instructions that target a Port Latch register as the destination. The read-modify-write instructions when operating on a port SFR are the following: ANL, ORL, XRL, JBC, CPL, INC, DEC, DJNZ and MOV, CLR or SETB, when the destination is an individual bit in a port SFR. For these instructions, the value of the latch register (not the pin) is read, modified, and written back to the SFR.

## 12.4 Port I/O Control Registers

### 12.4.1 XBR0: Port I/O Crossbar 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | CP0AE | CP0E | SYSCKE | SMB0E | SPI0E | URT0E |
| Access | RW | | RW | RW | RW | RW | RW | RW |
| Reset | 0x0 | | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xE1 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:6* | *Reserved* | *Must write reset value.* | | |
| 5 | CP0AE | 0 | RW | **Comparator0 Asynchronous Output Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Asynchronous CP0 unavailable at Port pin. |
| | 1 | ENABLED | | Asynchronous CP0 routed to Port pin. |
| 4 | CP0E | 0 | RW | **Comparator0 Output Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | CP0 unavailable at Port pin. |
| | 1 | ENABLED | | CP0 routed to Port pin. |
| 3 | SYSCKE | 0 | RW | **SYSCLK Output Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | SYSCLK unavailable at Port pin. |
| | 1 | ENABLED | | SYSCLK output routed to Port pin. |
| 2 | SMB0E | 0 | RW | **SMB0 I/O Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | SMBus 0 I/O unavailable at Port pins. |
| | 1 | ENABLED | | SMBus 0 I/O routed to Port pins. |
| 1 | SPI0E | 0 | RW | **SPI I/O Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | SPI I/O unavailable at Port pins. |
| | 1 | ENABLED | | SPI I/O routed to Port pins. The SPI can be assigned either 3 or 4 GPIO pins. |
| 0 | URT0E | 0 | RW | **UART I/O Output Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | UART I/O unavailable at Port pin. |
| | 1 | ENABLED | | UART TX, RX routed to Port pins P0.4 and P0.5. |

### 12.4.2 XBR1: Port I/O Crossbar 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | T1E | T0E | ECIE | PCA0ME | | |
| Access | RW | | RW | RW | RW | RW | | |
| Reset | 0x0 | | 0 | 0 | 0 | 0x0 | | |
| SFR Page = 0x0; SFR Address: 0xE2 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:6* | *Reserved* | *Must write reset value.* | | |
| 5 | T1E | 0 | RW | **T1 Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | T1 unavailable at Port pin. |
| | 1 | ENABLED | | T1 routed to Port pin. |
| 4 | T0E | 0 | RW | **T0 Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | T0 unavailable at Port pin. |
| | 1 | ENABLED | | T0 routed to Port pin. |
| 3 | ECIE | 0 | RW | **PCA0 External Counter Input Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | ECI unavailable at Port pin. |
| | 1 | ENABLED | | ECI routed to Port pin. |
| 2:0 | PCA0ME | 0x0 | RW | **PCA Module I/O Enable.** |
| | Value | Name | | Description |
| | 0x0 | DISABLED | | All PCA I/O unavailable at Port pins. |
| | 0x1 | CEX0 | | CEX0 routed to Port pin. |
| | 0x2 | CEX0_CEX1 | | CEX0, CEX1 routed to Port pins. |
| | 0x3 | CEX0_CEX1_CEX2 | | CEX0, CEX1, CEX2 routed to Port pins. |

### 12.4.3 XBR2: Port I/O Crossbar 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | WEAKPUD | XBARE | Reserved | | | | | |
| Access | RW | RW | R | | | | | |
| Reset | 0 | 0 | 0x00 | | | | | |
| SFR Page = 0x0; SFR Address: 0xE3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | WEAKPUD | 0 | RW | **Port I/O Weak Pullup Disable.** |
| | Value | Name | | Description |
| | 0 | PULL_UPS_ENABLED | | Weak Pullups enabled (except for Ports whose I/O are configured for analog mode). |
| | 1 | PULL_UPS_DISABLED | | Weak Pullups disabled. |
| 6 | XBARE | 0 | RW | **Crossbar Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Crossbar disabled. |
| | 1 | ENABLED | | Crossbar enabled. |
| *5:0* | *Reserved* | *Must write reset value.* | | |
| The Crossbar must be enabled (XBARE = 1) to use any port pin as a digital output. | | | | |

### 12.4.4 P0MASK: Port 0 Mask

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xC7 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 0 Bit 7 Mask Value.** |
| | Value | Name | | Description |
| | 0 | IGNORED | | P0.7 pin logic value is ignored and will not cause a port mismatch event. |
| | 1 | COMPARED | | P0.7 pin logic value is compared to P0MAT.7. |
| 6 | B6 | 0 | RW | **Port 0 Bit 6 Mask Value.** |
| | See bit 7 description | | | |
| 5 | B5 | 0 | RW | **Port 0 Bit 5 Mask Value.** |
| | See bit 7 description | | | |
| 4 | B4 | 0 | RW | **Port 0 Bit 4 Mask Value.** |
| | See bit 7 description | | | |
| 3 | B3 | 0 | RW | **Port 0 Bit 3 Mask Value.** |
| | See bit 7 description | | | |
| 2 | B2 | 0 | RW | **Port 0 Bit 2 Mask Value.** |
| | See bit 7 description | | | |
| 1 | B1 | 0 | RW | **Port 0 Bit 1 Mask Value.** |
| | See bit 7 description | | | |
| 0 | B0 | 0 | RW | **Port 0 Bit 0 Mask Value.** |
| | See bit 7 description | | | |

### 12.4.5 P0MAT: Port 0 Match

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SFR Page = 0x0; SFR Address: 0xD7 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 1 | RW | **Port 0 Bit 7 Match Value.** |
| | Value | Name | | Description |
| | 0 | LOW | | P0.7 pin logic value is compared with logic LOW. |
| | 1 | HIGH | | P0.7 pin logic value is compared with logic HIGH. |
| 6 | B6 | 1 | RW | **Port 0 Bit 6 Match Value.** |
| | See bit 7 description | | | |
| 5 | B5 | 1 | RW | **Port 0 Bit 5 Match Value.** |
| | See bit 7 description | | | |
| 4 | B4 | 1 | RW | **Port 0 Bit 4 Match Value.** |
| | See bit 7 description | | | |
| 3 | B3 | 1 | RW | **Port 0 Bit 3 Match Value.** |
| | See bit 7 description | | | |
| 2 | B2 | 1 | RW | **Port 0 Bit 2 Match Value.** |
| | See bit 7 description | | | |
| 1 | B1 | 1 | RW | **Port 0 Bit 1 Match Value.** |
| | See bit 7 description | | | |
| 0 | B0 | 1 | RW | **Port 0 Bit 0 Match Value.** |
| | See bit 7 description | | | |

### 12.4.6 P0: Port 0 Pin Latch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Page = ALL; SFR Address: 0x80 (bit-addressable)

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 1 | RW | **Port 0 Bit 7 Latch.** |
| | Value | Name | | Description |
| | 0 | LOW | | P0.7 is low. Set P0.7 to drive low. |
| | 1 | HIGH | | P0.7 is high. Set P0.7 to drive or float high. |
| 6 | B6 | 1 | RW | **Port 0 Bit 6 Latch.** |
| | See bit 7 description | | | |
| 5 | B5 | 1 | RW | **Port 0 Bit 5 Latch.** |
| | See bit 7 description | | | |
| 4 | B4 | 1 | RW | **Port 0 Bit 4 Latch.** |
| | See bit 7 description | | | |
| 3 | B3 | 1 | RW | **Port 0 Bit 3 Latch.** |
| | See bit 7 description | | | |
| 2 | B2 | 1 | RW | **Port 0 Bit 2 Latch.** |
| | See bit 7 description | | | |
| 1 | B1 | 1 | RW | **Port 0 Bit 1 Latch.** |
| | See bit 7 description | | | |
| 0 | B0 | 1 | RW | **Port 0 Bit 0 Latch.** |
| | See bit 7 description | | | |

Writing this register sets the port latch logic value for the associated I/O pins configured as digital I/O.

Reading this register returns the logic value at the pin, regardless if it is configured as output or input.

### 12.4.7 P0MDIN: Port 0 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SFR Page = 0x0; SFR Address: 0xF1 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 1 | RW | **Port 0 Bit 7 Input Mode.** |
| | Value | Name | | Description |
| | 0 | ANALOG | | P0.7 pin is configured for analog mode. |
| | 1 | DIGITAL | | P0.7 pin is configured for digital mode. |
| 6 | B6 | 1 | RW | **Port 0 Bit 6 Input Mode.** |
| | See bit 7 description | | | |
| 5 | B5 | 1 | RW | **Port 0 Bit 5 Input Mode.** |
| | See bit 7 description | | | |
| 4 | B4 | 1 | RW | **Port 0 Bit 4 Input Mode.** |
| | See bit 7 description | | | |
| 3 | B3 | 1 | RW | **Port 0 Bit 3 Input Mode.** |
| | See bit 7 description | | | |
| 2 | B2 | 1 | RW | **Port 0 Bit 2 Input Mode.** |
| | See bit 7 description | | | |
| 1 | B1 | 1 | RW | **Port 0 Bit 1 Input Mode.** |
| | See bit 7 description | | | |
| 0 | B0 | 1 | RW | **Port 0 Bit 0 Input Mode.** |
| | See bit 7 description | | | |
| Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled. | | | | |

### 12.4.8 P0MDOUT: Port 0 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xA4 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 0 Bit 7 Output Mode.** |
| | Value | Name | | Description |
| | 0 | OPEN_DRAIN | | P0.7 output is open-drain. |
| | 1 | PUSH_PULL | | P0.7 output is push-pull. |
| 6 | B6 | 0 | RW | **Port 0 Bit 6 Output Mode.** |
| | See bit 7 description | | | |
| 5 | B5 | 0 | RW | **Port 0 Bit 5 Output Mode.** |
| | See bit 7 description | | | |
| 4 | B4 | 0 | RW | **Port 0 Bit 4 Output Mode.** |
| | See bit 7 description | | | |
| 3 | B3 | 0 | RW | **Port 0 Bit 3 Output Mode.** |
| | See bit 7 description | | | |
| 2 | B2 | 0 | RW | **Port 0 Bit 2 Output Mode.** |
| | See bit 7 description | | | |
| 1 | B1 | 0 | RW | **Port 0 Bit 1 Output Mode.** |
| | See bit 7 description | | | |
| 0 | B0 | 0 | RW | **Port 0 Bit 0 Output Mode.** |
| | See bit 7 description | | | |

### 12.4.9 P0SKIP: Port 0 Skip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xD4 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 0 Bit 7 Skip.** |
| | Value | Name | | Description |
| | 0 | NOT_SKIPPED | | P0.7 pin is not skipped by the crossbar. |
| | 1 | SKIPPED | | P0.7 pin is skipped by the crossbar. |
| 6 | B6 | 0 | RW | **Port 0 Bit 6 Skip.** |
| | See bit 7 description | | | |
| 5 | B5 | 0 | RW | **Port 0 Bit 5 Skip.** |
| | See bit 7 description | | | |
| 4 | B4 | 0 | RW | **Port 0 Bit 4 Skip.** |
| | See bit 7 description | | | |
| 3 | B3 | 0 | RW | **Port 0 Bit 3 Skip.** |
| | See bit 7 description | | | |
| 2 | B2 | 0 | RW | **Port 0 Bit 2 Skip.** |
| | See bit 7 description | | | |
| 1 | B1 | 0 | RW | **Port 0 Bit 1 Skip.** |
| | See bit 7 description | | | |
| 0 | B0 | 0 | RW | **Port 0 Bit 0 Skip.** |
| | See bit 7 description | | | |

### 12.4.10 P0DRV: Port 0 Drive Strength

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0xF; SFR Address: 0x99 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 0 Bit 7 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.7 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P0.7 output has high output drive strength. |
| 6 | B6 | 0 | RW | **Port 0 Bit 6 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.6 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P0.6 output has high output drive strength. |
| 5 | B5 | 0 | RW | **Port 0 Bit 5 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.5 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P0.5 output has high output drive strength. |
| 4 | B4 | 0 | RW | **Port 0 Bit 4 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.4 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P0.4 output has high output drive strength. |
| 3 | B3 | 0 | RW | **Port 0 Bit 3 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.3 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P0.3 output has high output drive strength. |
| 2 | B2 | 0 | RW | **Port 0 Bit 2 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.2 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P0.2 output has high output drive strength. |
| 1 | B1 | 0 | RW | **Port 0 Bit 1 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.1 output has low output drive strength. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | 1 | HIGH_DRIVE | | P0.1 output has high output drive strength. |
| 0 | B0 | 0 | RW | **Port 0 Bit 0 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P0.0 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P0.0 output has high output drive strength. |

### 12.4.11 P1MASK: Port 1 Mask

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xBF | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 1 Bit 7 Mask Value.** |
| | Value | Name | | Description |
| | 0 | IGNORED | | P1.7 pin logic value is ignored and will not cause a port mismatch event. |
| | 1 | COMPARED | | P1.7 pin logic value is compared to P1MAT.7. |
| 6 | B6 | 0 | RW | **Port 1 Bit 6 Mask Value.** |
| | See bit 7 description | | | |
| 5 | B5 | 0 | RW | **Port 1 Bit 5 Mask Value.** |
| | See bit 7 description | | | |
| 4 | B4 | 0 | RW | **Port 1 Bit 4 Mask Value.** |
| | See bit 7 description | | | |
| 3 | B3 | 0 | RW | **Port 1 Bit 3 Mask Value.** |
| | See bit 7 description | | | |
| 2 | B2 | 0 | RW | **Port 1 Bit 2 Mask Value.** |
| | See bit 7 description | | | |
| 1 | B1 | 0 | RW | **Port 1 Bit 1 Mask Value.** |
| | See bit 7 description | | | |
| 0 | B0 | 0 | RW | **Port 1 Bit 0 Mask Value.** |
| | See bit 7 description | | | |

### 12.4.12 P1MAT: Port 1 Match

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SFR Page = 0x0; SFR Address: 0xCF | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 1 | RW | **Port 1 Bit 7 Match Value.** |
| | Value | Name | | Description |
| | 0 | LOW | | P1.7 pin logic value is compared with logic LOW. |
| | 1 | HIGH | | P1.7 pin logic value is compared with logic HIGH. |
| 6 | B6 | 1 | RW | **Port 1 Bit 6 Match Value.** |
| | See bit 7 description | | | |
| 5 | B5 | 1 | RW | **Port 1 Bit 5 Match Value.** |
| | See bit 7 description | | | |
| 4 | B4 | 1 | RW | **Port 1 Bit 4 Match Value.** |
| | See bit 7 description | | | |
| 3 | B3 | 1 | RW | **Port 1 Bit 3 Match Value.** |
| | See bit 7 description | | | |
| 2 | B2 | 1 | RW | **Port 1 Bit 2 Match Value.** |
| | See bit 7 description | | | |
| 1 | B1 | 1 | RW | **Port 1 Bit 1 Match Value.** |
| | See bit 7 description | | | |
| 0 | B0 | 1 | RW | **Port 1 Bit 0 Match Value.** |
| | See bit 7 description | | | |

### 12.4.13 P1: Port 1 Pin Latch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

SFR Page = ALL; SFR Address: 0x90 (bit-addressable)

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 1 | RW | **Port 1 Bit 7 Latch.** |
| | Value | Name | | Description |
| | 0 | LOW | | P1.7 is low. Set P1.7 to drive low. |
| | 1 | HIGH | | P1.7 is high. Set P1.7 to drive or float high. |
| 6 | B6 | 1 | RW | **Port 1 Bit 6 Latch.** |
| | See bit 7 description | | | |
| 5 | B5 | 1 | RW | **Port 1 Bit 5 Latch.** |
| | See bit 7 description | | | |
| 4 | B4 | 1 | RW | **Port 1 Bit 4 Latch.** |
| | See bit 7 description | | | |
| 3 | B3 | 1 | RW | **Port 1 Bit 3 Latch.** |
| | See bit 7 description | | | |
| 2 | B2 | 1 | RW | **Port 1 Bit 2 Latch.** |
| | See bit 7 description | | | |
| 1 | B1 | 1 | RW | **Port 1 Bit 1 Latch.** |
| | See bit 7 description | | | |
| 0 | B0 | 1 | RW | **Port 1 Bit 0 Latch.** |
| | See bit 7 description | | | |

Writing this register sets the port latch logic value for the associated I/O pins configured as digital I/O.

Reading this register returns the logic value at the pin, regardless if it is configured as output or input.

### 12.4.14 P1MDIN: Port 1 Input Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| SFR Page = 0x0; SFR Address: 0xF2 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 1 | RW | **Port 1 Bit 7 Input Mode.** |
| | Value | Name | | Description |
| | 0 | ANALOG | | P1.7 pin is configured for analog mode. |
| | 1 | DIGITAL | | P1.7 pin is configured for digital mode. |
| 6 | B6 | 1 | RW | **Port 1 Bit 6 Input Mode.** |
| | See bit 7 description | | | |
| 5 | B5 | 1 | RW | **Port 1 Bit 5 Input Mode.** |
| | See bit 7 description | | | |
| 4 | B4 | 1 | RW | **Port 1 Bit 4 Input Mode.** |
| | See bit 7 description | | | |
| 3 | B3 | 1 | RW | **Port 1 Bit 3 Input Mode.** |
| | See bit 7 description | | | |
| 2 | B2 | 1 | RW | **Port 1 Bit 2 Input Mode.** |
| | See bit 7 description | | | |
| 1 | B1 | 1 | RW | **Port 1 Bit 1 Input Mode.** |
| | See bit 7 description | | | |
| 0 | B0 | 1 | RW | **Port 1 Bit 0 Input Mode.** |
| | See bit 7 description | | | |
| Port pins configured for analog mode have their weak pullup, digital driver, and digital receiver disabled. | | | | |

### 12.4.15 P1MDOUT: Port 1 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xA5 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 1 Bit 7 Output Mode.** |
| | Value | Name | | Description |
| | 0 | OPEN_DRAIN | | P1.7 output is open-drain. |
| | 1 | PUSH_PULL | | P1.7 output is push-pull. |
| 6 | B6 | 0 | RW | **Port 1 Bit 6 Output Mode.** |
| | See bit 7 description | | | |
| 5 | B5 | 0 | RW | **Port 1 Bit 5 Output Mode.** |
| | See bit 7 description | | | |
| 4 | B4 | 0 | RW | **Port 1 Bit 4 Output Mode.** |
| | See bit 7 description | | | |
| 3 | B3 | 0 | RW | **Port 1 Bit 3 Output Mode.** |
| | See bit 7 description | | | |
| 2 | B2 | 0 | RW | **Port 1 Bit 2 Output Mode.** |
| | See bit 7 description | | | |
| 1 | B1 | 0 | RW | **Port 1 Bit 1 Output Mode.** |
| | See bit 7 description | | | |
| 0 | B0 | 0 | RW | **Port 1 Bit 0 Output Mode.** |
| | See bit 7 description | | | |

### 12.4.16 P1SKIP: Port 1 Skip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xD5 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 1 Bit 7 Skip.** |
| | Value | Name | | Description |
| | 0 | NOT_SKIPPED | | P1.7 pin is not skipped by the crossbar. |
| | 1 | SKIPPED | | P1.7 pin is skipped by the crossbar. |
| 6 | B6 | 0 | RW | **Port 1 Bit 6 Skip.** |
| | See bit 7 description | | | |
| 5 | B5 | 0 | RW | **Port 1 Bit 5 Skip.** |
| | See bit 7 description | | | |
| 4 | B4 | 0 | RW | **Port 1 Bit 4 Skip.** |
| | See bit 7 description | | | |
| 3 | B3 | 0 | RW | **Port 1 Bit 3 Skip.** |
| | See bit 7 description | | | |
| 2 | B2 | 0 | RW | **Port 1 Bit 2 Skip.** |
| | See bit 7 description | | | |
| 1 | B1 | 0 | RW | **Port 1 Bit 1 Skip.** |
| | See bit 7 description | | | |
| 0 | B0 | 0 | RW | **Port 1 Bit 0 Skip.** |
| | See bit 7 description | | | |

### 12.4.17 P1DRV: Port 1 Drive Strength

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0xF; SFR Address: 0x9B | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 1 Bit 7 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.7 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P1.7 output has high output drive strength. |
| 6 | B6 | 0 | RW | **Port 1 Bit 6 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.6 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P1.6 output has high output drive strength. |
| 5 | B5 | 0 | RW | **Port 1 Bit 5 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.5 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P1.5 output has high output drive strength. |
| 4 | B4 | 0 | RW | **Port 1 Bit 4 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.4 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P1.4 output has high output drive strength. |
| 3 | B3 | 0 | RW | **Port 1 Bit 3 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.3 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P1.3 output has high output drive strength. |
| 2 | B2 | 0 | RW | **Port 1 Bit 2 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.2 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P1.2 output has high output drive strength. |
| 1 | B1 | 0 | RW | **Port 1 Bit 1 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.1 output has low output drive strength. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | 1 | HIGH_DRIVE | | P1.1 output has high output drive strength. |
| 0 | B0 | 0 | RW | **Port 1 Bit 0 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P1.0 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P1.0 output has high output drive strength. |

### 12.4.18 P2: Port 2 Pin Latch

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | Reserved | | | | | | |
| Access | RW | RW | | | | | | |
| Reset | 1 | 0x00 | | | | | | |
| SFR Page = ALL; SFR Address: 0xA0 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 1 | RW | **Port 2 Bit 7 Latch.** |
| | Value | Name | | Description |
| | 0 | LOW | | P2.7 is low. Set P2.7 to drive low. |
| | 1 | HIGH | | P2.7 is high. Set P2.7 to drive or float high. |
| *6:0* | *Reserved* | *Must write reset value.* | | |

Writing this register sets the port latch logic value for the associated I/O pins configured as digital I/O.

Reading this register returns the logic value at the pin, regardless if it is configured as output or input.

### 12.4.19 P2MDOUT: Port 2 Output Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | Reserved | | | | | | |
| Access | RW | RW | | | | | | |
| Reset | 0 | 0x00 | | | | | | |
| SFR Page = 0x0; SFR Address: 0xA6 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 2 Bit 7 Output Mode.** |
| | Value | Name | | Description |
| | 0 | OPEN_DRAIN | | P2.7 output is open-drain. |
| | 1 | PUSH_PULL | | P2.7 output is push-pull. |
| *6:0* | *Reserved* | *Must write reset value.* | | |

### 12.4.20 P2DRV: Port 2 Drive Strength

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | B7 | Reserved | | | | | | |
| Access | RW | RW | | | | | | |
| Reset | 0 | 0x00 | | | | | | |
| SFR Page = 0xF; SFR Address: 0x9D | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | B7 | 0 | RW | **Port 2 Bit 7 Drive Strength.** |
| | Value | Name | | Description |
| | 0 | LOW_DRIVE | | P2.7 output has low output drive strength. |
| | 1 | HIGH_DRIVE | | P2.7 output has high output drive strength. |
| *6:0* | *Reserved* | *Must write reset value.* | | |

## 12.5 INT0 and INT1 Control Registers

### 12.5.1 IT01CF: INT0/INT1 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IN1PL | IN1SL | | | IN0PL | IN0SL | | |
| Access | RW | RW | | | RW | RW | | |
| Reset | 0 | 0x0 | | | 0 | 0x1 | | |
| SFR Page = 0x0; SFR Address: 0xE4 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | IN1PL | 0 | RW | **INT1 Polarity.** |

| Value | Name | Description |
|---|---|---|
| 0 | ACTIVE_LOW | INT1 input is active low. |
| 1 | ACTIVE_HIGH | INT1 input is active high. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6:4 | IN1SL | 0x0 | RW | **INT1 Port Pin Selection.**<br>These bits select which port pin is assigned to INT1. This pin assignment is independent of the Crossbar; INT1 will monitor the assigned port pin without disturbing the peripheral that has been assigned the port pin via the Crossbar. The Crossbar will not assign the port pin to a peripheral if it is configured to skip the selected pin. |

| Value | Name | Description |
|---|---|---|
| 0x0 | P0_0 | Select P0.0. |
| 0x1 | P0_1 | Select P0.1. |
| 0x2 | P0_2 | Select P0.2. |
| 0x3 | P0_3 | Select P0.3. |
| 0x4 | P0_4 | Select P0.4. |
| 0x5 | P0_5 | Select P0.5. |
| 0x6 | P0_6 | Select P0.6. |
| 0x7 | P0_7 | Select P0.7. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 3 | IN0PL | 0 | RW | **INT0 Polarity.** |

| Value | Name | Description |
|---|---|---|
| 0 | ACTIVE_LOW | INT0 input is active low. |
| 1 | ACTIVE_HIGH | INT0 input is active high. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2:0 | IN0SL | 0x1 | RW | **INT0 Port Pin Selection.**<br>These bits select which port pin is assigned to INT0. This pin assignment is independent of the Crossbar; INT0 will monitor the assigned port pin without disturbing the peripheral that has been assigned the port pin via the Crossbar. The Crossbar will not assign the port pin to a peripheral if it is configured to skip the selected pin. |

| Value | Name | Description |
|---|---|---|
| 0x0 | P0_0 | Select P0.0. |
| 0x1 | P0_1 | Select P0.1. |
| 0x2 | P0_2 | Select P0.2. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | 0x3 | P0_3 | | Select P0.3. |
| | 0x4 | P0_4 | | Select P0.4. |
| | 0x5 | P0_5 | | Select P0.5. |
| | 0x6 | P0_6 | | Select P0.6. |
| | 0x7 | P0_7 | | Select P0.7. |

# 13. Analog-to-Digital Converter (ADC0)

## 13.1 Introduction

The ADC is a successive-approximation-register (SAR) ADC with 12-, 10-, and 8-bit modes, integrated track-and hold and a programmable window detector. The ADC is fully configurable under software control via several registers. The ADC may be configured to measure different signals using the analog multiplexer. The voltage reference for the ADC is selectable between internal and external reference sources.

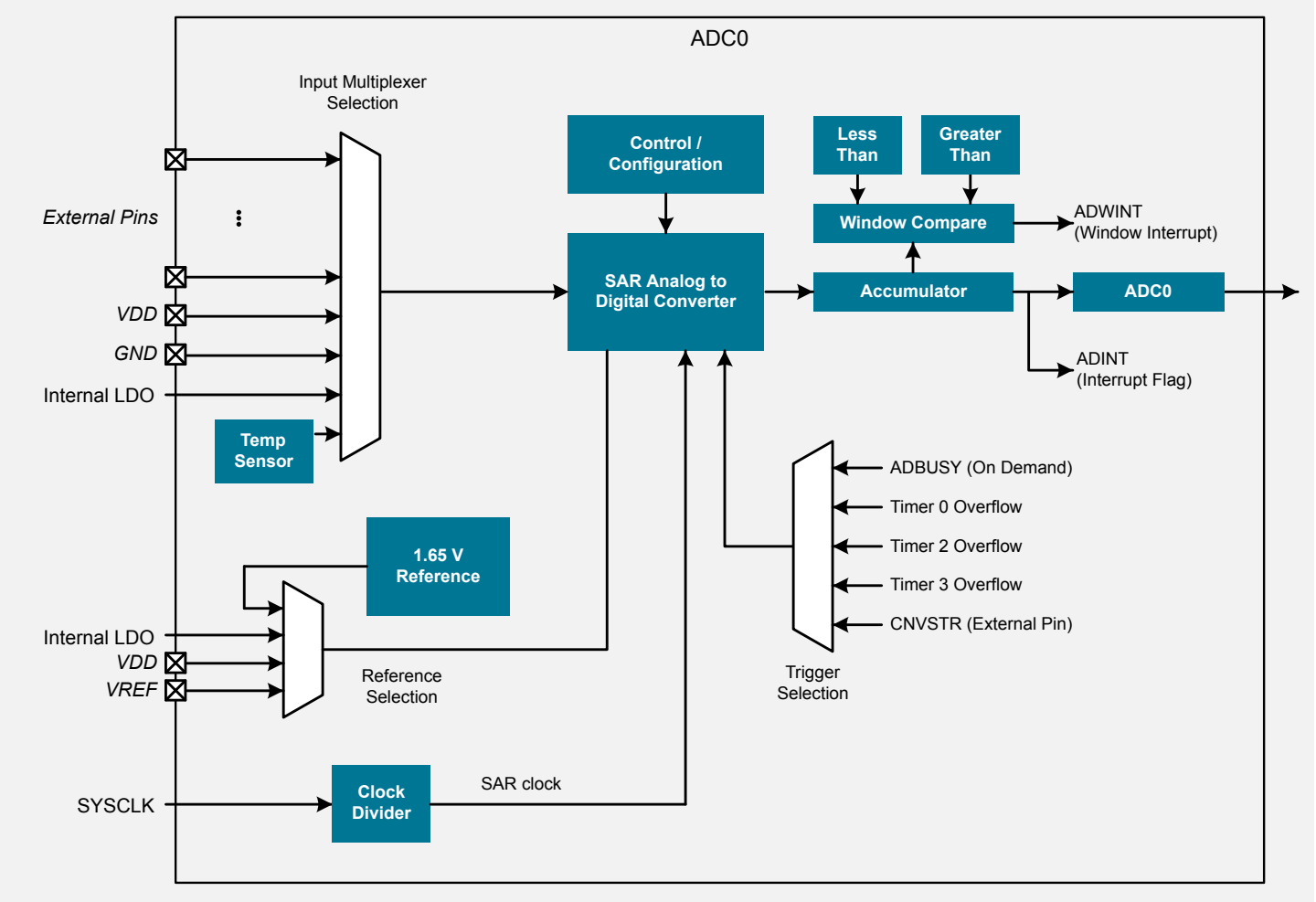



**Figure 13.1. ADC Block Diagram**

## 13.2 Features

- Up to 10 external inputs.
- Single-ended 12-bit and 10-bit modes.
- Supports an output update rate of 75 ksps samples per second in 12-bit mode or 300 ksps samples per second in 10-bit mode.
- Operation in low power modes at lower conversion speeds.
- Asynchronous hardware conversion trigger, selectable between software, external I/O and internal timer sources.
- Output data window comparator allows automatic range checking.
- Support for burst mode, which produces one set of accumulated data per conversion-start trigger with programmable power-on settling and tracking time.
- Conversion complete and window compare interrupts supported.
- Flexible output data formatting.
- Includes an internal 1.65 V fast-settling reference and support for external reference.
- Integrated temperature sensor.

## 13.3 Functional Description

### 13.3.1 Clocking

The ADC is clocked by an adjustable conversion clock (SARCLK). SARCLK is a divided version of the selected system clock when burst mode is disabled (ADBMEN = 0), or a divided version of the LPOSC0 oscillator when burst mode is enabled (ADBMEN = 1). The clock divide value is determined by the AD0SC field. In most applications, SARCLK should be adjusted to operate as fast as possible, without exceeding the maximum electrical specifications. The SARCLK does not directly determine sampling times or sampling rates.

### 13.3.2 Voltage Reference Options

The voltage reference multiplexer is configurable to use a number of different internal and external reference sources. The ground reference mux allows the ground reference for ADC0 to be selected between the ground pin (GND) or a port pin dedicated to analog ground (AGND). The voltage and ground reference options are configured using the REF0CN register. The REFSL field selects between the different reference options, while GNDSL configures the ground connection.

#### 13.3.2.1 Internal Voltage Reference

The high-speed internal reference offers two programmable voltage levels, and is self-contained and stabilized. It is not routed to an external pin and requires no external decoupling. When selected, the internal reference will be automatically enabled/disabled on an as-needed basis by the ADC. The reference can be set to one of two voltage values: 1.65 V or 2.4 V, depending on the value of the IREFLVL bit. The electrical specifications tables detail SAR clock and throughput limitations for each reference source.

#### 13.3.2.2 Supply or LDO Voltage Reference

For applications with a non-varying power supply voltage, using the power supply as the voltage reference can provide the ADC with added dynamic range at the cost of reduced power supply noise rejection. Additionally, the internal 1.8 V LDO supply to the core may be used as a reference. Neither of these reference sources are routed to the VREF pin, and do not require additional external decoupling.

#### 13.3.2.3 External Voltage Reference

An external reference may be applied to the VREF pin. Bypass capacitors should be added as recommended by the manufacturer of the external voltage reference. If the manufacturer does not provide recommendations, a 4.7 µF in parallel with a 0.1 µF capacitor is recommended.
**Note:** The VREF pin is a multi-function GPIO pin. When using an external voltage reference, VREF should be configured as an analog input and skipped by the crossbar.

#### 13.3.2.4 Ground Reference

To prevent ground noise generated by switching digital logic from affecting sensitive analog measurements, a separate analog ground reference option is available. When enabled, the ground reference for the ADC during both the tracking/sampling and the conversion periods is taken from the AGND pin. Any external sensors sampled by the ADC should be referenced to the AGND pin. If an external voltage reference is used, the AGND pin should be connected to the ground of the external reference and its associated decoupling capacitor. The separate analog ground reference option is enabled by setting GNDSL to 1. Note that when sampling the internal temperature sensor, the internal chip ground is always used for the sampling operation, regardless of the setting of the GNDSL bit. Similarly, whenever the internal high-speed reference is selected, the internal chip ground is always used during the conversion period, regardless of the setting of the GNDSL bit.
**Note:** The AGND pin is a multi-function GPIO pin. When using AGND as the ground reference to the ADC, AGND should be configured as an analog input and skipped by the crossbar.

### 13.3.3 Input Selection

The ADC has an analog multiplexer which allows selection of external pins, the on-chip temperature sensor, the internal regulated supply, the VDD supply, or GND. ADC input channels are selected using the ADC0MX register.
**Note:** Any port pins selected as ADC inputs should be configured as analog inputs in their associated port configuration register, and configured to be skipped by the crossbar.

#### 13.3.3.1 Multiplexer Channel Selection

**Table 13.1. ADC0 Input Multiplexer Channels**

| ADC0MX setting | Signal Name | Enumeration Name | QSOP24 Pin Name | QFN24 Pin Name | QFN20 Pin Name |
|---|---|---|---|---|---|
| 00000 | ADC0.0 | ADC0P0 | Reserved | Reserved | Reserved |
| 00001 | ADC0.1 | ADC0P1 | P0.1 | P0.1 | P0.1 |
| 00010 | ADC0.2 | ADC0P2 | P0.2 | P0.2 | P0.2 |
| 00011 | ADC0.3 | ADC0P3 | P0.3 | P0.3 | P0.3 |
| 00100 | ADC0.4 | ADC0P4 | P0.4 | P0.4 | P0.4 |
| 00101 | ADC0.5 | ADC0P5 | P0.5 | P0.5 | P0.5 |
| 00110 | ADC0.6 | ADC0P6 | P0.6 | P0.6 | P0.6 |
| 00111 | ADC0.7 | ADC0P7 | P0.7 | P0.7 | P0.7 |
| 01000 - 01001 | ADC0.8 - ADC0.9 | | Reserved | Reserved | Reserved |
| 01010 | ADC0.10 | ADC0P10 | P1.2 | P1.2 | P1.2 |
| 01011 | ADC0.11 | ADC0P11 | P1.3 | P1.3 | P1.3 |
| 01100 | ADC0.12 | ADC0P12 | P1.4 | P1.4 | Reserved |
| 01101 - 11010 | ADC0.13 - ADC0.26 | ADC0P13 | Reserved | Reserved | Reserved |
| 11011 | ADC0.27 | TEMP | Internal Temperature Sensor | | |
| 11100 | ADC0.28 | VDD | VDD Supply Pin | | |
| 11101 | ADC0.29 | LDO_OUT | Internal 1.8 V LDO Output | | |
| 11110 | ADC0.30 | | Reserved | | |
| 11111 | ADC0.31 | GND | GND Supply Pin | | |

### 13.3.4 Gain Setting

The ADC has gain settings of 1x and 0.5x. In 1x mode, the full scale reading of the ADC is determined directly by VREF. In 0.5x mode, the full-scale reading of the ADC occurs when the input voltage is VREF x 2. The 0.5x gain setting can be useful to obtain a higher input voltage range when using a small VREF voltage, or to measure input voltages that are between VREF and the supply voltage. Gain settings for the ADC are controlled by the ADGN bit in register ADC0CF. Note that even with a gain setting of 0.5, voltages above the supply rail cannot be measured directly by the ADC.

### 13.3.5 Initiating Conversions

A conversion can be initiated in many ways, depending on the programmed state of the ADCM bitfield. Conversions may be initiated by one of the following:

1. Software-triggered—Writing a 1 to the ADBUSY bit initiates the conversion.
2. Hardware-triggered—An automatic internal event such as a timer overflow initiates the conversion.
3. External pin-triggered—A rising edge on the CNVSTR input signal initiates the conversion.

Writing a 1 to ADBUSY provides software control of ADC0 whereby conversions are performed "on-demand". All other trigger sources occur autonomous to code execution. When the conversion is complete, the ADC posts the result to its output register and sets the ADC interrupt flag (ADINT). ADINT may be used to trigger a system interrupts, if enabled, or polled by firmware.

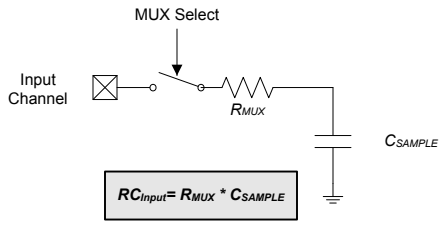
During a conversion, the ADBUSY bit is set to logic 1 and reset to logic 0 when the conversion is complete. However, the ADBUSY bit should not be used to poll for ADC conversion completion. The ADC0 interrupt flag (ADINT) should be used instead of the ADBUSY bit. Converted data is available in the ADC0 data registers, ADC0H:ADC0L, when the conversion is complete.
**Note:** The CNVSTR pin is a multi-function GPIO pin. When the CNVSTR input is used as the ADC conversion source, the associated port pin should be skipped in the crossbar settings.

### 13.3.6 Input Tracking

Each ADC conversion must be preceded by a minimum tracking time to allow the voltage on the sampling capacitor to settle, and for the converted result to be accurate.

#### Settling Time Requirements

The absolute minimum tracking time is given in the electrical specifications tables. It may be necessary to track for longer than the minimum tracking time specification, depending on the application. For example, if the ADC input is presented with a large series impedance, it will take longer for the sampling cap to settle on the final value during the tracking phase. The exact amount of tracking time required is a function of all series impedance (including the internal mux impedance and any external impedance sources), the sampling capacitance, and the desired accuracy.




Note: The value of $C_{SAMPLE}$ depends on the PGA gain. See the electrical specifications for details.

**Figure 13.2. ADC Eqivalent Input Circuit**

The required ADC0 settling time for a given settling accuracy (SA) may be approximated as follows:

$$t = \ln\left(\frac{2^n}{SA}\right) \times R_{TOTAL} \times C_{SAMPLE}$$

Where: SA is the settling accuracy, given as a fraction of an LSB (for example, 0.25 to settle within 1/4 LSB)

t is the required settling time in seconds

$R_{TOTAL}$ is the sum of the ADC mux resistance and any external source resistance.

$C_{SAMPLE}$ is the size of the ADC sampling capacitor.

n is the ADC resolution in bits.

When measuring any internal source, $R_{TOTAL}$ reduces to $R_{MUX}$. See the electrical specification tables in the datasheet for ADC minimum settling time requirements as well as the mux impedance and sampling capacitor values.

### Configuring the Tracking Time

When burst mode is disabled, the ADTM bit controls the ADC track-and-hold mode. In its default state the ADC input is continuously tracked, except when a conversion is in progress. A conversion will begin immediately when the start-of-conversion trigger occurs. When the ADTM bit is logic 1, each conversion is preceded by a tracking period of 4 SAR clocks (after the start-of-conversion signal) for any internal conversion trigger source. When the CNVSTR signal is used to initiate conversions with ADTM set to 1, ADC0 tracks only when CNVSTR is low; conversion begins on the rising edge of CNVSTR. Setting ADTM to 1 is primarily useful when AMUX settings are frequently changed and conversions are started using the ADBUSY bit.

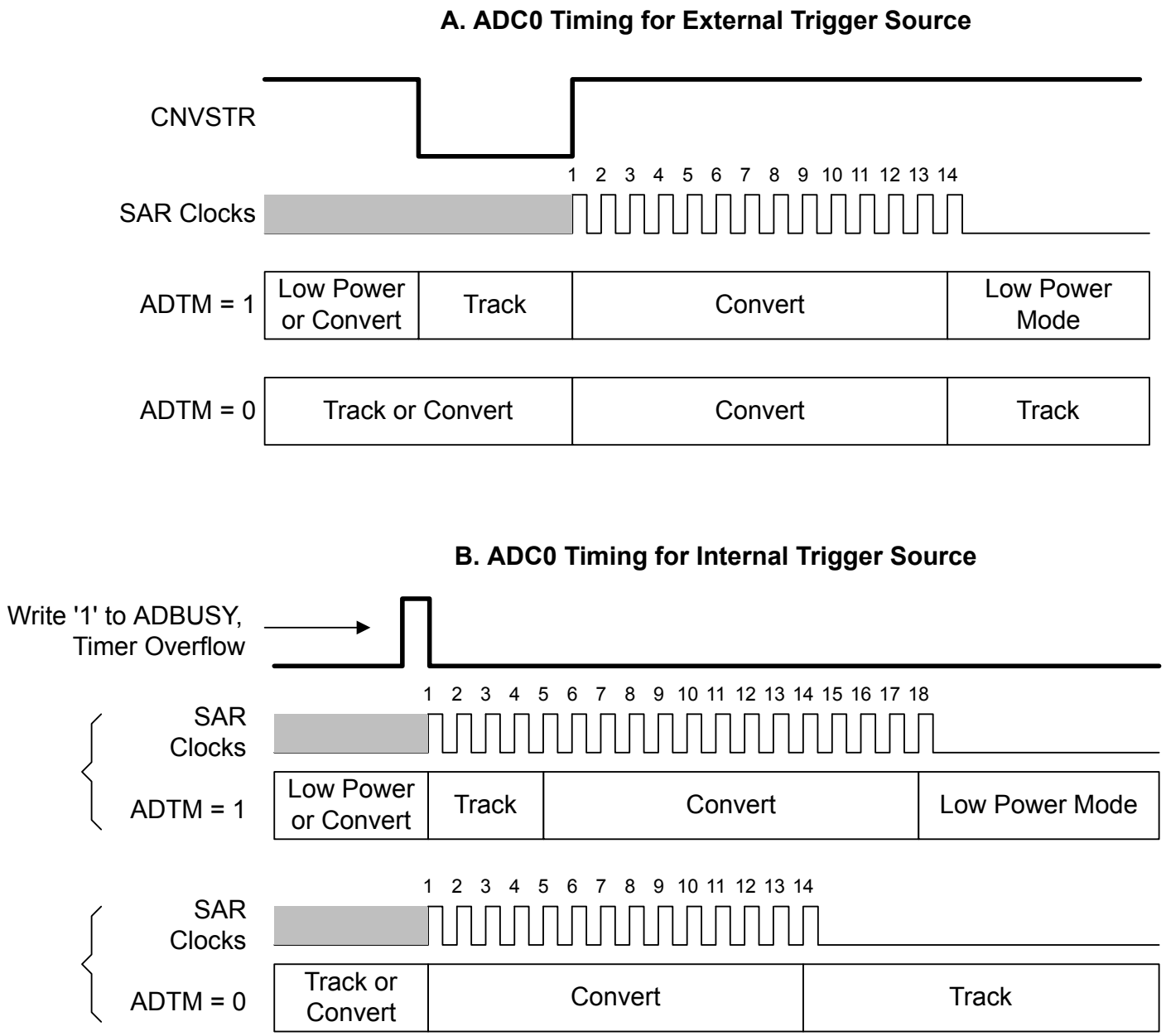
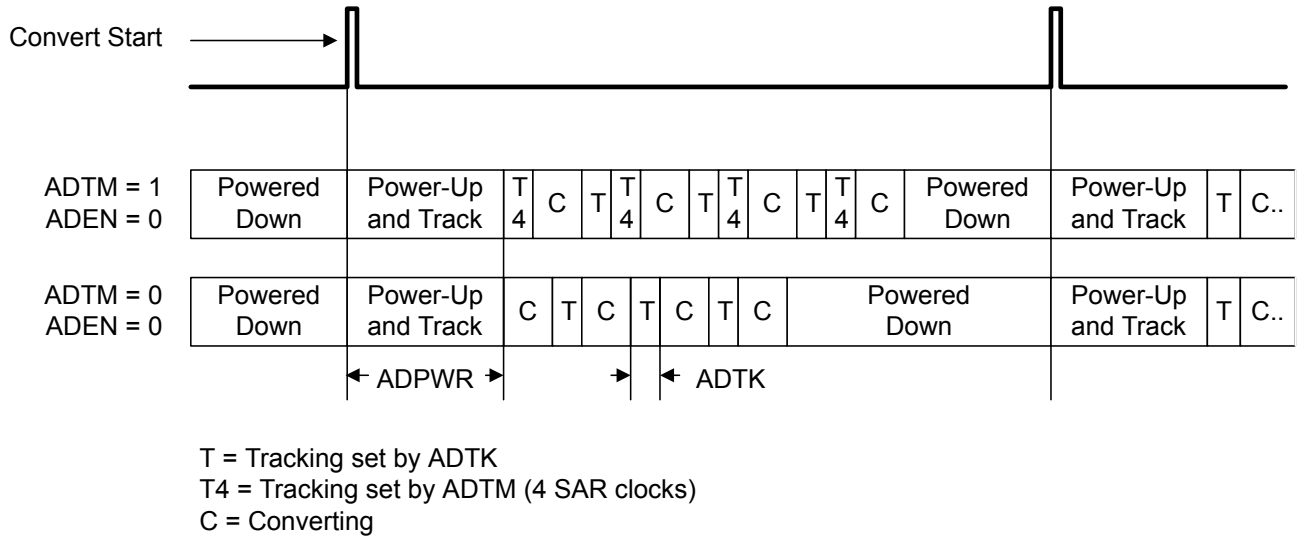



**Figure 13.3. Track and Conversion Example Timing (Normal, Non-Burst Operation)**

When burst mode is enabled, additional tracking times may need to be specified. Because burst mode may power the ADC on from an unpowered state and take multiple conversions for each start-of-conversion source, two additional timing fields are provided. If the ADC is powered down when the burst sequence begins, it will automatically power up and wait for the time specified in the ADPWR bit field. If the ADC is already powered on, tracking depends solely on ADTM for the first conversion. The ADTK field determines the amount of tracking time given to any subsequent samples in burst mode—essentially, ADTK specifies how long the ADC will wait between burt-mode conversions. If ADTM is set, an additional 4 SAR clocks will be added to the tracking phase of all conversions in burst mode.

**Figure 13.4. Burst Mode Timing**




T = Tracking set by ADTK
T4 = Tracking set by ADTM (4 SAR clocks)
C = Converting

### 13.3.7 Burst Mode

Burst mode is a power saving feature that allows the ADC to remain in a low power state between conversions. When burst mode is enabled, the ADC wakes from a low power state, accumulates 1, 4, 8, 16, 32, or 64 samples using the internal low-power high-frequency oscillator, then re-enters a low power state. Since the burst mode clock is independent of the system clock, the ADC can perform multiple conversions then enter a low power state within a single system clock cycle, even if the system clock is running from a slow oscillator.
**Note:** When using burst mode, care must be taken to issue a convert start signal no faster than once every four SYSCLK periods. This includes external convert start signals. The ADC will ignore convert start signals which arrive before a burst is finished.

Burst mode is enabled by setting ADBMEN to logic 1. When in burst mode, ADEN controls the ADC idle power state (i.e., the state the ADC enters when not tracking or performing conversions). If ADEN is set to logic 0, the ADC is powered down after each burst. If ADEN is set to logic 1, the ADC remains enabled after each burst. On each convert start signal, the ADC is awakened from its idle power state. If the ADC is powered down, it will automatically power up and wait for the amount of time programmed to the ADPWR bits before performing a conversion. Otherwise, the ADC will start tracking and converting immediately.

When burst mode is enabled, a single convert start will initiate a number of conversions equal to the repeat count. When burst mode is disabled, a convert start is required to initiate each conversion. In both modes, the ADC end of conversion interrupt flag (ADINT) will be set after “repeat count” conversions have been accumulated. Similarly, the window comparator will not compare the result to the greater-than and less-than registers until “repeat count” conversions have been accumulated.

### 13.3.8 8-Bit Mode

Setting the AD8BE bit to 1 will put the ADC in 8-bit mode. In 8-bit mode, only the 8 MSBs of data are converted, allowing the conversion to be completed in fewer SAR clock cycles than a 10-bit conversion. The two LSBs of a conversion are always 00 in this mode, and the ADC0L register will always read back 0x00.

### 13.3.9 12-Bit Mode

When configured for 12-bit conversions, the ADC performs four 10-bit conversions using four different reference voltages and combines the results into a single 12-bit value. Unlike simple averaging techniques, this method provides true 12-bit resolution of ac or dc input signals without depending on noise to provide dithering. The converter also employs a hardware dynamic element matching algorithm that reconfigures the largest elements of the internal DAC for each of the four 10-bit conversions. This reconfiguration cancels any matching errors and enables the converter to achieve 12-bit linearity performance to go along with its 12-bit resolution.

The 12-bit mode is enabled by setting the AD12BE bit in register ADC0AC to logic 1 and configuring the ADC in burst mode (ADBMEN = 1) for four or more conversions. The conversion can be initiated using any of the conversion start sources, and the 12-bit result will appear in the ADC0H and ADC0L registers. Since the 12-bit result is formed from a combination of four 10-bit results, the maximum output value is 4 x (1023) = 4092, rather than the max value of (2^12 – 1) = 4095 that is produced by a traditional 12-bit converter. To further increase resolution, the burst mode repeat value may be configured to any multiple of four conversions. For example, if a repeat value of 16 is selected, the ADC0 output will be a 14-bit number (sum of four 12-bit numbers) with 13 effective bits of resolution.

The AD12SM bit in register ADC0TK controls when the ADC will track and sample the input signal. When AD12SM is set to 1, the selected input signal will be tracked before the first conversion of a set and held internally during all four conversions. When AD12SM is cleared to 0, the ADC will track and sample the selected input before each of the four conversions in a set. When maximum throughput (180-200 ksps) is needed, it is recommended that AD12SM be set to 1 and ADTK to 0x3F, and that the ADC be placed in always-on mode (ADEN = 1). For sample rates under 180 ksps, or when accumulating multiple samples, AD12SM should normally be cleared to 0, and ADTK should be configured to provide the appropriate settling time for the subsequent conversions.

### 13.3.10 Output Formatting

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 ADSJST field. When the repeat count is set to 1 in 10-bit mode, 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.

**Table 13.2. 10-Bit Output Code Example**

| Input Voltage | Right-Justified (ADSJST = 000)<br>ADC0H:L | Left-Justified (ADSJST = 100)<br>ADC0H:L |
|---|---|---|
| VREF x 1023/1024 | 0x03FF | 0xFFC0 |
| VREF x 512/1024 | 0x0200 | 0x8000 |
| VREF x 256/1024 | 0x0100 | 0x4000 |
| 0 | 0x0000 | 0x0000 |

When the repeat count is greater than 1, the output conversion code represents the accumulated result of the conversions performed and is updated after the last conversion in the series is finished. Sets of 4, 8, 16, 32, or 64 consecutive samples can be accumulated and represented in unsigned integer format. The repeat count can be selected using the ADRPT bit field. When a repeat count is higher than 1, the ADC output must be right-justified (ADSJST = 0xx); unused bits in the ADC0H and ADC0L registers are set to 0. The example below shows the right-justified result for various input voltages and repeat counts. Notice that accumulating 2n samples is equivalent to left-shifting by n bit positions when all samples returned from the ADC have the same value.

**Table 13.3. Effects of ADRPT on Output Code**

| Input Voltage | Repeat Count = 4 | Repeat Count = 16 | Repeat Count = 64 |
|---|---|---|---|
| VREF x 1023/1024 | 0x0FFC | 0x3FF0 | 0xFFC0 |
| VREF x 512/1024 | 0x0800 | 0x2000 | 0x8000 |
| VREF x 511/1024 | 0x07FC | 0x1FF0 | 0x7FC0 |
| 0 | 0x0000 | 0x0000 | 0x0000 |

Additionally, the ADSJST bit field can be used to format the contents of the 16-bit accumulator. The accumulated result can be shifted right by 1, 2, or 3 bit positions. Based on the principles of oversampling and averaging, the effective ADC resolution increases by 1 bit each time the oversampling rate is increased by a factor of 4. The example below shows how to increase the effective ADC resolution by 1, 2, and 3 bits to obtain an effective ADC resolution of 11- bit, 12-bit, or 13-bit respectively without CPU intervention.

**Table 13.4. Using ADSJST for Output Formatting**

| Input Voltage | Repeat Count = 4<br>Shift Right = 1<br>11-Bit Result | Repeat Count = 16<br>Shift Right = 2<br>12-Bit Result | Repeat Count = 64<br>Shift Right = 3<br>12-Bit Result |
|---|---|---|---|
| VREF x 1023/1024 | 0x07F7 | 0x0FFC | 0x1FF8 |
| VREF x 512/1024 | 0x0400 | 0x0800 | 0x1000 |
| VREF x 511/1024 | 0x03FE | 0x04FC | 0x0FF8 |
| 0 | 0x0000 | 0x0000 | 0x0000 |

### 13.3.11 Window Comparator

The ADC's programmable window detector continuously compares the ADC 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 (ADWINT) can also be used in polled mode. The ADC 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 ADC0GT and ADC0LT registers. The following tables show how the ADC0GT and ADC0LT registers may be configured to set the ADWINT flag when the ADC output code is above, below, beween, or outside of specific values.

**Table 13.5. ADC Window Comparator Example (Above 0x0080)**

| Comparison Register Settings | Output Code (ADC0H:L) | ADWINT Effects |
|---|---|---|
| | 0x03FF | ADWINT = 1 |
| | ... | |
| | 0x0081 | |
| ADC0GTH:L = 0x0080 | 0x0080 | ADWINT Not Affected |
| | 0x007F | |
| | ... | |
| | 0x0001 | |
| ADC0LTH:L = 0x0000 | 0x0000 | |

**Table 13.6. ADC Window Comparator Example (Below 0x0040)**

| Comparison Register Settings | Output Code (ADC0H:L) | ADWINT Effects |
|---|---|---|
| ADC0GTH:L = 0x03FF | 0x03FF | ADWINT Not Affected |
| | 0x03FE | |
| | ... | |
| | 0x0041 | |
| ADC0LTH:L = 0x0040 | 0x0040 | |
| | 0x003F | ADWINT = 1 |
| | ... | |
| | 0x0000 | |

**Table 13.7. ADC Window Comparator Example (Between 0x0040 and 0x0080)**

| Comparison Register Settings | Output Code (ADC0H:L) | ADWINT Effects |
|---|---|---|
| | 0x03FF | ADWINT Not Affected |
| | ... | |
| | 0x0081 | |
| ADC0LTH:L = 0x0080 | 0x0080 | |
| | 0x007F | ADWINT = 1 |
| | ... | |
| | 0x0041 | |

| Comparison Register Settings | Output Code (ADC0H:L) | ADWINT Effects |
|---|---|---|
| ADC0GTH:L = 0x0040 | 0x0040 | ADWINT Not Affected |
| | 0x003F | |
| | ... | |
| | 0x0000 | |

**Table 13.8. ADC Window Comparator Example (Outside the 0x0040 to 0x0080 range)**

| Comparison Register Settings | Output Code (ADC0H:L) | ADWINT Effects |
|---|---|---|
| | 0x03FF | ADWINT = 1 |
| | ... | |
| | 0x0081 | |
| ADC0GTH:L = 0x0080 | 0x0080 | ADWINT Not Affected |
| | 0x007F | |
| | ... | |
| | 0x0041 | |
| ADC0LTH:L = 0x0040 | 0x0040 | |
| | 0x003F | ADWINT = 1 |
| | ... | |
| | 0x0000 | |

### 13.3.12 Temperature Sensor

An on-chip analog temperature sensor is available to the ADC multiplexer input. To use the ADC to measure the temperature sensor, the ADC mux channel should select the temperature sensor. The temperature sensor transfer function is shown in Figure 13.5 Temperature Sensor Transfer Function on page 134. The output voltage ($V_{TEMP}$) is the positive ADC input when the ADC multiplexer is set correctly. The TEMPE bit in register REF0CN enables/ disables the temperature sensor. While disabled, the temperature sensor defaults to a high impedance state and any ADC measurements performed on the sensor will result in meaningless data. Refer to the electrical specification tables for the slope and offset parameters of the temperature sensor.

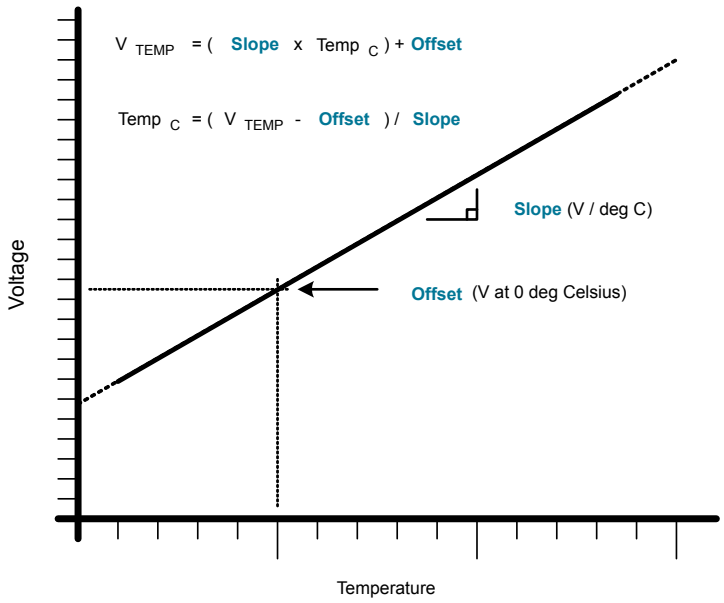



**Figure 13.5. Temperature Sensor Transfer Function**

#### 13.3.12.1 Temperature Sensor Calibration

The uncalibrated temperature sensor output is extremely linear and suitable for relative temperature measurements. For absolute temperature measurements, offset and/or gain calibration is recommended. Typically a 1-point (offset) calibration includes the following steps:

1. Control/measure the ambient temperature (this temperature must be known).
2. Power the device, and delay for a few seconds to allow for self-heating.
3. Perform an ADC conversion with the temperature sensor selected as the ADC input.
4. Calculate the offset characteristics, and store this value in non-volatile memory for use with subsequent temperature sensor measurements.

Although more precision can be obtained by calibrating the temperature sensor in the end system, a single-point offset measurement of the temperature sensor is performed on each device during production test. The measurement is performed at 25 °C ±5 °C, using the ADC with the internal high speed reference buffer selected as the Voltage Reference. The direct ADC result of this measurement is stored in the SFR registers TOFFH and TOFFL

## 13.4 ADC0 Control Registers

### 13.4.1 ADC0CN0: ADC0 Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADEN | ADBMEN | ADINT | ADBUSY | ADWINT | ADCM | | |
| Access | RW | RW | RW | RW | RW | RW | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0x0 | | |
| SFR Page = 0x0; SFR Address: 0xE8 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | ADEN | 0 | RW | **ADC Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable ADC0 (low-power shutdown). |
| | 1 | ENABLED | | Enable ADC0 (active and ready for data conversions). |
| 6 | ADBMEN | 0 | RW | **Burst Mode Enable.** |
| | Value | Name | | Description |
| | 0 | BURST_DISABLED | | Disable ADC0 burst mode. |
| | 1 | BURST_ENABLED | | Enable ADC0 burst mode. |
| 5 | ADINT | 0 | RW | **Conversion Complete Interrupt Flag.** |
| | Set by hardware upon completion of a data conversion (ADBMEN=0), or a burst of conversions (ADBMEN=1). Can trigger an interrupt. Must be cleared by firmware. | | | |
| 4 | ADBUSY | 0 | RW | **ADC Busy.** |
| | Writing 1 to this bit initiates an ADC conversion when ADCM = 000. This bit should not be polled to indicate when a conversion is complete. Instead, the ADINT bit should be used when polling for conversion completion. | | | |
| 3 | ADWINT | 0 | RW | **Window Compare Interrupt Flag.** |
| | Set by hardware when the contents of ADC0H:ADC0L fall within the window specified by ADC0GTH:ADC0GTL and ADC0LTH:ADC0LTL. Can trigger an interrupt. Must be cleared by firmware. | | | |
| 2:0 | ADCM | 0x0 | RW | **Start of Conversion Mode Select.** |
| | Specifies the ADC0 start of conversion source. All remaining bit combinations are reserved. | | | |
| | Value | Name | | Description |
| | 0x0 | ADBUSY | | ADC0 conversion initiated on write of 1 to ADBUSY. |
| | 0x1 | TIMER0 | | ADC0 conversion initiated on overflow of Timer 0. |
| | 0x2 | TIMER2 | | ADC0 conversion initiated on overflow of Timer 2. |
| | 0x3 | TIMER3 | | ADC0 conversion initiated on overflow of Timer 3. |
| | 0x4 | CNVSTR | | ADC0 conversion initiated on rising edge of CNVSTR. |

### 13.4.2 ADC0CF: ADC0 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADSC | | | | | AD8BE | ADTM | ADGN |
| Access | RW | | | | | RW | RW | RW |
| Reset | 0x1F | | | | | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0x97 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:3 | ADSC | 0x1F | RW | **SAR Clock Divider.**<br>This field sets the ADC clock divider value. It should be configured to be as close to the maximum SAR clock speed as the datasheet will allow. The SAR clock frequency is given by the following equation:<br>Fclksar = (Fadcclk) / (ADSC + 1)<br>$F_{ADCCLK}$ is equal to the selected SYSCLK when ADBMEN is 0 and the high-frequency oscillator when ADBMEN is 1. |
| 2 | AD8BE | 0 | RW | **8-Bit Mode Enable.** |
| | Value | Name | | Description |
| | 0 | NORMAL | | ADC0 operates in 10-bit or 12-bit mode (normal operation). |
| | 1 | 8_BIT | | ADC0 operates in 8-bit mode. |
| 1 | ADTM | 0 | RW | **Track Mode.**<br>Selects between Normal or Delayed Tracking Modes. |
| | Value | Name | | Description |
| | 0 | TRACK_NORMAL | | Normal Track Mode. When ADC0 is enabled, conversion begins immediately following the start-of-conversion signal. |
| | 1 | TRACK_DELAYED | | Delayed Track Mode. When ADC0 is enabled, conversion begins 3 SAR clock cycles following the start-of-conversion signal. The ADC is allowed to track during this time. |
| 0 | ADGN | 0 | RW | **Gain Control.** |
| | Value | Name | | Description |
| | 0 | GAIN_0P5 | | The on-chip PGA gain is 0.5. |
| | 1 | GAIN_1 | | The on-chip PGA gain is 1. |

### 13.4.3 ADC0AC: ADC0 Accumulator Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AD12BE | ADAE | ADSJST | | | ADRPT | | |
| Access | RW | RW | RW | | | RW | | |
| Reset | 0 | 0 | 0x0 | | | 0x0 | | |
| SFR Page = 0x0; SFR Address: 0xBA | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | AD12BE | 0 | RW | **12-Bit Mode Enable.** |

Enables 12-bit mode. In 12-bit mode, the ADC throughput is reduced by a factor of 4.

| Value | Name | Description |
|---|---|---|
| 0 | 12_BIT_DISABLED | Disable 12-bit mode. |
| 1 | 12_BIT_ENABLED | Enable 12-bit mode. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | ADAE | 0 | RW | **Accumulate Enable.** |

Enables multiple conversions to be accumulated when burst mode is disabled.

| Value | Name | Description |
|---|---|---|
| 0 | ACC_DISABLED | ADC0H:ADC0L contain the result of the latest conversion when Burst Mode is disabled. |
| 1 | ACC_ENABLED | ADC0H:ADC0L contain the accumulated conversion results when Burst Mode is disabled. Firmware must write 0x0000 to ADC0H:ADC0L to clear the accumulated result. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5:3 | ADSJST | 0x0 | RW | **Accumulator Shift and Justify.** |

Specifies the format of data read from ADC0H:ADC0L. All remaining bit combinations are reserved.

| Value | Name | Description |
|---|---|---|
| 0x0 | RIGHT_NO_SHIFT | Right justified. No shifting applied. |
| 0x1 | RIGHT_SHIFT_1 | Right justified. Shifted right by 1 bit. |
| 0x2 | RIGHT_SHIFT_2 | Right justified. Shifted right by 2 bits. |
| 0x3 | RIGHT_SHIFT_3 | Right justified. Shifted right by 3 bits. |
| 0x4 | LEFT_NO_SHIFT | Left justified. No shifting applied. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2:0 | ADRPT | 0x0 | RW | **Repeat Count.** |

Selects the number of conversions to perform and accumulate in Burst Mode. This bit field must be set to 000 if Burst Mode is disabled.

| Value | Name | Description |
|---|---|---|
| 0x0 | ACC_1 | Perform and Accumulate 1 conversion (not used in 12-bit mode). |
| 0x1 | ACC_4 | Perform and Accumulate 4 conversions (1 conversion in 12-bit mode). |
| 0x2 | ACC_8 | Perform and Accumulate 8 conversions (2 conversions in 12-bit mode). |
| 0x3 | ACC_16 | Perform and Accumulate 16 conversions (4 conversions in 12-bit mode). |
| 0x4 | ACC_32 | Perform and Accumulate 32 conversions (8 conversions in 12-bit mode). |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | 0x5 | ACC_64 | | Perform and Accumulate 64 conversions (16 conversions in 12-bit mode). |

### 13.4.4 ADC0PWR: ADC0 Power Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADLPM | Reserved | | | ADPWR | | | |
| Access | RW | RW | | | RW | | | |
| Reset | 0 | 0x0 | | | 0xF | | | |
| SFR Page = ALL; SFR Address: 0xBB | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | ADLPM | 0 | RW | **Low Power Mode Enable.**<br>This bit can be used to reduce power to the ADC's internal common mode buffer. It can be set to 1 to reduce power when tracking times in the application are longer (slower sample rates). |
| | Value | Name | | Description |
| | 0 | LOW_POWER_DISABLED | | Disable low power mode. |
| | 1 | LOW_POWER_ENABLED | | Enable low power mode (requires extended tracking time). |
| *6:4* | *Reserved* | *Must write reset value.* | | |
| 3:0 | ADPWR | 0xF | RW | **Burst Mode Power Up Time.**<br>This field sets the time delay allowed for the ADC to power up from a low power state. When ADTM is set, an additional 3 SARCLKs are added to this time.<br>Tpwrtime = (8 * ADPWR) / (Fhfosc) |

### 13.4.5 ADC0TK: ADC0 Burst Mode Track Time

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | ADTK | | | | | |
| Access | R | | RW | | | | | |
| Reset | 0x0 | | 0x1E | | | | | |
| SFR Page = ALL; SFR Address: 0xBC | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:6* | *Reserved* | *Must write reset value.* | | |
| 5:0 | ADTK | 0x1E | RW | **Burst Mode Tracking Time.**<br>This field sets the time delay between consecutive conversions performed in Burst Mode. When ADTM is set, an additional 3 SARCLKs are added to this time.<br>Tbmtk = (64 - ADTK) / (Fhfosc)<br>The Burst Mode track delay is not inserted prior to the first conversion. The required tracking time for the first conversion should be defined with the ADPWR field. |

### 13.4.6 ADC0H: ADC0 Data Word High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0H | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xBE | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ADC0H | 0x00 | RW | **Data Word High Byte.**<br>When read, this register returns the most significant byte of the 16-bit ADC0 accumulator, formatted according to the settings in ADSJST. The register may also be written, to set the upper byte of the 16-bit ADC0 accumulator. |
| If Accumulator shifting is enabled, the most significant bits of the value read will be zeros. | | | | |

### 13.4.7 ADC0L: ADC0 Data Word Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0L | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xBD | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ADC0L | 0x00 | RW | **Data Word Low Byte.**<br>When read, this register returns the least significant byte of the 16-bit ADC0 accumulator, formatted according to the settings in ADSJST. The register may also be written, to set the lower byte of the 16-bit ADC0 accumulator. |
| If Accumulator shifting is enabled, the most significant bits of the value read will be zeros. | | | | |

### 13.4.8 ADC0GTH: ADC0 Greater-Than High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0GTH | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0xFF | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xC4 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ADC0GTH | 0xFF | RW | **Greater-Than High Byte.**<br>Most significant byte of the 16-bit greater-than window compare register. |

### 13.4.9 ADC0GTL: ADC0 Greater-Than Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0GTL | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0xFF | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xC3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ADC0GTL | 0xFF | RW | **Greater-Than Low Byte.**<br>Least significant byte of the 16-bit greater-than window compare register. |
| In 8-bit mode, this register should be set to 0x00. | | | | |

### 13.4.10 ADC0LTH: ADC0 Less-Than High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0LTH | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xC6 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ADC0LTH | 0x00 | RW | **Less-Than High Byte.**<br>Most significant byte of the 16-bit less-than window compare register. |

### 13.4.11 ADC0LTL: ADC0 Less-Than Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADC0LTL | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xC5 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | ADC0LTL | 0x00 | RW | **Less-Than Low Byte.**<br>Least significant byte of the 16-bit less-than window compare register. |
| In 8-bit mode, this register should be set to 0x00. | | | | |

### 13.4.12 ADC0MX: ADC0 Multiplexer Selection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | ADC0MX | | | | |
| Access | R | | | RW | | | | |
| Reset | 0x0 | | | 0x1F | | | | |
| SFR Page = 0x0; SFR Address: 0x96 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:5* | *Reserved* | *Must write reset value.* | | |
| 4:0 | ADC0MX | 0x1F | RW | **AMUX0 Positive Input Selection.**<br>Selects the positive input channel for ADC0. For reserved bit combinations, no input is selected. |

### 13.4.13 REF0CN: Voltage Reference Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | GNDSL | REFSL | | TEMPE | Reserved | |
| Access | R | | RW | RW | | RW | R | |
| Reset | 0x0 | | 0 | 0x3 | | 0 | 0x0 | |
| SFR Page = 0x0; SFR Address: 0xD1 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:6* | *Reserved* | *Must write reset value.* | | |
| 5 | GNDSL | 0 | RW | **Analog Ground Reference.**<br>Selects the ADC0 ground reference. |
| | Value | Name | | Description |
| | 0 | GND_PIN | | The ADC0 ground reference is the GND pin. |
| | 1 | AGND_PIN | | The ADC0 ground reference is the P0.1/AGND pin. |
| 4:3 | REFSL | 0x3 | RW | **Voltage Reference Select.**<br>Selects the ADC0 voltage reference. |
| | Value | Name | | Description |
| | 0x0 | VREF_PIN | | The ADC0 voltage reference is the P0.0/VREF pin. |
| | 0x1 | VDD_PIN | | The ADC0 voltage reference is the VDD pin. |
| | 0x2 | INTERNAL_LDO | | The ADC0 voltage reference is the internal 1.8 V digital supply voltage. |
| | 0x3 | HIGH_SPEED_VREF | | The ADC0 voltage reference is the internal 1.65 V high speed voltage reference. |
| 2 | TEMPE | 0 | RW | **Temperature Sensor Enable.**<br>Enables/Disables the internal temperature sensor. |
| | Value | Name | | Description |
| | 0 | TEMP_DISABLED | | Disable the Temperature Sensor. |
| | 1 | TEMP_ENABLED | | Enable the Temperature Sensor. |
| *1:0* | *Reserved* | *Must write reset value.* | | |

### 13.4.14 TOFFH: Temperature Sensor Offset High

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TOFF | | | | | | | |
| Access | R | | | | | | | |
| Reset | Varies | | | | | | | |
| SFR Page = 0xF; SFR Address: 0x8E | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TOFF | Varies | R | **Temperature Sensor Offset High.**<br>Most Significant Bits of the 10-bit temperature sensor offset measurement. |

### 13.4.15 TOFFL: Temperature Sensor Offset Low

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TOFF | | Reserved | | | | | |
| Access | R | | R | | | | | |
| Reset | Varies | | 0x00 | | | | | |
| SFR Page = 0xF; SFR Address: 0x8D | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:6 | TOFF | Varies | R | **Temperature Sensor Offset Low.**<br>Least Significant Bits of the 10-bit temperature sensor offset measurement. |
| *5:0* | *Reserved* | *Must write reset value.* | | |

# 14. Programmable Current Reference (IREF0)

## 14.1 Introduction

The programmable current reference (IREF0) module enables current source or sink with two output current settings: Low Power Mode and High Current Mode. The maximum current output in Low Power Mode is 63 µA (1 µA steps) and the maximum current output in High Current Mode is 504 µA (8 µA steps).

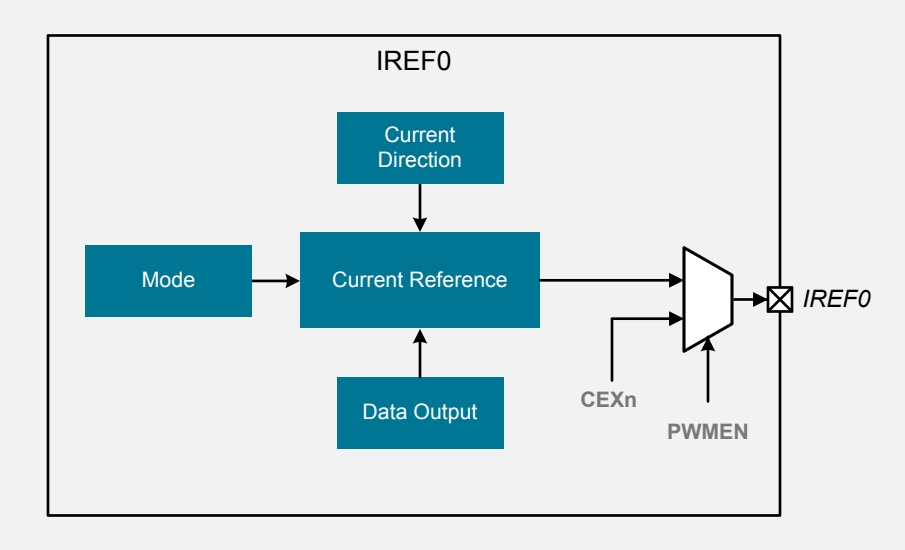



**Figure 14.1. IREF Block Diagram**

## 14.2 Features

The IREF module includes the following features:

- Capable of sourcing or sinking current in programmable steps.
- Two operational modes: Low Power Mode and High Current Mode.
- Fine-tuning mode for higher output precision available in conjunction with the PCA0 module.

## 14.3 Functional Description

### 14.3.1 Overview

The programmable current reference (IREF0) generates a current output in either source or sink mode. Each mode has two output current settings: Low Power Mode and High Current Mode. The maximum current output in Low Power Mode is 63 µA (1 µA steps) and the maximum current output in High Current Mode is 504 µA (8 µA steps). The port I/O pin associated with the IREF0 output should be configured as an analog input and skipped in the crossbar.

### 14.3.2 PWM Enhanced Mode

The precision of the current reference can be increased by fine tuning the IREF0 output using a PWM signal generated by the PCA. This mode allows the IREF0DAT bits in the IREF0CN0 register to perform a course adjustment on the IREF0 output. Any available PCA channel can perform a fine adjustment on the IREF0 output. When enabled, the CEX signal selected using the PWMSS bit field is internally routed to IREF0 to control the on time of a current source having the weight of 2 LSBs. With the two least significant bits of IREF0DAT set to 00b, applying a 100% duty cycle on the CEX signal will be equivalent to setting the two LSBs of IREF0DAT to 10b. PWM enhanced mode is enabled and set up using the IREF0CF register.

## 14.4 IREF0 Control Registers

### 14.4.1 IREF0CN0: Current Reference Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SINK | MDSEL | IREF0DAT | | | | | |
| Access | RW | RW | RW | | | | | |
| Reset | 0 | 0 | 0x00 | | | | | |
| SFR Page = 0x0; SFR Address: 0xD6 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | SINK | 0 | RW | **IREF0 Current Sink Enable.**<br>Selects if IREF0 is a current source or a current sink. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | IREF0 is a current source. |
| 1 | ENABLED | IREF0 is a current sink. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | MDSEL | 0 | RW | **IREF0 Output Mode Select.**<br>Selects Low Power or High Current Mode. |

| Value | Name | Description |
|---|---|---|
| 0 | LOW_POWER | Low Current Mode is selected (step size = 1 uA). |
| 1 | HIGH_CURRENT | High Current Mode is selected (step size = 8 uA). |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5:0 | IREF0DAT | 0x00 | RW | **IREF0 Data Word.**<br>Specifies the number of steps required to achieve the desired output current.<br>Output current = direction x step size x IREF0DAT.<br>IREF0 is in a low power state when IREF0DAT is set to 0x00. |

### 14.4.2 IREF0CF: Current Reference Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PWMEN | Reserved | | | | PWMSS | | |
| Access | RW | RW | | | | RW | | |
| Reset | 0 | 0x0 | | | | 0x0 | | |
| SFR Page = ALL; SFR Address: 0xB9 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | PWMEN | 0 | RW | **PWM Enhanced Mode Enable.** |
| | Enables the PWM Enhanced Mode. | | | |
| | Value | Name | | Description |
| | 0 | ENHANCED_DISABLED | | Disable PWM Enhanced Mode. |
| | 1 | ENHANCED_ENABLED | | Enable PWM Enhanced Mode. |
| *6:3* | *Reserved* | *Must write reset value.* | | |
| 2:0 | PWMSS | 0x0 | RW | **PWM Source Select.** |
| | Selects the PCA channel to use for the fine-tuning control signal. | | | |
| | Value | Name | | Description |
| | 0x0 | FINE_TUNE_CEX0 | | CEX0 selected as the fine-tuning control signal. |
| | 0x1 | FINE_TUNE_CEX1 | | CEX1 selected as the fine-tuning control signal. |
| | 0x2 | FINE_TUNE_CEX2 | | CEX2 selected as the fine-tuning control signal. |

# 15. Comparator (CMP0)

## 15.1 Introduction

An analog comparator is used to compare the voltage of two analog inputs, with a digital output indicating which input voltage is higher. External input connections to device I/O pins and internal connections are available through separate multiplexers on the positive and negative inputs. Hysteresis, response time, and current consumption may be programmed to suit the specific needs of the application.

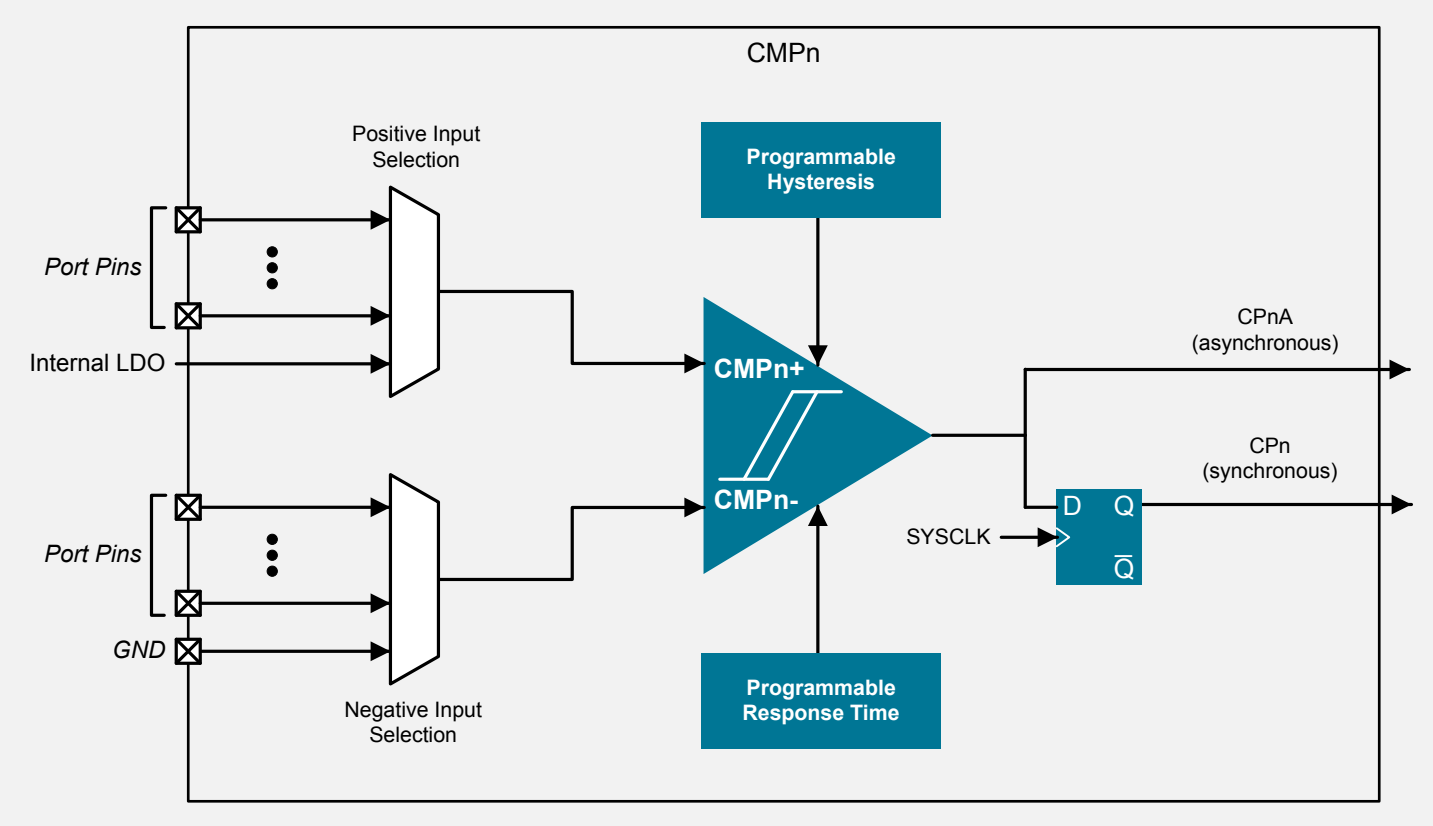



**Figure 15.1. Comparator Block Diagram**

## 15.2 Features

The comparator module includes the following features:

- Input options in addition to the pins:
  - Capacitive Sense Comparator output.
  - VDD.
  - VDD divided by 2.
  - Internal connection to LDO output.
  - Direct connection to GND.
- Synchronous and asynchronous outputs can be routed to pins via crossbar.
- Programmable hysteresis between 0 and ±20 mV.
- Programmable response time.
- Interrupts generated on rising, falling, or both edges.

## 15.3 Functional Description

### 15.3.1 Response Time and Supply Current

Response time is the amount of time delay between a change at the comparator inputs and the comparator's reaction at the output. The comparator response time may be configured in software via the CPMD field in the CMPnMD register. Selecting a longer response time reduces the comparator supply current, while shorter response times require more supply current.

### 15.3.2 Hysteresis

The comparator hysteresis is software-programmable via its Comparator Control register CMPnCN. 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 programmable using the CPHYN and CPHYP fields in the Comparator Control Register CMPnCN. The amount of negative hysteresis voltage is determined by the settings of the CPHYN bits. Settings of 20, 10, or 5 mV (nominal) 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 CPHYP bits.

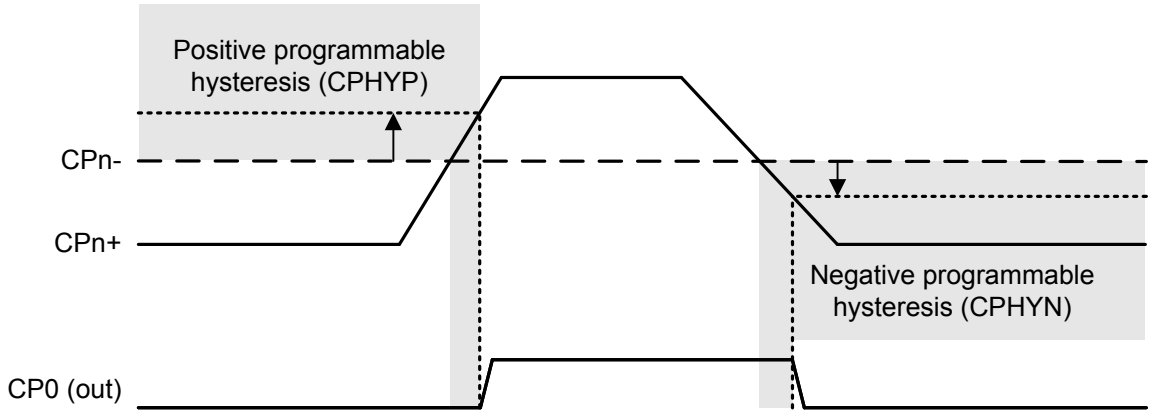



**Figure 15.2. Comparator Hysteresis Plot**

### 15.3.3 Input Selection

Comparator inputs may be routed to port I/O pins or internal signals. When connected externally, the comparator inputs can be driven from –0.25 V to (VDD) +0.25 V without damage or upset. The CMPnMX register selects the inputs for the associated comparator. The CMXP field selects the comparator's positive input (CPnP.x) and the CMXN field selects the comparator's negative input (CPnN.x).
**Note:** Any 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.

#### 15.3.3.1 Multiplexer Channel Selection

**Table 15.1. CMP0 Positive Input Multiplexer Channels**

| CMXP Setting in Register CMP0MX | Signal Name | Enumeration Name | QSOP24 Pin Name | QFN24 Pin Name | QFN20 Pin Name |
|---|---|---|---|---|---|
| 0000 - 0011 | CMP0P.0 - CMP0P.3 | | Reserved | Reserved | Reserved |
| 0100 | CMP0P.4 | CMP0P4 | P1.0 | P1.0 | P1.0 |
| 0101 - 1011 | CMP0P.5 - CMP0P.11 | | Reserved | Reserved | Reserved |
| 1100 | CMP0P.12 | CS_COMPARE | Capacitive Sense Compare | | |
| 1101 | CMP0P.13 | VDD_DIV_2 | VDD divided by 2 | | |
| 1110 | CMP0P.14 | VDD | VDD Supply Voltage | | |
| 1111 | CMP0P.15 | NONE | No connection / Reserved | | |

**Table 15.2. CMP0 Negative Input Multiplexer Channels**

| CMXN Setting in Register CMP0MX | Signal Name | Enumeration Name | QSOP24 Pin Name | QFN24 Pin Name | QFN20 Pin Name |
|---|---|---|---|---|---|
| 0000 - 0011 | CMP0N.0 - CMP0N.3 | | Reserved | Reserved | Reserved |
| 0100 | CMP0N.4 | CMP0N4 | P1.1 | P1.1 | P1.1 |
| 0101 - 1011 | CMP0N.5 - CMP0N.11 | | Reserved | Reserved | Reserved |
| 1100 | CMP0N.12 | CS_COMPARE | Capacitive Sense Compare | | |
| 1101 | CMP0N.13 | VDD_DIV_2 | VDD divided by 2 | | |
| 1110 | CMP0N.14 | LDO_OUT | Internal 1.8V LDO output | | |
| 1111 | CMP0N.15 | GND | Ground | | |

### 15.3.4 Output Routing

The comparator's synchronous and asynchronous outputs can optionally be routed to port I/O pins through the port I/O crossbar. The output of either comparator may be configured to generate a system interrupt on rising, falling, or both edges. CMP0 may also be used as a reset source or as a trigger to kill a PCA output channel.

The output state of the comparator can be obtained at any time by reading the CPOUT bit. The comparator is enabled by setting the CPEN bit to logic 1, and is disabled by clearing this bit to logic 0. When disabled, the comparator output (if assigned to a port I/O pin via the crossbar) defaults to the logic low state, and the power supply to the comparator is turned off.

Comparator interrupts can be generated on both rising-edge and falling-edge output transitions. The CPFIF flag is set to logic 1 upon a comparator falling-edge occurrence, and the CPRIF flag is set to logic 1 upon the comparator rising-edge occurrence. Once set, these bits remain set until cleared by software. The comparator rising-edge interrupt mask is enabled by setting CPRIE to a logic 1. The comparator falling-edge interrupt mask is enabled by setting CPFIE to a logic 1.

False rising edges and falling edges may be detected when the comparator is first powered on or if changes are made to the hysteresis or response time control bits. Therefore, it is recommended that the rising-edge and falling-edge flags be explicitly cleared to logic 0 a short time after the comparator is enabled or its mode bits have been changed, before enabling comparator interrupts.

## 15.4 CMP0 Control Registers

### 15.4.1 CMP0CN0: Comparator 0 Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CPEN | CPOUT | CPRIF | CPFIF | CPHYP | | CPHYN | |
| Access | RW | R | RW | RW | RW | | RW | |
| Reset | 0 | 0 | 0 | 0 | 0x0 | | 0x0 | |
| SFR Page = 0x0; SFR Address: 0x9B | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | CPEN | 0 | RW | **Comparator Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Comparator disabled. |
| | 1 | ENABLED | | Comparator enabled. |
| 6 | CPOUT | 0 | R | **Comparator Output State Flag.** |
| | Value | Name | | Description |
| | 0 | POS_LESS_THAN_NEG | | Voltage on CP0P < CP0N. |
| | 1 | POS_GREATER_THAN_NEG | | Voltage on CP0P > CP0N. |
| 5 | CPRIF | 0 | RW | **Comparator Rising-Edge Flag.**<br>Must be cleared by firmware. |
| | Value | Name | | Description |
| | 0 | NOT_SET | | No comparator rising edge has occurred since this flag was last cleared. |
| | 1 | RISING_EDGE | | Comparator rising edge has occurred. |
| 4 | CPFIF | 0 | RW | **Comparator Falling-Edge Flag.**<br>Must be cleared by firmware. |
| | Value | Name | | Description |
| | 0 | NOT_SET | | No comparator falling edge has occurred since this flag was last cleared. |
| | 1 | FALLING_EDGE | | Comparator falling edge has occurred. |
| 3:2 | CPHYP | 0x0 | RW | **Comparator Positive Hysteresis Control.** |
| | Value | Name | | Description |
| | 0x0 | DISABLED | | Positive Hysteresis disabled. |
| | 0x1 | ENABLED_MODE1 | | Positive Hysteresis = Hysteresis 1. |
| | 0x2 | ENABLED_MODE2 | | Positive Hysteresis = Hysteresis 2. |
| | 0x3 | ENABLED_MODE3 | | Positive Hysteresis = Hysteresis 3 (Maximum). |
| 1:0 | CPHYN | 0x0 | RW | **Comparator Negative Hysteresis Control.** |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | Value | Name | | Description |
| | 0x0 | DISABLED | | Negative Hysteresis disabled. |
| | 0x1 | ENABLED_MODE1 | | Negative Hysteresis = Hysteresis 1. |
| | 0x2 | ENABLED_MODE2 | | Negative Hysteresis = Hysteresis 2. |
| | 0x3 | ENABLED_MODE3 | | Negative Hysteresis = Hysteresis 3 (Maximum). |

### 15.4.2 CMP0MD: Comparator 0 Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | CPRIE | CPFIE | Reserved | | CPMD | |
| Access | RW | R | RW | RW | R | | RW | |
| Reset | 0x2 | | 0 | 0 | 0x0 | | 0x2 | |
| SFR Page = 0x0; SFR Address: 0x9D | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:6* | *Reserved* | *Must write reset value.* | | |
| 5 | CPRIE | 0 | RW | **Comparator Rising-Edge Interrupt Enable.** |
| | Value | Name | | Description |
| | 0 | RISE_INT_DISABLED | | Comparator rising-edge interrupt disabled. |
| | 1 | RISE_INT_ENABLED | | Comparator rising-edge interrupt enabled. |
| 4 | CPFIE | 0 | RW | **Comparator Falling-Edge Interrupt Enable.** |
| | Value | Name | | Description |
| | 0 | FALL_INT_DISABLED | | Comparator falling-edge interrupt disabled. |
| | 1 | FALL_INT_ENABLED | | Comparator falling-edge interrupt enabled. |
| *3:2* | *Reserved* | *Must write reset value.* | | |
| 1:0 | CPMD | 0x2 | RW | **Comparator Mode Select.** |
| | These bits affect the response time and power consumption of the comparator. | | | |
| | Value | Name | | Description |
| | 0x0 | MODE0 | | Mode 0 (Fastest Response Time, Highest Power Consumption) |
| | 0x1 | MODE1 | | Mode 1 |
| | 0x2 | MODE2 | | Mode 2 |
| | 0x3 | MODE3 | | Mode 3 (Slowest Response Time, Lowest Power Consumption) |

### 15.4.3 CMP0MX: Comparator 0 Multiplexer Selection

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CMXN | | | | CMXP | | | |
| Access | RW | | | | RW | | | |
| Reset | 0x4 | | | | 0x4 | | | |
| SFR Page = 0x0; SFR Address: 0x9F | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:4 | CMXN | 0x4 | RW | **Comparator Negative Input MUX Selection.**<br>This field selects the negative input for the comparator. |
| 3:0 | CMXP | 0x4 | RW | **Comparator Positive Input MUX Selection.**<br>This field selects the positive input for the comparator. |

# 16. Capacitive Sense (CS0)

## 16.1 Introduction

The Capacitive Sense subsystem uses a capacitance-to-digital circuit to determine the capacitance on a port pin. The module can take measurements from different port pins using the module's analog multiplexer. The module can be configured to take measurements on one port pin, a group of port pins one-by-one using auto-scan, or the total capacitance on multiple channels together. A selectable gain circuit allows the designer to adjust the maximum allowable capacitance. An accumulator is also included, which can be configured to average multiple conversions on an input channel. Interrupts can be generated when the CS0 peripheral completes a conversion or when the measured value crosses a configurable threshold.

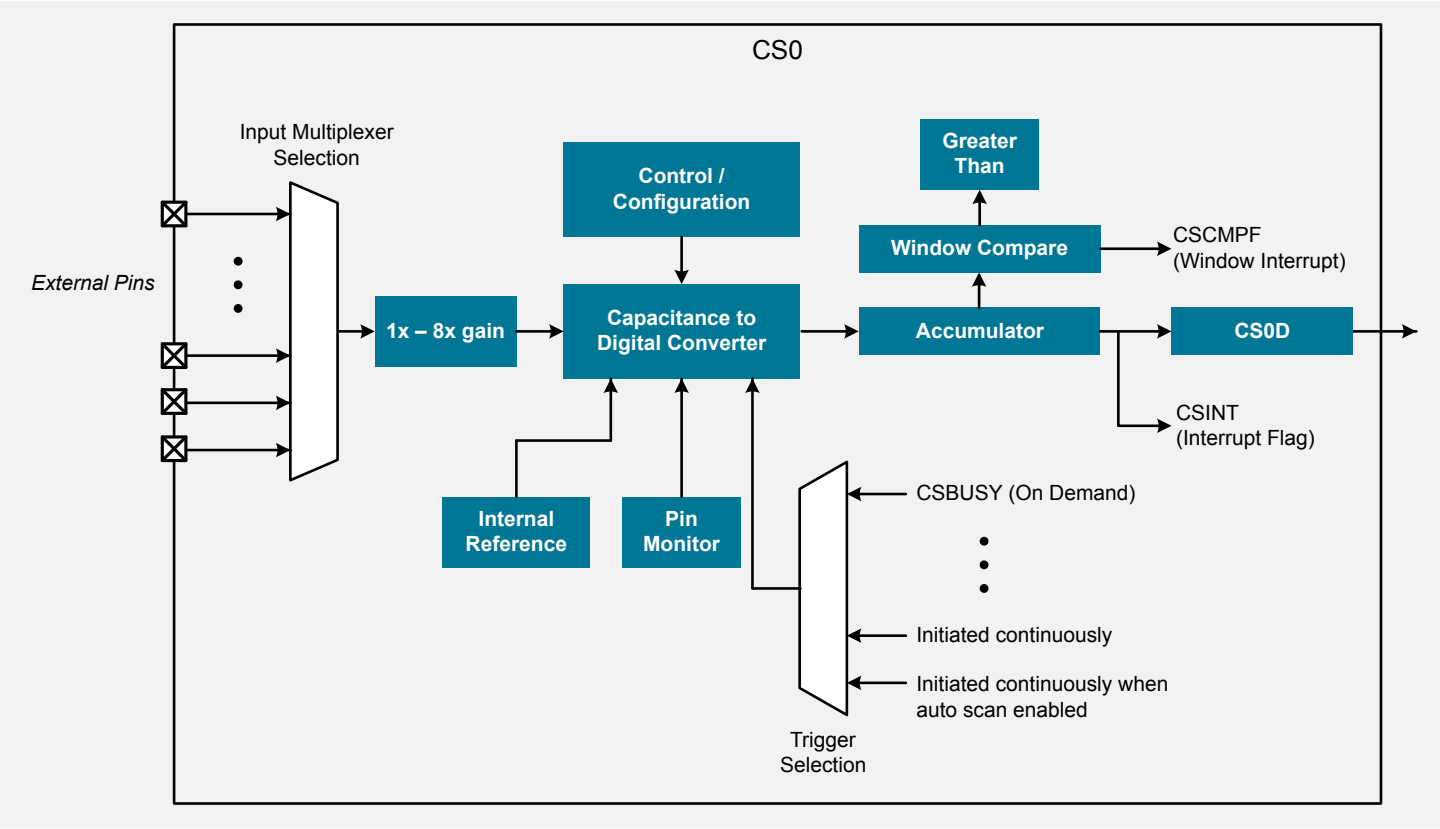



**Figure 16.1. Capacitive Sense Block Diagram**

## 16.2 Features

The Capacitive Sense module includes the following features:

- Measure multiple pins one-by-one using auto-scan or total capacitance on multiple channels together.
- Configurable input gain.
- Hardware auto-accumulate and average.
- Multiple internal start-of-conversion sources.
- Operational in Suspend when all other clocks are disabled.
- Interrupts available at the end of a conversion or when the measured value crosses a configurable threshold.

## 16.3 Functional Description

### 16.3.1 Port I/O Configuration

In order for a port pin to be measured by the CS0 module, that port pin must be configured as an analog input. Configuring the input multiplexer to a port pin not configured as an analog input will cause the capacitive sense comparator to output incorrect measurements.

Pins are selected as inputs for capacitive sense using the input multiplexer. This multiplexer can be controlled through two methods: Firmware can write to the CS0MX register directly, or the register can be configured automatically using the module auto-scan functionality (see for more information ).

#### 16.3.1.1 Multiplexer Channel Selection

**Table 16.1. CS0 Input Multiplexer Channels**

| CS0MX setting | Signal Name | Enumeration Name | QSOP24 Pin Name | QFN24 Pin Name | QFN20 Pin Name |
|---|---|---|---|---|---|
| 0000 | CS0.0 | CS0P0 | P0.0 | P0.0 | P0.0 |
| 0001 | CS0.1 | CS0P1 | P0.1 | P0.1 | P0.1 |
| 0010 | CS0.2 | CS0P2 | P0.2 | P0.2 | P0.2 |
| 0011 | CS0.3 | CS0P3 | P0.3 | P0.3 | P0.3 |
| 0100 | CS0.4 | CS0P4 | P0.4 | P0.4 | P0.4 |
| 0101 | CS0.5 | CS0P5 | P0.5 | P0.5 | P0.5 |
| 0110 | CS0.6 | CS0P6 | P0.6 | P0.6 | P0.6 |
| 0111 | CS0.7 | CS0P7 | P0.7 | P0.7 | P0.7 |
| 1000 | CS0.8 | CS0P8 | P1.0 | P1.0 | P1.0 |
| 1001 | CS0.9 | CS0P9 | P1.1 | P1.1 | P1.1 |
| 1010 | CS0.10 | CS0P10 | P1.2 | P1.2 | P1.2 |
| 1011 | CS0.11 | CS0P11 | P1.3 | P1.3 | P1.3 |
| 1100 | CS0.12 | CS0P12 | P1.4 | P1.4 | Reserved |
| 1101 | CS0.13 | CS0P13 | P1.5 | P1.5 | P1.5 |
| 1110 - 1111 | CS0.14 - CS0.15 | | Reserved | | |

### 16.3.2 Initializing the Capacitive Sensing Peripheral

The following procedure is recommended for properly initializing the capacitive sense peripheral:

1. Enable the capacitive sense module (CSEN = 1) before performing any other initializations.
2. Initialize the start-of-conversion source (CS0CM) to the desired mode.
3. Continue initializing all remaining CS0 registers.

### 16.3.3 Start-of-Conversion Sources

A capacitive sense conversion can be initiated in one of multiple ways, depending on the setting of the CS0CM field in the CS0CF register. Conversions may be initiated by one of the following:

1. Writing a 1 to the CSBUSY bit of register CS0CN0.
2. Timer 0 overflow.
3. Timer 2 overflow.
4. Timer 1 overflow.
5. Timer 3 overflow.
6. Convert continuously.
7. Convert continuously with auto-scan enabled.
8. Perform a single scan of all enabled channels Conversions can be configured to be initiated continuously through one of two methods.

CS0 can be configured to convert at a single channel continuously or it can be configured to convert continuously with autoscan enabled. When configured to convert continuously, conversions will begin after firmware sets the CSBUSY bit in CS0CN0 register to 1. An interrupt will be generated if CS0 conversion complete interrupts are enabled. Single scan mode allows all channels enabled in the CS0SCAN0 and CS0SCAN1 registers to be scanned in a single pass. An end of scan interrupt can be enabled to trigger once all selected channels have been converted.

The CS0 module uses a method of successive approximation to determine the value of an external capacitance. The number of bits the CS0 module converts is adjustable using the CS0CR field in the CS0MD2 register. Conversions are 13 bits long by default, but they can be adjusted to 12, 13, 14, or 16 bits depending on the needs of the application. Unconverted bits will be set to 0. Shorter conversion lengths produce faster conversion rates, and vice-versa. Applications can take advantage of faster conversion rates when the unconverted bits fall below the noise floor.
**Note:** CS0 conversion complete interrupt behavior depends on the settings of the CS0 accumulator. If CS0 is configured to accumulate multiple conversions on an input channel, a CS0 conversion complete interrupt will be generated only after the last conversion completes.

### 16.3.4 Multiple Channel Enable

The capacitive sense module has the capability of measuring the total capacitance of multiple channels using a single conversion. When the multiple channel feature is enabled (CS0MCEN = 1), Channels selected by CS0SCAN0/1 are internally shorted together, and the combined node is selected as the capacitive sense input. This mode can be used to detect a capacitance change on multiple channels using a single conversion and is useful for implementing low power mode wake on multiple channels.

### 16.3.5 Gain Adjustment

The gain of the CS0 circuit can be adjusted in integer increments from 1x to 8x, where 8x is the default. High gain gives the best sensitivity and resolution for small capacitors, such as those typically implemented as touch-sensitive PCB features. To measure larger capacitance values, the gain should be lowered accordingly using the CS0CG field in the CS0MD1 register.

### 16.3.6 Using CS0 with Low Power Modes

#### Waking from Suspend

CS0 has the capability of waking the device from a low power Suspend mode upon detection of a touch using the digital comparator. When channel scan masking is enabled, CS0 may also wake up the device after an end of scan event when in single scan mode or after each conversion when in one of the continuously scanning modes. If the accumulate feature is enabled, the device wakes up after all samples have been accumulated. The CS0WOI bit in the CS0MD1 register can be used to configure desire wake from suspend behavior.

#### Using CS0 in Applications that Utilize Sleep Mode

To achieve maximum power efficiency, CS0 should be enabled only when taking a conversion and disabled at all other times. CS0 must be disabled by software prior to entering Sleep mode.

### 16.3.7 Automatic Scanning

#### Method 1—Channel Scan Masking Disabled

CAPSENSE0 can be configured to automatically scan a sequence of contiguous input channels by configuring and enabling autoscan. Using autoscan with the comparator interrupt enabled allows a system to detect a change in measured capacitance without requiring any additional dedicated MCU resources. Autoscan is enabled using the the start-of-conversion field CS0CM in the CS0CF register. After enabling autoscan, the starting and ending channels should be set to appropriate values in CS0SS and CS0SE, respectively. Writing to CS0SS when autoscan is enabled will cause the value written to CS0SS to be copied into CS0MX. After enabling autoscan, firmware can write a 1 to CSBUSY to start autoscan conversions. When autoscan completes the number of conversions defined in the accumulator field CS0ACU, autoscan configures the input multiplexer (CS0MX) to the next sequential port pin configured as an analog input and begins a conversion on that channel. All other pins between CS0SS and CS0SE which are set as analog inputs are grounded during the conversion. This scan sequence continues until CS0MX reaches the ending input channel value defined in CS0SE. After one or more conversions have been taken at this channel, autoscan configures CS0MX back to the starting input channel.
**Note:** Autoscan attempts one conversion on an input channel regardless of whether that channel's port pin has been configured as an analog input. Autoscan will also complete the current rotation when the device is halted for debugging.

If autoscan is enabled when the device enters Suspend mode, autoscan will remain enabled and running. This feature allows the device to wake from suspend through a capacitive sense greater-than comparator event on any configured capacitive sense input included in the autoscan sequence of inputs.

As an example:

```
CS0CN0 = CS0CN0_CSEN__ENABLED;  // Enables the module
CS0CF = CS0CF_CS0CM__AUTO_SCAN;  // Enables autoscan as the start of conversion source
CS0SS = 0x02;  // Sets P0.2 as the autoscan starting channel
CS0SE = 0x0B;  // Sets P1.3 as the autoscan starting channel
P0MDIN = P0MDIN_B0__ANALOG |
         P0MDIN_B1__DIGITAL |
         P0MDIN_B2__ANALOG |
         P0MDIN_B3__ANALOG |
         P0MDIN_B4__DIGITAL |
         P0MDIN_B5__DIGITAL |
         P0MDIN_B6__DIGITAL |
         P0MDIN_B7__DIGITAL;
P1MDIN = P1MDIN_B0__ANALOG |
         P1MDIN_B1__ANALOG |
         P1MDIN_B2__DIGITAL |
         P1MDIN_B3__ANALOG |
         P1MDIN_B4__ANALOG |
         P1MDIN_B5__ANALOG |
         P1MDIN_B6__DIGITAL |
         P1MDIN_B7__DIGITAL;
```

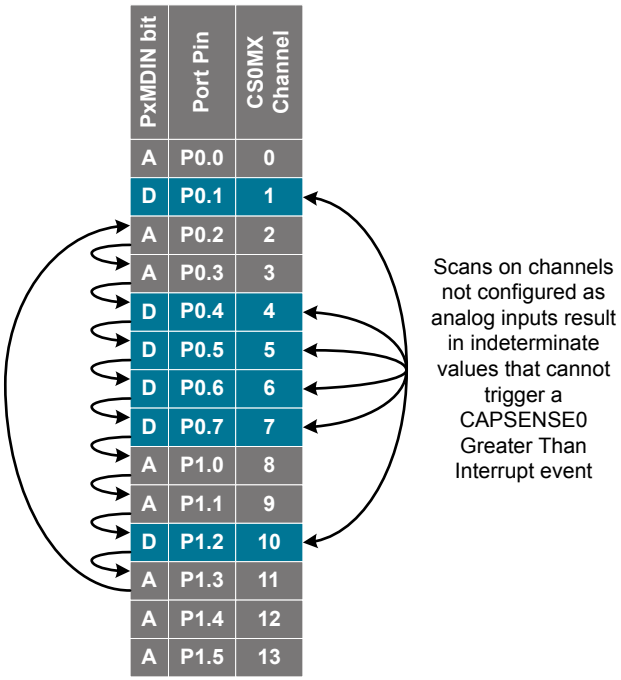



**Figure 16.2. Autoscan Example**

**Method 1—Channel Scan Masking Enabled**

When channel scan masking is enabled (CS0SMEN = 1), the capacitive sense module uses an alternate autoscanning method that uses the contents of CS0SCAN0 and CS0SCAN1 to determine which channels to include in the scan. This maximizes flexibility for application development and can result in more energy-efficient scanning. The following procedure can be used to configure the device for automatic scanning with channel scan masking enabled:

1. Set the CS0SMEN bit to 1.
2. Select the start of conversion mode (CS0CM) if not already configured. Single scan mode is the mode of choice for most systems.
3. Configure the CS0SCAN0 and CS0SCAN1 registers to enable channels in the scan.
4. Configure the digital comparator threshold (CS0THH:L) and polarity (CS0POL).
5. Enable wake from Suspend on end of scan (CS0WOI = 1) if this functionality is desired.
6. Set CS0SS to point to the first channel in the scan.
   **Note:** CS0SS uses the same bit mapping as the CS0MX register.
7. Issue a start of conversion (CSBUSY = 1).
8. (Optional) Enable the CAPSENSE0 wakeup source and place the device in Suspend mode.

If using single-scan mode, scanning will stop once a touch (value above the programmed threshold) has been detected using the digital comparator. The input multiplexer (CS0MX) register will contain the channel mux value of the channel that caused the interrupt. Setting the busy bit when servicing the interrupt will cause the scan to continue where it left off. Scanning will also stop after all channels have been sampled and no touches have been detected. If the CS0WOI bit is set, a wake from suspend event will be generated.
**Note:** When automatic scanning is enabled, the contents of the CS0MX register are only valid when the digital comparator interrupt is set and CSBUSY = 0.

### 16.3.8 Comparator

The CS0 comparator compares the latest capacitive sense conversion result with the value stored in the threshold (CS0THH:L). If the result is less than or equal to the stored value, hardware clears the CSCMPF bit in the CS0CN0 register to 0. If the result is greater than the stored value, hardware sets CSCMPF to 1 and an interrupt will be generated if the greater-than comparator interrupts are enabled. If the conversion accumulator is configured to accumulate multiple conversions, a comparison will not be made until the last conversion has been accumulated. If autoscan is running when the comparator sets the CSCMPF bit, no further autoscan-initiated conversions will start until firmware sets CSBUSY to 1.

A greater-than comparator event can wake a device from Suspend mode. This feature is useful in systems configured to continuously sample one or more capacitive sense channels. The device will remain in the low-power suspend state until the captured value of one of the scanned channels causes a greater-than comparator event to occur. It is not necessary to have capacitive sense comparator interrupts enabled in order to wake a device from Suspend with a greater-than event.

### 16.3.9 Conversion Accumulator

The CAPSENSE0 module can be configured to accumulate multiple conversions on an input channel. The accumulator can accumulate 1, 4, 8, 16, 32, or 64 samples. After the defined number of samples have been accumulated, the result is divided by either 1, 4, 8, 16, 32, or 64 depending on the accumulation setting and copied to the data output registers.

**Table 16.2. Operation with Autoscan and Accumulate**

| Autoscan Enabled | Accumulator Enabled | Converstion Complete Interrupt Behavior (CSINT) | Greater-Than Interrupt Behavior (CSCMPF) | Input Multiplexer Behavior |
|---|---|---|---|---|
| N | N | CSINT Interrupt serviced after 1 conversion completes | Interrupt serviced after 1 conversion completes if value in the data registers is greater than the threshold | Input multiplexer unchanged |
| N | Y | CSINT Interrupt serviced after M conversions complete | Interrupt serviced after M conversions complete if value in the data registers (post accumulate and divide) is greater than threshold | Input multiplexer unchanged |
| Y | N | CSINT Interrupt serviced after 1 conversion completes | Interrupt serviced after conversion completes if value in the data registers is greater than the threshold; Autoscan stopped | If greater-than comparator detects conversion value is greater than the threshold, the input multiplexer is left unchanged; otherwise, the input multiplexer updates to the next channel (CS0MX + 1) and wraps back to the starting channel after passing the ending channel |
| Y | Y | CSINT Interrupt serviced after M conversions complete | Interrupt serviced after M conversions complete if value in the data registers (post accumulate and divide) is greater than the threshold; Autoscan stopped | If greater-than comparator detects conversion value is greater than the threshold, the input multiplexer is left unchanged; otherwise, the input multiplexer updates to the next channel (CS0MX + 1) and wraps back to the starting channel after passing the ending channel |

**Note:** M = Accumulator setting (1x, 4x, 8x, 16x, 32x, 64x).

### 16.3.10 Pin Monitor

The CS0 module provides accurate conversions in all operating modes of the core, peripherals, and I/O ports. Pin monitoring circuits are provided to improve interference immunity from high-current output pin switching.

Conversions in the CS0 module are immune to any change on digital inputs and immune to most output switching. Even high-speed serial data transmission will not affect CS0 operation as long as the output load is limited. Output changes that switch large loads such as LEDs and heavily-loaded communications lines can affect conversion accuracy. For this reason, the CS0 module includes pin monitoring circuits that will, if enabled, automatically adjust conversion timing if necessary to eliminate any effect from high-current output pin switching.

The pin monitor enable bit should be set for any output signal that is expected to drive a large load. For example, if the SMBus in a system is heavily loaded with multiple slaves and a long PCB route, enable the SMBus pin monitor. In another example, Timer2 controls an LED on Port 1, pin 3 to provide variable dimming. Set the port I/O pin monitor enable PIOPM to 1. In a final example, the SPI bus is used to communicate to a nearby host. The pin monitor is not needed because the output is not heavily loaded, so firmware should keep SPIPM at 0, the default reset state.

Pin monitors should not be enabled unless they are required. The pin monitor works by repeating any portion of a conversion that may have been corrupted by a change on an output pin. Setting pin monitor enable bits will slow CS0 conversions. The frequency of CS0 retry operations can be limited by setting the CSPMMD field. In the default (reset) state, all converter retry requests will be performed, which is the recommended setting for all applications. The number of retries per conversion can be limited to either two or four retries by changing CSPMMD. Limiting the number of retries per conversion ensures that even in circumstances where extremely frequent high-power output switching occurs, conversions will be completed, though there may be some loss of accuracy due to switching noise.

Activity of the pin monitor circuit can be detected by reading the CSPME bit in the CS0CN0 register. This bit will be set if any CS0 converter retries have occurred. It remains set until cleared by firmware or a device reset.

### 16.3.11 Other Considerations

There are several configuration options in the CS0 module designed to modify the operation of the circuit and address special situations. In particular, any circuit with more than 500 ohms of series impedance between the sensor and the device pin may require adjustments for optimal performance. Typical applications which may require adjustments include the following:

- Touch panel sensors fabricated using a resistive conductor such as indium-tin-oxide (ITO).
- Circuits using a high-value series resistor to isolate the sensor element for high ESD protection.

Most systems will require no fine tuning, and the default settings for CS0DT, CS0DR, CS0IA, CS0RP and CS0LP should be used.

#### Adjusting the Reset Timing

The CS0 module determines capacitance by discharging an external capacitor and then measuring how quickly that capacitor charges. In order to do this, the external capacitor must be fully discharged before every test. There are two timers inside the capacitive sense module that determine the timing for the reset (discharge) operation.

CS0 performs a two-stage discharge (double reset) of the external capacitor at the start of every bit conversion to improve performance in high-noise environments. In this method, most of the charge in the external capacitor is removed in a first reset stage through a low-resistance switch to ground. A second reset is then performed using a high-resistance switch to ground. This second reset removes any ambient noise energy that might have been captured in the external capacitor at the end of the first reset stage.

The lengths of both reset periods are independently adjustable. Longer periods are used when the external capacitor is separated from the CS0 module by a large resistor (more than 500 Ω) because that series resistor would slow the rate of discharge.

Determining the appropriate settings for CS0DT (the primary reset) and CS0DR (the secondary reset) are two of a series of related adjustments which must be made when using CS0 to measure capacitive loads in the presence of high resistance.

### Adjusting Primary Reset Timing: CS0DT

Primary reset timing adjustment is performed to provide peak sensitivity for highly-resistive loads and peak linearity for capacitive loads linked thorough a distributed resistance (such as an ITO touch panel) while minimizing the required conversion time. The adjustment for CS0DT should be performed while the CS0DR and CS0IA fields are set at their maximum values (CS0DR = 11b, CS0IA = 001b).

1. Begin the adjustment with CS0DT set to maximum delay (CS0DT = 111b). Measure the untouched average capacitive sense result for the channel under test.
2. Record the average touched capacitive sense value with CS0DT = 111b. The touched value should be higher than the untouched value. The magnitude of the difference between the touched and untouched average capacitive sense values is the figure of merit for touch sensitivity.
3. Decrease the primary reset time CS0DT by one and repeat the touched and untouched capacitive sense measurements. Repeat this step until values have been recorded for all eight CS0DT settings. As the CS0DT setting decreases, the average sensitivity of the measured value may begin to decrease significantly.
4. CS0DT should be set high enough that there is not a significant decrease in sensitivity due to resistance. Select the CS0DT setting that occurred prior to the observed drop in measured touch sensitivity.

### Adjusting Secondary Reset Timing: CS0DR

Adjustments for CS0IA and CS0DT should be set to their maximum value (CS0IA = 001b, CS0DT = 111b) while the CS0DR adjustments are being performed. Because the only function of CS0DR is to reduce the effect of environmental noise, establishing the proper CS0DR adjustment can only be performed in a test environment with the highest expected level of ambient noise while connected to the sensor which is specific to the intended application. Increasing the CS0DR adjustment does not increase the level of possible noise rejection; it only changes the amount of time that the capacitive sense module will wait for the secondary reset circuit to finish its noise-reduction operation. Resistive sensors require longer CS0DR operating periods, and their CS0DR settings will be necessarily higher. Higher settings for CS0DR cause the capacitive sense conversion process to slow substantially. The adjustment method is intended to find the lowest (fastest) CS0DR setting that delivers full function.

1. Begin the adjustment with CS0DR set to maximum delay (CS0DR = 11b). Record a series of CS0 output values for the sensor when it is being touched.
2. For this test, the standard deviation of data in the series is the figure of merit used to define the level of noise received by the CS0 converter. The secondary reset circuit reduces noise. An increase in standard deviation indicates that the secondary reset circuit is no longer working optimally. The best adjustment point for CS0DR is the lowest setting for which there is an acceptably-low standard deviation.
3. Decrease the value of the CS0DR setting by one (from 11b to 10b). Record a new data set and determine its standard deviation. Repeat this process for CS0DR settings 01b and 00b. Compare the standard deviations calculated for the four CS0DR settings. Select the lowest CS0DR setting for which there is not a significant increase in standard deviation.

### Adjusting CS0 Ramp Timing: CS0IA

In the presence of larger series resistors between the device pin and the capacitive sensor, it is necessary to also adjust the ramp time for the CS0 conversion. This is done by using CS0IA to modify the source current used to charge up the capacitive sensor. If this source current and the series impedance are both high, the CS0 module will see less of the capacitor on the other side of the impedance. Reducing the current allows the pin voltage to more directly reflect the voltage at the capacitive sensor.

The adjustment for CS0IA should be performed while CS0DR and CS0DT bits are set at their maximum values (CS0DR = 11b, CS0DT = 111b).

1. Begin the adjustment with CS0IA set to minimum current (CS0IA = 001b). Measure the untouched average CS0 result for the channel under test.
2. Record the average touched CS0 value with CS0IA = 001b. The touched value should be higher than the untouched value. The magnitude of the difference between the touched and untouched average CS0 values is the figure of merit for touch sensitivity.
3. Increase the current control CS0IA by one and repeat the touched and untouched CS0 measurements. Repeat this step until values have been recorded for all eight CS0IA settings (including CS0IA = 000b, which is the highest current setting). As CS0IA is changed, the average sensitivity of the CS0 value may begin to decrease significantly.
4. CS0IA should be chosen such that there is not a significant decrease in sensitivity due to resistance. Select the CS0IA setting that occurred prior to the observed drop in CS0 touch sensitivity.

### Low-Pass Filter Adjustments

A programmable active low-pass filter is provided to limit external noise interference in CS0 operation. The filter is programmable in two ways. The filter can be tailored for optimal performance with slow-rising signals by adjusting the low-pass filter ramping control, CS0RP. Another control, CS0LP, allows the user to adjust the filter's corner frequency. For most applications, the default settings for the filter controls (CS0LP = 000b, CS0RP = 00b) should be used.

### Adjusting CS0 Ramp Timing: CS0RP

Determining the appropriate setting for CS0RP is one of the last in a series of related adjustments to be made for high-resistance loads. It requires that the adjustments for gain (CS0CG), output current (CS0IA), and the reset timing (CS0DT and CS0DR) have already been made. The adjustment values determined for those settings should be programmed into the CS0 module when performing the CS0RP adjustment. Configure the CS0 module to perform continuously repeated capacitance sensing operations. Using an oscilloscope, measure the maximum rise time seen on the CS0 sensor pin and subtract 200 ns. This is the CS0 ramp time for this channel.

### Adjusting CS0LP for Non-Default CS0RP Settings

The default setting for the low-pass filter corner frequency (CS0LP = 000b) gives the best sensing response for all applications using default ramp timing (CS0RP = 00b). For applications with slower ramp timing, the corner frequency should always be modified to match the edge-rate of the input ramp. For all non-default settings of CS0RP (CS0RP = 01b, 10b or 11b), set CS0LP = 001b.

### Other Options for Adjusting CS0LP

In some circumstances, it may be preferable to trade CS0 sensitivity for increased noise filtering. Decreasing the filter's corner frequency below the natural ramp rate of the converter will cause a lower capacitance value to be reported. The change in capacitance due to a touch event will also be attenuated. As a result, lowering the corner frequency will not necessarily increase the signal-to-noise ratio for capacitive touch events.

Although signal-to-noise is the figure of merit for this adjustment, there may be acceptable trade-offs in the adjustment of CS0LP which result in an overall lower SNR but better operation over a wide range of environmental conditions. Some applications may call for adaptive changes to the corner frequency based on measurements of input noise, trading sensitivity for noise rejection only when necessary. Because this optional adjustment requires a subjective trade-off between noise rejection and sensitivity, the ultimate determination of acceptable results for this adjustment will be determined by the end application.

When performing these tests, all other CS0 configuration registers should be properly adjusted for the channel under test. CS0LP operation can only be analyzed when the CS0 is otherwise optimally adjusted. CS0LP adjustments should only be performed during performance tuning for a specific application in a well-defined noise environment.

CS0LP settings adjust the CS0 response to environmental noise. As in the adjustment of CS0DR settings, CS0LP adjustment can only be performed in a test environment with the highest expected level of ambient noise while connected to the sensor which is specific to the intended application.

Higher settings for CS0LP cause the low-pass filter corner frequency to drop. Noise will be reduced and reported capacitance will be reduced. This adjustment process incrementally increases CS0LP settings to determine which, if any, of the settings provide a higher SNR. For this test, the optimum setting for CS0LP will provide higher SNR results for the system in this high-interference environment, although the same setting is likely to reduce SNR for the same system in a low-interference environment.

1. Begin the adjustment with CS0LP set to maximum corner frequency (CS0LP = 000b). Record a series of CS0 output values for the sensor when it is untouched and record another series of CS0 output values from the sensor when it is being touched.
2. Increase the value of the CS0LP setting by one (from 000b to 001b). Record a new data set and determine its SNR. Repeat this process for all remaining CS0LP settings.
3. This series of tests can be repeated in a variety of noise environments. Comparison of the resulting SNR tables can then be used to determine how CS0LP adjustments might be used to improve capacitive sensing in high-interference environments.

## 16.4 CS0 Control Registers

### 16.4.1 CS0CN0: Capacitive Sense 0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CSEN | CSEOS | CSINT | CSBUSY | CSCMPEN | Reserved | CSPME | CSCMPF |
| Access | RW | R | RW | RW | RW | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = 0x0; SFR Address: 0xB0 (bit-addressable)

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | CSEN | 0 | RW | **CS0 Enable.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | CS0 disabled and in low-power mode. |
| | 1 | ENABLED | | CS0 enabled and ready to convert. |
| 6 | CSEOS | 0 | R | **CS0 End of Scan Interrupt Flag.** |
| | This bit must be cleared by firmware. | | | |
| | Value | Name | | Description |
| | 0 | NOT_SET | | CS0 has not completed a scan since the last time CS0EOS was cleared. |
| | 1 | SET | | CS0 has completed a scan. |
| 5 | CSINT | 0 | RW | **CS0 Interrupt Flag.** |
| | This bit must be cleared by firmware. | | | |
| | Value | Name | | Description |
| | 0 | NOT_SET | | CS0 has not completed a data conversion since the last time CS0INT was cleared. |
| | 1 | SET | | CS0 has completed a data conversion. |
| 4 | CSBUSY | 0 | RW | **CS0 Busy.** |
| | Read: A 1 indicates a CS0 conversion is in progress. | | | |
| | Write: Writing a 1 to this bit initiates a CS0 conversion if CS0CM[2:0] = 000b, 110b, or 111b. | | | |
| 3 | CSCMPEN | 0 | RW | **CS0 Digital Comparator Enable.** |
| | Enables the digital comparator, which compares accumulated CS0 conversion output to the value stored in CS0THH:CS0THL. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable CS0 digital comparator. |
| | 1 | ENABLED | | Enable CS0 digital comparator. |
| *2* | *Reserved* | *Must write reset value.* | | |
| 1 | CSPME | 0 | R | **CS0 Pin Monitor Event.** |
| | Set if any converter re-tries have occurred due to a pin monitor event. This bit must be cleared by firmware. | | | |
| 0 | CSCMPF | 0 | R | **CS0 Digital Comparator Interrupt Flag.** |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | Value | Name | | Description |
| | 0 | NOT_SET | | CS0 result is smaller than the value set by CS0THH and CS0THL since the last time CS0CMPF was cleared. |
| | 1 | SET | | CS0 result is greater than the value set by CS0THH and CS0THL since the last time CS0CMPF was cleared. |

### 16.4.2 CS0CF: Capacitive Sense 0 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0SMEN | CS0CM | | | CS0MCEN | CS0ACU | | |
| Access | RW | RW | | | R | RW | | |
| Reset | 0 | 0x0 | | | 0 | 0x0 | | |
| SFR Page = 0x0; SFR Address: 0xAA | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | CS0SMEN | 0 | RW | **CS0 Channel Scan Masking Enable.** |
| | Value | Name | | Description |
| | 0 | SCAN_MASK_DISABLED | | The CS0SCAN0 and CS0SCAN1 register contents are ignored. |
| | 1 | SCAN_MASK_ENABLED | | The CS0SCAN0 and CS0SCAN1 registers determine which channels will be included in the scan. |
| 6:4 | CS0CM | 0x0 | RW | **CS0 Start of Conversion Mode Select.** |
| | Value | Name | | Description |
| | 0x0 | CS0BUSY | | Conversion initiated on every write of 1 to CS0BUSY. |
| | 0x1 | TIMER0 | | Conversion initiated on overflow of Timer 0. |
| | 0x2 | TIMER2 | | Conversion initiated on overflow of Timer 2. |
| | 0x3 | TIMER1 | | Conversion initiated on overflow of Timer 1. |
| | 0x4 | TIMER3 | | Conversion initiated on overflow of Timer 3. |
| | 0x5 | SINGLE_SCAN | | When CS0SMEN is set to 1, the converter completes a Single Scan of the channels selected by CS0SCAN0/1. This setting is invalid when CS0SMEN is cleared to 0. |
| | 0x6 | SINGLE_CONTINUOUS | | Conversion initiated continuously on the channel selected by CS0MX after writing 1 to CS0BUSY. |
| | 0x7 | AUTO_SCAN | | When CS0SMEN is set to 1, the converter enters Auto Scan Mode and continuously scans the channels selected by CS0SCAN0/1. When CS0SMEN is cleared to 0, the converter scans continuously on channels from CS0SS to CS0SE after firmware writes 1 to CS0BUSY. |
| 3 | CS0MCEN | 0 | R | **CS0 Multiple Channel Enable.** |
| | Value | Name | | Description |
| | 0 | MULT_CHAN_DISABLED | | Multiple channel feature is disabled. |
| | 1 | MULT_CHAN_ENABLED | | Channels selected by CS0SCAN0/1 are internally shorted together and the combined node is selected as the CS0 input. This mode can be used to detect a capacitance change on multiple channels using a single conversion. |
| 2:0 | CS0ACU | 0x0 | RW | **CS0 Accumulator Mode Select.** |
| | Value | Name | | Description |
| | 0x0 | ACC_1 | | Accumulate 1 sample. |
| | 0x1 | ACC_4 | | Accumulate 4 samples. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | 0x2 | ACC_8 | | Accumulate 8 samples. |
| | 0x3 | ACC_16 | | Accumulate 16 samples |
| | 0x4 | ACC_32 | | Accumulate 32 samples. |
| | 0x5 | ACC_64 | | Accumulate 64 samples. |

### 16.4.3 CS0DH: Capacitive Sense 0 Data High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0DH | | | | | | | |
| Access | R | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xEE | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CS0DH | 0x00 | R | **CS0 Data High Byte.**<br>Stores the high byte of the last completed 16-bit Capacitive Sense conversion. |

### 16.4.4 CS0DL: Capacitive Sense 0 Data Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0DL | | | | | | | |
| Access | R | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xED | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CS0DL | 0x00 | R | **CS0 Data Low Byte.**<br>Stores the low byte of the last completed 16-bit Capacitive Sense conversion. |

### 16.4.5 CS0SCAN0: Capacitive Sense 0 Channel Scan Mask 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0SCAN0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xD2 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CS0SCAN0 | 0x00 | RW | **Channel Scan Mask for Port 0.**<br>The selected channels are included in Auto Scan when scan masking is enabled (CS0SMEN = 1). |

### 16.4.6 CS0SCAN1: Capacitive Sense 0 Channel Scan Mask 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | CS0SCAN1 | | | | | |
| Access | R | | RW | | | | | |
| Reset | 0x0 | | 0x00 | | | | | |
| SFR Page = 0x0; SFR Address: 0xD3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:6* | *Reserved* | *Must write reset value.* | | |
| 5:0 | CS0SCAN1 | 0x00 | RW | **Channel Scan Mask for Port 1.**<br>The selected channels are included in the Auto Scan when scan masking is enabled (CS0SMEN = 1). |

### 16.4.7 CS0SS: Capacitive Sense 0 Auto Scan Start Channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | CS0SS | | | | |
| Access | R | | | RW | | | | |
| Reset | 0x0 | | | 0x00 | | | | |
| SFR Page = 0x0; SFR Address: 0xDD | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:5* | *Reserved* | *Must write reset value.* | | |
| 4:0 | CS0SS | 0x00 | RW | **Starting Channel for Auto Scan.**<br>Sets the first CS0 channel to be selected by the mux for Capacitive Sense conversion when Auto Scan is enabled and active. All channels detailed in CS0MX are possible choices for this register. When Auto Scan is enabled, a write to CS0SS will also update CS0MX. |

### 16.4.8 CS0SE: Capacitive Sense 0 Auto Scan End Channel

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | CS0SE | | | | |
| Access | R | | | RW | | | | |
| Reset | 0x0 | | | 0x00 | | | | |
| SFR Page = 0x0; SFR Address: 0xDE | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:5* | *Reserved* | *Must write reset value.* | | |
| 4:0 | CS0SE | 0x00 | RW | **Ending Channel for Auto Scan.**<br>Sets the last CS0 channel to be selected by the mux for Capacitive Sense conversion when Auto Scan is enabled and active. All channels detailed in CS0MX are possible choices for this register. |

### 16.4.9 CS0THH: Capacitive Sense 0 Comparator Threshold High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0THH | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xFE | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CS0THH | 0x00 | RW | **CS0 Comparator Threshold High Byte.**<br>High byte of the 16-bit value compared to the Capacitive Sense conversion result. |

### 16.4.10 CS0THL: Capacitive Sense 0 Comparator Threshold Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0THL | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xFD | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CS0THL | 0x00 | RW | **CS0 Comparator Threshold Low Byte.**<br>Low byte of the 16-bit value compared to the Capacitive Sense conversion result. |

### 16.4.11 CS0MD1: Capacitive Sense 0 Mode 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | CS0POL | CS0DR | | CS0WOI | CS0CG | | |
| Access | RW | R | R | | R | RW | | |
| Reset | 0 | 0 | 0x0 | | 0 | 0x7 | | |
| SFR Page = 0x0; SFR Address: 0xAF | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7* | *Reserved* | *Must write reset value.* | | |
| 6 | CS0POL | 0 | R | **CS0 Digital Comparator Polarity Select.** |

| Value | Name | Description |
|---|---|---|
| 0 | GREATER_THAN | The digital comparator generates an interrupt if the conversion is greater than the threshold. |
| 1 | LESS_THAN_OR_EQUAL | The digital comparator generates an interrupt if the conversion is less than or equal to the threshold. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5:4 | CS0DR | 0x0 | R | **CS0 Double Reset Select.** |

These bits adjust the secondary CS0 reset time. For most touch-sensitive switches, the default (fastest) value is sufficient.

| Value | Name | Description |
|---|---|---|
| 0x0 | DOUBLE_RESET_DISABLED | No additional time is used for secondary reset. |
| 0x1 | ADD_0P75 | An additional 0.75 us is used for secondary reset. |
| 0x2 | ADD_1P5 | An additional 1.5 us is used for secondary reset. |
| 0x3 | ADD_2P25 | An additional 2.25 us is used for secondary reset. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 3 | CS0WOI | 0 | R | **CS0 Wake on Interrupt Configuration.** |

| Value | Name | Description |
|---|---|---|
| 0 | COMPARATOR_ONLY | Wake-up event generated on digital comparator interrupt only. |
| 1 | EOS_OR_COMPARATOR | Wake-up event generated on end of scan or digital comparator interrupt. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2:0 | CS0CG | 0x7 | RW | **CS0 Capacitance Gain Select.** |

These bits select the gain applied to the capacitance measurement. Lower gain values increase the size of the capacitance that can be measured with the CS0 module. The capacitance gain is equivalent to CS0CG[2:0] + 1.

| Value | Name | Description |
|---|---|---|
| 0x0 | GAIN_1 | Gain = 1x |
| 0x1 | GAIN_2 | Gain = 2x |
| 0x2 | GAIN_3 | Gain = 3x |
| 0x3 | GAIN_4 | Gain = 4x |
| 0x4 | GAIN_5 | Gain = 5x |
| 0x5 | GAIN_6 | Gain = 6x |
| 0x6 | GAIN_7 | Gain = 7x |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | 0x7 | GAIN_8 | | Gain = 8x |

### 16.4.12 CS0MD2: Capacitive Sense 0 Mode 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CS0CR | | CS0DT | | | CS0IA | | |
| Access | RW | | RW | | | RW | | |
| Reset | 0x1 | | 0x0 | | | 0x0 | | |
| SFR Page = 0x0; SFR Address: 0xF3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:6 | CS0CR | 0x1 | RW | **CS0 Conversion Rate.**<br>These bits control the conversion rate of the CS0 module. |

| Value | Name | Description |
|---|---|---|
| 0x0 | 12_BITS | Conversions last 12 internal CS0 clocks and are 12 bits in length. |
| 0x1 | 13_BITS | Conversions last 13 internal CS0 clocks and are 13 bits in length. |
| 0x2 | 14_BITS | Conversions last 14 internal CS0 clocks and are 14 bits in length. |
| 0x3 | 16_BITS | Conversions last 16 internal CS0 clocks.and are 16 bits in length. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5:3 | CS0DT | 0x0 | RW | **CS0 Discharge Time.**<br>These bits adjust the primary CS0 reset time. For most touch-sensitive switches, the default (fastest) value is sufficient. |

| Value | Name | Description |
|---|---|---|
| 0x0 | DISCHARGE_MODE0 | Discharge time is 0.75 us (recommended for most switches). |
| 0x1 | DISCHARGE_MODE1 | Discharge time is 1.0 us. |
| 0x2 | DISCHARGE_MODE2 | Discharge time is 1.2 us. |
| 0x3 | DISCHARGE_MODE3 | Discharge time is 1.5 us. |
| 0x4 | DISCHARGE_MODE4 | Discharge time is 2 us. |
| 0x5 | DISCHARGE_MODE5 | Discharge time is 3 us. |
| 0x6 | DISCHARGE_MODE6 | Discharge time is 6 us. |
| 0x7 | DISCHARGE_MODE7 | Discharge time is 12 us. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2:0 | CS0IA | 0x0 | RW | **CS0 Output Current Adjustment.**<br>These bits adjust the output current used to charge up the capacitive sensor element. For most touch-sensitive switches, the default (highest) current is sufficient. |

| Value | Name | Description |
|---|---|---|
| 0x0 | CURRENT_MODE0 | Full current. |
| 0x1 | CURRENT_MODE1 | 1/8 current. |
| 0x2 | CURRENT_MODE2 | 1/4 current. |
| 0x3 | CURRENT_MODE3 | 3/8 current. |
| 0x4 | CURRENT_MODE4 | 1/2 current. |
| 0x5 | CURRENT_MODE5 | 5/8 current. |
| 0x6 | CURRENT_MODE6 | 3/4 current. |
| 0x7 | CURRENT_MODE7 | 7/8 current. |

### 16.4.13 CS0MD3: Capacitive Sense 0 Mode 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | CS0RP | | CS0LP | | |
| Access | RW | | | RW | | RW | | |
| Reset | 0x0 | | | 0x0 | | 0x0 | | |
| SFR Page = 0xF; SFR Address: 0xF3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:5* | *Reserved* | *Must write reset value.* | | |
| 4:3 | CS0RP | 0x0 | RW | **CS0 Ramp Selection.**<br>These bits are used to compensate CS0 conversions for circuits requiring slower ramp times. For most touch-sensitive switches, the default (fastest) value is sufficient. |
| 2:0 | CS0LP | 0x0 | RW | **CS0 Low Pass Filter Selection.**<br>These bits set the internal corner frequency of the CS0 low-pass filter. Higher values of CS0LP result in a lower internal corner frequency.<br>For most touch-sensitive switches, the default setting of 000b should be used. If the CS0RP bits are adjusted from their default value, the CS0LP bits should normally be set to 001b. Settings higher than 001b will result in attenuated readings from the CS0 module and should be used only under special circumstances. |

| Value | Name | Description |
|---|---|---|
| 0x0 | RAMP_MODE0 | Ramp time is less than 1.5 us. |
| 0x1 | RAMP_MODE1 | Ramp time is between 1.5 us and 3 us. |
| 0x2 | RAMP_MODE2 | Ramp time is between 3 us and 6 us. |
| 0x3 | RAMP_MODE3 | Ramp time is greater than 6 us. |

### 16.4.14 CS0PM: Capacitive Sense 0 Pin Monitor

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | UAPM | SPIPM | SMBPM | PCAPM | PIOPM | CP0PM | CSPMMD | |
| Access | RW | RW | RW | RW | RW | RW | RW | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0x0 | |
| SFR Page = 0xF; SFR Address: 0xDE | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | UAPM | 0 | RW | **UART Pin Monitor Enable.** |
| | Enables monitoring of the UART TX pin. | | | |
| 6 | SPIPM | 0 | RW | **SPI Pin Monitor Enable.** |
| | Enables monitoring SPI output pins. | | | |
| 5 | SMBPM | 0 | RW | **SMBus Pin Monitor Enable.** |
| | Enables monitoring of the SMBus pins. | | | |
| 4 | PCAPM | 0 | RW | **PCA Pin Monitor Enable.** |
| | Enables monitoring of PCA output pins. | | | |
| 3 | PIOPM | 0 | RW | **Port I/O Pin Monitor Enable.** |
| | Enables monitoring of writes to the port latch registers. | | | |
| 2 | CP0PM | 0 | RW | **CP0 Pin Monitor Enable.** |
| | Enables monitoring of the comparator CP0 output. | | | |
| 1:0 | CSPMMD | 0x0 | RW | **CS0 Pin Monitor Mode.** |
| | Selects the operation to take when a monitored signal changes state. | | | |

| Value | Name | Description |
|---|---|---|
| 0x0 | ALWAYS_RETRY | Always retry bit cycles on a pin state change. |
| 0x1 | RETRY_TWO_TIMES | Retry up to twice on consecutive bit cycles. |
| 0x2 | RETRY_FOUR_TIMES | Retry up to four times on consecutive bit cycles. |

### 16.4.15 CS0MX: Capacitive Sense 0 Mux Channel Select

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | CS0MX | | | |
| Access | RW | | | | RW | | | |
| Reset | 0x8 | | | | 0xF | | | |
| SFR Page = 0x0; SFR Address: 0xAB | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:4* | *Reserved* | *Must write reset value.* | | |
| 3:0 | CS0MX | 0xF | RW | **CS0 Mux Channel Select.** |
| | Selects a single input channel for Capacitive Sense conversion. | | | |

# 17. Cyclic Redundancy Check (CRC0)

## 17.1 Introduction

The cyclic redundancy check (CRC) module performs a CRC using a 16-bit polynomial. CRC0 accepts a stream of 8-bit data and posts the 16-bit result to an internal register. In addition to using the CRC block for data manipulation, hardware can automatically CRC the flash contents of the device.

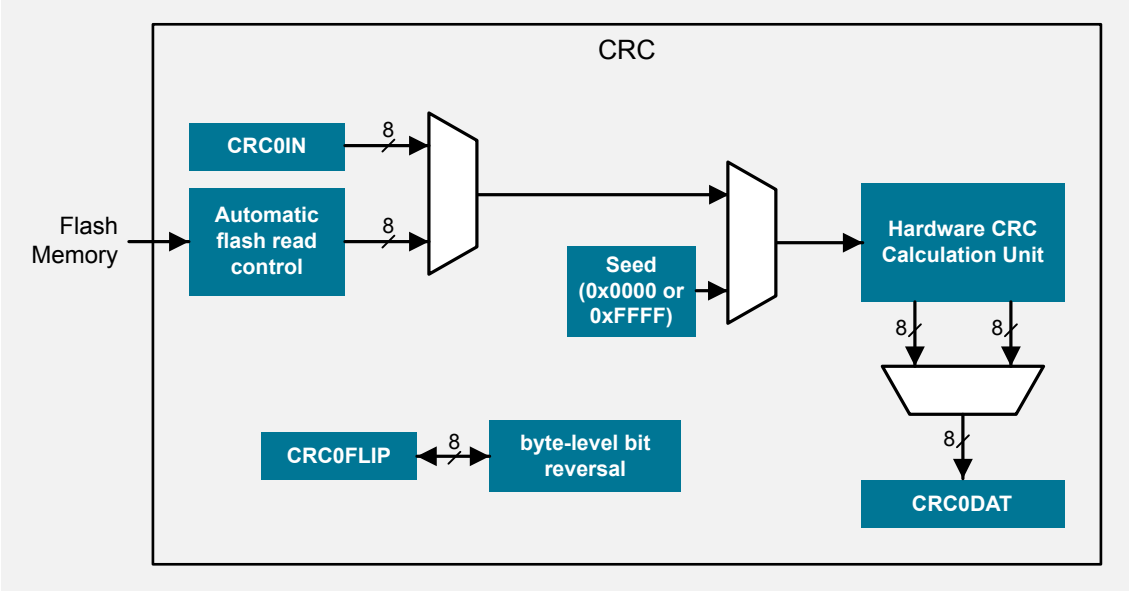



**Figure 17.1. CRC Functional Block Diagram**

## 17.2 Features

The CRC module is designed to provide hardware calculations for flash memory verification and communications protocols. The CRC module supports the standard CCITT-16 16-bit polynomial (0x1021), and includes the following features:

- Support for CCITT-16 polynomial
- Byte-level bit reversal
- Automatic CRC of flash contents on one or more 256-byte blocks
- Initial seed selection of 0x0000 or 0xFFFF

## 17.3 Functional Description

### 17.3.1 16-bit CRC Algorithm

The CRC unit generates a 16-bit CRC result equivalent to the following algorithm:

1. XOR the input with the most-significant bits of the current CRC result. 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, shift the CRC result and XOR the result with the polynomial.
3. If the MSB of the CRC result is not set, shift the CRC result.
4. Repeat steps 2 and 3 for all 8 bits.

The algorithm is also described in the following example.

```
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 the final remainder (CRC value)
    return CRC_acc;
}
```

The following table lists several input values and the associated outputs using the 16-bit CRC algorithm:

**Table 17.1. Example 16-bit CRC Outputs**

| Input | Output |
|---|---|
| 0x63 | 0xBD35 |
| 0x8C | 0xB1F4 |
| 0x7D | 0x4ECA |
| 0xAA, 0xBB, 0xCC | 0x6CF6 |
| 0x00, 0x00, 0xAA, 0xBB, 0xCC | 0xB166 |

### 17.3.2 Using the CRC on a Data Stream

The CRC module may be used to perform CRC calculations on any data set available to the firmware. To perform a CRC on an arbitrary data sream:

1. Select the initial result value using CRCVAL.
2. Set the result to its initial value (write 1 to CRCINIT).
3. Write the data to CRC0IN one byte at a time. The CRC result registers are automatically updated after each byte is written.
4. Write the CRCPNT bit to 0 to target the low byte of the result.
5. Read CRC0DAT multiple times to access each byte of the CRC result. CRCPNT will automatically point to the next value after each read.

### 17.3.3 Using the CRC to Check Code Memory

The CRC module may be configured to automatically perform a CRC on one or more blocks of code memory. To perform a CRC on code contents:

1. Select the initial result value using CRCVAL.
2. Set the result to its initial value (write 1 to CRCINIT).
3. Write the high byte of the starting address to the CRCST bit field.
4. Set the AUTOEN bit to 1.
5. Write the number of byte blocks to perform in the CRC calculation to CRCCNT.
6. Write any value to CRC0CN0 (or OR its contents with 0x00) to initiate the CRC calculation. The CPU will not execute code any additional code until the CRC operation completes.
   **Note:** Upon initiation of an automatic CRC calculation, the three cycles following a write to CRC0CN0 that initiate a CRC operation must only contain instructions which execute in the same number of cycles as the number of bytes in the instruction. An example of such an instruction is a 3-byte MOV that targets the CRC0FLIP register. When programming in C, the dummy value written to CRC0FLIP should be a non-zero value to prevent the compiler from generating a 2-byte MOV instruction.
7. Clear the AUTOEN.
8. Write the CRCPNT bit to 0 to target the low byte of the result.
9. Read CRC0DAT multiple times to access each byte of the CRC result. CRCPNT will automatically point to the next value after each read.

### 17.3.4 Bit Reversal

CRC0 includes hardware to reverse the bit order of each bit in a byte. Writing a byte to CRC0FLIP initiates the bit reversal operation, and the result may be read back from CRC0FLIP on the next instruction. For example, if 0xC0 is written to CRC0FLIP, the data read back is 0x03. Bit reversal can be used to change the order of information passing through the CRC engine and is also used in algorithms such as FFT.

## 17.4 CRC0 Control Registers

### 17.4.1 CRC0CN0: CRC0 Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | | CRCINIT | CRCVAL | Reserved | CRCPNT |
| Access | R | | | | RW | RW | R | RW |
| Reset | 0x1 | | | | 0 | 0 | 0 | 0 |
| SFR Page = ALL; SFR Address: 0x84 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:4* | *Reserved* | *Must write reset value.* | | |
| 3 | CRCINIT | 0 | RW | **CRC Initialization Enable.**<br>Writing a 1 to this bit initializes the entire CRC result based on CRCVAL. |
| 2 | CRCVAL | 0 | RW | **CRC Initialization Value.**<br>This bit selects the set value of the CRC result. |

| Value | Name | Description |
|---|---|---|
| 0 | SET_ZEROES | CRC result is set to 0x0000 on write of 1 to CRCINIT. |
| 1 | SET_ONES | CRC result is set to 0xFFFF on write of 1 to CRCINIT. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *1* | *Reserved* | *Must write reset value.* | | |
| 0 | CRCPNT | 0 | RW | **CRC Result Pointer.**<br>Specifies the byte of the CRC result to be read/written on the next access to CRC0DAT. This bit will automatically toggle upon each read or write. |

| Value | Name | Description |
|---|---|---|
| 0 | ACCESS_LOWER | CRC0DAT accesses bits 7-0 of the 16-bit CRC result. |
| 1 | ACCESS_UPPER | CRC0DAT accesses bits 15-8 of the 16-bit CRC result. |

Upon initiation of an automatic CRC calculation, the three cycles following a write to CRC0CN0 that initiate a CRC operation must only contain instructions which execute in the same number of cycles as the number of bytes in the instruction. An example of such an instruction is a 3-byte MOV that targets the CRC0FLIP register. When programming in C, the dummy value written to CRC0FLIP should be a non-zero value to prevent the compiler from generating a 2-byte MOV instruction.

### 17.4.2 CRC0IN: CRC0 Data Input

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CRC0IN | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0x85 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CRC0IN | 0x00 | RW | **CRC Data Input.**<br>Each write to CRC0IN results in the written data being computed into the existing CRC result according to the CRC algorithm. |

### 17.4.3 CRC0DAT: CRC0 Data Output

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CRC0DAT | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0x86 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CRC0DAT | 0x00 | RW | **CRC Data Output.**<br>Each read or write performed on CRC0DAT targets the CRC result bits pointed to by the CRC0 Result Pointer (CRCPNT bits in CRC0CN0). |
| CRC0DAT may not be valid for one cycle after setting the CRCINIT bit in the CRC0CN0 register to 1. Any time CRCINIT is written to 1 by firmware, at least one instruction should be performed before reading CRC0DAT. | | | | |

### 17.4.4 CRC0AUTO: CRC0 Automatic Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AUTOEN | CRCDN | Reserved | CRCST | | | | |
| Access | RW | R | RW | RW | | | | |
| Reset | 0 | 1 | 0 | 0x00 | | | | |
| SFR Page = ALL; SFR Address: 0x9E | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | AUTOEN | 0 | RW | **Automatic CRC Calculation Enable.**<br>When AUTOEN is set to 1, any write to CRC0CN0 will initiate an automatic CRC starting at flash sector CRCST and continuing for CRCCNT sectors. |
| 6 | CRCDN | 1 | R | **Automatic CRC Calculation Complete.**<br>Set to 0 when a CRC calculation is in progress. Code execution is stopped during a CRC calculation; therefore, reads from firmware will always return 1. |
| *5* | *Reserved* | *Must write reset value.* | | |
| 4:0 | CRCST | 0x00 | RW | **Automatic CRC Calculation Starting Block.**<br>These bits specify the flash block to start the automatic CRC calculation. The starting address of the first flash block included in the automatic CRC calculation is CRCST x block_size, where block_size is 256 bytes. |

### 17.4.5 CRC0CNT: CRC0 Automatic Flash Sector Count

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | Reserved | | | CRCCNT | | | | |
| Access | R | | | RW | | | | |
| Reset | 0x0 | | | 0x00 | | | | |
| SFR Page = ALL; SFR Address: 0x9A | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *7:5* | *Reserved* | *Must write reset value.* | | |
| 4:0 | CRCCNT | 0x00 | RW | **Automatic CRC Calculation Block Count.**<br>These bits specify the number of flash blocks to include in an automatic CRC calculation. The last address of the last flash block included in the automatic CRC calculation is (CRCST+CRCCNT) x Block Size - 1. The block size is 256 bytes. |

### 17.4.6 CRC0FLIP: CRC0 Bit Flip

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CRC0FLIP | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = ALL; SFR Address: 0x9C | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | CRC0FLIP | 0x00 | RW | **CRC0 Bit Flip.**<br>Any byte written to CRC0FLIP is read back in a bit-reversed order, i.e., the written LSB becomes the MSB. For example:<br>If 0xC0 is written to CRC0FLIP, the data read back will be 0x03.<br>If 0x05 is written to CRC0FLIP, the data read back will be 0xA0. |

# 18. Programmable Counter Array (PCA0)

## 18.1 Introduction

The programmable counter array (PCA) provides multiple channels of enhanced timer and PWM functionality while requiring less CPU intervention than standard counter/timers. The PCA consists of a dedicated 16-bit counter/timer and one 16-bit capture/compare module for each channel. The counter/timer is driven by a programmable timebase that has flexible external and internal clocking options. Each capture/compare module may be configured to operate independently in one of five modes: Edge-Triggered Capture, Software Timer, High-Speed Output, Frequency Output, or Pulse-Width Modulated (PWM) Output. Each capture/compare module has its own associated I/O line (CEXn) which is routed through the crossbar to port I/O when enabled.

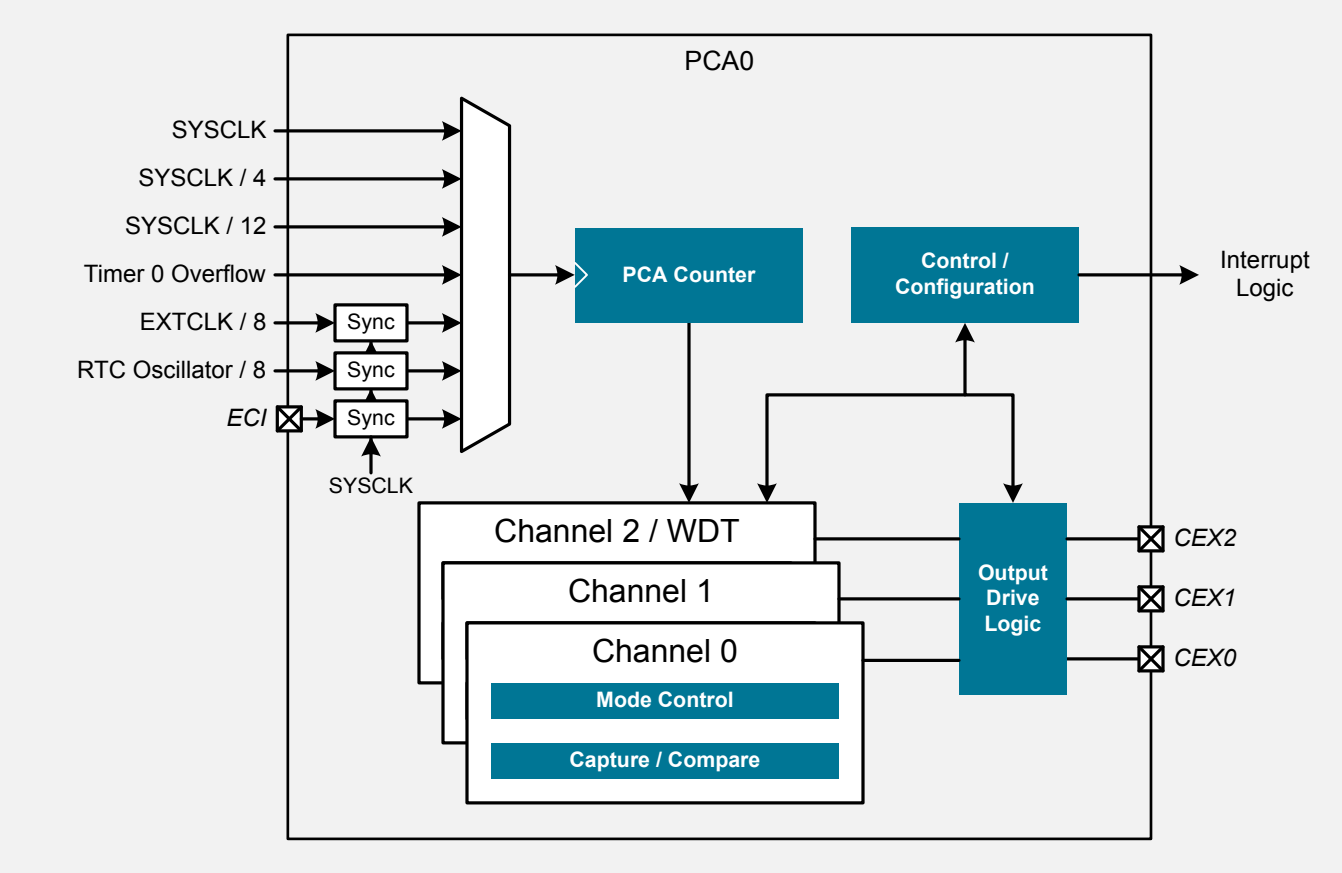



**Figure 18.1. PCA Block Diagram**

## 18.2 Features

- 16-bit time base.
- Programmable clock divisor and clock source selection.
- Up to three independently-configurable channels
- 8, 9, 10, 11 and 16-bit PWM modes (edge-aligned operation).
- Frequency output mode.
- Capture on rising, falling or any edge.
- Compare function for arbitrary waveform generation.
- Software timer (internal compare) mode.
- Integrated watchdog timer.

## 18.3 Functional Description

### 18.3.1 Counter / Timer

The 16-bit PCA counter/timer consists of two 8-bit SFRs: PCA0L and PCA0H. PCA0H is the high byte of the 16-bit counter/timer and PCA0L is the low byte. Reading PCA0L automatically latches the value of PCA0H into a “snapshot” register; the following PCA0H read accesses this “snapshot” register.
**Note:** 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.

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 18.1. PCA Timebase Input Options**

| CPS2:0 | Timebase |
|---|---|
| 000 | System clock divided by 12 |
| 001 | System clock divided by 4 |
| 010 | Timer 0 overflow |
| 011 | High-to-low transitions on ECI (max rate = system clock divided by 4) [1] |
| 100 | System clock |
| 101 | External oscillator source divided by 8 [1] |
| 110 | Low frequency oscillator divided by 8 [1] |
| 111 | Reserved |

**Note:**
1. Synchronized with the system clock.

### 18.3.2 Interrupt Sources

The PCA0 module shares one interrupt vector among all of its modules. There are are several event flags that can be used to generate a PCA0 interrupt. They are as follows: the main PCA counter overflow flag (CF), which is set upon a 16-bit overflow of the PCA0 counter; an intermediate overflow flag (COVF), which can be set on an overflow from the 8th–11th bit of the PCA0 counter; and the individual flags for each PCA channel (CCFn), which are set according to the operation mode of that module. These event flags are always set when the trigger condition occurs. Each of these flags can be individually selected to generate a PCA0 interrupt using the corresponding interrupt enable flag (ECF for CF, ECOV for COVF, and ECCFn for each CCFn). PCA0 interrupts must be globally enabled before any individual interrupt sources are recognized by the processor. PCA0 interrupts are globally enabled by setting the EA bit and the EPCA0 bit to logic 1.

### 18.3.3 Capture/Compare Modules

Each module can be configured to operate independently in one of six operation modes: edge-triggered capture, software timer, high-speed output, frequency output, 8 to 11-bit pulse width modulator, or 16-bit pulse width modulator. Table 18.2 PCA0CPM and PCA0PWM Bit Settings for PCA Capture/Compare Modules on page 181 summarizes the bit settings in the PCA0CPMn and PCA0PWM registers used to select the PCA capture/compare module’s operating mode. All modules set to use 8-, 9-, 10-, or 11-bit PWM mode must use the same cycle length (8–11 bits). Setting the ECCFn bit in a PCA0CPMn register enables the module's CCFn interrupt.

**Table 18.2. PCA0CPM and PCA0PWM Bit Settings for PCA Capture/Compare Modules**

| Operational Mode | PCA0CPMn | | | | | | | | PCA0PWM | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit Name | PWM16 | ECOM | CAPP | CAPN | MAT | TOG | PWM | ECCF | ARSEL | ECOV | COVF | Reserved | CLSEL |
| Capture triggered by positive edge on CEXn | X | X | 1 | 0 | 0 | 0 | 0 | A | 0 | X | B | X | X |
| Capture triggered by negative edge on CEXn | X | X | 0 | 1 | 0 | 0 | 0 | A | 0 | X | B | X | X |
| Capture triggered by any transition on CEXn | X | X | 1 | 1 | 0 | 0 | 0 | A | 0 | X | B | X | X |
| Software Timer | X | C | 0 | 0 | 1 | 0 | 0 | A | 0 | X | B | X | X |
| High Speed Output | X | C | 0 | 0 | 1 | 1 | 0 | A | 0 | X | B | X | X |
| Frequency Output | X | C | 0 | 0 | 0 | 1 | 1 | A | 0 | X | B | X | X |
| 8-Bit Pulse Width Modulator[7] | 0 | C | 0 | 0 | E | 0 | 1 | A | 0 | X | B | X | 0 |
| 9-Bit Pulse Width Modulator[7] | 0 | C | 0 | 0 | E | 0 | 1 | A | D | X | B | X | 1 |
| 10-Bit Pulse Width Modulator[7] | 0 | C | 0 | 0 | E | 0 | 1 | A | D | X | B | X | 2 |
| 11-Bit Pulse Width Modulator[7] | 0 | C | 0 | 0 | E | 0 | 1 | A | D | X | B | X | 3 |
| 16-Bit Pulse Width Modulator | 1 | C | 0 | 0 | E | 0 | 1 | A | 0 | X | B | X | X |

Notes:

1. X = Don't Care (no functional difference for individual module if 1 or 0).
2. A = Enable interrupts for this module (PCA interrupt triggered on CCFn set to 1).
3. B = Enable 8th–11th bit overflow interrupt (Depends on setting of CLSEL).
4. C = When set to 0, the digital comparator is off. For high speed and frequency output modes, the associated pin will not toggle. In any of the PWM modes, this generates a 0% duty cycle (output = 0).
5. D = Selects whether the Capture/Compare register (0) or the Auto-Reload register (1) for the associated channel is accessed via addresses PCA0CPHn and PCA0CPLn.
6. E = When set, a match event will cause the CCFn flag for the associated channel to be set.
7. All modules set to 8, 9, 10 or 11-bit PWM mode use the same cycle length setting.

### 18.3.4 Edge-Triggered Capture Mode

In this mode, a valid transition on the CEXn pin causes the PCA to capture the value of the PCA counter/timer and load it into the corresponding module's 16-bit capture/compare register (PCA0CPLn and PCA0CPHn). The CAPPn and CAPNn bits in the PCA0CPMn register are used to select the type of transition that triggers the capture: low-to-high transition (positive edge), high-to-low transition (negative edge), or either transition (positive or negative edge). When a capture occurs, the Capture/Compare Flag (CCFn) in PCA0CN0 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. If both CAPPn and CAPNn bits are set to logic 1, then the state of the port pin associated with CEXn can be read directly to determine whether a rising-edge or falling-edge caused the capture.

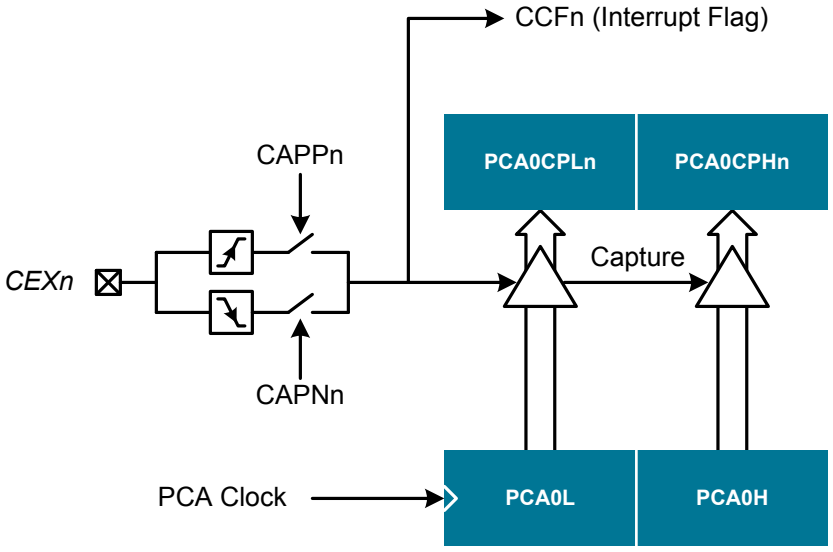



**Figure 18.2. PCA Capture Mode Diagram**

**Note:** The CEXn input signal must remain high or low for at least 2 system clock cycles to be recognized by the hardware.

### 18.3.5 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 PCA0CN0 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 it must be cleared by software. Setting the ECOMn and MATn bits in the PCA0CPMn register enables Software Timer mode.

**Note:** 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.

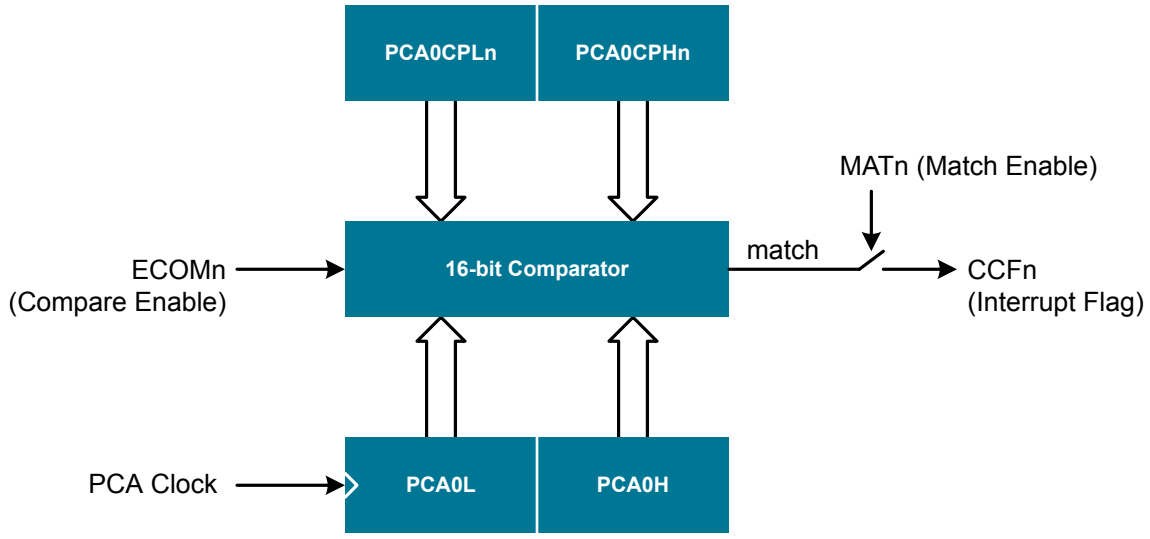



**Figure 18.3. PCA Software Timer Mode Diagram**

### 18.3.6 High-Speed Output Mode

In High-Speed Output mode, a module's associated CEXn pin is toggled each time a match occurs between the PCA Counter and the module's 16-bit capture/compare register (PCA0CPHn and PCA0CPLn). When a match occurs, the capture/compare flag (CCFn) in PCA0CN0 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. It must be cleared by software. Setting the TOGn, MATn, and ECOMn bits in the PCA0CPMn register enables the High-Speed Output mode. If ECOMn is cleared, the associated pin retains its state and not toggle on the next match event.

**Note:** 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.

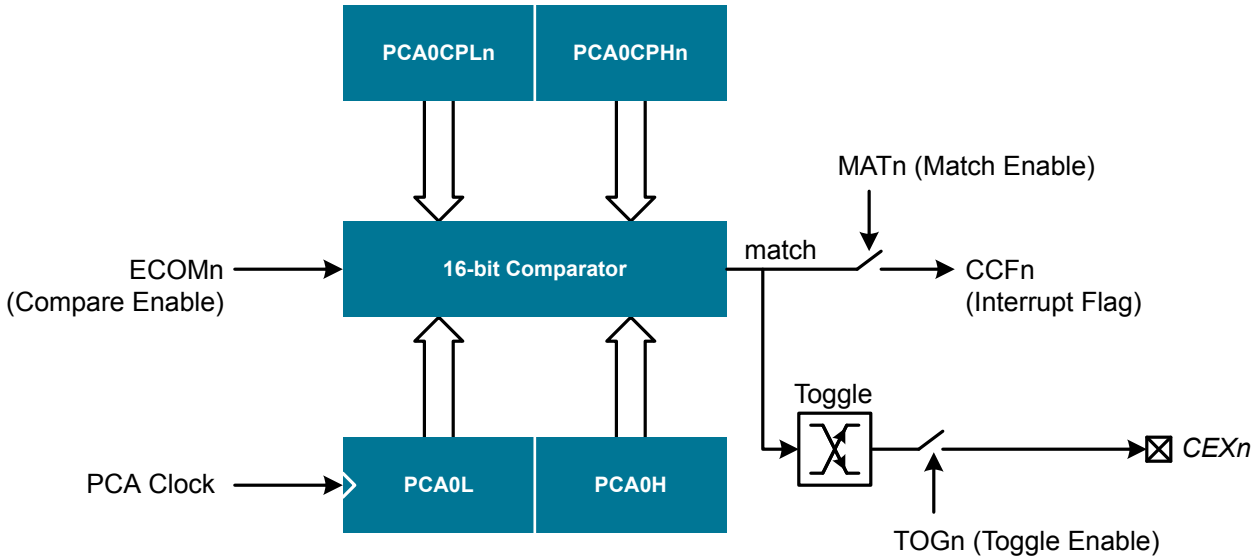



**Figure 18.4. PCA High-Speed Output Mode Diagram**

### 18.3.7 Frequency Output Mode

Frequency Output Mode produces a programmable-frequency square wave on the module's associated CEXn pin. The capture/compare module high byte holds the number of PCA clocks to count before the output is toggled. The frequency of the square wave is then defined as follows:

$$F_{CEXn} = \frac{F_{PCA}}{2 \times PCA0CPHn}$$

**Note:** A value of 0x00 in the PCA0CPHn register is equal to 256 for this equation.

Where $F_{PCA}$ is the frequency of the clock selected by the CPS2–0 bits in the PCA mode register PCA0MD. The lower byte of the capture/compare module is compared to the PCA counter low byte; on a match, n is toggled and the offset held in the high byte is added to the matched value in PCA0CPLn. Frequency Output Mode is enabled by setting the ECOMn, TOGn, and PWMn bits in the PCA0CPMn register.
**Note:** The MATn bit should normally be set to 0 in this mode. If the MATn bit is set to 1, the CCFn flag for the channel will be set when the 16-bit PCA0 counter and the 16-bit capture/compare register for the channel are equal.

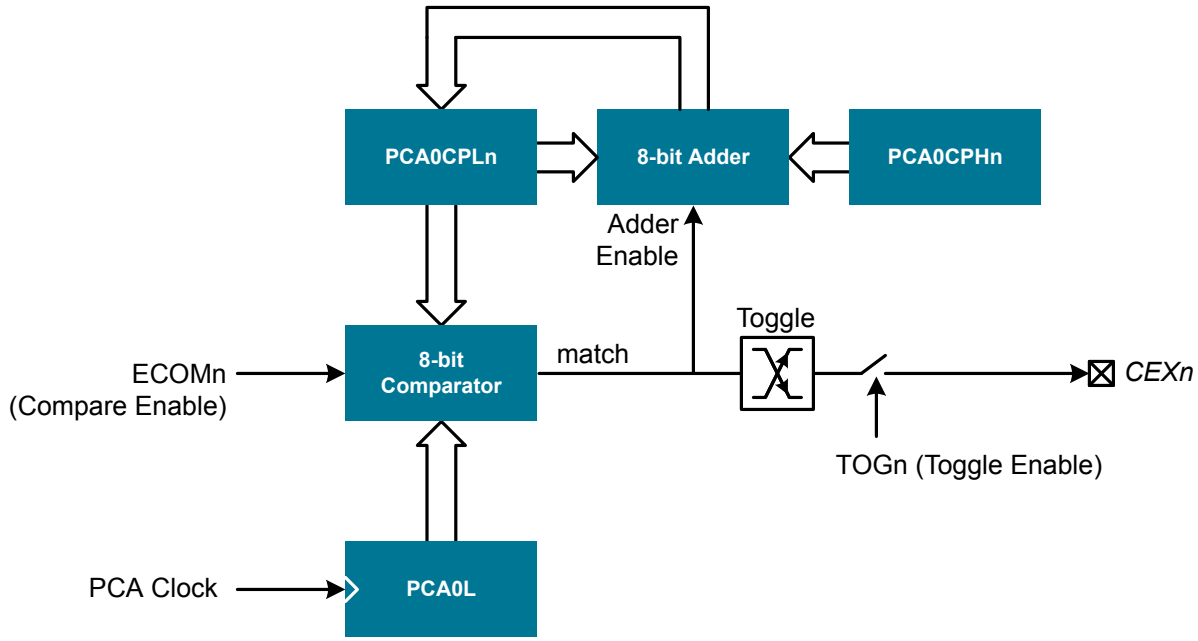



**Figure 18.5. PCA Frequency Output Mode**

### 18.3.8 PWM Waveform Generation

The PCA can generate edge-aligned PWM waveforms with resolutions of 8, 9, 10, 11, or 16 bits. PWM resolution depends on the module setup, as specified within the individual module PCA0CPMn registers as well as the PCA0PWM register. Modules can be configured for 8-11 bit mode or for 16-bit mode individually using the PCA0CPMn registers. All modules configured for 8-11 bit mode have the same resolution, specified by the PCA0PWM register.

### Edge Aligned PWM

When configured for edge-aligned mode, a module generates an edge transition at two points for every $2^N$ PCA clock cycles, where N is the selected PWM resolution in bits. In edge-aligned mode, these two edges are referred to as the “match” and “overflow” edges. The polarity at the output pin is selectable and can be inverted by setting the appropriate channel bit to 1 in the PCA0POL register. Prior to inversion, a match edge sets the channel to logic high, and an overflow edge clears the channel to logic low.

The match edge occurs when the the lowest N bits of the module’s PCA0CPn register match the corresponding bits of the main PCA0 counter register. For example, with 10-bit PWM, the match edge occurs any time bits 9-0 of the PCA0CPn register match bits 9-0 of the PCA0 counter value.

The overflow edge occurs when an overflow of the PCA0 counter happens at the desired resolution. For example, with 10-bit PWM, the overflow edge occurs when bits 0-9 of the PCA0 counter transition from all 1s to all 0s. All modules configured for edge-aligned mode at the same resolution align on the overflow edge of the waveforms.

An example of the PWM timing in edge-aligned mode for two channels is shown here.

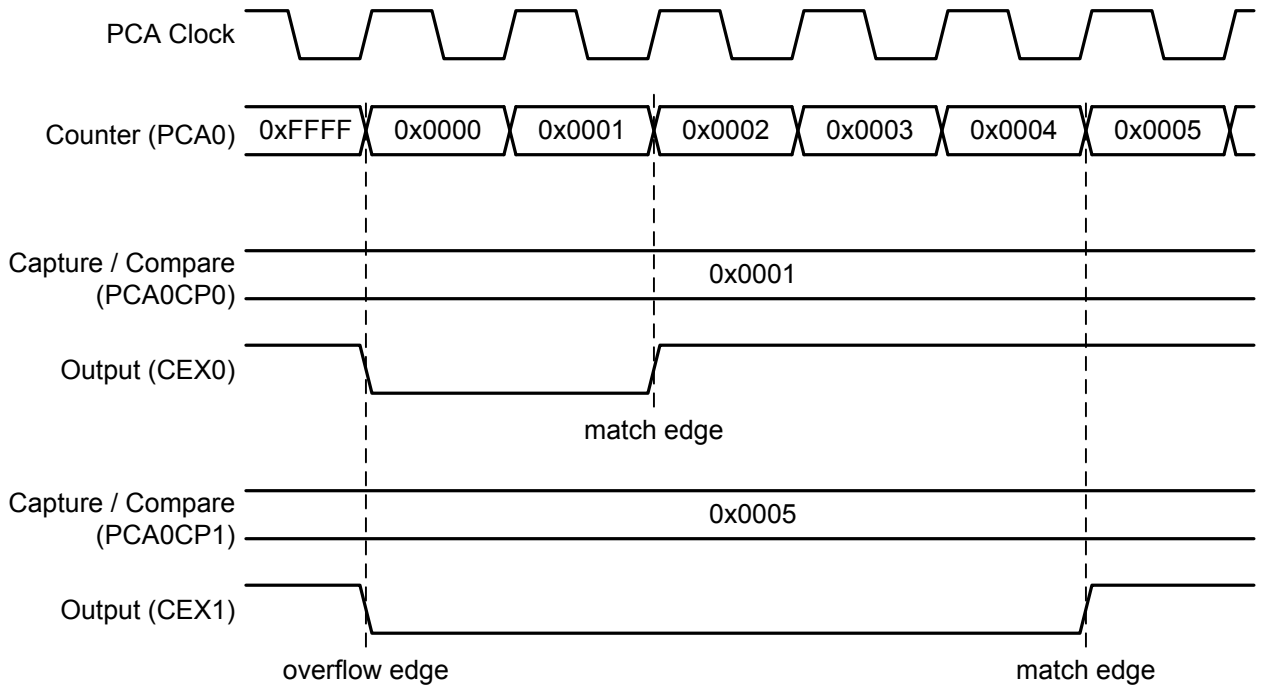



**Figure 18.6. Edge-Aligned PWM Timing**

For a given PCA resolution, the unused high bits in the PCA0 counter and the PCA0CPn compare registers are ignored, and only the used bits of the PCA0CPn register determine the duty cycle. A 0% duty cycle for the channel is achieved by clearing the module’s ECOM bit to 0. This will disable the comparison, and prevent the match edge from occuring.
**Note:** Although the PCA0CPn compare register determines the duty cycle, it is not always appropriate for firmware to update this register directly. See the sections on 8 to 11-bit and 16-bit PWM mode for additional details on adjusting duty cycle in the various modes.

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

**Figure 18.7. N-bit Edge-Aligned PWM Duty Cycle (N = PWM resolution)**

#### 18.3.8.1 8 to 11-Bit PWM Modes

Each module can be used independently to generate a pulse width modulated (PWM) output on its associated CEXn pin. The frequency of the output is dependent on the timebase for the PCA counter/timer and the setting of the PWM cycle length (8 through 11-bits). For backwards-compatibility with the 8-bit PWM mode available on other devices, the 8-bit PWM mode operates slightly different than 9 through 11-bit PWM modes.

**Important:** All channels configured for 8 to 11-bit PWM mode use the same cycle length. It is not possible to configure one channel for 8-bit PWM mode and another for 11-bit mode (for example). However, other PCA channels can be configured to Pin Capture, High-Speed Output, Software Timer, Frequency Output, or 16-bit PWM mode independently. Each channel configured for a PWM mode can be individually selected to operate in edge-aligned or center-aligned mode.

##### 8-bit Pulse Width Modulator Mode

In 8-bit PWM mode, the duty cycle is determined by the value of the low byte of the PCA0CPn register (PCA0CPLn). To adjust the duty cycle, PCA0CPLn should not normally be written directly. Instead, the recommendation is to adjust the duty cycle using the high byte of the PCA0CPn register (register PCA0CPHn). This allows seamless updating of the PWM waveform as PCA0CPLn is reloaded automatically with the value stored in PCA0CPHn during the overflow edge (in edge-aligned mode) or the up edge (in center-aligned mode).

Setting the ECOMn and PWMn bits in the PCA0CPMn register and setting the CLSEL bits in register PCA0PWM to 00b enables 8-Bit pulse width modulator mode. If the MATn bit is set to 1, the CCFn flag for the module is set each time a match edge or up edge occurs. The COVF flag in PCA0PWM can be used to detect the overflow (falling edge), which occurs every 256 PCA clock cycles.

##### 9- to 11-bit Pulse Width Modulator Mode

In 9 to 11-bit PWM mode, the duty cycle is determined by the value of the least significant N bits of the PCA0CPn register, where N is the selected PWM resolution.

To adjust the duty cycle, PCA0CPn should not normally be written directly. Instead, the recommendation is to adjust the duty cycle 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. This allows seamless updating of the PWM waveform, as the PCA0CPn register is reloaded automatically with the value stored in the auto-reload registers during the overflow edge (in edge-aligned mode) or the up edge (in center-aligned mode).

Setting the ECOMn and PWMn bits in the PCA0CPMn register and setting the CLSEL bits in register PCA0PWM to 00b enables 8-Bit pulse width modulator mode. If the MATn bit is set to 1, the CCFn flag for the module is set each time a match edge or up edge occurs. The COVF flag in PCA0PWM can be used to detect the overflow or down edge.

The 9 to 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 is set each time a match edge or up edge occurs. The COVF flag in PCA0PWM can be used to detect the overflow or down edge.

**Important:** 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.

#### 18.3.8.2 16-Bit PWM Mode

A PCA module may also be operated in 16-Bit PWM mode. 16-bit PWM mode is independent of the other PWM modes. The entire PCA0CP register is used to determine the duty cycle in 16-bit PWM mode.

To output a varying duty cycle, new value writes should be synchronized with the PCA CCFn match flag to ensure seamless updates.

16-Bit PWM mode is enabled by setting the ECOMn, PWMn, and PWM16n bits in the PCA0CPMn register. For a varying duty cycle, the match interrupt flag should be enabled (ECCFn = 1 AND MATn = 1) to help synchronize the capture/compare register writes. If the MATn bit is set to 1, the CCFn flag for the module is set each time a match edge or up edge occurs. The CF flag in PCA0CN0 can be used to detect the overflow or down edge.
**Important:** 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.

### 18.3.9 Watchdog Timer Mode

A programmable watchdog timer (WDT) function is available through the last PCA module (module 2). The WDT is used to generate a reset if the time between writes to the WDT update register (PCA0CPH2) exceed a specified limit. The WDT can be configured and enabled/disabled as needed by software. With the WDTE bit set in the PCA0MD register, the last module operates as a watchdog timer (WDT). The module 2 high byte is compared to the PCA counter high byte; the module 2 low byte holds the offset to be used when WDT updates are performed. The Watchdog Timer is enabled on reset. Writes to some PCA registers are restricted while the Watchdog Timer is enabled. The WDT will generate a reset shortly after code begins execution. To avoid this reset, the WDT should be explicitly disabled (and optionally re-configured and re-enabled if it is used in the system).

#### Watchdog Timer Operation

While the WDT is enabled:

- PCA counter is forced on.
- Writes to PCA0L and PCA0H are not allowed.
- PCA clock source CPS field is frozen.
- PCA Idle control bit (CIDL) is frozen.
- Module 2 is forced into software timer mode.
- Writes to the Module 2 mode register (PCA0CPM2) are disabled.

While the WDT is enabled, writes to the CR bit will not change the PCA counter state; the counter will run until the WDT is disabled. The PCA counter run control bit (CR) will read zero if the WDT is enabled but user software has not enabled the PCA counter. If a match occurs between PCA0CPH2 and PCA0H while the WDT is enabled, a reset will be generated. To prevent a WDT reset, the WDT may be updated with a write of any value to PCA0CPH2. Upon a PCA0CPH2 write, PCA0H plus the offset held in PCA0CPL2 is loaded into PCA0CPH2.

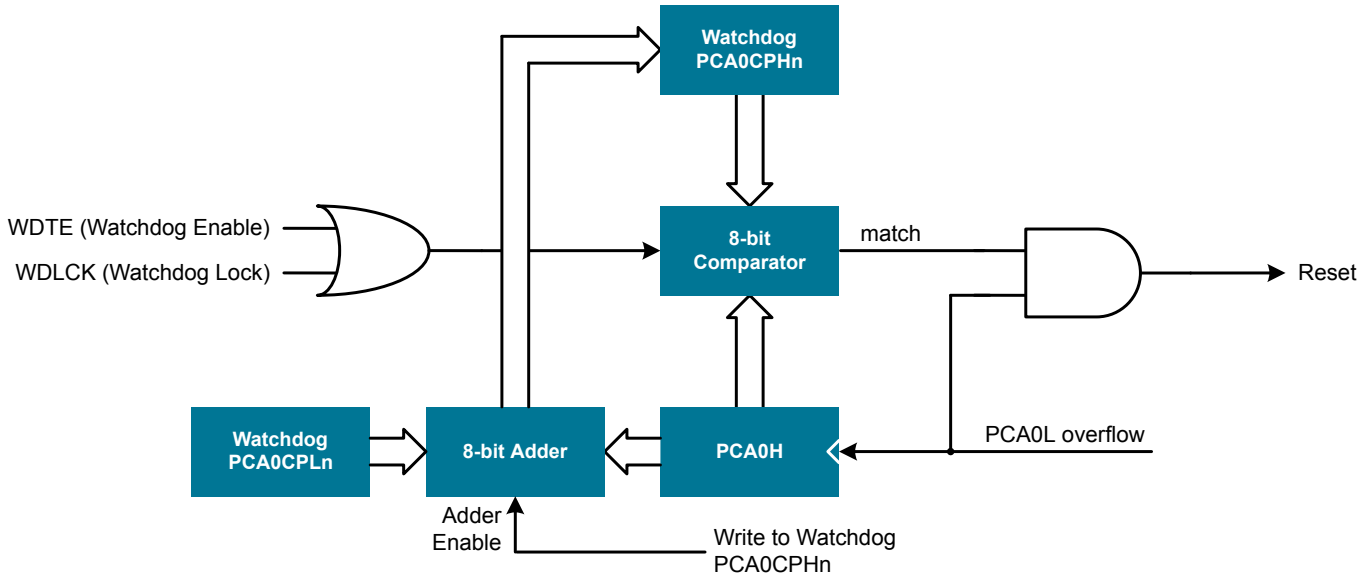



**Figure 18.8. PCA Module 2 with Watchdog Timer Enabled**

The 8-bit offset held in PCA0CPH2 is compared to the upper byte of the 16-bit PCA counter. This offset value is the number of PCA0L overflows before a reset. Up to 256 PCA clocks may pass before the first PCA0L overflow occurs, depending on the value of the PCA0L when the update is performed. The total offset is then given by the following equation in PCA clocks:

Offset = (256 × *PCA0CPL* ) + (256 – *PCA0L* )

**Note:** PCA0L is the value of the PCA0L register at the time of the update in this equation.

The WDT reset is generated when PCA0L overflows while there is a match between PCA0CPH2 and PCA0H. Software may force a WDT reset by writing a 1 to the CCF2 flag in the PCA0CN0 register while the WDT is enabled.

### Watchdog Timer Usage

To configure the WDT, perform the following tasks:

1. Disable the WDT by writing a 0 to the WDTE bit.
2. Select the desired PCA clock source (with the CPS field).
3. Load the WDT PCA0CPL with the desired WDT update offset value.
4. Configure the PCA Idle mode (set CIDL if the WDT should be suspended while the CPU is in Idle mode).
5. Enable the WDT by setting the WDTE bit to 1.
6. Reset the WDT timer by writing to PCA0CPH2.

The PCA clock source and Idle mode select cannot be changed while the WDT is enabled. The watchdog timer is enabled by setting the WDTE or WDLCK bits in the PCA0MD register. When WDLCK is set, the WDT cannot be disabled until the next system reset. If WDLCK is not set, the WDT is disabled by clearing the WDTE bit. The WDT is enabled following any reset. The PCA0 counter clock defaults to the system clock divided by 12, PCA0L defaults to 0x00, and PCA0CPL2 defaults to 0x00. This results in a WDT timeout interval of 256 PCA clock cycles, or 3072 system clock cycles. lists some example timeout intervals for typical system clocks.

**Table 18.3. Watchdog Timer Timeout Intervals**

| System Clock (Hz) | PCA0CPL2 | Timeout Interval (ms) |
|---|---|---|
| 24,500,000 | 255 | 32.1 |
| 24,500,000 | 128 | 16.2 |
| 24,500,000 | 32 | 4.1 |
| 3,062,500 | 255 | 257 |
| 3,062,500 | 128 | 129.5 |
| 3,062,500 | 32 | 33.1 |
| 32,000 | 255 | 24576 |
| 32,000 | 128 | 12384 |
| 32,000 | 32 | 3168 |
| **Note:** The values in this table assume SYSCLK/12 as the PCA clock source and a PCA0L value of 0x00 at the update time. | | |

## 18.4 PCA0 Control Registers

### 18.4.1 PCA0CN0: PCA Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CF | CR | Reserved | | | CCF2 | CCF1 | CCF0 |
| Access | RW | RW | R | | | RW | RW | RW |
| Reset | 0 | 0 | 0x0 | | | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xD8 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | CF | 0 | RW | **PCA Counter/Timer Overflow Flag.**<br>Set by hardware when the PCA Counter/Timer overflows from 0xFFFF to 0x0000. When the Counter/Timer Overflow (CF) interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by firmware. |
| 6 | CR | 0 | RW | **PCA Counter/Timer Run Control.**<br>This bit enables/disables the PCA Counter/Timer.<br>Value: 0 – STOP – Stop the PCA Counter/Timer.<br>Value: 1 – RUN – Start the PCA Counter/Timer running. |
| *5:3* | *Reserved* | *Must write reset value.* | | |
| 2 | CCF2 | 0 | RW | **PCA Module 2 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF2 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by firmware. |
| 1 | CCF1 | 0 | RW | **PCA Module 1 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF1 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by firmware. |
| 0 | CCF0 | 0 | RW | **PCA Module 0 Capture/Compare Flag.**<br>This bit is set by hardware when a match or capture occurs. When the CCF0 interrupt is enabled, setting this bit causes the CPU to vector to the PCA interrupt service routine. This bit is not automatically cleared by hardware and must be cleared by firmware. |

### 18.4.2 PCA0MD: PCA Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CIDL | WDTE | WDLCK | Reserved | CPS | | | ECF |
| Access | RW | RW | RW | R | RW | | | RW |
| Reset | 0 | 1 | 0 | 0 | 0x0 | | | 0 |
| SFR Page = 0x0; SFR Address: 0xD9 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | CIDL | 0 | RW | **PCA Counter/Timer Idle Control.** |
| | Specifies PCA behavior when CPU is in Idle Mode. | | | |
| | Value | Name | | Description |
| | 0 | NORMAL | | PCA continues to function normally while the system controller is in Idle Mode. |
| | 1 | SUSPEND | | PCA operation is suspended while the system controller is in Idle Mode. |
| 6 | WDTE | 1 | RW | **Watchdog Timer Enable.** |
| | If this bit is set, PCA Module 2 is used as the watchdog timer. | | | |
| | Value | Name | | Description |
| | 0 | DISABLED | | Disable Watchdog Timer. |
| | 1 | ENABLED | | Enable PCA Module 2 as the Watchdog Timer. |
| 5 | WDLCK | 0 | RW | **Watchdog Timer Lock.** |
| | This bit locks/unlocks the Watchdog Timer Enable. When WDLCK is set, the Watchdog Timer may not be disabled until the next system reset. | | | |
| | Value | Name | | Description |
| | 0 | UNLOCKED | | Watchdog Timer Enable unlocked. |
| | 1 | LOCKED | | Watchdog Timer Enable locked. |
| *4* | *Reserved* | *Must write reset value.* | | |
| 3:1 | CPS | 0x0 | RW | **PCA Counter/Timer Pulse Select.** |
| | These bits select the timebase source for the PCA counter. | | | |
| | Value | Name | | Description |
| | 0x0 | SYSCLK_DIV_12 | | System clock divided by 12. |
| | 0x1 | SYSCLK_DIV_4 | | System clock divided by 4. |
| | 0x2 | T0_OVERFLOW | | Timer 0 overflow. |
| | 0x3 | ECI | | High-to-low transitions on ECI (max rate = system clock divided by 4). |
| | 0x4 | SYSCLK | | System clock. |
| | 0x5 | EXTOSC_DIV_8 | | External clock divided by 8 (synchronized with the system clock). |
| | 0x6 | RTC_DIV_8 | | RTC divided by 8. |
| 0 | ECF | 0 | RW | **PCA Counter/Timer Overflow Interrupt Enable.** |
| | This bit sets the masking of the PCA Counter/Timer Overflow (CF) interrupt. | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| | Value | Name | | Description |
| | 0 | OVF_INT_DISABLED | | Disable the CF interrupt. |
| | 1 | OVF_INT_ENABLED | | Enable a PCA Counter/Timer Overflow interrupt request when CF is set. |

When the WDTE bit is set to 1, the other bits in the PCA0MD register cannot be modified. To change the contents of the PCA0MD register, the Watchdog Timer must first be disabled.

### 18.4.3 PCA0PWM: PCA PWM Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ARSEL | ECOV | COVF | Reserved | | | CLSEL | |
| Access | RW | RW | RW | R | | | RW | |
| Reset | 0 | 0 | 0 | 0x0 | | | 0x0 | |
| SFR Page = 0x0; SFR Address: 0xDF | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | ARSEL | 0 | RW | **Auto-Reload Register Select.** |

This bit selects whether to read and write the normal PCA capture/compare registers (PCA0CPn), or the Auto-Reload registers at the same SFR addresses. This function is used to define the reload value for 9 to 11-bit PWM modes. In all other modes, the Auto-Reload registers have no function.

| Value | Name | Description |
|---|---|---|
| 0 | CAPTURE_COMPARE | Read/Write Capture/Compare Registers at PCA0CPHn and PCA0CPLn. |
| 1 | AUTORELOAD | Read/Write Auto-Reload Registers at PCA0CPHn and PCA0CPLn. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | ECOV | 0 | RW | **Cycle Overflow Interrupt Enable.** |

This bit sets the masking of the Cycle Overflow Flag (COVF) interrupt.

| Value | Name | Description |
|---|---|---|
| 0 | COVF_MASK_DISABLED | COVF will not generate PCA interrupts. |
| 1 | COVF_MASK_ENABLED | A PCA interrupt will be generated when COVF is set. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5 | COVF | 0 | RW | **Cycle Overflow Flag.** |

This bit indicates an overflow of the 8th to 11th bit of the main PCA counter (PCA0). The specific bit used for this flag depends on the setting of the Cycle Length Select bits. The bit can be set by hardware or firmware, but must be cleared by firmware.

| Value | Name | Description |
|---|---|---|
| 0 | NO_OVERFLOW | No overflow has occurred since the last time this bit was cleared. |
| 1 | OVERFLOW | An overflow has occurred since the last time this bit was cleared. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *4:2* | *Reserved* | *Must write reset value.* | | |
| 1:0 | CLSEL | 0x0 | RW | **Cycle Length Select.** |

When 16-bit PWM mode is not selected, these bits select the length of the PWM cycle. This affects all channels configured for PWM which are not using 16-bit PWM mode. These bits are ignored for individual channels configured to 16-bit PWM mode.

| Value | Name | Description |
|---|---|---|
| 0x0 | 8_BITS | 8 bits. |
| 0x1 | 9_BITS | 9 bits. |
| 0x2 | 10_BITS | 10 bits. |
| 0x3 | 11_BITS | 11 bits. |

### 18.4.4 PCA0L: PCA Counter/Timer Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0L | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xF9 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0L | 0x00 | RW | **PCA Counter/Timer Low Byte.**<br>The PCA0L register holds the low byte (LSB) of the 16-bit PCA Counter/Timer. |
| When the WDTE bit is set to 1, the PCA0L register cannot be modified by firmware. To change the contents of the PCA0L register, the Watchdog Timer must first be disabled. | | | | |

### 18.4.5 PCA0H: PCA Counter/Timer High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0H | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xFA | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0H | 0x00 | RW | **PCA Counter/Timer High Byte.**<br>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. |
| When the WDTE bit is set to 1, the PCA0H register cannot be modified by firmware. To change the contents of the PCA0H register, the Watchdog Timer must first be disabled. | | | | |

### 18.4.6 PCA0CPM0: PCA Channel 0 Capture/Compare Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PWM16 | ECOM | CAPP | CAPN | MAT | TOG | PWM | ECCF |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xDA | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | PWM16 | 0 | RW | **Channel 0 16-bit Pulse Width Modulation Enable.**<br>This bit enables 16-bit mode when Pulse Width Modulation mode is enabled. |

| Value | Name | Description |
|---|---|---|
| 0 | 8_BIT | 8 to 11-bit PWM selected. |
| 1 | 16_BIT | 16-bit PWM selected. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | ECOM | 0 | RW | **Channel 0 Comparator Function Enable.**<br>This bit enables the comparator function. |
| 5 | CAPP | 0 | RW | **Channel 0 Capture Positive Function Enable.**<br>This bit enables the positive edge capture capability. |
| 4 | CAPN | 0 | RW | **Channel 0 Capture Negative Function Enable.**<br>This bit enables the negative edge capture capability. |
| 3 | MAT | 0 | RW | **Channel 0 Match Function Enable.**<br>This bit enables the match function. When enabled, matches of the PCA counter with a module's capture/compare register cause the CCF0 bit in the PCA0MD register to be set to logic 1. |
| 2 | TOG | 0 | RW | **Channel 0 Toggle Function Enable.**<br>This bit enables the toggle function. When enabled, matches of the PCA counter with the capture/compare register cause the logic level on the CEX0 pin to toggle. If the PWM bit is also set to logic 1, the module operates in Frequency Output Mode. |
| 1 | PWM | 0 | RW | **Channel 0 Pulse Width Modulation Mode Enable.**<br>This bit enables the PWM function. When enabled, a pulse width modulated signal is output on the CEX0 pin. 8 to 11-bit PWM is used if PWM16 is cleared to 0; 16-bit mode is used if PWM16 is set to 1. If the TOG bit is also set, the module operates in Frequency Output Mode. |
| 0 | ECCF | 0 | RW | **Channel 0 Capture/Compare Flag Interrupt Enable.**<br>This bit sets the masking of the Capture/Compare Flag (CCF0) interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable CCF0 interrupts. |
| 1 | ENABLED | Enable a Capture/Compare Flag interrupt request when CCF0 is set. |

### 18.4.7 PCA0CPL0: PCA Channel 0 Capture Module Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPL0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xFB | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0CPL0 | 0x00 | RW | **PCA Channel 0 Capture Module Low Byte.**<br>The PCA0CPL0 register holds the low byte (LSB) of the 16-bit capture module. This register address also allows access to the low byte of the corresponding PCA channel's auto-reload value for 9 to 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| A write to this register will clear the module's ECOM bit to a 0. | | | | |

### 18.4.8 PCA0CPH0: PCA Channel 0 Capture Module High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPH0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xFC | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0CPH0 | 0x00 | RW | **PCA Channel 0 Capture Module High Byte.**<br>The PCA0CPH0 register holds the high byte (MSB) of the 16-bit capture module. This register address also allows access to the high byte of the corresponding PCA channel's auto-reload value for 9 to 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| A write to this register will set the module's ECOM bit to a 1. | | | | |

### 18.4.9 PCA0CPM1: PCA Channel 1 Capture/Compare Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PWM16 | ECOM | CAPP | CAPN | MAT | TOG | PWM | ECCF |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xDB | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | PWM16 | 0 | RW | **Channel 1 16-bit Pulse Width Modulation Enable.**<br>This bit enables 16-bit mode when Pulse Width Modulation mode is enabled.<br>Value: 0 – Name: 8_BIT – Description: 8 to 11-bit PWM selected.<br>Value: 1 – Name: 16_BIT – Description: 16-bit PWM selected. |
| 6 | ECOM | 0 | RW | **Channel 1 Comparator Function Enable.**<br>This bit enables the comparator function. |
| 5 | CAPP | 0 | RW | **Channel 1 Capture Positive Function Enable.**<br>This bit enables the positive edge capture capability. |
| 4 | CAPN | 0 | RW | **Channel 1 Capture Negative Function Enable.**<br>This bit enables the negative edge capture capability. |
| 3 | MAT | 0 | RW | **Channel 1 Match Function Enable.**<br>This bit enables the match function. When enabled, matches of the PCA counter with a module's capture/compare register cause the CCF1 bit in the PCA0MD register to be set to logic 1. |
| 2 | TOG | 0 | RW | **Channel 1 Toggle Function Enable.**<br>This bit enables the toggle function. When enabled, matches of the PCA counter with the capture/compare register cause the logic level on the CEX1 pin to toggle. If the PWM bit is also set to logic 1, the module operates in Frequency Output Mode. |
| 1 | PWM | 0 | RW | **Channel 1 Pulse Width Modulation Mode Enable.**<br>This bit enables the PWM function. When enabled, a pulse width modulated signal is output on the CEX1 pin. 8 to 11-bit PWM is used if PWM16 is cleared to 0; 16-bit mode is used if PWM16 is set to 1. If the TOG bit is also set, the module operates in Frequency Output Mode. |
| 0 | ECCF | 0 | RW | **Channel 1 Capture/Compare Flag Interrupt Enable.**<br>This bit sets the masking of the Capture/Compare Flag (CCF1) interrupt.<br>Value: 0 – Name: DISABLED – Description: Disable CCF1 interrupts.<br>Value: 1 – Name: ENABLED – Description: Enable a Capture/Compare Flag interrupt request when CCF1 is set. |

### 18.4.10 PCA0CPL1: PCA Channel 1 Capture Module Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPL1 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xE9 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0CPL1 | 0x00 | RW | **PCA Channel 1 Capture Module Low Byte.**<br>The PCA0CPL1 register holds the low byte (LSB) of the 16-bit capture module. This register address also allows access to the low byte of the corresponding PCA channel's auto-reload value for 9 to 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| A write to this register will clear the module's ECOM bit to a 0. | | | | |

### 18.4.11 PCA0CPH1: PCA Channel 1 Capture Module High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPH1 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xEA | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0CPH1 | 0x00 | RW | **PCA Channel 1 Capture Module High Byte.**<br>The PCA0CPH1 register holds the high byte (MSB) of the 16-bit capture module. This register address also allows access to the high byte of the corresponding PCA channel's auto-reload value for 9 to 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| A write to this register will set the module's ECOM bit to a 1. | | | | |

### 18.4.12 PCA0CPM2: PCA Channel 2 Capture/Compare Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PWM16 | ECOM | CAPP | CAPN | MAT | TOG | PWM | ECCF |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0xDC | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | PWM16 | 0 | RW | **Channel 2 16-bit Pulse Width Modulation Enable.**<br>This bit enables 16-bit mode when Pulse Width Modulation mode is enabled. |

| Value | Name | Description |
|---|---|---|
| 0 | 8_BIT | 8 to 11-bit PWM selected. |
| 1 | 16_BIT | 16-bit PWM selected. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | ECOM | 0 | RW | **Channel 2 Comparator Function Enable.**<br>This bit enables the comparator function. |
| 5 | CAPP | 0 | RW | **Channel 2 Capture Positive Function Enable.**<br>This bit enables the positive edge capture capability. |
| 4 | CAPN | 0 | RW | **Channel 2 Capture Negative Function Enable.**<br>This bit enables the negative edge capture capability. |
| 3 | MAT | 0 | RW | **Channel 2 Match Function Enable.**<br>This bit enables the match function. When enabled, matches of the PCA counter with a module's capture/compare register cause the CCF2 bit in the PCA0MD register to be set to logic 1. |
| 2 | TOG | 0 | RW | **Channel 2 Toggle Function Enable.**<br>This bit enables the toggle function. When enabled, matches of the PCA counter with the capture/compare register cause the logic level on the CEX2 pin to toggle. If the PWM bit is also set to logic 1, the module operates in Frequency Output Mode. |
| 1 | PWM | 0 | RW | **Channel 2 Pulse Width Modulation Mode Enable.**<br>This bit enables the PWM function. When enabled, a pulse width modulated signal is output on the CEX2 pin. 8 to 11-bit PWM is used if PWM16 is cleared to 0; 16-bit mode is used if PWM16 is set to 1. If the TOG bit is also set, the module operates in Frequency Output Mode. |
| 0 | ECCF | 0 | RW | **Channel 2 Capture/Compare Flag Interrupt Enable.**<br>This bit sets the masking of the Capture/Compare Flag (CCF2) interrupt. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable CCF2 interrupts. |
| 1 | ENABLED | Enable a Capture/Compare Flag interrupt request when CCF2 is set. |

### 18.4.13 PCA0CPL2: PCA Channel 2 Capture Module Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPL2 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xEB | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0CPL2 | 0x00 | RW | **PCA Channel 2 Capture Module Low Byte.**<br>The PCA0CPL2 register holds the low byte (LSB) of the 16-bit capture module. This register address also allows access to the low byte of the corresponding PCA channel's auto-reload value for 9 to 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| A write to this register will clear the module's ECOM bit to a 0. | | | | |

### 18.4.14 PCA0CPH2: PCA Channel 2 Capture Module High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCA0CPH2 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xEC | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | PCA0CPH2 | 0x00 | RW | **PCA Channel 2 Capture Module High Byte.**<br>The PCA0CPH2 register holds the high byte (MSB) of the 16-bit capture module. This register address also allows access to the high byte of the corresponding PCA channel's auto-reload value for 9 to 11-bit PWM mode. The ARSEL bit in register PCA0PWM controls which register is accessed. |
| A write to this register will set the module's ECOM bit to a 1. | | | | |

# 19. Serial Peripheral Interface (SPI0)

## 19.1 Introduction

The serial peripheral interface (SPI) module provides access to a flexible, full-duplex synchronous serial bus. The SPI can operate as a master or slave device in both 3-wire or 4-wire modes, and supports multiple masters and slaves on a single SPI bus. The slave-select (NSS) signal can be configured as an input to select the SPI in slave mode, or to disable master mode operation in a multi-master environment, avoiding contention on the SPI bus when more than one master attempts simultaneous data transfers. NSS can also be configured as a firmware-controlled chip-select output in master mode, or disabled to reduce the number of pins required. Additional general purpose port I/O pins can be used to select multiple slave devices in master mode.

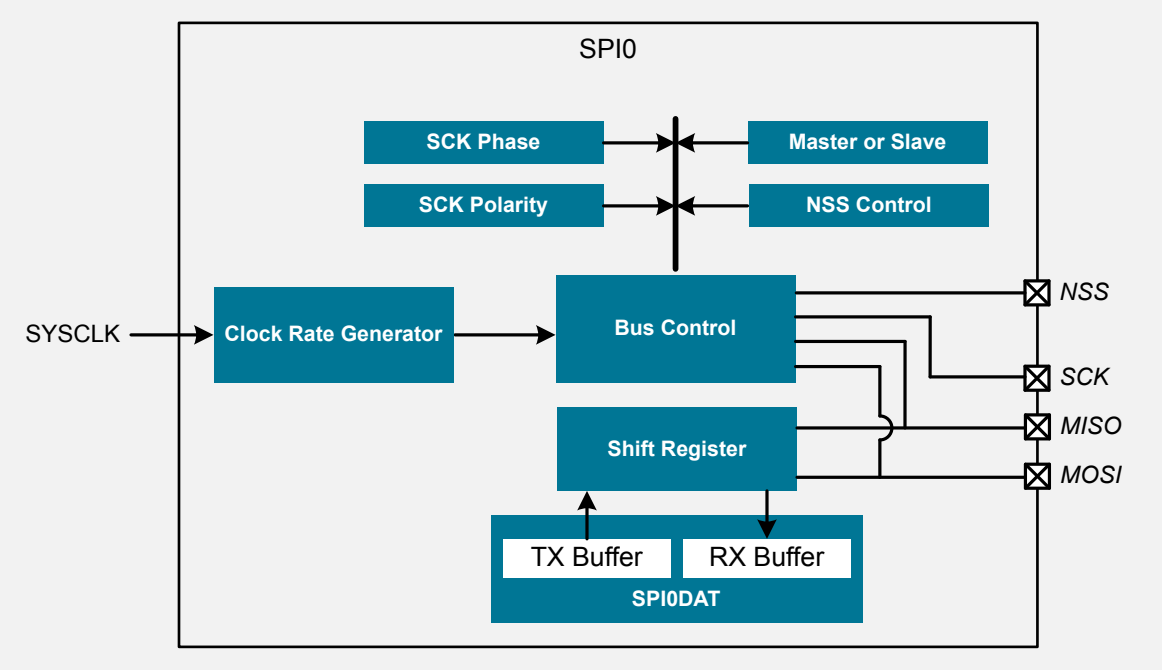



**Figure 19.1. SPI Block Diagram**

## 19.2 Features

The SPI module includes the following features:

- Supports 3- or 4-wire operation in master or slave modes.
- Supports external clock frequencies up to SYSCLK / 2 in master mode and SYSCLK / 10 in slave mode.
- Support for four clock phase and polarity options.
- 8-bit dedicated clock clock rate generator.
- Support for multiple masters on the same data lines.

## 19.3 Functional Description

### 19.3.1 Signals

The SPI interface consists of up to four signals: MOSI, MISO, SCK, and NSS.

**Master Out, Slave In (MOSI):** The MOSI signal is the data output pin when configured as a master device and the data input pin when configured as a slave. It is used to serially transfer data from the master to the slave. Data is transferred on the MOSI pin most-significant bit first. When configured as a master, MOSI is driven from the internal shift register in both 3- and 4-wire mode.

**Master In, Slave Out (MISO):** The MISO signal is the data input pin when configured as a master device and the data output pin when configured as a slave. It is used to serially transfer data from the slave to the master. Data is transferred on the MISO pin most-significant bit first. The MISO pin is placed in a high-impedance state when the SPI module is disabled or when the SPI operates in 4-wire mode as a slave that is not selected. When acting as a slave in 3-wire mode, MISO is always driven from the internal shift register.

**Serial Clock (SCK):** The SCK signal is an output from the master device and an input to slave devices. It is used to synchronize the transfer of data between the master and slave on the MOSI and MISO lines. The SPI module generates this signal when operating as a master and receives it as a slave. The SCK signal is ignored by a SPI slave when the slave is not selected in 4-wire slave mode.

**Slave Select (NSS):** The function of the slave-select (NSS) signal is dependent on the setting of the NSSMD bitfield. There are three possible modes that can be selected with these bits:

- NSSMD[1:0] = 00: 3-Wire Master or 3-Wire Slave Mode: The SPI operates in 3-wire mode, and NSS is disabled. When operating as a slave device, the SPI is always selected in 3-wire mode. Since no select signal is present, the SPI must be the only slave on the bus in 3-wire mode. This is intended for point-to-point communication between a master and a single slave.
- NSSMD[1:0] = 01: 4-Wire Slave or Multi-Master Mode: The SPI operates in 4-wire mode, and NSS is configured as an input. When operating as a slave, NSS selects the SPI device. When operating as a master, a 1-to- 0 transition of the NSS signal disables the master function of the SPI module so that multiple master devices can be used on the same SPI bus.
- NSSMD[1:0] = 1x: 4-Wire Master Mode: The SPI operates in 4-wire mode, and NSS is enabled as an output. The setting of NSSMD0 determines what logic level the NSS pin will output. This configuration should only be used when operating the SPI as a master device.

The setting of NSSMD bits affects the pinout of the device. When in 3-wire master or 3-wire slave mode, the NSS pin will not be mapped by the crossbar. In all other modes, the NSS signal will be mapped to a pin on the device.

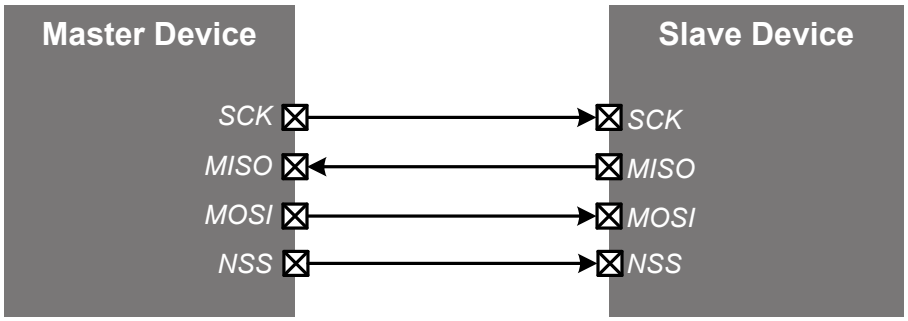



**Figure 19.2. 4-Wire Connection Diagram**

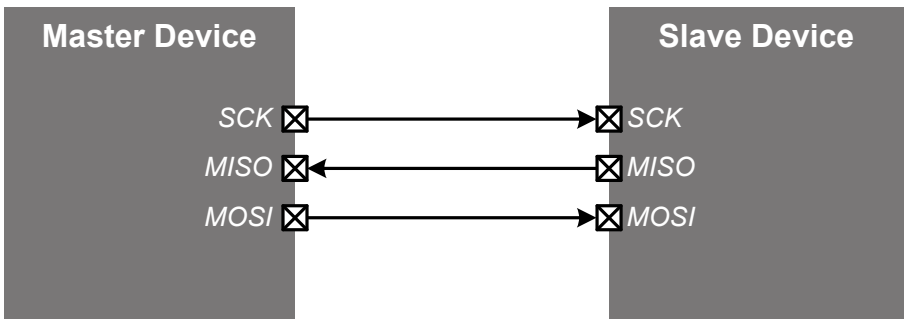



**Figure 19.3. 3-Wire Connection Diagram**

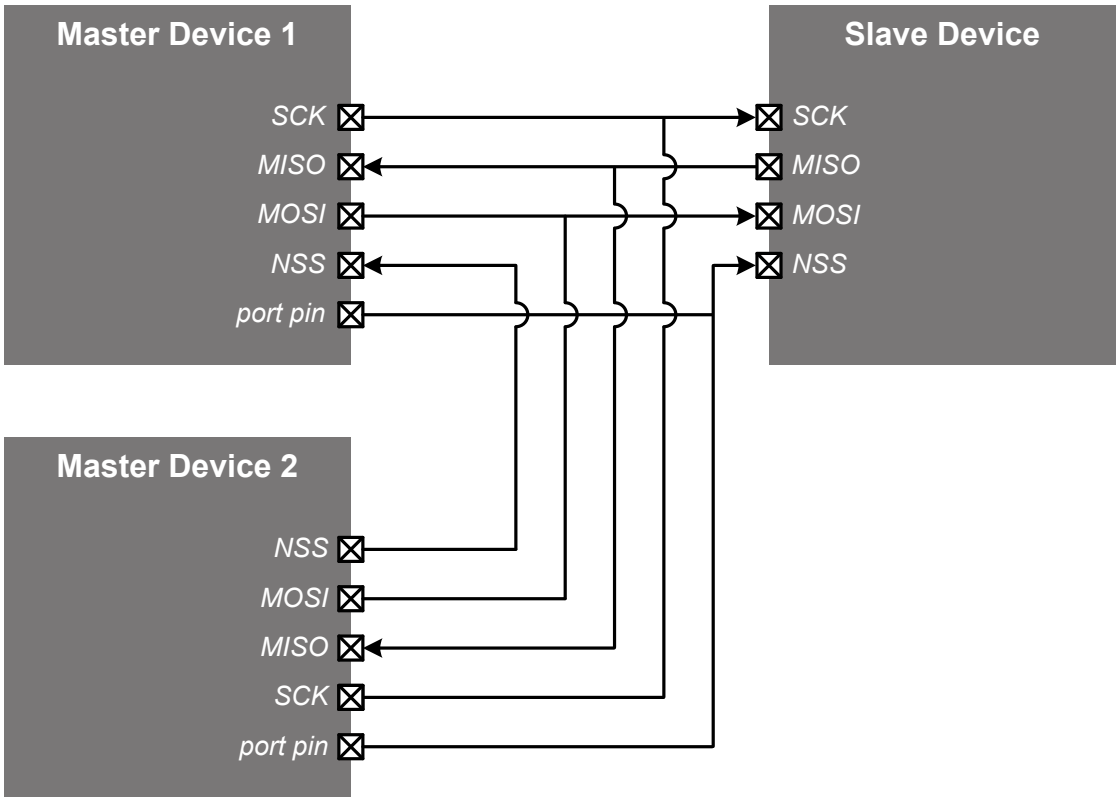



**Figure 19.4. Multi-Master Connection Diagram**

### 19.3.2 Master Mode Operation

An SPI master device initiates all data transfers on a SPI bus. It drives the SCK line and controls the speed at which data is transferred. To place the SPI in master mode, the MSTEN bit should be set to 1. Writing a byte of data to the SPInDAT register writes to the transmit buffer. If the SPI shift register is empty, a byte is moved from the transmit buffer into the shift register, and a bi-directional data transfer begins. The SPI module provides the serial clock on SCK, while simultaneously shifting data out of the shift register MSB-first on MOSI and into the shift register MSB-first on MISO. Upon completing a transfer, the data received is moved from the shift register into the receive buffer. If the transmit buffer is not empty, the next byte in the transmit buffer will be moved into the shift register and the next data transfer will begin. If no new data is available in the transmit buffer, the SPI will halt and wait for new data to initiate the next transfer. Bytes that have been received and stored in the receive buffer may be read from the buffer via the SPInDAT register.

### 19.3.3 Slave Mode Operation

When the SPI block is enabled and not configured as a master, it will operate as a SPI slave. As a slave, bytes are shifted in through the MOSI pin and out through the MISO pin by an external master device controlling the SCK signal. A bit counter in the SPI logic counts SCK edges. When 8 bits have been shifted through the shift register, a byte is copied into the receive buffer. Data is read from the receive buffer by reading SPInDAT. A slave device cannot initiate transfers. Data to be transferred to the master device is pre-loaded into the transmit buffer by writing to SPInDAT and will transfer to the shift register on byte boundaries in the order in which they were written to the buffer.

When configured as a slave, SPI0 can be configured for 4-wire or 3-wire operation. In the default, 4-wire slave mode, the NSS signal is routed to a port pin and configured as a digital input. The SPI interface is enabled when NSS is logic 0, and disabled when NSS is logic 1. The internal shift register bit counter is reset on a falling edge of NSS. When operated in 3-wire slave mode, NSS is not mapped to an external port pin through the crossbar. Since there is no way of uniquely addressing the device in 3-wire slave mode, the SPI must be the only slave device present on the bus. It is important to note that in 3-wire slave mode there is no external means of resetting the bit counter that determines when a full byte has been received. The bit counter can only be reset by disabling and re-enabling the SPI module with the SPIEN bit.

### 19.3.4 Clock Phase and Polarity

Four combinations of serial clock phase and polarity can be selected using the clock control bits in the SPInCFG register. The CKPHA bit selects one of two clock phases (edge used to latch the data). The CKPOL bit selects between an active-high or active-low clock. Both master and slave devices must be configured to use the same clock phase and polarity. The SPI module should be disabled (by clearing the SPIEN bit) when changing the clock phase or polarity. Note that CKPHA should be set to 0 on both the master and slave SPI when communicating between two Silicon Labs devices.

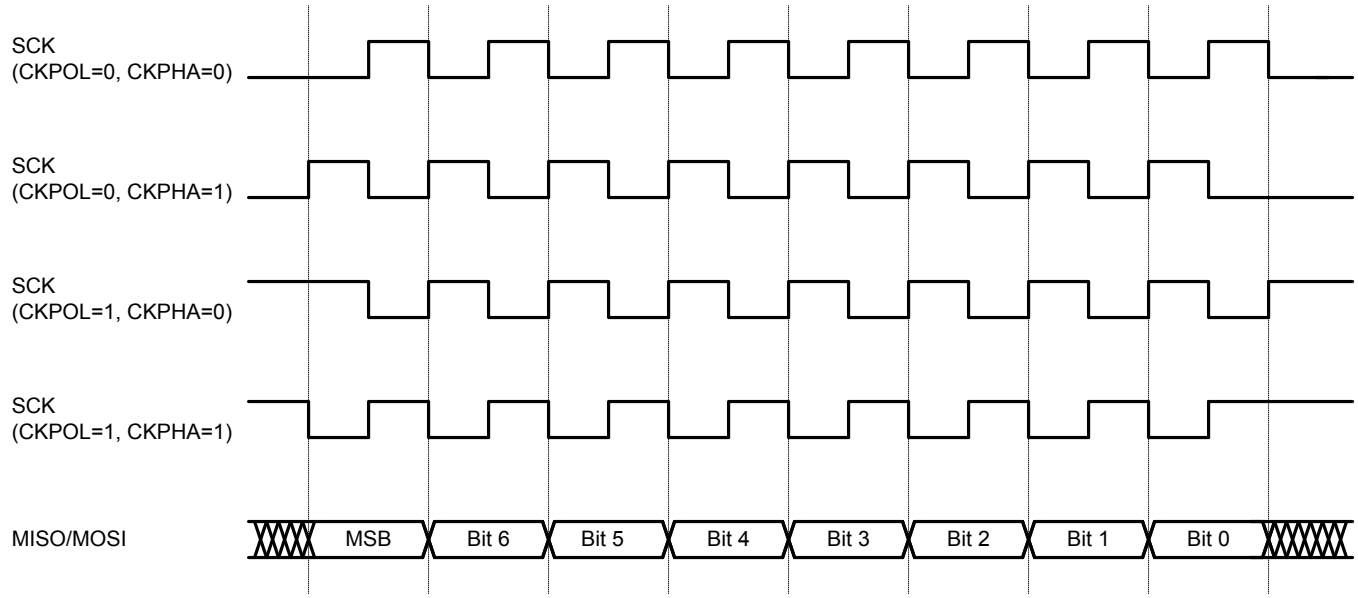



**Figure 19.5. Master Mode Data/Clock Timing**

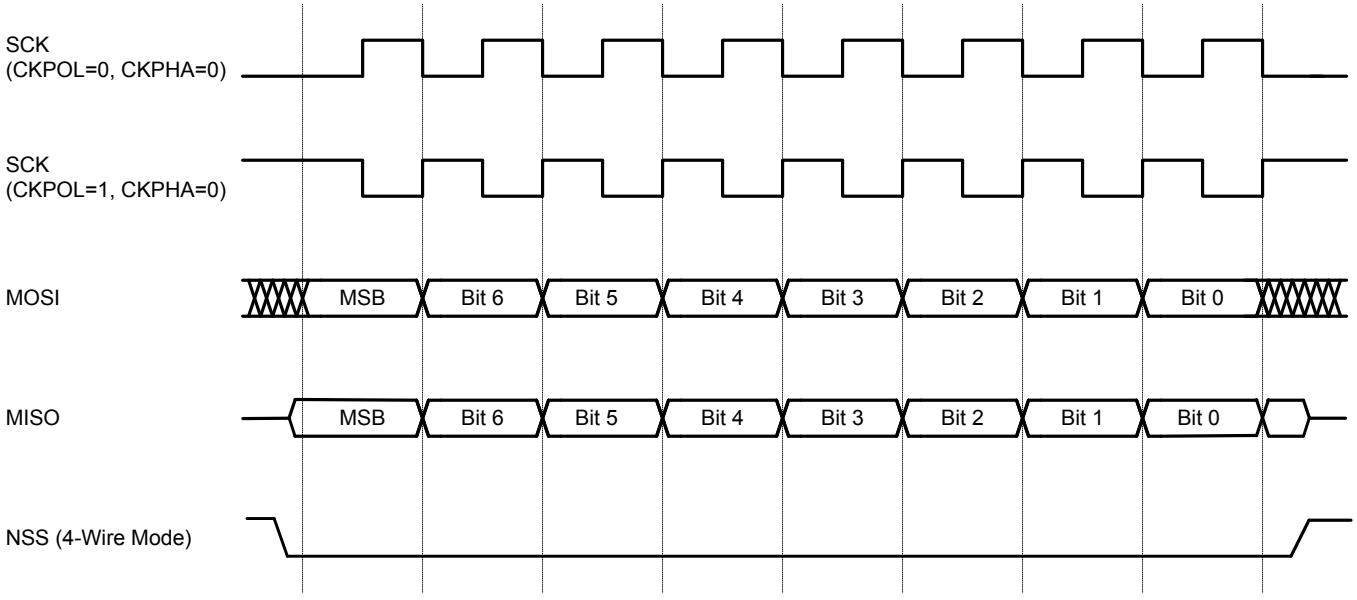



**Figure 19.6. Slave Mode Data/Clock Timing (CKPHA = 0)**

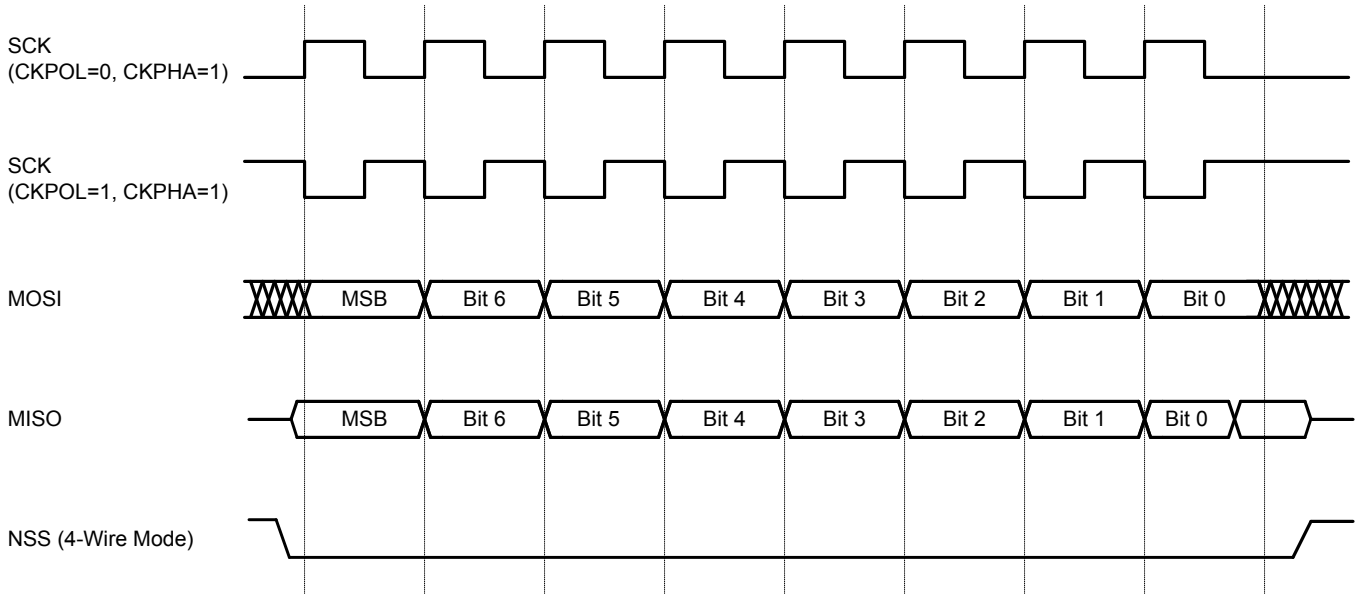



**Figure 19.7. Slave Mode Data/Clock Timing (CKPHA = 1)**

### 19.3.5 Basic Data Transfer

The SPI bus is inherently full-duplex. It sends and receives a single byte on every transfer. The SPI peripheral may be operated on a byte-by-byte basis using the SPInDAT register and the SPIF flag. The method firmware uses to send and receive data through the SPI interface is the same in either mode, but the hardware will react differently.

#### Master Transfers

As an SPI master, all transfers are initiated with a write to SPInDAT, and the SPIF flag will be set by hardware to indicate the end of each transfer. The general method for a single-byte master transfer follows:

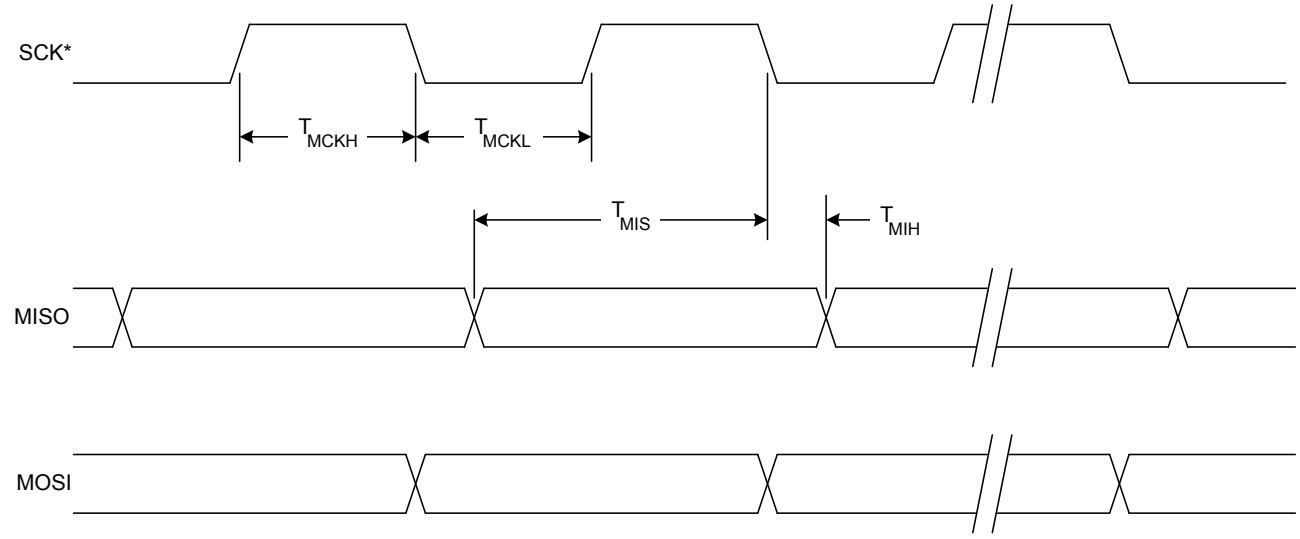
1. Write the data to be sent to SPInDAT. The transfer will begin on the bus at this time.
2. Wait for the SPIF flag to generate an interrupt, or poll SPIF until it is set to 1.
3. Read the received data from SPInDAT.
4. Clear the SPIF flag to 0.
5. Repeat the sequence for any additional transfers.

#### Slave Transfers

As a SPI slave, the transfers are initiated by an external master device driving the bus. Slave firmware may anticipate any output data needs by pre-loading the SPInDAT register before the master begins the transfer.

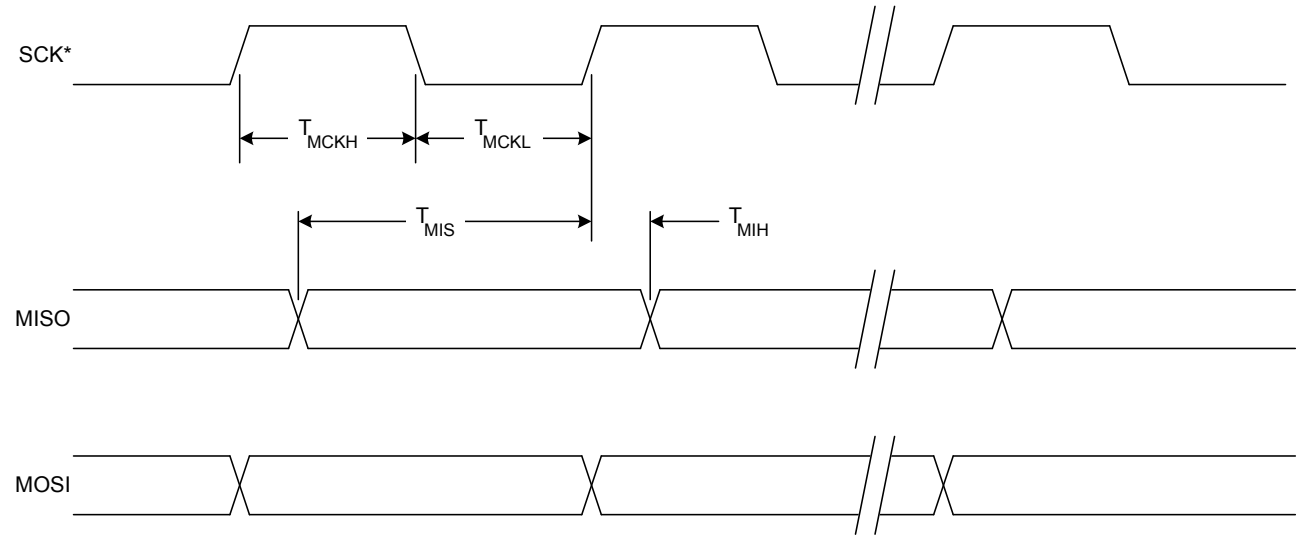
1. Write any data to be sent to SPInDAT. The transfer will not begin until the external master device initiates it.
2. Wait for the SPIF flag to generate an interrupt, or poll SPIF until it is set to 1.
3. Read the received data from SPInDAT.
4. Clear the SPIF flag to 0.
5. Repeat the sequence for any additional transfers.

### 19.3.6 SPI Timing Diagrams



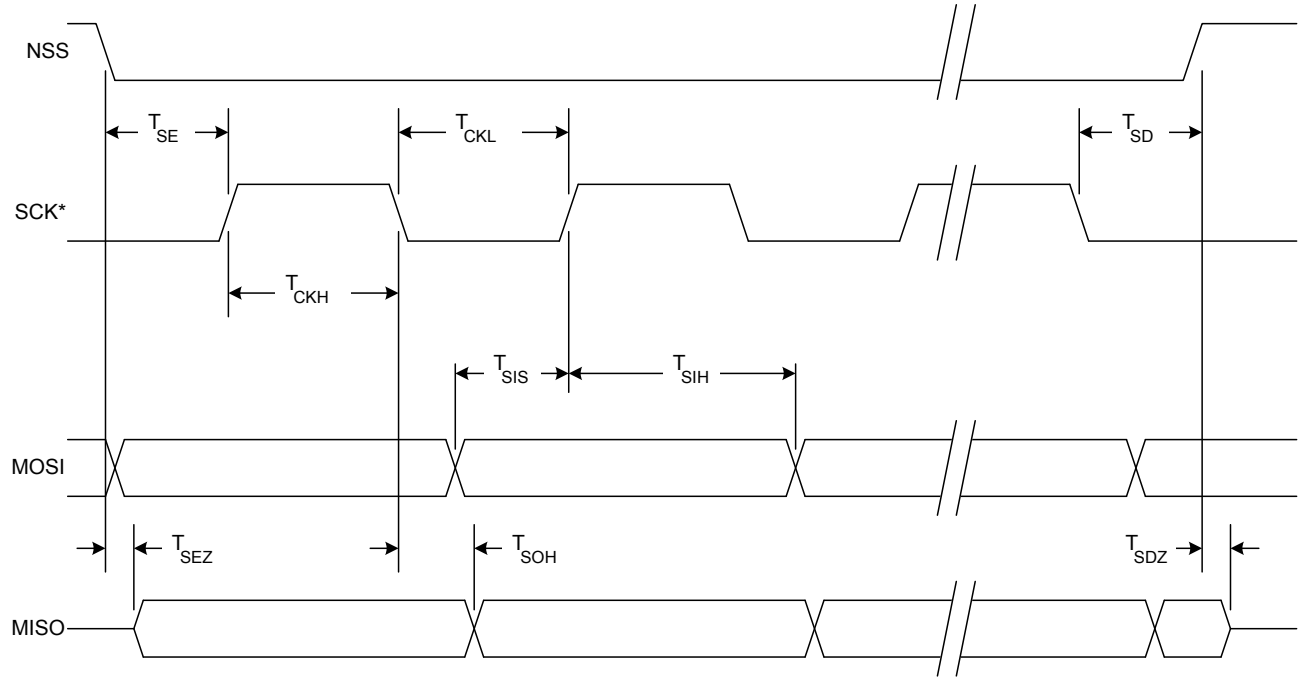

* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

**Figure 19.8. SPI Master Timing (CKPHA = 0)**



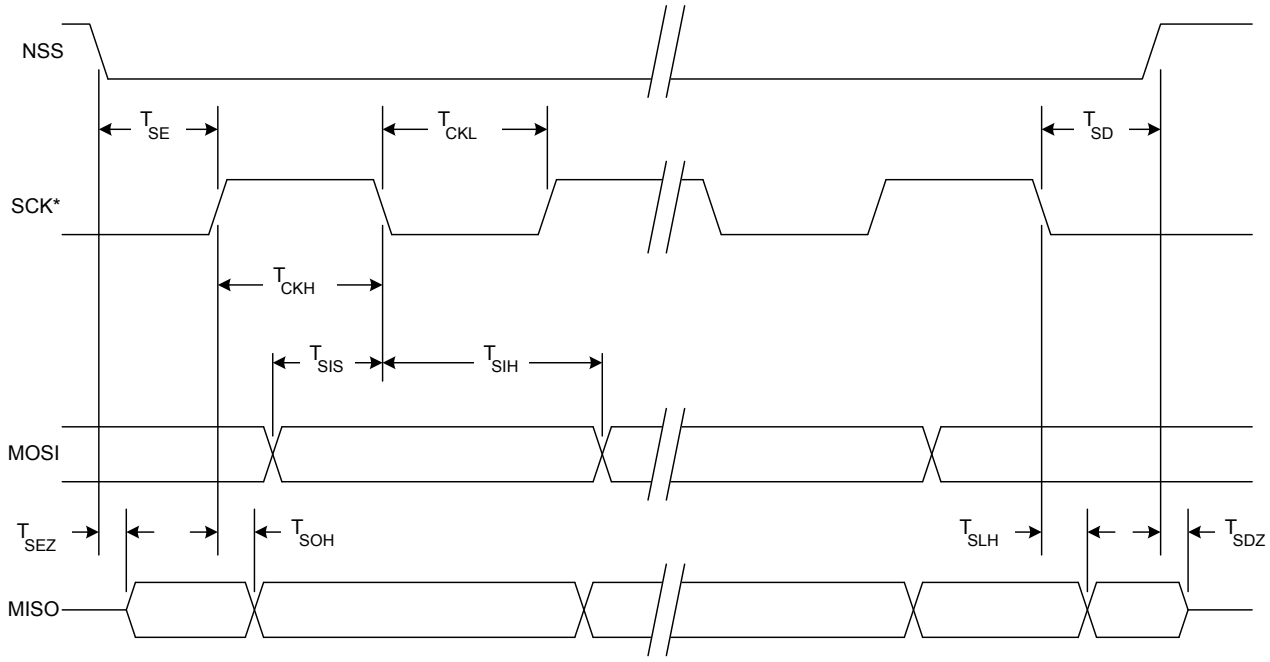

* SCK is shown for CKPOL = 0. SCK is the opposite polarity for CKPOL = 1.

**Figure 19.9. SPI Master Timing (CKPHA = 1)**




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

**Figure 19.10. SPI Slave Timing (CKPHA = 0)**




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

**Figure 19.11. SPI Slave Timing (CKPHA = 1)**

**Table 19.1. SPI Timing Parameters**

| Parameter | Description | Min | Max | Units |
|---|---|---|---|---|
| **Master Mode Timing** | | | | |
| $T_{MCKH}$ | SCK High Time | 1 x $T_{SYSCLK}$ | — | ns |
| $T_{MCKL}$ | SCK Low Time | 1 x $T_{SYSCLK}$ | — | ns |
| $T_{MIS}$ | MISO Valid to SCK Shift Edge | 1 x $T_{SYSCLK}$ + 20 | — | ns |
| $T_{MIH}$ | SCK Shift Edge to MISO Change | 0 | — | ns |
| **Slave Mode Timing** | | | | |
| $T_{SE}$ | NSS Falling to First SCK Edge | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SD}$ | Last SCK Edge to NSS Rising | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SEZ}$ | NSS Falling to MISO Valid | — | 4 x $T_{SYSCLK}$ | ns |
| $T_{SDZ}$ | NSS Rising to MISO High-Z | — | 4 x $T_{SYSCLK}$ | ns |
| $T_{CKH}$ | SCK High Time | 5 x $T_{SYSCLK}$ | — | ns |
| $T_{CKL}$ | SCK Low Time | 5 x $T_{SYSCLK}$ | — | ns |
| $T_{SIS}$ | MOSI Valid to SCK Sample Edge | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SIH}$ | SCK Sample Edge to MOSI Change | 2 x $T_{SYSCLK}$ | — | ns |
| $T_{SOH}$ | SCK Shift Edge to MISO Change | — | 4 x $T_{SYSCLK}$ | ns |
| $T_{SLH}$ | Last SCK Edge to MISO Change (CKPHA = 1 ONLY) | 6 x $T_{SYSCLK}$ | 8 x $T_{SYSCLK}$ | ns |

**Note:**
1. $T_{SYSCLK}$ is equal to one period of the device system clock (SYSCLK).

## 19.4 SPI0 Control Registers

### 19.4.1 SPI0CFG: SPI0 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPIBSY | MSTEN | CKPHA | CKPOL | SLVSEL | NSSIN | SRMT | RXBMT |
| Access | R | RW | RW | RW | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| SFR Page = 0x0; SFR Address: 0xA1 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | SPIBSY | 0 | R | **SPI Busy.**<br>This bit is set to logic 1 when a SPI transfer is in progress (master or slave mode). |
| 6 | MSTEN | 0 | RW | **Master Mode Enable.** |
| | Value | Name | | Description |
| | 0 | MASTER_DISABLED | | Disable master mode. Operate in slave mode. |
| | 1 | MASTER_ENABLED | | Enable master mode. Operate as a master. |
| 5 | CKPHA | 0 | RW | **SPI0 Clock Phase.** |
| | Value | Name | | Description |
| | 0 | DATA_CEN-TERED_FIRST | | Data centered on first edge of SCK period. |
| | 1 | DATA_CEN-TERED_SECOND | | Data centered on second edge of SCK period. |
| 4 | CKPOL | 0 | RW | **SPI0 Clock Polarity.** |
| | Value | Name | | Description |
| | 0 | IDLE_LOW | | SCK line low in idle state. |
| | 1 | IDLE_HIGH | | SCK line high in idle state. |
| 3 | SLVSEL | 0 | R | **Slave Selected Flag.**<br>This bit is set to logic 1 whenever the NSS pin is low indicating SPI0 is the selected slave. It is cleared to logic 0 when NSS is high (slave not selected). This bit does not indicate the instantaneous value at the NSS pin, but rather a de-glitched version of the pin input. |
| 2 | NSSIN | 1 | R | **NSS Instantaneous Pin Input.**<br>This bit mimics the instantaneous value that is present on the NSS port pin at the time that the register is read. This input is not de-glitched. |
| 1 | SRMT | 1 | R | **Shift Register Empty.**<br>This bit will be set to logic 1 when all data has been transferred in/out of the shift register, and there is no new information available to read from the transmit buffer or write to the receive buffer. It returns to logic 0 when a data byte is transferred to the shift register from the transmit buffer or by a transition on SCK. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 0 | RXBMT | 1 | R | **Receive Buffer Empty.**<br>This bit is valid in slave mode only and will be set to logic 1 when the receive buffer has been read and contains no new information. If there is new information available in the receive buffer that has not been read, this bit will return to logic 0. RXBMT = 1 when in Master Mode. |
| In slave mode, data on MOSI is sampled in the center of each data bit. In master mode, data on MISO is sampled one SYSCLK before the end of each data bit, to provide maximum settling time for the slave device. | | | | |

### 19.4.2 SPI0CN0: SPI0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPIF | WCOL | MODF | RXOVRN | NSSMD | | TXBMT | SPIEN |
| Access | RW | RW | RW | RW | RW | | R | RW |
| Reset | 0 | 0 | 0 | 0 | 0x1 | | 1 | 0 |
| SFR Page = 0x0; SFR Address: 0xF8 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | SPIF | 0 | RW | **SPI0 Interrupt Flag.**<br>This bit is set to logic 1 by hardware at the end of a data transfer. If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by firmware. |
| 6 | WCOL | 0 | RW | **Write Collision Flag.**<br>This bit is set to logic 1 if a write to SPI0DAT is attempted when TXBMT is 0. When this occurs, the write to SPI0DAT will be ignored, and the transmit buffer will not be written. If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by firmware. |
| 5 | MODF | 0 | RW | **Mode Fault Flag.**<br>This bit is set to logic 1 by hardware when a master mode collision is detected (NSS is low, MSTEN = 1, and NSSMD = 01). If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by firmware. |
| 4 | RXOVRN | 0 | RW | **Receive Overrun Flag.**<br>This bit is valid for slave mode only and is set to logic 1 by hardware when the receive buffer still holds unread data from a previous transfer and the last bit of the current transfer is shifted into the SPI0 shift register. If SPI interrupts are enabled, an interrupt will be generated. This bit is not automatically cleared by hardware, and must be cleared by firmware. |
| 3:2 | NSSMD | 0x1 | RW | **Slave Select Mode.**<br>Selects between the following NSS operation modes: |

| Value | Name | Description |
|---|---|---|
| 0x0 | 3_WIRE | 3-Wire Slave or 3-Wire Master Mode. NSS signal is not routed to a port pin. |
| 0x1 | 4_WIRE_SLAVE | 4-Wire Slave or Multi-Master Mode. NSS is an input to the device. |
| 0x2 | 4_WIRE_MASTER_NSS_LOW | 4-Wire Single-Master Mode. NSS is an output and logic low. |
| 0x3 | 4_WIRE_MASTER_NSS_HIGH | 4-Wire Single-Master Mode. NSS is an output and logic high. |

| | | | | |
|---|---|---|---|---|
| 1 | TXBMT | 1 | R | **Transmit Buffer Empty.**<br>This bit will be set to logic 0 when new data has been written to the transmit buffer. When data in the transmit buffer is transferred to the SPI shift register, this bit will be set to logic 1, indicating that it is safe to write a new byte to the transmit buffer. |
| 0 | SPIEN | 0 | RW | **SPI0 Enable.** |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable the SPI module. |
| 1 | ENABLED | Enable the SPI module. |

### 19.4.3 SPI0CKR: SPI0 Clock Rate

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPI0CKR | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xA2 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | SPI0CKR | 0x00 | RW | **SPI0 Clock Rate.**<br>These bits determine the frequency of the SCK output when the SPI0 module is configured for master mode operation. The SCK clock frequency is a divided version of the system clock, and is given in the following equation, where SYSCLK is the system clock frequency and SPI0CKR is the 8-bit value held in the SPI0CKR register.<br>fsck = SYSCLK / (2 * (SPI0CKR + 1))<br>for 0 <= SPI0CKR <= 255 |

### 19.4.4 SPI0DAT: SPI0 Data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SPI0DAT | | | | | | | |
| Access | RW | | | | | | | |
| Reset | Varies | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xA3 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | SPI0DAT | Varies | RW | **SPI0 Transmit and Receive Data.**<br>The SPI0DAT register is used to transmit and receive SPI0 data. Writing data to SPI0DAT places the data into the transmit buffer and initiates a transfer when in master mode. A read of SPI0DAT returns the contents of the receive buffer. |

# 20. System Management Bus / I2C (SMB0)

## 20.1 Introduction

The SMBus I/O interface is a two-wire, bi-directional serial bus. The SMBus is compliant with the System Management Bus Specification, version 1.1, and compatible with the $I^2C$ serial bus.

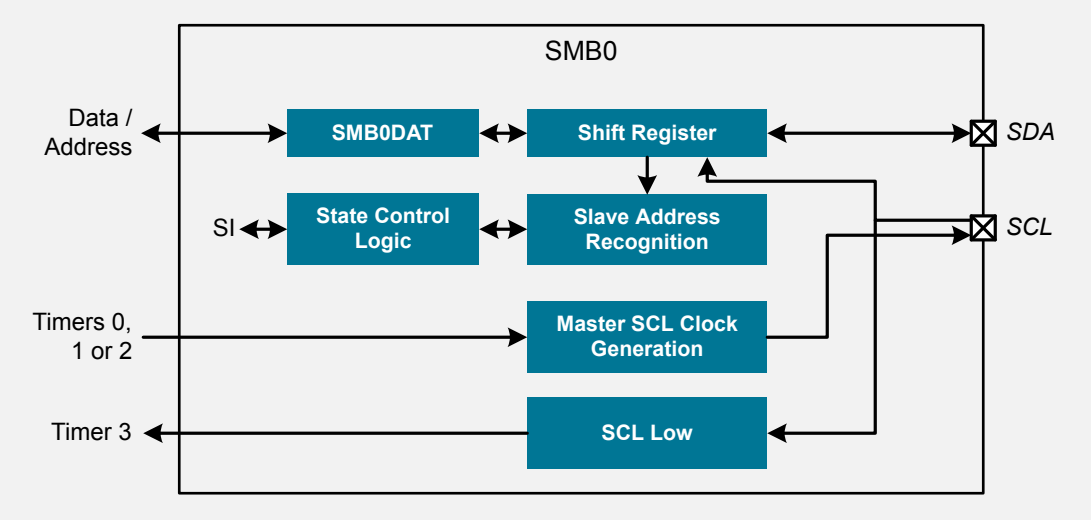



**Figure 20.1. SMBus 0 Block Diagram**

## 20.2 Features

The SMBus module includes the following features:

- Standard (up to 100 kbps) and Fast (400 kbps) transfer speeds.
- Support for master, slave, and multi-master modes.
- Hardware synchronization and arbitration for multi-master mode.
- Clock low extending (clock stretching) to interface with faster masters.
- Hardware support for 7-bit slave and general call address recognition.
- Firmware support for 10-bit slave address decoding.
- Ability to inhibit all slave states.
- Programmable data setup/hold times.

## 20.3 Functional Description

### 20.3.1 Supporting Documents

It is assumed the reader is familiar with or has access to the following supporting documents:

- The $I^2C$-Bus and How to Use It (including specifications), Philips Semiconductor.
- The $I^2C$-Bus Specification—Version 2.0, Philips Semiconductor.
- System Management Bus Specification—Version 1.1, SBS Implementers Forum.

### 20.3.2 SMBus Protocol

The SMBus specification allows any recessive voltage between 3.0 and 5.0 V; different devices on the bus may operate at different voltage levels. However, the maximum voltage on any port pin must conform to the electrical characteristics specifications. The bi-directional SCL (serial clock) and SDA (serial data) lines must be connected to a positive power supply voltage through a pullup resistor or similar circuit. Every device connected to the bus must have an open-drain or open-collector output for both the SCL and SDA lines, so that both are pulled high (recessive state) when the bus is free. The maximum number of devices on the bus is limited only by the requirement that the rise and fall times on the bus not exceed 300 ns and 1000 ns, respectively.

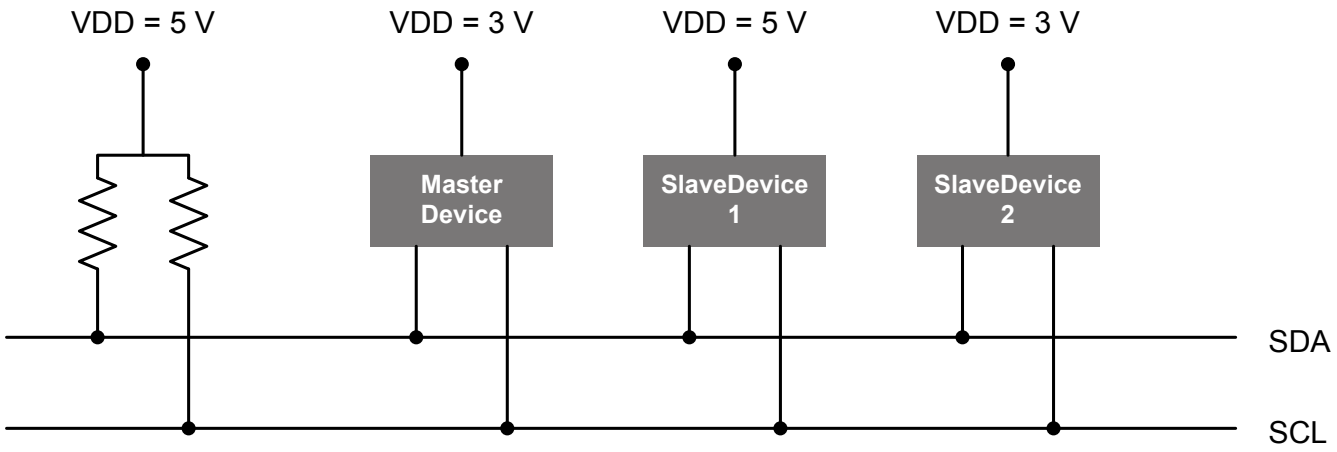



**Figure 20.2. Typical SMBus System Connection**

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. 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. Bytes that are received (by a master or slave) are acknowledged (ACK) with a low SDA during a high SCL (see Figure 20.3 SMBus Transaction on page 215). 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 20.3 SMBus Transaction on page 215 illustrates a typical SMBus transaction.

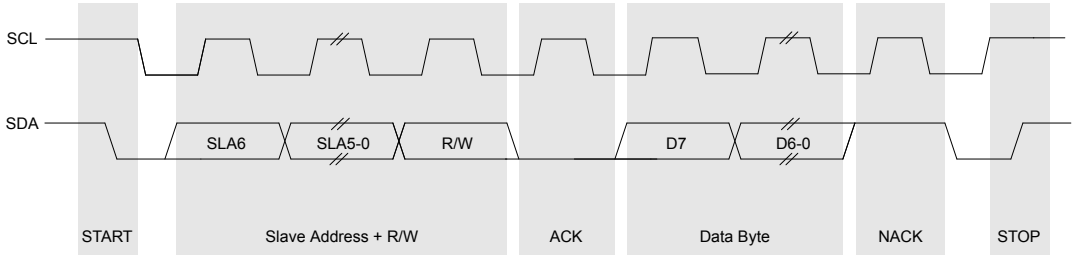



**Figure 20.3. SMBus Transaction**

### Transmitter vs. Receiver

On the SMBus communications interface, a device is the “transmitter” when it is sending an address or data byte to another device on the bus. A device is a “receiver” when an address or data byte is being sent to it from another device on the bus. The transmitter controls the SDA line during the address or data byte. After each byte of address or data information is sent by the transmitter, the receiver sends an ACK or NACK bit during the ACK phase of the transfer, during which time the receiver controls the SDA line.

### 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 ● SCL High (SMBus Free) Timeout on page 215). 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.

### Clock Low Extension

SMBus provides a clock synchronization mechanism, similar to $I^2C$, which allows devices with different speed capabilities to coexist on the bus. A clock-low extension is used during a transfer in order to allow slower slave devices to communicate with faster masters. The slave may temporarily hold the SCL line LOW to extend the clock low period, effectively decreasing the serial clock frequency.

### SCL Low Timeout

If the SCL line is held low by a slave device on the bus, no further communication is possible. Furthermore, the master cannot force the SCL line high to correct the error condition. To solve this problem, the SMBus protocol specifies that devices participating in a transfer must detect any clock cycle held low longer than 25 ms as a “timeout” condition. Devices that have detected the timeout condition must reset the communication no later than 10 ms after detecting the timeout condition.

For the SMBus 0 interface, Timer 3 is used to implement SCL low timeouts. The SCL low timeout feature is enabled by setting the SMB0TOE bit in SMB0CF. The associated timer is forced to reload when SCL is high, and allowed to count when SCL is low. With the associated timer enabled and configured to overflow after 25 ms (and SMB0TOE set), the timer interrupt service routine can be used to reset (disable and re-enable) the SMBus in the event of an SCL low timeout.

### SCL High (SMBus Free) Timeout

The SMBus specification stipulates that if the SCL and SDA lines remain high for more that 50 µs, the bus is designated as free. When the SMB0FTE bit in SMB0CF is set, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods (as defined by the timer configured for the SMBus clock source). If the SMBus is waiting to generate a Master START, the START will be generated following this timeout. A clock source is required for free timeout detection, even in a slave-only implementation.

### 20.3.3 Configuring the SMBus Module

The SMBus can operate in both Master and Slave modes. The interface provides timing and shifting control for serial transfers; higher level protocol is determined by user software. The SMBus interface provides the following application-independent features:

- Byte-wise serial data transfers
- Clock signal generation on SCL (Master Mode only) and SDA data synchronization
- Timeout/bus error recognition, as defined by the SMB0CF configuration register
- START/STOP timing, detection, and generation
- Bus arbitration
- Interrupt generation
- Status information
- Optional hardware recognition of slave address and automatic acknowledgement of address/data

SMBus interrupts are generated for each data byte or slave address that is transferred. When hardware acknowledgement is disabled, the point at which the interrupt is generated depends on whether the hardware is acting as a data transmitter or receiver. When a transmitter (i.e., sending address/data, receiving an ACK), this interrupt is generated after the ACK cycle so that software may read the received ACK value; when receiving data (i.e., receiving address/data, sending an ACK), this interrupt is generated before the ACK cycle so that software may define the outgoing ACK value. If hardware acknowledgement is enabled, these interrupts are always generated after the ACK cycle. Interrupts are also generated to indicate the beginning of a transfer when a master (START generated), or the end of a transfer when a slave (STOP detected). Software should read the SMB0CN0 register to find the cause of the SMBus interrupt.

### SMBus Configuration Register

The SMBus Configuration register (SMB0CF) is used to enable the SMBus master and/or slave modes, select the SMBus clock source, and select the SMBus timing and timeout options. When the ENSMB bit is set, the SMBus is enabled for all master and slave events. Slave events may be disabled by setting the INH bit. With slave events inhibited, the SMBus interface will still monitor the SCL and SDA pins; however, the interface will NACK all received addresses and will not generate any slave interrupts. When the INH bit is set, all slave events will be inhibited following the next START (interrupts will continue for the duration of the current transfer).

The SMBCS bit field selects the SMBus clock source, which is used only when operating as a master or when the Free Timeout detection is enabled. When operating as a master, overflows from the selected source determine both the bit rate and the absolute minimum SCL low and high times. The selected clock source may be shared by other peripherals so long as the timer is left running at all times. The selected clock source should typically be configured to overflow at three times the desired bit rate. When the interface is operating as a master (and SCL is not driven or extended by any other devices on the bus), the device will hold the SCL line low for one overflow period, and release it for two overflow periods. $T_{HIGH}$ is typically twice as large as $T_{LOW}$. The actual SCL output may vary due to other devices on the bus (SCL may be extended low by slower slave devices, driven low by contending master devices, or have long ramp times). The SMBus hardware will ensure that once SCL does return high, it reads a logic high state for a minimum of one overflow period.

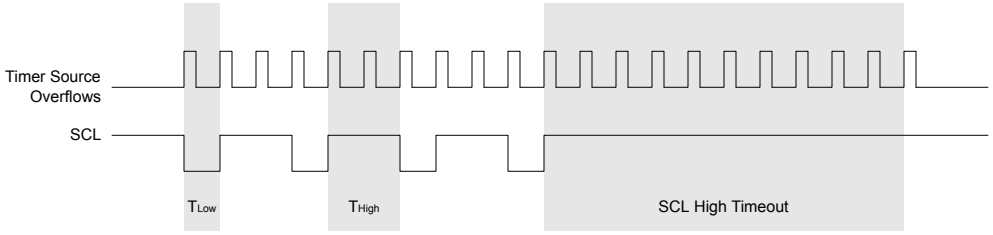



**Figure 20.4. Typical SMBus SCL Generation**

Setting the EXTHOLD bit extends the minimum setup and hold times for the SDA line. The minimum SDA setup time defines the absolute minimum time that SDA is stable before SCL transitions from low-to-high. The minimum SDA hold time defines the absolute minimum time that the current SDA value remains stable after SCL transitions from high-to-low. EXTHOLD should be set so that the minimum setup and hold times meet the SMBus Specification requirements of 250 ns and 300 ns, respectively. Setup and hold time extensions are typically necessary for SMBus compliance when SYSCLK is above 10 MHz.

**Table 20.1. Minimum SDA Setup and Hold Times**

| EXTHOLD | Minimum SDA Setup Time | Minimum SDA Hold Time |
|---|---|---|
| 0 | $T_{low}$ – 4 system clocks or 1 system clock + s/w delay | 3 system clocks |
| 1 | 11 system clocks | 12 system clocks |
| **Note:** Setup Time for ACK bit transmissions and the MSB of all data transfers. When using software acknowledgment, the s/w delay occurs between the time SMB0DAT or ACK is written and when SI is cleared. Note that if SI is cleared in the same write that defines the outgoing ACK value, s/w delay is zero. | | |

With the SMBTOE bit set, Timer 3 should be configured to overflow after 25 ms in order to detect SCL low timeouts. The SMBus interface will force the associated timer to reload while SCL is high, and allow the timer to count when SCL is low. The timer interrupt service routine should be used to reset SMBus communication by disabling and re-enabling the SMBus. SMBus Free Timeout detection can be enabled by setting the SMBFTE bit. When this bit is set, the bus will be considered free if SDA and SCL remain high for more than 10 SMBus clock source periods.

### SMBus Pin Swap

The SMBus peripheral is assigned to pins using the priority crossbar decoder. By default, the SMBus signals are assigned to port pins starting with SDA on the lower-numbered pin, and SCL on the next available pin. The SWAP bit in the SMBTC register can be set to 1 to reverse the order in which the SMBus signals are assigned.

### SMBus Timing Control

The SDD field in the SMBTC register is used to restrict the detection of a START condition under certain circumstances. In some systems where there is significant mismatch between the impedance or the capacitance on the SDA and SCL lines, it may be possible for SCL to fall after SDA during an address or data transfer. Such an event can cause a false START detection on the bus. These kind of events are not expected in a standard SMBus or I2C-compliant system.
**Note:** In most systems this parameter should not be adjusted, and it is recommended that it be left at its default value.

By default, if the SCL falling edge is detected after the falling edge of SDA (i.e., one SYSCLK cycle or more), the device will detect this as a START condition. The SDD field is used to increase the amount of hold time that is required between SDA and SCL falling before a START is recognized. An additional 2, 4, or 8 SYSCLKs can be added to prevent false START detection in systems where the bus conditions warrant this.

### SMBus Control Register

SMB0CN0 is used to control the interface and to provide status information. The higher four bits of SMB0CN0 (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.
**Note:** The SMBus interface is stalled while SI is set; if SCL is held low at this time, the bus is stalled until software clears SI.

### Hardware ACK Generation

When the EHACK bit in register SMB0ADM is set to 1, automatic slave address recognition and ACK generation is enabled. 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 20.2. Sources for Hardware Changes to SMB0CN0**

| Bit | Set by Hardware When: | Cleared by Hardware When: |
|---|---|---|
| MASTER | A START is generated. | A STOP is generated.<br>Arbitration is lost. |
| TXMODE | START is generated.<br>SMB0DAT is written before the start of an SMBus frame. | A START is detected.<br>Arbitration is lost.<br>SMB0DAT is not written before the start of an SMBus frame. |
| STA | A START followed by an address byte is received. | Must be cleared by software. |
| STO | A STOP is detected while addressed as a slave.<br>Arbitration is lost due to a detected STOP. | A pending STOP is generated. |
| ACKRQ | A byte has been received and an ACK response value is needed (only when hardware ACK is not enabled). | After each ACK cycle. |
| ARBLOST | A repeated START is detected as a MASTER when STA is low (unwanted repeated START).<br>SCL is sensed low while attempting to generate a STOP or repeated START condition.<br>SDA is sensed low while transmitting a 1 (excluding ACK bits). | Each time SIn is cleared. |
| ACK | The incoming ACK value is low (ACKNOWLEDGE). | The incoming ACK value is high (NOT ACKNOWLEDGE). |
| SI | A START has been generated.<br>Lost arbitration.<br>A byte has been transmitted and an ACK/NACK received.<br>A byte has been received.<br>A START or repeated START followed by a slave address + R/W has been received.<br>A STOP has been received. | Must be cleared by software. |

### Hardware Slave Address Recognition

The SMBus hardware has the capability to automatically recognize incoming slave addresses and send an ACK without software intervention. Automatic slave address recognition is enabled by setting the EHACK bit in register SMB0ADM to 1. This will enable both automatic slave address recognition and automatic hardware ACK generation for received bytes (as a master or slave).

The registers used to define which address(es) are recognized by the hardware are the SMBus Slave Address register and the SMBus Slave Address Mask register. A single address or range of addresses (including the General Call Address 0x00) can be specified using these two registers. The most-significant seven bits of the two registers are used to define which addresses will be ACKed. A 1 in a bit of the slave address mask SLVM enables a comparison between the received slave address and the hardware's slave address SLV for that bit. A 0 in a bit of the slave address mask means that bit will be treated as a "don't care" for comparison purposes. In this case, either a 1 or a 0 value are acceptable on the incoming slave address. Additionally, if the GC bit in register SMB0ADR is set to 1, hardware will recognize the General Call Address (0x00).

**Table 20.3. Hardware Address Recognition Examples (EHACK=1)**

| Hardware Slave Address SLV | Slave Address Mask SLVM | GC bit | Slave Addresses Recognized by Hardware |
|---|---|---|---|
| 0x34 | 0x7F | 0 | 0x34 |
| 0x34 | 0x7F | 1 | 0x34, 0x00 (General Call) |
| 0x34 | 0x7E | 0 | 0x34, 0x35 |
| 0x34 | 0x7E | 1 | 0x34, 0x35, 0x00 (General Call) |
| 0x70 | 0x73 | 0 | 0x70, 0x74, 0x78, 0x7C |
| **Note:** These addresses must be shifted to the left by one bit when writing to the SMB0ADR register. | | | |

### Software ACK Generation

In general, it is recommended for applications to use hardware ACK and address recognition. In some cases it may be desirable to drive ACK generation and address recognition from firmware. 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.

### SMBus Data Register

The SMBus Data register SMB0DAT holds a byte of serial data to be transmitted or one that has just been received. Software may safely read or write to the data register when the SI flag is set. Software should not attempt to access the SMB0DAT register when the SMBus is enabled and the SI flag is cleared to logic 0.
**Note:** Certain device families have a transmit and receive buffer interface which is accessed by reading and writing the SMB0DAT register. To promote software portability between devices with and without this buffer interface it is recommended that SMB0DAT not be used as a temporary storage location. On buffer-enabled devices, writing the register multiple times will push multiple bytes into the transmit FIFO.

## 20.3.4 Operational Modes

The SMBus interface may be configured to operate as master and/or slave. At any particular time, it will be operating in one of the following four modes: Master Transmitter, Master Receiver, Slave Transmitter, or Slave Receiver. The SMBus interface enters Master Mode any time a START is generated, and remains in Master Mode until it loses an arbitration or generates a STOP. An SMBus interrupt is generated at the end of all SMBus byte frames. The position of the ACK interrupt when operating as a receiver depends on whether hardware ACK generation is enabled. As a receiver, the interrupt for an ACK occurs before the ACK with hardware ACK generation disabled, and after the ACK when hardware ACK generation is enabled. As a transmitter, interrupts occur after the ACK, regardless of whether hardware ACK generation is enabled or not.

### Master Write Sequence

During a write sequence, an SMBus master writes data to a slave device. The master in this transfer will be a transmitter during the address byte, and a transmitter during all data bytes. The SMBus interface generates the START condition and transmits the first byte containing the address of the target slave and the data direction bit. In this case the data direction bit (R/W) will be logic 0 (WRITE). The master then transmits one or more bytes of serial data. After each byte is transmitted, an acknowledge bit is generated by the slave. The transfer is ended when the STO bit is set and a STOP is generated. The interface will switch to Master Receiver Mode if SMB0DAT is not written following a Master Transmitter interrupt. Figure 20.5 Typical Master Write Sequence on page 221 shows a typical master write sequence as it appears on the bus, and Figure 20.6 Master Write Sequence State Diagram (EHACK = 1) on page 222 shows the corresponding firmware state machine. Two transmit data bytes are shown, though any number of bytes may be transmitted. Notice that all of the “data byte transferred” interrupts occur after the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

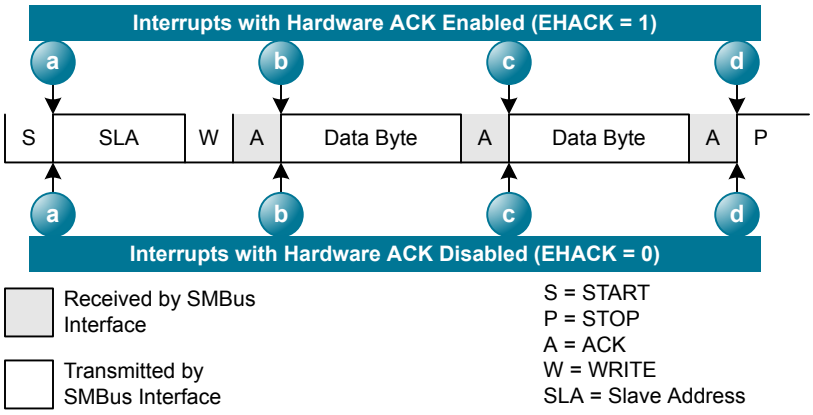



**Figure 20.5. Typical Master Write Sequence**

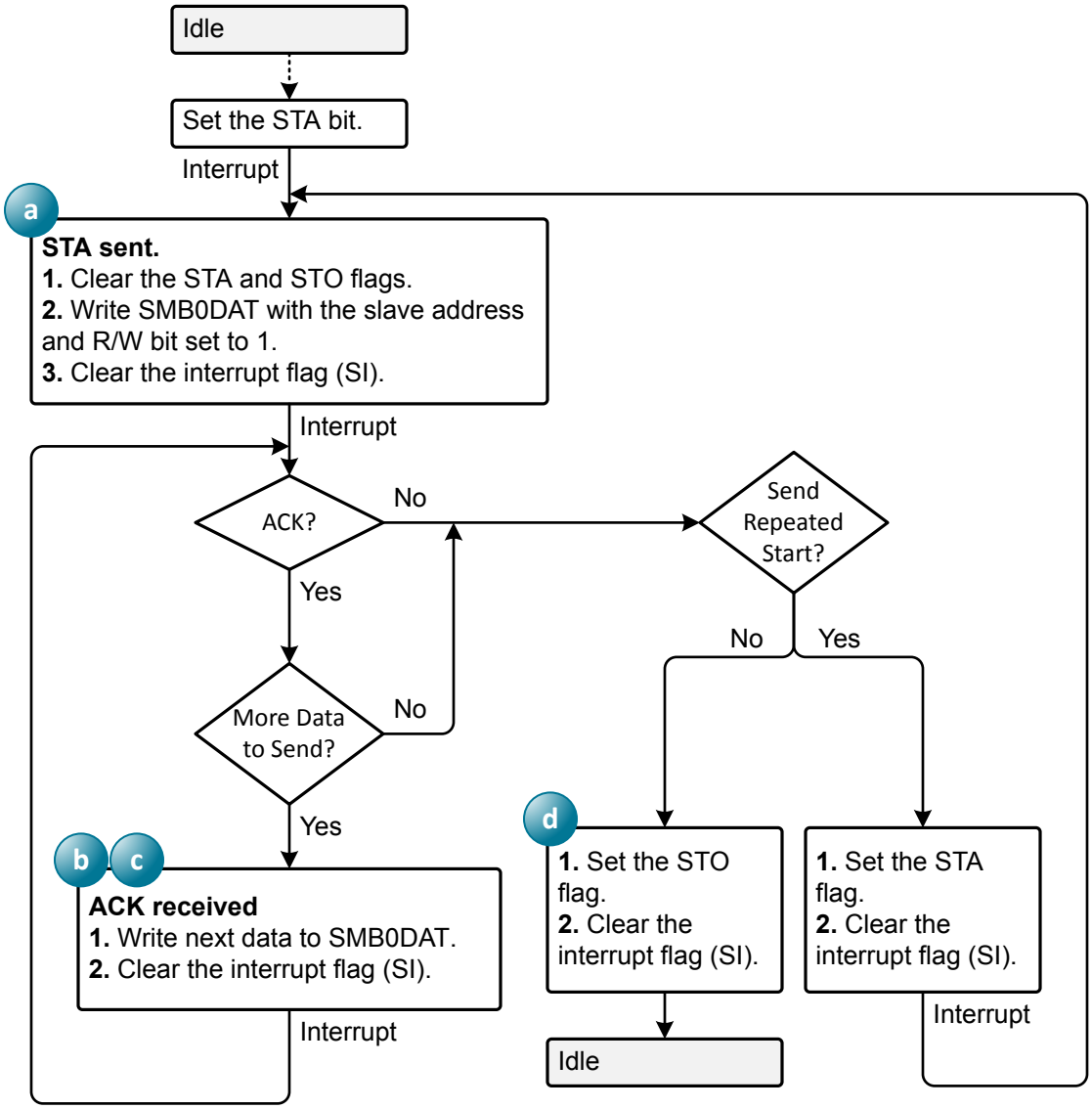



**Figure 20.6. Master Write Sequence State Diagram (EHACK = 1)**

### Master Read Sequence

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

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

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

Writing a 1 to the ACK bit generates an ACK; writing a 0 generates a NACK. Software should write a 0 to the ACK bit for the last data transfer, to transmit a NACK. The interface exits Master Receiver Mode after the STO bit is set and a STOP is generated. The interface will switch to Master Transmitter Mode if SMB0DAT is written while an active Master Receiver. Figure 20.7 Typical Master Read Sequence on page 223 shows a typical master read sequence as it appears on the bus, and Figure 20.8 Master Read Sequence State Diagram (EHACK = 1) on page 224 shows the corresponding firmware state machine. Two received data bytes are shown, though any number of bytes may be received. Notice that the "data byte transferred" interrupts occur at different places in the sequence, depending on whether hardware ACK generation is enabled. The interrupt occurs before the ACK with hardware ACK generation disabled, and after the ACK when hardware ACK generation is enabled.

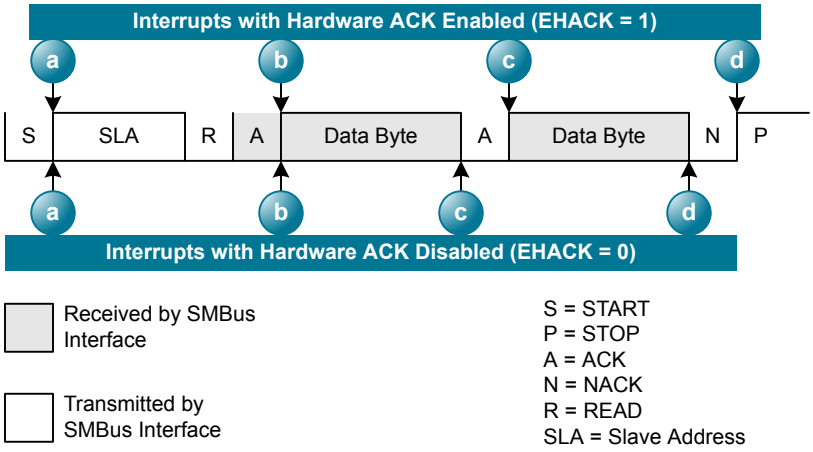



**Figure 20.7. Typical Master Read Sequence**

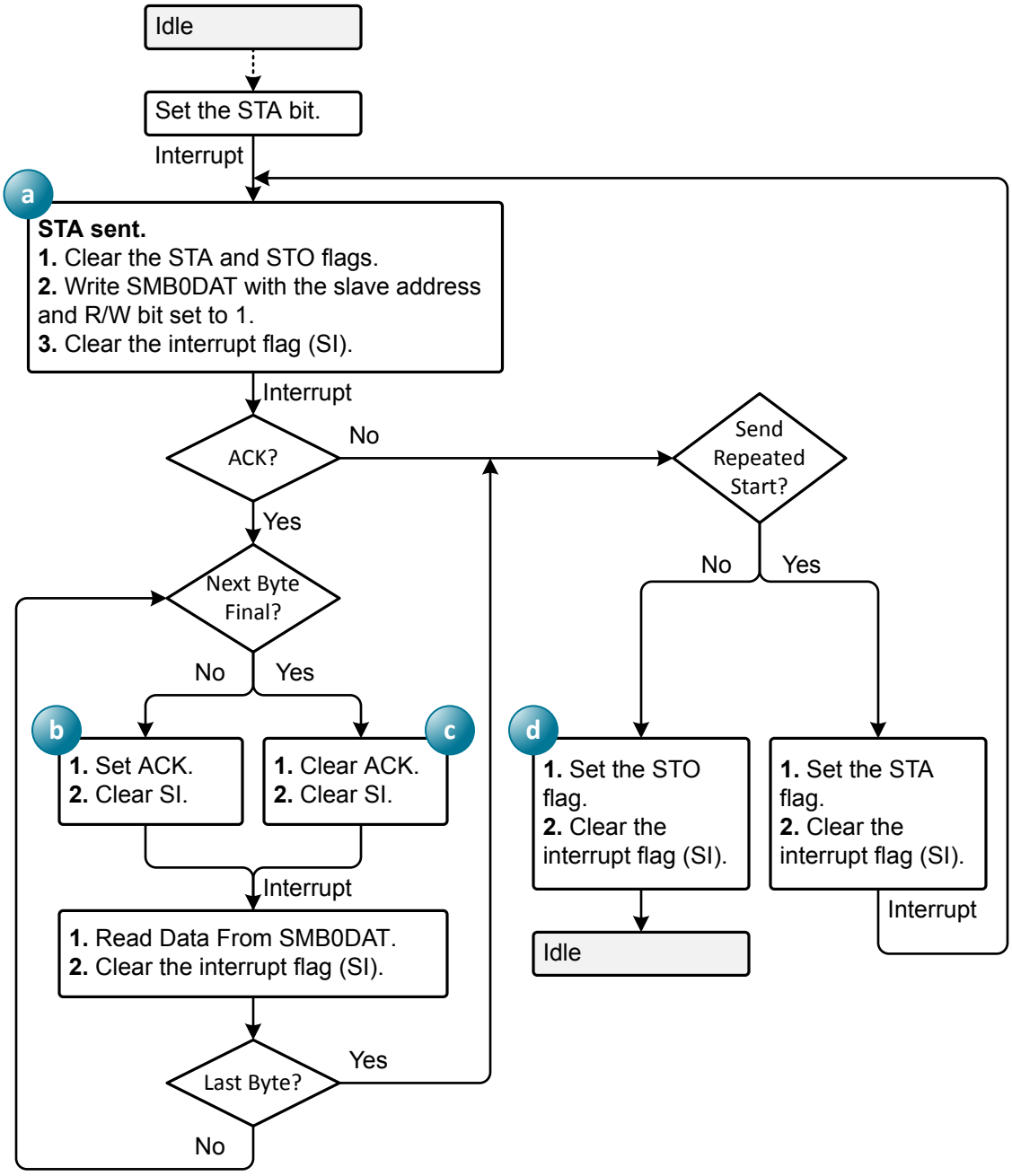



**Figure 20.8. Master Read Sequence State Diagram (EHACK = 1)**

### Slave Write Sequence

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

If the received slave address is ignored (by software or hardware), slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are received.

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

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

The interface exits Slave Receiver Mode after receiving a STOP. The interface will switch to Slave Transmitter Mode if SMB0DAT is written while an active Slave Receiver. Figure 20.9 Typical Slave Write Sequence on page 225 shows a typical slave write sequence as it appears on the bus. The corresponding firmware state diagram (combined with the slave read sequence) is shown in Figure 20.10 Slave State Diagram (EHACK = 1) on page 226. Two received data bytes are shown, though any number of bytes may be received. Notice that the "data byte transferred" interrupts occur at different places in the sequence, depending on whether hardware ACK generation is enabled. The interrupt occurs before the ACK with hardware ACK generation disabled, and after the ACK when hardware ACK generation is enabled.

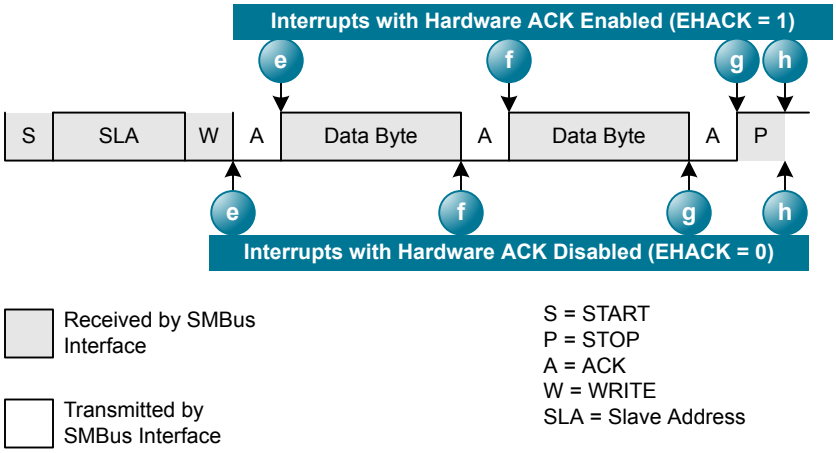



**Figure 20.9. Typical Slave Write Sequence**

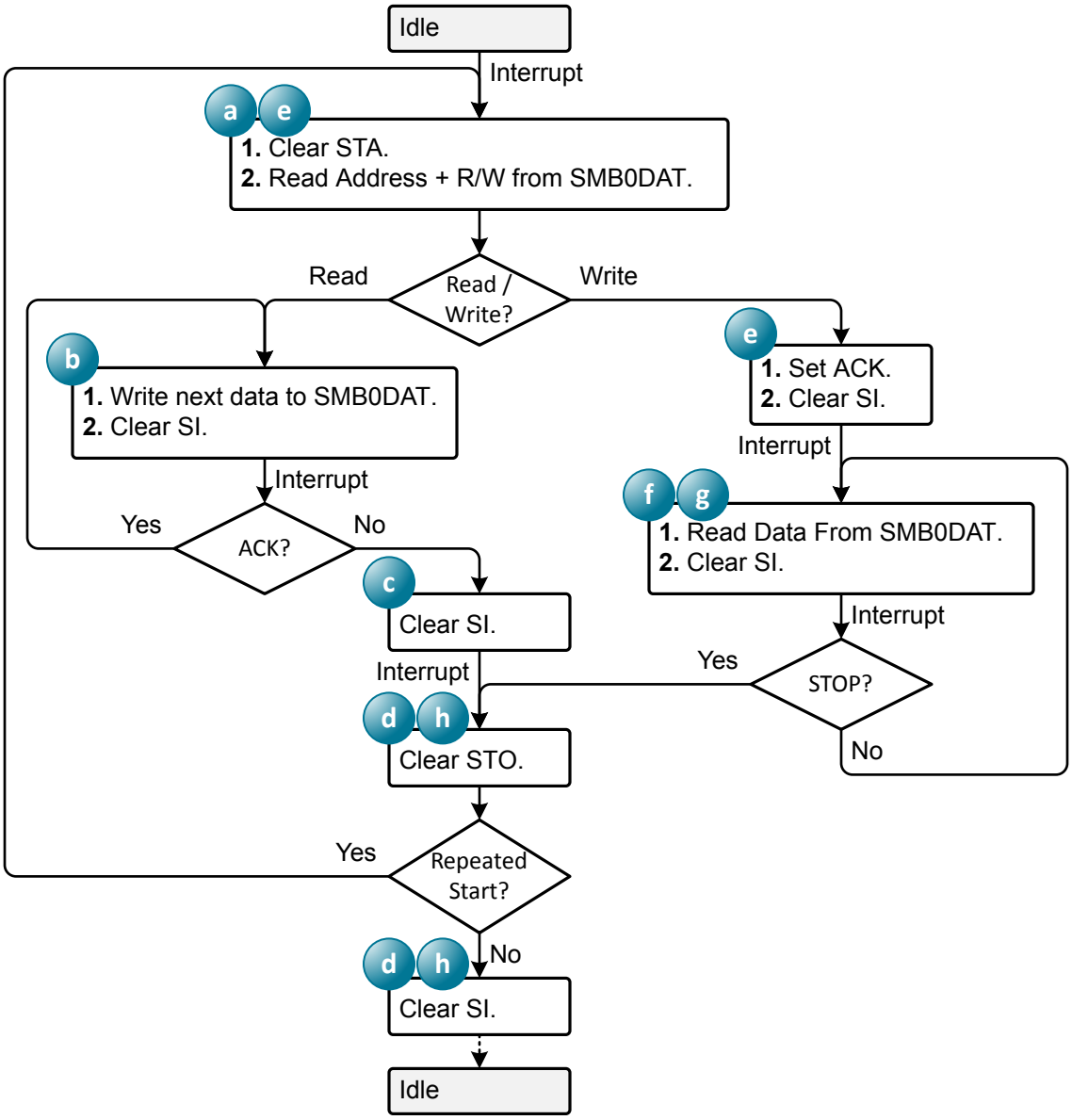



**Figure 20.10. Slave State Diagram (EHACK = 1)**

### Slave Read Sequence

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

If the received slave address is ignored (by software or hardware), slave interrupts will be inhibited until the next START is detected. If the received slave address is acknowledged, zero or more data bytes are transmitted. If the received slave address is acknowledged, data should be written to SMB0DAT to be transmitted. The interface enters slave transmitter mode, and transmits one or more bytes of data. After each byte is transmitted, the master sends an acknowledge bit; if the acknowledge bit is an ACK, SMB0DAT should be written with the next data byte. If the acknowledge bit is a NACK, SMB0DAT should not be written to before SI is cleared (an error condition may be generated if SMB0DAT is written following a received NACK while in slave transmitter mode). The interface exits slave transmitter mode after receiving a STOP. The interface will switch to slave receiver mode if SMB0DAT is not written following a Slave Transmitter interrupt. Figure 20.11 Typical Slave Read Sequence on page 227 shows a typical slave read sequence as it appears on the bus. The corresponding firmware state diagram (combined with the slave read sequence) is shown in Figure 20.10 Slave State Diagram (EHACK = 1) on page 226. Two transmitted data bytes are shown, though any number of bytes may be transmitted. Notice that all of the “data byte transferred” interrupts occur after the ACK cycle in this mode, regardless of whether hardware ACK generation is enabled.

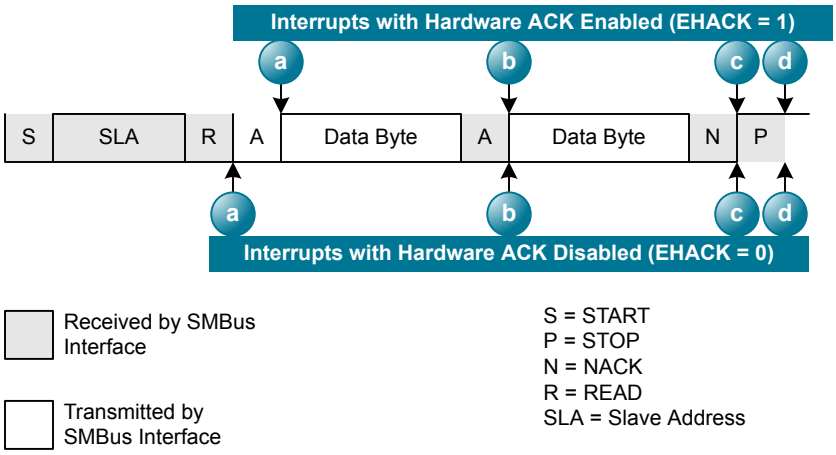



**Figure 20.11. Typical Slave Read Sequence**

## 20.4 SMB0 Control Registers

### 20.4.1 SMB0CF: SMBus 0 Configuration

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ENSMB | INH | BUSY | EXTHOLD | SMBTOE | SMBFTE | SMBCS | |
| Access | RW | RW | R | RW | RW | RW | RW | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0x0 | |
| SFR Page = 0x0; SFR Address: 0xC1 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | ENSMB | 0 | RW | **SMBus Enable.**<br>This bit enables the SMBus interface when set to 1. When enabled, the interface constantly monitors the SDA and SCL pins. |
| 6 | INH | 0 | RW | **SMBus Slave Inhibit.**<br>When this bit is set to logic 1, the SMBus does not generate an interrupt when slave events occur. This effectively removes the SMBus slave from the bus. Master Mode interrupts are not affected. |
| 5 | BUSY | 0 | R | **SMBus Busy Indicator.**<br>This bit is set to logic 1 by hardware when a transfer is in progress. It is cleared to logic 0 when a STOP or free-timeout is sensed. |
| 4 | EXTHOLD | 0 | RW | **SMBus Setup and Hold Time Extension Enable.**<br>This bit controls the SDA setup and hold times. |

| Value | Name | Description |
|---|---|---|
| 0 | DISABLED | Disable SDA extended setup and hold times. |
| 1 | ENABLED | Enable SDA extended setup and hold times. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 3 | SMBTOE | 0 | RW | **SMBus SCL Timeout Detection Enable.**<br>This bit enables SCL low timeout detection. If set to logic 1, the SMBus forces Timer 3 to reload while SCL is high and allows Timer 3 to count when SCL goes low. If Timer 3 is configured to Split Mode, only the High Byte of the timer is held in reload while SCL is high. Timer 3 should be programmed to generate interrupts at 25 ms, and the Timer 3 interrupt service routine should reset SMBus communication. |
| 2 | SMBFTE | 0 | RW | **SMBus Free Timeout Detection Enable.**<br>When this bit is set to logic 1, the bus will be considered free if SCL and SDA remain high for more than 10 SMBus clock source periods. |
| 1:0 | SMBCS | 0x0 | RW | **SMBus Clock Source Selection.**<br>This field selects the SMBus clock source, which is used to generate the SMBus bit rate. See the SMBus clock timing section for additional details. |

| Value | Name | Description |
|---|---|---|
| 0x0 | TIMER0 | Timer 0 Overflow. |
| 0x1 | TIMER1 | Timer 1 Overflow. |
| 0x2 | TIMER2_HIGH | Timer 2 High Byte Overflow. |
| 0x3 | TIMER2_LOW | Timer 2 Low Byte Overflow. |

### 20.4.2 SMB0CN0: SMBus 0 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MASTER | TXMODE | STA | STO | ACKRQ | ARBLOST | ACK | SI |
| Access | R | R | RW | RW | R | R | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = 0x0; SFR Address: 0xC0 (bit-addressable)

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | MASTER | 0 | R | **SMBus Master/Slave Indicator.**<br>This read-only bit indicates when the SMBus is operating as a master. |

| Value | Name | Description |
|---|---|---|
| 0 | SLAVE | SMBus operating in slave mode. |
| 1 | MASTER | SMBus operating in master mode. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | TXMODE | 0 | R | **SMBus Transmit Mode Indicator.**<br>This read-only bit indicates when the SMBus is operating as a transmitter. |

| Value | Name | Description |
|---|---|---|
| 0 | RECEIVER | SMBus in Receiver Mode. |
| 1 | TRANSMITTER | SMBus in Transmitter Mode. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5 | STA | 0 | RW | **SMBus Start Flag.**<br>When reading STA, a '1' indicates that a start or repeated start condition was detected on the bus.<br>Writing a '1' to the STA bit initiates a start or repeated start on the bus. |
| 4 | STO | 0 | RW | **SMBus Stop Flag.**<br>When reading STO, a '1' indicates that a stop condition was detected on the bus (in slave mode) or is pending (in master mode).<br>When acting as a master, writing a '1' to the STO bit initiates a stop condition on the bus. This bit is cleared by hardware. |
| 3 | ACKRQ | 0 | R | **SMBus Acknowledge Request.** |

| Value | Name | Description |
|---|---|---|
| 0 | NOT_SET | No ACK requested. |
| 1 | REQUESTED | ACK requested. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2 | ARBLOST | 0 | R | **SMBus Arbitration Lost Indicator.** |

| Value | Name | Description |
|---|---|---|
| 0 | NOT_SET | No arbitration error. |
| 1 | ERROR | Arbitration error occurred. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 1 | ACK | 0 | RW | **SMBus Acknowledge.**<br>When read as a master, the ACK bit indicates whether an ACK (1) or NACK (0) is received during the most recent byte transfer.<br>As a slave, this bit should be written to send an ACK (1) or NACK (0) to a master request. Note that the logic level of the ACK bit on the SMBus interface is inverted from the logic of the register ACK bit. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 0 | SI | 0 | RW | **SMBus Interrupt Flag.**<br>This bit is set by hardware to indicate that the current SMBus state machine operation (such as writing a data or address byte) is complete, and the hardware needs additional control from the firmware to proceed. While SI is set, SCL is held low and SMBus is stalled. SI must be cleared by firmware. Clearing SI initiates the next SMBus state machine operation. |

### 20.4.3 SMB0ADR: SMBus 0 Slave Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SLV | | | | | | | GC |
| Access | RW | | | | | | | RW |
| Reset | 0x00 | | | | | | | 0 |
| SFR Page = 0x0; SFR Address: 0xF4 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:1 | SLV | 0x00 | RW | **SMBus Hardware Slave Address.**<br>Defines the SMBus Slave Address(es) for automatic hardware acknowledgement. Only address bits which have a 1 in the corresponding bit position in SLVM are checked against the incoming address. This allows multiple addresses to be recognized. |
| 0 | GC | 0 | RW | **General Call Address Enable.**<br>When hardware address recognition is enabled (EHACK = 1), this bit will determine whether the General Call Address (0x00) is also recognized by hardware. |

| Value | Name | Description |
|---|---|---|
| 0 | IGNORED | General Call Address is ignored. |
| 1 | RECOGNIZED | General Call Address is recognized. |

### 20.4.4 SMB0ADM: SMBus 0 Slave Address Mask

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SLVM | | | | | | | EHACK |
| Access | RW | | | | | | | RW |
| Reset | 0x7F | | | | | | | 0 |
| SFR Page = 0x0; SFR Address: 0xF5 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:1 | SLVM | 0x7F | RW | **SMBus Slave Address Mask.**<br>Defines which bits of register SMB0ADR are compared with an incoming address byte, and which bits are ignored. Any bit set to 1 in SLVM enables comparisons with the corresponding bit in SLV. Bits set to 0 are ignored (can be either 0 or 1 in the incoming address). |
| 0 | EHACK | 0 | RW | **Hardware Acknowledge Enable.**<br>Enables hardware acknowledgement of slave address and received data bytes. |

| Value | Name | Description |
|---|---|---|
| 0 | ADR_ACK_MANUAL | Firmware must manually acknowledge all incoming address and data bytes. |
| 1 | ADR_ACK_AUTOMAT-IC | Automatic slave address recognition and hardware acknowledge is enabled. |

### 20.4.5 SMB0DAT: SMBus 0 Data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SMB0DAT | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xC2 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | SMB0DAT | 0x00 | RW | **SMBus 0 Data.**<br>The SMB0DAT register contains a byte of data to be transmitted on the SMBus serial interface or a byte that has just been received on the SMBus serial interface. The CPU can safely read from or write to this register whenever the SI serial interrupt flag is set to logic 1. The serial data in the register remains stable as long as the SI flag is set. When the SI flag is not set, the system may be in the process of shifting data in/out and the CPU should not attempt to access this register. |

# 21. Timers (Timer0, Timer1, Timer2, and Timer3)

## 21.1 Introduction

Four counter/timers ar included in the device: two are 16-bit counter/timers compatible with those found in the standard 8051, and two are 16-bit auto-reload timers for timing peripherals or for general purpose use. These timers can be used to measure time intervals, count external events and generate periodic interrupt requests. Timer 0 and Timer 1 are nearly identical and have four primary modes of operation. Timer 2 and Timer 3 are also identical and offer both 16-bit and split 8-bit timer functionality with auto-reload capabilities. Timer 2 and Timer 3 both offer a capture function, but are different in their system-level connections. Timer 2 is capable of performing a capture function on the RTC clock output or Comparator 0 output, while Timer 3 is capable of performing a capture function on the RTC clock output or external oscillator divided by 8.

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

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

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

**Table 21.1. Timer Modes**

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

## 21.2 Features

Timer 0 and Timer 1 include the following features:

- Standard 8051 timers, supporting backwards-compatibility with firmware and hardware.
- Clock sources include SYSCLK, SYSCLK divided by 12, 4, or 48, the External Clock divided by 8, or an external pin.
- 8-bit auto-reload counter/timer mode
- 13-bit counter/timer mode
- 16-bit counter/timer mode
- Dual 8-bit counter/timer mode (Timer 0)

Timer 2 and Timer 3 are 16-bit timers including the following features:

- Clock sources include SYSCLK, SYSCLK divided by 12, or the External Clock divided by 8.
- 16-bit auto-reload timer mode
- Dual 8-bit auto-reload timer mode
- Comparator 0 or RTC0 capture (Timer 2)
- RTC0 or EXTCLK/8 capture (Timer 3)

## 21.3 Functional Description

### 21.3.1 System Connections

All four timers are capable of clocking other peripherals and triggering events in the system. The individual peripherals select which timer to use for their respective functions. Note that the Timer 2 and Timer 3 high overflows apply to the full timer when operating in 16-bit mode or the high-byte timer when operating in 8-bit split mode.

**Table 21.2. Timer Peripheral Clocking / Event Triggering**

| Function | T0 Overflow | T1 Overflow | T2 High Overflow | T2 Low Overflow | T3 High Overflow | T3 Low Overflow |
|---|---|---|---|---|---|---|
| UART0 Baud Rate | | Yes | | | | |
| SMBus 0 Clock Rate (Master) | Yes | Yes | Yes | Yes | | |
| SMBus 0 SCL Low Timeout | | | | | Yes | |
| PCA0 Clock | Yes | | | | | |
| ADC0 Conversion Start | Yes | | Yes[1] | Yes[1] | Yes[1] | Yes[1] |

Notes:
1. The high-side overflow is used when the timer is in 16-bit mode. The low-side overflow is used in 8-bit mode.

### 21.3.2 Timer 0 and Timer 1

Timer 0 and Timer 1 are each implemented as a 16-bit register accessed as two separate bytes: a low byte (TL0 or TL1) and a high byte (TH0 or TH1). The Counter/Timer Control register (TCON) is used to enable Timer 0 and Timer 1 as well as indicate status. Timer 0 interrupts can be enabled by setting the ET0 bit in the IE register. Timer 1 interrupts can be enabled by setting the ET1 bit in the IE register. Both counter/timers operate in one of four primary modes selected by setting the Mode Select bits T1M1–T0M0 in the Counter/Timer Mode register (TMOD). Each timer can be configured independently for the supported operating modes.

#### 21.3.2.1 Operational Modes

**Mode 0: 13-bit Counter/Timer**

Timer 0 and Timer 1 operate as 13-bit counter/timers in Mode 0. The following describes the configuration and operation of Timer 0. However, both timers operate identically, and Timer 1 is configured in the same manner as described for Timer 0.

The TH0 register holds the eight MSBs of the 13-bit counter/timer. TL0 holds the five LSBs in bit positions TL0.4–TL0.0. The three upper bits of TL0 (TL0.7–TL0.5) are indeterminate and should be masked out or ignored when reading. As the 13-bit timer register increments and overflows from 0x1FFF (all ones) to 0x0000, the timer overflow flag TF0 in TCON is set and an interrupt occurs if Timer 0 interrupts are enabled. The overflow rate for Timer 0 in 13-bit mode is:

$$F_{TIMER0} = \frac{F_{Input\ Clock}}{2^{13} - TH0{:}TL0} = \frac{F_{Input\ Clock}}{8192 - TH0{:}TL0}$$

The CT0 bit in the TMOD register selects the counter/timer's clock source. When CT0 is set to logic 1, high-to-low transitions at the selected Timer 0 input pin (T0) increment the timer register. Events with a frequency of up to one-fourth the system clock frequency can be counted. The input signal need not be periodic, but it must be held at a given level for at least two full system clock cycles to ensure the level is properly sampled. Clearing CT selects the clock defined by the T0M bit in register CKCON0. When T0M is set, Timer 0 is clocked by the system clock. When T0M is cleared, Timer 0 is clocked by the source selected by the Clock Scale bits in CKCON0.

Setting the TR0 bit enables the timer when either GATE0 in the TMOD register is logic 0 or based on the input signal INT0. The IN0PL bit setting in IT01CF changes which state of INT0 input starts the timer counting. Setting GATE0 to 1 allows the timer to be controlled by the external input signal INT0, facilitating pulse width measurements.

**Table 21.3. Timer 0 Run Control Options**

| TR0 | GATE0 | INT0 | IN0PL | Counter/Timer |
|---|---|---|---|---|
| 0 | X | X | X | Disabled |
| 1 | 0 | X | X | Enabled |
| 1 | 1 | 0 | 0 | Disabled |
| 1 | 1 | 0 | 1 | Enabled |
| 1 | 1 | 1 | 0 | Enabled |
| 1 | 1 | 1 | 1 | Disabled |

**Note:**
1. X = Don't Care

Setting TR0 does not force the timer to reset. The timer registers should be loaded with the desired initial value before the timer is enabled.

TL1 and TH1 form the 13-bit register for Timer 1 in the same manner as described above for TL0 and TH0. Timer 1 is configured and controlled using the relevant TCON and TMOD bits just as with Timer 0. The input signal INT1 is used with Timer 1, and IN1PL in register IT01CF determines the INT1 state that starts Timer 1 counting.

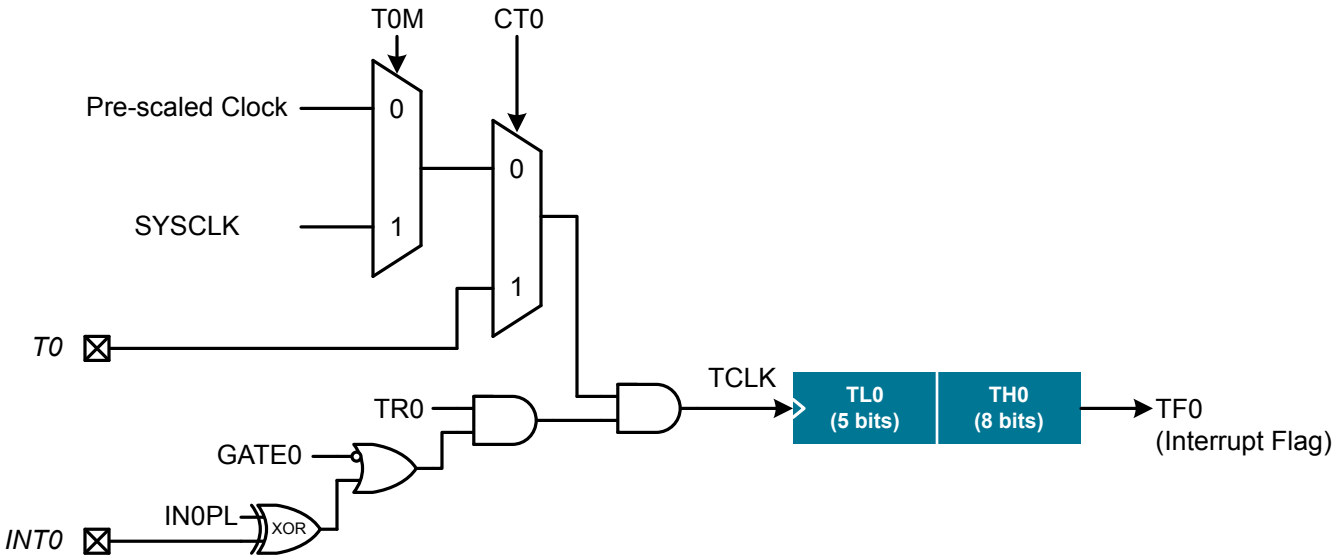



**Figure 21.1. T0 Mode 0 Block Diagram**

**Mode 1: 16-bit Counter/Timer**

Mode 1 operation is the same as Mode 0, except that the counter/timer registers use all 16 bits. The counter/timers are enabled and configured in Mode 1 in the same manner as for Mode 0. The overflow rate for Timer 0 in 16-bit mode is:

$$F_{TIMER0} = \frac{F_{Input\ Clock}}{2^{16} - TH0:TL0} = \frac{F_{Input\ Clock}}{65536 - TH0:TL0}$$

### Mode 2: 8-bit Counter/Timer with Auto-Reload

Mode 2 configures Timer 0 and Timer 1 to operate as 8-bit counter/timers with automatic reload of the start value. TL0 holds the count and TH0 holds the reload value. When the counter in TL0 overflows from all ones to 0x00, the timer overflow flag TF0 in the TCON register is set and the counter in TL0 is reloaded from TH0. If Timer 0 interrupts are enabled, an interrupt will occur when the TF0 flag is set. The reload value in TH0 is not changed. TL0 must be initialized to the desired value before enabling the timer for the first count to be correct. When in Mode 2, Timer 1 operates identically to Timer 0.

The overflow rate for Timer 0 in 8-bit auto-reload mode is:

$$F_{TIMER0} = \frac{F_{Input\ Clock}}{2^8 - TH0} = \frac{F_{Input\ Clock}}{256 - TH0}$$

Both counter/timers are enabled and configured in Mode 2 in the same manner as Mode 0. Setting the TR0 bit enables the timer when either GATE0 in the TMOD register is logic 0 or when the input signal INT0 is active as defined by bit IN0PL in register IT01CF.

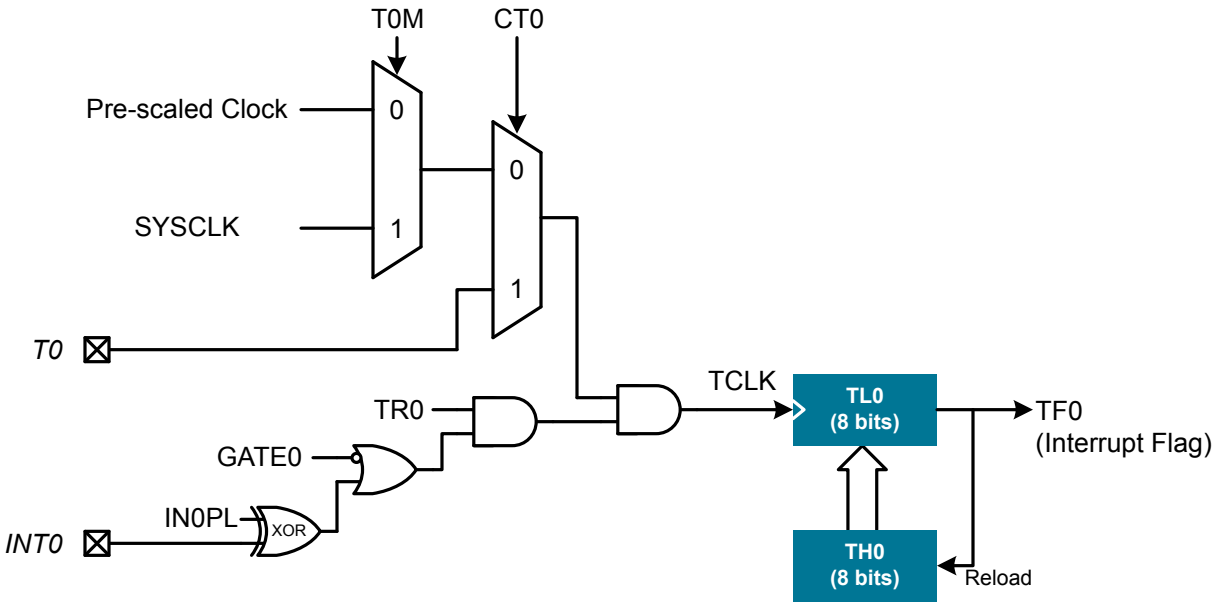



**Figure 21.2. T0 Mode 2 Block Diagram**

**Mode 3: Two 8-bit Counter/Timers (Timer 0 Only)**

In Mode 3, Timer 0 is configured as two separate 8-bit counter/timers held in TL0 and TH0. The counter/timer in TL0 is controlled using the Timer 0 control/status bits in TCON and TMOD: TR0, CT0, GATE0, and TF0. TL0 can use either the system clock or an external input signal as its timebase. The TH0 register is restricted to a timer function sourced by the system clock or prescaled clock. TH0 is enabled using the Timer 1 run control bit TR1. TH0 sets the Timer 1 overflow flag TF1 on overflow and thus controls the Timer 1 interrupt.

The overflow rate for Timer 0 Low in 8-bit mode is:

$$F_{TIMER0} = \frac{F_{Input\ Clock}}{2^8 - TL0} = \frac{F_{Input\ Clock}}{256 - TL0}$$

The overflow rate for Timer 0 High in 8-bit mode is:

$$F_{TIMER0} = \frac{F_{Input\ Clock}}{2^8 - TH0} = \frac{F_{Input\ Clock}}{256 - TH0}$$

Timer 1 is inactive in Mode 3. When Timer 0 is operating in Mode 3, Timer 1 can be operated in Modes 0, 1 or 2, but cannot be clocked by external signals nor set the TF1 flag and generate an interrupt. However, the Timer 1 overflow can be used to generate baud rates for the SMBus and/or UART, and/or initiate ADC conversions. While Timer 0 is operating in Mode 3, Timer 1 run control is handled through its mode settings. To run Timer 1 while Timer 0 is in Mode 3, set the Timer 1 Mode as 0, 1, or 2. To disable Timer 1, configure it for Mode 3.

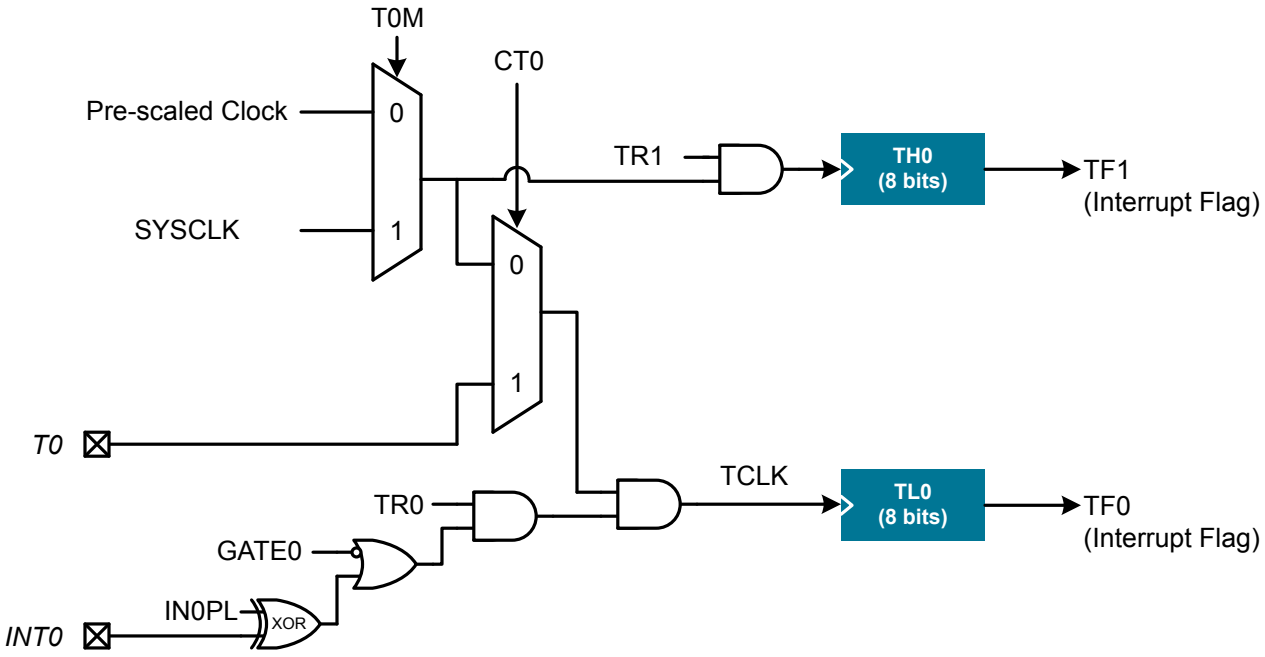



**Figure 21.3. T0 Mode 3 Block Diagram**

### 21.3.3 Timer 2 and Timer 3

Timer 2 and Timer 3 are functionally equivalent, with the only differences being the top-level connections to other parts of the system.

The timers are 16 bits wide, formed by two 8-bit SFRs: TMRnL (low byte) and TMRnH (high byte). Each timer may operate in 16-bit auto-reload mode, dual 8-bit auto-reload (split) mode, or capture mode.

### Clock Selection

Clocking for each timer is configured using the TnXCLK bit field and the TnML and TnMH bits. Timer 2 may be clocked by the system clock, system clock divided by 12, Comparator 0 output, or RTC oscillator divided by 8. Timer 3 may be clocked by the system clock, the system clock divided by 12, the external oscillator clock source divided by 8 (synchronized with SYSCLK), or the Comparator 1 output.

$$F_{SYSCLK} > F_{EXTOSC} \times \frac{6}{7}$$

When operating in one of the 16-bit modes, the low-side timer clock is used to clock the entire 16-bit timer.

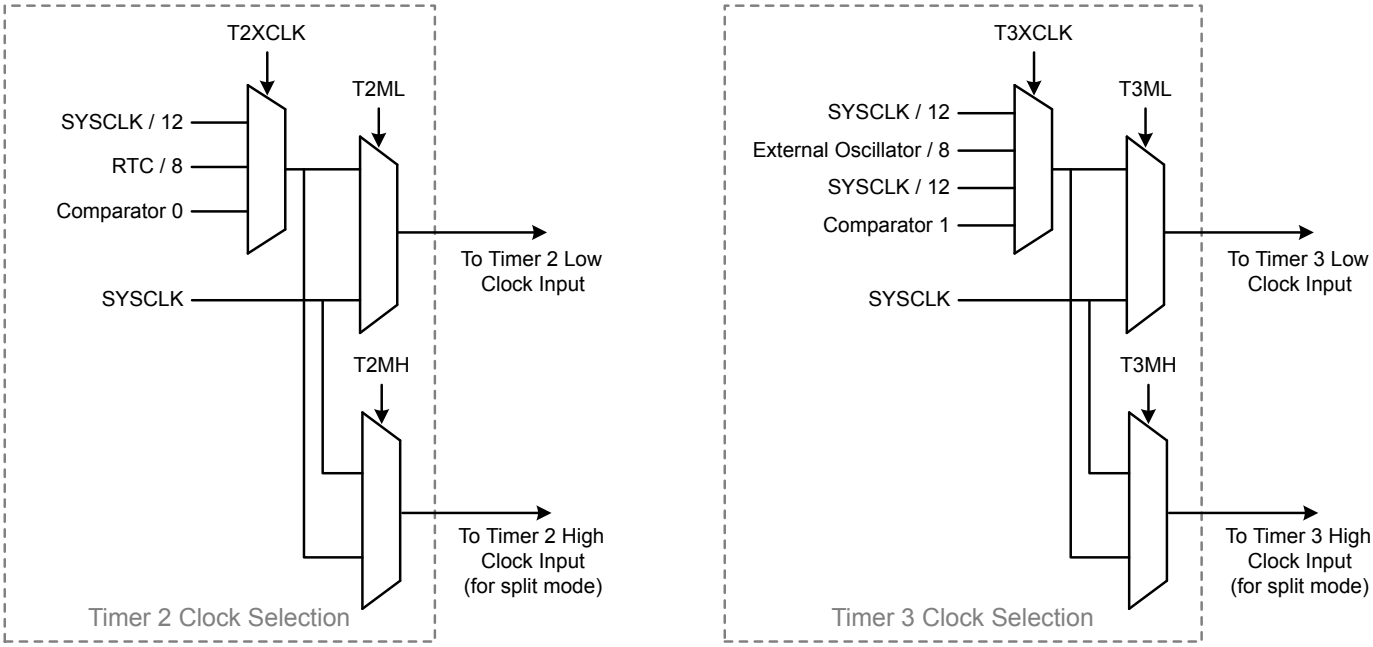



**Figure 21.4. Timer 2 and 3 Clock Source Selection**

### Capture Sources

Capture mode allows an input to be measured against the selected clock source. Timer 2 is capable of performing a capture function on the RTC clock output divided by 8 or Comparator 0 output, while Timer 3 is capable of performing a capture function on the Comparator 1 output or external oscillator divided by 8.

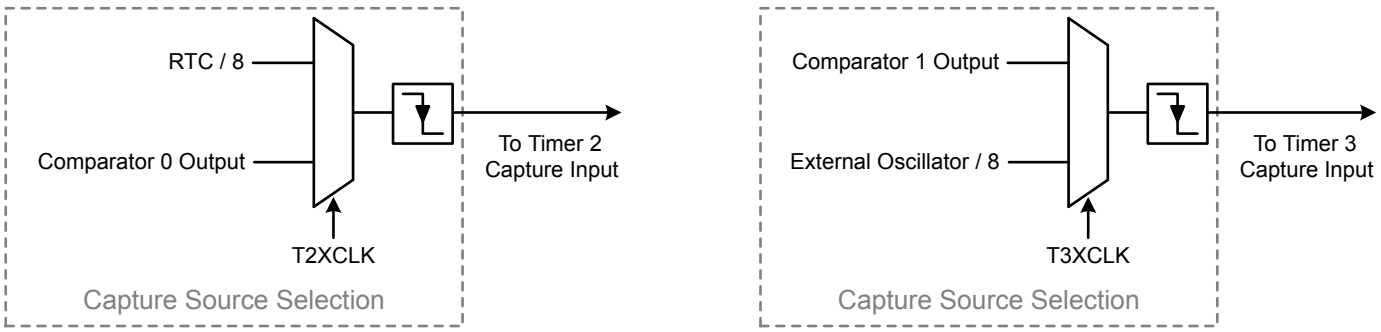



**Figure 21.5. Timer 2 and 3 Capture Sources**

#### 21.3.3.1 16-bit Timer with Auto-Reload

When TnSPLIT is zero, the timer operates as a 16-bit timer with auto-reload. In this mode, the selected clock source increments the timer on every clock. As the 16-bit timer register increments and overflows from 0xFFFF to 0x0000, the 16-bit value in the timer reload registers (TMRnRLH and TMRnRLL) is loaded into the main timer count register, and the High Byte Overflow Flag (TFnH) is set. If the timer interrupts are enabled, an interrupt is generated on each timer overflow. Additionally, if the timer interrupts are enabled and the TFnLEN bit is set, an interrupt is generated each time the lower 8 bits (TMRnL) overflow from 0xFF to 0x00.

The overflow rate of the timer in split 16-bit auto-reload mode is:

$$F_{TIMERn} = \frac{F_{Input\ Clock}}{2^{16} - TMRnRLH{:}TMRnRLL} = \frac{F_{Input\ Clock}}{65536 - TMRnRLH{:}TMRnRLL}$$

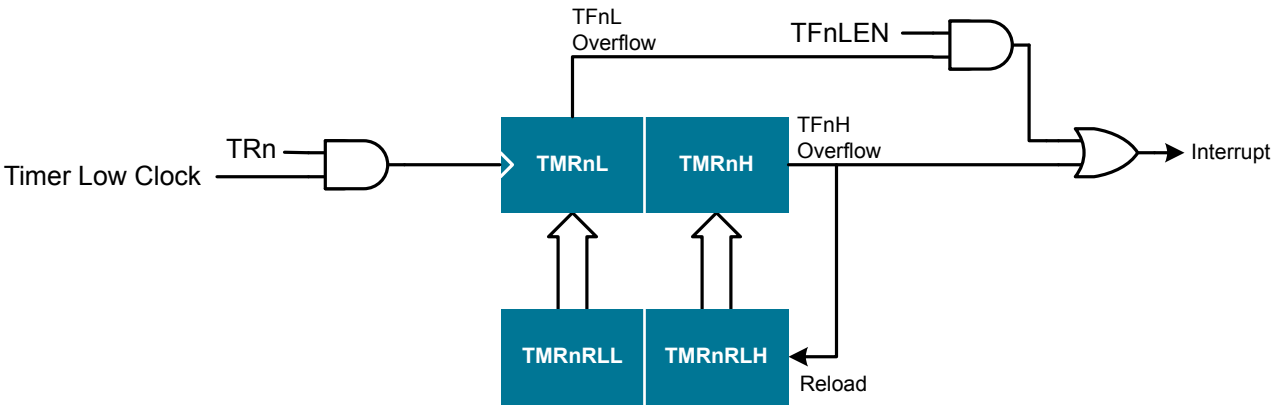



**Figure 21.6. 16-Bit Mode Block Diagram**

#### 21.3.3.2 8-bit Timers with Auto-Reload (Split Mode)

When TnSPLIT is set, the timer operates as two 8-bit timers (TMRnH and TMRnL). Both 8-bit timers operate in auto-reload mode. TMRnRLL holds the reload value for TMRnL; TMRnRLH holds the reload value for TMRnH. The TRn bit in TMRnCN handles the run control for TMRnH. TMRnL is always running when configured for 8-bit auto-reload mode. As shown in the clock source selection tree, the two halves of the timer may be clocked from SYSCLK or by the source selected by the TnXCLK bits.

The overflow rate of the low timer in split 8-bit auto-reload mode is:

$$F_{\text{TIMERn Low}} = \frac{F_{\text{Input Clock}}}{2^8 - \text{TMRnRLL}} = \frac{F_{\text{Input Clock}}}{256 - \text{TMRnRLL}}$$

The overflow rate of the high timer in split 8-bit auto-reload mode is:

$$F_{\text{TIMERn High}} = \frac{F_{\text{Input Clock}}}{2^8 - \text{TMRnRLH}} = \frac{F_{\text{Input Clock}}}{256 - \text{TMRnRLH}}$$

The TFnH bit is set when TMRnH overflows from 0xFF to 0x00; the TFnL bit is set when TMRnL overflows from 0xFF to 0x00. When timer interrupts are enabled, an interrupt is generated each time TMRnH overflows. If timer interrupts are enabled and TFnLEN is set, an interrupt is generated each time either TMRnL or TMRnH overflows. When TFnLEN is enabled, software must check the TFnH and TFnL flags to determine the source of the timer interrupt. The TFnH and TFnL interrupt flags are not cleared by hardware and must be manually cleared by software.

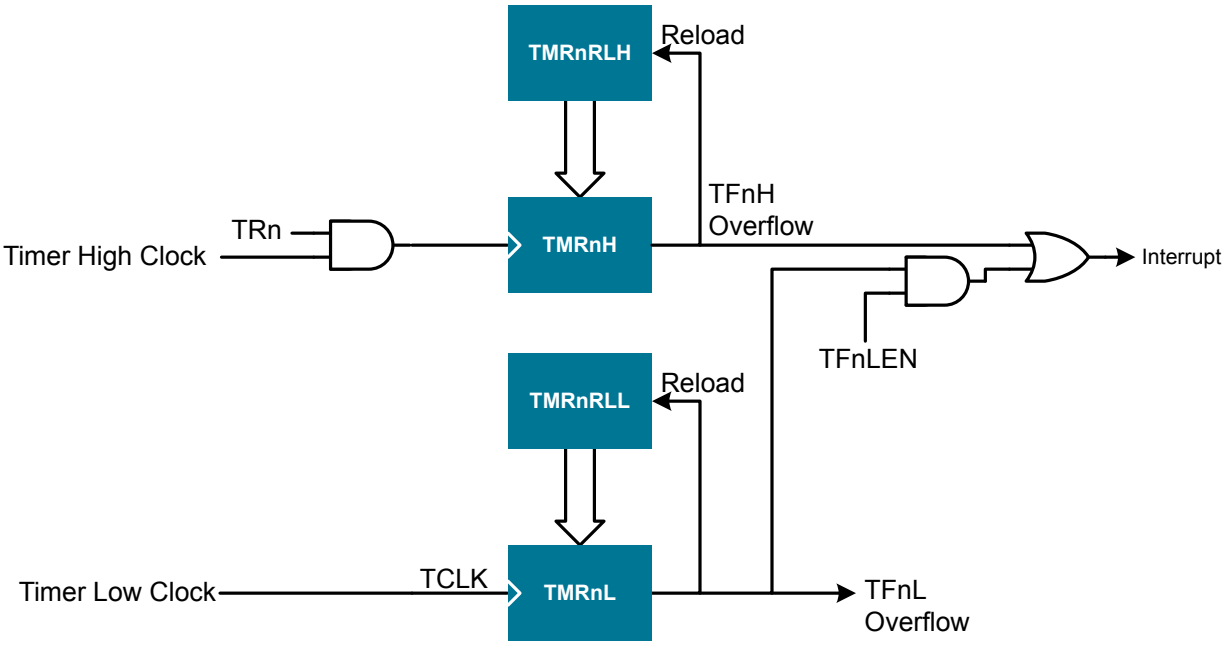



**Figure 21.7. 8-Bit Split Mode Block Diagram**

#### 21.3.3.3 Capture Mode

Capture mode allows a system event to be measured against the selected clock source. When used in capture mode, the timer clocks normally from the selected clock source through the entire range of 16-bit values from 0x0000 to 0xFFFF.

Setting TFnCEN to 1 enables capture mode. In this mode, TnSPLIT should be set to 0, as the full 16-bit timer is used. Upon a falling edge of the input capture signal, the contents of the timer register (TMRnH:TMRnL) are loaded into the reload registers (TMRnRLH:TMRnRLL) and the TFnH flag is set. By recording the difference between two successive timer capture values, the period of the captured signal can be determined with respect to the selected timer clock.

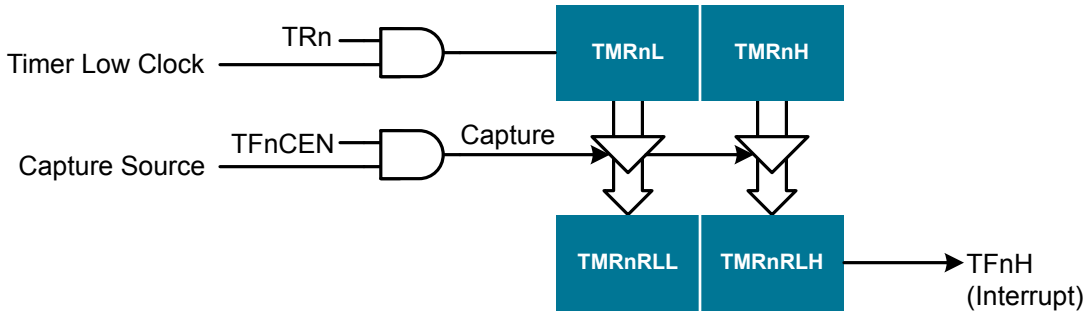



**Figure 21.8. Capture Mode Block Diagram**

## 21.4 Timer 0, 1, 2 and 3, Control Registers

### 21.4.1 CKCON0: Clock Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T3MH | T3ML | T2MH | T2ML | T1M | T0M | SCA | |
| Access | RW | RW | RW | RW | RW | RW | RW | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0x0 | |
| SFR Page = 0x0; SFR Address: 0x8E | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | T3MH | 0 | RW | **Timer 3 High Byte Clock Select.**<br>Selects the clock supplied to the Timer 3 high byte (split 8-bit timer mode only). |

| Value | Name | Description |
|---|---|---|
| 0 | EXTERNAL_CLOCK | Timer 3 high byte uses the clock defined by T3XCLK in TMR3CN0. |
| 1 | SYSCLK | Timer 3 high byte uses the system clock. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 6 | T3ML | 0 | RW | **Timer 3 Low Byte Clock Select.**<br>Selects the clock supplied to Timer 3. Selects the clock supplied to the lower 8-bit timer in split 8-bit timer mode. |

| Value | Name | Description |
|---|---|---|
| 0 | EXTERNAL_CLOCK | Timer 3 low byte uses the clock defined by T3XCLK in TMR3CN0. |
| 1 | SYSCLK | Timer 3 low byte uses the system clock. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 5 | T2MH | 0 | RW | **Timer 2 High Byte Clock Select.**<br>Selects the clock supplied to the Timer 2 high byte (split 8-bit timer mode only). |

| Value | Name | Description |
|---|---|---|
| 0 | EXTERNAL_CLOCK | Timer 2 high byte uses the clock defined by T2XCLK in TMR2CN0. |
| 1 | SYSCLK | Timer 2 high byte uses the system clock. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 4 | T2ML | 0 | RW | **Timer 2 Low Byte Clock Select.**<br>Selects the clock supplied to Timer 2. If Timer 2 is configured in split 8-bit timer mode, this bit selects the clock supplied to the lower 8-bit timer. |

| Value | Name | Description |
|---|---|---|
| 0 | EXTERNAL_CLOCK | Timer 2 low byte uses the clock defined by T2XCLK in TMR2CN0. |
| 1 | SYSCLK | Timer 2 low byte uses the system clock. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 3 | T1M | 0 | RW | **Timer 1 Clock Select.**<br>Selects the clock source supplied to Timer 1. Ignored when C/T1 is set to 1. |

| Value | Name | Description |
|---|---|---|
| 0 | PRESCALE | Timer 1 uses the clock defined by the prescale field, SCA. |
| 1 | SYSCLK | Timer 1 uses the system clock. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2 | T0M | 0 | RW | **Timer 0 Clock Select.** |
| | Selects the clock source supplied to Timer 0. Ignored when C/T0 is set to 1. | | | |

| Value | Name | Description |
|---|---|---|
| 0 | PRESCALE | Counter/Timer 0 uses the clock defined by the prescale field, SCA. |
| 1 | SYSCLK | Counter/Timer 0 uses the system clock. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 1:0 | SCA | 0x0 | RW | **Timer 0/1 Prescale.** |
| | These bits control the Timer 0/1 Clock Prescaler: | | | |

| Value | Name | Description |
|---|---|---|
| 0x0 | SYSCLK_DIV_12 | System clock divided by 12. |
| 0x1 | SYSCLK_DIV_4 | System clock divided by 4. |
| 0x2 | SYSCLK_DIV_48 | System clock divided by 48. |
| 0x3 | EXTOSC_DIV_8 | External oscillator divided by 8 (synchronized with the system clock). |

### 21.4.2 TCON: Timer 0/1 Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |
| Access | RW | RW | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| SFR Page = 0x0; SFR Address: 0x88 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | TF1 | 0 | RW | **Timer 1 Overflow Flag.**<br>Set to 1 by hardware when Timer 1 overflows. This flag can be cleared by firmware but is automatically cleared when the CPU vectors to the Timer 1 interrupt service routine. |
| 6 | TR1 | 0 | RW | **Timer 1 Run Control.**<br>Timer 1 is enabled by setting this bit to 1. |
| 5 | TF0 | 0 | RW | **Timer 0 Overflow Flag.**<br>Set to 1 by hardware when Timer 0 overflows. This flag can be cleared by firmware but is automatically cleared when the CPU vectors to the Timer 0 interrupt service routine. |
| 4 | TR0 | 0 | RW | **Timer 0 Run Control.**<br>Timer 0 is enabled by setting this bit to 1. |
| 3 | IE1 | 0 | RW | **External Interrupt 1.**<br>This flag is set by hardware when an edge/level of type defined by IT1 is detected. It can be cleared by firmware but is automatically cleared when the CPU vectors to the External Interrupt 1 service routine in edge-triggered mode. |
| 2 | IT1 | 0 | RW | **Interrupt 1 Type Select.**<br>This bit selects whether the configured INT1 interrupt will be edge or level sensitive. INT1 is configured active low or high by the IN1PL bit in register IT01CF. |

| Value | Name | Description |
|---|---|---|
| 0 | LEVEL | INT1 is level triggered. |
| 1 | EDGE | INT1 is edge triggered. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 1 | IE0 | 0 | RW | **External Interrupt 0.**<br>This flag is set by hardware when an edge/level of type defined by IT0 is detected. It can be cleared by firmware but is automatically cleared when the CPU vectors to the External Interrupt 0 service routine in edge-triggered mode. |
| 0 | IT0 | 0 | RW | **Interrupt 0 Type Select.**<br>This bit selects whether the configured INT0 interrupt will be edge or level sensitive. INT0 is configured active low or high by the IN0PL bit in register IT01CF. |

| Value | Name | Description |
|---|---|---|
| 0 | LEVEL | INT0 is level triggered. |
| 1 | EDGE | INT0 is edge triggered. |

### 21.4.3 TMOD: Timer 0/1 Mode

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | GATE1 | CT1 | T1M | | GATE0 | CT0 | T0M | |
| Access | RW | RW | RW | | RW | RW | RW | |
| Reset | 0 | 0 | 0x0 | | 0 | 0 | 0x0 | |
| SFR Page = 0x0; SFR Address: 0x89 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | GATE1 | 0 | RW | **Timer 1 Gate Control.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Timer 1 enabled when TR1 = 1 irrespective of INT1 logic level. |
| | 1 | ENABLED | | Timer 1 enabled only when TR1 = 1 and INT1 is active as defined by bit IN1PL in register IT01CF. |
| 6 | CT1 | 0 | RW | **Counter/Timer 1 Select.** |
| | Value | Name | | Description |
| | 0 | TIMER | | Timer Mode. Timer 1 increments on the clock defined by T1M in the CKCON0 register. |
| | 1 | COUNTER | | Counter Mode. Timer 1 increments on high-to-low transitions of an external pin (T1). |
| 5:4 | T1M | 0x0 | RW | **Timer 1 Mode Select.** |
| | These bits select the Timer 1 operation mode. | | | |
| | Value | Name | | Description |
| | 0x0 | MODE0 | | Mode 0, 13-bit Counter/Timer |
| | 0x1 | MODE1 | | Mode 1, 16-bit Counter/Timer |
| | 0x2 | MODE2 | | Mode 2, 8-bit Counter/Timer with Auto-Reload |
| | 0x3 | MODE3 | | Mode 3, Timer 1 Inactive |
| 3 | GATE0 | 0 | RW | **Timer 0 Gate Control.** |
| | Value | Name | | Description |
| | 0 | DISABLED | | Timer 0 enabled when TR0 = 1 irrespective of INT0 logic level. |
| | 1 | ENABLED | | Timer 0 enabled only when TR0 = 1 and INT0 is active as defined by bit IN0PL in register IT01CF. |
| 2 | CT0 | 0 | RW | **Counter/Timer 0 Select.** |
| | Value | Name | | Description |
| | 0 | TIMER | | Timer Mode. Timer 0 increments on the clock defined by T0M in the CKCON0 register. |
| | 1 | COUNTER | | Counter Mode. Timer 0 increments on high-to-low transitions of an external pin (T0). |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 1:0 | T0M | 0x0 | RW | **Timer 0 Mode Select.** |

These bits select the Timer 0 operation mode.

| Value | Name | Description |
|---|---|---|
| 0x0 | MODE0 | Mode 0, 13-bit Counter/Timer |
| 0x1 | MODE1 | Mode 1, 16-bit Counter/Timer |
| 0x2 | MODE2 | Mode 2, 8-bit Counter/Timer with Auto-Reload |
| 0x3 | MODE3 | Mode 3, Two 8-bit Counter/Timers |

### 21.4.4 TL0: Timer 0 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TL0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x8A | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TL0 | 0x00 | RW | **Timer 0 Low Byte.** |

The TL0 register is the low byte of the 16-bit Timer 0.

### 21.4.5 TL1: Timer 1 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TL1 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x8B | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TL1 | 0x00 | RW | **Timer 1 Low Byte.** |

The TL1 register is the low byte of the 16-bit Timer 1.

### 21.4.6 TH0: Timer 0 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TH0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x8C | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TH0 | 0x00 | RW | **Timer 0 High Byte.**<br>The TH0 register is the high byte of the 16-bit Timer 0. |

### 21.4.7 TH1: Timer 1 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TH1 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x8D | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TH1 | 0x00 | RW | **Timer 1 High Byte.**<br>The TH1 register is the high byte of the 16-bit Timer 1. |

### 21.4.8 TMR2CN0: Timer 2 Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF2H | TF2L | TF2LEN | TF2CEN | T2SPLIT | TR2 | T2XCLK | |
| Access | RW | RW | RW | RW | RW | RW | RW | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0x0 | |
| SFR Page = 0x0; SFR Address: 0xC8 (bit-addressable) | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | TF2H | 0 | RW | **Timer 2 High Byte Overflow Flag.**<br>Set by hardware when the Timer 2 high byte overflows from 0xFF to 0x00. In 16-bit mode, this will occur when Timer 2 overflows from 0xFFFF to 0x0000. When the Timer 2 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 2 interrupt service routine. This bit must be cleared by firmware. |
| 6 | TF2L | 0 | RW | **Timer 2 Low Byte Overflow Flag.**<br>Set by hardware when the Timer 2 low byte overflows from 0xFF to 0x00. TF2L will be set when the low byte overflows regardless of the Timer 2 mode. This bit must be cleared by firmware. |
| 5 | TF2LEN | 0 | RW | **Timer 2 Low Byte Interrupt Enable.**<br>When set to 1, this bit enables Timer 2 Low Byte interrupts. If Timer 2 interrupts are also enabled, an interrupt will be generated when the low byte of Timer 2 overflows. |
| 4 | TF2CEN | 0 | RW | **Timer 2 Capture Enable.**<br>When set to 1, this bit enables Timer 2 Capture Mode. If TF2CEN is set and Timer 2 interrupts are enabled, an interrupt will be generated based on the selected input capture source, and the current 16-bit timer value in TMR2H:TMR2L will be copied to TMR2RLH:TMR2RLL. |
| 3 | T2SPLIT | 0 | RW | **Timer 2 Split Mode Enable.**<br>When this bit is set, Timer 2 operates as two 8-bit timers with auto-reload. |

| Value | Name | Description |
|---|---|---|
| 0 | 16_BIT_RELOAD | Timer 2 operates in 16-bit auto-reload mode. |
| 1 | 8_BIT_RELOAD | Timer 2 operates as two 8-bit auto-reload timers. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2 | TR2 | 0 | RW | **Timer 2 Run Control.**<br>Timer 2 is enabled by setting this bit to 1. In 8-bit mode, this bit enables/disables TMR2H only; TMR2L is always enabled in split mode. |
| 1:0 | T2XCLK | 0x0 | RW | **Timer 2 External Clock Select.**<br>This bit selects the external clock source for Timer 2. If Timer 2 is in 8-bit mode, this bit selects the external oscillator clock source for both timer bytes. However, the Timer 2 Clock Select bits (T2MH and T2ML) may still be used to select between the external clock and the system clock for either timer. Note: External clock sources are synchronized with the system clock. |

| Value | Name | Description |
|---|---|---|
| 0x0 | SYSCLK_DIV_12_CAP_RTC | External Clock is SYSCLK/12. Capture trigger is RTC/8. |
| 0x1 | CMP_0_CAP_RTC | External Clock is Comparator 0. Capture trigger is RTC/8. |
| 0x2 | SYSCLK_DIV_12_CAP_CMP | External Clock is SYSCLK/12. Capture trigger is Comparator 0. |
| 0x3 | RTC_DIV_8_CAP_CMP | External Clock is RTC/8. Capture trigger is Comparator 0. |

### 21.4.9 TMR2RLL: Timer 2 Reload Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2RLL | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xCA | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR2RLL | 0x00 | RW | **Timer 2 Reload Low Byte.**<br>When operating in one of the auto-reload modes, TMR2RLL holds the reload value for the low byte of Timer 2 (TMR2L). When operating in capture mode, TMR2RLL is the captured value of TMR2L. |

### 21.4.10 TMR2RLH: Timer 2 Reload High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2RLH | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xCB | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR2RLH | 0x00 | RW | **Timer 2 Reload High Byte.**<br>When operating in one of the auto-reload modes, TMR2RLH holds the reload value for the high byte of Timer 2 (TMR2H). When operating in capture mode, TMR2RLH is the captured value of TMR2H. |

### 21.4.11 TMR2L: Timer 2 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2L | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xCC | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR2L | 0x00 | RW | **Timer 2 Low Byte.**<br>In 16-bit mode, the TMR2L register contains the low byte of the 16-bit Timer 2. In 8-bit mode, TMR2L contains the 8-bit low byte timer value. |

### 21.4.12 TMR2H: Timer 2 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR2H | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0xCD | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR2H | 0x00 | RW | **Timer 2 High Byte.**<br>In 16-bit mode, the TMR2H register contains the high byte of the 16-bit Timer 2. In 8-bit mode, TMR2H contains the 8-bit high byte timer value. |

### 21.4.13 TMR3CN0: Timer 3 Control 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TF3H | TF3L | TF3LEN | TF3CEN | T3SPLIT | TR3 | T3XCLK | |
| Access | RW | RW | RW | RW | RW | RW | R | RW |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0x0 | |
| SFR Page = 0x0; SFR Address: 0x91 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | TF3H | 0 | RW | **Timer 3 High Byte Overflow Flag.**<br>Set by hardware when the Timer 3 high byte overflows from 0xFF to 0x00. In 16-bit mode, this will occur when Timer 3 overflows from 0xFFFF to 0x0000. When the Timer 3 interrupt is enabled, setting this bit causes the CPU to vector to the Timer 3 interrupt service routine. This bit must be cleared by firmware. |
| 6 | TF3L | 0 | RW | **Timer 3 Low Byte Overflow Flag.**<br>Set by hardware when the Timer 3 low byte overflows from 0xFF to 0x00. TF3L will be set when the low byte overflows regardless of the Timer 3 mode. This bit must be cleared by firmware. |
| 5 | TF3LEN | 0 | RW | **Timer 3 Low Byte Interrupt Enable.**<br>When set to 1, this bit enables Timer 3 Low Byte interrupts. If Timer 3 interrupts are also enabled, an interrupt will be generated when the low byte of Timer 3 overflows. |
| 4 | TF3CEN | 0 | RW | **Timer 3 Capture Enable.**<br>When set to 1, this bit enables Timer 3 Capture Mode. If TF3CEN is set and Timer 3 interrupts are enabled, an interrupt will be generated based on the selected input capture source, and the current 16-bit timer value in TMR3H:TMR3L will be copied to TMR3RLH:TMR3RLL. |
| 3 | T3SPLIT | 0 | RW | **Timer 3 Split Mode Enable.**<br>When this bit is set, Timer 3 operates as two 8-bit timers with auto-reload. |

| Value | Name | Description |
|---|---|---|
| 0 | 16_BIT_RELOAD | Timer 3 operates in 16-bit auto-reload mode. |
| 1 | 8_BIT_RELOAD | Timer 3 operates as two 8-bit auto-reload timers. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 2 | TR3 | 0 | RW | **Timer 3 Run Control.**<br>Timer 3 is enabled by setting this bit to 1. In 8-bit mode, this bit enables/disables TMR3H only; TMR3L is always enabled in split mode. |
| 1:0 | T3XCLK | 0x0 | RW | **Timer 3 External Clock Select.**<br>This bit selects the external clock source for Timer 3. If Timer 3 is in 8-bit mode, this bit selects the external oscillator clock source for both timer bytes. However, the Timer 3 Clock Select bits (T3MH and T3ML) may still be used to select between the external clock and the system clock for either timer. Note: External clock sources are synchronized with the system clock. |

| Value | Name | Description |
|---|---|---|
| 0x0 | SYSCLK_DIV_12_CAP_RTC | External Clock is SYSCLK/12. Capture trigger is RTC. |
| 0x1 | EXTOSC_DIV_8_CAP_RTC | External Clock is External Oscillator/8. Capture trigger is RTC. |
| 0x2 | SYSCLK_DIV_12_CAP_EXTOSC | External Clock is SYSCLK/12. Capture trigger is External Oscillator/8. |
| 0x3 | RTC_CAP_EXTOSC | External Clock is RTC. Capture trigger is External Oscillator/8. |

### 21.4.14 TMR3RLL: Timer 3 Reload Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3RLL | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x92 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR3RLL | 0x00 | RW | **Timer 3 Reload Low Byte.**<br>When operating in one of the auto-reload modes, TMR3RLL holds the reload value for the low byte of Timer 3 (TMR3L). When operating in capture mode, TMR3RLL is the captured value of TMR3L. |

### 21.4.15 TMR3RLH: Timer 3 Reload High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3RLH | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x93 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR3RLH | 0x00 | RW | **Timer 3 Reload High Byte.**<br>When operating in one of the auto-reload modes, TMR3RLH holds the reload value for the high byte of Timer 3 (TMR3H). When operating in capture mode, TMR3RLH is the captured value of TMR3H. |

### 21.4.16 TMR3L: Timer 3 Low Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3L | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x94 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR3L | 0x00 | RW | **Timer 3 Low Byte.**<br>In 16-bit mode, the TMR3L register contains the low byte of the 16-bit Timer 3. In 8-bit mode, TMR3L contains the 8-bit low byte timer value. |

### 21.4.17 TMR3H: Timer 3 High Byte

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TMR3H | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x95 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | TMR3H | 0x00 | RW | **Timer 3 High Byte.**<br>In 16-bit mode, the TMR3H register contains the high byte of the 16-bit Timer 3. In 8-bit mode, TMR3H contains the 8-bit high byte timer value. |

# 22. Universal Asynchronous Receiver/Transmitter 0 (UART0)

## 22.1 Introduction

UART0 is an asynchronous, full duplex serial port offering modes 1 and 3 of the standard 8051 UART. Enhanced baud rate support allows a wide range of clock sources to generate standard baud rates. Received data buffering allows UART0 to start reception of a second incoming data byte before software has finished reading the previous data byte.

UART0 has two associated SFRs: Serial Control Register 0 (SCON0) and Serial Data Buffer 0 (SBUF0). The single SBUF0 location provides access to both transmit and receive registers.

**Note:** Writes to SBUF0 always access the transmit register. Reads of SBUF0 always access the buffered receive register; it is not possible to read data from the transmit register.

With UART0 interrupts enabled, an interrupt is generated each time a transmit is completed (TI is set in SCON0), or a data byte has been received (RI is set in SCON0). The UART0 interrupt flags are not cleared by hardware when the CPU vectors to the interrupt service routine. They must be cleared manually by software, allowing software to determine the cause of the UART0 interrupt (transmit complete or receive complete).

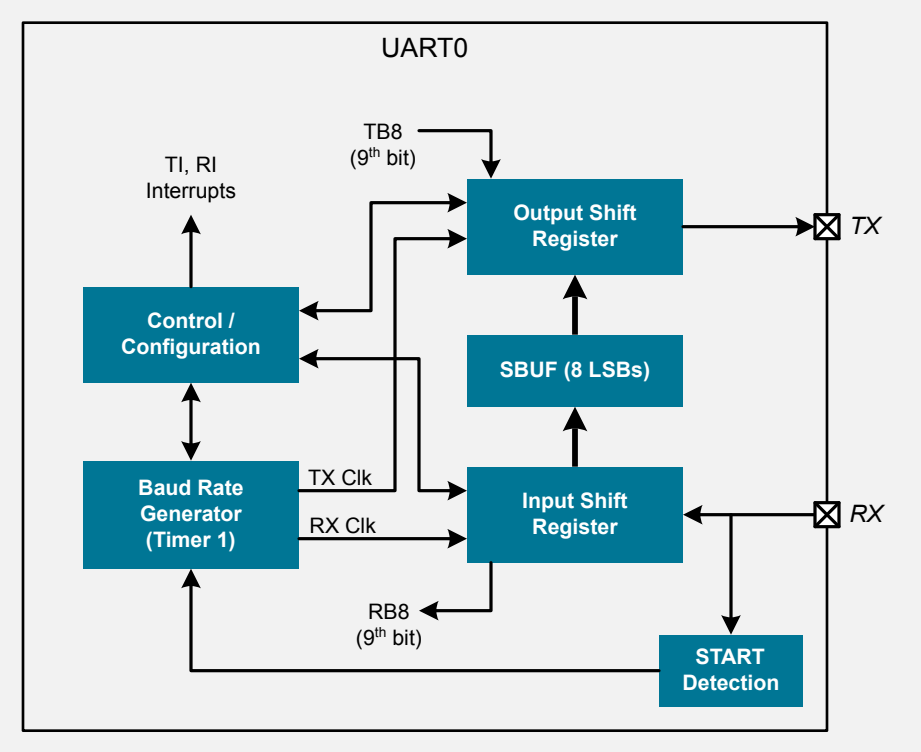



**Figure 22.1. UART0 Block Diagram**

## 22.2 Features

The UART uses two signals (TX and RX) and a predetermined fixed baud rate to provide asynchronous communications with other devices.

The UART module provides the following features:

- Asynchronous transmissions and receptions
- Baud rates up to SYSCLK/2 (transmit) or SYSCLK/8 (receive)
- 8- or 9-bit data
- Automatic start and stop generation

## 22.3 Functional Description

### 22.3.1 Baud Rate Generation

The UART0 baud rate is generated by Timer 1 in 8-bit auto-reload mode. The TX clock is generated by TL1; the RX clock is generated by a copy of TL1, which is not user-accessible. Both TX and RX timer overflows are divided by two to generate the TX and RX baud rates. The RX timer runs when Timer 1 is enabled and uses the same reload value (TH1). However, an RX timer reload is forced when a START condition is detected on the RX pin. This allows a receive to begin any time a START is detected, independent of the TX timer state.

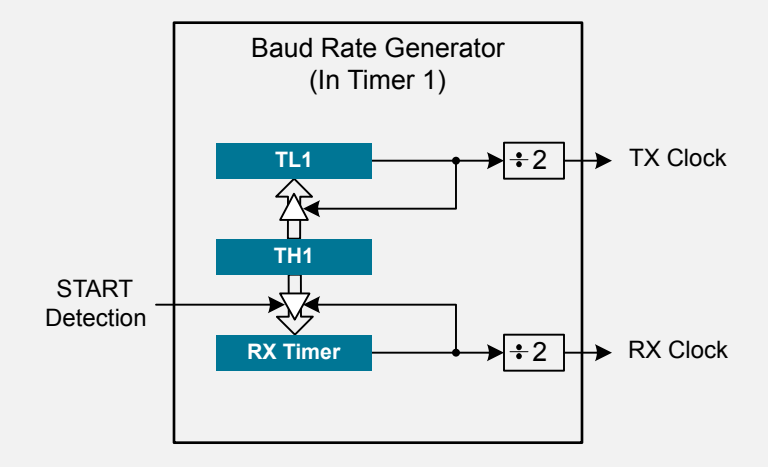



**Figure 22.2. UART0 Baud Rate Logic Block Diagram**

Timer 1 should be configured for 8-bit auto-reload mode (mode 2). The Timer 1 reload value and prescaler should be set so that overflows occur at twice the desired UART0 baud rate. The UART0 baud rate is half of the Timer 1 overflow rate. Configuring the Timer 1 overflow rate is discussed in the timer sections.

### 22.3.2 Data Format

UART0 has two options for data formatting. All data transfers begin with a start bit (logic low), followed by the data (sent LSB-first), and end with a stop bit (logic high). The data length of the UART0 module is normally 8 bits. An extra 9th bit may be added to the MSB of data field for use in multi-processor communications or for implementing parity checks on the data. The S0MODE bit in the SCON register selects between 8 or 9-bit data transfers.

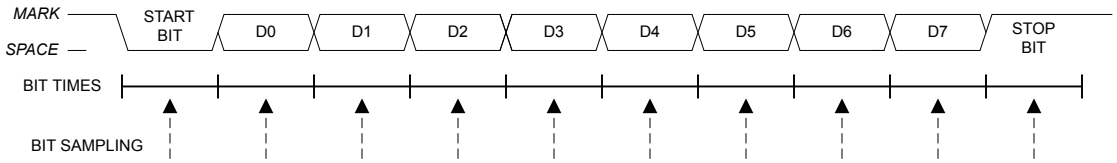



**Figure 22.3. 8-Bit Data Transfer**

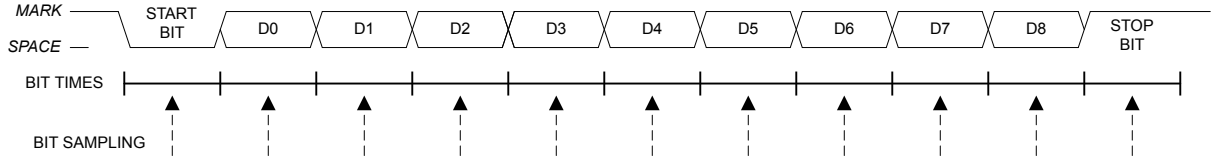



**Figure 22.4. 9-Bit Data Transfer**

### 22.3.3 Data Transfer

UART0 provides standard asynchronous, full duplex communication. All data sent or received goes through the SBUF0 register and (in 9-bit mode) the RB8 bit in the SCON0 register.

#### Transmitting Data

Data transmission is initiated when software writes a data byte to the SBUF0 register. If 9-bit mode is used, software should set up the desired 9th bit in TB8 prior to writing SBUF0. Data is transmitted LSB first from the TX pin. The TI flag in SCON0 is set at the end of the transmission (at the beginning of the stop-bit time). If TI interrupts are enabled, TI will trigger an interrupt.

#### Receiving Data

To enable data reception, firmware should write the REN bit to 1. Data reception begins when a start condition is recognized on the RX pin. Data will be received at the selected baud rate through the end of the data phase. Data will be transferred into the receive buffer under the following conditions:

- There is room in the receive buffer for the data.
- MCE is set to 1 and the stop bit is also 1 (8-bit mode).
- MCE is set to 1 and the 9th bit is also 1 (9-bit mode).
- MCE is 0 (stop or 9th bit will be ignored).

In the event that there is not room in the receive buffer for the data, the most recently received data will be lost. The RI flag will be set any time that valid data has been pushed into the receive buffer. If RI interrupts are enabled, RI will trigger an interrupt. Firmware may read the 8 LSBs of received data by reading the SBUF0 register. The RB8 bit in SCON0 will represent the 9th received bit (in 9-bit mode) or the stop bit (in 8-bit mode), and should be read prior to reading SBUF0.

### 22.3.4 Multiprocessor Communications

9-Bit UART mode supports multiprocessor communication between a master processor and one or more slave processors by special use of the ninth data bit. When a master processor wants to transmit to one or more slaves, it first sends an address byte to select the target(s). An address byte differs from a data byte in that its ninth bit is logic 1; in a data byte, the ninth bit is always set to logic 0.

Setting the MCE bit of a slave processor configures its UART such that when a stop bit is received, the UART will generate an interrupt only if the ninth bit is logic 1 (RB8 = 1) signifying an address byte has been received. In the UART interrupt handler, software will compare the received address with the slave's own assigned 8-bit address. If the addresses match, the slave will clear its MCE bit to enable interrupts on the reception of the following data byte(s). Slaves that weren't addressed leave their MCE bits set and do not generate interrupts on the reception of the following data bytes, thereby ignoring the data. Once the entire message is received, the addressed slave resets its MCE bit to ignore all transmissions until it receives the next address byte.

Multiple addresses can be assigned to a single slave and/or a single address can be assigned to multiple slaves, thereby enabling "broadcast" transmissions to more than one slave simultaneously. The master processor can be configured to receive all transmissions or a protocol can be implemented such that the master/slave role is temporarily reversed to enable half-duplex transmission between the original master and slave(s).

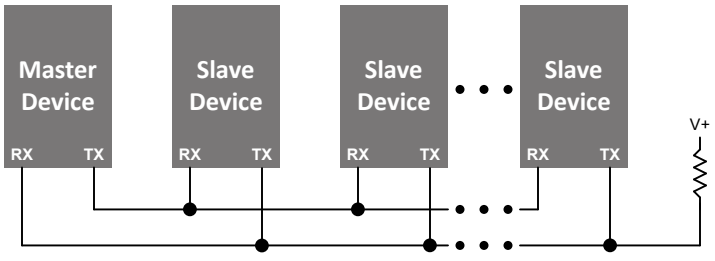



**Figure 22.5. Multi-Processor Mode Interconnect Diagram**

## 22.4 UART0 Control Registers

### 22.4.1 SCON0: UART0 Serial Port Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SMODE | Reserved | MCE | REN | TB8 | RB8 | TI | RI |
| Access | RW | R | RW | RW | RW | RW | RW | RW |
| Reset | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

SFR Page = 0x0; SFR Address: 0x98 (bit-addressable)

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7 | SMODE | 0 | RW | **Serial Port 0 Operation Mode.**<br>Selects the UART0 Operation Mode. |

| Value | Name | Description |
|---|---|---|
| 0 | 8_BIT | 8-bit UART with Variable Baud Rate (Mode 0). |
| 1 | 9_BIT | 9-bit UART with Variable Baud Rate (Mode 1). |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| *6* | *Reserved* | *Must write reset value.* | | |
| 5 | MCE | 0 | RW | **Multiprocessor Communication Enable.**<br>This bit enables checking of the stop bit or the 9th bit in multi-drop communication buses. The function of this bit is dependent on the UART0 operation mode selected by the SMODE bit. In Mode 0 (8-bits), the peripheral will check that the stop bit is logic 1. In Mode 1 (9-bits) the peripheral will check for a logic 1 on the 9th bit. |

| Value | Name | Description |
|---|---|---|
| 0 | MULTI_DISABLED | Ignore level of 9th bit / Stop bit. |
| 1 | MULTI_ENABLED | RI is set and an interrupt is generated only when the stop bit is logic 1 (Mode 0) or when the 9th bit is logic 1 (Mode 1). |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 4 | REN | 0 | RW | **Receive Enable.** |

| Value | Name | Description |
|---|---|---|
| 0 | RECEIVE_DISABLED | UART0 reception disabled. |
| 1 | RECEIVE_ENABLED | UART0 reception enabled. |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 3 | TB8 | 0 | RW | **Ninth Transmission Bit.**<br>The logic level of this bit will be sent as the ninth transmission bit in 9-bit UART Mode (Mode 1). Unused in 8-bit mode (Mode 0). |
| 2 | RB8 | 0 | RW | **Ninth Receive Bit.**<br>RB8 is assigned the value of the STOP bit in Mode 0; it is assigned the value of the 9th data bit in Mode 1. |
| 1 | TI | 0 | RW | **Transmit Interrupt Flag.**<br>Set by hardware when a byte of data has been transmitted by UART0 (after the 8th bit in 8-bit UART Mode, or at the beginning of the STOP bit in 9-bit UART Mode). When the UART0 interrupt is enabled, setting this bit causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by firmware. |
| 0 | RI | 0 | RW | **Receive Interrupt Flag.**<br>Set to 1 by hardware when a byte of data has been received by UART0 (set at the STOP bit sampling time). When the UART0 interrupt is enabled, setting this bit to 1 causes the CPU to vector to the UART0 interrupt service routine. This bit must be cleared manually by firmware. |

### 22.4.2 SBUF0: UART0 Serial Port Data Buffer

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SBUF0 | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| SFR Page = 0x0; SFR Address: 0x99 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | SBUF0 | 0x00 | RW | **Serial Data Buffer.**<br>This SFR accesses two registers: a transmit shift register and a receive latch register. When data is written to SBUF0, it goes to the transmit shift register and is held for serial transmission. Writing a byte to SBUF0 initiates the transmission. A read of SBUF0 returns the contents of the receive latch. |

# 23. C2 Debug and Programming Interface

## 23.1 Introduction

The device includes an on-chip Silicon Labs 2-Wire (C2) debug interface that allows flash programming and in-system debugging with the production part installed in the end application. The C2 interface uses a clock signal (C2CK) and a bi-directional C2 data signal (C2D) to transfer information between the device and a host system. Details on the C2 protocol can be found in the C2 Interface Specification.

## 23.2 Features

The C2 interface provides the following features:

- In-system device programming and debugging.
- Non-intrusive - no firmware or hardware peripheral resources required.
- Allows inspection and modification of all memory spaces and registers.
- Provides hardware breakpoints and single-step capabilites.
- Can be locked via flash security mechanism to prevent unwanted access.

## 23.3 Pin Sharing

The C2 protocol allows the C2 pins to be shared with user functions so that in-system debugging and flash programming may be performed. C2CK is shared with the RSTb pin, while the C2D signal is shared with a port I/O pin. This is possible because C2 communication is typically performed when the device is in the halt state, where all on-chip peripherals and user software are stalled. In this halted state, the C2 interface can safely "borrow" the C2CK and C2D pins. In most applications, external resistors are required to isolate C2 interface traffic from the user application.

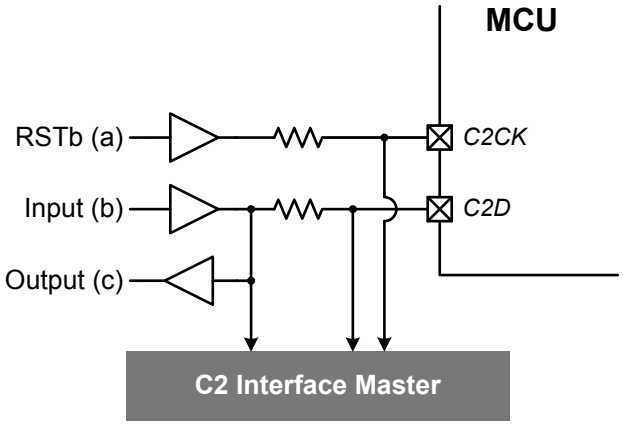



**Figure 23.1. Typical C2 Pin Sharing**

The configuration above assumes the following:

- The user input (b) cannot change state while the target device is halted.
- The RSTb pin on the target device is used as an input only.

Additional resistors may be necessary depending on the specific application.

## 23.4 C2 Interface Registers

### 23.4.1 C2ADD: C2 Address

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | C2ADD | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| This register is part of the C2 protocol. | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | C2ADD | 0x00 | RW | **C2 Address.**<br>The C2ADD register is accessed via the C2 interface. The value written to C2ADD selects the target data register for C2 Data Read and Data Write commands.<br>0x00: C2DEVID<br>0x01: C2REVID<br>0x02: C2FPCTL<br>0xB4: C2FPDAT |

### 23.4.2 C2DEVID: C2 Device ID

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | C2DEVID | | | | | | | |
| Access | R | | | | | | | |
| Reset | 0x25 | | | | | | | |
| C2 Address: 0x00 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | C2DEVID | 0x25 | R | **Device ID.**<br>This read-only register returns the 8-bit device ID: 0x25. |

### 23.4.3 C2REVID: C2 Revision ID

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | C2REVID | | | | | | | |
| Access | R | | | | | | | |
| Reset | Varies | | | | | | | |
| C2 Address: 0x01 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | C2REVID | Varies | R | **Revision ID.**<br>This read-only register returns the 8-bit revision ID. For example: 0x02 = Revision A. |

### 23.4.4 C2FPCTL: C2 Flash Programming Control

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | C2FPCTL | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| C2 Address: 0x02 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | C2FPCTL | 0x00 | RW | **Flash Programming Control Register.**<br>This register is used to enable flash programming via the C2 interface. To enable C2 flash programming, the following codes must be written in order: 0x02, 0x01. Note that once C2 flash programming is enabled, a system reset must be issued to resume normal operation. |

### 23.4.5 C2FPDAT: C2 Flash Programming Data

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | C2FPDAT | | | | | | | |
| Access | RW | | | | | | | |
| Reset | 0x00 | | | | | | | |
| C2 Address: 0xB4 | | | | | | | | |

| Bit | Name | Reset | Access | Description |
|---|---|---|---|---|
| 7:0 | C2FPDAT | 0x00 | RW | **C2 Flash Programming Data Register.**<br>This register is used to pass flash commands, addresses, and data during C2 flash accesses. Valid commands are listed below.<br>0x03: Device Erase<br>0x06: Flash Block Read<br>0x07: Flash Block Write<br>0x08: Flash Page Erase |

**1. System Overview . . . 1**
1.1 Introduction . . . 1
1.2 Power . . . 2
1.3 I/O . . . 2
1.4 Clocking . . . 2
1.5 Counters/Timers and PWM . . . 3
1.6 Communications and Other Digital Peripherals . . . 4
1.7 Analog . . . 5
1.8 Reset Sources . . . 7
1.9 Debugging . . . 7
1.10 Bootloader . . . 7
**2. Memory Organization . . . 8**
2.1 Memory Organization . . . 8
2.2 Program Memory . . . 8
2.3 Data Memory . . . 8
2.4 Memory Map . . . 10
**3. Special Function Registers . . . 14**
3.1 Special Function Register Access . . . 14
3.2 Special Function Register Memory Map . . . 15
3.3 SFR Access Control Registers . . . 20
3.3.1 SFRPAGE: SFR Page . . . 20
**4. Flash Memory . . . 21**
4.1 Introduction . . . 21
4.2 Features . . . 23
4.3 Functional Description . . . 24
4.3.1 Security Options . . . 24
4.3.2 Programming the Flash Memory . . . 25
4.3.2.1 Flash Lock and Key Functions . . . 25
4.3.2.2 Flash Page Erase Procedure . . . 25
4.3.2.3 Flash Byte Write Procedure . . . 25
4.3.3 Flash Write and Erase Precautions . . . 26
4.3.4 Minimizing Flash Read Current . . . 27
4.4 Flash Control Registers . . . 28
4.4.1 PSCTL: Program Store Control . . . 28
4.4.2 FLKEY: Flash Lock and Key . . . 29
4.4.3 FLSCL: Flash Scale . . . 29
**5. Device Identification . . . 30**
5.1 Device Identification . . . 30
5.2 Unique Identifier . . . 30

5.3 Device Identification Registers . . . . . .30
5.3.1 DERIVID: Device Identification . . . . . .30
5.3.2 REVID: Revision Identifcation . . . . . .31

**6. Interrupts . . . . . 32**

6.1 Introduction . . . . . .32

6.2 Interrupt Sources and Vectors . . . . . .32
6.2.1 Interrupt Priorities . . . . . .32
6.2.2 Interrupt Latency . . . . . .32
6.2.3 Interrupt Summary . . . . . .33

6.3 Interrupt Control Registers . . . . . .34
6.3.1 IE: Interrupt Enable . . . . . .34
6.3.2 IP: Interrupt Priority . . . . . .36
6.3.3 EIE1: Extended Interrupt Enable 1 . . . . . .38
6.3.4 EIP1: Extended Interrupt Priority 1 . . . . . .40
6.3.5 EIE2: Extended Interrupt Enable 2 . . . . . .42
6.3.6 EIP2: Extended Interrupt Priority 2 . . . . . .44

**7. Power Management and Internal Regulators . . . . . 46**

7.1 Introduction . . . . . .46

7.2 Features . . . . . .47

7.3 Idle Mode . . . . . .47

7.4 Stop Mode . . . . . .48

7.5 Suspend Mode . . . . . .48

7.6 Sleep Mode . . . . . .49
7.6.1 Configuring Wakeup Sources . . . . . .49
7.6.2 Determining the Event that Caused the Last Wakeup . . . . . .50

7.7 Power Management Control Registers . . . . . .50
7.7.1 PCON0: Power Control 0 . . . . . .50
7.7.2 PMU0CF: Power Management Unit Configuration . . . . . .51
7.7.3 PMU0FL: Power Management Unit Flag . . . . . .52
7.7.4 PMU0MD: Power Management Unit Mode . . . . . .52
7.7.5 REG0CN: Voltage Regulator Control . . . . . .53

**8. Clocking and Oscillators . . . . . 54**

8.1 Introduction . . . . . .54

8.2 Features . . . . . .54

8.3 Functional Description . . . . . .54
8.3.1 Clock Selection . . . . . .54
8.3.2 LPOSC0 20 MHz Internal Oscillator . . . . . .54
8.3.3 HFOSC0 24.5 MHz Internal Oscillator . . . . . .55
8.3.4 RTC0 Oscillator . . . . . .55
8.3.5 External Crystal . . . . . .56
8.3.6 External RC and C Modes . . . . . .58
8.3.7 External CMOS . . . . . .60

8.4 Clocking and Oscillator Control Registers . . . . . .61

8.4.1 CLKSEL: Clock Select . . . . . .61
8.4.2 HFO0CAL: High Frequency Oscillator Calibration . . . . . .62
8.4.3 HFO0CN: High Frequency Oscillator Control . . . . . .62
8.4.4 XOSC0CN: External Oscillator Control . . . . . .63

**9. Real Time Clock (RTC0) . . . . . 64**

9.1 Introduction. . . . . .64
9.2 Features. . . . . .64
9.3 Functional Description . . . . . .64
9.3.1 Interface . . . . . .64
9.3.2 Clocking Options . . . . . .66
9.3.3 Timer and Alarm . . . . . .70
9.4 RTC Control Registers . . . . . .71
9.4.1 RTC0KEY: RTC Lock and Key . . . . . .71
9.4.2 RTC0ADR: RTC Address . . . . . .72
9.4.3 RTC0DAT: RTC Data . . . . . .72
9.4.4 RTC0CN0: RTC Control 0 . . . . . .73
9.4.5 RTC0XCN0: RTC Oscillator Control 0 . . . . . .75
9.4.6 RTC0XCF: RTC Oscillator Configuration . . . . . .76
9.4.7 CAPTURE0: RTC Timer Capture 0 . . . . . .76
9.4.8 CAPTURE1: RTC Timer Capture 1 . . . . . .77
9.4.9 CAPTURE2: RTC Timer Capture 2 . . . . . .77
9.4.10 CAPTURE3: RTC Timer Capture 3. . . . . .77
9.4.11 ALARM0: RTC Alarm Programmed Value 0. . . . . .78
9.4.12 ALARM1: RTC Alarm Programmed Value 1. . . . . .78
9.4.13 ALARM2: RTC Alarm Programmed Value 2. . . . . .78
9.4.14 ALARM3: RTC Alarm Programmed Value 3. . . . . .79

**10. Reset Sources and Power Supply Monitor . . . . . 80**

10.1 Introduction . . . . . .80
10.2 Features . . . . . .80
10.3 Functional Description . . . . . .81
10.3.1 Device Reset . . . . . .81
10.3.2 Power-On Reset . . . . . .82
10.3.3 Supply Monitor Reset . . . . . .83
10.3.4 External Reset . . . . . .83
10.3.5 Missing Clock Detector Reset . . . . . .83
10.3.6 Comparator (CMP0) Reset . . . . . .83
10.3.7 PCA Watchdog Timer Reset . . . . . .84
10.3.8 Flash Error Reset . . . . . .84
10.3.9 Software Reset . . . . . .84
10.3.10 RTC Reset . . . . . .84
10.4 Reset Sources and Supply Monitor Control Registers . . . . . .85
10.4.1 RSTSRC: Reset Source . . . . . .85
10.4.2 VDM0CN: VDD Supply Monitor Control . . . . . .86

**11. CIP-51 Microcontroller Core . . . . . 87**

11.1 Introduction . . . . . .87

11.2 Features . . . . . .88
11.3 Functional Description . . . . . .88
11.3.1 Programming and Debugging Support . . . . . .88
11.3.2 Instruction Set . . . . . .89
11.4 CPU Core Registers . . . . . .92
11.4.1 DPL: Data Pointer Low . . . . . .92
11.4.2 DPH: Data Pointer High . . . . . .93
11.4.3 SP: Stack Pointer . . . . . .93
11.4.4 ACC: Accumulator . . . . . .93
11.4.5 B: B Register . . . . . .94
11.4.6 PSW: Program Status Word . . . . . .95
**12. Port I/O, Crossbar, External Interrupts, and Port Match . . . . . 96**
12.1 Introduction . . . . . .96
12.2 Features . . . . . .96
12.3 Functional Description . . . . . .97
12.3.1 Port I/O Modes of Operation . . . . . .97
12.3.1.1 Pin Drive Strength . . . . . .98
12.3.2 Analog and Digital Functions . . . . . .98
12.3.2.1 Port I/O Analog Assignments . . . . . .98
12.3.2.2 Port I/O Digital Assignments . . . . . .99
12.3.3 Priority Crossbar Decoder . . . . . 100
12.3.3.1 Crossbar Functional Map . . . . . 101
12.3.4 INT0 and INT1 . . . . . 102
12.3.5 Port Match . . . . . 102
12.3.6 Direct Port I/O Access (Read/Write) . . . . . 102
12.4 Port I/O Control Registers . . . . . 103
12.4.1 XBR0: Port I/O Crossbar 0 . . . . . 103
12.4.2 XBR1: Port I/O Crossbar 1 . . . . . 104
12.4.3 XBR2: Port I/O Crossbar 2 . . . . . 105
12.4.4 P0MASK: Port 0 Mask . . . . . 106
12.4.5 P0MAT: Port 0 Match . . . . . 107
12.4.6 P0: Port 0 Pin Latch . . . . . 108
12.4.7 P0MDIN: Port 0 Input Mode . . . . . 109
12.4.8 P0MDOUT: Port 0 Output Mode . . . . . 110
12.4.9 P0SKIP: Port 0 Skip . . . . . 111
12.4.10 P0DRV: Port 0 Drive Strength . . . . . 112
12.4.11 P1MASK: Port 1 Mask . . . . . 113
12.4.12 P1MAT: Port 1 Match . . . . . 114
12.4.13 P1: Port 1 Pin Latch . . . . . 115
12.4.14 P1MDIN: Port 1 Input Mode . . . . . 116
12.4.15 P1MDOUT: Port 1 Output Mode . . . . . 117
12.4.16 P1SKIP: Port 1 Skip . . . . . 118
12.4.17 P1DRV: Port 1 Drive Strength . . . . . 119
12.4.18 P2: Port 2 Pin Latch . . . . . 120
12.4.19 P2MDOUT: Port 2 Output Mode . . . . . 120
12.4.20 P2DRV: Port 2 Drive Strength . . . . . 121
12.5 INT0 and INT1 Control Registers . . . . . 122

12.5.1 IT01CF: INT0/INT1 Configuration . . . 122

**13. Analog-to-Digital Converter (ADC0) . . . 124**

13.1 Introduction . . . 124
13.2 Features . . . 124
13.3 Functional Description . . . 125
13.3.1 Clocking . . . 125
13.3.2 Voltage Reference Options . . . 125
13.3.2.1 Internal Voltage Reference . . . 125
13.3.2.2 Supply or LDO Voltage Reference . . . 125
13.3.2.3 External Voltage Reference . . . 125
13.3.2.4 Ground Reference . . . 125
13.3.3 Input Selection . . . 125
13.3.3.1 Multiplexer Channel Selection . . . 126
13.3.4 Gain Setting . . . 126
13.3.5 Initiating Conversions . . . 126
13.3.6 Input Tracking . . . 127
13.3.7 Burst Mode . . . 129
13.3.8 8-Bit Mode . . . 129
13.3.9 12-Bit Mode . . . 130
13.3.10 Output Formatting . . . 131
13.3.11 Window Comparator . . . 132
13.3.12 Temperature Sensor . . . 134
13.3.12.1 Temperature Sensor Calibration . . . 134
13.4 ADC0 Control Registers . . . 135
13.4.1 ADC0CN0: ADC0 Control 0 . . . 135
13.4.2 ADC0CF: ADC0 Configuration . . . 136
13.4.3 ADC0AC: ADC0 Accumulator Configuration . . . 137
13.4.4 ADC0PWR: ADC0 Power Control . . . 138
13.4.5 ADC0TK: ADC0 Burst Mode Track Time . . . 138
13.4.6 ADC0H: ADC0 Data Word High Byte . . . 139
13.4.7 ADC0L: ADC0 Data Word Low Byte . . . 139
13.4.8 ADC0GTH: ADC0 Greater-Than High Byte . . . 139
13.4.9 ADC0GTL: ADC0 Greater-Than Low Byte . . . 140
13.4.10 ADC0LTH: ADC0 Less-Than High Byte . . . 140
13.4.11 ADC0LTL: ADC0 Less-Than Low Byte . . . 140
13.4.12 ADC0MX: ADC0 Multiplexer Selection . . . 141
13.4.13 REF0CN: Voltage Reference Control . . . 142
13.4.14 TOFFH: Temperature Sensor Offset High . . . 142
13.4.15 TOFFL: Temperature Sensor Offset Low . . . 143

**14. Programmable Current Reference (IREF0) . . . 144**

14.1 Introduction . . . 144
14.2 Features . . . 144
14.3 Functional Description . . . 144
14.3.1 Overview . . . 144
14.3.2 PWM Enhanced Mode . . . 144
14.4 IREF0 Control Registers . . . 145

14.4.1 IREF0CN0: Current Reference Control 0 . . . 145
14.4.2 IREF0CF: Current Reference Configuration . . . 146

**15. Comparator (CMP0) . . . 147**

15.1 Introduction . . . 147
15.2 Features . . . 147
15.3 Functional Description . . . 147
15.3.1 Response Time and Supply Current . . . 147
15.3.2 Hysteresis . . . 148
15.3.3 Input Selection . . . 148
15.3.3.1 Multiplexer Channel Selection . . . 149
15.3.4 Output Routing . . . 149
15.4 CMP0 Control Registers . . . 150
15.4.1 CMP0CN0: Comparator 0 Control 0 . . . 150
15.4.2 CMP0MD: Comparator 0 Mode . . . 151
15.4.3 CMP0MX: Comparator 0 Multiplexer Selection . . . 152

**16. Capacitive Sense (CS0) . . . 153**

16.1 Introduction . . . 153
16.2 Features . . . 153
16.3 Functional Description . . . 154
16.3.1 Port I/O Configuration . . . 154
16.3.1.1 Multiplexer Channel Selection . . . 154
16.3.2 Initializing the Capacitive Sensing Peripheral . . . 154
16.3.3 Start-of-Conversion Sources . . . 155
16.3.4 Multiple Channel Enable . . . 155
16.3.5 Gain Adjustment . . . 155
16.3.6 Using CS0 with Low Power Modes . . . 155
16.3.7 Automatic Scanning . . . 156
16.3.8 Comparator . . . 158
16.3.9 Conversion Accumulator . . . 158
16.3.10 Pin Monitor . . . 159
16.3.11 Other Considerations . . . 159
16.4 CS0 Control Registers . . . 162
16.4.1 CS0CN0: Capacitive Sense 0 Control . . . 162
16.4.2 CS0CF: Capacitive Sense 0 Configuration . . . 164
16.4.3 CS0DH: Capacitive Sense 0 Data High Byte . . . 165
16.4.4 CS0DL: Capacitive Sense 0 Data Low Byte . . . 165
16.4.5 CS0SCAN0: Capacitive Sense 0 Channel Scan Mask 0 . . . 165
16.4.6 CS0SCAN1: Capacitive Sense 0 Channel Scan Mask 1 . . . 166
16.4.7 CS0SS: Capacitive Sense 0 Auto Scan Start Channel . . . 166
16.4.8 CS0SE: Capacitive Sense 0 Auto Scan End Channel . . . 166
16.4.9 CS0THH: Capacitive Sense 0 Comparator Threshold High Byte . . . 167
16.4.10 CS0THL: Capacitive Sense 0 Comparator Threshold Low Byte . . . 167
16.4.11 CS0MD1: Capacitive Sense 0 Mode 1 . . . 168
16.4.12 CS0MD2: Capacitive Sense 0 Mode 2 . . . 170
16.4.13 CS0MD3: Capacitive Sense 0 Mode 3 . . . 171
16.4.14 CS0PM: Capacitive Sense 0 Pin Monitor . . . 172

16.4.15 CS0MX: Capacitive Sense 0 Mux Channel Select . . . 172

**17. Cyclic Redundancy Check (CRC0) . . . 173**

17.1 Introduction . . . 173

17.2 Features . . . 173

17.3 Functional Description . . . 174
17.3.1 16-bit CRC Algorithm . . . 174
17.3.2 Using the CRC on a Data Stream . . . 175
17.3.3 Using the CRC to Check Code Memory . . . 175
17.3.4 Bit Reversal . . . 175

17.4 CRC0 Control Registers . . . 176
17.4.1 CRC0CN0: CRC0 Control 0 . . . 176
17.4.2 CRC0IN: CRC0 Data Input . . . 176
17.4.3 CRC0DAT: CRC0 Data Output . . . 177
17.4.4 CRC0AUTO: CRC0 Automatic Control . . . 177
17.4.5 CRC0CNT: CRC0 Automatic Flash Sector Count . . . 178
17.4.6 CRC0FLIP: CRC0 Bit Flip . . . 178

**18. Programmable Counter Array (PCA0) . . . 179**

18.1 Introduction . . . 179

18.2 Features . . . 179

18.3 Functional Description . . . 180
18.3.1 Counter / Timer . . . 180
18.3.2 Interrupt Sources . . . 180
18.3.3 Capture/Compare Modules . . . 180
18.3.4 Edge-Triggered Capture Mode . . . 182
18.3.5 Software Timer (Compare) Mode . . . 183
18.3.6 High-Speed Output Mode . . . 184
18.3.7 Frequency Output Mode . . . 185
18.3.8 PWM Waveform Generation . . . 185
18.3.8.1 8 to 11-Bit PWM Modes . . . 187
18.3.8.2 16-Bit PWM Mode . . . 187
18.3.9 Watchdog Timer Mode . . . 188

18.4 PCA0 Control Registers . . . 190
18.4.1 PCA0CN0: PCA Control 0 . . . 190
18.4.2 PCA0MD: PCA Mode . . . 191
18.4.3 PCA0PWM: PCA PWM Configuration . . . 193
18.4.4 PCA0L: PCA Counter/Timer Low Byte . . . 194
18.4.5 PCA0H: PCA Counter/Timer High Byte . . . 194
18.4.6 PCA0CPM0: PCA Channel 0 Capture/Compare Mode . . . 195
18.4.7 PCA0CPL0: PCA Channel 0 Capture Module Low Byte . . . 196
18.4.8 PCA0CPH0: PCA Channel 0 Capture Module High Byte . . . 196
18.4.9 PCA0CPM1: PCA Channel 1 Capture/Compare Mode . . . 197
18.4.10 PCA0CPL1: PCA Channel 1 Capture Module Low Byte . . . 198
18.4.11 PCA0CPH1: PCA Channel 1 Capture Module High Byte . . . 198
18.4.12 PCA0CPM2: PCA Channel 2 Capture/Compare Mode . . . 199
18.4.13 PCA0CPL2: PCA Channel 2 Capture Module Low Byte . . . 200
18.4.14 PCA0CPH2: PCA Channel 2 Capture Module High Byte . . . 200

**19. Serial Peripheral Interface (SPI0) . . . . . 201**
19.1 Introduction . . . . . 201
19.2 Features . . . . . 201
19.3 Functional Description . . . . . 202
19.3.1 Signals . . . . . 202
19.3.2 Master Mode Operation . . . . . 203
19.3.3 Slave Mode Operation . . . . . 203
19.3.4 Clock Phase and Polarity . . . . . 204
19.3.5 Basic Data Transfer . . . . . 205
19.3.6 SPI Timing Diagrams . . . . . 206
19.4 SPI0 Control Registers . . . . . 209
19.4.1 SPI0CFG: SPI0 Configuration . . . . . 209
19.4.2 SPI0CN0: SPI0 Control . . . . . 211
19.4.3 SPI0CKR: SPI0 Clock Rate . . . . . 212
19.4.4 SPI0DAT: SPI0 Data . . . . . 212
**20. System Management Bus / I2C (SMB0) . . . . . 213**
20.1 Introduction . . . . . 213
20.2 Features . . . . . 213
20.3 Functional Description . . . . . 213
20.3.1 Supporting Documents . . . . . 213
20.3.2 SMBus Protocol . . . . . 214
20.3.3 Configuring the SMBus Module . . . . . 216
20.3.4 Operational Modes . . . . . 220
20.4 SMB0 Control Registers . . . . . 228
20.4.1 SMB0CF: SMBus 0 Configuration . . . . . 228
20.4.2 SMB0CN0: SMBus 0 Control . . . . . 229
20.4.3 SMB0ADR: SMBus 0 Slave Address . . . . . 230
20.4.4 SMB0ADM: SMBus 0 Slave Address Mask . . . . . 231
20.4.5 SMB0DAT: SMBus 0 Data . . . . . 231
**21. Timers (Timer0, Timer1, Timer2, and Timer3) . . . . . 232**
21.1 Introduction . . . . . 232
21.2 Features . . . . . 232
21.3 Functional Description . . . . . 233
21.3.1 System Connections . . . . . 233
21.3.2 Timer 0 and Timer 1 . . . . . 233
21.3.2.1 Operational Modes . . . . . 234
21.3.3 Timer 2 and Timer 3 . . . . . 237
21.3.3.1 16-bit Timer with Auto-Reload . . . . . 239
21.3.3.2 8-bit Timers with Auto-Reload (Split Mode) . . . . . 240
21.3.3.3 Capture Mode . . . . . 241
21.4 Timer 0, 1, 2 and 3, Control Registers . . . . . 242
21.4.1 CKCON0: Clock Control 0 . . . . . 242
21.4.2 TCON: Timer 0/1 Control . . . . . 244
21.4.3 TMOD: Timer 0/1 Mode . . . . . 245
21.4.4 TL0: Timer 0 Low Byte . . . . . 246

21.4.5 TL1: Timer 1 Low Byte . . . 246
21.4.6 TH0: Timer 0 High Byte . . . 247
21.4.7 TH1: Timer 1 High Byte . . . 247
21.4.8 TMR2CN0: Timer 2 Control 0 . . . 248
21.4.9 TMR2RLL: Timer 2 Reload Low Byte . . . 249
21.4.10 TMR2RLH: Timer 2 Reload High Byte . . . 249
21.4.11 TMR2L: Timer 2 Low Byte . . . 249
21.4.12 TMR2H: Timer 2 High Byte . . . 250
21.4.13 TMR3CN0: Timer 3 Control 0 . . . 251
21.4.14 TMR3RLL: Timer 3 Reload Low Byte . . . 252
21.4.15 TMR3RLH: Timer 3 Reload High Byte . . . 252
21.4.16 TMR3L: Timer 3 Low Byte . . . 252
21.4.17 TMR3H: Timer 3 High Byte . . . 253

**22. Universal Asynchronous Receiver/Transmitter 0 (UART0) . . . 254**

22.1 Introduction . . . 254
22.2 Features . . . 254
22.3 Functional Description . . . 255
22.3.1 Baud Rate Generation . . . 255
22.3.2 Data Format . . . 255
22.3.3 Data Transfer . . . 256
22.3.4 Multiprocessor Communications . . . 256
22.4 UART0 Control Registers . . . 257
22.4.1 SCON0: UART0 Serial Port Control . . . 257
22.4.2 SBUF0: UART0 Serial Port Data Buffer . . . 258

**23. C2 Debug and Programming Interface . . . 259**

23.1 Introduction . . . 259
23.2 Features . . . 259
23.3 Pin Sharing . . . 259
23.4 C2 Interface Registers . . . 260
23.4.1 C2ADD: C2 Address . . . 260
23.4.2 C2DEVID: C2 Device ID . . . 260
23.4.3 C2REVID: C2 Revision ID . . . 260
23.4.4 C2FPCTL: C2 Flash Programming Control . . . 261
23.4.5 C2FPDAT: C2 Flash Programming Data . . . 261

**Table of Contents . . . 262**




## Simpilcity Studio

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

*www.silabs.com/simplicity*

**MCU Portfolio**
*www.silabs.com/mcu*

**SW/HW**
*www.silabs.com/simplicity*

**Quality**
*www.silabs.com/quality*

**Support and Community**
*community.silabs.com*

**Disclaimer**
Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products must not be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are generally not intended for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

**Trademark Information**
Silicon Laboratories Inc., Silicon Laboratories, Silicon Labs, SiLabs and the Silicon Labs logo, CMEMS®, EFM, EFM32, EFR, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZMac®, EZRadio®, EZRadioPRO®, DSPLL®, ISOmodem ®, Precision32®, ProSLIC®, SiPHY®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.

SILICON LABS®

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

**http://www.silabs.com**