CY7C63310
CY7C638xx

enCoRe™ II
Low-Speed USB Peripheral Controller

1.0 Features

- enCoRe^{TM} II USB—“enhanced Component Reduction”
  - — Crystalless oscillator with support for an external clock. The internal oscillator eliminates the need for an external crystal or resonator
  - — Two internal 3.3V regulators and internal USB pull-up resistor
  - — Configurable IO for real-world interface without external components
- USB Specification Compliance
  - — Conforms to USB Specification, Version 2.0
  - — Conforms to USB HID Specification, Version 1.1
  - — Supports one Low-Speed USB device address
  - — Supports one control endpoint and two data endpoints
  - — Integrated USB transceiver with dedicated 3.3V regulator for USB signalling and D- pull up.
- Enhanced 8-bit microcontroller
  - — Harvard architecture
  - — M8C CPU speed can be up to 24 MHz or sourced by an external clock signal
- Internal memory
  - — Up to 256 bytes of RAM
  - — Up to eight Kbytes of Flash including EEROM emulation
- Interface can autoconfigure to operate as PS/2 or USB
  - — No external components for switching between PS/2 and USB modes
  - — No GPIO pins needed to manage dual-mode capability
- Low power consumption
  - — Typically 10 mA at 6 MHz
  - — 10 μA sleep
- In-system re-programmability
  - — Allows easy firmware update
- General purpose I/O ports
  - — Up to 20 General Purpose I/O (GPIO) pins
  - — High current drive on GPIO pins. Configurable 8- or 50-mA/pin current sink on designated pins
  - — Each GPIO port supports high-impedance inputs, configurable pull up, open drain output, CMOS/TTL inputs, and CMOS output
  - — Maskable interrupts on all I/O pins
- A dedicated 3.3V regulator for the USB PHY. Aids in signalling and D-line pull-up
- 125 mA 3.3V voltage regulator can power external 3.3V devices
- 3.3V I/O pins
  - — 4 I/O pins with 3.3V logic levels
  - — Each 3.3V pin supports high-impedance input, internal pull up, open drain output or traditional CMOS output
- SPI serial communication
  - — Master or slave operation
  - — Configurable up to 4 Mbit/second transfers in the master mode
  - — Supports half duplex single data line mode for optical sensors
- 2-channel 8-bit or 1-channel 16-bit capture timer registers. Capture timer registers store both rising and falling edge times
  - — Two registers each for two input pins
  - — Separate registers for rising and falling edge capture
  - — Simplifies interface to RF inputs for wireless applications
- Internal low-power wake-up timer during suspend mode
  - — Periodic wake-up with no external components
- 12-bit Programmable Interval Timer with interrupts
- Advanced development tools based on Cypress MicroSystems PSoC™ tools
- Watchdog timer (WDT)
- Low-voltage detection with user-configurable threshold voltages
- Operating voltage from 4.0V to 5.5VDC
- Operating temperature from 0–70°C
- Available in 16/18-pin PDIP, 16/18/24-pin SOIC, 24-pin QSOP and 32-lead QFN packages
- Industry standard programmer support

1.1 Applications

The CY7C63310/CY7C638xx is targeted for the following applications:

- PC HID devices
  - — Mice (optomechanical, optical, trackball)
- Gaming
  - — Joysticks
  - — Game pad
- General-purpose
  - — Barcode scanners
  - — POS terminal
  - — Consumer electronics
  - — Toys
  - — Remote controls
  - — Security dongles

2.0 Introduction

Cypress has reinvented its leadership position in the low-speed USB market with a new family of innovative microcontrollers. Introducing enCoRe II USB — “enhanced Component Reduction.” Cypress has leveraged its design expertise in USB solutions to advance its family of low-speed USB microcontrollers, which enable peripheral developers to design new products with a minimum number of components. The enCoRe II USB technology builds on to the enCoRe family. The enCoRe family has an integrated oscillator that eliminates the external crystal or resonator, reducing overall cost. Also integrated into this chip are other external components commonly found in low-speed USB applications such as pull-up resistors, wake-up circuitry, and a 3.3V regulator. All of this reduces the overall system cost.

The enCoRe II is an 8-bit Flash-programmable microcontroller with integrated low-speed USB interface. The instruction set has been optimized specifically for USB and PS/2 operations, although the microcontrollers can be used for a variety of other embedded applications.

The enCoRe II features up to 20 general-purpose I/O (GPIO) pins to support USB, PS/2 and other applications. The I/O pins are grouped into four ports (Port 0 to 3). The pins on Port 0 and Port 1 may each be configured individually while the pins on Ports 2 and 3 may only be configured as a group. Each GPIO port supports high-impedance inputs, configurable pull up, open drain output, CMOS/TTL inputs, and CMOS output with up to five pins that support programmable drive strength of up to 50 mA sink current. GPIO Port 1 features four pins that interface at a voltage level of 3.3 volts. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. Each GPIO port has its own GPIO interrupt vector; in addition GPIO Port 0 has three dedicated pins that have independent interrupt vectors (P0.2 - P0.4).

The enCoRe II features an internal oscillator. With the presence of USB traffic, the internal oscillator can be set to precisely tune to USB timing requirements (24 MHz ±1.5%). Optionally, an external 12 MHz or 24 MHz clock can be used to provide a higher precision reference for USB operation. The clock generator provides the 12 MHz and 24 MHz clocks that remain internal to the microcontroller. The enCoRe II also has a 12-bit programmable interval timer and a 16-bit Free-Running Timer with Capture Timer registers. In addition, the enCoRe II includes a Watchdog timer and a vectored interrupt controller

The enCoRe II has up to eight Kbytes of Flash for user’s code and up to 256 bytes of RAM for stack space and user variables.

The Power-on reset circuit detects logic when power is applied to the device, resets the logic to a known state, and begins executing instructions at Flash address 0x0000. When power falls below a programmable trip voltage generates reset or may be configured to generate interrupt. There is a Low-voltage detect circuit that detects when V_{CC} drops below a programmable trip voltage. It may be configurable to generate an LVD interrupt to inform the processor about the low-voltage event. POR and LVD share the same interrupt. There is no separate interrupt for each. The Watchdog timer can be used to ensure the firmware never gets stalled in an infinite loop.

The microcontroller supports 22 maskable interrupts in the vectored interrupt controller. Interrupt sources include a USB bus reset, LVR/POR, a programmable interval timer, a 1.024-ms output from the Free Running Timer, three USB endpoints, two capture timers, four GPIO Ports, three Port 0 pins, two SPI, a 16-bit free running timer wrap, an internal sleep timer, and a bus active interrupt. The sleep timer causes periodic interrupts when enabled. The USB endpoints interrupt after a USB transaction complete is on the bus. The capture timers interrupt whenever a new timer value is saved due to a selected GPIO edge event. A total of seven GPIO interrupts support both TTL or CMOS thresholds. For additional flexibility, on the edge sensitive GPIO pins, the interrupt polarity is programmable to be either rising or falling.

The free-running 16-bit timer provides two interrupt sources: the 1.024 ms outputs and the free running counter wrap interrupt. The programmable interval timer can provide up to 1 µsec resolution and provides an interrupt everytime it expires. These timers can be used to measure the duration of an event under firmware control by reading the desired timer at the start and at the end of an event, then calculating the difference between the two values. The two 8-bit capture timer registers save a programmable 8-bit range of the free-running timer when a GPIO edge occurs on the two capture pins (P0.5, P0.6). The two 8-bit captures can be ganged into a single 16-bit capture.

The enCoRe II includes an integrated USB serial interface engine (SIE) that allows the chip to easily interface to a USB host. The hardware supports one USB device address with three endpoints.

The USB D+ and D– pins can optionally be used as PS/2 SCLK and SDATA signals so that products can be designed to respond to either USB or PS/2 modes of operation. PS/2 operation is supported with internal 5 KΩ pull-up resistors on P1.0 (D+) and P1.1 (D–) and an interrupt to signal the start of PS/2 activity. In USB mode, the integrated 1.5 KΩ pull-up resistor on D– can be controlled under firmware. No external components are necessary for dual USB and PS/2 systems, and no GPIO pins need to be dedicated to switching between modes.

The enCoRe II supports in-system programming by using the D+ and D– pins as the serial programming mode interface. The programming protocol is not USB.

3.0 Conventions

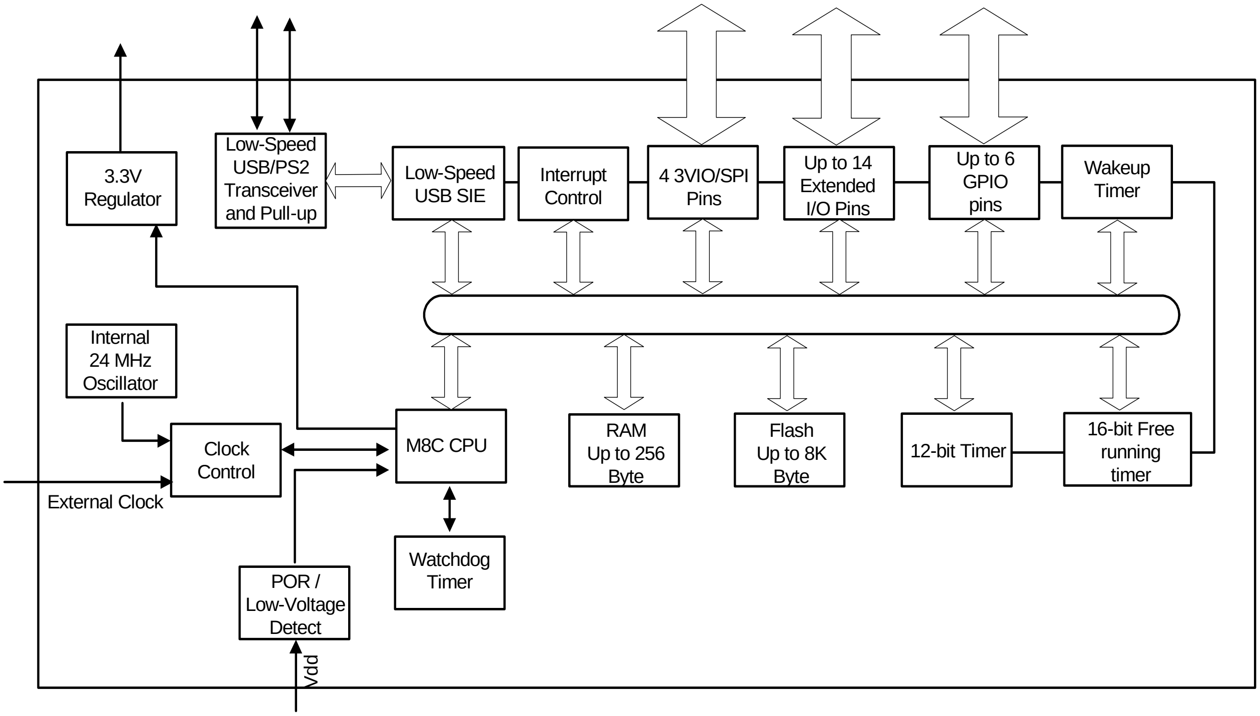
In this document, bit positions in the registers are shaded to indicate which members of the enCoRe II family implement the bits.

□ Available in all enCoRe II family members

▨ CY7C638xx only

4.0 Logic Block Diagram

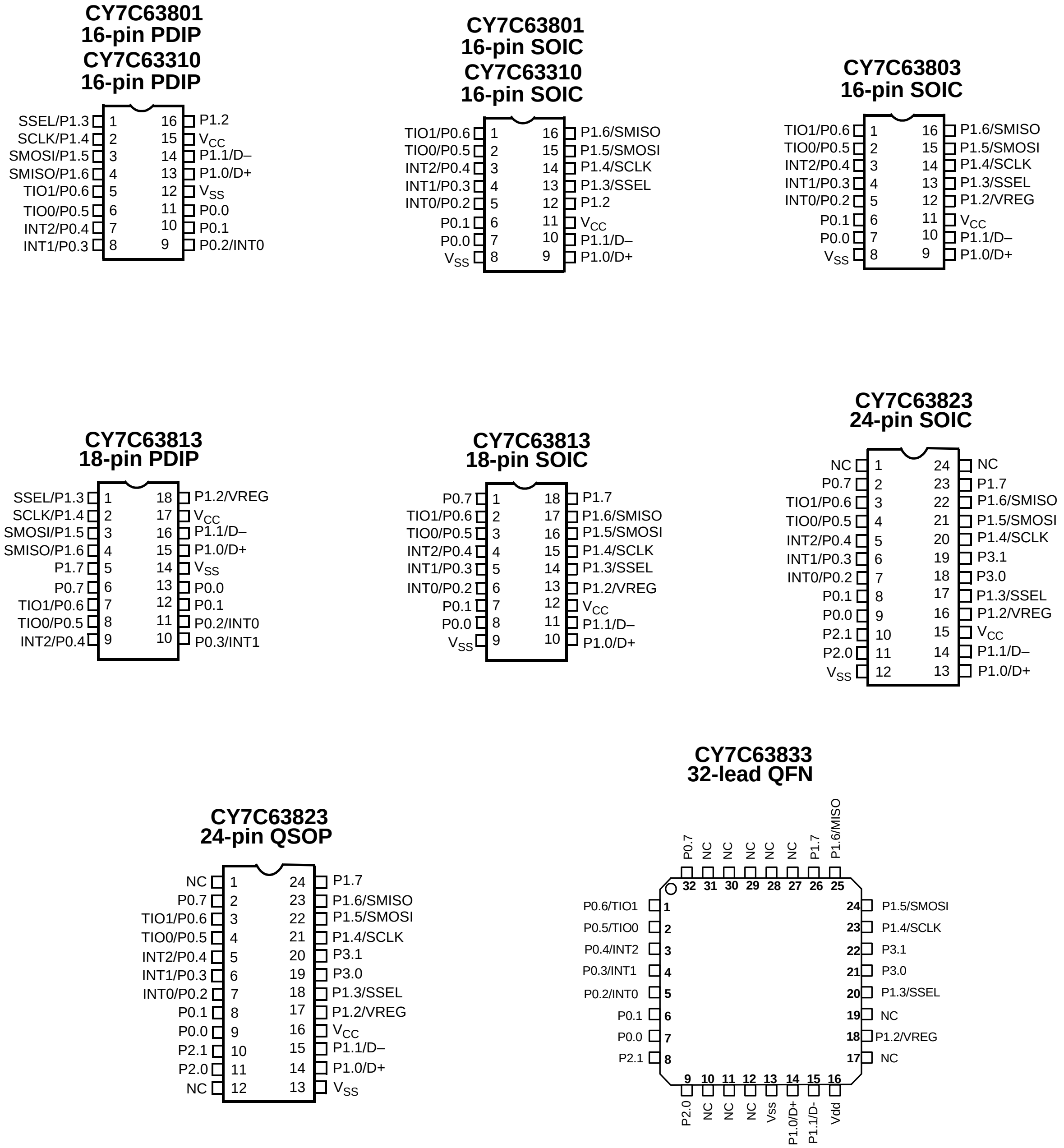
Figure 4-1. CY7C63310/CY7C638xx Block Diagram



5.0 Packages/Pinouts

Figure 5-1. Package Configurations

Top View



CY7C63801 16-pin PDIP
CY7C63310 16-pin PDIP

CY7C63801 16-pin SOIC
CY7C63310 16-pin SOIC

CY7C63803 16-pin SOIC

CY7C63813 18-pin PDIP

CY7C63813 18-pin SOIC

CY7C63823 24-pin SOIC

CY7C63823 24-pin QSOP

CY7C63833 32-lead QFN

5.1 Pinouts Assignments

Table 5-1. Pin Assignments

32 QFN	24 QSOP	24 SOIC	18 SIOC	18 PDIP	16 SOIC	16 PDIP	Name	Description
21	19	18					P3.0	GPIO Port 3 – configured as a group (byte)
22	20	19					P3.1	
9	11	11					P2.0	GPIO Port 2 – configured as a group (byte)
8	10	10					P2.1	
14	14	13	10	15	9	13	P1.0/D+	GPIO Port 1 bit 0/USB D+[1] If this pin is used as a General Purpose output, it will draw current. This pin must be configured as an input to reduce current draw.
15	15	14	11	16	10	14	P1.1/D–	GPIO Port 1 bit 1/USB D–[1] If this pin is used as a General Purpose output, it will draw current. This pin must be configured as an input to reduce current draw.
18	17	16	13	18	12	16	P1.2/VREG	GPIO Port 1 bit 2—Configured individually. 3.3V if regulator is enabled. (The 3.3V regulator is not available in the CY7C63310 and CY7C63801.) A 1-μF min, 2-μF max capacitor is required on Vreg output.
20	18	17	14	1	13	1	P1.3/SSEL	GPIO Port 1 bit 3—Configured individually. Alternate function is SSEL signal of the SPI bus TTL voltage thresholds. Although Vreg is not available with the CY7C63310, 3.3V I/O is still available.
23	21	20	15	2	14	2	P1.4/SCLK	GPIO Port 1 bit 4—Configured individually. Alternate function is SCLK signal of the SPI bus TTL voltage thresholds. Although Vreg is not available with the CY7C63310, 3.3V I/O is still available.
24	22	21	16	3	15	3	P1.5/SMOSI	GPIO Port 1 bit 5—Configured individually. Alternate function is SMOSI signal of the SPI bus TTL voltage thresholds. Although Vreg is not available with the CY7C63310, 3.3V I/O is still available.
25	23	22	17	4	16	4	P1.6/SMISO	GPIO Port 1 bit 6—Configured individually. Alternate function is SMISO signal of the SPI bus TTL voltage thresholds. Although Vreg is not available with the CY7C63310, 3.3V I/O is still available.
26	24	23	18	5			P1.7	GPIO Port 1 bit 7—Configured individually. TTL voltage threshold.
7	9	9	8	13	7	11	P0.0	GPIO Port 0 bit 0—Configured individually. On CY7C638xx and CY7C63310, external clock input when configured as Clock In.
6	8	8	7	12	6	10	P0.1	GPIO Port 0 bit 1—Configured individually On CY7C638xx and CY7C63310, clock output when configured as Clock Out.
5	7	7	6	11	5	9	P0.2/INT0	GPIO port 0 bit 2—Configured individually Optional rising edge interrupt INT0
4	6	6	5	10	4	8	P0.3/INT1	GPIO port 0 bit 3—Configured individually Optional rising edge interrupt INT1
3	5	5	4	9	3	7	P0.4/INT2	GPIO port 0 bit 4—Configured individually Optional rising edge interrupt INT2

Note

[1] P1.0(D+) and P1.1(D-) pins must be in I/O mode when used as GPIO and in I_{sb} mode.

Table 5-1. Pin Assignments (continued)

32 QFN	24 QSOP	24 SOIC	18 SIOC	18 PDIP	16 SOIC	16 PDIP	Name	Description
2	4	4	3	8	2	6	P0.5/TIO0	GPIO port 0 bit 5—Configured individually Alternate function Timer capture inputs or Timer output TIO0
1	3	3	2	7	1	5	P0.6/TIO1	GPIO port 0 bit 6—Configured individually Alternate function Timer capture inputs or Timer output TIO1
32	2	2	1	6			P0.7	GPIO port 0 bit 7—Configured individually Not present in the 16 pin PDIP or SOIC package
10	1	1					NC	No connect
11	12	24					NC	No connect
12							NC	No connect
17							NC	No connect
19							NC	No connect
27							NC	No connect
28							NC	No connect
29							NC	No connect
30							NC	No connect
31							NC	No connect
16	16	15	12	17	11	15	Vcc	Supply
13	13	12	9	14	8	12	V_{SS}	Ground

6.0 CPU Architecture

This family of microcontrollers is based on a high performance, 8-bit, Harvard-architecture microprocessor. Five registers control the primary operation of the CPU core. These registers are affected by various instructions, but are not directly accessible through the register space by the user.

Table 6-1. CPU Registers and Register Names

Register	Register Name
Flags	CPU_F
Program Counter	CPU_PC
Accumulator	CPU_A
Stack Pointer	CPU_SP
Index	CPU_X

The 16-bit Program Counter Register (CPU_PC) allows for direct addressing of the full eight Kbytes of program memory space.

The Accumulator Register (CPU_A) is the general-purpose register that holds the results of instructions that specify any of the source addressing modes.

The Index Register (CPU_X) holds an offset value that is used in the indexed addressing modes. Typically, this is used to address a block of data within the data memory space.

The Stack Pointer Register (CPU_SP) holds the address of the current top-of-stack in the data memory space. It is affected by the PUSH, POP, LCALL, CALL, RETI, and RET instructions, which manage the software stack. It can also be affected by the SWAP and ADD instructions.

The Flag Register (CPU_F) has three status bits: Zero Flag bit [1]; Carry Flag bit [2]; Supervisory State bit [3]. The Global Interrupt Enable bit [0] is used to globally enable or disable interrupts. The user cannot manipulate the Supervisory State status bit [3]. The flags are affected by arithmetic, logic, and shift operations. The manner in which each flag is changed is dependent upon the instruction being executed (i.e., AND, OR, XOR). See *Table 8-1*.

7.0 CPU Registers

7.1 Flags Register

The Flags Register can only be set or reset with logical instruction.

Table 7-1. CPU Flags Register (CPU_F) [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved			XIO	Super	Carry	Zero	Global IE
Read/Write	–	–	–	R/W	R	RW	RW	RW
Default	0	0	0	0	0	0	1	0

Bit [7:5]: Reserved

Bit 4: XIO

Set by the user to select between the register banks

0 = Bank 0

1 = Bank 1

Bit 3: Super

Indicates whether the CPU is executing user code or Supervisor Code. (This code cannot be accessed directly by the user)

0 = User Code

1 = Supervisor Code

Bit 2: Carry

Set by CPU to indicate whether there has been a carry in the previous logical/arithmetic operation

0 = No Carry

1 = Carry

Bit 1: Zero

Set by CPU to indicate whether there has been a zero result in the previous logical/arithmetic operation

0 = Not Equal to Zero

1 = Equal to Zero

Bit 0: Global IE

Determines whether all interrupts are enabled or disabled

0 = Disabled

1 = Enabled

Note: CPU_F register is only readable with explicit register address 0xF7. The *OR F, expr and AND F, expr* instructions must be used to set and clear the CPU_F bits

Table 7-2. CPU Accumulator Register (CPU_A)

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	CPU Accumulator [7:0]							
Read/Write	–	–	–	–	–	–	–	–
Default	0	0	0	0	0	0	0	0

Bit [7:0]: CPU Accumulator [7:0]

8-bit data value holds the result of any logical/arithmetic instruction that uses a source addressing mode

Table 7-3. CPU X Register (CPU_X)

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	X [7:0]							
Read/Write	–	–	–	–	–	–	–	–
Default	0	0	0	0	0	0	0	0

Bit [7:0]: X [7:0]

8-bit data value holds an index for any instruction that uses an indexed addressing mode

Table 7-4. CPU Stack Pointer Register (CPU_SP)

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Stack Pointer [7:0]							
Read/Write	–	–	–	–	–	–	–	–
Default	0	0	0	0	0	0	0	0

Bit [7:0]: Stack Pointer [7:0]

8-bit data value holds a pointer to the current top-of-stack

Table 7-5. CPU Program Counter High Register (CPU_PCH)

Bit #	7	6	5	4	3	2	1	0
Field	Program Counter [15:8]							
Read/Write	–	–	–	–	–	–	–	–
Default	0	0	0	0	0	0	0	0

Bit [7:0]: Program Counter [15:8]
8-bit data value holds the higher byte of the program counter

Table 7-6. CPU Program Counter Low Register (CPU_PCL)

Bit #	7	6	5	4	3	2	1	0
Field	Program Counter [7:0]							
Read/Write	–	–	–	–	–	–	–	–
Default	0	0	0	0	0	0	0	0

Bit [7:0]: Program Counter [7:0]
8-bit data value holds the lower byte of the program counter

7.2 Addressing Modes

7.2.1 Source Immediate

The result of an instruction using this addressing mode is placed in the A register, the F register, the SP register, or the X register, which is specified as part of the instruction opcode. Operand 1 is an immediate value that serves as a source for the instruction. Arithmetic instructions require two sources; the second one is the A or the X register specified in the opcode. Instructions using this addressing mode are two bytes in length.

Table 7-7. Source Immediate

Opcode	Operand 1
Instruction	Immediate Value

Examples

ADD	A,	7	;In this case, the immediate value; of 7 is added with the Accumulator,; and the result is placed in the; Accumulator.
MOV	X,	8	;In this case, the immediate value; of 8 is moved to the X register.
AND	F,	9	;In this case, the immediate value; of 9 is logically ANDed with the F; register and the result is placed; in the F register.

7.2.2 Source Direct

The result of an instruction using this addressing mode is placed in either the A register or the X register, which is specified as part of the instruction opcode. Operand 1 is an address that points to a location in either the RAM memory space or the register space that is the source for the instruction. Arithmetic instructions require two sources; the second source is the A register or X register specified in the opcode. Instructions using this addressing mode are two bytes in length.

Table 7-8. Source Direct

Opcode	Operand 1
Instruction	Source Address

Examples

ADD	A,	[7]	;In this case, the; value in; the RAM memory location at; address 7 is added with the; Accumulator, and the result; is placed in the Accumulator.
MOV	X,	REG[8]	;In this case, the value in; the register space at address; 8 is moved to the X register.

7.2.3 Source Indexed

The result of an instruction using this addressing mode is placed in either the A register or the X register, which is specified as part of the instruction opcode. Operand 1 is added to the X register forming an address that points to a location in either the RAM memory space or the register space that is the source for the instruction. Arithmetic instructions require two sources; the second source is the A register or X register specified in the opcode. Instructions using this addressing mode are two bytes in length.

Table 7-9. Source Indexed

Opcode	Operand 1
Instruction	Source Index

Examples

ADD	A,	[X+7]	;In this case, the value in; the memory location at; address X + 7 is added with; the Accumulator, and the; result is placed in the; Accumulator.
MOV	X,	REG[X+8]	;In this case, the value in; the register space at; address X + 8 is moved to; the X register.

7.2.4 Destination Direct

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is an address that points to the location of the result. The source for the instruction is either the A register or the X register, which is specified as part of the instruction opcode. Arithmetic instructions require two sources; the second source is the location specified by Operand 1. Instructions using this addressing mode are two bytes in length.

Table 7-10. Destination Direct

Opcode	Operand 1
Instruction	Destination Address

Examples

```
ADD     [7],        A       ;In this case, the value in; the memory
                            location at; address 7 is added with
                            the; Accumulator, and the result; is
                            placed in the memory; location at
                            address 7. The; Accumulator is
                            unchanged.
MOV     REG[8],     A       ;In this case, the Accumulator is
                            moved to the register space location
                            at address 8. The Accumulator is
                            unchanged.
```

7.2.5 Destination Indexed

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is added to the X register forming the address that points to the location of the result. The source for the instruction is the A register. Arithmetic instructions require two sources; the second source is the location specified by Operand 1 added with the X register. Instructions using this addressing mode are two bytes in length.

Table 7-11. Destination Indexed

Opcode	Operand 1
Instruction	Destination Index

Example

```
ADD   [X+7],      A     ;In this case, the value in the; memory
                        location at address X+7; is added with the
                        Accumulator,; and the result is placed in;
                        the memory location at address; x+7. The
                        Accumulator is; unchanged.
```

7.2.6 Destination Direct Source Immediate

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is the address of the result. The source for the instruction is Operand 2, which is an immediate value. Arithmetic instructions require two sources; the second source is the location specified by Operand 1. Instructions using this addressing mode are three bytes in length.

Table 7-12. Destination Direct Source Immediate

Opcode	Operand 1	Operand 2
Instruction	Destination Address	Immediate Value

Examples

```
ADD   [7],        5     ;In this case, value in the memory location
                        at address 7 is added to the immediate
                        value of 5, and the result is placed in the
                        memory location at address 7.
MOV   REG[8],     6     ;In this case, the immediate; value of 6 is
                        moved into the; register space location at;
                        address 8.
```

7.2.7 Destination Indexed Source Immediate

The result of an instruction using this addressing mode is placed within either the RAM memory space or the register space. Operand 1 is added to the X register to form the address of the result. The source for the instruction is Operand 2, which is an immediate value. Arithmetic instructions require two sources; the second source is the location specified by Operand 1 added with the X register. Instructions using this addressing mode are three bytes in length.

Table 7-13. Destination Indexed Source Immediate

Opcode	Operand 1	Operand 2
Instruction	Destination Index	Immediate Value

Examples

```
ADD   [X+7],        5     ;In this case, the value in; the mem-
                          ory location at; address X+7 is
                          added with; the immediate value of
                          5,; and the result is placed; in the
                          memory location at; address X+7.
MOV   REG[X+8],     6     ;In this case, the immediate value of
                          6 is moved; into the location in the;
                          register space at; address X+8.
```

7.2.8 Destination Direct Source Direct

The result of an instruction using this addressing mode is placed within the RAM memory. Operand 1 is the address of the result. Operand 2 is an address that points to a location in the RAM memory that is the source for the instruction. This addressing mode is only valid on the MOV instruction. The instruction using this addressing mode is three bytes in length.

Table 7-14. Destination Direct Source Direct

Opcode	Operand 1	Operand 2
Instruction	Destination Address	Source Address

Example

```
MOV    [7],     [8]     ;In this case, the value in the; memory loca-
                        tion at address 8 is; moved to the memory
                        location at; address 7.
```

7.2.9 Source Indirect Post Increment

The result of an instruction using this addressing mode is placed in the Accumulator. Operand 1 is an address pointing to a location within the memory space, which contains an address (the indirect address) for the source of the instruction. The indirect address is incremented as part of the instruction execution. This addressing mode is only valid on the MVI instruction. The instruction using this addressing mode is two bytes in length. Refer to the *PSoC Designer: Assembly Language User Guide* for further details on MVI instruction.

Table 7-15. Source Indirect Post Increment

Opcode	Operand 1
Instruction	Source Address Address

Example

```
MVI    A,     [8]     ;In this case, the value in the; memory loca-
                      tion at address 8 is an indirect address. The
                      memory location pointed to by the indirect
                      address is moved into the Accumulator. The
                      indirect address is then incremented.
```

7.2.10 Destination Indirect Post Increment

The result of an instruction using this addressing mode is placed within the memory space. Operand 1 is an address pointing to a location within the memory space, which contains an address (the indirect address) for the destination of the instruction. The indirect address is incremented as part of the instruction execution. The source for the instruction is the Accumulator. This addressing mode is only valid on the MVI instruction. The instruction using this addressing mode is two bytes in length.

Table 7-16. Destination Indirect Post Increment

Opcode	Operand 1
Instruction	Destination Address Address

Example

```
MVI    [8],    A      ;In this case, the value in; the memory
                      location at; address 8 is an indirect;
                      address. The Accumulator is; moved
                      into the memory location pointed to by
                      the indirect address. The indirect;
                      address is then incremented.
```

8.0 Instruction Set Summary

The instruction set is summarized in *Table 8-1* numerically and serves as a quick reference. If more information is needed, the Instruction Set Summary tables are described in detail in the *PSoC Designer Assembly Language User Guide* (available on the www.cypress.com web site).

Table 8-1. Instruction Set Summary Sorted Numerically by Opcode Order[2, 3]

Opcode Hex	Cycles	Bytes	Instruction Format	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags
00	15	1	SSC		2D	8	2	OR [X+expr], A	Z	5A	5	2	MOV [expr], X	
01	4	2	ADD A, expr	C, Z	2E	9	3	OR [expr], expr	Z	5B	4	1	MOV A, X	Z
02	6	2	ADD A, [expr]	C, Z	2F	10	3	OR [X+expr], expr	Z	5C	4	1	MOV X, A	
03	7	2	ADD A, [X+expr]	C, Z	30	9	1	HALT		5D	6	2	MOV A, reg[expr]	Z
04	7	2	ADD [expr], A	C, Z	31	4	2	XOR A, expr	Z	5E	7	2	MOV A, reg[X+expr]	Z
05	8	2	ADD [X+expr], A	C, Z	32	6	2	XOR A, [expr]	Z	5F	10	3	MOV [expr], [expr]	
06	9	3	ADD [expr], expr	C, Z	33	7	2	XOR A, [X+expr]	Z	60	5	2	MOV reg[expr], A	
07	10	3	ADD [X+expr], expr	C, Z	34	7	2	XOR [expr], A	Z	61	6	2	MOV reg[X+expr], A	
08	4	1	PUSH A		35	8	2	XOR [X+expr], A	Z	62	8	3	MOV reg[expr], expr	
09	4	2	ADC A, expr	C, Z	36	9	3	XOR [expr], expr	Z	63	9	3	MOV reg[X+expr], expr	
0A	6	2	ADC A, [expr]	C, Z	37	10	3	XOR [X+expr], expr	Z	64	4	1	ASL A	C, Z
0B	7	2	ADC A, [X+expr]	C, Z	38	5	2	ADD SP, expr		65	7	2	ASL [expr]	C, Z
0C	7	2	ADC [expr], A	C, Z	39	5	2	CMP A, expr	if (A=B) Z=1 if (A<B) C=1	66	8	2	ASL [X+expr]	C, Z
0D	8	2	ADC [X+expr], A	C, Z	3A	7	2	CMP A, [expr]		67	4	1	ASR A	C, Z
0E	9	3	ADC [expr], expr	C, Z	3B	8	2	CMP A, [X+expr]		68	7	2	ASR [expr]	C, Z
0F	10	3	ADC [X+expr], expr	C, Z	3C	8	3	CMP [expr], expr		69	8	2	ASR [X+expr]	C, Z
10	4	1	PUSH X		3D	9	3	CMP [X+expr], expr		6A	4	1	RLC A	C, Z
11	4	2	SUB A, expr	C, Z	3E	10	2	MVI A, [[expr]++]	Z	6B	7	2	RLC [expr]	C, Z
12	6	2	SUB A, [expr]	C, Z	3F	10	2	MVI [[expr]++], A		6C	8	2	RLC [X+expr]	C, Z
13	7	2	SUB A, [X+expr]	C, Z	40	4	1	NOP		6D	4	1	RRC A	C, Z
14	7	2	SUB [expr], A	C, Z	41	9	3	AND reg[expr], expr	Z	6E	7	2	RRC [expr]	C, Z
15	8	2	SUB [X+expr], A	C, Z	42	10	3	AND reg[X+expr], expr	Z	6F	8	2	RRC [X+expr]	C, Z
16	9	3	SUB [expr], expr	C, Z	43	9	3	OR reg[expr], expr	Z	70	4	2	AND F, expr	C, Z
17	10	3	SUB [X+expr], expr	C, Z	44	10	3	OR reg[X+expr], expr	Z	71	4	2	OR F, expr	C, Z
18	5	1	POP A	Z	45	9	3	XOR reg[expr], expr	Z	72	4	2	XOR F, expr	C, Z
19	4	2	SBB A, expr	C, Z	46	10	3	XOR reg[X+expr], expr	Z	73	4	1	CPL A	Z
1A	6	2	SBB A, [expr]	C, Z	47	8	3	TST [expr], expr	Z	74	4	1	INC A	C, Z
1B	7	2	SBB A, [X+expr]	C, Z	48	9	3	TST [X+expr], expr	Z	75	4	1	INC X	C, Z
1C	7	2	SBB [expr], A	C, Z	49	9	3	TST reg[expr], expr	Z	76	7	2	INC [expr]	C, Z
1D	8	2	SBB [X+expr], A	C, Z	4A	10	3	TST reg[X+expr], expr	Z	77	8	2	INC [X+expr]	C, Z
1E	9	3	SBB [expr], expr	C, Z	4B	5	1	SWAP A, X	Z	78	4	1	DEC A	C, Z
1F	10	3	SBB [X+expr], expr	C, Z	4C	7	2	SWAP A, [expr]	Z	79	4	1	DEC X	C, Z
20	5	1	POP X		4D	7	2	SWAP X, [expr]		7A	7	2	DEC [expr]	C, Z
21	4	2	AND A, expr	Z	4E	5	1	SWAP A, SP	Z	7B	8	2	DEC [X+expr]	C, Z
22	6	2	AND A, [expr]	Z	4F	4	1	MOV X, SP		7C	13	3	LCALL	
23	7	2	AND A, [X+expr]	Z	50	4	2	MOV A, expr	Z	7D	7	3	LJMP	
24	7	2	AND [expr], A	Z	51	5	2	MOV A, [expr]	Z	7E	10	1	RETI	C, Z
25	8	2	AND [X+expr], A	Z	52	6	2	MOV A, [X+expr]	Z	7F	8	1	RET	
26	9	3	AND [expr], expr	Z	53	5	2	MOV [expr], A		8x	5	2	JMP	
27	10	3	AND [X+expr], expr	Z	54	6	2	MOV [X+expr], A		9x	11	2	CALL	
28	11	1	ROMX	Z	55	8	3	MOV [expr], expr		Ax	5	2	JZ	
29	4	2	OR A, expr	Z	56	9	3	MOV [X+expr], expr		Bx	5	2	JNZ	
2A	6	2	OR A, [expr]	Z	57	4	2	MOV X, expr		Cx	5	2	JC	
2B	7	2	OR A, [X+expr]	Z	58	6	2	MOV X, [expr]		Dx	5	2	JNC	
2C	7	2	OR [expr], A	Z	59	7	2	MOV X, [X+expr]		Ex	7	2	JACC	
										Fx	13	2	INDEX	Z

Notes

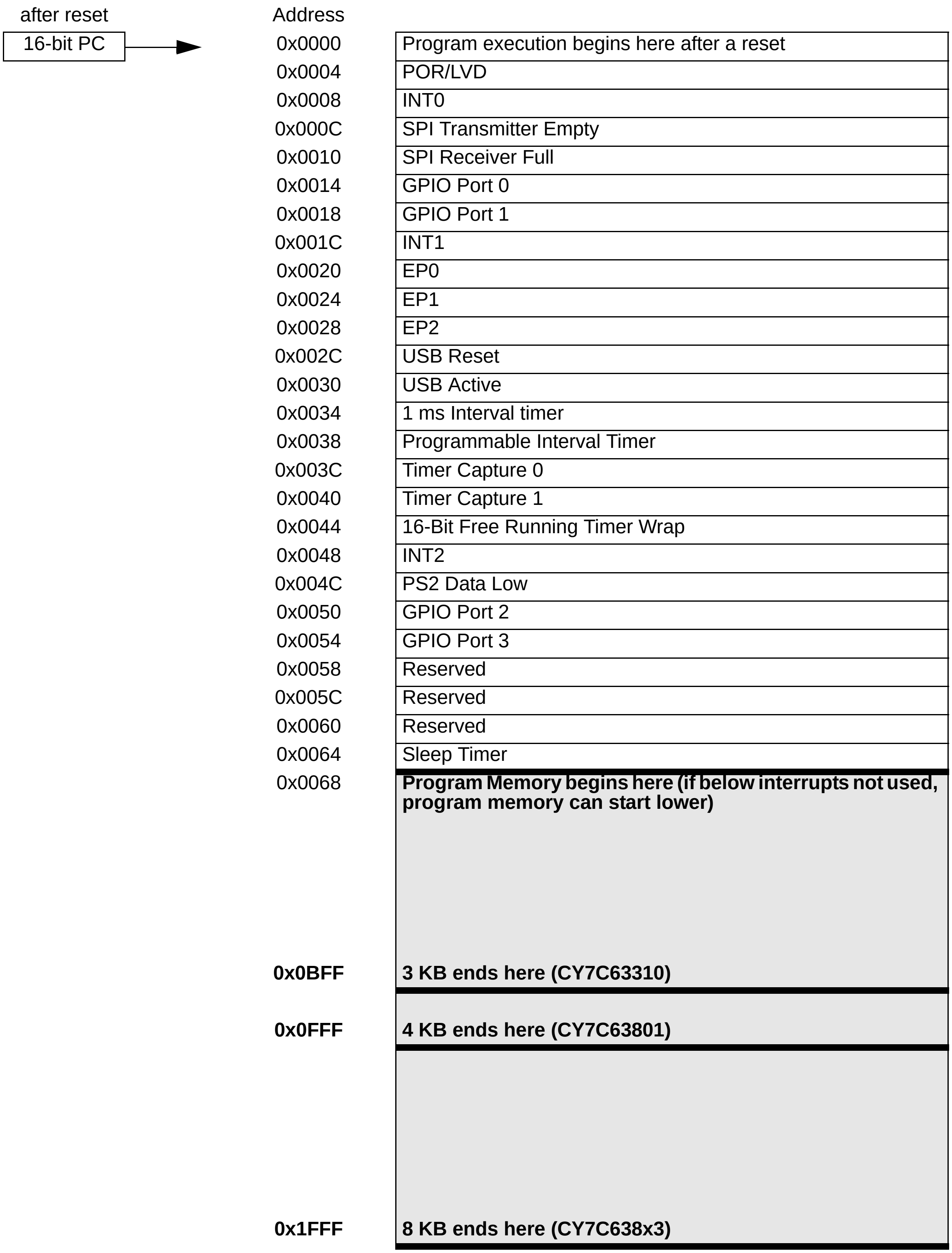
[2] Interrupt routines take 13 cycles before execution resumes at interrupt vector table.

[3] The number of cycles required by an instruction is increased by one for instructions that span 256-byte boundaries in the Flash memory space.

9.0 Memory Organization

9.1 Flash Program Memory Organization

Figure 9-1. Program Memory Space with Interrupt Vector Table

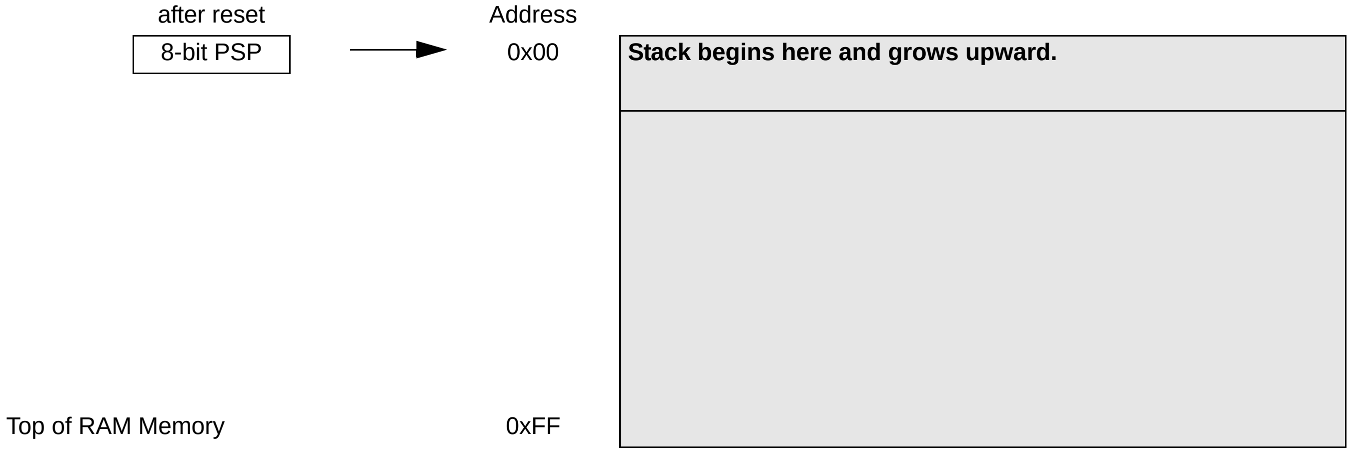


Address	
0x0000	Program execution begins here after a reset
0x0004	POR/LVD
0x0008	INT0
0x000C	SPI Transmitter Empty
0x0010	SPI Receiver Full
0x0014	GPIO Port 0
0x0018	GPIO Port 1
0x001C	INT1
0x0020	EP0
0x0024	EP1
0x0028	EP2
0x002C	USB Reset
0x0030	USB Active
0x0034	1 ms Interval timer
0x0038	Programmable Interval Timer
0x003C	Timer Capture 0
0x0040	Timer Capture 1
0x0044	16-Bit Free Running Timer Wrap
0x0048	INT2
0x004C	PS2 Data Low
0x0050	GPIO Port 2
0x0054	GPIO Port 3
0x0058	Reserved
0x005C	Reserved
0x0060	Reserved
0x0064	Sleep Timer
0x0068	**Program Memory begins here (if below interrupts not used, program memory can start lower)**
0x0BFF	**3 KB ends here (CY7C63310)**
0x0FFF	**4 KB ends here (CY7C63801)**
0x1FFF	**8 KB ends here (CY7C638x3)**

9.2 Data Memory Organization

The CY7C63310/638xx microcontrollers provide up to 256 bytes of data RAM.

Figure 9-2. Data Memory Organization



9.3 Flash

This section describes the Flash block of the enCoRe II. Much of the user-visible Flash functionality including programming and security are implemented in the M8C Supervisory Read Only Memory (SROM). enCoRe II Flash has an endurance of 1000 cycles and a 10 year data retention capability.

9.3.1 Flash Programming and Security

All Flash programming is performed by code in the SROM. The registers that control the Flash programming are only visible to the M8C CPU when it is executing out of SROM. This makes it impossible to read, write or erase the Flash by bypassing the security mechanisms implemented in the SROM.

Customer firmware can only program the Flash via SROM calls. The data or code images can be sourced via any interface with the appropriate support firmware. This type of programming requires a ‘boot-loader’—a piece of firmware resident on the Flash. For safety reasons this boot-loader must not be overwritten during firmware rewrites.

The Flash provides four extra auxiliary rows that are used to hold Flash block protection flags, boot time calibration values, configuration tables, and any device values. The routines for accessing these auxiliary rows are documented in the SROM section. The auxiliary rows are not affected by the device erase function.

9.3.2 In-System Programming

Most designs that include an enCoRe II part will have a USB connector attached to the USB D+/D– pins on the device. These designs require the ability to program or reprogram a part through these two pins alone.

enCoRe II devices enable this type of in-system programming by using the D+ and D– pins as the serial programming mode interface. This allows an external controller to cause the enCoRe II part to enter serial programming mode and then to use the test queue to issue Flash access functions in the SROM. The programming protocol is not USB.

9.4 SROM

The SROM holds code that is used to boot the part, calibrate circuitry, and perform Flash operations. (*Table 9-1* lists the SROM functions.) The functions of the SROM may be accessed in normal user code or operating from Flash. The SROM exists in a separate memory space from user code. The SROM functions are accessed by executing the Supervisory System Call instruction (SSC), which has an opcode of 00h. Prior to executing the SSC the M8C’s accumulator needs to be loaded with the desired SROM function code from *Table 9-1*. Undefined functions will cause a HALT if called from user code. The SROM functions are executing code with calls; therefore, the functions require stack space. With the exception of Reset, all of the SROM functions have a *parameter block* in SRAM that must be configured before executing the SSC. *Table 9-2* lists all possible parameter block variables. The meaning of each parameter, with regards to a specific SROM function, is described later in this chapter.

Table 9-1. SROM Function Codes

Function Code	Function Name	Stack Space
00h	SWBootReset	0
01h	ReadBlock	7
02h	WriteBlock	10
03h	EraseBlock	9
05h	EraseAll	11
06h	TableRead	3
07h	CheckSum	3

Two important variables that are used for all functions are KEY1 and KEY2. These variables are used to help discriminate between valid SSCs and inadvertent SSCs. KEY1 must always have a value of 3Ah, while KEY2 must have the same value as the stack pointer when the SROM function begins execution. This would be the Stack Pointer value when the SSC opcode is executed, plus three. If either of the keys do not match the expected values, the M8C will halt (with the exception of the SWBootReset function). The following code puts the correct value in KEY1 and KEY2. The code starts with a halt, to force the program to jump directly into the setup code and not run into it.

```
halt
SSCOP: mov [KEY1], 3ah
mov X, SP
mov A, X
add A, 3
mov [KEY2], A
```

Table 9-2. SROM Function Parameters

Variable Name	SRAM Address
Key1/Counter/Return Code	0,F8h
Key2/TMP	0,F9h
BlockID	0,FAh
Pointer	0,FBh
Clock	0,FCh
Mode	0,FDh
Delay	0,FEh
PCL	0,FFh

9.4.1 Return Codes

The SROM also features Return Codes and Lockouts.

Return codes aid in the determination of success or failure of a particular function. The return code is stored in KEY1’s position in the parameter block. The CheckSum and TableRead functions do not have return codes because KEY1’s position in the parameter block is used to return other data.

Table 9-3. SROM Return Codes

Return Code	Description
00h	Success
01h	Function not allowed due to level of protection on block.
02h	Software reset without hardware reset.
03h	Fatal error, SROM halted.

Read, write, and erase operations may fail if the target block is read or write protected. Block protection levels are set during device programming.

The EraseAll function overwrites data in addition to leaving the entire user Flash in the erase state. The EraseAll function loops through the number of Flash macros in the product, executing the following sequence: erase, bulk program all zeros, erase. After all the user space in all the Flash macros are erased, a second loop erases and then programs each protection block with zeros.

9.5 SROM Function Descriptions

9.5.1 SWBootReset Function

The SROM function, SWBootReset, is the function that is responsible for transitioning the device from a reset state to running user code. The SWBootReset function is executed whenever the SROM is entered with an M8C accumulator value of 00h: the SRAM parameter block is not used as an input to the function. This will happen, by design, after a hardware reset, because the M8C's accumulator is reset to 00h or when user code executes the SSC instruction with an accumulator value of 00h. The SWBootReset function will not execute when the SSC instruction is executed with a bad key value and a non-zero function code. An enCoRe II device will execute the HALT instruction if a bad value is given for either KEY1 or KEY2.

The SWBootReset function verifies the integrity of the calibration data by way of a 16-bit checksum, before releasing the M8C to run user code.

9.5.2 ReadBlock Function

The ReadBlock function is used to read 64 contiguous bytes from Flash: a block.

The first thing this function does is to check the protection bits and determine if the desired BLOCKID is readable. If read protection is turned on, the ReadBlock function will exit setting the accumulator and KEY2 back to 00h. KEY1 will have a value of 01h, indicating a read failure. If read protection is not enabled, the function will read 64 bytes from the Flash using a ROMX instruction and store the results in SRAM using an MVI instruction. The first of the 64 bytes will be stored in SRAM at the address indicated by the value of the POINTER parameter. When the ReadBlock completes successfully the accumulator, KEY1 and KEY2 will all have a value of 00h.

Table 9-4. ReadBlock Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value, when SSC is executed.
BLOCKID	0,FAh	Flash block number
POINTER	0,FBh	First of 64 addresses in SRAM where returned data must be stored

9.5.3 WriteBlock Function

The WriteBlock function is used to store data in the Flash. Data is moved 64 bytes at a time from SRAM to Flash using this function. The first thing the WriteBlock function does is to check the protection bits and determine if the desired BLOCKID is writable. If write protection is turned on, the WriteBlock function will exit setting the accumulator and KEY2 back to 00h. KEY1 will have a value of 01h, indicating a write failure. The configuration of the WriteBlock function is straightforward. The BLOCKID of the Flash block, where the data is stored, must be determined and stored at SRAM address FAh.

The SRAM address of the first of the 64 bytes to be stored in Flash must be indicated using the POINTER variable in the parameter block (SRAM address FBh). Finally, the CLOCK and DELAY value must be set correctly. The CLOCK value determines the length of the write pulse that will be used to store the data in the Flash. The CLOCK and DELAY values are dependent on the CPU speed and must be set correctly.

Table 9-5. WriteBlock Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value, when SSC is executed.
BLOCKID	0,FAh	8KB Flash block number (00h–7Fh) 4KB Flash block number (00h–3Fh) 3KB Flash block number (00h–2Fh)
POINTER	0,FBh	First of 64 addresses in SRAM, where the data to be stored in Flash is located prior to calling WriteBlock.
CLOCK	0,FCh	Clock divider used to set the write pulse width.
DELAY	0,FEh	For a CPU speed of 12 MHz set to 56h

9.5.4 EraseBlock Function

The EraseBlock function is used to erase a block of 64 contiguous bytes in Flash. The first thing the EraseBlock function does is to check the protection bits and determine if the desired BLOCKID is writable. If write protection is turned on, the EraseBlock function will exit setting the accumulator and KEY2 back to 00h. KEY1 will have a value of 01h, indicating a write failure. The EraseBlock function is only useful as the first step in programming. Erasing a block will not cause data in a block to be one hundred percent unreadable. If the objective is to obliterate data in a block, the best method is to perform an EraseBlock followed by a WriteBlock of all zeros.

To set up the parameter block for the EraseBlock function, correct key values must be stored in KEY1 and KEY2. The block number to be erased must be stored in the BLOCKID variable and the CLOCK and DELAY values must be set based on the current CPU speed.

Table 9-6. EraseBlock Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value, when SSC is executed.
BLOCKID	0,FAh	Flash block number (00h–7Fh)
CLOCK	0,FCh	Clock divider used to set the erase pulse width.
DELAY	0,FEh	For a CPU speed of 12 MHz set to 56h

9.5.5 ProtectBlock Function

The enCoRe II devices offer Flash protection on a block-by-block basis. *Table 9-7* lists the protection modes available. In the table, ER and EW are used to indicate the ability to perform external reads and writes. For internal writes, IW is used. Internal reading is always permitted by way of the ROMX instruction. The ability to read by way of the SROM ReadBlock function is indicated by SR. The protection level is stored in two bits according to *Table 9-7*. These bits are bit packed into the 64 bytes of the protection block. Therefore, each protection block byte stores the protection level for four Flash blocks. The bits are packed into a byte, with the lowest numbered block's protection level stored in the lowest numbered bits *Table 9-7*.

The first address of the protection block contains the protection level for blocks 0 through 3; the second address is for blocks 4 through 7. The 64th byte will store the protection level for blocks 252 through 255.

Table 9-7. Protection Modes

Mode	Settings	Description	Marketing
00b	SR ER EW IW	Unprotected	Unprotected
01b	$\overline{SR}$ $\overline{ER}$ EW IW	Read protect	Factory upgrade
10b	$\overline{SR}$ $\overline{ER}$ $\overline{EW}$ $\overline{IW}$	Disable external write	Field upgrade
11b	$\overline{SR}$ $\overline{ER}$ $\overline{EW}$ $\overline{IW}$	Disable internal write	Full protection

7	6	5	4	3	2	1	0
Block n+3		Block n+2		Block n+1		Block n	

The level of protection is only decreased by an EraseAll, which places zeros in all locations of the protection block. To set the level of protection, the ProtectBlock function is used. This function takes data from SRAM, starting at address 80h, and ORs it with the current values in the protection block. The result of the OR operation is then stored in the protection block. The EraseBlock function does not change the protection level for a block. Because the SRAM location for the protection data is fixed and there is only one protection block per Flash macro, the ProtectBlock function expects very few variables in the parameter block to be set prior to calling the function. The parameter block values that must be set, besides the keys, are the CLOCK and DELAY values.

Table 9-8. ProtectBlock Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value when SSC is executed.
CLOCK	0,FCh	Clock divider used to set the write pulse width.
DELAY	0,FEh	For a CPU speed of 12 MHz set to 56h

9.5.6 EraseAll Function

The EraseAll function performs a series of steps that destroy the user data in the Flash macros and resets the protection block in each Flash macro to all zeros (the unprotected state). The EraseAll function does not affect the three hidden blocks above the protection block, in each Flash macro. The first of these four hidden blocks is used to store the protection table for its eight Kbytes of user data.

The EraseAll function begins by erasing the user space of the Flash macro with the highest address range. A bulk program of all zeros is then performed on the same Flash macro, to destroy all traces of the previous contents. The bulk program is followed by a second erase that leaves the Flash macro in a state ready for writing. The erase, program, erase sequence is then performed on the next lowest Flash macro in the address space if it exists. Following the erase of the user space, the protection block for the Flash macro with the highest address range is erased. Following the erase of the protection block, zeros are written into every bit of the protection table. The next lowest Flash macro in the address space then has its protection block erased and filled with zeros.

The end result of the EraseAll function is that all user data in the Flash is destroyed and the Flash is left in an unprogrammed state, ready to accept one of the various write commands. The protection bits for all user data are also reset to the zero state

The parameter block values that must be set, besides the keys, are the CLOCK and DELAY values.

Table 9-9. EraseAll Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value when SSC is executed.
CLOCK	0,FCh	Clock divider used to set the write pulse width.
DELAY	0,FEh	For a CPU speed of 12 MHz set to 56h

9.5.7 TableRead Function

The TableRead function gives the user access to part-specific data stored in the Flash during manufacturing. It also returns a Revision ID for the die (not to be confused with the Silicon ID).

Table 9-10. Table Read Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value when SSC is executed.
BLOCKID	0,FAh	Table number to read.

The table space for the enCoRe II is simply a 64-byte row broken up into eight tables of eight bytes. The tables are numbered zero through seven. All user and hidden blocks in the CY7C638xx parts consist of 64 bytes.

An internal table holds the Silicon ID and returns the Revision ID. The Silicon ID is returned in SRAM, while the Revision ID is returned in the CPU_A and CPU_X registers. The Silicon ID is a value placed in the table by programming the Flash and is controlled by Cypress Semiconductor Product Engineering. The Revision ID is hard coded into the SROM. The Revision ID is discussed in more detail later in this section.

An internal table holds alternate trim values for the device and returns a one-byte internal revision counter. The internal revision counter starts out with a value of zero and is incremented each time one of the other revision numbers is not incremented. It is reset to zero each time one of the other revision numbers is incremented. The internal revision count is returned in the CPU_A register. The CPU_X register will always be set to FFh when trim values are read. The BLOCKID value, in the parameter block, is used to indicate which table should be returned to the user. Only the three least significant bits of the BLOCKID parameter are used by TableRead function for the CY7C638xx. The upper five bits are ignored. When the function is called, it transfers bytes from the table to SRAM addresses F8h–FFh.

The M8C's A and X registers are used by the TableRead function to return the die's Revision ID. The Revision ID is a 16-bit value hard coded into the SROM that uniquely identifies the die's design.

9.5.8 Checksum Function

The Checksum function calculates a 16-bit checksum over a user specifiable number of blocks, within a single Flash macro (Bank) starting from block zero. The BLOCKID parameter is used to pass in the number of blocks to calculate the checksum over. A BLOCKID value of 1 will calculate the checksum of only block 0, while a BLOCKID value of 0 will calculate the checksum of all 256 user blocks. The 16-bit checksum is returned in KEY1 and KEY2. The parameter KEY1 holds the lower eight bits of the checksum and the parameter KEY2 holds the upper eight bits of the checksum.

The checksum algorithm executes the following sequence of three instructions over the number of blocks times 64 to be checksummed.

```
romx
add [KEY1], A
adc [KEY2], 0
```

Table 9-11. Checksum Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value when SSC is executed.
BLOCKID	0,FAh	Number of Flash blocks to calculate checksum on.

10.0 Clocking

The enCoRe II internal oscillator outputs two frequencies, the Internal 24 MHz Oscillator and the 32 KHz Low-power Oscillator.

The Internal 24 MHz Oscillator is designed such that it may be trimmed to an output frequency of 24 MHz over temperature and voltage variation. With the presence of USB traffic, the Internal 24 MHz Oscillator can be set to precisely tune to USB timing requirements (24 MHz ± 1.5%). Without USB traffic, the Internal 24 MHz Oscillator accuracy is 24 MHz ± 5% (between 0°–70°C). No external components are required to achieve this level of accuracy.

The internal low-speed oscillator of nominally 32 KHz provides a slow clock source for the enCoRe II in suspend mode, particularly to generate a periodic wake-up interrupt and also to provide a clock to sequential logic during power-up and power-down events when the main clock is stopped. In addition, this oscillator can also be used as a clocking source for the Interval Timer clock (ITMRCLK) and Capture Timer clock (TCAPCLK). The 32 KHz Low-power Oscillator can operate in low-power mode or can provide a more accurate clock in normal mode. The Internal 32 KHz Low-power Oscillator accuracy ranges (between 0°–70° C) as follows:

5V Normal mode: –8% to + 16%

5V LP mode: +12% to + 48%

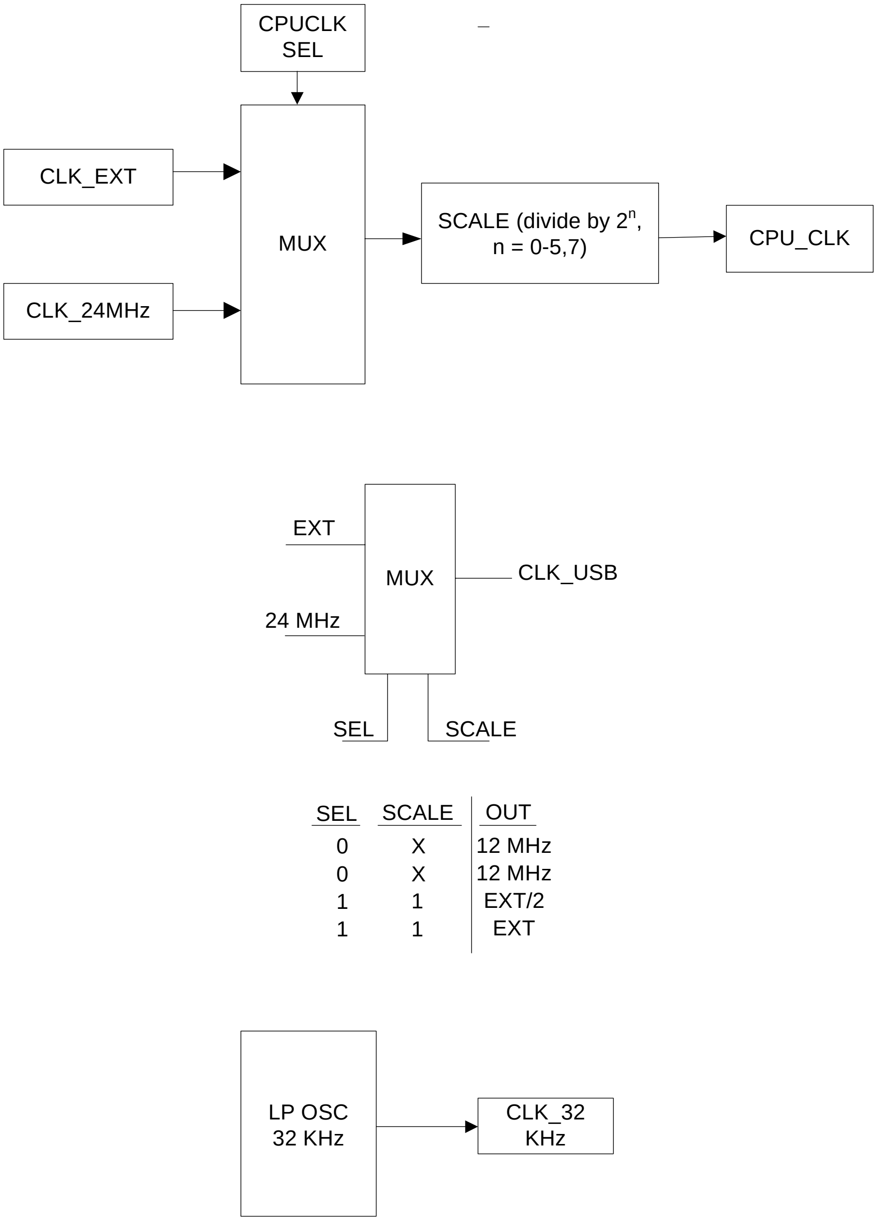
3.3V Normal mode: –10% to + 8%

3.3V LP mode: +4% to 35%

When using the 32 KHz oscillator the PITMRL/H registers must be read until 2 consecutive readings match before sending/receiving data. The following firmware example assumes the developer is interested in the lower byte of the PIT.

```
Read_PIT_counter:
mov A, reg[PITMRL]
mov [57h], A
mov A, reg[PITMRL]
mov [58h], A
mov [59h], A
mov A, reg[PITMRL]
mov [60h], A
;;;Start comparison
mov A, [60h]
mov X, [59h]
sub A, [59h]
jz done
mov A, [59h]
mov X, [58h]
sub A, [58h]
jz done
mov X, [57h]
;;;correct data is in memory location 57h
done:
mov [57h], X
ret
```

Figure 10-1. Clock Block Diagram



SEL	SCALE	OUT
0	X	12 MHz
0	X	12 MHz
1	1	EXT/2
1	1	EXT

10.1 Clock Architecture Description

The enCoRe II clock selection circuitry allows the selection of independent clocks for the CPU, USB, Interval Timers and Capture Timers.

The CPU clock, CPUCLK, can be sourced from an external clock or the Internal 24 MHz Oscillator. The selected clock source can optionally be divided by 2^n where n is 0-5,7 (see *Table 10-4*).

USBCLK, which must be 12 MHz for the USB SIE to function properly, can be sourced by the Internal 24 MHz Oscillator or an external 12 MHz/24 MHz clock. An optional divide by two allows the use of 24 MHz source.

The Interval Timer clock (ITMRCLK), can be sourced from an external clock, the Internal 24 MHz Oscillator, the Internal 32 KHz Low-power Oscillator, or from the timer capture clock (TCAPCLK). A programmable prescaler of 1, 2, 3, 4 then divides the selected source.

The Timer Capture clock (TCAPCLK) can be sourced from an external clock, Internal 24 MHz Oscillator, or the Internal 32 KHz Low-power Oscillator.

The CLKOUT pin (P0.1) can be driven from one of many sources. This is used for test and can also be used in some applications. The sources that can drive the CLKOUT are:

- CLKIN after the optional EFTB filter
- Internal 24 MHz Oscillator
- Internal 32 KHz Low-power Oscillator
- CPUCLK after the programmable divider

Table 10-1. IOSC Trim (IOSCTR) [0x34] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	foffset[2:0]			Gain[4:0]				
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	D	D	D	D	D

The IOSC Calibrate register is used to calibrate the internal oscillator. The reset value is undefined but during boot the SROM writes a calibration value that is determined during manufacturing test. This value should not require change during normal use. This is the meaning of ‘D’ in the Default field
Bit [7:5]: foffset [2:0]
This value is used to trim the frequency of the internal oscillator. These bits are not used in factory calibration and will be zero. Setting each of these bits causes the appropriate fine offset in oscillator frequency.
foffset bit 0 = 7.5 KHz
foffset bit 1 = 15 KHz
foffset bit 2 = 30 KHz
Bit [4:0]: Gain [4:0]
The effective frequency change of the offset input is controlled through the gain input. A lower value of the gain setting increases the gain of the offset input. This value sets the size of each offset step for the internal oscillator. Nominal gain change (KHz/offsetStep) at each bit, typical conditions (24 MHz operation):
Gain bit 0 = –1.5 KHz
Gain bit 1 = –3.0 KHz
Gain bit 2 = –6 KHz
Gain bit 3 = –12 KHz
Gain bit 4 = –24 KHz

Table 10-2. LPOSC Trim (LPOSCTR) [0x36] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	32-KHz Low Power	Reserved	32-KHz Bias Trim [1:0]		32-KHz Freq Trim [3:0]			
Read/Write	R/W	–	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	D	D	D	D	D	D	D

This register is used to calibrate the 32 KHz Low-speed Oscillator. The reset value is undefined but during boot the SROM writes a calibration value that is determined during manufacturing test. This value should not require change during normal use. This is the meaning of ‘D’ in the Default field. If the 32 KHz Low-power bit needs to be written, care must be taken not to disturb the 32 KHz Bias Trim and the 32 KHz Freq Trim fields from their factory calibrated values

Bit 7: 32 KHz Low Power
0 = The 32 KHz Low-speed Oscillator operates in normal mode
1 = The 32 KHz Low-speed Oscillator operates in a low-power mode. The oscillator continues to function normally but with reduced accuracy

Bit 6: Reserved

Bit [5:4]: 32 KHz Bias Trim [1:0]
These bits control the bias current of the low-power oscillator.
0 0 = Mid bias
0 1 = High bias
1 0 = Reserved
1 1 = Reserved

Important Note: Do not program the 32 KHz Bias Trim [1:0] field with the reserved 10b value as the oscillator does not oscillate at all corner conditions with this setting

Bit [3:0]: 32 KHz Freq Trim [3:0]
These bits are used to trim the frequency of the low-power oscillator

Table 10-3. CPU/USB Clock Config (CPUCLKCR) [0x30] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	USB CLK/2 Disable	USB CLK Select	Reserved				CPUCLK Select
Read/Write	–	R/W	R/W	–	–	–	–	R/W
Default	0	0	0	0	0	0	0	0

Bit 7: Reserved

Bit 6: USB CLK/2 Disable
This bit only affects the USBCLK when the source is the external crystal oscillator. When the USBCLK source is the Internal 24 MHz Oscillator, the divide by two is always enabled
0 = USBCLK source is divided by two. This is the correct setting to use when the Internal 24 MHz Oscillator is used, or when the external source is used with a 24 MHz clock
1 = USBCLK is undivided. Use this setting only with a 12 MHz external clock

Bit 5: USB CLK Select
This bit controls the clock source for the USB SIE
0 = Internal 24 MHz Oscillator. With the presence of USB traffic, the Internal 24 MHz Oscillator can be trimmed to meet the USB requirement of 1.5% tolerance (see *Table 10-5*)
1 = External clock—Internal Oscillator is not trimmed to USB traffic. **Proper USB SIE operation requires a 12 MHz or 24 MHz clock accurate to <1.5%.**

Bit [4:1]: Reserved

Bit 0: CPU CLK Select
0 = Internal 24 MHz Oscillator.
1 = External clock—External clock at CLKIN (P0.0) pin.

Note: the CPU speed selection is configured using the OSC_CR0 Register (*Table 10-4*)

Table 10-4. OSC Control 0 (OSC_CR0) [0x1E0] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved		No Buzz	Sleep Timer [1:0]		CPU Speed [2:0]		
Read/Write	–	–	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:6]: Reserved
Bit 5: No Buzz
During sleep (the Sleep bit is set in the CPU_SCR Register—*Table 11-1*), the LVD and POR detection circuit is turned on periodically to detect any POR and LVD events on the V_{CC} pin (the Sleep Duty Cycle bits in the ECO_TR are used to control the duty cycle—*Table 13-3*). To facilitate the detection of POR and LVD events, the No Buzz bit is used to force the LVD and POR detection circuit to be continuously enabled during sleep. This results in a faster response to an LVD or POR event during sleep at the expense of a slightly higher than average sleep current
0 = The LVD and POR detection circuit is turned on periodically as configured in the Sleep Duty Cycle
1 = The Sleep Duty Cycle value is overridden. The LVD and POR detection circuit is always enabled
Note: The periodic Sleep Duty Cycle enabling is independent with the sleep interval shown in the Sleep [1:0] bits below
Bit [4:3]: Sleep Timer [1:0]

Sleep Timer [1:0]	Sleep Timer Clock Frequency (Nominal)	Sleep Period (Nominal)	Watchdog Period (Nominal)
00	512 Hz	1.95 ms	6 ms
01	64 Hz	15.6 ms	47 ms
10	8 Hz	125 ms	375 ms
11	1 Hz	1 sec	3 sec

Note: Sleep intervals are approximate
Bit [2:0]: CPU Speed [2:0]
The enCoRe II may operate over a range of CPU clock speeds. The reset value for the CPU Speed bits is zero; therefore, the default CPU speed is one-eighth of the internal 24 MHz, or 3 MHz
Regardless of the CPU Speed bit's setting, if the actual CPU speed is greater than 12 MHz, the 24 MHz operating requirements apply. An example of this scenario is a device that is configured to use an external clock, which is supplying a frequency of 20 MHz. If the CPU speed register's value is 0b011, the CPU clock will be 20 MHz. Therefore the supply voltage requirements for the device are the same as if the part was operating at 24 MHz. The operating voltage requirements are not relaxed until the CPU speed is at 12 MHz or less

CPU Speed [2:0]	CPU when Internal Oscillator is selected	External Clock
000	3 MHz (Default)	Clock In/8
001	6 MHz	Clock In/4
010	12 MHz	Clock In/2
011	24 MHz	Clock In/1
100	1.5 MHz	Clock In/16
101	750 KHz	Clock In/32
110	187 KHz	Clock In/128
111	Reserved	Reserved

Important Note: Correct USB operations require the CPU clock speed be at least 1.5 MHz or not less than USB clock/8. If the two clocks have the same source then the CPU clock divider should not be set to divide by more than 8. If the two clocks have different sources, care must be taken to ensure that the maximum ratio of USB Clock/CPU Clock can never exceed 8 across the full specification range of both clock sources

Table 10-5. USB Osclock Clock Configuration (OSCLCKCR) [0x39] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved						Fine Tune Only	USB Osclock Disable
Read/Write	–	–	–	–	–	–	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register is used to trim the Internal 24 MHz Oscillator using received low-speed USB packets as a timing reference. The USB Osclock circuit is active when the Internal 24 MHz Oscillator provides the USB clock
Bit [7:2]: Reserved
Bit 1: Fine Tune Only
0 = Enable
1 = Disable the oscillator lock from performing the coarse-tune portion of its retuning. The oscillator lock must be allowed to perform a coarse tuning in order to tune the oscillator for correct USB SIE operation. After the oscillator is properly tuned this bit can be set to reduce variance in the internal oscillator frequency that would be caused course tuning
Bit 0: USB Osclock Disable
0 = Enable. With the presence of USB traffic, the Internal 24 MHz Oscillator precisely tunes to 24 MHz ± 1.5%
1 = Disable. The Internal 24 MHz Oscillator is not trimmed based on USB packets. This setting is useful when the internal oscillator is not sourcing the USBSIE clock

Table 10-6. Timer Clock Config (TMRCLKCR) [0x31] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	TCAPCLK Divider		TCAPCLK Select		ITMRCLK Divider		ITMRCLK Select	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	1	0	0	0	1	1	1	1

Bit [7:6]: TCAPCLK Divider [1:0]
TCAPCLK Divider controls the TCAPCLK divisor
0 0 = Divider Value 2
0 1 = Divider Value 4
1 0 = Divider Value 6
1 1 = Divider Value 8
Bit [5:4]: TCAPCLK Select
The TCAPCLK Select field controls the source of the TCAPCLK
0 0 = Internal 24 MHz Oscillator
0 1 = External clock—external clock at CLKIN (P0.0) input.
1 0 = Internal 32-KHz Low-power Oscillator
1 1 = TCAPCLK Disabled
Note: The 1024-μs interval timer is based on the assumption that TCAPCLK is running at 4 MHz. Changes in TCAPCLK frequency will cause a corresponding change in the 1024-μs interval timer frequency
Bit [3:2]: ITMRCLK Divider
ITMRCLK Divider controls the ITMRCLK divisor.
0 0 = Divider value of 1
0 1 = Divider value of 2
1 0 = Divider value of 3
1 1 = Divider value of 4
Bit [1:0]: ITMRCLK Select
0 0 = Internal 24 MHz Oscillator
0 1 = External clock—external clock at CLKIN (P0.0) input.
1 0 = Internal 32-KHz Low-power Oscillator
1 1 = TCAPCLK

10.1.1 Interval Timer Clock (ITMRCLK)

The Interval Timer Clock (TITMRCLK), can be sourced from an external clock, the Internal 24 MHz Oscillator, the Internal 32-KHz Low-power Oscillator, or the Timer Capture clock. A programmable prescaler of 1, 2, 3 or 4 then divides the selected source. The 12-bit Programmable Interval Timer is a simple down counter with a programmable reload value. It provides a 1 μs resolution by default. When the down counter reaches zero, the next clock is spent reloading. The reload value can be read and written while the counter is running, but care must be taken to ensure that the counter does not unintentionally reload while the 12-bit reload value is only partially stored—i.e., between the two writes of the 12-bit value. The programmable interval timer generates an interrupt to the CPU on each reload.

The parameters to be set will show up on the device editor view of PSoC Designer once you place the enCoRe II Timer User Module. The parameters are PITIMER_Source and PITIMER_Divider. The PITIMER_Source is the clock to the timer and the PITMER_Divider is the value the clock is divided by.

The interval register (PITMR) holds the value that is loaded into the PIT counter on terminal count. The PIT counter is a down counter.

The Programmable Interval Timer resolution is configurable. For example:

TCAPCLK divide by x of CPU clock (for example TCAPCLK divide by 2 of a 24 MHz CPU clock will give a frequency of 12 MHz)

ITMRCLK divide by x of TCAPCLK (for example, ITMRCLK divide by 3 of TCAPCLK is 4 MHz so resolution is 0.25 µs)

10.1.2 Timer Capture Clock (TCAPCLK)

The Timer Capture clock can be sourced from an external clock, Internal 24 MHz Oscillator or the Internal 32-KHz Low-power Oscillator. A programmable pre-scaler of 2, 4, 6, or 8 then divides the selected source.

Figure 10-2. Programmable Interval Timer Block Diagram

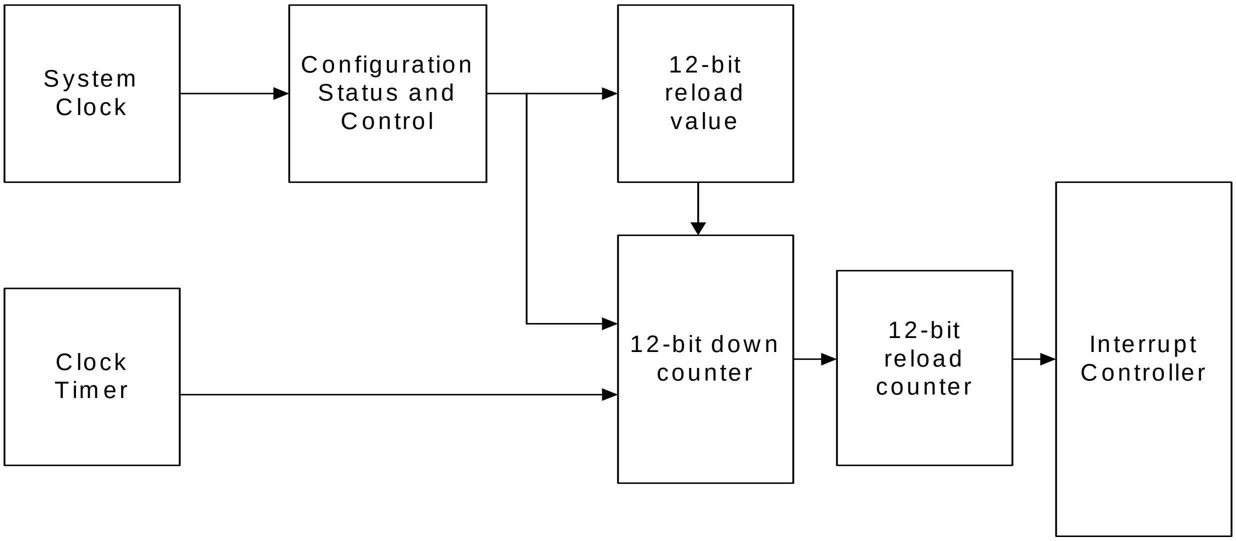


Figure 10-3. Timer Capture Block Diagram

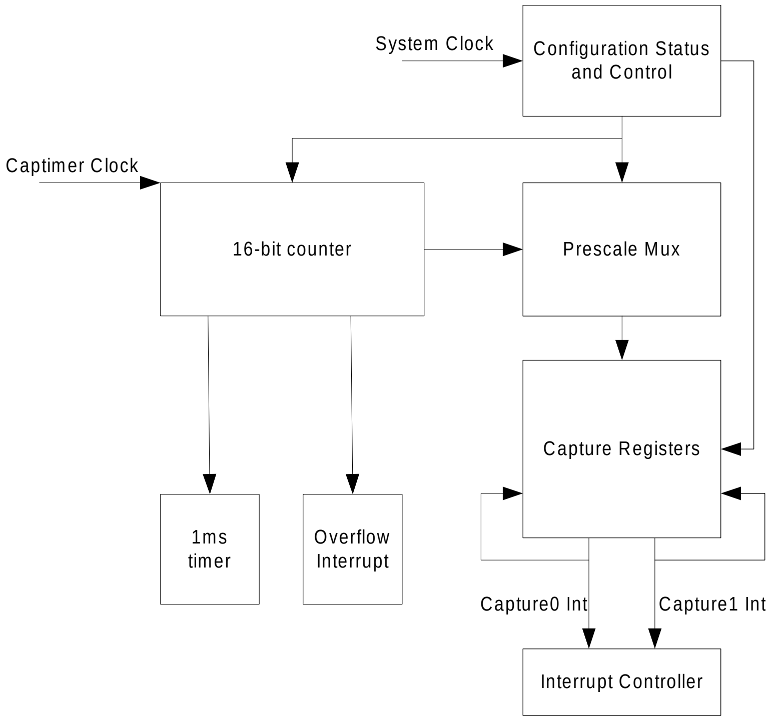


Table 10-7. Clock I/O Config (CLKIOCR) [0x32] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved						CLKOUT Select	
Read/Write	–	–	–	-	-	-	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:2]: Reserved
Bit [1:0]: CLKOUT Select
0 0 = Internal 24 MHz Oscillator
0 1 = External clock – external clock at CLKIN (P0.0)
1 0 = Internal 32-KHz Low-power Oscillator
1 1 = CPUCLK

10.2 CPU Clock During Sleep Mode

When the CPU enters sleep mode the CPUCLK Select (Bit [0], *Table 10-3*) is forced to the Internal Oscillator, and the oscillator is stopped. When the CPU comes out of sleep mode it is running on the internal oscillator. The internal oscillator recovery time is three clock cycles of the Internal 32-KHz Low-power Oscillator.

If the system requires the CPU to run off the external clock after awaking from sleep mode, firmware will need to switch the clock source for the CPU.

11.0 Reset

The microcontroller supports two types of resets: Power-on Reset (POR) and Watchdog Reset (WDR). When reset is initiated, all registers are restored to their default states and all interrupts are disabled.

The occurrence of a reset is recorded in the System Status and Control Register (CPU_SCR). Bits within this register record the occurrence of POR and WDR Reset respectively. The firmware can interrogate these bits to determine the cause of a reset.

The microcontroller resumes execution from Flash address 0x0000 after a reset. The internal clocking mode is active after a reset, until changed by user firmware.

Note The CPU clock defaults to 3 MHz (Internal 24 MHz Oscillator divide-by-8 mode) at POR to guarantee operation at the low V_{CC} that might be present during the supply ramp.

Table 11-1. System Status and Control Register (CPU_SCR) [0xFF] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	GIES	Reserved	WDRS	PORS	Sleep	Reserved		Stop
Read/Write	R	–	R/C[4]	R/C[4]	R/W	–	–	R/W
Default	0	0	0	1	0	0	0	0

The bits of the CPU_SCR register are used to convey status and control of events for various functions of an enCoRe II device
Bit 7: GIES
The Global Interrupt Enable Status bit is a read only status bit and its use is discouraged. The GIES bit is a legacy bit, which was used to provide the ability to read the GIE bit of the CPU_F register. However, the CPU_F register is now readable. When this bit is set, it indicates that the GIE bit in the CPU_F register is also set which, in turn, indicates that the microprocessor will service interrupts
0 = Global interrupts disabled
1 = Global interrupt enabled
Bit 6: Reserved
Bit 5: WDRS
The WDRS bit is set by the CPU to indicate that a WDR event has occurred. The user can read this bit to determine the type of reset that has occurred. The user can clear but not set this bit
0 = No WDR
1 = A WDR event has occurred
Bit 4: PORS
The PORS bit is set by the CPU to indicate that a POR event has occurred. The user can read this bit to determine the type of reset that has occurred. The user can clear but not set this bit
0 = No POR
1 = A POR event has occurred. (Note that WDR events will not occur until this bit is cleared)
Bit 3: SLEEP
Set by the user to enable CPU sleep state. CPU will remain in sleep mode until any interrupt is pending. The Sleep bit is covered in more detail in the Sleep Mode section
0 = Normal operation
1 = Sleep
Bit [2:1]: Reserved
Bit 0: STOP
This bit is set by the user to halt the CPU. The CPU will remain halted until a reset (WDR, POR, or external reset) has taken place. If an application wants to stop code execution until a reset, the preferred method would be to use the HALT instruction rather than writing to this bit
0 = Normal CPU operation
1 = CPU is halted (not recommended)

11.1 Power-on Reset

POR occurs every time the power to the device is switched on. POR is released when the supply is typically 2.6V for the upward supply transition, with typically 50 mV of hysteresis during the power-on transient. Bit 4 of the System Status and Control Register (CPU_SCR) is set to record this event (the register contents are set to 00010000 by the POR). After a POR, the microprocessor is held off for approximately 20 ms for the V_{CC} supply to stabilize before executing the first instruction at address 0x00 in the Flash. If the V_{CC} voltage drops below the POR downward supply trip point, POR is reasserted. The V_{CC} supply needs to ramp linearly from 0 to 4V in less than 200 ms.

Important: The PORS status bit is set at POR and can only be cleared by the user. It cannot be set by firmware.

11.2 Watchdog Timer Reset

The user has the option to enable the WDT. The WDT is enabled by clearing the PORS bit. Once the PORS bit is cleared, the WDT cannot be disabled. The only exception to this is if a POR event takes place, which will disable the WDT.

The sleep timer is used to generate the sleep time period and the Watchdog time period. The sleep timer uses the Internal 32-KHz Low-power Oscillator system clock to produce the sleep time period. The user can program the sleep time period using the Sleep Timer bits of the OSC_CR0 Register (*Table 10-4*). When the sleep time elapses (sleep timer overflows), an interrupt to the Sleep Timer Interrupt Vector will be generated.

The Watchdog Timer period is automatically set to be three counts of the Sleep Timer overflows. This represents between two and three sleep intervals depending on the count in the Sleep Timer at the previous WDT clear. When this timer reaches three, a WDR is generated.

The user can either clear the WDT, or the WDT and the Sleep Timer. Whenever the user writes to the Reset WDT Register (RES_WDT), the WDT will be cleared. If the data that is written is the hex value 0x38, the Sleep Timer will also be cleared at the same time.

Note
[4] C = Clear. This bit can only be cleared by the user and cannot be set by firmware.

Table 11-2. Reset Watchdog Timer (RESWDT) [0xE3] [W]

Bit #	7	6	5	4	3	2	1	0
Field	Reset Watchdog Timer [7:0]							
Read/Write	W	W	W	W	W	W	W	W
Default	0	0	0	0	0	0	0	0
Any write to this register will clear Watchdog Timer, a write of 0x38 will also clear the Sleep Timer **Bit [7:0]:** Reset Watchdog Timer [7:0]								

12.0 Sleep Mode

The CPU can only be put to sleep by the firmware. This is accomplished by setting the Sleep bit in the System Status and Control Register (CPU_SCR). This stops the CPU from executing instructions, and the CPU will remain asleep until an interrupt comes pending, or there is a reset event (either a Power-on Reset, or a Watchdog Timer Reset).

The Low-voltage Detection circuit (LVD) drops into fully functional power-reduced states, and the latency for the LVD is increased. The actual latency can be traded against power consumption by changing Sleep Duty Cycle field of the ECO_TR Register.

The Internal 32 KHz Low-speed Oscillator remains running. Prior to entering suspend mode, firmware can optionally configure the 32 KHz Low-speed Oscillator to operate in a low-power mode to help reduce the over all power consumption (Using Bit 7, *Table 10-2*). This will help save approximately 5 μA; however, the trade off is that the 32 KHz Low-speed Oscillator will be less accurate.

All interrupts remain active. Only the occurrence of an interrupt will wake the part from sleep. The Stop bit in the System Status and Control Register (CPU_SCR) must be cleared for a part to resume out of sleep. The Global Interrupt Enable bit of the CPU Flags Register (CPU_F) does not have any effect. Any unmasked interrupt will wake the system up. As a result, any interrupts not intended for waking must be disabled through the Interrupt Mask Registers.

When the CPU enters sleep mode the CPUCLK Select (Bit 1, *Table 10-3*) is forced to the Internal Oscillator. The internal oscillator recovery time is three clock cycles of the Internal 32 KHz Low-power Oscillator. The Internal 24 MHz Oscillator restarts immediately on exiting Sleep mode. If an external clock is used, firmware will need to switch the clock source for the CPU.

On exiting sleep mode, once the clock is stable and the delay time has expired, the instruction immediately following the sleep instruction is executed before the interrupt service routine (if enabled).

The Sleep interrupt allows the microcontroller to wake up periodically and poll system components while maintaining very low average power consumption. The Sleep interrupt may also be used to provide periodic interrupts during non-sleep modes.

12.1 Sleep Sequence

The SLEEP bit is an input into the sleep logic circuit. This circuit is designed to sequence the device into and out of the hardware sleep state. The hardware sequence to put the device to sleep is shown in *Figure 12-1* and is defined as follows.

1.Firmware sets the SLEEP bit in the CPU_SCR0 register. The Bus Request (BRQ) signal to the CPU is immediately asserted. This is a request by the system to halt CPU operation at an instruction boundary. The CPU samples BRQ on the positive edge of CPUCLK.

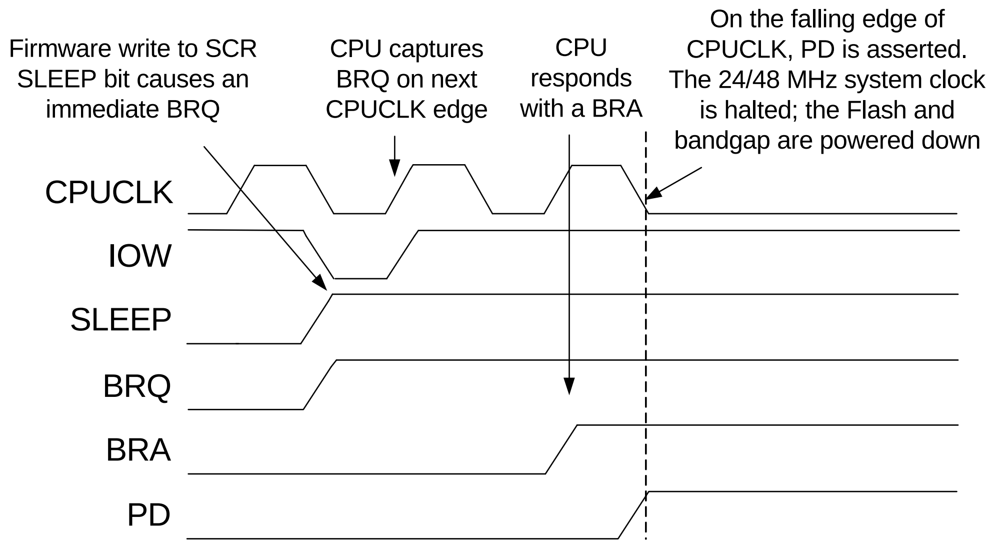
2.Due to the specific timing of the register write, the CPU issues a Bus Request Acknowledge (BRA) on the following positive edge of the CPU clock. The sleep logic waits for the following negative edge of the CPU clock and then asserts a system-wide Power Down (PD) signal. In *Figure 12-1* the CPU is halted and the system-wide power down signal is asserted.

3. The system-wide PD (power down) signal controls several major circuit blocks: The Flash memory module, the internal 24 MHz oscillator, the EFTB filter and the bandgap voltage reference. These circuits transition into a zero power state. The only operational circuits on chip are the Low Power oscillator, the bandgap refresh circuit, and the supply voltage monitor (POR/LVD) circuit.

Note: To achieve the lowest possible power consumption during suspend/sleep, the following conditions must be observed in addition to considerations for the sleep timer:

- All GPIOs must be set to outputs and driven low
- The USB pins P1.0 and P1.1 should be configured as inputs with their pull-ups enabled.

Figure 12-1. Sleep Timing


Firmware write to SCR SLEEP bit causes an immediate BRQ
CPU captures BRQ on next CPUCLK edge
CPU responds with a BRA
On the falling edge of CPUCLK, PD is asserted. The 24/48 MHz system clock is halted; the Flash and bandgap are powered down
CPUCLK
IOW
SLEEP
BRQ
BRA
PD

12.2 Wake up Sequence

Once asleep, the only event that can wake the system up is an interrupt. The global interrupt enable of the CPU flag register does not need to be set. Any unmasked interrupt will wake the system up. It is optional for the CPU to actually take the interrupt after the wake up sequence. The wake up sequence is synchronized to the 32 kHz clock for purposes of sequencing a startup delay, to allow the Flash memory module enough time to power up before the CPU asserts the first read access. Another reason for the delay is to allow the oscillator, Bandgap, and LVD/POR circuits time to settle before actually being used in the system. As shown in *Figure 12-2*, the wake up sequence is as follows:

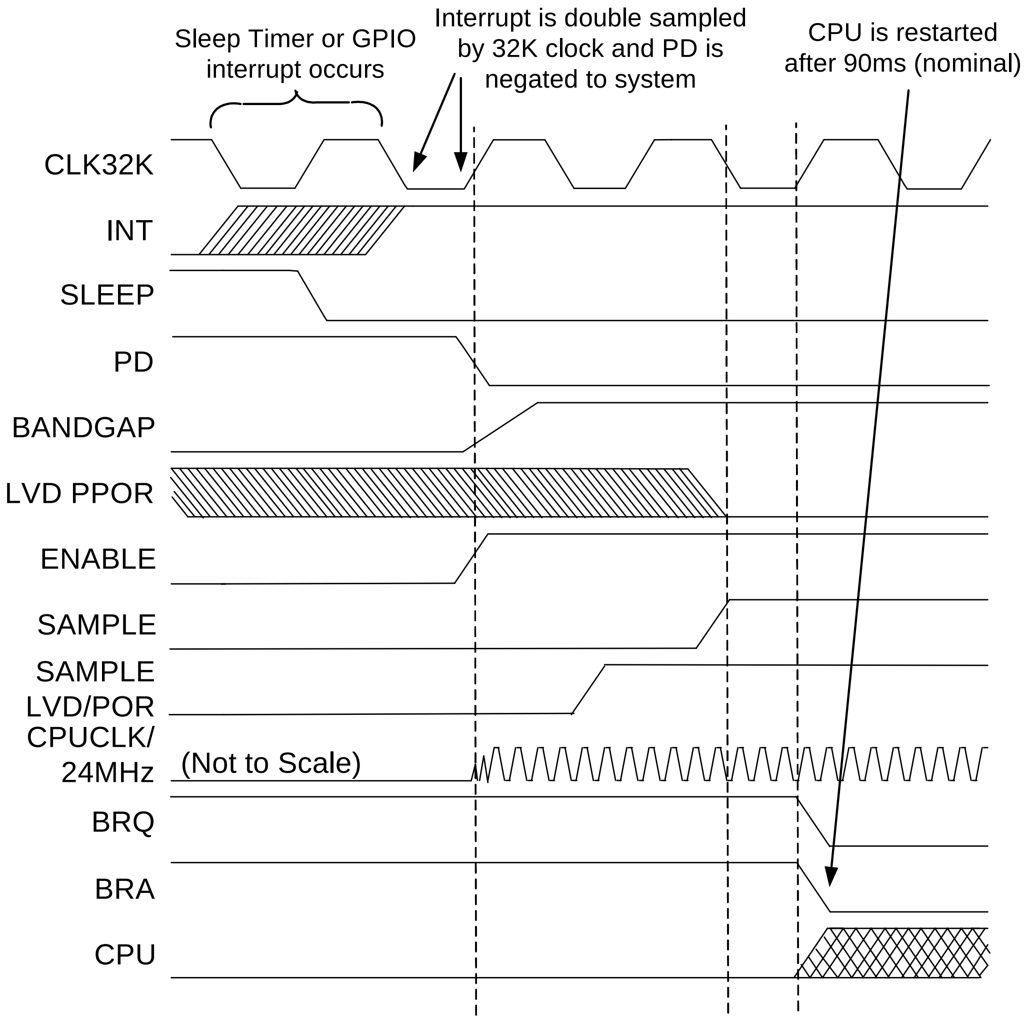
1.The wake up interrupt occurs and is synchronized by the negative edge of the 32 kHz clock.

2.At the following positive edge of the 32 kHz clock, the system wide PD signal is negated. The Flash memory module, internal oscillator, EFTB, and bandgap circuit are all powered up to a normal operating state.

3.At the following positive edge of the 32 kHz clock, the current values for the precision POR and LVD have settled and are sampled.

4.At the following negative edge of the 32 kHz clock (after about 15 µS nominal), the BRQ signal is negated by the sleep logic circuit. On the following CPUCLK, BRA is negated by the CPU and instruction execution resumes. Note that in *Figure 12-2* fixed function blocks, such as Flash, internal oscillator, EFTB, and bandgap, have about 15 µSec start up. The wake-up times (interrupt to CPU operational) will range from 75 µS to 105 µS.

Figure 12-2. Wake up Timing


Sleep Timer or GPIO interrupt occurs
Interrupt is double sampled by 32K clock and PD is negated to system
CPU is restarted after 90ms (nominal)
CLK32K
INT
SLEEP
PD
BANDGAP
LVD PPOR
ENABLE
SAMPLE
SAMPLE LVD/POR
CPUCLK/ 24MHz
(Not to Scale)
BRQ
BRA
CPU

13.0 Low-voltage Detect Control

Table 13-1. Low-voltage Control Register (LVDCR) [0x1E3] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved		PORLEV[1:0]		Reserved	VM[2:0]		
Read/Write	–	–	R/W	R/W	–	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register controls the configuration of the Power-on Reset/Low-voltage Detection block
Bit [7:6]: Reserved
Bit [5:4]: PORLEV[1:0]
This field controls the level below which the precision power-on-reset (PPOR) detector generates a reset
0 0 = 2.7V Range (trip near 2.6V)
0 1 = 3V Range (trip near 2.9V)
1 0 = 5V Range, ≥4.75V (trip near 4.65V)
1 1 = PPOR will not generate a reset, but values read from the Voltage Monitor Comparators Register (*Table 13-2*) give the internal PPOR comparator state with trip point set to the 3V range setting
Bit 3: Reserved
Bit [2:0]: VM[2:0]
This field controls the level below which the low-voltage-detect trips—possibly generating an interrupt and the level at which the Flash is enabled for operation.

VM[2:0]	LVD Trip Point (V) Min	LVD Trip Point (V) Typ	LVD Trip Point (V) Max
000	2.681	2.70	2.735
001	2.892	2.92	2.950
010	2.991	3.02	3.053
011	3.102	3.13	3.164
100	4.439	4.48	4.528
101	4.597	4.64	4.689
110	4.680	4.73	4.774
111	4.766	4.82	4.862

Table 13-2. Voltage Monitor Comparators Register (VLTCMP) [0x1E4] [R]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved						LVD	PPOR
Read/Write	–	–	–	–	–	–	R	R
Default	0	0	0	0	0	0	0	0

This read-only register allows reading the current state of the Low-voltage-Detection and Precision-Power-On-Reset comparators
Bit [7:2]: Reserved
Bit 1: LVD
This bit is set to indicate that the low-voltage-detect comparator has tripped, indicating that the supply voltage has gone below the trip point set by VM[2:0] (See *Table 13-1*)
0 = No low-voltage-detect event
1 = A low-voltage-detect has tripped
Bit 0: PPOR
This bit is set to indicate that the precision-power-on-reset comparator has tripped, indicating that the supply voltage is below the trip point set by PORLEV[1:0]
0 = No precision-power-on-reset event
1 = A precision-power-on-reset event has occurred
Note: This register can only be accessed in the second bank of I/O space. This requires setting the XIO bit in the CPU flags register.

13.0.1 ECO Trim Register

Table 13-3. ECO (ECO_TR) [0x1EB] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Sleep Duty Cycle [1:0]		Reserved					
Read/Write	R/W	R/W	–	–	–	–	–	–
Default	0	0	0	0	0	0	0	0

This register controls the ratios (in numbers of 32-KHz clock periods) of "on" time versus "off" time for LVD and POR detection circuit
Bit [7:6]: Sleep Duty Cycle [1:0]
0 0 = 128 periods of the Internal 32 KHz Low-speed Oscillator
0 1 = 512 periods of the Internal 32 KHz Low-speed Oscillator
1 0 = 32 periods of the Internal 32 KHz Low-speed Oscillator
1 1 = 8 periods of the Internal 32 KHz Low-speed Oscillator

14.0 General-purpose I/O Ports

14.1 Port Data Registers

Table 14-1. P0 Data Register (P0DATA)[0x00] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	P0.7	P0.6/TIO1	P0.5/TIO0	P0.4/INT2	P0.3/INT1	P0.2/INT0	P0.1/CLKOUT	P0.0/CLKIN
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register contains the data for Port 0. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 0 pins.
Bit 7: P0.7 Data
P0.7 only exists in the CY7C638xx
Bit [6:5]: P0.6–P0.5 Data/TIO1 and TIO0
Beside their use as the P0.6–P0.5 GPIOs, these pins can also be used for the alternate functions as the Capture Timer input or Timer output pins (TIO1 and TIO0). To configure the P0.5 and P0.6 pins, refer to the P0.5/TIO0–P0.6/TIO1 Configuration Register (*Table 14-8*)
The use of the pins as the P0.6–P0.5 GPIOs and the alternate functions exist in all the enCoRe II parts
Bit [4:2]: P0.4–P0.2 Data/INT2 – INT0
Beside their use as the P0.4–P0.2 GPIOs, these pins can also be used for the alternate functions as the Interrupt pins (INT0–INT2). To configure the P0.4–P0.2 pins, refer to the P0.2/INT0–P0.4/INT2 Configuration Register (*Table 14-7*)
The use of the pins as the P0.4–P0.2 GPIOs and the alternate functions exist in all the enCoRe II parts
Bit 1: P0.1/CLKOUT
Beside its use as the P0.1 GPIO, this pin can also be used for the alternate function as the CLK OUT pin. To configure the P0.1 pin, refer to the P0.1/CLKOUT Configuration Register (*Table 14-6*)
Bit 0: P0.0/CLKIN
Beside its use as the P0.0 GPIO, this pin can also be used for the alternate function as the CLKIN pin. To configure the P0.0 pin, refer to the P0.0/CLKIN Configuration Register (*Table 14-5*)

Table 14-2. P1 Data Register (P1DATA) [0x01] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	P1.7	P1.6/SMISO	P1.5/SMOSI	P1.4/SCLK	P1.3/SSEL	P1.2/VREG	P1.1/D–	P1.0/D+
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register contains the data for Port 1. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 1 pins.
Bit 7: P1.7 Data
P1.7 only exists in the CY7C638xx
Bit [6:3]: P1.6–P1.3 Data/SPI Pins (SMISO, SMOSI, SCLK, SSEL)
Beside their use as the P1.6–P1.3 GPIOs, these pins can also be used for the alternate function as the SPI interface pins. To configure the P1.6–P1.3 pins, refer to the P1.3–P1.6 Configuration Register (*Table 14-13*)
The use of the pins as the P1.6–P1.3 GPIOs and the alternate functions exist in all the enCoRe II parts.
Bit 2: P1.2/VREG
On the CY7C638(2/3)3, this pin can be used as the P1.2 GPIO or the VREG output. If the VREG output is enabled (Bit 0 *Table 19-1* is set), a 3.3V source is placed on the pin and the GPIO function of the pin is disabled
On the CY7C63813, this pin can only be used as the VREG output when USB mode is enabled. In non-USB mode, this pin can be used as the P1.2 GPIO. The VREG functionality is not present in the CY7C63310 and the CY7C63801 variants. A 1 μF min, 2 μF max capacitor is required on VREG output.
Bit [1:0]: P1.1–P1.0/D– and D+
When USB mode is disabled (Bit 7 in *Table 21-1* is clear), the P1.1 and P1.0 bits are used to control the state of the P1.0 and P1.1 pins. When the USB mode is enabled, the P1.1 and P1.0 pins are used as the D– and D+ pins respectively. If the USB Force State bit (Bit 0 in *Table 18-1*) is set, the state of the D– and D+ pins can be controlled by writing to the D– and D+ bits

Table 14-3. P2 Data Register (P2DATA) [0x02] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved						P2.1–P2.0	
Read/Write	-	-	-	-	-	-	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register contains the data for Port 2. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 2 pins
Bit [7:2]: Reserved Data [7:2]
Bit [1:0]: P2 Data [1:0]
P2.1–P2.0 only exist in the CY7C638(2/3)3

Table 14-4. P3 Data Register (P3DATA) [0x03] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved						P3.1–P3.0	
Read/Write	-	-	-	-	-	-	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register contains the data for Port 3. Writing to this register sets the bit values to be output on output enabled pins. Reading from this register returns the current state of the Port 3 pins
Bit [7:2]: Reserved Data [7:2]
Bit [1:0]: P3 Data [1:0]
P3.1–P3.0 only exist in the CY7C638(2/3)3

14.2 GPIO Port Configuration

All the GPIO configuration registers have common configuration controls. The following are the bit definitions of the GPIO configuration registers

14.2.1 Int Enable

When set, the Int Enable bit allows the GPIO to generate interrupts. Interrupt generate can occur regardless of whether the pin is configured for input or output. All interrupts are edge sensitive, however for any interrupt that is shared by multiple sources (i.e., Ports 2, 3, and 4) all inputs must be deasserted before a new interrupt can occur.

When clear, the corresponding interrupt is disabled on the pin.

It is possible to configure GPIOs as outputs, enable the interrupt on the pin and then to generate the interrupt by driving the appropriate pin state. This is useful in test and may have value in applications as well.

14.2.2 Int Act Low

When set, the corresponding interrupt is active on the falling edge.

When clear, the corresponding interrupt is active on the rising edge.

14.2.3 TTL Thresh

When set, the input has TTL threshold. When clear, the input has standard CMOS threshold.

14.2.4 High Sink

When set, the output can sink up to 50 mA.

When clear, the output can sink up to 8 mA.

On the CY7C638xx, only the P1.7–P1.3 have 50 mA sink drive capability. Other pins have 8 mA sink drive capability.

14.2.5 Open Drain

When set, the output on the pin is determined by the Port Data Register. If the corresponding bit in the Port Data Register is set, the pin is in high-impedance state. If the corresponding bit in the Port Data Register is clear, the pin is driven low.

When clear, the output is driven LOW or HIGH.

14.2.6 Pull up Enable

When set the pin has a 7K pull up to V_{CC} (or VREG for ports with V3.3 enabled).

When clear, the pull up is disabled.

14.2.7 Output Enable

When set, the output driver of the pin is enabled.

When clear, the output driver of the pin is disabled.

For pins with shared functions there are some special cases.

14.2.8 VREG Output/SPI Use

The P1.2(VREG), P1.3(SSEL), P1.4(SCLK), P1.5(SMOSI) and P1.6(SMISO) pins can be used for their dedicated functions or for GPIO.

To enable the pin for GPIO, clear the corresponding VREG Output or SPI Use bit. The SPI function controls the output enable for its dedicated function pins when their GPIO enable bit is clear. The VREG output is not available on the CY7C63801 and CY7C63310.

14.2.9 3.3V Drive

The P1.3(SSEL), P1.4(SCLK), P1.5(SMOSI) and P1.6(SMISO) pins have an alternate voltage source from the voltage regulator. If the 3.3V Drive bit is set a high level is driven from the voltage regulator instead of from V_{CC}.

Setting the 3.3V Drive bit does not enable the voltage regulator. That must be done explicitly by setting the VREG Enable bit in the VREGCR Register (*Table 19-1*).

Figure 14-1. Block Diagram of a GPIO

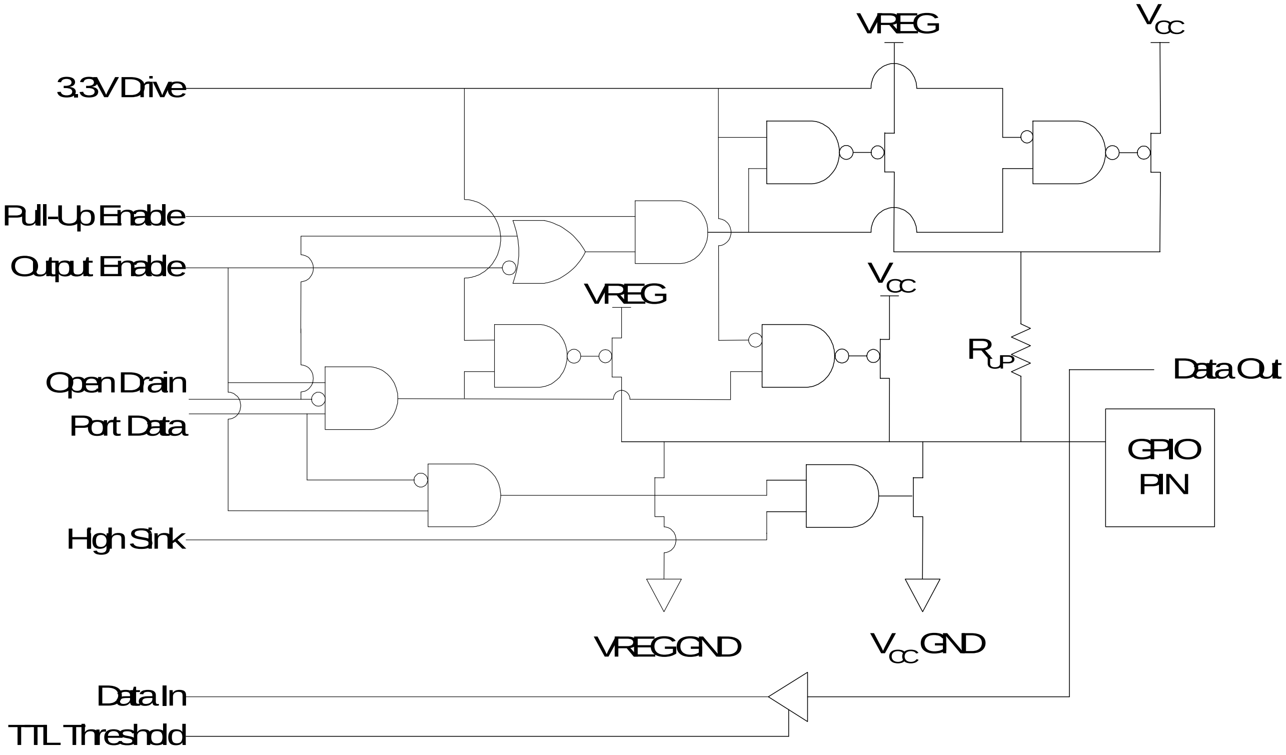


Table 14-5. P0.0/CLKIN Configuration (P00CR) [0x05] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable
Read/Write	--	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This pin is shared between the P0.0 GPIO use and the CLKIN pin for an external clock. When the external clock input is enabled the settings of this register are ignored
The use of the pin as the P0.0 GPIO is available in all the enCoRe II parts.
In the CY7C638xx, only 8 mA sink drive capability is available on this pin regardless of the setting of the High Sink bit

Table 14-6. P0.1/CLKOUT Configuration (P01CR) [0x06] R/W]

Bit #	7	6	5	4	3	2	1	0
Field	CLK Output	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This pin is shared between the P0.1 GPIO use and the CLKOUT pin. When CLK output is set, the internally selected clock is sent out onto P0.1CLKOUT pin.
The use of the pin as the P0.1 GPIO is available in all the enCoRe II parts. In the CY7C638xx, only 8 mA sink drive capability is available on this pin regardless of the setting of the High Sink bit
Bit 7: CLK Output
0 = The clock output is disabled
1 = The clock selected by the CLK Select field (Bit [1:0] of the CLKIOCR Register—*Table 10-7*) is driven out to the pin

Table 14-7. P0.2/INT0–P0.4/INT2 Configuration (P02CR–P04CR) [0x07–0x09] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved		Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable
Read/Write	–	–	R/W	R/W	–	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

These registers control the operation of pins P0.2–P0.4 respectively. These pins are shared between the P0.2–P0.4 GPIOs and the INT0–INT2. These registers exist in all enCoRe II parts. The INT0–INT2 interrupts are different than all the other GPIO interrupts. These pins are connected directly to the interrupt controller to provide three edge-sensitive interrupts with independent interrupt vectors. These interrupts occur on a rising edge when Int act Low is clear and on a falling edge when Int act Low is set. These pins are enabled as interrupt sources in the interrupt controller registers (*Table 17-8* and *Table 17-6*).
To use these pins as interrupt inputs configure them as inputs by clearing the corresponding Output Enable. If the INT0–INT2 pins are configured as outputs with interrupts enabled, firmware can generate an interrupt by writing the appropriate value to the P0.2, P0.3 and P0.4 data bits in the P0 Data Register
Regardless of whether the pins are used as Interrupt or GPIO pins the Int Enable, Int act Low, TTL Threshold, Open Drain, and Pull-up Enable bits control the behavior of the pin
The P0.2/INT0–P0.4/INT2 pins are individually configured with the P02CR (0x07), P03CR (0x08), and P04CR (0x09) respectively.
Note: Changing the state of the Int Act Low bit can cause an unintentional interrupt to be generated. When configuring these interrupt sources, it is best to follow the following procedure:

1. Disable interrupt source
2. Configure interrupt source
3. Clear any pending interrupts from the source
4. Enable interrupt source

Table 14-8. P0.5/TIO0 – P0.6/TIO1 Configuration (P05CR–P06CR) [0x0A–0x0B] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	TIO Output	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable
Read/Write	–	R/W	R/W	R/W	–	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

These registers control the operation of pins P0.5 through P0.6, respectively. These registers exist in all enCoRe II parts.
P0.5 and P0.6 are shared with TIO0 and TIO1, respectively. To use these pins as Capture Timer inputs, configure them as inputs by clearing the corresponding Output Enable. To use TIO0 and TIO1 as Timer outputs, set the TIOx Output and Output Enable bits. If these pins are configured as outputs and the TIO Output bit is clear, firmware can control the TIO0 and TIO1 inputs by writing the value to the P0.5 and P0.6 data bits in the P0 Data Register
Regardless of whether either pin is used as a TIO or GPIO pin the Int Enable, Int act Low, TTL Threshold, Open Drain, and Pull-up Enable control the behavior of the pin.
TIO0(P0.5) when enabled outputs a positive pulse from the 1024-μs interval timer. This is the same signal that is used internally to generate the 1024-μs timer interrupt. This signal is not gated by the interrupt enable state.
TIO1(P0.6) when enabled outputs a positive pulse from the programmable interval timer. This is the same signal that is used internally to generate the programmable timer interval interrupt. This signal is not gated by the interrupt enable state
The P0.5/TIO0 and P0.6/TIO1 pins are individually configured with the P05CR (0x0A) and P06CR (0x0B), respectively

Table 14-9. P0.7 Configuration (P07CR) [0x0C] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable
Read/Write	–	R/W	R/W	R/W	–	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register controls the operation of pin P0.7. The P0.7 pin only exists in the CY7C638(2/3)3

Table 14-10. P1.0/D+ Configuration (P10CR) [0x0D] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	Reserved			PS/2 Pull-up Enable	Output Enable
Read/Write	R/W	R/W	R/W	–	–	–	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register controls the operation of the P1.0 (D+) pin when the USB interface is not enabled, allowing the pin to be used as a PS2 interface or a GPIO. See *Table 21-1* for information on enabling USB. When USB is enabled, none of the controls in this register have any affect on the P1.0 pin.
Note: The P1.0 is an open drain only output. It can actively drive a signal low, but cannot actively drive a signal high.
Bit 1: PS/2 Pull-up Enable
0 = Disable the 5K ohm pull-up resistors
1 = Enable 5K ohm pull-up resistors for both P1.0 and P1.1. Enable the use of the P1.0 (D+) and P1.1 (D–) pins as a PS2 style interface

Table 14-11. P1.1/D– Configuration (P11CR) [0x0E] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	Reserved		Open Drain	Reserved	Output Enable
Read/Write	–	R/W	R/W	–	–	R/W	–	R/W
Default	0	0	0	0	0	0	0	0

This register controls the operation of the P1.1 (D–) pin when the USB interface is not enabled, allowing the pin to be used as a PS2 interface or a GPIO. See *Table 21-1* for information on enabling USB. When USB is enabled, none of the controls in this register have any affect on the P1.1 pin. When USB is disabled, the 5K ohm pull-up resistor on this pin can be enabled by the PS/2 Pull-up Enable bit of the P10CR Register (*Table 14-10*)
Note: There is no 2 mA sourcing capability on this pin. The pin can only sink 5 mA at V_{OL3} (Section 26.0)

Table 14-12. P1.2 Configuration (P12CR) [0x0F] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	CLK Output	Int Enable	Int Act Low	TTL Threshold	Reserved	Open Drain	Pull-up Enable	Output Enable
Read/Write	R/W	R/W	R/W	R/W	–	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register controls the operation of the P1.2
Bit 7: CLK Output
0 = The internally selected clock is not sent out onto P1.2 pin
1 = When CLK Output is set, the internally selected clock is sent out onto P1.2 pin

Table 14-13. P1.3 Configuration (P13CR) [0x10] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	3.3V Drive	High Sink	Open Drain	Pull-up Enable	Output Enable
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register controls the operation of the P1.3 pin. This register exists in all enCoRe II parts
The P1.3 GPIO’s threshold is always set to TTL
When the SPI hardware is enabled, the output enable and output state of the pin is controlled by the SPI circuitry. When the SPI hardware is disabled, the pin is controlled by the Output Enable bit and the corresponding bit in the P1 data register.
Regardless of whether the pin is used as an SPI or GPIO pin the Int Enable, Int act Low, 3.3V Drive, High Sink, Open Drain, and Pull-up Enable control the behavior of the pin
The 50 mA sink drive capability is only available in the CY7C638xx.

Table 14-14. P1.4–P1.6 Configuration (P14CR–P16CR) [0x11–0x13] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	SPI Use	Int Enable	Int Act Low	3.3V Drive	High Sink	Open Drain	Pull-up Enable	Output Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

These registers control the operation of pins P1.4–P1.6, respectively. These registers exist in all enCoRe II parts
The P1.4–P1.6 GPIO’s threshold is always set to TTL
When the SPI hardware is enabled, pins that are configured as SPI Use have their output enable and output state controlled by the SPI circuitry. When the SPI hardware is disabled or a pin has its SPI Use bit clear, the pin is controlled by the Output Enable bit and the corresponding bit in the P1 data register.
Regardless of whether any pin is used as an SPI or GPIO pin the Int Enable, Int act Low, 3.3V Drive, High Sink, Open Drain, and Pull-up Enable control the behavior of the pin
The 50 mA sink drive capability is only available in the CY7C638xx.
Bit 7: SPI Use
0 = Disable the SPI alternate function. The pin is used as a GPIO
1 = Enable the SPI function. The SPI circuitry controls the output of the pin

Important Note for Comm Modes 01 or 10 (SPI Master or SPI Slave, see *Table 15-2***):**
When configured for SPI (SPI Use = 1 and Comm Modes [1:0] = SPI Master or SPI Slave mode), the input/output direction of pins P1.3, P1.5, and P1.6 is set automatically by the SPI logic. However, pin P1.4's input/output direction is NOT automatically set; it must be explicitly set by firmware. For SPI Master mode, pin P1.4 must be configured as an output; for SPI Slave mode, pin P1.4 must be configured as an input.

Table 14-15. P1.7 Configuration (P17CR) [0x14] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	1	0

This register controls the operation of pin P1.7. This register only exists in CY7C638xx
The 50 mA sink drive capability is only available in the CY7C638xx. The P1.7 GPIO’s threshold is always set to TTL

Table 14-16. P2 Configuration (P2CR) [0x15] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
This register only exists in CY7C638xx. This register controls the operation of pins P2.0–P2.1. In the CY7C638xx, only 8 mA sink drive capability is available on this pin regardless of the setting of the High Sink bit								

Table 14-17. P3 Configuration (P3CR) [0x16] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	1	0
This register exists in CY7C638xx. This register controls the operation of pins P3.0–P3.1. In the CY7C638xx, only 8 mA sink drive capability is available on this pin regardless of the setting of the High Sink bit								

15.0 Serial Peripheral Interface (SPI)

The SPI Master/Slave Interface core logic runs on the SPI clock domain, making its functionality independent of system clock speed. SPI is a four pin serial interface comprised of a clock, an enable and two data pins.

15.1 SPI Data Register

Table 15-1. SPI Data Register (SPIDATA) [0x3C] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	SPIData[7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
When read, this register returns the contents of the receive buffer. When written, it loads the transmit holding register **Bit [7:0]:** SPI Data [7:0]								

When an interrupt occurs to indicate to firmware that a byte of receive data is available, or the transmitter holding register is empty, firmware has 7 SPI clocks to manage the buffers—to empty the receiver buffer, or to refill the transmit holding register. Failure to meet this timing requirement will result in incorrect data transfer.

15.2 SPI Configure Register

Table 15-2. SPI Configure Register (SPICR) [0x3D] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Swap	LSB First	Comm Mode		CPOL	CPHA	SCLK Select	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit 7: Swap

0 = Swap function disabled

1 = The SPI block swaps its use of SMOSI and SMISO. Among other things, this can be useful in implementing single wire SPI-like communications

Bit 6: LSB First

0 = The SPI transmits and receives the MSB (Most Significant Bit) first

1 = The SPI transmits and receives the LSB (Least Significant Bit) first.

Bit [5:4]: Comm Mode [1:0]

0 0: All SPI communication disabled

0 1: SPI master mode

1 0: SPI slave mode

1 1: Reserved

Bit 3: CPOL

This bit controls the SPI clock (SCLK) idle polarity

0 = SCLK idles low

1 = SCLK idles high

Bit 2: CPHA

The Clock Phase bit controls the phase of the clock on which data is sampled. *Table 15-3* below shows the timing for the various combinations of LSB First, CPOL, and CPHA

Bit [1:0]: SCLK Select

This field selects the speed of the master SCLK. When in master mode, SCLK is generated by dividing the base CPUCLK

Important Note for Comm Modes 01b or 10b (SPI Master or SPI Slave):
When configured for SPI, (SPI Use = 1—*Table 14-14*), the input/output direction of pins P1.3, P1.5, and P1.6 is set automatically by the SPI logic. However, pin P1.4's input/output direction is NOT automatically set; it must be explicitly set by firmware. For SPI Master mode, pin P1.4 must be configured as an output; for SPI Slave mode, pin P1.4 must be configured as an input.

15.3 SPI Interface Pins

The SPI interface uses the P1.3–P1.6 pins. These pins are configured using the P1.3 and P1.4–P1.6 Configuration.

Table 15-3. SPI Mode Timing vs. LSB First, CPOL and CPHA

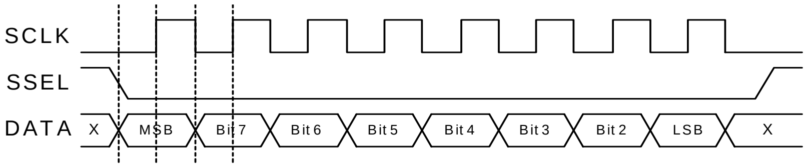
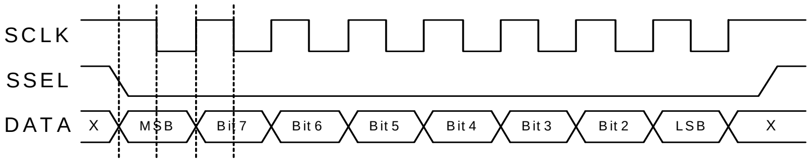
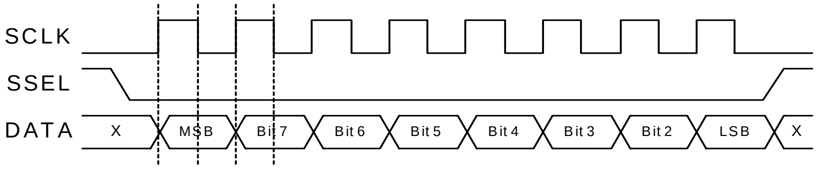
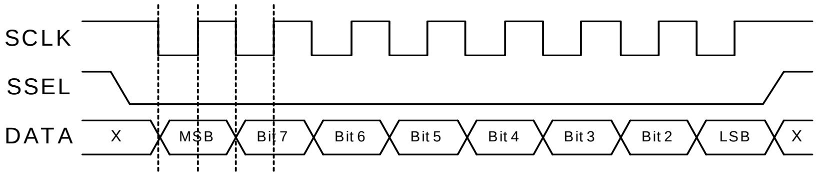
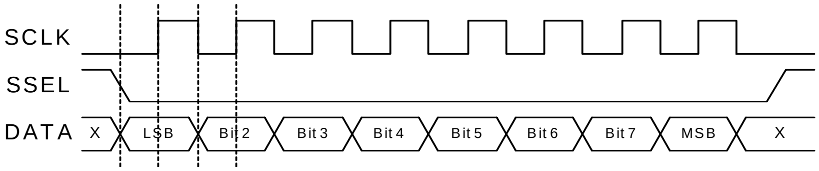
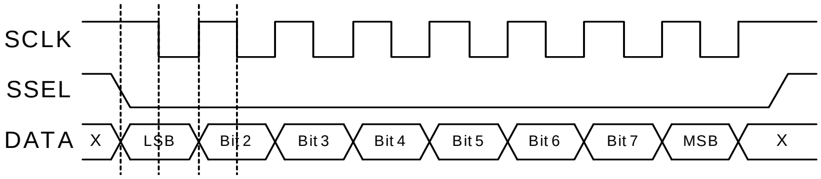
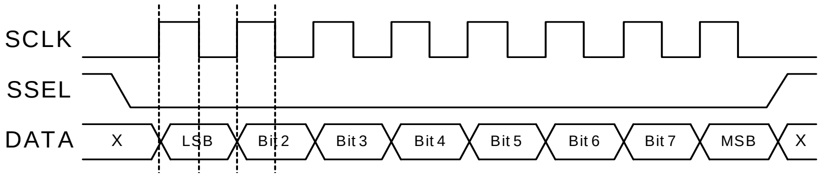
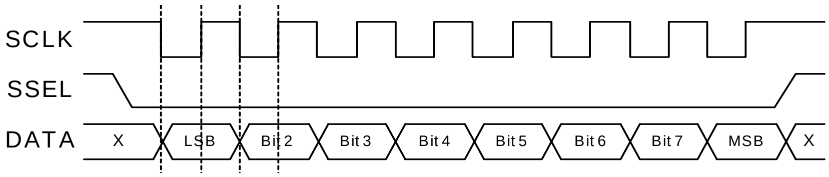
LSB First	CPHA	CPOL	Diagram
0	0	0	 SCLK SSEL DATA X MSB Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 LSB X
0	0	1	 SCLK SSEL DATA X MSB Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 LSB X
0	1	0	 SCLK SSEL DATA X MSB Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 LSB X
0	1	1	 SCLK SSEL DATA X MSB Bit 7 Bit 6 Bit 5 Bit 4 Bit 3 Bit 2 LSB X
1	0	0	 SCLK SSEL DATA X LSB Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 Bit 7 MSB X
1	0	1	 SCLK SSEL DATA X LSB Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 Bit 7 MSB X
1	1	0	 SCLK SSEL DATA X LSB Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 Bit 7 MSB X
1	1	1	 SCLK SSEL DATA X LSB Bit 2 Bit 3 Bit 4 Bit 5 Bit 6 Bit 7 MSB X

Table 15-4. SPI SCLK Frequency

SCLK Select	CPUCLK Divisor	SCLK Frequency when CPUCLK =	
		12 MHz	24 MHz
00	6	2 MHz	4 MHz
01	12	1 MHz	2 MHz
10	48	250 KHz	500 KHz
11	96	125 KHz	250 KHz

16.0 Timer Registers

All timer functions of the enCoRe II are provided by a single timer block. The timer block is asynchronous from the CPU clock.

16.1 Registers

16.1.1 Free Running Counter

The 16 bit free-running counter is clocked by the Timer Capture Clock (TCAPCLK). It can be read in software for use as a general-purpose time base. When the low-order byte is read, the high-order byte is registered. Reading the high-order byte reads this register allowing the CPU to read the 16-bit value atomically (loads all bits at one time). The free-running timer generates an interrupt at 1024-µs rate when clocked by a 4MHz source. It can also generate an interrupt when the free-running counter overflow occurs—every 16.384 ms(with a 4MHz source). This allows extending the length of the timer in software..

Figure 16-1. 16-bit Free Running Counter Block Diagram

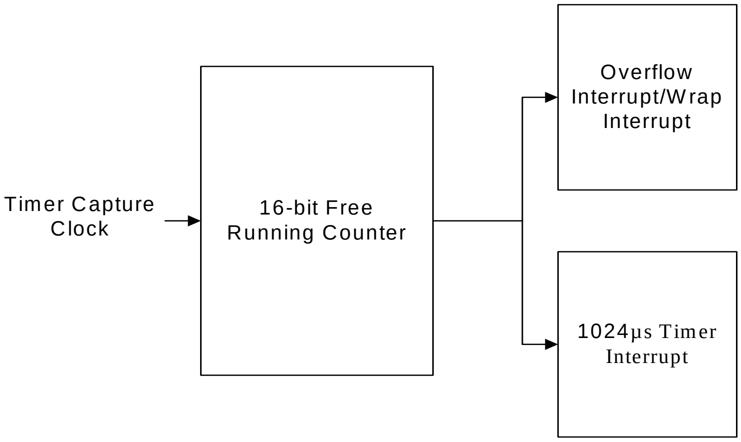

Overflow Interrupt/Wrap Interrupt
Timer Capture Clock
16-bit Free Running Counter
1024µs Timer Interrupt

Table 16-1. Free-running Timer Low-order Byte (FRTMRL) [0x20] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Free-running Timer [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
Bit [7:0]: Free-running Timer [7:0] This register holds the low-order byte of the 16-bit free-running timer. Reading this register causes the high-order byte to be moved into a holding register allowing an automatic read of all 16 bits simultaneously. For reads, the actual read occurs in the cycle when the low order is read. For writes, the actual time the write occurs is the cycle when the high order is written. When reading the Free Running Timer, the low-order byte must be read first and the high-order second. When writing, the low-order byte must be written first then the high-order byte								

Table 16-2. Free-running Timer High-order Byte (FRTMRH) [0x21] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Free-running Timer [15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
Bit [7:0]: Free-running Timer [15:8] When reading the Free-running Timer, the low-order byte must be read first and the high-order second. When writing, the low-order byte must be written first then the high-order byte								

Table 16-3. Timer Capture 0 Rising (TCAP0R) [0x22] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 0 Rising [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
Bit [7:0]: Capture 0 Rising [7:0] This register holds the value of the Free-running Timer when the last rising edge occurred on the TCAP0 input. When Capture 0 is in 8-bit mode, the bits that are stored here are selected by the Prescale [2:0] bits in the Timer Configuration register. When Capture 0 is in 16-bit mode this register holds the lower order 8 bits of the 16-bit timer								

Table 16-4. Timer Capture 1 Rising (TCAP1R) [0x23] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 1 Rising [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
Bit [7:0]: Capture 1 Rising [7:0] This register holds the value of the Free-running Timer when the last rising edge occurred on the TCAP1 input. The bits that are stored here are selected by the Prescale [2:0] bits in the Timer Configuration register. When Capture 0 is in 16-bit mode this register holds the high-order 8 bits of the 16-bit timer from the last Capture 0 rising edge. When Capture 0 is in 16-bit mode this register will be loaded with high-order 8 bits of the 16-bit timer on TCAP0 rising edge								

Table 16-5. Timer Capture 0 Falling (TCAP0F) [0x24] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 0 Falling [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
Bit [7:0]: Capture 0 Falling [7:0] This register holds the value of the Free-running Timer when the last falling edge occurred on the TCAP0 input. When Capture 0 is in 8-bit mode, the bits that are stored here are selected by the Prescale [2:0] bits in the Timer Configuration register. When Capture 0 is in 16-bit mode this register holds the lower-order 8 bits of the 16-bit timer								

Table 16-6. Timer Capture 1 Falling (TCAP1F) [0x25] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 1 Falling [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
Bit [7:0]: Capture 1Falling [7:0] This register holds the value of the Free-running Timer when the last falling edge occurred on the TCAP1 input. The bits that are stored here are selected by the Prescale [2:0] bits in the Timer Configuration register. When capture 0 is in 16-bit mode this register holds the high-order 8 bits of the 16-bit timer from the last Capture 0 falling edge. When Capture 0 is in 16-bit mode this register will be loaded with high-order 8 bits of the 16-bit timer on TCAP0 falling edge								

Table 16-7. Programmable Interval Timer Low (PITMRL) [0x26] [R]

Bit #	7	6	5	4	3	2	1	0
Field	Prog Interval Timer [7:0]							
Read/Write	R	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

Bit [7:0]: Prog Interval Timer [7:0]
This register holds the low-order byte of the 12-bit programmable interval timer. Reading this register causes the high-order byte to be moved into a holding register allowing an automatic read of all 12 bits simultaneously

Table 16-8. Programmable Interval Timer High (PITMRH) [0x27] [R]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved				Prog Interval Timer [11:8]			
Read/Write	–	–	–	–	R	R	R	R
Default	0	0	0	0	0	0	0	0

Bit [7:4]: Reserved
Bit [3:0]: Prog Internal Timer [11:8]
This register holds the high-order nibble of the 12-bit programmable interval timer. Reading this register returns the high-order nibble of the 12-bit timer at the instant that the low-order byte was last read

Table 16-9. Programmable Interval Reload Low (PIRL) [0x28] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Prog Interval [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:0]: Prog Interval [7:0]
This register holds the lower 8 bits of the timer. While writing into the 12-bit reload register, write lower byte first then the higher nibble

Table 16-10. Programmable Interval Reload High (PIRH) [0x29] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved				Prog Interval[11:8]			
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:4]: Reserved
Bit [3:0]: Prog Interval [11:8]
This register holds the higher 4 bits of the timer. While writing into the 12-bit reload register, write lower byte first then the higher nibble

16.1.2 Timer Capture

enCoRe II has two 8-bit captures. Each capture has separate registers for the rising and falling time. The two eight bit captures can be configured as a single 16-bit capture. When configured in this way, the capture 1 registers hold the high order byte of the 16-bit timer capture value. Each of the four capture registers can be programmed to generatean interrupt when it is loaded.

Figure 16-2. Programmable Interval Timer Block Diagram

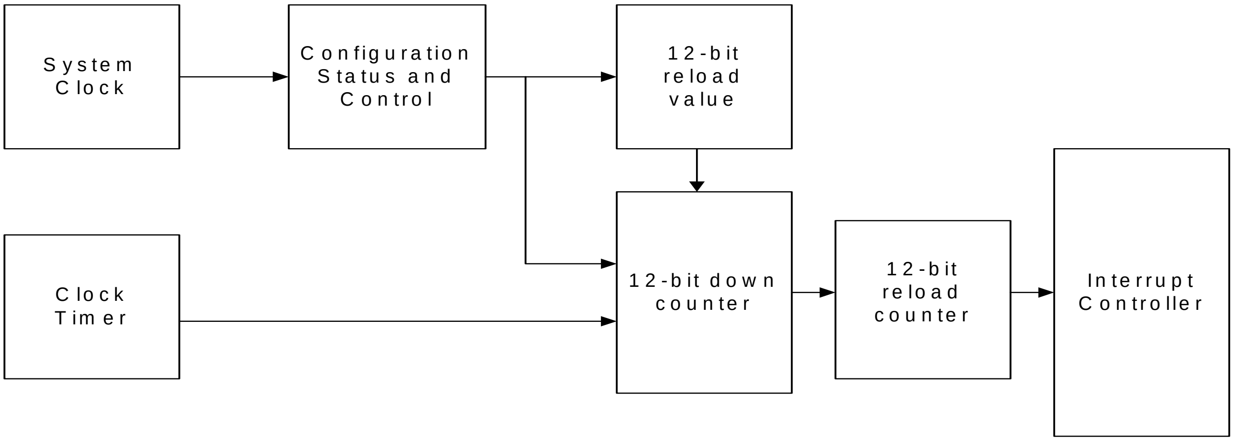


Table 16-11. Timer Configuration (TMRCR) [0x2A] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	First Edge Hold	8-bit Capture Prescale [2:0]			Cap0 16bit Enable	Reserved		
Read/Write	R/W	R/W	R/W	R/W	R/W	–	–	–
Default	0	0	0	0	0	0	0	0

Bit 7: First Edge Hold
The First Edge Hold function applies to all four capture timers.
0 = The time of the most recent edge is held in the Capture Timer Data Register. If multiple edges have occurred since reading the capture timer, the time for the most recent one will be read
1 = The time of the first occurrence of an edge is held in the Capture Timer Data Register until the data is read. Subsequent edges are ignored until the Capture Timer Data Register is read.
Bit [6:4]: 8-bit Capture Prescale [2:0]
This field controls which 8 bits of the 16 Free Running Timer are captured when in bit mode
0 0 0 = capture timer[7:0]
0 0 1 = capture timer[8:1]
0 1 0 = capture timer[9:2]
0 1 1 = capture timer[10:3]
1 0 0 = capture timer[11:4]
1 0 1 = capture timer[12:5]
1 1 0 = capture timer[13:6]
1 1 1 = capture timer[14:7]
Bit 3: Cap0 16-bit Enable
0 = Capture 0 16-bit mode is disabled
1 = Capture 0 16-bit mode is enabled. Capture 1 is disabled and the Capture 1 rising and falling registers are used as an extension to the Capture 0 registers—extending them to 16 bits
Bit [2:0]: Reserved

Table 16-12. Capture Interrupt Enable (TCAPINTE) [0x2B] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved				Cap1 Fall Enable	Cap1 Rise Enable	Cap0 Fall Enable	Cap0 Rise Enable
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:4]: Reserved
Bit 3: Cap1 Fall Enable
0 = Disable the capture 1 falling edge interrupt
1 = Enable the capture 1 falling edge interrupt
Bit 2: Cap1 Rise Enable
0 = Disable the capture 1 rising edge interrupt
1 = Enable the capture 1 rising edge interrupt
Bit 1: Cap0 Fall Enable
0 = Disable the capture 0 falling edge interrupt
1 = Enable the capture 0 falling edge interrupt
Bit 0: Cap0 Rise Enable
0 = Disable the capture 0 rising edge interrupt
1 = Enable the capture 0 rising edge interrupt

Table 16-13. Capture Interrupt Status (TCAPINTS) [0x2C] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved				Cap1 Fall Active	Cap1 Rise Active	Cap0 Fall Active	Cap0 Rise Active
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:4]: Reserved
Bit 3: Cap1 Fall Active
0 = No event
1 = A falling edge has occurred on Cap1
Bit 2: Cap1 Rise Active
0 = No event
1 = A rising edge has occurred on Cap1
Bit 1: Cap0 Fall Active
0 = No event
1 = A falling edge has occurred on Cap0
Bit 0: Cap0 Rise Active
0 = No event
1 = A rising edge has occurred on Cap0
Note: The interrupt status bits must be cleared by firmware to enable subsequent interrupts. This is achieved by writing a ‘1’ to the corresponding Interrupt status bit.

Figure 16-3. Timer Functional Timing Diagram

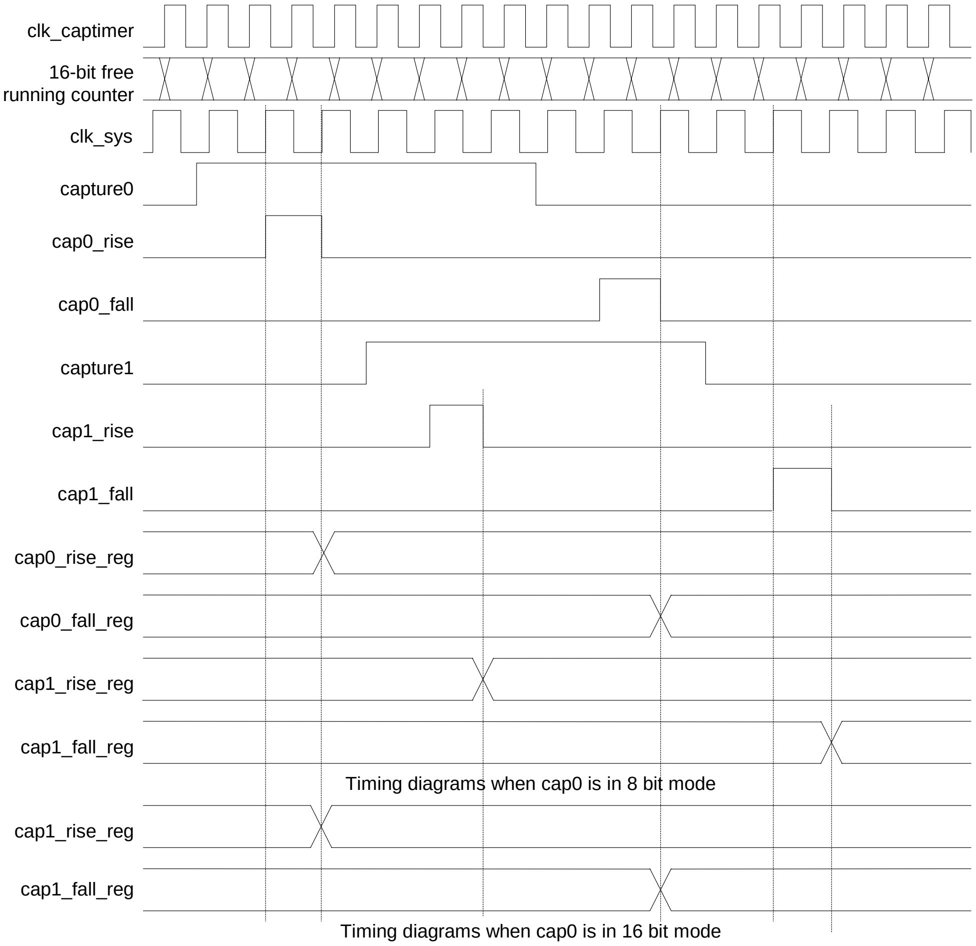


Figure 16-4. 16-bit Free Running Counter Loading Timing Diagram

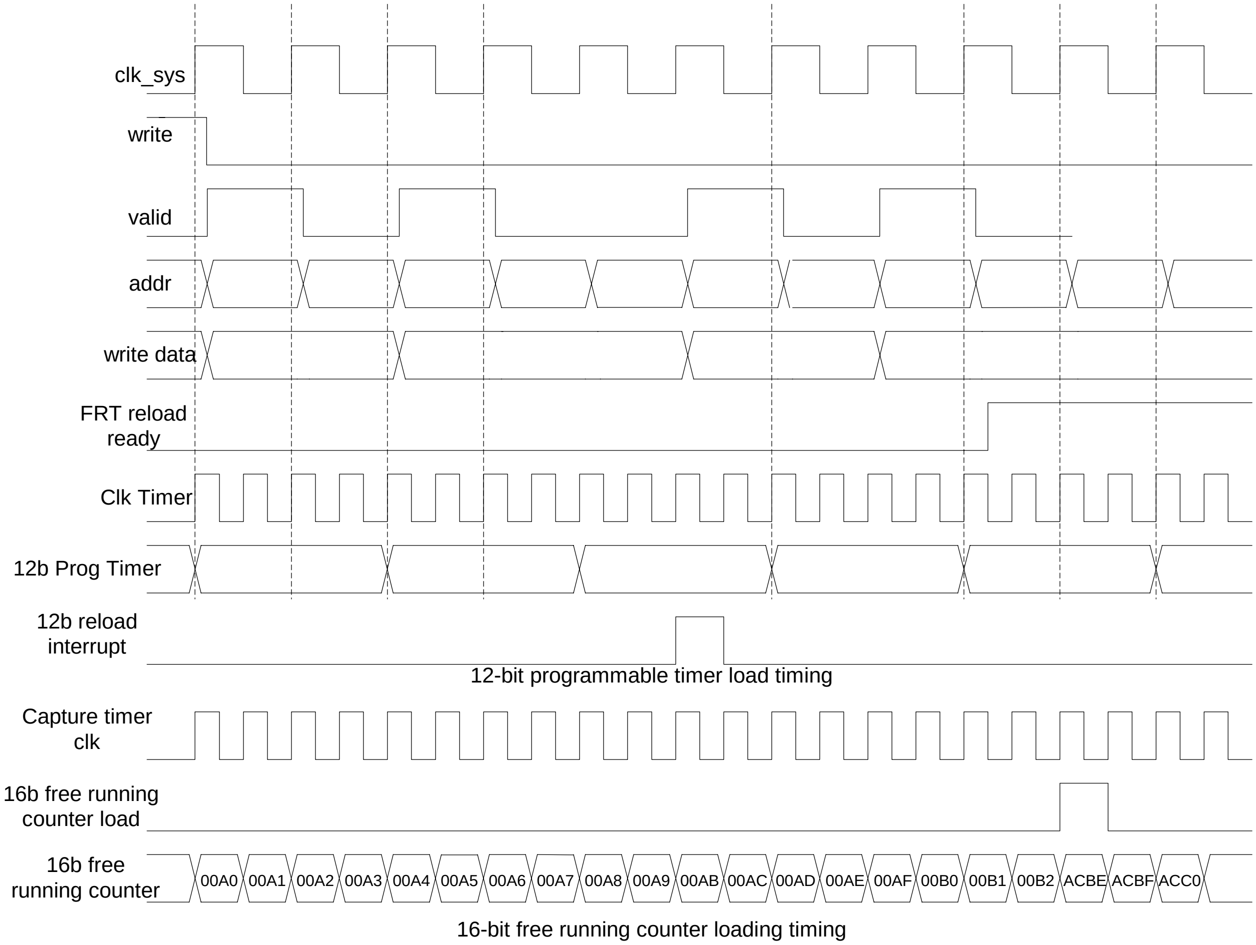
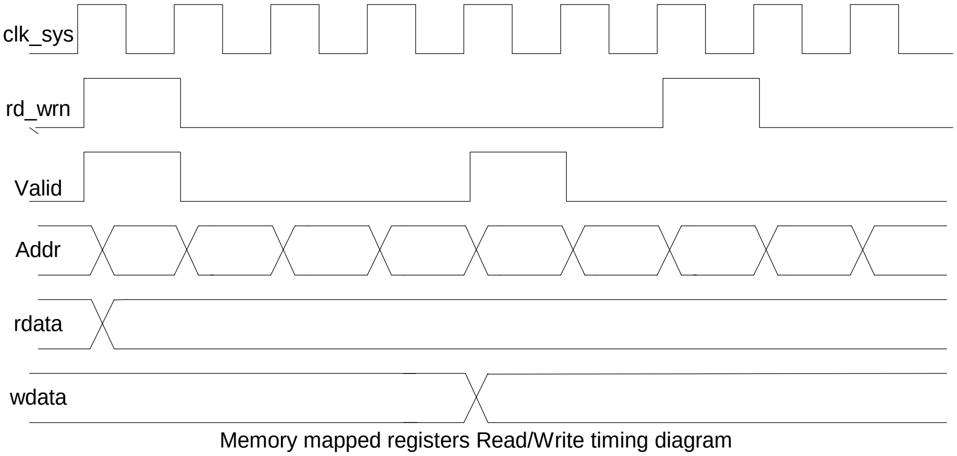


Figure 16-5. Memory Mapped Registers Read/Write Timing Diagram



17.0 Interrupt Controller

The interrupt controller and its associated registers allow the user’s code to respond to an interrupt from almost every functional block in the enCoRe II devices. The registers associated with the interrupt controller allow interrupts to be disabled either globally or individually. The registers also provide a mechanism by which a user may clear all pending and posted interrupts, or clear individual posted or pending interrupts.

The following table lists all interrupts and the priorities that are available in the enCoRe II devices.

Table 17-1. Interrupt Numbers, Priorities, Vectors

Interrupt Priority	Interrupt Address	Name
0	0000h	Reset
1	0004h	POR/LVD
2	0008h	INT0
3	000Ch	SPI Transmitter Empty
4	0010h	SPI Receiver Full
5	0014h	GPIO Port 0
6	0018h	GPIO Port 1
7	001Ch	INT1
8	0020h	EP0
9	0024h	EP1
10	0028h	EP2
11	002Ch	USB Reset
12	0030h	USB Active
13	0034h	1 mS Interval timer
14	0038h	Programmable Interval Timer
15	003Ch	Timer Capture 0
16	0040h	Timer Capture 1
17	0044h	16-bit Free Running Timer Wrap
18	0048h	INT2
19	004Ch	PS2 Data Low
20	0050h	GPIO Port 2
21	0054h	GPIO Port 3
22	0058h	Reserved
23	005Ch	Reserved
24	0060h	Reserved
25	0064h	Sleep Timer

17.1 Architectural Description

An interrupt is posted when its interrupt conditions occur. This results in the flip-flop in *Figure 17-1* clocking in a ‘1’. The interrupt will remain posted until the interrupt is taken or until it is cleared by writing to the appropriate INT_CLRx register.

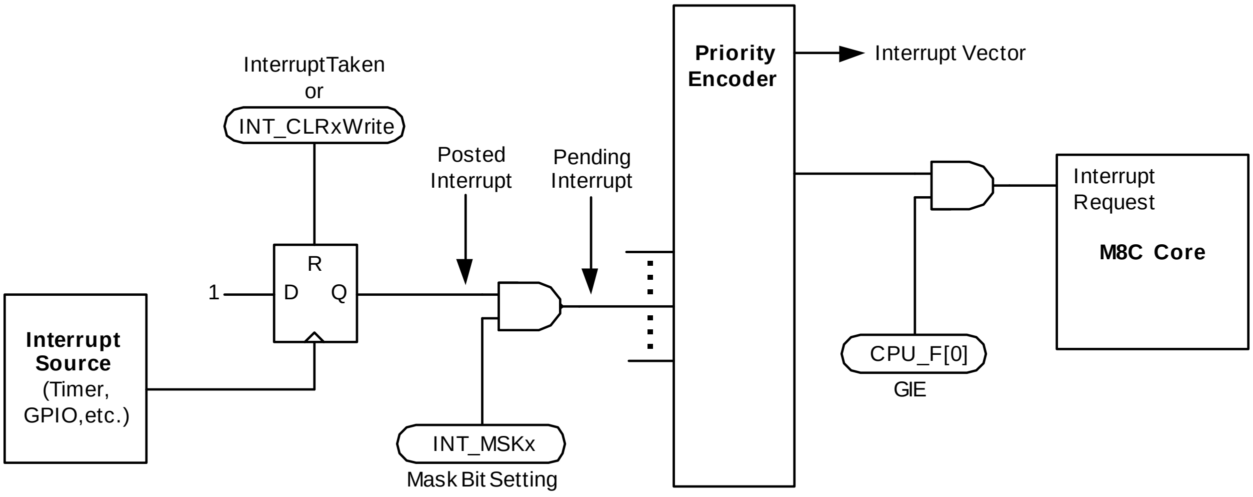
A posted interrupt is not pending unless it is enabled by setting its interrupt mask bit (in the appropriate INT_MSKx register). All pending interrupts are processed by the Priority Encoder to determine the highest priority interrupt which will be taken by the M8C if the Global Interrupt Enable bit is set in the CPU_F register.

Disabling an interrupt by clearing its interrupt mask bit (in the INT_MSKx register) does not clear a posted interrupt, nor does it prevent an interrupt from being posted. It simply prevents a posted interrupt from becoming pending.

Nested interrupts can be accomplished by re-enabling interrupts inside an interrupt service routine. To do this, set the IE bit in the Flag Register.

A block diagram of the enCoRe II Interrupt Controller is shown in *Figure 17-1*.

Figure 17-1. Interrupt Controller Block Diagram



17.2 Interrupt Processing

The sequence of events that occur during interrupt processing is as follows:

1. An interrupt becomes active, either because:
   a. The interrupt condition occurs (e.g., a timer expires)
   b. A previously posted interrupt is enabled through an update of an interrupt mask register
   c. An interrupt is pending and GIE is set from 0 to 1 in the CPU Flag register.
2. The current executing instruction finishes.
3. The internal interrupt is dispatched, taking 13 cycles. During this time, the following actions occur: he MSB and LSB of Program Counter and Flag registers (CPU_PC and CPU_F) are stored onto the program stack by an automatic CALL instruction (13 cycles) generated during the interrupt acknowledge process.
   a. The PCH, PCL, and Flag register (CPU_F) are stored onto the program stack (in that order) by an automatic CALL instruction (13 cycles) generated during the interrupt acknowledge process
   b. The CPU_F register is then cleared. Since this clears the GIE bit to 0, additional interrupts are temporarily disabled
   c. The PCH (PC[15:8]) is cleared to zero
   d. The interrupt vector is read from the interrupt controller and its value placed into PCL (PC[7:0]). This sets the program counter to point to the appropriate address in the interrupt table (e.g., 0004h for the POR/LVD interrupt)
4. Program execution vectors to the interrupt table. Typically, a LJMP instruction in the interrupt table sends execution to the user's Interrupt Service Routine (ISR) for this interrupt
5. The ISR executes. Note that interrupts are disabled since GIE = 0. In the ISR, interrupts can be re-enabled if desired by setting GIE = 1 (care must be taken to avoid stack overflow).
6. The ISR ends with a RETI instruction which restores the Program Counter and Flag registers (CPU_PC and CPU_F). The restored Flag register re-enables interrupts, since GIE = 1 again.
7. Execution resumes at the next instruction, after the one that occurred before the interrupt. However, if there are more pending interrupts, the subsequent interrupts will be processed before the next normal program instruction.

17.3 Interrupt Trigger Conditions

Trigger conditions for most interrupts in Table 17-1.have been explained in the relevant sections. However, conditions under which the USB Active (interrupt address 0030h) and PS2 Data Low (interrupt address 004Ch) interrupts are triggered are explained below.

1. USB Active Interrupt: Triggered when the D+/- lines are in a non-idle state i.e., K-state or SE0 state
2. PS2 Data Low Interrupt: Triggered when SDATA becomes low when the SDATA pad is in the input mode for at least 6-7 32 KHz cycles.

17.4 Interrupt Latency

The time between the assertion of an enabled interrupt and the start of its ISR can be calculated from the following equation.

Latency = Time for current instruction to finish + Time for internal interrupt routine to execute + Time for LJMP instruction in interrupt table to execute.

For example, if the 5-cycle JMP instruction is executing when an interrupt becomes active, the total number of CPU clock cycles before the ISR begins would be as follows:

(1 to 5 cycles for JMP to finish) + (13 cycles for interrupt routine) + (7 cycles for LJMP) = 21 to 25 cycles.

In the example above, at 24 MHz, 25 clock cycles take 1.042 μs.

17.5 Interrupt Registers

The Interrupt Clear Registers (INT_CLRx) are used to enable the individual interrupt sources' ability to clear posted interrupts.

When an INT_CLRx register is read, any bits that are set indicates an interrupt has been posted for that hardware resource. Therefore, reading these registers gives the user the ability to determine all posted interrupts.

Table 17-2. Interrupt Clear 0 (INT_CLR0) [0xDA] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	GPIO Port 1	Sleep Timer	INT1	GPIO Port 0	SPI Receive	SPI Transmit	INT0	POR/LVD
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
When reading this register, 0 = There's no posted interrupt for the corresponding hardware 1 = Posted interrupt for the corresponding hardware present Writing a '0' to the bits will clear the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT (Bit 7 of the INT_MSK3 Register) will post the corresponding hardware interrupt								

Table 17-3. Interrupt Clear 1 (INT_CLR1) [0xDB] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	TCAP0	Prog Interval Timer	1-ms Timer	USB Active	USB Reset	USB EP2	USB EP1	USB EP0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 17-3. Interrupt Clear 1 (INT_CLR1) [0xDB] [R/W] (continued)

Default	0	0	0	0	0	0	0	0
When reading this register, 0 = There’s no posted interrupt for the corresponding hardware 1 = Posted interrupt for the corresponding hardware present Writing a ‘0’ to the bits will clear the posted interrupts for the corresponding hardware. Writing a 57‘1’ to the bits AND to the ENSWINT (Bit 7 of the INT_MSK3 Register) will post the corresponding hardware interrupt								

Table 17-4. Interrupt Clear 2 (INT_CLR2) [0xDC] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	Reserved	GPIO Port 3	GPIO Port 2	PS/2 Data Low	INT2	16-bit Counter Wrap	TCAP1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0
When reading this register, 0 = There’s no posted interrupt for the corresponding hardware 1 = Posted interrupt for the corresponding hardware present Writing a ‘0’ to the bits will clear the posted interrupts for the corresponding hardware. Writing a ‘1’ to the bits AND to the ENSWINT (Bit 7 of the INT_MSK3 Register) will post the corresponding hardware interrupt								

17.5.1 Interrupt Mask Registers

The Interrupt Mask Registers (INT_MSKx) are used to enable the individual interrupt sources’ ability to create pending interrupts.

There are four Interrupt Mask Registers (INT_MSK0, INT_MSK1, INT_MSK2, and INT_MSK3) which may be referred to in general as INT_MSKx. If cleared, each bit in an INT_MSKx register prevents a posted interrupt from becoming a pending interrupt (input to the priority encoder). However, an interrupt can still post even if its mask bit is zero. All INT_MSKx bits are independent of all other INT_MSKx bits.

If an INT_MSKx bit is set, the interrupt source associated with that mask bit may generate an interrupt that will become a pending interrupt.

The Enable Software Interrupt (ENSWINT) bit in INT_MSK3[7] determines the way an individual bit value written to an INT_CLRx register is interpreted. When is cleared, writing 1's to an INT_CLRx register has no effect. However, writing 0's to an INT_CLRx register, when ENSWINT is cleared, will cause the corresponding interrupt to clear. If the ENSWINT bit is set, any 0s written to the INT_CLRx registers are ignored. However, 1s written to an INT_CLRx register, while ENSWINT is set, will cause an interrupt to post for the corresponding interrupt.

Software interrupts can aid in debugging interrupt service routines by eliminating the need to create system level interactions that are sometimes necessary to create a hardware-only interrupt.

Table 17-5. Interrupt Mask 3 (INT_MSK3) [0xDE] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ENSWINT	Reserved						
Read/Write	R/W	–	–	–	–	–	–	–
Default	0	0	0	0	0	0	0	0
Bit 7: Enable Software Interrupt (ENSWINT) 0= Disable. Writing 0s to an INT_CLRx register, when ENSWINT is cleared, will cause the corresponding interrupt to clear 1= Enable. Writing 1s to an INT_CLRx register, when ENSWINT is set, will cause the corresponding interrupt to post. **Bit [6:0]:** Reserved								

Table 17-6. Interrupt Mask 2 (INT_MSK2) [0xDF] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Reserved	Reserved	GPIO Port 3 Int Enable	GPIO Port 2 Int Enable	PS/2 Data Low Int Enable	INT2 Int Enable	16-bit Counter Wrap Int Enable	TCAP1 Int Enable
Read/Write	–	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Table 17-6. Interrupt Mask 2 (INT_MSK2) [0xDF] [R/W] (continued)

Bit 7: Reserved
Bit 6: GPIO Port 4 Interrupt Enable
0 = Mask GPIO Port 4 interrupt
1 = Unmask GPIO Port 4 interrupt
Bit 5: GPIO Port 3 Interrupt Enable
0 = Mask GPIO Port 3 interrupt
1 = Unmask GPIO Port 3 interrupt
Bit 4: GPIO Port 2 Interrupt Enable
0 = Mask GPIO Port 2 interrupt
1 = Unmask GPIO Port 2 interrupt
Bit 3: PS/2 Data Low Interrupt Enable
0 = Mask PS/2 Data Low interrupt
1 = Unmask PS/2 Data Low interrupt
Bit 2: INT2 Interrupt Enable
0 = Mask INT2 interrupt
1 = Unmask INT2 interrupt
Bit 1: 16-bit Counter Wrap Interrupt Enable
0 = Mask 16-bit Counter Wrap interrupt
1 = Unmask 16-bit Counter Wrap interrupt
Bit 0: TCAP1 Interrupt Enable
0 = Mask TCAP1 interrupt
1 = Unmask TCAP1 interrupt

Table 17-7. Interrupt Mask 1 (INT_MSK1) [0xE1] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	TCAP0 Int Enable	Prog Interval Timer Int Enable	1-ms Timer Int Enable	USB Active Int Enable	USB Reset Int Enable	USB EP2 Int Enable	USB EP1 Int Enable	USB EP0 Int Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit 7: TCAP0 Interrupt Enable
0 = Mask TCAP0 interrupt
1 = Unmask TCAP0 interrupt
Bit 6: Prog Interval Timer Interrupt Enable
0 = Mask Prog Interval Timer interrupt
1 = Unmask Prog Interval Timer interrupt
Bit 5: 1-ms Timer Interrupt Enable
0 = Mask 1-ms interrupt
1 = Unmask 1-ms interrupt
Bit 4: USB Active Interrupt Enable
0 = Mask USB Active interrupt
1 = Unmask USB Active interrupt
Bit 3: USB Reset Interrupt Enable
0 = Mask USB Reset interrupt
1 = Unmask USB Reset interrupt
Bit 2: USB EP2 Interrupt Enable
0 = Mask EP2 interrupt
1 = Unmask EP2 interrupt
Bit 1: USB EP1 Interrupt Enable
0 = Mask EP1 interrupt
1 = Unmask EP1 interrupt
Bit 0: USB EP0 Interrupt Enable
0 = Mask EP0 interrupt
1 = Unmask EP0 interrupt

Table 17-8. Interrupt Mask 0 (INT_MSK0) [0xE0] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	GPIO Port 1 Int Enable	Sleep Timer Int Enable	INT1 Int Enable	GPIO Port 0 Int Enable	SPI Receive Int Enable	SPI Transmit Int Enable	INT0 Int Enable	POR/LVD Int Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Table 17-8. Interrupt Mask 0 (INT_MSK0) [0xE0] [R/W]

Bit 7: GPIO Port 1 Interrupt Enable
0 = Mask GPIO Port 1 interrupt
1 = Unmask GPIO Port 1 interrupt
Bit 6: Sleep Timer Interrupt Enable
0 = Mask Sleep Timer interrupt
1 = Unmask Sleep Timer interrupt
Bit 5: INT1 Interrupt Enable
0 = Mask INT1 interrupt
1 = Unmask INT1 interrupt
Bit 4: GPIO Port 0 Interrupt Enable
0 = Mask GPIO Port 0 interrupt
1 = Unmask GPIO Port 0 interrupt
Bit 3: SPI Receive Interrupt Enable
0 = Mask SPI Receive interrupt
1 = Unmask SPI Receive interrupt
Bit 2: SPI Transmit Interrupt Enable
0 = Mask SPI Transmit interrupt
1 = Unmask SPI Transmit interrupt
Bit 1: INT0 Interrupt Enable
0 = Mask INT0 interrupt
1 = Unmask INT0 interrupt
Bit 0: POR/LVD Interrupt Enable
0 = Mask POR/LVD interrupt
1 = Unmask POR/LVD interrupt

Table 17-9. Interrupt Vector Clear Register (INT_VC) [0xE2] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Pending Interrupt [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

The Interrupt Vector Clear Register (INT_VC) holds the interrupt vector for the highest priority pending interrupt when read, and when written will clear all pending interrupts
Bit [7:0]: Pending Interrupt [7:0]
8-bit data value holds the interrupt vector for the highest priority pending interrupt. Writing to this register will clear all pending interrupts.

18.0 USB/PS2 Transceiver

Although the USB transceiver has features to assist in interfacing to PS/2 these features are not controlled using these registers. These registers only control the USB interfacing features. PS/2 interfacing options are controlled by the D+/D– GPIO Configuration register (See *Table 14-2*).

18.1 USB Transceiver Configuration

Table 18-1. USB Transceiver Configure Register (USBXCR) [0x74] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	USB Pull-up Enable	Reserved						USB Force State
Read/Write	R/W	–	–	–	–	–	–	R/W
Default	0	0	0	0	0	0	0	0

Bit 7: USB Pull-up Enable

0 = Disable the pull-up resistor on D–

1 = Enable the pull-up resistor on D–. This pull up is to V_{CC} IF VREG is not enabled or to the internally generated 3.3V when VREG is enabled

Bit [6:1]: Reserved

Bit 0: USB Force State

This bit allows the state of the USB I/O pins D- and D+ to be forced to a state while USB is enabled

0 = Disable USB Force State
1 = Enable USB Force State. Allows the D- and D+ pins to be controlled by P1.1 and P1.0 respectively when the USBIO is in USB mode. Refer to *Table 14-2* for more information

Note: The USB transceiver has a dedicated 3.3V regulator for USB signalling purposes and to provide for the 1.5K D-pull-up. Unlike the other 3.3V regulator, this regulator cannot be controlled/accessed by firmware. When the device is suspended, this regulator is disabled along with the bandgap (which provides the reference voltage to the regulator) and the D-line is pulled up to 5V through an alternate 6.5K resistor. During wake up following a suspend, the band gap and the regulator are switched on in any order. Under an extremely rare case when the device wakes up following a bus reset condition and the voltage regulator and the band gap turn on in that particular order, there is possibility of a glitch/low pulse occurring on the D- line. The host can misinterpret this as a deattach condition. This condition, although rare, can be avoided by keeping the bandgap circuitry enabled during sleep. This is achieved by setting the ‘No Buzz’ bit, bit[5] in the OSC_CR0 register. This is an issue only if the device is put to sleep during a bus reset condition.

19.0 USB Regulator Output

19.1 VREG Control

Table 19-1. VREG Control Register (VREGCR) [0x73] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved						Keep Alive	VREG Enable
Read/Write	–	–	–	–	–	–	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:2]: Reserved

Bit 1: Keep Alive

Keep Alive, when set, allows the voltage regulator to source up to 20 µA of current when the voltage regulator is disabled, P12CR[0],P12CR[7] must be cleared.

0 = Disabled

1 = Enabled

Bit 0: VREG Enable

This bit turns on the 3.3V voltage regulator. The voltage regulator only functions within specifications when V_{CC} is above 4.35V. This block must not be enabled when V_{CC} is below 4.35V—although no damage or irregularities will occur if it is enabled below 4.35V

0 = Disable the 3.3V voltage regulator output on the VREG/P1.2 pin
1 = Enable the 3.3V voltage regulator output on the VREG/P1.2 pin. GPIO functionality of P1.2 is disabled
Note: Use of the alternate drive on pins P1.3–P1.6 requires that the VREG Enable bit be set to enable the regulator and provide the alternate voltage

20.0 USB Serial Interface Engine (SIE)

The SIE allows the microcontroller to communicate with the USB host at low-speed data rates (1.5 Mbps). The SIE simplifies the interface between the microcontroller and USB by incorporating hardware that handles the following USB bus activity independently of the microcontroller:

- Translate the encoded received data and format the data to be transmitted on the bus.
- CRC checking and generation. Flag the microcontroller if errors exist during transmission.
- Address checking. Ignore the transactions not addressed to the device.
- Send appropriate ACK/NAK/STALL handshakes.
- Token type identification (SETUP, IN, or OUT). Set the appropriate token bit once a valid token is received.
- Place valid received data in the appropriate endpoint FIFOs.
- Send and update the data toggle bit (Data1/0).
- Bit stuffing/unstuffing.

Firmware is required to handle the rest of the USB interface with the following tasks:

- Coordinate enumeration by decoding USB device requests.
- Fill and empty the FIFOs.
- Suspend/Resume coordination.
- Verify and select Data toggle values.

21.0 USB Device

21.1 USB Device Address

Table 21-1. USB Device Address (USBCR) [0x40] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	USB Enable	Device Address[6:0]						
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

The content of this register is cleared when a USB Bus Reset condition occurs

Bit 7: USB Enable

This bit must be enabled by firmware before the serial interface engine (SIE) will respond to USB traffic at the address specified in Device Address [6:0]. When this bit is cleared, the USB transceiver enters power-down state. User's firmware must clear this bit prior to entering sleep mode to save power

0 = Disable USB device address and put the USB transceiver into power-down state

1 = Enable USB device address and put the USB transceiver into normal operating mode

Bit [6:0]: Device Address [6:0]

These bits must be set by firmware during the USB enumeration process (i.e., SetAddress) to the non-zero address assigned by the USB host

21.2 Endpoint 0, 1, and 2 Count

Table 21-2. Endpoint 0, 1, and 2 Count (EP0CNT–EP2CNT) [0x41, 0x43, 0x45] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Data Toggle	Data Valid	Reserved		Byte Count[3:0]			
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit 7: Data Toggle

This bit selects the DATA packet's toggle state. For IN transactions, firmware must set this bit to the select the transmitted Data Toggle. For OUT or SETUP transactions, the hardware sets this bit to the state of the received Data Toggle bit.

0 = DATA0

1 = DATA1

Bit 6: Data Valid

This bit is used for OUT and SETUP tokens only. This bit is cleared to '0' if CRC, bitstuff, or PID errors have occurred. This bit does not update for some endpoint mode settings

0 = Data is invalid. If enabled, the endpoint interrupt will occur even if invalid data is received

1 = Data is valid

Bit [5:4]: Reserved

Bit [3:0]: Byte Count Bit [3:0]

Byte Count Bits indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint FIFO. Valid values are 0 to 8 inclusive. For OUT or SETUP transactions, the count is updated by hardware to the number of data bytes received, plus 2 for the CRC bytes. Valid values are 2–10 inclusive.

For Endpoint 0 Count Register, whenever the count updates from a SETUP or OUT transaction, the count register locks and cannot be written by the CPU. Reading the register unlocks it. This prevents firmware from overwriting a status update on it.

21.3 Endpoint 0 Mode

Because both firmware and the SIE are allowed to write to the Endpoint 0 Mode and Count Registers the SIE provides an interlocking mechanism to prevent accidental overwriting of data.

When the SIE writes to these registers they are locked and the processor cannot write to them until after it has read them. Writing to this register clears the upper four bits regardless of the value written.

Table 21-3. Endpoint 0 Mode (EP0MODE) [0x44] [R/W]

Bit #	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	Setup Received	IN Received	OUT Received	ACK’d Trans	Mode[3:0]			
Read/Write	R/C[4]	R/C[4]	R/C[4]	R/C[4]	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit 7: SETUP Received

This bit is set by hardware when a valid SETUP packet is received. It is forced HIGH from the start of the data packet phase of the SETUP transactions until the end of the data phase of a control write transfer and cannot be cleared during this interval. While this bit is set to ‘1’, the CPU cannot write to the EP0 FIFO. This prevents firmware from overwriting an incoming SETUP transaction before firmware has a chance to read the SETUP data.

This bit is cleared by any non-locked writes to the register.

0 = No SETUP received
1 = SETUP received

Bit 6: IN Received

This bit when set indicates a valid IN packet has been received. This bit is updated to ‘1’ after the host acknowledges an IN data packet.When clear, it indicates either no IN has been received or that the host didn’t acknowledge the IN data by sending ACK handshake.

This bit is cleared by any non-locked writes to the register.

0 = No IN received
1 = IN received

Bit 5: OUT Received

This bit when set indicates a valid OUT packet has been received and ACKed. This bit is updated to ‘1’ after the last received packet in an OUT transaction. When clear, it indicates no OUT received.

This bit is cleared by any non-locked writes to the register.

0 = No OUT received
1 = OUT received

Bit 4: ACK’d Transaction

The ACK’d transaction bit is set whenever the SIE engages in a transaction to the register’s endpoint that completes with a ACK packet.

This bit is cleared by any non-locked writes to the register

1 = The transaction completes with an ACK
0 = The transaction does not complete with an ACK

Bit [3:0]: Mode [3:0]
The endpoint modes determine how the SIE responds to USB traffic that the host sends to the endpoint. The mode controls how the USB SIE responds to traffic and how the USB SIE will change the mode of that endpoint as a result of host packets to the endpoint.

21.4 Endpoint 1 and 2 Mode

Table 21-4. Endpoint 1 and 2 Mode (EP1MODE – EP2MODE) [0x45, 0x46] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Stall	Reserved	NAK Int Enable	ACK'd Transaction	Mode[3:0]			
Read/Write	R/W	R/W	R/W	R/C (Note 4)	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit 7: Stall

When this bit is set the SIE will stall an OUT packet if the Mode Bits are set to ACK-OUT, and the SIE will stall an IN packet if the mode bits are set to ACK-IN. This bit must be clear for all other modes

Bit 6: Reserved

Bit 5: NAK Int Enable

This bit when set causes an endpoint interrupt to be generated even when a transfer completes with a NAK. Unlike enCoRe, enCoRe II family members do not generate an endpoint interrupt under these conditions unless this bit is set

0 = Disable interrupt on NAK'd transactions
1 = Enable interrupt on NAK'd transaction

Bit 4: ACK'd Transaction

The ACK'd transaction bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet.

This bit is cleared by any writes to the register

0 = The transaction does not complete with an ACK
1 = The transaction completes with an ACK

Bit [3:0]: Mode [3:0]
The endpoint modes determine how the SIE responds to USB traffic that the host sends to the endpoint. The mode controls how the USB SIE responds to traffic and how the USB SIE will change the mode of that endpoint as a result of host packets to the endpoint.

Note: When the SIE writes to the EP1MODE or the EP2MODE register it blocks firmware writes to the EP2MODE or the EP1MODE registers respectively (if both writes occur in the same clock cycle). This is because the design employs only one common 'update' signal for both EP1MODE and EP2MODE registers. Thus, when SIE writes to say EP1MODE register, the update signal is set and this prevents firmware writes to EP2MODE register. SIE writes to the endpoint mode registers have higher priority than firmware writes. This mode register write block situation can put the endpoints in incorrect modes. Firmware must read the EP1/2MODE registers immediately following a firmware write and rewrite if the value read is incorrect.

Table 21-5. Endpoint 0 Data (EP0DATA) [0x50-0x57] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Endpoint 0 Data Buffer [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown

The Endpoint 0 buffer is comprised of 8 bytes located at address 0x50 to 0x57

Table 21-6. Endpoint 1 Data (EP1DATA) [0x58-0x5F] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Endpoint 1 Data Buffer [7:0]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown

The Endpoint 1buffer is comprised of 8 bytes located at address 0x58 to 0x5F

Table 21-7. Endpoint 2 Data (EP2DATA) [0x60-0x67] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Endpoint 2 Data Buffer [7:0]							

Table 21-7. Endpoint 2 Data (EP2DATA) [0x60-0x67] [R/W] (continued)

Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown	Unknown
The Endpoint 2 buffer is comprised of 8 bytes located at address 0x60 to 0x67								

The three data buffers used to hold data for both IN and OUT transactions. Each data buffer is 8 bytes long.

The reset values of the Endpoint Data Registers are unknown.

Unlike past enCoRe parts the USB data buffers are only accessible in the I/O space of the processor.

22.0 USB Mode Tables

Mode	Encoding	SETUP	IN	OUT	Comments
DISABLE	0000	Ignore	Ignore	Ignore	Ignore all USB traffic to this endpoint. Used by Data and Control endpoints
NAK IN/OUT	0001	Accept	NAK	NAK	NAK IN and OUT token. Control endpoint only
STATUS OUT ONLY	0010	Accept	STALL	Check	STALL IN and ACK zero byte OUT. Control endpoint only
STALL IN/OUT	0011	Accept	STALL	STALL	STALL IN and OUT token. Control endpoint only
STATUS IN ONLY	0110	Accept	TX0 byte	STALL	STALL OUT and send zero byte data for IN token. Control endpoint only
ACK OUT – STATUS IN	1011	Accept	TX0 byte	ACK	ACK the OUT token or send zero byte data for IN token. Control endpoint only
ACK IN – STATUS OUT	1111	Accept	TX Count	Check	Respond to IN data or Status OUT. Control endpoint only
NAK OUT	1000	Ignore	Ignore	NAK	Send NAK handshake to OUT token. Data endpoint only
ACK OUT (STALL = 0)	1001	Ignore	Ignore	ACK	This mode is changed by the SIE to mode 1000 on issuance of ACK handshake to an OUT. Data endpoint only
ACK OUT (STALL = 1)	1001	Ignore	Ignore	STALL	STALL the OUT transfer
NAK IN	1100	Ignore	NAK	Ignore	Send NAK handshake for IN token. Data endpoint only
ACK IN (STALL = 0)	1101	Ignore	TX Count	Ignore	This mode is changed by the SIE to mode 1100 after receiving ACK handshake to an IN data. Data endpoint only
ACK IN (STALL = 1)	1101	Ignore	STALL	Ignore	STALL the IN transfer. Data endpoint only
Reserved	0101	Ignore	Ignore	Ignore	These modes are not supported by SIE. Firmware must not use this mode in Control and Data endpoints
Reserved	0111	Ignore	Ignore	Ignore	
Reserved	1010	Ignore	Ignore	Ignore	
Reserved	0100	Ignore	Ignore	Ignore	
Reserved	1110	Ignore	Ignore	Ignore	

Mode Column

The 'Mode' column contains the mnemonic names given to the modes of the endpoint. The mode of the endpoint is determined by the four-bit binaries in the 'Encoding' column as discussed below. The Status IN and Status OUT represent the status IN or OUT stage of the control transfer.

Encoding Column

The contents of the 'Encoding' column represent the Mode Bits [3:0] of the Endpoint Mode Registers (*Table 21-3* and *Table 21-4*). The endpoint modes determine how the SIE responds to different tokens that the host sends to the endpoints. For example, if the Mode Bits [3:0] of the Endpoint 0 Mode Register are set to '0001', which is NAK IN/OUT mode, the SIE will send an ACK handshake in response to SETUP tokens and NAK any IN or OUT tokens.

SETUP, IN, and OUT Columns

Depending on the mode specified in the 'Encoding' column, the 'SETUP', 'IN', and 'OUT' columns contain the SIE's responses when the endpoint receives SETUP, IN, and OUT tokens, respectively.

A 'Check' in the Out column means that upon receiving an OUT token the SIE checks to see whether the OUT is of zero length and has a Data Toggle (Data1/0) of 1. If these conditions are true, the SIE responds with an ACK. If any of the above conditions is not met, the SIE will respond with either a STALL or Ignore.

A 'TX Count' entry in the IN column means that the SIE will transmit the number of bytes specified in the Byte Count Bit [3:0] of the Endpoint Count Register (*Table 21-2*) in response to any IN token.

23.0 Details of Mode for Differing Traffic Conditions

Control Endpoint																
SIE	**Bus Event**				**SIE**	**EP0 Mode Register**					**EP0 Count Register**			**EP0**	**Interrupt**	**Comments**
Mode	**Token**	**Count**	**Dval**	**D0/1**	**Response**	**S**	**I**	**O**	**A**	**MODE**	**DTOG**	**DVAL**	**COUNT**	**FIFO**		
DISABLED																
0000	x	x	x	x												Ignore All
STALL_IN_OUT																
0011	SETUP	>10	x	x										junk		Ignore
0011	SETUP	<=10	invalid	x										junk		Ignore
0011	SETUP	<=10	valid	x	ACK	1			1	0001	update	1	update	data	Yes	ACK SETUP
0011	IN	x	x	x	STALL											Stall IN
0011	OUT	>10	x	x												Ignore
0011	OUT	<=10	invalid	x												Ignore
0011	OUT	<=10	valid	x	STALL											Stall OUT
NAK_IN_OUT																
0001	SETUP	>10	x	x										junk		Ignore
0001	SETUP	<=10	invalid	x										junk		Ignore
0001	SETUP	<=10	valid	x	ACK	1			1	0001	update	1	update	data	Yes	ACK SETUP
0001	IN	x	x	x	NAK											NAK IN
0001	OUT	>10	x	x												Ignore
0001	OUT	<=10	invalid	x												Ignore
0001	OUT	<=10	valid	x	NAK											NAK OUT
ACK_IN_STATUS_OUT																
1111	SETUP	>10	x	x										junk		Ignore
1111	SETUP	<=10	invalid	x										junk		Ignore
1111	SETUP	<=10	valid	x	ACK	1			1	0001	update	1	update	data	Yes	ACK SETUP
1111	IN	x	x	x	TX											Host Not ACK'd
1111	IN	x	x	x	TX		1		1	0001					Yes	Host ACK'd
1111	OUT	>10	x	x												Ignore
1111	OUT	<=10	invalid	x												Ignore
1111	OUT	<=10, <>2	valid	x	STALL					0011					Yes	Bad Status
1111	OUT	2	valid	0	STALL					0011					Yes	Bad Status
1111	OUT	2	valid	1	ACK			1	1	0010	1	1	2		Yes	Good Status
STATUS_OUT																
0010	SETUP	>10	x	x										junk		Ignore
0010	SETUP	<=10	invalid	x										junk		Ignore
0010	SETUP	<=10	valid	x	ACK	1			1	0001	update	1	update	data	Yes	ACK SETUP
0010	IN	x	x	x	STALL					0011					Yes	Stall IN
0010	OUT	>10	x	x												Ignore
0010	OUT	<=10	invalid	x												Ignore
0010	OUT	<=10, <>2	valid	x	STALL					0011					Yes	Bad Status
0010	OUT	2	valid	0	STALL					0011					Yes	Bad Status
0010	OUT	2	valid	1	ACK			1	1		1	1	2		Yes	Good Status
ACK_OUT_STATUS_IN																
1011	SETUP	>10	x	x										junk		Ignore
1011	SETUP	<=10	invalid	x										junk		Ignore
1011	SETUP	<=10	valid	x	ACK	1			1	0001	update	1	update	data	Yes	ACK SETUP

23.0 Details of Mode for Differing Traffic Conditions (continued)

1011	IN	x	x	x	TX 0											Host Not ACK'd
1011	IN	x	x	x	TX 0		1		1	0011					Yes	Host ACK'd
1011	OUT	>10	x	x										junk		Ignore
1011	OUT	<=10	invalid	x										junk		Ignore
1011	OUT	<=10	valid	x	ACK			1	1	0001	update	1	update	data	Yes	Good OUT
STATUS_IN																
0110	SETUP	>10	x	x										junk		Ignore
0110	SETUP	<=10	invalid	x										junk		Ignore
0110	SETUP	<=10	valid	x	ACK	1			1	0001	update	1	update	data	Yes	ACK SETUP
0110	IN	x	x	x	TX 0											Host Not ACK'd
0110	IN	x	x	x	TX 0		1		1	0011					Yes	Host ACK'd
0110	OUT	>10	x	x												Ignore
0110	OUT	<=10	invalid	x												Ignore
0110	OUT	<=10	valid	x	STALL					0011					Yes	Stall OUT
Data Out Endpoints																
SIE	**Bus Event**				**SIE**	**EP0 Mode Register**					**EP0 Count Register**			**EP0**	**Interrupt**	**Comments**
Mode	**Token**	**Count**	**Dval**	**D0/1**	**Response**	**S**	**I**	**O**	**A**	**MODE**	**DTOG**	**DVAL**	**COUNT**	**FIFO**		
ACK OUT (STALL Bit = 0)																
1001	IN	x	x	x												Ignore
1001	OUT	>MAX	x	x										junk		Ignore
1001	OUT	<=MAX	invalid	invalid										junk		Ignore
1001	OUT	<=MAX	valid	valid	ACK				1	1000	update	1	update	data	Yes	ACK OUT
ACK OUT (STALL Bit = 1)																
1001	IN	x	x	x												Ignore
1001	OUT	>MAX	x	x												Ignore
1001	OUT	<=MAX	invalid	invalid												Ignore
1001	OUT	<=MAX	valid	valid	STALL											Stall OUT
NAK OUT																
1000	IN	x	x	x												Ignore
1000	OUT	>MAX	x	x												Ignore
1000	OUT	<=MAX	invalid	invalid												Ignore
1000	OUT	<=MAX	valid	valid	NAK										If Enabled	NAK OUT
Data In Endpoints																
SIE	**Bus Event**				**SIE**	**EP0 Mode Register**					**EP0 Count Register**			**EP0**	**Interrupt**	**Comments**
Mode	**Token**	**Count**	**Dval**	**D0/1**	**Response**	**S**	**I**	**O**	**A**	**MODE**	**DTOG**	**DVAL**	**COUNT**	**FIFO**		
ACK IN (STALL Bit = 0)																
1101	OUT	x	x	x												Ignore
1101	IN	x	x	x												Host Not ACK'd
1101	IN	x	x	x	TX				1	1100					Yes	Host ACK'd
ACK IN (STALL Bit = 1)																
1101	OUT	x	x	x												Ignore
1101	IN	x	x	x	STALL											Stall IN
NAK IN																
1100	OUT	x	x	x												Ignore
1100	IN	x	x	x	NAK										If Enabled	NAK IN

24.0 Register Summary

Addr	Name	7	6	5	4	3	2	1	0	R/W	Default
00	P0DATA	P0.7	P0.6/TIO1	P0.5/TIO0	P0.4/INT2	P0.3/INT1	P0.2/INT0	P0.1/CLK-OUT	P0.0/CLKIN	bbbbbbbb	00000000
01	P1DATA	P1.7	P1.6/SMISO	P1.5/SMOSI	P1.4/SCLK	P1.3/SSEL	P1.2/VREG	P1.1/D–	P1.0/D+	bbbbbbbb	00000000
02	P2DATA	Res						P2.1–P2.0		bbbbbbbb	00000000
03	P3DATA	Res						P3.1–P3.0		bbbbbbbb	00000000
04	P4DATA	Res				Res				----bbbb	00000000
05	P00CR	Reserved	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
06	P01CR	CLK Output	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	bbbbbbbb	00000000
07–09	P02CR–P04CR	Reserved	Reserved	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	--bbbbbb	00000000
0A–0B	P05CR–P06CR	TIO Output	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	bbbbbbbb	00000000
0C	P07CR	Reserved	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
0D	P10CR	Reserved	Int Enable	Int Act Low	Reserved			PS/2 Pull-up Enable	Output Enable	-bb---bb	00000000
0E	P11CR	Reserved	Int Enable	Int Act Low	Reserved		Open Drain	Reserved	Output Enable	-bb--b-b	00000000
0F	P12CR	CLK Output	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	bbbbbbbb	00000000
10	P13CR	Reserved	Int Enable	Int Act Low	3.3V Drive	High Sink	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
11–13	P14CR–P16CR	SPI Use	Int Enable	Int Act Low	3.3V Drive	High Sink	Open Drain	Pull-up Enable	Output Enable	bbbbbbbb	00000000
14	P17CR	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
15	P2CR	Reserved	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
16	P3CR	Reserved	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
20	FRTMRL	Free Running Timer [7:0]								bbbbbbbb	00000000
21	FRTMRH	Free Running Timer [15:8]								bbbbbbbb	00000000
22	TCAP0R	Capture 0 Rising [7:0]								bbbbbbbb	00000000
23	TCAP1R	Capture 1 Rising [7:0]								bbbbbbbb	00000000
24	TCAP0F	Capture 0 Falling [7:0]								bbbbbbbb	00000000
25	TCAP1F	Capture 1 Falling [7:0]								bbbbbbbb	00000000
26	PITMRL	Prog Interval Timer [7:0]								bbbbbbbb	00000000
27	PITMRH	Reserved				Prog Interval Timer [11:8]				----bbbb	00000000
28	PIRL	Prog Interval [7:0]								bbbbbbbb	00000000
29	PIRH	Reserved				Prog Interval [11:8]				----bbbb	00000000
2A	TMRCR	First Edge Hold	8-bit capture Prescale			Cap0 16bit Enable	Reserved			bbbbb---	00000000
2B	TCAPINTE	Reserved				Cap1 Fall Active	Cap1 Rise Active	Cap0 Fall Active	Cap0 Rise Active	----bbbb	00000000
2C	TCAPINTS	Reserved				Cap1 Fall Active	Cap1 Rise Active	Cap0 Fall Active	Cap0 Rise Active	----bbbb	00000000
30	CPUCLKCR	Reserved	USB CLK/2 Disable	USB CLK Select	Reserved				CPU CLK Select	-bb----b	00010000
31	ITMRCLKCR	TCAPCLK Divider		TCAPCLK Select		ITMRCLK Divider		ITMRCLK Select		bbbbbbbb	10001111
32	CLKIOCR	Reserved			Reserved			CLKOUT Select		---bbbbb	00000000
34	IOSCTR	foffset[2:0]			Gain[4:0]					bbbbbbbb	000ddddd
36	LPOSCTR	32-KHz Low Power	Reserved	32-KHz Bias Trim [1:0]		32-KHz Freq Trim [3:0]				b-bbbbbb	dddddddd
39	OSCLCKCR	Reserved						Fine Tune Only	USB Osclock Disable	------bb	00000000

24.0 Register Summary (continued)

Addr	Name	7	6	5	4	3	2	1	0	R/W	Default
3C	SPIDATA	SPIData[7:0]								bbbbbbbb	00000000
3D	SPICR	Swap	LSB First	Comm Mode		CPOL	CPHA	SCLK Select		bbbbbbbb	00000000
40	USBCR	USB Enable	Device Address[6:0]							bbbbbbbb	00000000
41	EP0CNT	Data Toggle	Data Valid	Reserved		Byte Count[3:0]				bbbbbbbb	00000000
42	EP1CNT	Data Toggle	Data Valid	Reserved		Byte Count[3:0]				bbbbbbbb	00000000
43	EP2CNT	Data Toggle	Data Valid	Reserved		Byte Count[3:0]				bbbbbbbb	00000000
44	EP0MODE	Setup rcv'd	IN rcv'd	OUT rcv'd	ACK'd trans	Mode[3:0]				ccccbbbb	00000000
45	EP1MODE	Stall	Reserved	NAK Int Enable	Ack'd trans	Mode[3:0]				b-bcbbbb	00000000
46	EP2MODE	Stall	Reserved	NAK Int Enable	Ack'd trans	Mode[3:0]				b-bcbbbb	00000000
50–57	EP0DATA	Endpoint 0 Data Buffer [7:0]								bbbbbbbb	????????
58–5F	EP1DATA	Endpoint 1 Data Buffer [7:0]								bbbbbbbb	????????
60–67	EP2DATA	Endpoint 2 Data Buffer [7:0]								bbbbbbbb	????????
73	VREGCR	Reserved						Keep Alive	VREG Enable	------bb	00000000
74	USBXCR	USB Pull-up Enable	Reserved						USB Force State	b------b	00000000
DA	INT_CLR0	GPIO Port 1	Sleep Timer	INT1	GPIO Port 0	SPI Receive	SPI Transmit	INT0	POR/LVD	bbbbbbbb	00000000
DB	INT_CLR1	TCAP0	Prog Interval Timer	1-ms Timer	USB Active	USB Reset	USB EP2	USB EP1	USB EP0	bbbbbbbb	00000000
DC	INT_CLR2	Reserved	Reserved	GPIO Port 3	GPIO Port 2	PS/2 Data Low	INT2	16-bit Counter Wrap	TCAP1	-bbbbbbb	00000000
DE	INT_MSK3	ENSWINT	Reserved							b-------	00000000
DF	INT_MSK2	Reserved	Reserved	GPIO Port 3 Int Enable	GPIO Port 2 Int Enable	PS/2 Data Low Int Enable	INT2 Int Enable	16-bit Counter Wrap Int Enable	TCAP1 Int Enable	-bbbbbbb	00000000
E1	INT_MSK1	TCAP0 Int Enable	Prog Interval Timer Int Enable	1-ms Timer Int Enable	USB Active Int Enable	USB Reset Int Enable	USB EP2 Int Enable	USB EP1 Int Enable	USB EP0 Int Enable	bbbbbbbb	00000000
E0	INT_MSK0	GPIO Port 1 Int Enable	Sleep Timer Int Enable	INT1 Int Enable	GPIO Port 0 Int Enable	SPI Receive Int Enable	SPI Transmit Int Enable	INT0 Int Enable	POR/LVD Int Enable	bbbbbbbb	00000000
E2	INT_VC	Pending Interrupt [7:0]								bbbbbbbb	00000000
E3	RESWDT	Reset Watchdog Timer [7:0]								wwwwwwww	00000000
--	CPU_A	Temporary Register T1 [7:0]								--------	00000000
--	CPU_X	X[7:0]								--------	00000000
--	CPU_PCL	Program Counter [7:0]								--------	00000000
--	CPU_PCH	Program Counter [15:8]								--------	00000000
--	CPU_SP	Stack Pointer [7:0]								--------	00000000
-	CPU_F	Reserved			XOI	Super	Carry	Zero	Global IE	---brwww	00000010
FF	CPU_SCR	GIES	Reserved	WDRS	PORS	Sleep	Reserved	Reserved	Stop	r-ccb--b	00010000
1E0	OSC_CR0	Reserved		No Buzz	Sleep Timer [1:0]		CPU Speed [2:0]			--bbbbbb	00000000
1E3	LVDCR	Reserved		PORLEV[1:0]		Reserved	VM[2:0]			--bb-bbbb	00000000
1EB	ECO_TR	Sleep Duty Cycle [1:0]		Reserved						bb------	00000000
1E4	VLTCMP	Reserved						LVD	PPOR	------rr	00000000

Legend
In the R/W column,
b = Both Read and Write
r = Read Only
w = Write Only
c = Read/Clear
? = Unknown

d = calibration value. Must not change during normal use.

25.0 Absolute Maximum Ratings

Storage Temperature ....–65°C to +150°C

Ambient Temperature with Power Applied ..... –0°C to +70°C

Supply Voltage on V_{CC} Relative to V_{SS}..........–0.5V to +7.0V

DC Input Voltage .... –0.5V to + V_{CC} + 0.5V

DC Voltage Applied to Outputs in
High-Z State .... –0.5V to + V_{CC} + 0.5V

Maximum Total Sink Output Current into Port 0
and 1 and Pins...70 mA

Maximum Total Source Output Current into GPIO Pins30 mA

Maximum On-chip Power Dissipation
on any GPIO Pin...50 mW

Power Dissipation ....300 mW

Static Discharge Voltage 2200V

Latch-up Current .... 200 mA

26.0 DC Characteristics

	Description					
Parameter	General	Conditions	Min.	Typical	Max.	Unit
V_{CC1}	Operating Voltage	No USB activity, CPU speed ≤ 12 MHz	4.0		5.5	V
V_{CC2}	Operating Voltage	USB activity, CPU speed ≤ 12 MHz	4.35		5.25	V
V_{CC3}	Operating Voltage	Flash programming	4.35		5.25	V
V_{CC4}	Operating Voltage	No USB activity, CPU speed ≤ 24 MHz	4.75		5.5	V
T_{FP}	Operating Temp	Flash Programming	0		70	°C
I_{CC1}	V_{CC} Operating Supply Current	V_{CC} = 5.25V, no GPIO loading, 24 MHz			40	mA
I_{CC2}	V_{CC} Operating Supply Current	V_{CC} = 5.0V, no GPIO loading, 6 MHz		10		mA
I_{SB1}	Standby Current	Internal and External Oscillators, Bandgap, Flash, CPU Clock, Timer Clock, USB Clock all disabled			10	µA
Low-voltage Detect						
V_{LVD}	Low-voltage detect Trip Voltage (8 programmable trip points)		2.681		4.872	V
3.3V Regulator						
I_{VREG}	Max Regulator Output Current	4.35V ≤ V_{CC} ≤ 5.5V			125	mA
I_{KA}	Keep Alive Current	When regulator is disabled with “keep alive” enable			20	µA
V_{KA}	Keep Alive Voltage	Keep Alive bit set in VREGCR	2.35		3.8	V
V_{REG1}	V_{REG} Output Voltage	V_{CC} ≥ 4.35V, 0 < temp < 40°C, 25 mA ≤ I_{VREG} ≤ 125 mA (3.3V ± 8%) T = 0 to 70C	3.0		3.6	V
V_{REG2}	V_{REG} Output Voltage	V_{CC} ≥ 4.35V, 0 < temp < 40°C, 1 mA ≤ I_{VREG} ≤ 25 mA (3.3V ± 4%) T = 0 to 40C	3.15		3.45	V
C_{LOAD}	Capacitive load on Vreg pin		1		2	µF
LN_{REG}	Line Regulation				1	%/V
LD_{REG}	Load Regulation				0.04	%/mA
USB Interface						
V_{ON}	Static Output High	15K ± 5% Ohm to V_{SS}	2.8		3.6	V
V_{OFF}	Static Output Low	R_{UP} is enabled			0.3	V
V_{DI}	Differential Input Sensitivity		0.2			V
V_{CM}	Differential Input Common Mode Range		0.8		2.5	V
V_{SE}	Single Ended Receiver Threshold		0.8		2	V
C_{IN}	Transceiver Capacitance				20	pF
I_{IO}	Hi-Z State Data Line Leakage	0V < V_{IN} < 3.3V	–10		10	µA

26.0 DC Characteristics (continued)

Parameter	Description	Conditions	Min.	Typical	Max.	Unit
	General					
PS/2 Interface						
V_{OLP}	Static Output Low	SDATA or SCLK pins			0.4	V
R_{PS2}	Internal PS/2 Pull up Resistance	SDATA, SCLK pins, PS/2 Enabled	3		7	KΩ
General Purpose I/O Interface						
R_{UP}	Pull up Resistance		4		12	KΩ
V_{ICR}	Input Threshold Voltage Low, CMOS mode	Low to High edge	40%		65%	V_{CC}
V_{ICF}	Input Threshold Voltage Low, CMOS mode	High to Low edge	30%		55%	V_{CC}
V_{HC}	Input Hysteresis Voltage, CMOS Mode	High to low edge	3%		10%	V_{CC}
V_{ILTTL}	Input Low Voltage, TTL Mode	I/O-pin Supply = 2.9–3.6V			0.8	V
V_{IHTTL}	Input High Voltage, TTL Mode	I/O-pin Supply = 4.0–5.5V	2.0			V
V_{OL1}	Output Low Voltage, High Drive[5]	I_{OL1} = 50 mA			0.8	V
V_{OL2}	Output Low Voltage, High Drive[5]	I_{OL1} = 25 mA			0.4	V
V_{OL3}	Output Low Voltage, Low Drive[6]	I_{OL2} = 8 mA			0.4	V
V_{OH}	Output High Voltage[6]	I_{OH} = 2 mA	V_{CC} – 0.5			V
C_{LOAD}	Maximum load capacitance				50	pF

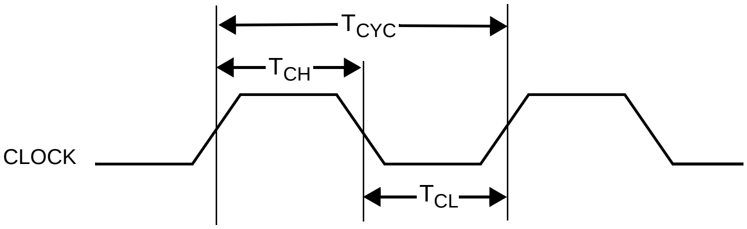
27.0 AC Characteristics

Parameter	Description	Conditions	Min.	Typical	Max.	Unit
Clock						
T_{ECLKDC}	External Clock Duty Cycle		45		55	%
T_{ECLK1} T_{ECLK2}	External Clock Frequency External Clock Frequency	External clock is the source of the CPUCLK External clock is not the source of the CPUCLK	0.187 0		24 24	MHz MHz
3.3V Regulator						
V_{ORIP}	Output Ripple Voltage	10Hz to 100MHz at CLOAD = 1µF			200	mV_{p-p}
USB Driver						
T_{R1}	Transition Rise Time	C_{LOAD} = 200 pF	75			ns
T_{R2}	Transition Rise Time	C_{LOAD} = 600 pF			300	ns
T_{F1}	Transition Fall Time	C_{LOAD} = 200 pF	75			ns
T_{F2}	Transition Fall Time	C_{LOAD} = 600 pF			300	ns
T_R	Rise/Fall Time Matching		80		125	%
V_{CRS}	Output Signal Crossover Voltage		1.3		2.0	V
USB Data Timing						
T_{DRATE}	Low-speed Data Rate	Ave. Bit Rate (1.5 Mbps ± 1.5%)	1.4775		1.5225	Mbps
T_{DJR1}	Receiver Data Jitter Tolerance	To next transition	–75		75	ns
T_{DJR2}	Receiver Data Jitter Tolerance	To pair transition	–45		45	ns
T_{DEOP}	Differential to EOP Transition Skew		–40		100	ns

27.0 AC Characteristics (continued)

Parameter	Description	Conditions	Min.	Typical	Max.	Unit
T_{EOPR1}	EOP Width at Receiver	Rejects as EOP			330	ns
T_{EOPR2}	EOP Width at Receiver	Accept as EOP	675			ns
T_{EOPT}	Source EOP Width		1.25		1.5	µs
T_{UDJ1}	Differential Driver Jitter	To next transition	–95		95	ns
T_{UDJ2}	Differential Driver Jitter	To pair transition	–95		95	ns
T_{LST}	Width of SE0 during Diff. Transition				210	ns
Non-USB Mode Driver Characteristics						
T_{FPS2}	SDATA/SCK Transition Fall Time		50		300	ns
GPIO Timing						
T_{R_GPIO}	Output Rise Time	Measured between 10 and 90% Vdd/Vreg with 50 pF load			50	ns
T_{F_GPIO}	Output Fall Time	Measured between 10 and 90% Vdd/Vreg with 50 pF load			15	ns
SPI Timing						
T_{SMCK}	SPI Master Clock Rate	$F_{CPUCLK}/6$			2	MHz
T_{SSCK}	SPI Slave Clock Rate				2.2	MHz
T_{SCKH}	SPI Clock High Time	High for CPOL = 0, Low for CPOL = 1	125			ns
T_{SCKL}	SPI Clock Low Time	Low for CPOL = 0, High for CPOL = 1	125			ns
T_{MDO}	Master Data Output Time[5]	SCK to data valid	–25		50	ns
T_{MDO1}	Master Data Output Time, First bit with CPHA = 0	Time before leading SCK edge	100			ns
T_{MSU}	Master Input Data Set-up time		50			ns
T_{MHD}	Master Input Data Hold time		50			ns
T_{SSU}	Slave Input Data Set-up Time		50			ns
T_{SHD}	Slave Input Data Hold Time		50			ns
T_{SDO}	Slave Data Output Time	SCK to data valid			100	ns
T_{SDO1}	Slave Data Output Time, First bit with CPHA = 0	Time after $\overline{\text{SS}}$ LOW to data valid			100	ns
T_{SSS}	Slave Select Set-up Time	Before first SCK edge	150			ns
T_{SSH}	Slave Select Hold Time	After last SCK edge	150			ns

Figure 27-1. Clock Timing



Note

[5] In Master mode first bit is available 0.5 SPICLK cycle before Master clock edge available on the SCLK pin.

Figure 27-2. GPIO Timing Diagram

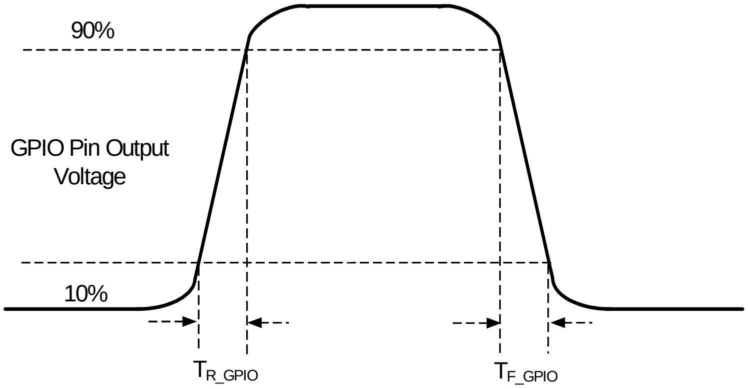


Figure 27-3. USB Data Signal Timing

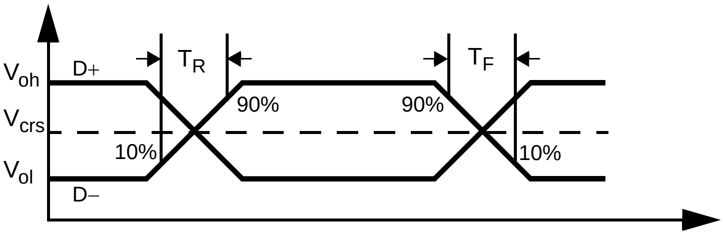


Figure 27-4. Receiver Jitter Tolerance

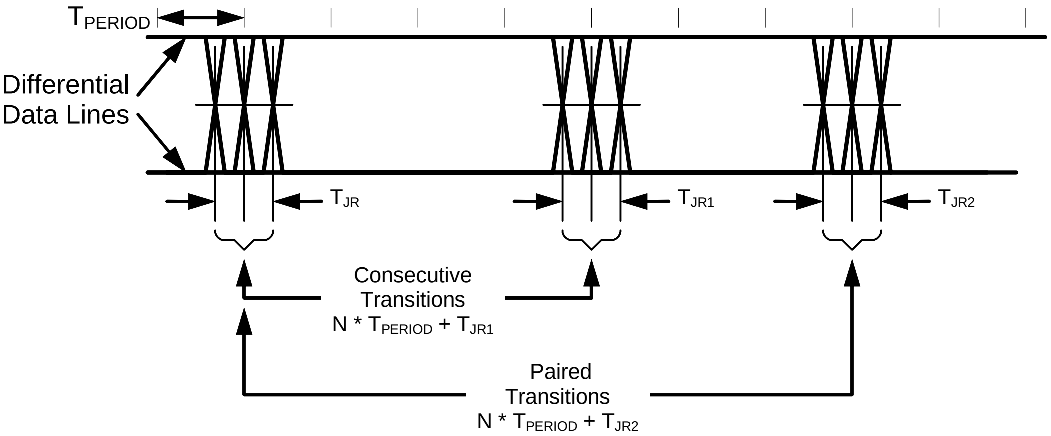


Figure 27-5. Differential to EOP Transition Skew and EOP Width

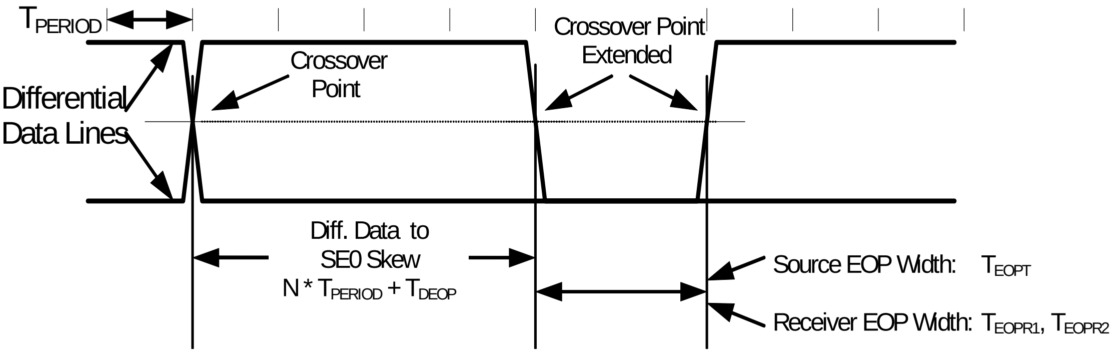


Figure 27-6. Differential Data Jitter

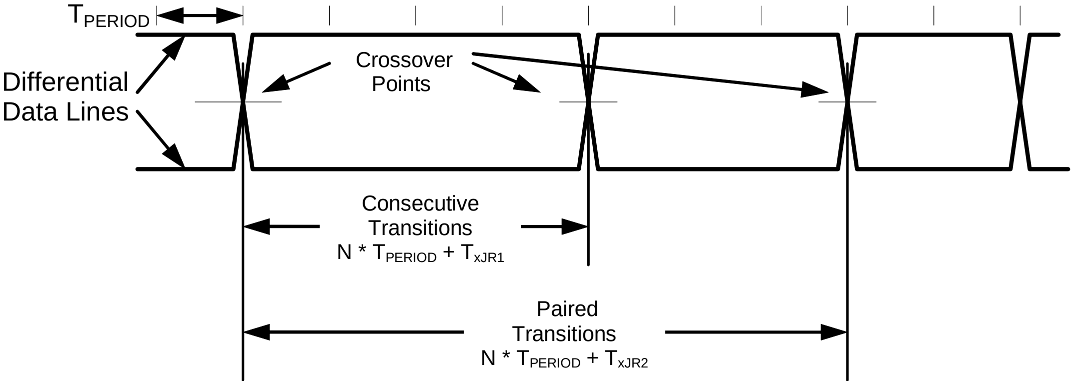


Figure 27-7. SPI Master Timing, CPHA = 1

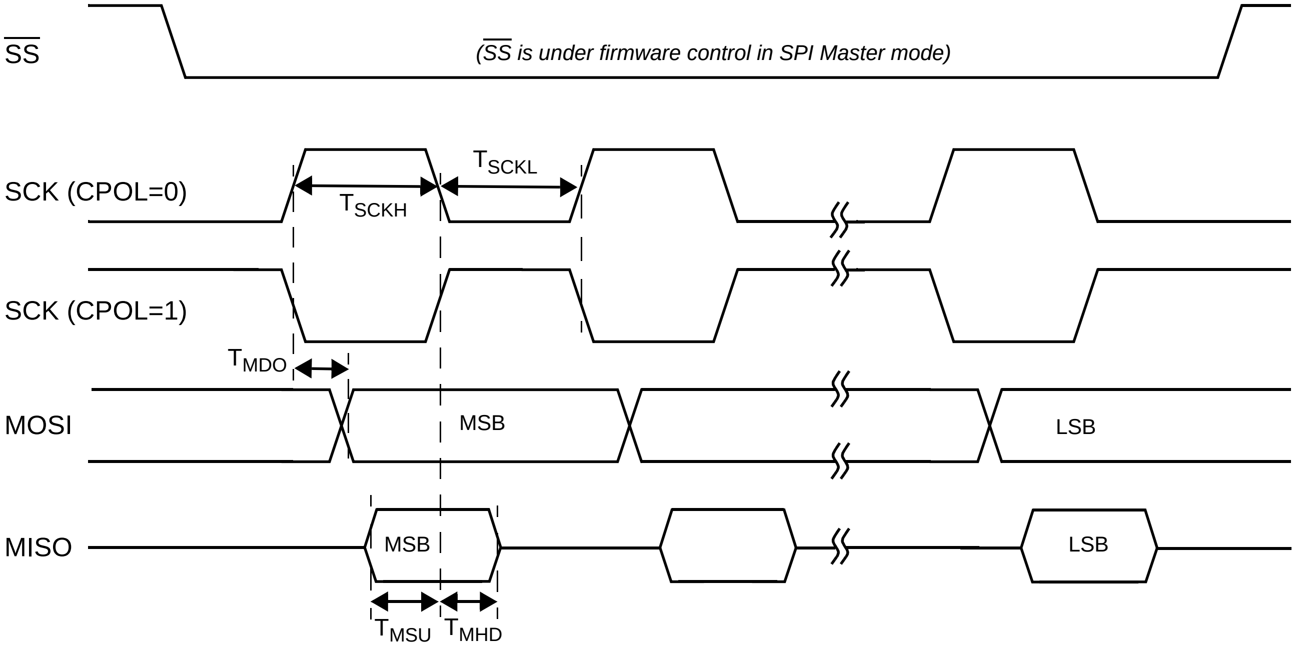


Figure 27-8. SPI Slave Timing, CPHA = 1

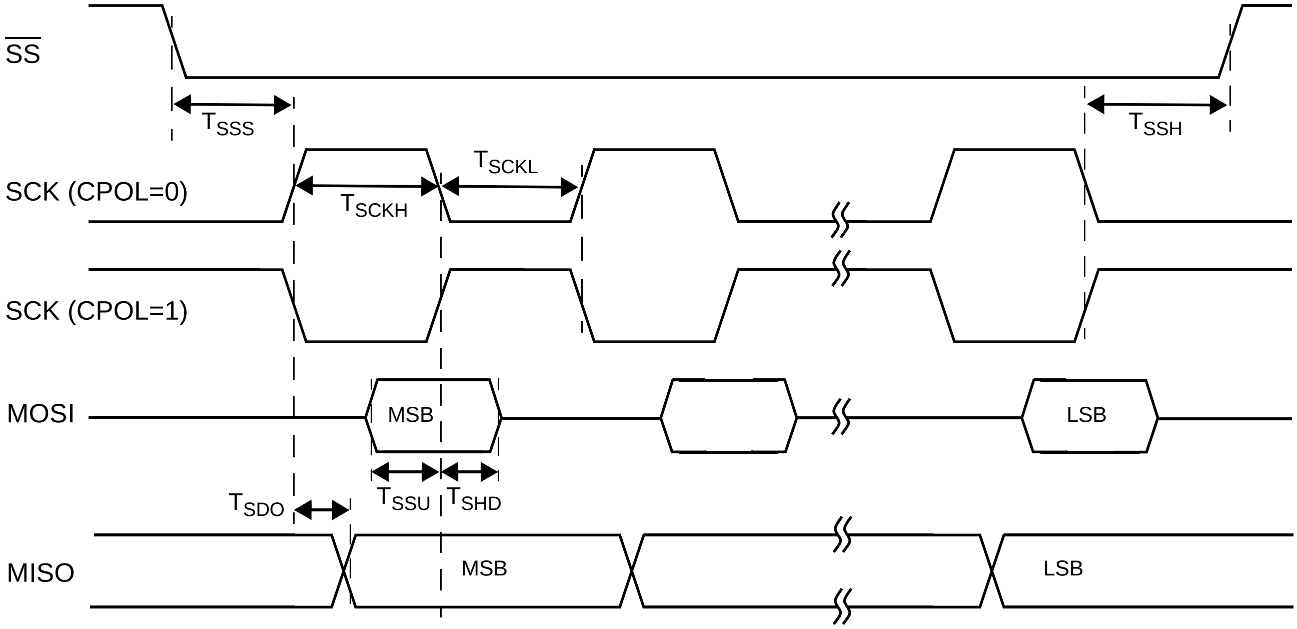


Figure 27-9. SPI Master Timing, CPHA = 0

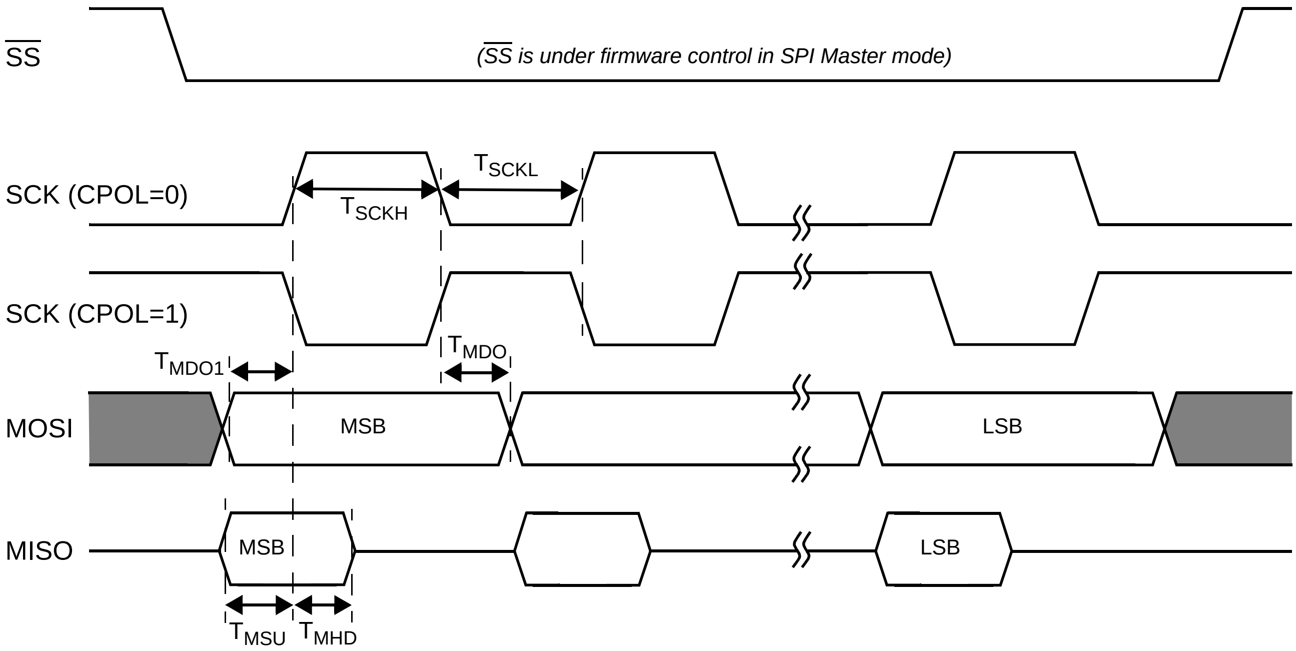
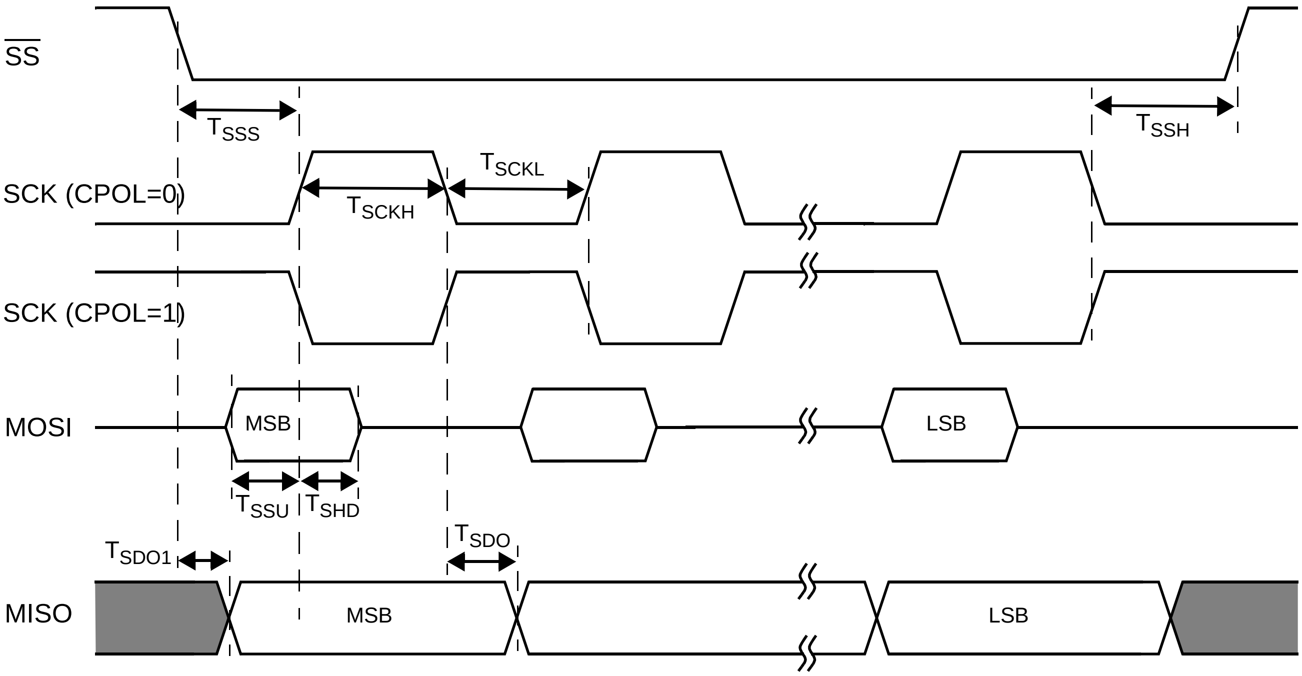


Figure 27-10. SPI Slave Timing, CPHA = 0



28.0 Ordering Information

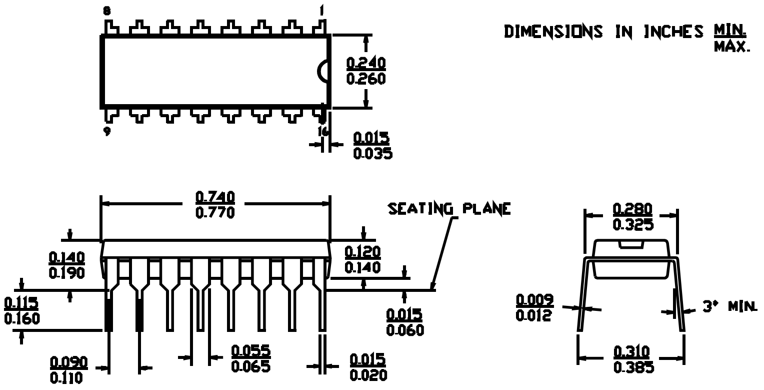
Ordering Code	FLASH Size	RAM Size	Package Type
CY7C63833-LFXC	8K	256	32-QFN
CY7C63823-SXC	8K	256	24-SOIC
CY7C63823-QXC	8K	256	24-QSOP

28.0 Ordering Information (continued)

Ordering Code	FLASH Size	RAM Size	Package Type
CY7C63813-PXC	8K	256	18-PDIP
CY7C63813-SXC	8K	256	18-SOIC
CY7C63803-SXC	8K	256	16-SOIC
CY7C63801-PXC	4K	256	16-PDIP
CY7C63801-SXC	4K	256	16-SOIC
CY7C63310-PXC	3K	128	16-PDIP
CY7C63310-SXC	3K	128	16-SOIC

29.0 Package Diagrams

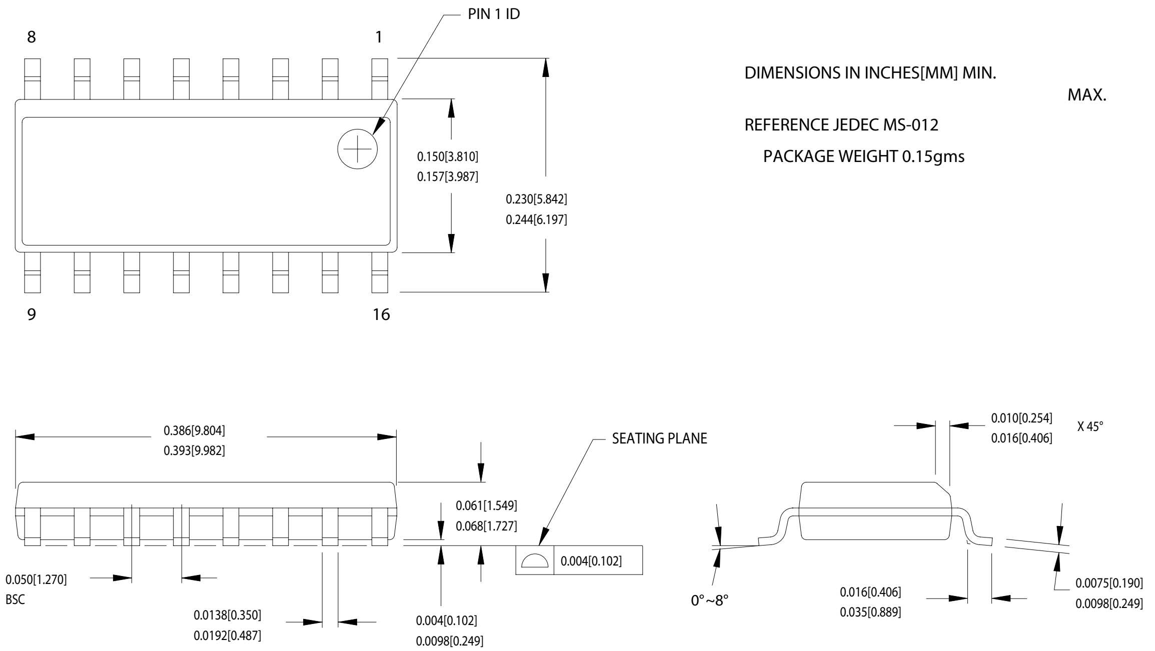
Figure 29-1. 16-Lead(300-Mil) Molded DIP P1



51-85009-*A

29.0 Package Diagrams (continued)

Figure 29-2. 16-Lead (150-Mil) SOIC S16.15



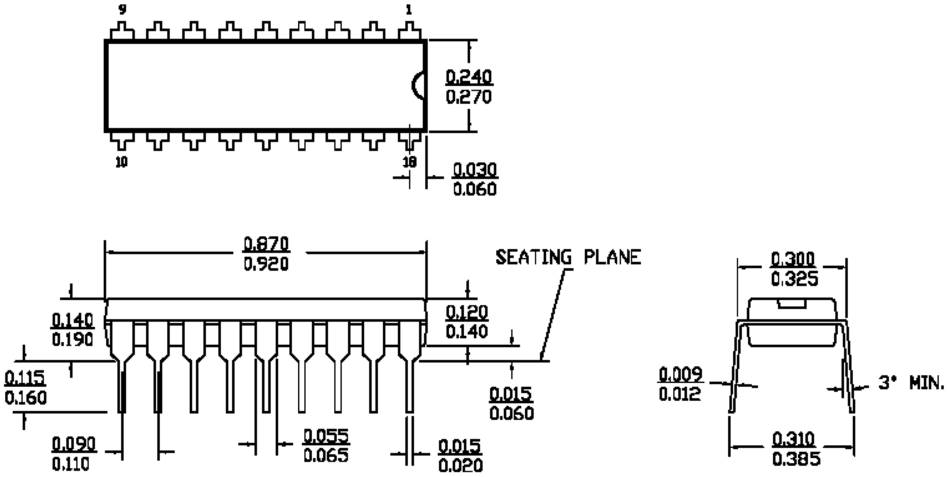
DIMENSIONS IN INCHES[MM] MIN.
MAX.

REFERENCE JEDEC MS-012

PACKAGE WEIGHT 0.15gms

51-85068-*B

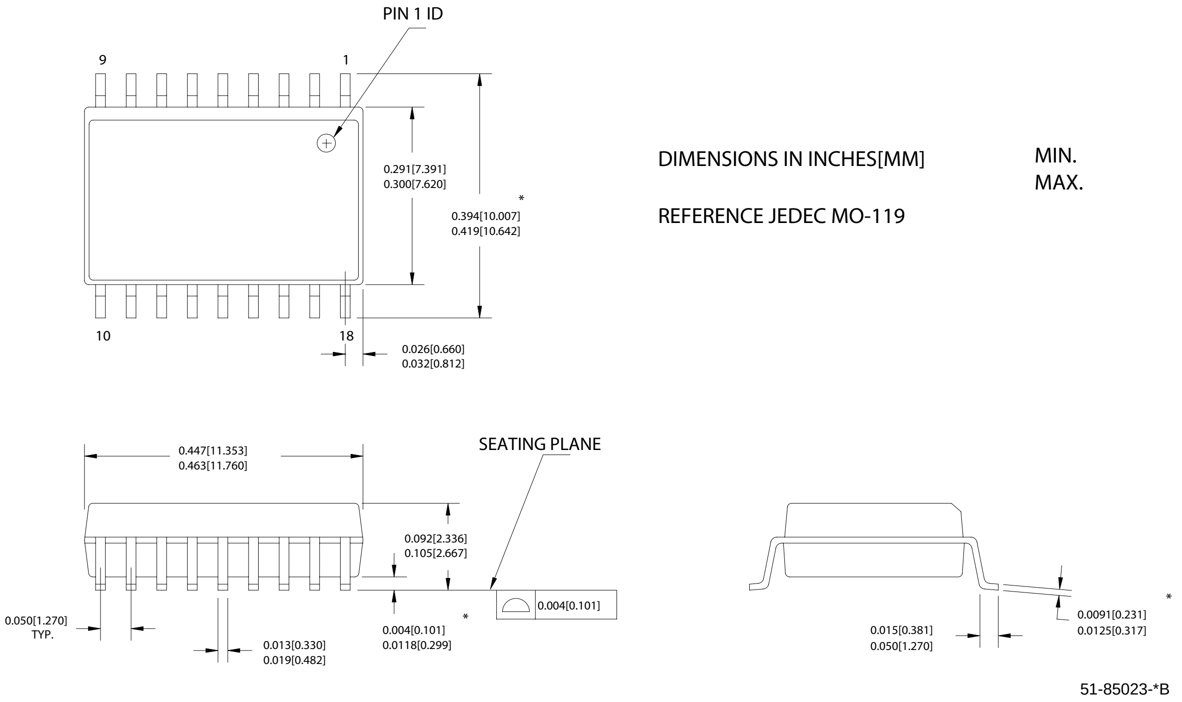
Figure 29-3. 18-Lead(300-Mil) Molded DIP P3



51-85010-*B

29.0 Package Diagrams (continued)

Figure 29-4. 18-Lead(300-Mil) Molded SOIC S3



29.0 Package Diagrams (continued)

Figure 29-5. 24-Lead (300-Mil) SOIC S13

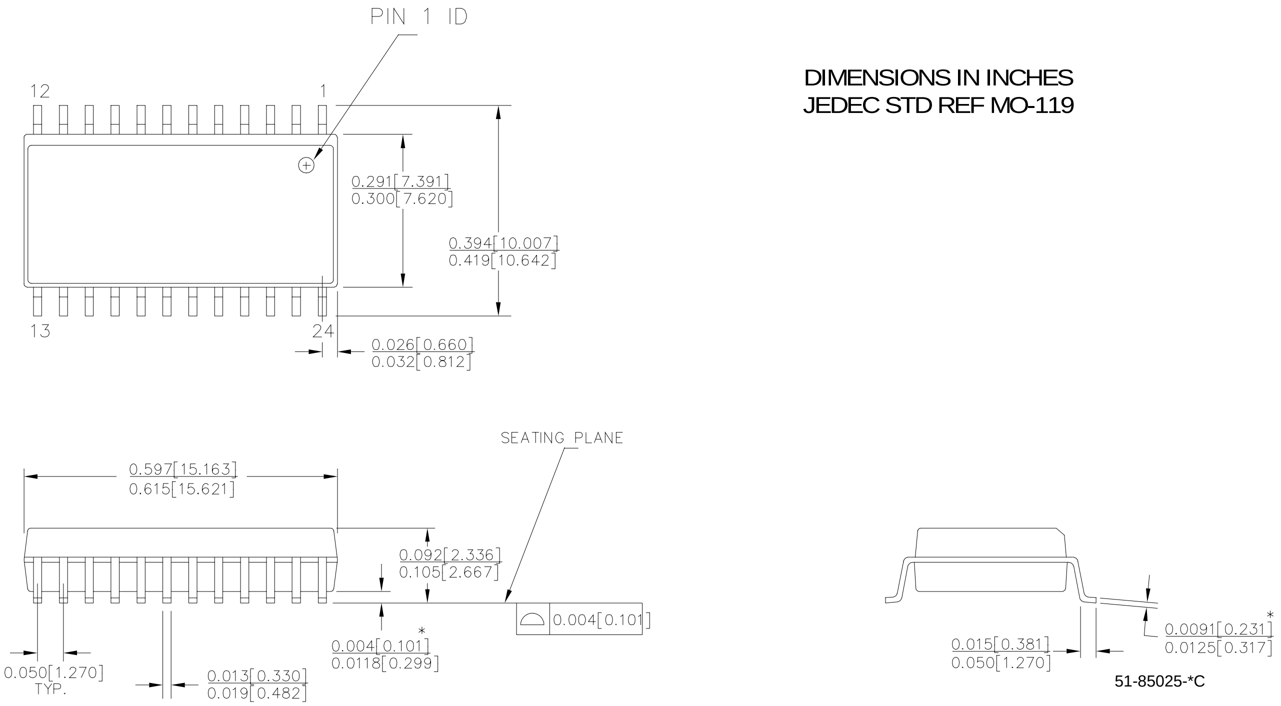
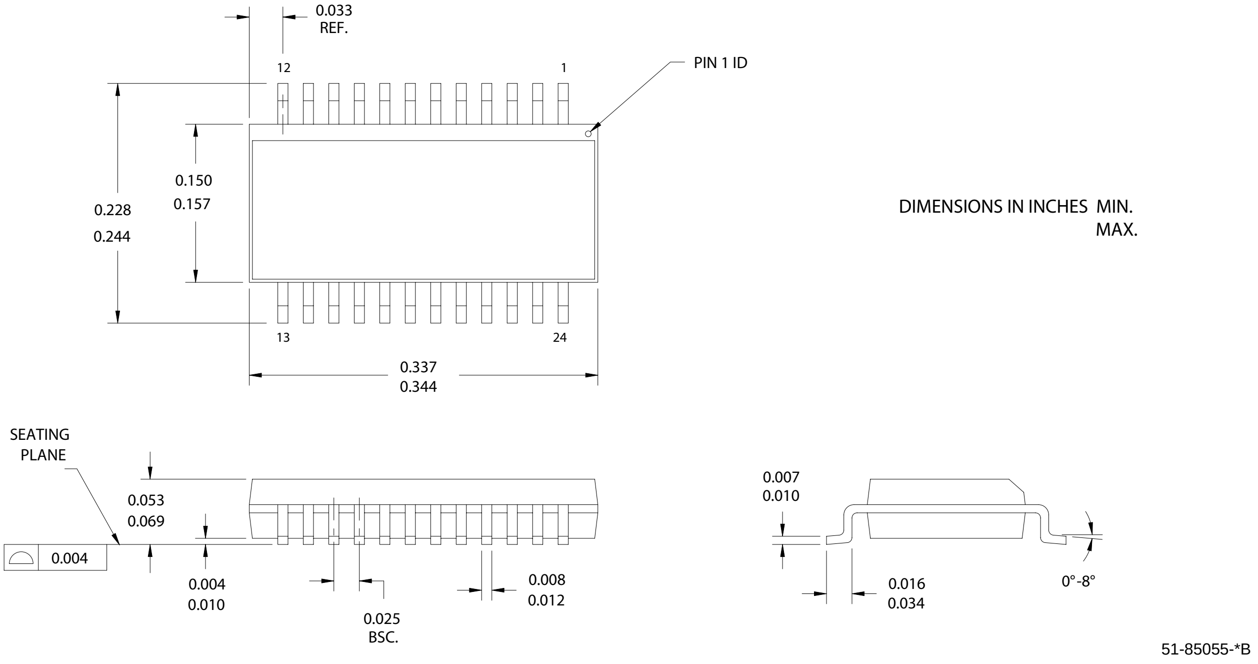
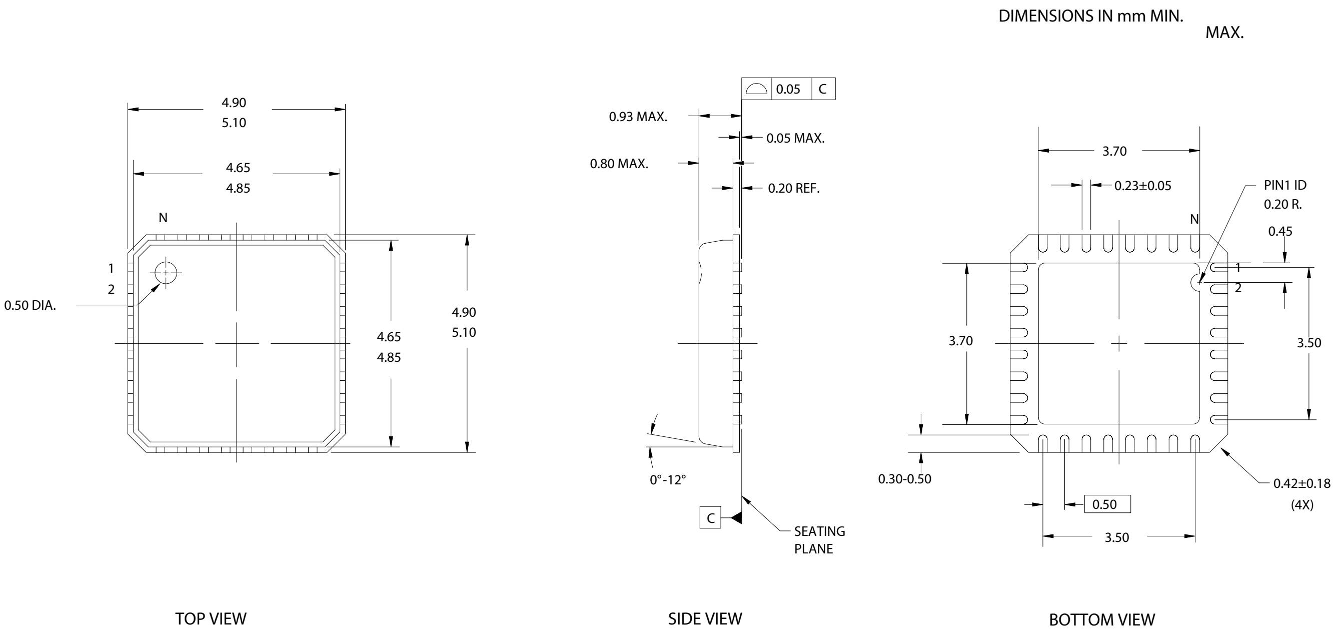


Figure 29-6. 24-lead QSOP O241



29.0 Package Diagrams (continued)

Figure 29-7. 32-Lead QFN Package



JEDEC # MO-220
Package Weight: 0.054 grams

51-85188-*A

PSoC is a trademark of Cypress MicroSystems. enCoRe is a trademark of Cypress Semiconductor Corporation. All product and company names mentioned in this document are the trademarks of their respective holders.

30.0 Document History Page

Document Title: CY7C63310/CY7C638xx enCoRe™ II Low-Speed USB Peripheral Controller
Document Number: 38-08035

Rev.	ECN No.	Issue Date	Orig. of Change	Description of Change
**	131323	12/11/03	XGR	New data sheet
*A	221881	See ECN	KKU	Added Register descriptions and package information, changed from advance information to preliminary
*B	271232	See ECN	BON	Reformatted Updated with the latest information
*C	299179	See ECN	BON	Corrected 24-PDIP pinout typo in *Table 5.1* Added *Table 10-1* Updated *Table 9-5*, *Table 10-3*, *Table 13-1*, *Table 17-2*, *Table 17-4*, *Table 17-6*. and *Table 15-2*. Added various updates to the GPIO Section (Section 14.0) Corrected *Table 15-3*. Corrected *Figure 27-7* and *Figure 27-8*. Added the 16-pin PDIP package diagram (Section 29.0)
*D	322053	See ECN	TVR BON	Introduction section: Last para removed Low-voltage reset. There is no LVR there is only LVD (Low voltage detect). explained more about LVD and POR Changed capture pins from P0.0,P0.1 to P0.5,P0.6 Table 6-1: Changed table heading (Removed Mnemonics and made as Register names). Table 9-5: Included #of rows for different flash sizes Section10-1: Changed CPUCLK selectable options from n=0-5,7,8 to n=0-5,7 Clocks section: Changed ITMRCLK division to 1,2,3,4. updated the sources to ITMRCLK, TCAPCLKs. Mentioned P17 is TTL enabled permanently. Corrected FRT, PIT data write order. Updated INTCLR,INTMSK registers. in the register table also. DC spec sheet: changed LVR to LVD included max min programmable trip points based on char data. Updated the 50ma sink pins on 638xx, 63903. Keep-alive voltage mentioned corresponding to Keep-alive current of 20uA. Included Notes regarding VOL,VOH on P1.0,P1.1 and TMDO spec. AC Specs: T_{MDO1}, T_{SDO1} In description column changed Phase to 0 Section 5: Removed the VREG from the CY7C63310 and CY7C63801 Removed SCLK and SDATA. Created a separate pinout diagram for the CY7C63813 Added the GPIO Block Diagram (Figure 14-1.) *Table 10-4*: Changed the Sleep Timer Clock unit from 32 KHz count to Hz *Table 21-1*: Added more descriptions to the register
*E	341277	See ECN	BHA	Corrected V_{IH} TTL value in DC Characteristics table Updated V_{IL} TTL value Added footnote to pin description table for D+/D- pins Added Typical Values to Low Voltage Detect table Corrected Pin label on 16-pin PDIP package Corrected minor typos
*F	408017	See ECN	TYJ	Corrected pin assignment for the 24-pin QSOP package - GPIO port 3 in Table 5-1: Pin Assignment New Assignments: Pin 19 assigned to P3.0 and pin 20 to P3.1 Table 17.7 INT_MASK1 changed to 0xE1 Table 17.8 INT_MASK0 changed to 0xE0 Table 24.0-Register Summary, address E0 assigned to INT_MASK0 and address E1 assigned to INT_MASK1
*G	424790	See ECN	TYJ	Minor text changes to make document more readable Removed CY7C639xx Removed CY7C639xx from Ordering Information Table Added text concerning current draw for P0.0 and P0.1 in Table 5-1 Corrected Figure 9-2 to represent single stack Added comment about availability of 3.3V I/O on P1.3-P1.6 in Table 5-1 Added information on Flash endurance and data retention to section 9.3 Added block diagrams and timing diagrams Added CY7C638xx die form diagrams, Pad assignment tables and Ordering information Keyboard references removed CY7C63923-XC die diagram removed, removed references to the 639xx parts Updated part numbers in the header

30.0 Document History Page (continued)

Document Title: CY7C63310/CY7C638xx enCoRe™ II Low-Speed USB Peripheral Controller Document Number: 38-08035				
Rev.	ECN No.	Issue Date	Orig. of Change	Description of Change
*H	491711	See ECN	TYJ	Minor text changes 32-QFN part added Removed 638xx die diagram and die form pad assignment Removed GPIO port 4 configuration details Corrected GPIO characteristics of P0.0 and P0.1 to P1.0 and P1.1 respectively
*I	504691	See ECN	TYJ	Minor text changes Removed all residual references to external crystal oscillator and GPIO4 Documented the dedicated 3.3V regulator for USB transceiver Documented bandgap/voltage regulator behavior on wake up Voltage regulator line/load regulation documented USB Active and PS2 Data low interrupt trigger conditions documented. GPIO capacitance and timing diagram included Method to clear Capture Interrupt Status bit discussed Sleep and Wake up sequence documented. EP1MODE/EP2MODE register issue discussed