CY7C601xx
CY7C602xx

Wireless enCoRe™ II Microcontroller

1.0 Features

- **Wireless enCoRe™ II —"enhanced Component Reduction"**
  - **Internal crystalless oscillator with support for an optional external crystal or resonator.**
  - **Configurable IO for real-world interface without external components**
- **Enhanced 8-bit microcontroller**
  - **Harvard architecture**
  - **M8C CPU speed can be up to 12 MHz or sourced by an external crystal, resonator, or clock signal**
- **Internal memory**
  - **256 bytes of RAM**
  - **8 Kbytes of Flash including EEROM emulation**
- **Low power consumption**
  - **Typically 1.97mA at 3 MHz**
  - **5-µA sleep**
- **In-system reprogrammability**
  - **Allows easy firmware update**
- **General-purpose I/O ports**
  - **Up to 36 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**
- **SPI serial communication**
  - **Master or slave operation**
  - **Configurable up to 2-Mbit/second transfers**
  - **Supports half duplex single data line mode for optical sensors**
- **2-channel 8-bit or 1-channel 16-bit capture timer. 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**
- **Programmable Interval Timer interrupts**
- **Reduced RF emissions at 27 MHz and 96 MHz**
- **Watchdog timer (WDT)**
- **Low-voltage Detection with user-selectable threshold voltages**
- **Improved output drivers to reduce EMI**
- **Operating voltage from 2.7V to 3.6VDC**
- **Operating temperature from 0–70°C**
- **Available in 24/40-pin PDIP, 24-pin SOIC, 24-pin QSOP/SSOP, 28-pin SSOP, 48-pin SSOP, and DIE form**
- **Advanced development tools based on Cypress PSoC™ tools**
- **Industry-standard programmer support**

1.1 Applications

The CY7C601xx/CY7C602xx is targeted for the following applications:

- Wireless HID devices
  - Mice (optomechanical, optical, trackball)
  - Keyboards
  - Presenter tools
- General purpose wireless applications
  - Remote controls
  - Barcode scanners
  - POS terminal
  - Consumer electronics
  - Toys

2.0 Introduction

The Wireless enCoRe II family brings the features and benefits of the enCoRe II to non-USB applications. The enCoRe II family has an integrated oscillator that eliminates the external crystal or resonator, reducing overall cost. Also integrated into this chip are other external components such as wake-up circuitry.

The Wireless enCoRe II is a low-voltage, low-cost 8-bit Flash-programmable microcontroller

The Wireless enCoRe II features up to 36 general-purpose I/O (GPIO) pins. The I/O pins are grouped into five ports (Port 0 to 4). The pins on Port 0 and Port 1 may each be configured individually while the pins on Ports 2, 3, and 4 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. 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 with the exception of GPIO Port 0. GPIO Port 0 has three dedicated pins that have independent interrupt vectors (P0.2–P0.4).

The Wireless enCoRe II features an internal oscillator. Optionally, an external 1-MHz to 24-MHz crystal can be used to provide a higher precision reference.

The WIreless enCoRe II has 8 Kbytes of Flash for user's code and 256 bytes of RAM for stack space and user variables.

In addition, enCoRe II includes a Watchdog timer, a vectored interrupt controller, a 16-bit Free-Running Timer with Capture registers and a 12-bit Programmable Interval Timer. The Power-on reset circuit detects 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 it generates a reset or may be configured to generate an interrupt. There is a Low-voltage detect circuit that detects when V_{CC} drops below a programmable trip voltage and it may be configurable to generate a 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 17 maskable interrupts in the vectored interrupt controller. All interrupts can be masked. Interrupt sources include LVR/POR, a programmable interval timer, a nominal 1.024-ms programmable output from the Free Running Timer, two capture timers, five GPIO Ports, three GPIO pins, two SPI, a 16-bit free-running timer wrap and an internal wake-up timer interrupt. The wake-up timer causes periodic interrupts when enabled. The capture timers interrupt whenever a new timer value is saved due to a selected GPIO edge event. A total of eight 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 timer generates an interrupt at 1024-µs rate. It can also generate an interrupt when the free-running counter overflow occurs—every 16.384 ms. The timer can be used to measure the duration of an event under firmware control by reading the timer at the start and at the end of an event, then calculating the difference between the two values. The two 8-bit capture timers 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 Wireless enCoRe II supports in-system programming by using the P1.0 and P1.1 pins as the serial programming mode interface.

3.0 Conventions

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

Available in all Wireless enCoRe II family members

CY7C601xx only

4.0 Logic Block Diagram

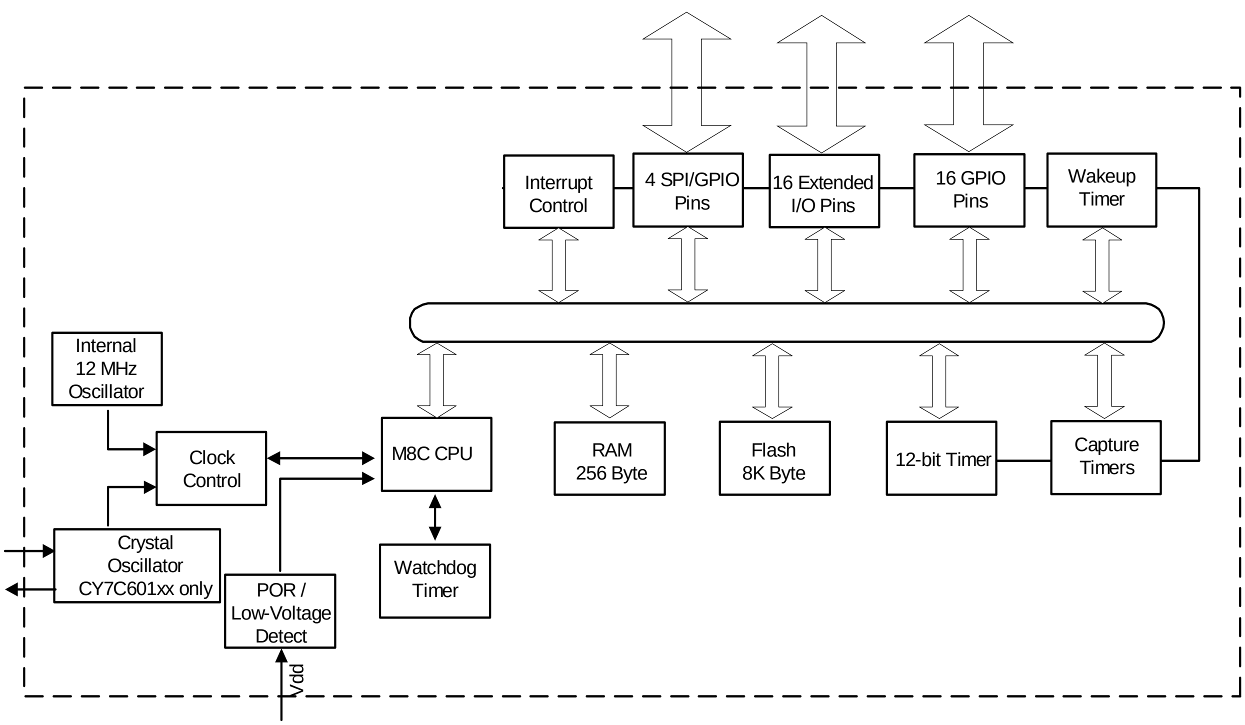


Figure 4-1. CY7C601xx/CY7C602xx Block Diagram

5.0 Packages/Pinouts

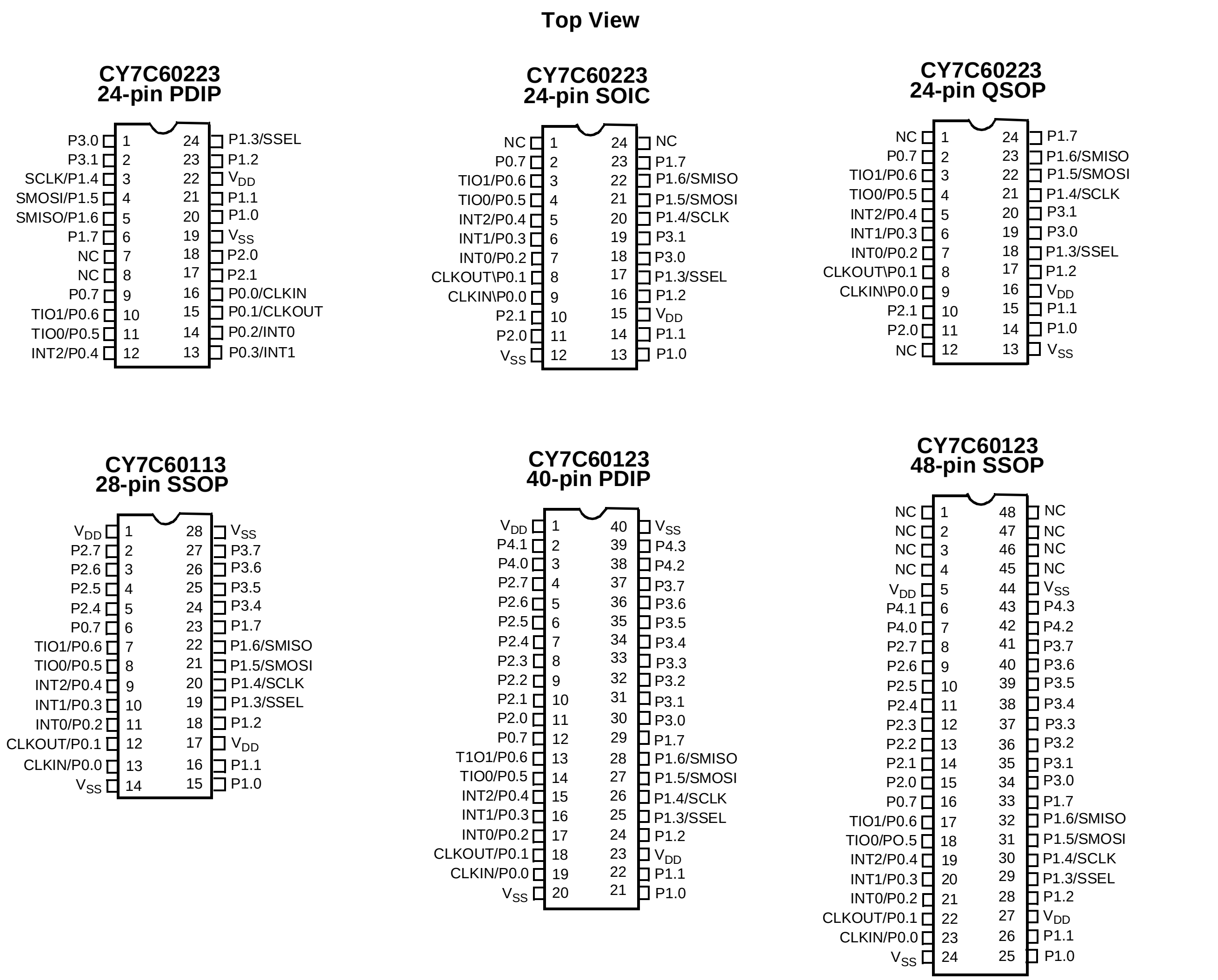


Figure 5-1. Package Configurations

5.1 Pinouts Assignments

Table 5-1. Pin Assignments

48 SSOP	40 PDIP	28 SSOP	24 QSOP	24 SOIC	24 PDIP	Name	Description
7	3					P4.0	GPIO Port 4—configured as a group (nibble)
6	2					P4.1	
42	38					P4.2	
43	39					P4.3	
34	30		19	18	1	P3.0	GPIO Port 3—configured as a group (byte)
35	31		20	19	2	P3.1	
36	32					P3.2	
37	33					P3.3	
38	34	24				P3.4	
39	35	25				P3.5	
40	36	26				P3.6	
41	37	27				P3.7	
15	11		11	11	18	P2.0	GPIO Port 2—configured as a group (byte)
14	10		10	10	17	P2.1	
13	9					P2.2	
12	8					P2.3	
11	7	5				P2.4	
10	6	4				P2.5	
9	5	3				P2.6	
8	4	2				P2.7	
25	21	15	14	13	20	P1.0	GPIO Port 1 bit 0
26	22	16	15	14	21	P1.1	GPIO Port 1 bit 1
28	24	18	17	16	23	P1.2	GPIO Port 1 bit 2
29	25	19	18	17	24	P1.3/SSEL	GPIO Port 1 bit 3—Configured individually. Alternate function is SSEL signal of the SPI bus
30	26	20	21	20	3	P1.4/SCLK	GPIO Port 1 bit 4—Configured individually. Alternate function is SCLK signal of the SPI bus
31	27	21	22	21	4	P1.5/SMOSI	GPIO Port 1 bit 5—Configured individually. Alternate function is SMOSI signal of the SPI bus
32	28	22	23	22	5	P1.6/SMISO	GPIO Port 1 bit 6—Configured individually. Alternate function is SMISO signal of the SPI bus
33	29	23	24	23	6	P1.7	GPIO Port 1 bit 7—Configured individually. TTL voltage threshold
23	19	13	9	9	16	P0.0/CLKIN	GPIO Port 0 bit 0—Configured individually. On CY7C601xx, optional Clock In when external oscillator is disabled or external oscillator input when external oscillator is enabled. On CY7C602xx, oscillator input when configured as Clock In If this pin is used as a General Purpose output it will draw current. This pin should be configured as an input to reduce current draw.

Table 5-1. Pin Assignments (continued)

48 SSOP	40 PDIP	28 SSOP	24 QSOP	24 SOIC	24 PDIP	Name	Description
22	18	12	8	8	15	P0.1 / CLKOUT	GPIO Port 0 bit 1—Configured individually On CY7C601xx, optional clock out when external oscillator is disabled or external oscillator output drive when external oscillator is enabled. On CY7C602xx, oscillator output when configured as Clock out. If this pin is used as a General Purpose output it will draw current. This pin should be configured as an input to reduce current draw.
21	17	11	7	7	14	P0.2/INT0	GPIO port 0 bit 2—Configured individually Optional rising edge interrupt INT0
20	16	10	6	6	13	P0.3/INT1	GPIO port 0 bit 3—Configured individually Optional rising edge interrupt INT1
19	15	9	5	5	12	P0.4/INT2	GPIO port 0 bit 4—Configured individually Optional rising edge interrupt INT2
18	14	8	4	4	11	P0.5/TIO0	GPIO port 0 bit 5—Configured individually Alternate function Timer capture inputs or Timer output TIO0
17	13	7	3	3	10	P0.6/TIO1	GPIO port 0 bit 6—Configured individually Alternate function Timer capture inputs or Timer output TIO1
16	12	6	2	2	9	P0.7	GPIO port 0 bit 7—Configured individually
1,2,3,4			1	1	7	NC	No connect
45,46, 47,48			12	24	8	NC	No connect
5	1	17				V_{DD}	Power
27	23	1	16	15	22		
44	40	14	–	–	–	V_{SS}	
24	20	28	13	12	19		

6.0 Register Summary

Wireless enCoRe II 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/ CLKOUT	P0.0/CLKIN	bbbbbbbb	00000000
01	P1DATA	P1.7	P1.6/SMISO	P1.5/SMOSI	P1.4/SCLK	P1.3/SSEL	P1.2	P1.1	P1.0	bbbbbbbb	00000000
02	P2DATA	P2.7–P2.2						P2.1–P2.0		bbbbbbbb	00000000
03	P3DATA	P3.7–P3.2						P3.1–P3.0		bbbbbbbb	00000000
04	P4DATA	Reserved				P4.3–P4.0				----bbbb	00000000
05	P00CR	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
06	P01CR	CLK Output	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable	bbbbbbbb	00000000
07–09	P02CR– P04CR	Reserved		Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	--bb-bbb	00000000
0A–0B	P05CR– P06CR	TIO Output	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	bbbb-bbb	00000000
0C	P07CR	Reserved	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	-bbb-bbb	00000000
0D	P10CR	Reserved	Int Enable	Int Act Low	Reserved				Output Enable	-bb----b	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 Threshold	Reserved	Open Drain	Pull-up Enable	Output Enable	bbbb-bbb	00000000
10	P13CR	Reserved	Int Enable	Int Act Low	Reserved	High Sink	Open Drain	Pull-up Enable	Output Enable	-bb-bbbb	00000000
11–13	P14CR– P16CR	SPI Use	Int Enable	Int Act Low	Reserved	High Sink	Open Drain	Pull-up Enable	Output Enable	bbb-bbbb	00000000
14	P17CR	Reserved	Int Enable	Int Act Low	Reserved	High Sink	Open Drain	Pull-up Enable	Output Enable	-bb-bbbb	00000000
15	P2CR	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
16	P3CR	Reserved	Int Enable	Int Act Low	TTL Thresh	High Sink	Open Drain	Pull-up Enable	Output Enable	-bbbbbbb	00000000
17	P4CR	Reserved	Int Enable	Int Act Low	TTL Thresh	Reserved	Open Drain	Pull-up Enable	Output Enable	-bbb-bbb	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]								rrrrrrrr	00000000
23	TCAP1R	Capture 1 Rising [7:0]								rrrrrrrr	00000000
24	TCAP0F	Capture 0 Falling [7:0]								rrrrrrrr	00000000
25	TCAP1F	Capture 1 Falling [7:0]								rrrrrrrr	00000000
26	PITMRL	Prog Interval Timer [7:0]								rrrrrrrr	00000000
27	PITMRH	Reserved				Prog Interval Timer [11:8]				----rrrr	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							CPU CLK Select	-------b	00000000
31	TMRCLKCR	TCAPCLK Divider		TCAPCLK Select		ITMRCLK Divider		ITMRCLK Select		bbbbbbbb	10001111
32	CLKIOCR	Reserved			XOSC Select	XOSC Enable	EFTB Disabled	CLKOUT Select		---bbbbb	00000000
34	IOSCTR	foffset[2:0]			Gain[4:0]					bbbbbbbb	000ddddd
35	XOSCTR	Reserved			XOSC XGM [2:0]			Reserved	Mode	---bbb-b	000ddddd
36	LPOSCTR	32-kHz Low Power	Reserved	32-kHz Bias Trim [1:0]		32-kHz Freq Trim [3:0]				b-bbbbbb	d-dddddd

Wireless enCoRe II 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
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	Reserved					bbb-----	00000000
DC	INT_CLR2	Reserved	GPIO Port 4	GPIO Port 3	GPIO Port 2	Reserved	INT2	16-bit Counter Wrap	TCAP1	-bbb-bbb	00000000
DE	INT_MSK3	ENSWINT	Reserved							r-------	00000000
DF	INT_MSK2	Reserved	GPIO Port 4 Int Enable	GPIO Port 3 Int Enable	GPIO Port 2 Int Enable	Reserved	INT2 Int Enable	16-bit Counter Wrap Int Enable	TCAP1 Int Enable	-bbb-bbb	00000000
E1	INT_MSK1	TCAP0 Int Enable	Prog Interval Timer Int Enable	1-ms Timer Int Enable	Reserved					bbb-----	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
F7	CPU_F	Reserved			XIO	Super	Carry	Zero	Global IE	---brbbb	00000010
FF	CPU_SCR	GIES	Reserved	WDRS	PORS	Sleep	Reserved	Reserved	Stop	r-ccb--b	00010100
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-bbb	00000000
1EB	ECO_TR	Sleep Duty Cycle [1:0]		Reserved						bb------	00000000
1E4	VLTCMP	Reserved						LVD	PPOR	------rr	00000000

Note: In the R/W column,

b = Both Read and Write

r = Read Only

w = Write Only

c = Read/Clear

d = calibration value. Should not change during normal use

7.0 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 7-1. CPU Registers and Register Name

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 9-1*.

8.0 CPU Registers

8.1 Flags Register

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

Table 8-1. 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: This register is readable with explicit address 0xF7. The *OR F, expr* and *AND F, expr* must be used to set and clear the CPU_F bits.

8.1.1 Accumulator Register

Table 8-2. 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.

8.1.2 Index Register

Table 8-3. 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.

8.1.3 Stack Pointer Register

Table 8-4. 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.								

8.1.4 CPU Program Counter High Register

Table 8-5. 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.								

8.1.5 CPU Program Counter Low Register

Table 8-6. 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.								

8.2 Addressing Modes

8.2.1 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. Instructions using this addressing mode are two bytes in length.

Table 8-7. 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.
```

8.2.2 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 8-8. 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.
```

8.2.3 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.

Table 8-9. 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.
```

8.2.4 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 8-10. 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 Accumula-
                        ;tor is moved to the regis-
                        ;ter space location at
                        ;address 8. The Accumulator
                        ;is unchanged.
```

8.2.5 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 8-11. 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.
```

8.2.6 Destination Direct 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 8-12. Destination Direct Immediate

Opcode	Operand 1	Operand 2
Instruction	Destination Address	Immediate Value

Examples

```
ADD   [7],      5    ;In this case, value in the mem-
                     ;ory 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.
```

8.2.7 Destination Indexed 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 8-13. Destination Indexed Immediate

Opcode	Operand 1	Operand 2
Instruction	Destination Index	Immediate Value

Examples

```
ADD    [X+7],      5      ;In this case, the value in
                          ;the memory 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 immedi-
                          ;ate value of 6 is moved
                          ;into the location in the
                          ;register space at
                          ;address X+8.
```

8.2.8 Destination Direct 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 8-14. Destination Direct Direct

Opcode	Operand 1	Operand 2
Instruction	Destination Address	Source Address

Example

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

8.2.9 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 8-15. Source Indirect Post Increment

Opcode	Operand 1
Instruction	Source Address Address

Example

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

8.2.10 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 8-16. 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 loca-
                     ;tion pointed to by the indi-
                     ;rect address. The indirect
                     ;address is then incremented.
```

9.0 Instruction Set Summary

The instruction set is summarized in *Table 9-1* 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 9-1. Instruction Set Summary Sorted Numerically by Opcode Order[1, 2]

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:
1. Interrupt routines take 13 cycles before execution resumes at interrupt vector table.
2. The number of cycles required by an instruction is increased by one for instructions that span 256-byte boundaries in the Flash memory space.

10.0 Memory Organization

10.1 Flash Program Memory Organization

after reset: 16-bit PC → 0x0000

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	Reserved
0x0024	Reserved
0x0028	Reserved
0x002C	Reserved
0x0030	Reserved
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	Reserved
0x0050	GPIO Port 2
0x0054	GPIO Port 3
0x0058	GPIO Port 4
0x005C	Reserved
0x0060	Reserved
0x0064	Sleep Timer
0x0068	**Program Memory begins here (if below interrupts not used, program memory can start lower)**
0x1FFF	

Figure 10-1. Program Memory Space with Interrupt Vector Table

10.2 Data Memory Organization

The CY7C601xx/CY7C602xx microcontrollers provide up to 256 bytes of data RAM.

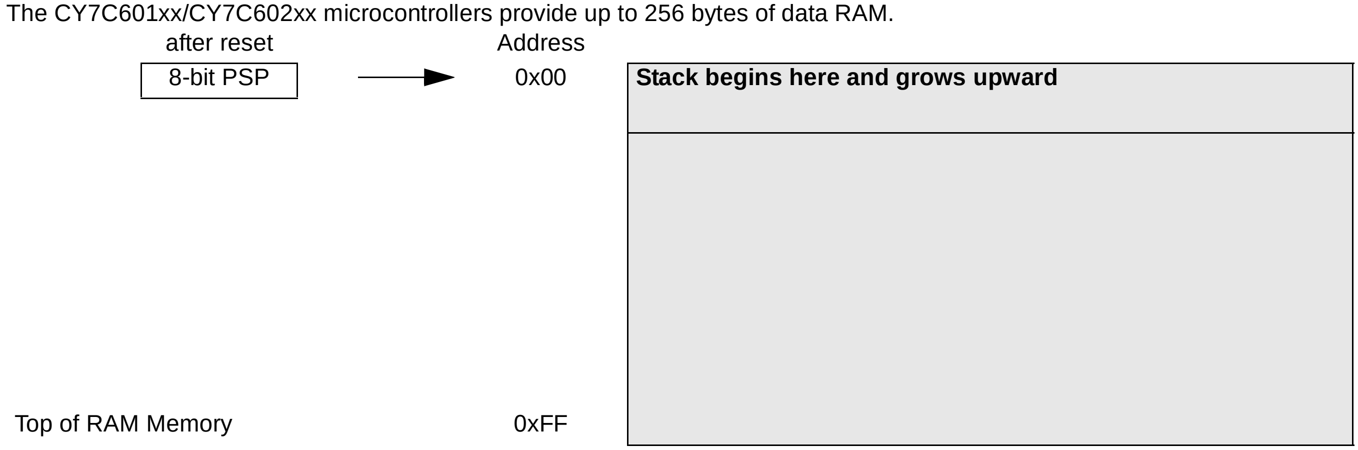


Figure 10-2. Data Memory Organization

10.3 Flash

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

10.3.1 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 should not be over written 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.

10.3.2 In-System Programming

Wireless enCoRe II devices enable this type of in-system programming by using the P1.0 and P1.1 pins as the serial programming mode interface. This allows an external controller to cause the Wireless enCoRe II part to enter serial programming mode and then to use the test queue to issue Flash access functions in the SROM.

10.4 SROM

The SROM holds code that is used to boot the part, calibrate circuitry, and perform Flash operations (*Table 10-1* 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 10-1*. 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 10-2* 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 10-1. 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 set-up code and not run into it.

```
halt
SSCOP: mov [KEY1], 3ah
```

```
mov X, SP
mov A, X
add A, 3
mov [KEY2], A
```

Table 10-2. 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

The SROM also features Return Codes and Lockouts.

10.4.1 Return Codes

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 10-3. 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.

10.5 SROM Function Descriptions

10.5.1 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. A Wireless 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.

10.5.2 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 10-4. 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 should be stored

10.5.3 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. Refer to “Clocking” Section for additional information.

Table 10-5. WriteBlock Parameters

Name	Address	Description
KEY1	0,F8h	3Ah
KEY2	0,F9h	Stack Pointer value, when SSC is executing
BLOCK ID	0,FAh	8KB Flash block number (00h–7Fh) 4KB Flash block number (00h–3Fh) 3KB Flash block number (00h–2Fh)
POINTER	0,FBh	First 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

10.5.4 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 10-6. 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

10.5.5 ProtectBlock Function

The Wireless enCoRe II devices offer Flash protection on a block-by-block basis. *Table 10-7* 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 10-7*. 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 10-7*.

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 10-7. 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	$\overline{SR}$ $\overline{ER}$ $\overline{EW}$ 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 10-8. 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

10.5.6 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 10-9. 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

10.5.7 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 10-10. 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 Wireless 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 Wireless enCoRe II 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 Wireless enCoRe II. 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.

10.5.8 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 10-11. 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

11.0 Clocking

The Wireless 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. The Internal 24-MHz Oscillator accuracy is 24 MHz –22% to +10% (between 0°–70°C). No external components are required to achieve this level of accuracy.

Firmware is responsible for selecting the correct trim values from the User row to match the power supply voltage in the end application and writing the values to the trim registers IOSCTR and LPOSCTR.

The internal low-speed oscillator of nominally 32-kHz provides a slow clock source for the Wireless 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 from –53.12% to +56.25%. The 32-kHz low power oscillator can be calibrated against the internal 24-MHz oscillator or another timing source if desired.

Wireless enCoRe II provides the ability to load new trim values for the 24-MHz oscillator based on voltage. This allows Vdd to be monitored and have firmware trim the oscillator based on voltage present. The IOSCTR register is used to set trim values for the 24-MHz oscillator. Wireless enCoRe II is initialized with 3.30V trim values at power-on, then firmware is responsible for transferring the correct set of trim values to the trim registers to match the application's actual Vdd. The 32-kHz oscillator generally does not require trim adjustments vs. voltage but trim values for the 32-kHz are also stored in Supervisory ROM.

Table 11-1. Oscillator Trim Values vs. Voltage Settings

Supervisory FLASH User Row Address	Function
0xC094	24-MHz IOSCTR @ 3.30V
0xC095	24-MHz IOSCTR @ 3.00V
0xC096	24-MHz IOSCTR @ 2.85V
0xC097	24-MHz IOSCTR @ 2.70V
0xC098	32-kHz LPOSCTR@3.30V
0xC099	32-kHz LPOSCTR@3.00V
0xC09A	32-kHz LPOSCTR@2.85V
0xC09B	32-kHz LPOSCTR@2.70V

When using the 32KHz oscillator the PITMRL/H should 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
```

The CY7C601xx part can optionally be sourced from an external crystal oscillator. The external clock driving on CLKIN range is from 187KHz to 24MHz.

11.1 Clock Architecture Description

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

On the CY7C601xx, the external oscillator can be sourced by the crystal oscillator or when the crystal oscillator is disabled it is sourced directly from the CLKIN pin. The external crystal oscillator is fed through the EFTB block, which can optionally be bypassed.

11.1.1 CPU Clock

The CPU clock, CPUCLK, can be sourced from the external crystal oscillator, the Internal 24-MHz Oscillator, or the Internal 32-KHz Low-power Oscillator. The selected clock source can optionally be divided by 2^{n-1} where *n* is 0–7 (see *Table 11-2*).

When it is not being used by the external crystal oscillator, the CLKOUT pin 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 Oscillator
- CPUCLK after the programmable divider

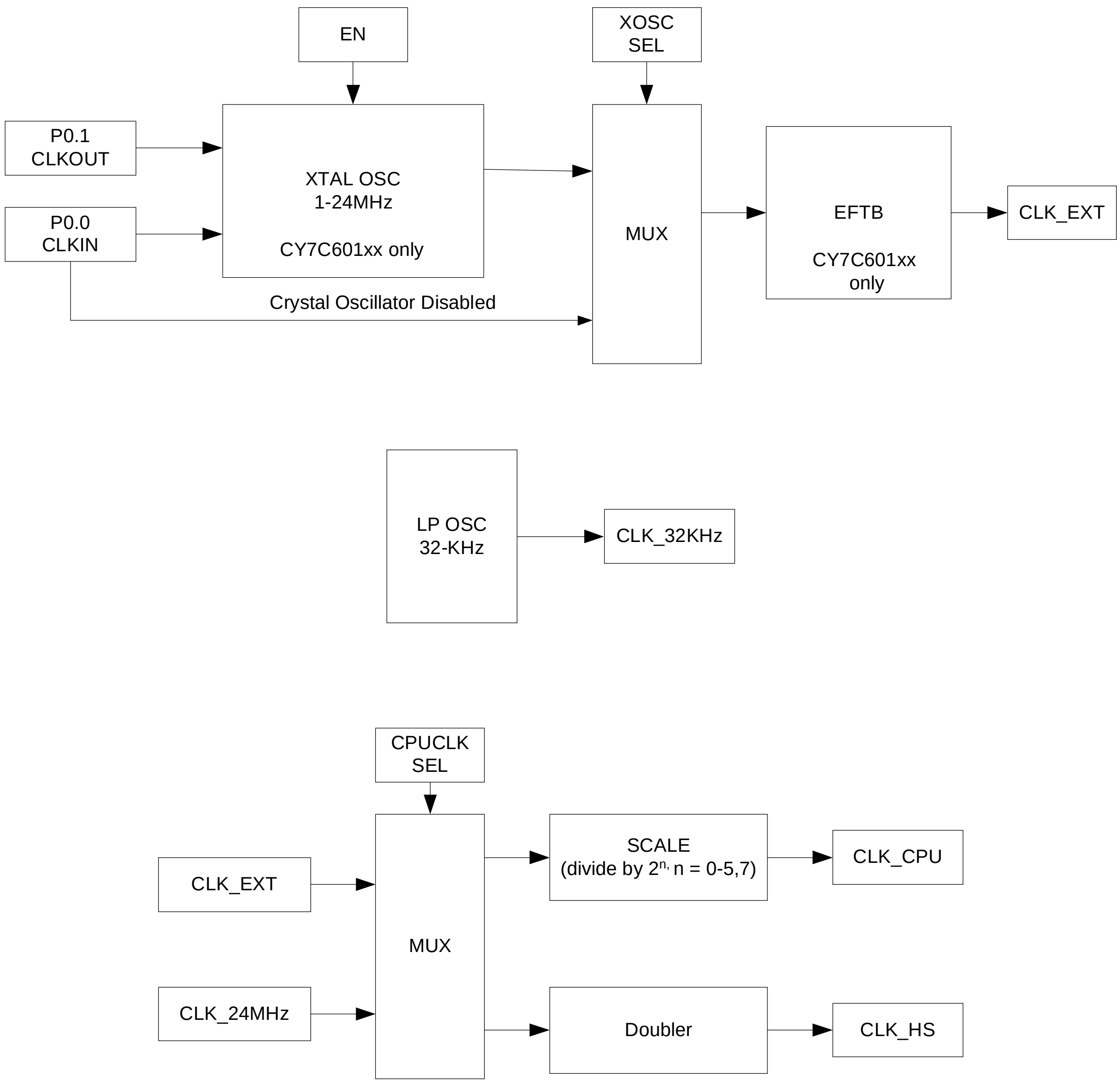


Figure 11-1. CPU Clock Block Diagram

Table 11-1. CPU Clock Config CPUCLKCR) [0x30] [R/W]

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

Bit [7:1]: Reserved
Bit 0: CPU CLK Select
0 = Internal 24-MHz Oscillator.
1 = External crystal oscillator—External crystal oscillator on CLKIN and CLKOUT if the external crystal oscillator is enabled, CLKIN input if the external crystal oscillator is disabled.
Note: the CPU speed selection is configured using the OSC_CR0 Register (*Table 11-2*.)

Table 11-2. 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 12-1*), 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 14-3*). 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. Obtaining the absolute lowest power usage in sleep mode requires the No Buzz bit be clear.
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 Wireless enCoRe II ma y operate over a range of CPU clock speeds. The reset value for the CPU Speed bits is zero; therefore, the default CPU speed is 3MHz.

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	Reserved	Reserved
100	1.5 MHz	Clock In / 16
101	750 KHz	Clock In / 32
110	187 KHz	Clock In / 128
111	Reserved	Reserved

Table 11-3. Clock I/O Config (CLKIOCR) [0x32] [R/W]

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

Bit [7:5]: Reserved
Bit 4: XOSC Select
This bit, when set, selects the external crystal oscillator clock as clock source of external clock. Care needs to be taken while selecting the crystal oscillator clock. First enable the crystal oscillator and wait for few cycles, which is oscillator stabilization period. Then select the crystal clock as clock source. Similarly, while deselect crystal clock, first deselect crystal clock as clock source then disable the crystal oscillator.
0 = Not select external crystal oscillator clock
1 = Select the external crystal oscillator clock
Bit 3: XOSC Enable
This bit is only available on the CY7C601xx
This bit when set enables the external crystal oscillator. The external crystal oscillator shares pads CLKIN and CLKOUT with two GPIOs—P0.0 and P0.1, respectively. When the external crystal oscillator is enabled, the CLKIN signal comes from the external crystal oscillator block and the output enables on the GPIOs for P0.0 and P0.1 are disabled, eliminating the possibility of contention. When the external crystal oscillator is disabled the source for CLKIN signal comes from the P0.0 GPIO input.
0 = Disable the external oscillator
1 = Enable the external oscillator
Note: The external crystal oscillator start-up time takes up to 2 ms.
Bit 2: EFTB Disabled
This bit is only available on the CY7C601xx.
0 = Enable the EFTB filter
1 = Disable the EFTB filter, causing CLKIN to bypass the EFTB filter
Bit [1:0]: CLKOUT Select
0 0 = Internal 24-MHz Oscillator
0 1 = External crystal oscillator – external crystal oscillator on CLKIN and CLKOUT if the external crystal oscillator is enabled, CLKIN input if the external oscillator is disabled.
1 0 = Internal 32-kHz Low-power Oscillator
1 1 = CPUCLK

11.1.2 Interval Timer Clock (ITMRCLK)

The Interval Timer clock (ITMRCLK), can be sourced from the external crystal oscillator, the Internal 24-MHz Oscillator, the Internal 32-kHz Low-power Oscillator, or the Timer Capture clock. A programmable prescaler of 1, 2, 3, 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 should 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 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 PITIMER_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)

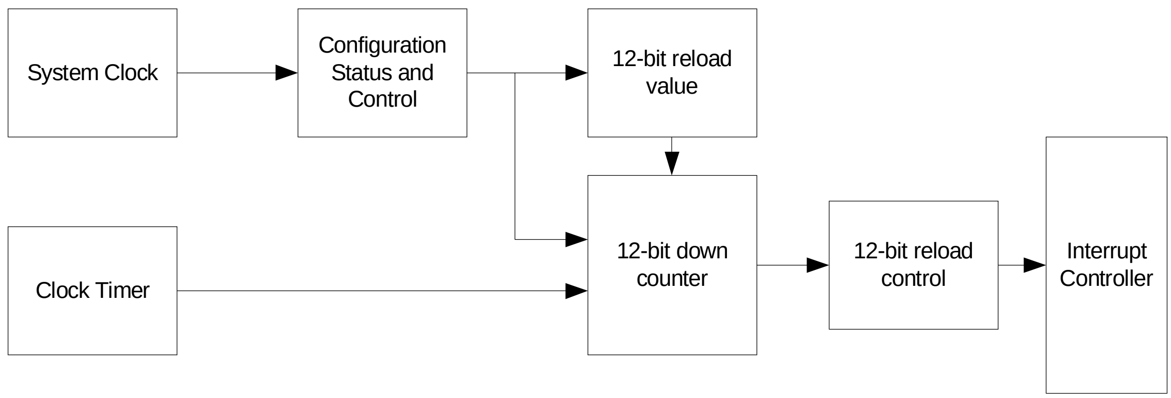


Figure 11-2. Programmable Interval Timer Block Diagram

11.1.3 Timer Capture Clock (TCAPCLK)

The Timer Capture clock (TCAPCLK) can be sourced from the external crystal oscillator, Internal 24-MHz Oscillator or the Internal 32-kHz Low-power Oscillator. A programmable prescaler of 2, 4, 6, or 8 then divides the selected source.

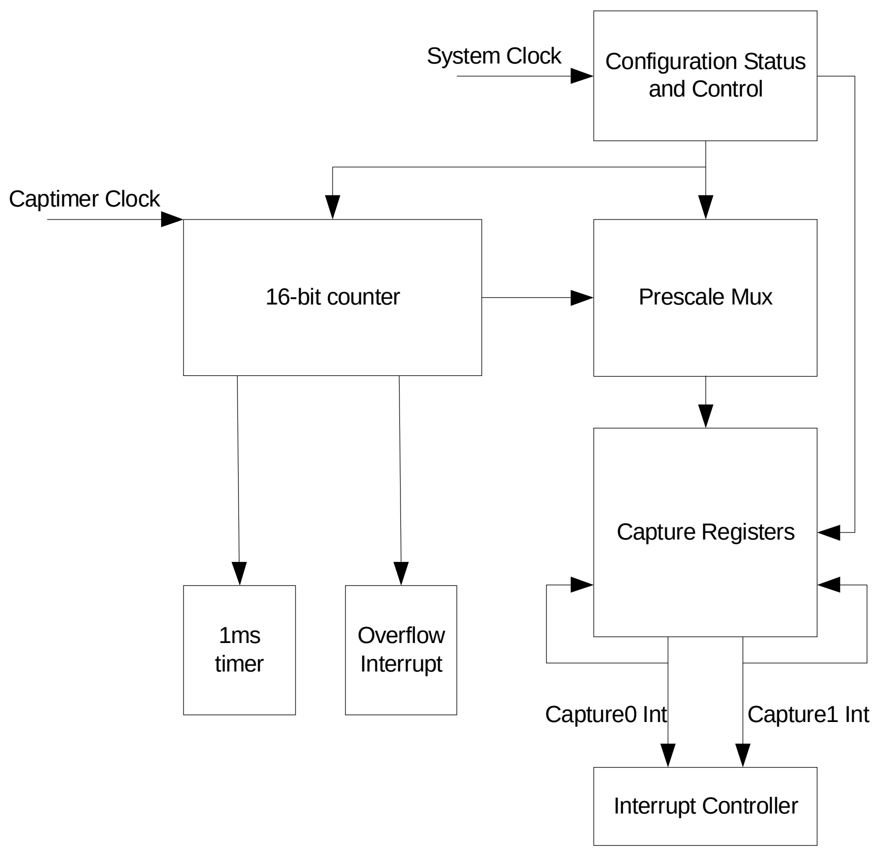


Figure 11-3. Timer Capture Block Diagram

Table 11-4. 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 crystal oscillator—external crystal oscillator on CLKIN and CLKOUT if the external crystal oscillator is enabled, CLKIN input if the external crystal oscillator is disabled (the XOSC Enable bit of the CLKIOCR Register is cleared—*Table 11-3*.)
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 crystal oscillator—external crystal oscillator on CLKIN and CLKOUT if the external crystal oscillator is enabled, CLKIN input if the external crystal oscillator is disabled.
1 0 = Internal 32-kHz Low-power Oscillator
1 1 = TCAPCLK
Note: Changing the source of TMRCLK requires that both the source and destination clocks be running. Attempting to change the clock source away from TCAPCLK after that clock has been stopped will not be successful.

11.1.4 Internal Clock Trim

Table 11-5. 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. The “D” indicates that the default value is trimmed to 24-MHz @ 3.30V at power-on.
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

11.1.5 External Clock Trim

Table 11-6. XOSC Trim (XOSCTR) [0x35] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved			XOSC XGM [2:0]			Reserved	Mode
Read/Write	–	–	–	R/W	R/W	R/W	–	R/W
Default	0	0	0	D	D	D	–	D

This register is used to calibrate the external crystal oscillator. The reset value is undefined but during boot the SROM writes a calibration value that is determined during manufacturing test. This is the meaning of ‘D’ in the Default field.
Bit [7:5]: Reserved
Bit [4:2]: XOSC XGM [2:0]
Amplifier transconductance setting. The Xgm settings are recommended for resonators with frequencies of interest for the Wireless enCoRe II as below:

Resonator	XGM Setting	Worst Case R (Ohms)
6-MHz Crystal	001	403
12-MHz Crystal	011	201
Reserved	111	-
6-MHz Ceramic	001	70.4
12-MHz Ceramic	011	41

Bit 1: Reserved
Bit 0: Mode
0 = Oscillator Mode
1 = Fixed Maximum Bias test Mode

11.1.6 LPOSC Trim

Table 11-7. 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	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 is the meaning of ‘D’ in the Default field. The trim value can be adjusted vs. voltage as noted in *Table 11-1*.
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.

11.2 CPU Clock During Sleep Mode

When the CPU enters sleep mode the CPUCLK Select (Bit 1, *Table 11-1*) 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. If the external clock source is the external oscillator and the oscillator is disabled, firmware will need to enable the external oscillator, wait for it to stabilize, and then change the clock source.

12.0 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 12-1. 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	Reserved	Stop
Read/Write	R	–	R/C[3]	R/C[3]	R/W	–	–	R/W
Default	0	0	0	1	0	1	0	0

The bits of the CPU_SCR register are used to convey status and control of events for various functions of an Wireless 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)

Note:
3. C = Clear. This bit can only be cleared by the user and cannot be set by firmware.

12.1 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 V_{CC} in 0 to 200 ms.

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

12.2 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 11-2*). 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.

Table 12-2. 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 the Watchdog Timer, a write of 0x38 will also clear the Sleep Timer. **Bit [7:0]:** Reset Watchdog Timer [7:0]								

13.0 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 the 32-kHz Low Power bit, *Table 11-7*). This will help save approximately 5 µA; however, the trade off is that the 32-KHz Low-speed Oscillator will be less accurate (–53.12% to +56.25% deviation).

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 should be disabled through the Interrupt Mask Registers.

When the CPU enters sleep mode the CPUCLK Select (Bit 1, *Table 11-1*) 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 the external crystal oscillator is used, firmware will need to switch the clock source for the CPU.

Unlike the Internal 24-MHz Oscillator, the external oscillator is not automatically shut-down during sleep. Systems that need the external oscillator disabled in sleep mode will need to disable the external oscillator prior to entering sleep mode. In systems where the CPU runs off the external oscillator, firmware will need to switch the CPU to the internal oscillator prior to disabling the external oscillator.

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.

14.0 Low-voltage Detect Control

Table 14-1. 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 circuit. 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.
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 = Reserved
1 1 = PPOR will not generate a reset, but values read from the Voltage Monitor Comparators Register (*Table 14-2*) 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.	Max.	Typical
000	2.69	2.72	2.7
001	2.90	2.94	2.92
010	3.00	3.04	3.02
011	3.10	3.15	3.13
100	Reserved		
101	Reserved		
110	Reserved		
111	Reserved		

14.0.1 POR Compare State

Table 14-2. 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 14-1*.)
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 tripped
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.

14.0.2 ECO Trim Register

Table 14-3. 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
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.

15.0 General Purpose I/O Ports

15.1 Port Data Registers

15.1.1 P0 Data

Table 15-1. 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
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 15-9*.)
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 15-8*)
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 15-7*.)
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 15-6*.)

15.1.2 P1 Data

Table 15-2. 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	P1.1	P1.0
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

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
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 15-14*.)
Bit [2:0]: P1.2–P1.0

15.1.3 P2 Data

Table 15-3. P2 Data Register (P2DATA) [0x02] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	P2.7–P2.2						P2.1 – P2.0	
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

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]: P2 Data [7:2]
Bit [1:0]: P2 Data [1:0]

15.1.4 P3 Data

Table 15-4. P3 Data Register (P3DATA) [0x03] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	P3.7–P3.2						P3.1– P3.0	
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

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]: P3 Data [7:2]
Bit [1:0]: P3 Data [1:0]

15.1.5 P4 Data

Table 15-5. P4 Data Register (P4DATA) [0x04] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved				P4.3–P4.0			
Read/Write	–	–	–	–	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

This register contains the data for Port 4. 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 [7:4]: Reserved
Bit [3:0]: P4 Data [3:0]
P4.3–P4.0 only exist in the CY7C601xx

15.2 GPIO Port Configuration

All the GPIO configuration registers have common configuration controls. The following are the bit definitions of the GPIO configuration registers. By default all GPIOs are configured as inputs. In order to prevent the inputs from floating the pull-up resistors are enabled. Firmware will need to configure each of the GPIOs prior to use.

15.2.1 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 find value in applications as well.

15.2.2 Int Act Low

When clear, the corresponding interrupt is active HIGH. When set, the interrupt is active LOW. For P0.2–P0.4 Int act Low causes interrupts to be active on the rising edge. Int act Low set causes interrupts to be active on the falling edge.

15.2.3 TTL Thresh

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

Important Note: The GPIOs default to CMOS threshold. User’s firmware needs to configure the threshold to TTL mode if necessary.

15.2.4 High Sink

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

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

On the CY7C601xx, only the P3.7, P2.7, P0.1, and P0.0 have 50-mA sink drive capability. Other pins have 8-mA sink drive capability.

On the CY7C602xx, only the P1.7–P1.3 have 50-mA sink drive capability. Other pins have 8-mA sink drive capability.

15.2.5 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.

15.2.6 Pull-up Enable

When set the pin has a 7K pull-up to V_{DD}.

When clear, the pull-up is disabled.

15.2.7 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.

P0.0(CLKIN) and P0.1(CLKOUT) can not be output enabled when the crystal oscillator is enabled. Output enables for these pins are overridden by XOSC Enable.

P1.3(SSEL), P1.4(SCLK), P1.5(SMOSI) and P1.6(SMISO) can be used for their dedicated functions or for GPIO. To enable the pin for GPIO use, clear the corresponding SPI Use bit or the Output Enable will have no effect.

15.2.8 SPI Use

The 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 SPI Use bit. The SPI function controls the output enable for its dedicated function pins when their GPIO enable bit is clear.

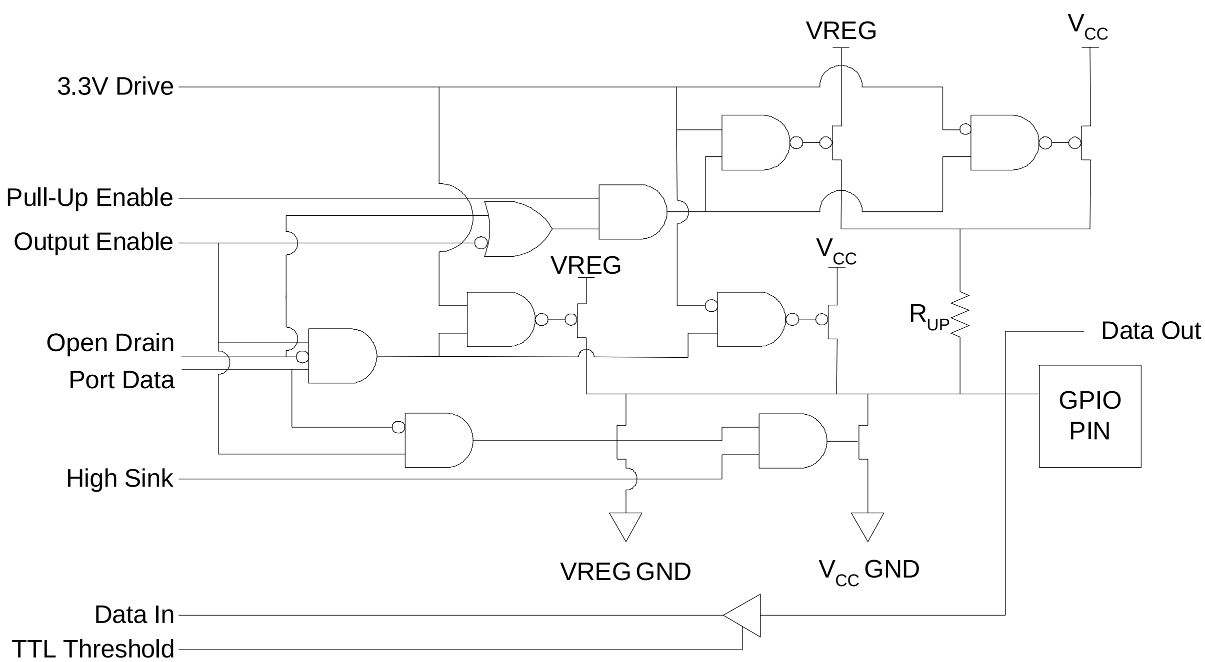


Figure 15-1. GPIO Block Diagram

15.2.9 P0.0/CLKIN Configuration

Table 15-6. 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 the external crystal oscillator. When the external oscillator is enabled the settings of this register are ignored.
The alternate function of the pin as the CLKIN is only available in the CY7C601xx. When the external oscillator is enabled (the XOSC Enable bit of the CLKIOCR Register is set—*Table 11-3*), the GPIO function of the pin is disabled.
The 50-mA sink drive capability is only available in the CY7C601xx. In the CY7C602xx, only 8-mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.
If this pin is used as a General Purpose output it will draw current. This pin should be configured as an input to reduce current draw.

15.2.10 P0.1/CLKOUT Configuration

Table 15-7. 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 for the external crystal oscillator. When the external oscillator is enabled the settings of this register are ignored. When CLK output is set, the internally selected clock is sent out onto P0.1CLKOUT pin.
The alternate function of the pin as the CLKOUT is only available in the CY7C601xx. When the external oscillator is enabled (the XOSC Enable bit of the CLKIOCR Register is set—*Table 11-3*), the GPIO function of the pin is disabled.
The 50-mA sink drive capability is only available in the CY7C601xx. In the CY7C602xx, only 8-mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.
If this pin is used as a General Purpose output it will draw current. This pin should be configured as an input to reduce current draw
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 11-3*) is driven out to the pin

15.2.11 P0.2/INT0–P0.4/INT2 Configuration

Table 15-8. 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. 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 18-8* and *Table 18-6*).
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

15.2.12 P0.5/TIO0–P0.6/TIO1 Configuration

Table 15-9. 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	R/W
Default	0	0	0	0	0	0	0	0

These registers control the operation of pins P0.5 through P0.6, respectively.
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

15.2.13 P0.7 Configuration

Table 15-10. 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

15.2.14 P1.0 Configuration

Table 15-11. P1.0 Configuration (P10CR) [0x0D] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	Reserved			P1.0 and P1.1 Pull-up Enable	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 P1.0 pin.
Bit1: P1.0 and P1.1 Pull-up Enable
0 = Disable the P1.0 and P1.1 pull-up resistors
1 = Enable the internal pull-up resistors for both the P1.0 and P1.1. Each of the P1.0 and P1.1 pins is pulled up with R_{UP1} (Section 20.0)
Note: There is no 2-mA sourcing capability on this pin. The pin can only sink 5 mA at V_{OL3} (Section 20.0.)
The P1.0 is an open drain only output. It can actively drive a signal low, but cannot actively drive a signal high.

15.2.15 P1.1 Configuration

Table 15-12. P1.1 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 pin.
The pull-up resistor on this pin is enabled by the P10CR Register
Note: There is no 2-mA sourcing capability on this pin. The pin can only sink 5 mA at V_{OL3} (Section 20.0.)

15.2.16 P1.2 Configuration

Table 15-13. 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 = This CLK Output is used to observe connected external crystal oscillator clock connected in CY7C601xx. When CLK Output is set, the internally selected clock is sent out onto P1.2 pin.

15.2.17 P1.3 Configuration (SSEL)

Table 15-14. P1.3 Configuration (P13CR) [0x10] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	Reserved	High Sink	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 the P1.3 pin.
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, 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 CY7C602xx. In the CY7C601xx, only 8-mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.

15.2.18 P1.4–P1.6 Configuration (SCLK, SMOSI, SMISO)

Table 15-15. 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	Reserved	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

These registers control the operation of pins P1.4–P1.6, respectively.
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, 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 CY7C602xx. In the CY7C601xx, only 8-mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.
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 16-2*):
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.

15.2.19 P1.7 Configuration

Table 15-16. P1.7 Configuration (P17CR) [0x14] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	Int Enable	Int Act Low	Reserved	High Sink	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 P1.7.
The 50-mA sink drive capability is only available in the CY7C602xx. In the CY7C601xx, only 8-mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.
The P1.7 GPIO's threshold is always set to TTL.

15.2.20 P2 Configuration

Table 15-17. 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

In CY7C602xx this register controls the operation of pins P2.0–P2.1. In the CY7C601xx, this register controls the operation of pins P2.0–P2.7.
The 50-mA sink drive capability is only available on pin P2.7 and only on the CY7C601xx. In the CY7C602xx, only 8-mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.

15.2.21 P3 Configuration

Table 15-18. 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	0	0

In CY7C602xx this register controls the operation of pins P3.0–P3.1. In the CY7C601xx, this register controls the operation of pins P3.0–P3.7.
The 50-mA sink drive capability is only available on pin P3.7 and only on the CY7C601xx. In the CY7C602xx, only 8-mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.

15.2.22 P4 Configuration

Table 15-19. P4 Configuration (P4CR) [0x17] [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 exists only in the CY7C601xx. This register controls the operation of pins P4.0–P4.3.

16.0 Serial Peripheral Interface (SPI)

The SPI Master/Slave Interface core logic runs on the SPI clock domain. The SPI clock is a divider off of the CPUCLK when in Master Mode. SPI is a four-pin serial interface comprised of a clock, an enable, and two data pins.

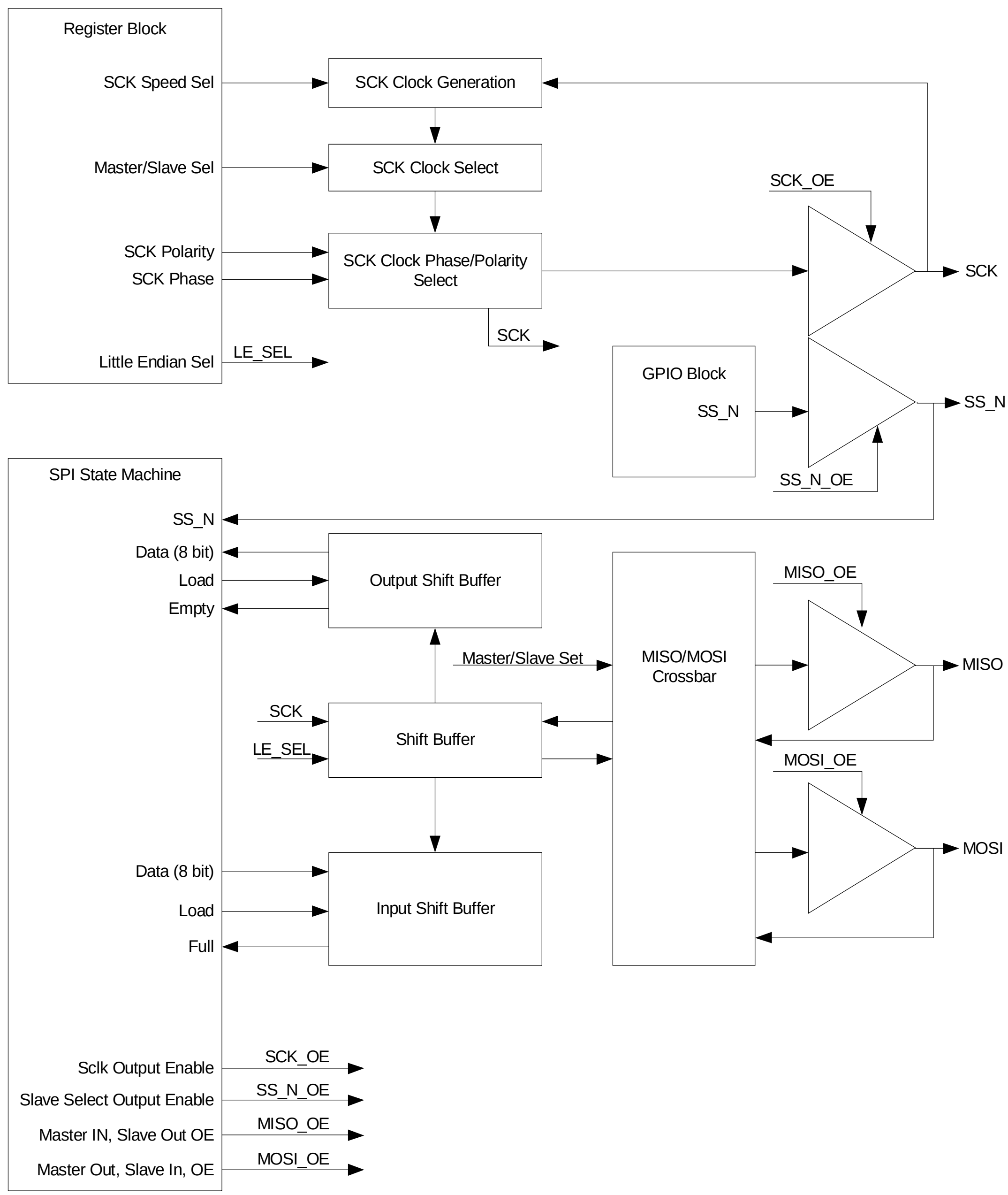


Figure 16-1. SPI Block Diagram

16.1 SPI Data Register

Table 16-1. 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 an 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.

16.2 SPI Configure Register

Table 16-2. 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 16-3* 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 15-15*), 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 16-3. SPI Mode Timing vs. LSB First, CPOL and CPHA

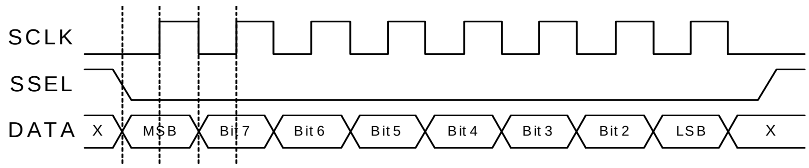
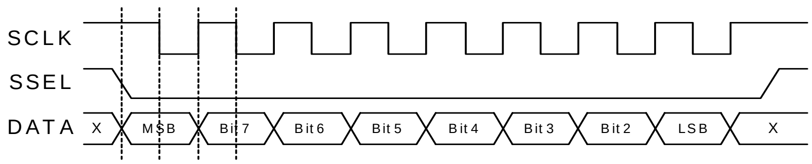
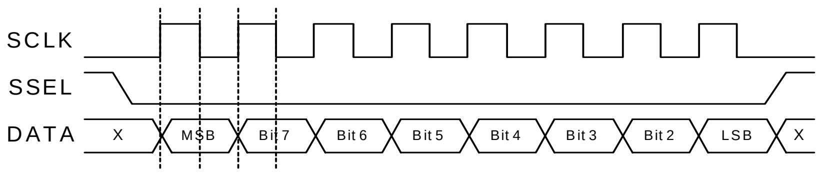
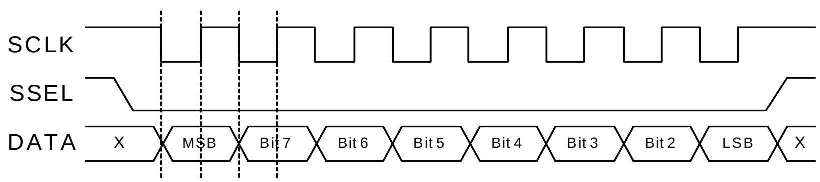
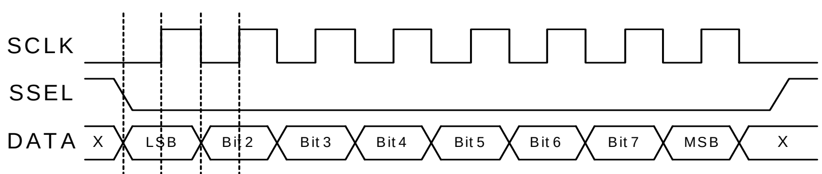
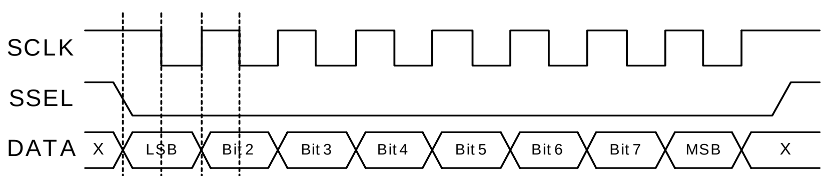
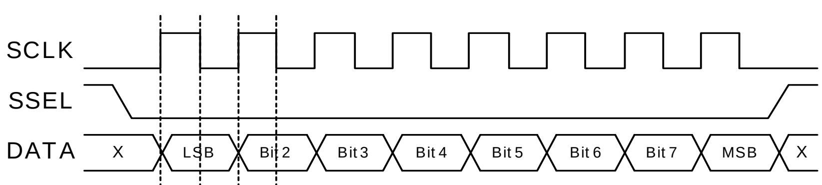
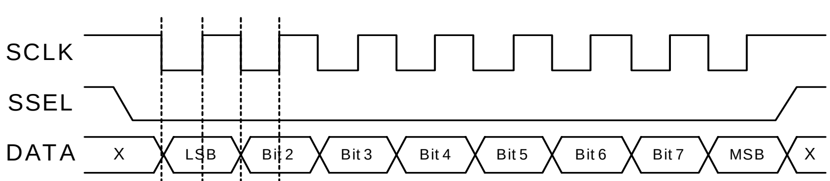
LSB First	CPHA	CPOL	Diagram
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Table 16-4. SPI SCLK Frequency

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

16.3 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.

17.0 Timer Registers

All timer functions of the Wireless enCoRe II are provided by a single timer block. The timer block is asynchronous from the CPU clock. The 16-bit free-running counter is used as the time-base for timer captures and can also be used as a general time-base by software.

17.1 Registers

17.1.1 Free-running Counter

The 16 bit free-running counter is clocked by a 4-/6-MHz source. 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. It can also generate an interrupt when the free-running counter overflow occurs—every 16.384 ms. This allows extending the length of the timer in software.

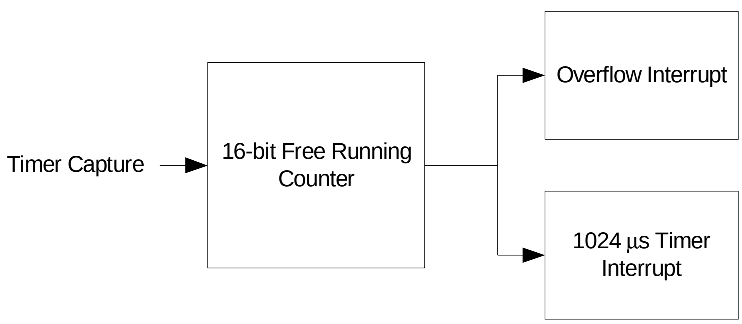


Figure 17-1. 16-bit Free Running Counter Block Diagram

Table 17-1. 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 should be read first and the high-order second. When writing, the low-order byte should be written first then the high-order byte.

Table 17-2. 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 should be read first and the high-order second. When writing, the low-order byte should be written first then the high-order byte.

17.1.2 Time Capture

Wireless enCoRe II has two eight bit captures. Each capture has separate register 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 resisters hold the high order byte of the 16-bit timer capture value. Each of the four capture registers can be programmed to generate an interrupt when it is loaded.

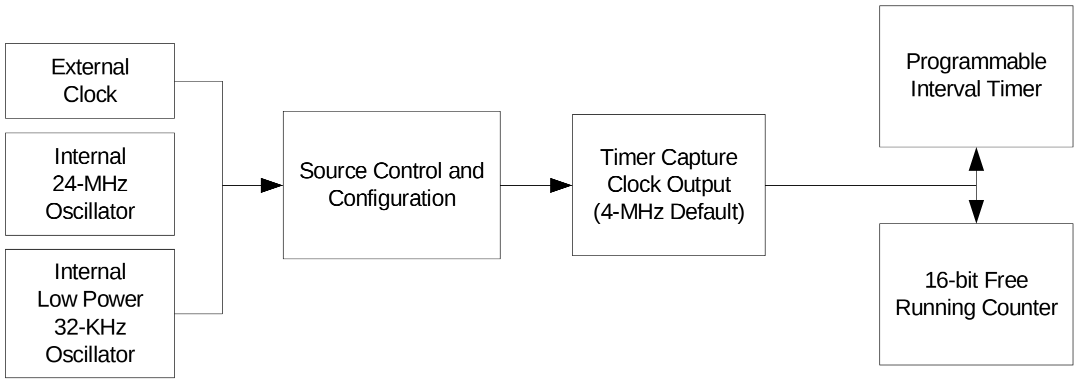


Figure 17-2. Time Capture Block Diagram

Table 17-3. 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 16-bit 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 17-4. 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 17-5. 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	R	R	R	R	R	R	R
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 17-6. 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	R	R	R	R	R	R	R
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 the high-order 8 bits of the 16-bit timer on TCAP0 rising edge.

Table 17-7. 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	R	R	R	R	R	R	R
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 17-8. 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	R	R	R	R	R	R	R
Default	0	0	0	0	0	0	0	0

Bit [7:0]: Capture 1 Falling [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 17-9. 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

These four bits contains the status bits for the four timer captures for the four timer block capture interrupt sources. Writing any of these bits with 1 clears that interrupt.
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

17.1.3 Programmable Interval Timer

Table 17-10. Programmable Interval Timer Low (PITMRL) [0x26] [R/W]

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 17-11. Programmable Interval Timer High (PITMRH) [0x27] [R/W]

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 17-12. 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 17-13. 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.

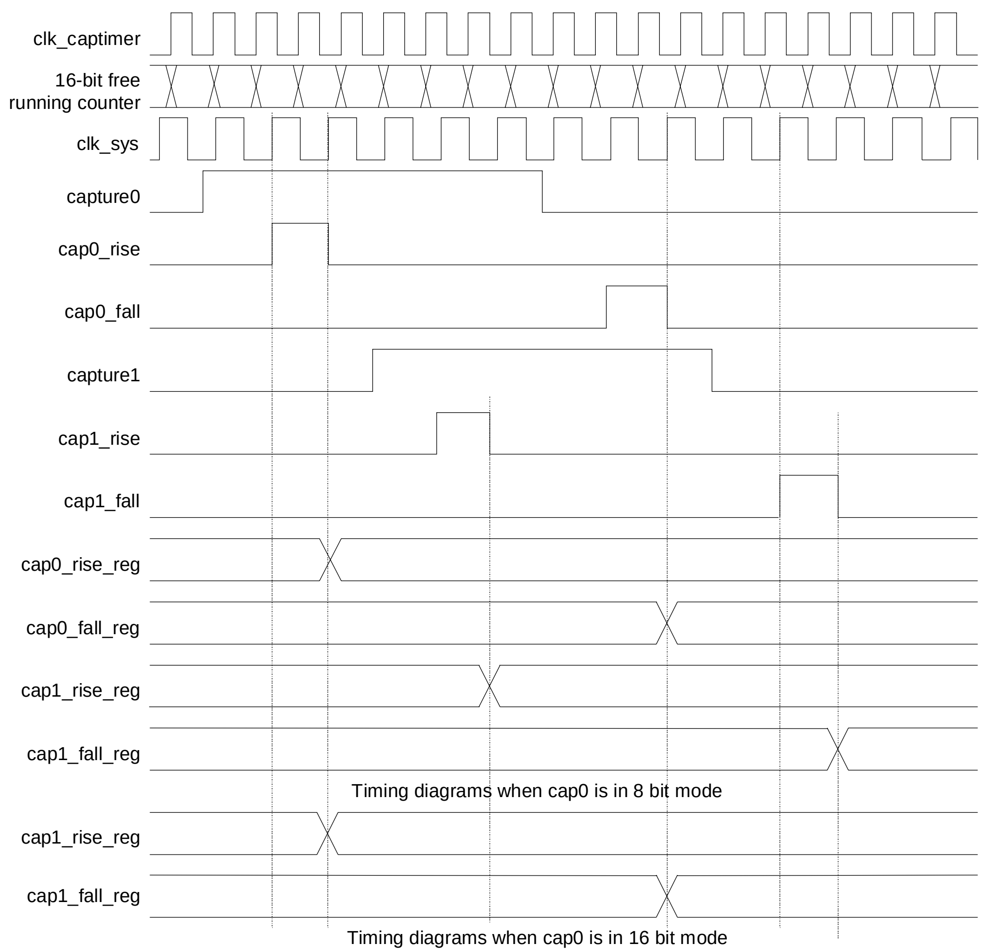


Figure 17-3. Timer Functional Timing Diagram

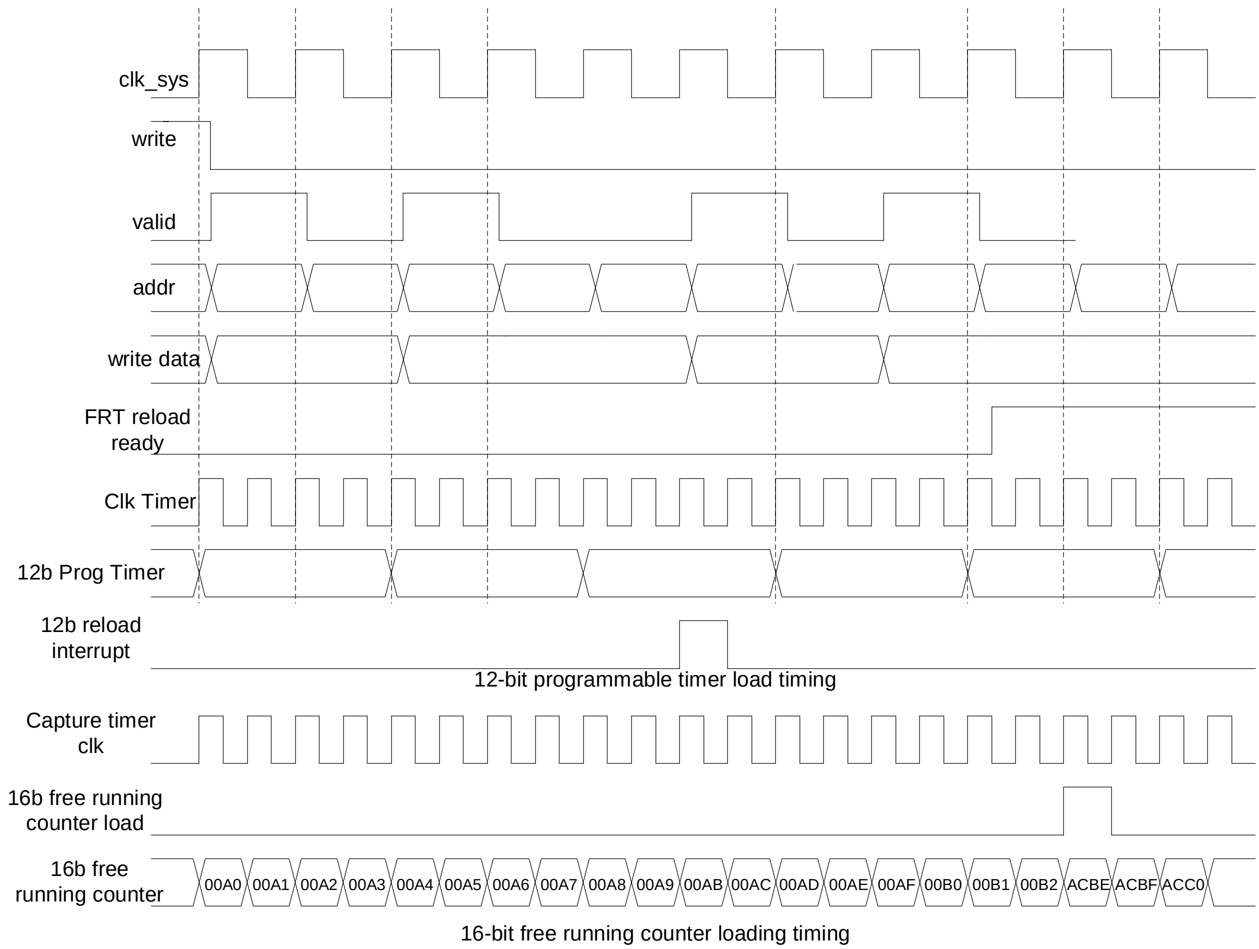


Figure 17-4. 16-bit Free Running Counter Loading Timing Diagram

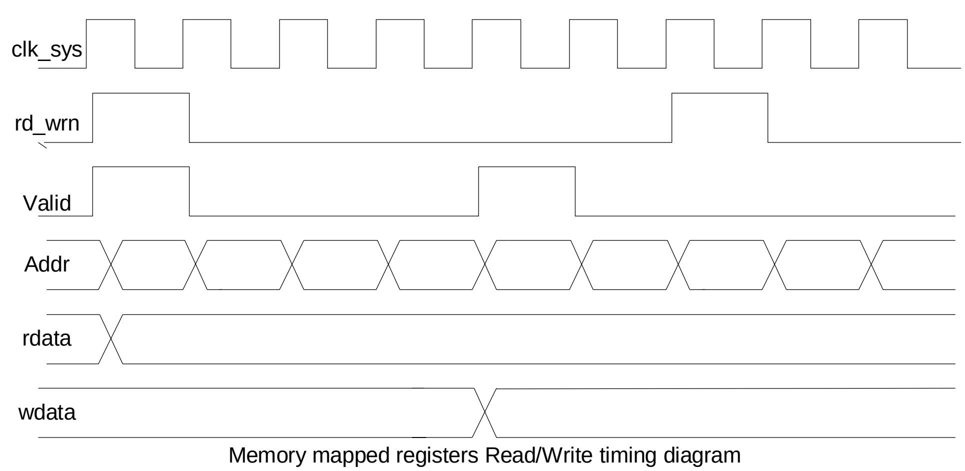


Figure 17-5. Memory Mapped Registers Read/Write Timing Diagram

18.0 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 Wireless enCoRe II devices. The registers associated with the interrupt controller allow 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 Wireless enCoRe II devices.

Table 18-1. Interrupt Priorities, Address, Name

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	Reserved
9	0024h	Reserved
10	0028h	Reserved
11	002Ch	Reserved
12	0030h	Reserved
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	Reserved
20	0050h	GPIO Port 2
21	0054h	GPIO Port 3
22	0058h	GPIO Port 4
23	005Ch	Reserved
24	0060h	Reserved
25	0064h	Sleep Timer

18.1 Architectural Description

An interrupt is posted when its interrupt conditions occur. This results in the flip-flop in *Figure 18-1* 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 reenabling interrupts inside an interrupt service routine. To do this, set the IE bit in the Flag Register. A block diagram of the Wireless enCoRe II Interrupt Controller is shown in *Figure 18-1*.

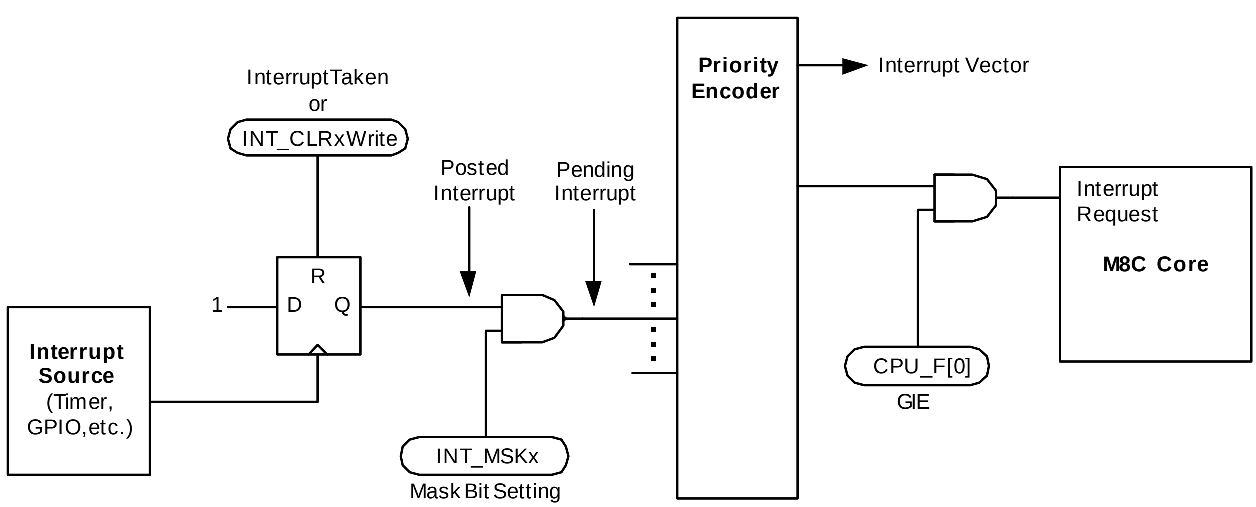


Figure 18-1. Interrupt Controller Block Diagram

18.2 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:
   a. The 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.
   b. 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.
   c. The CPU_F register is then cleared. Since this clears the GIE bit to 0, additional interrupts are temporarily disabled.
   d. The PCH (PC[15:8]) is cleared to zero.
   e. 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 interrup0t.
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.

18.3 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 12 MHz, 25 clock cycles take 2.08 µs.

18.4 Interrupt Registers

18.4.1 Interrupt Clear Register

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 18-2. 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 18-3. Interrupt Clear 1 (INT_CLR1) [0xDB] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	TCAP0	Prog Interval Timer	1-ms Programmable Interrupt	Reserved				
Read/Write	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 18-4. Interrupt Clear 2 (INT_CLR2) [0xDC] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	GPIO Port 4	GPIO Port 3	GPIO Port 2	Reserved	INT2	16-bit Counter Wrap	TCAP1
Read/Write	–	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.

18.4.2 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 0's written to the INT_CLRx registers are ignored. However, 1's 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 18-5. Interrupt Mask 3 (INT_MSK3) [0xDE] [R/W]

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

Bit 7: Enable Software Interrupt (ENSWINT)
0= Disable. Writing 0's to an INT_CLRx register, when ENSWINT is cleared, will cause the corresponding interrupt to clear.
1= Enable. Writing 1's to an INT_CLRx register, when ENSWINT is set, will cause the corresponding interrupt to post.
Bit [6:0]: Reserved

Table 18-6. Interrupt Mask 2 (INT_MSK2) [0xDF] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Reserved	GPIO Port 4 Int Enable	GPIO Port 3 Int Enable	GPIO Port 2 Int Enable	Reserved	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
Default	0	0	0	0	0	0	0	0

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: Reserved
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 18-7. 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	Reserved				
Read/Write	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 [4:0]: Reserved

Table 18-8. 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

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 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

18.4.3 Interrupt Vector Clear Register

Table 18-9. 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.

19.0 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

20.0 DC Characteristics

Parameter	Description General	Conditions	Min.	Typical	Max.	Unit
V_{CC1}	Operating Voltage	CPU speed <= 12 MHz	2.7		3.6	V
T_{FP}	Operating Temp	Flash Programming	0		70	°C
I_{CC1}	V_{CC} Operating Supply Current	CPU speed = 12 MHz			11	mA
I_{CC2}	V_{CC} Operating Supply Current	CPU speed = 6 MHz			9	mA
I_{SB1}	Standby Current	Internal and External Oscillators, Bandgap, Flash, CPU Clock, Timer Clock all disabled			10	µA
Low-voltage Detect						
V_{LVD}	Low-voltage detect Trip Voltage	LVDCR [2:0] set to 000	2.681		2.7	V
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				0.72	V
V_{IHTTL}	Input HIGH Voltage, TTL Mode		1.6			V
V_{OL1}	Output Low Voltage, High Drive[4]	I_{OL1} = 50 mA			1.4	V
V_{OL2}	Output Low Voltage, High Drive[4]	I_{OL1} = 25 mA			0.4	V
V_{OL3}	Output Low Voltage, Low Drive	I_{OL2} = 8 mA			0.8	V
V_{OH}	Output High Voltage[4]	I_{OH} = 2 mA	V_{CC} – 0.5			V

21.0 AC Characteristics

Parameter	Description	Conditions	Min.	Typical	Max.	Unit
Clock						
T_{ECLKDC}	External Clock Duty Cycle		45		55	%
T_{ECLK2}	External Clock Frequency		1		24	MHz
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

Notes:

4. Available only on CY7C601xx P2.7, P3.7, P0.0, P0.1; CY7C602xx P1.3,P1.4,P1.5,P1.6,P1.7.
5. In Master mode first bit is available 0.5 SPICLK cycle before Master clock edge available on the SCLK pin.

21.0 AC Characteristics (continued)

Parameter	Description	Conditions	Min.	Typical	Max.	Unit
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 $\overline{SS}$ 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

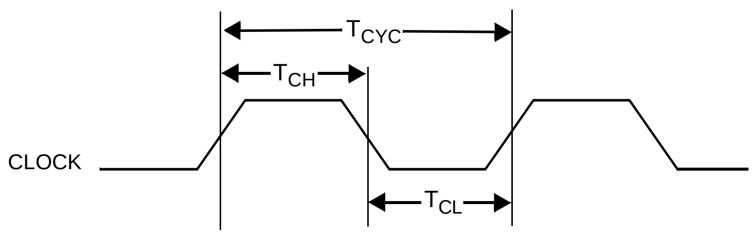


Figure 21-1. Clock Timing

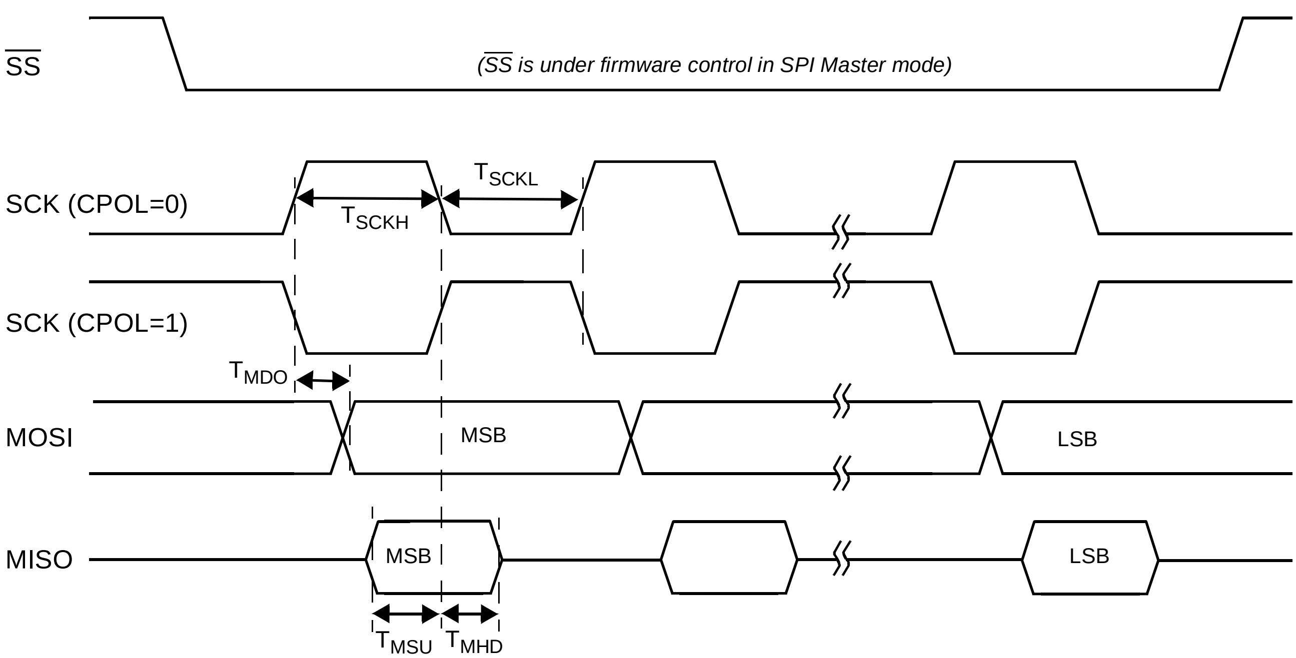


Figure 21-2. SPI Master Timing, CPHA = 1

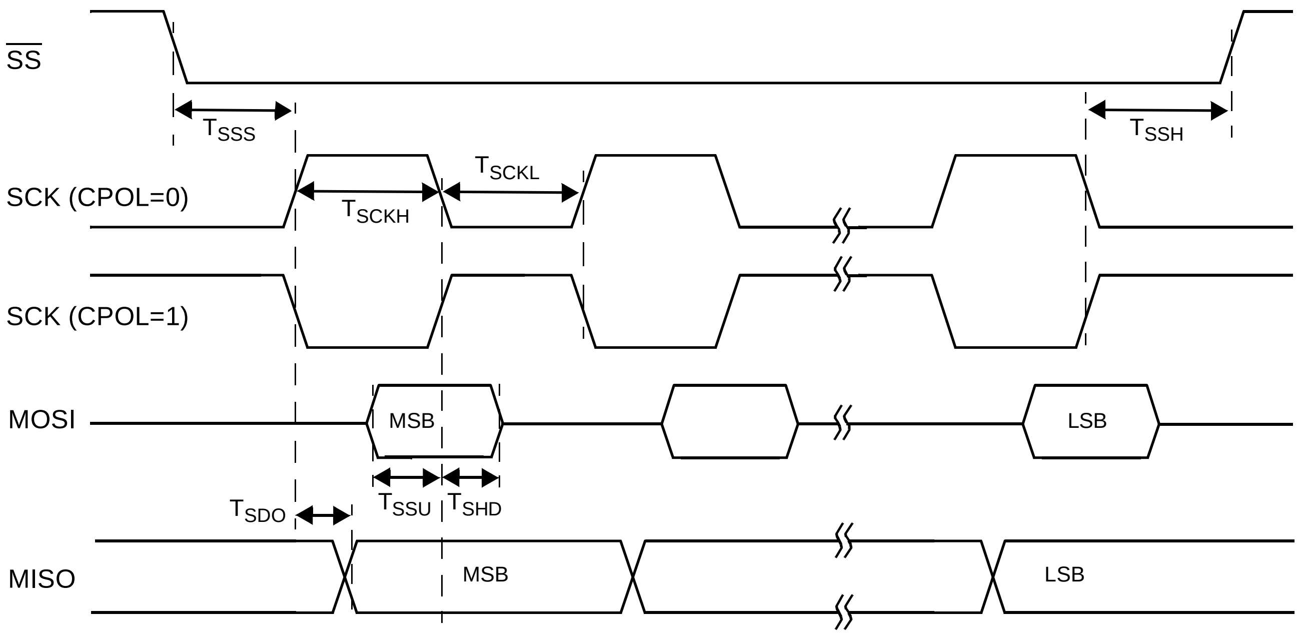


Figure 21-3. SPI Slave Timing, CPHA = 1

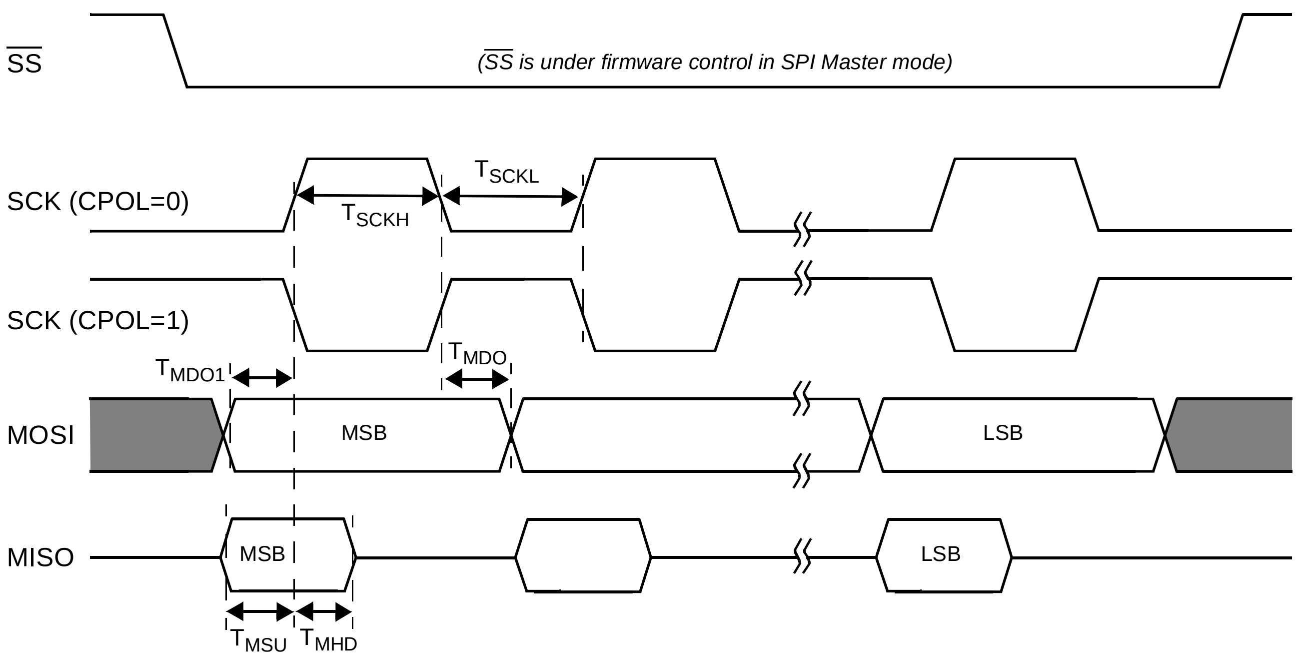


Figure 21-4. SPI Master Timing, CPHA = 0

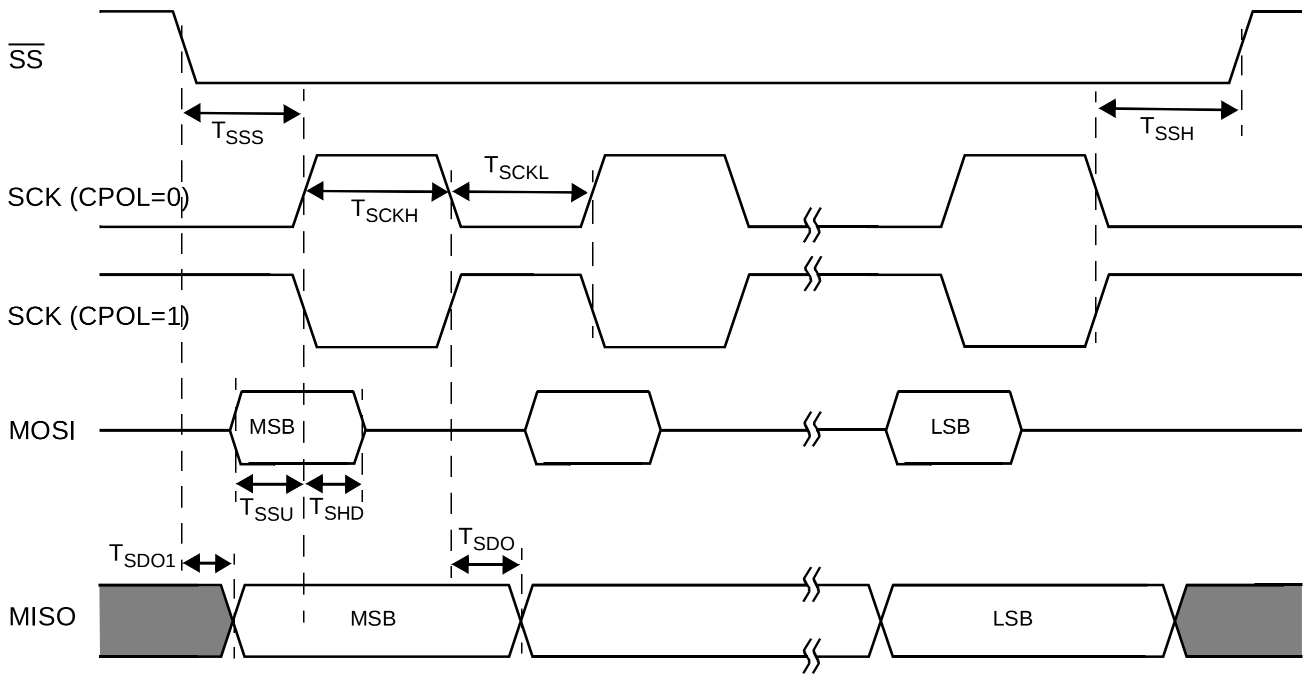


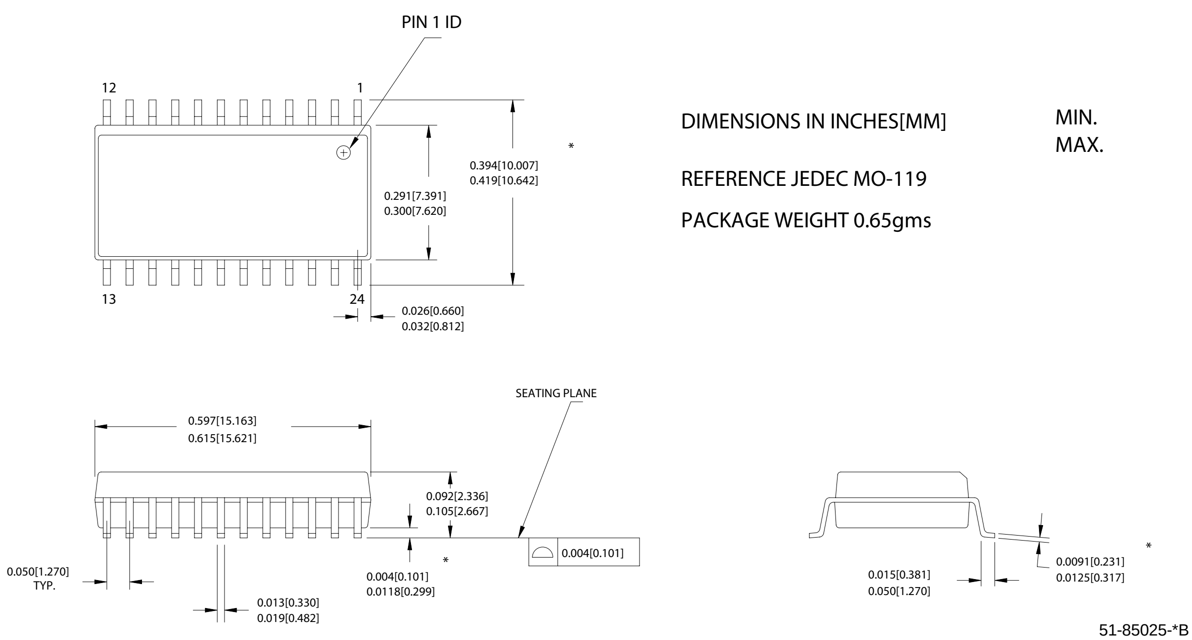
Figure 21-5. SPI Slave Timing, CPHA = 0

22.0 Ordering Information

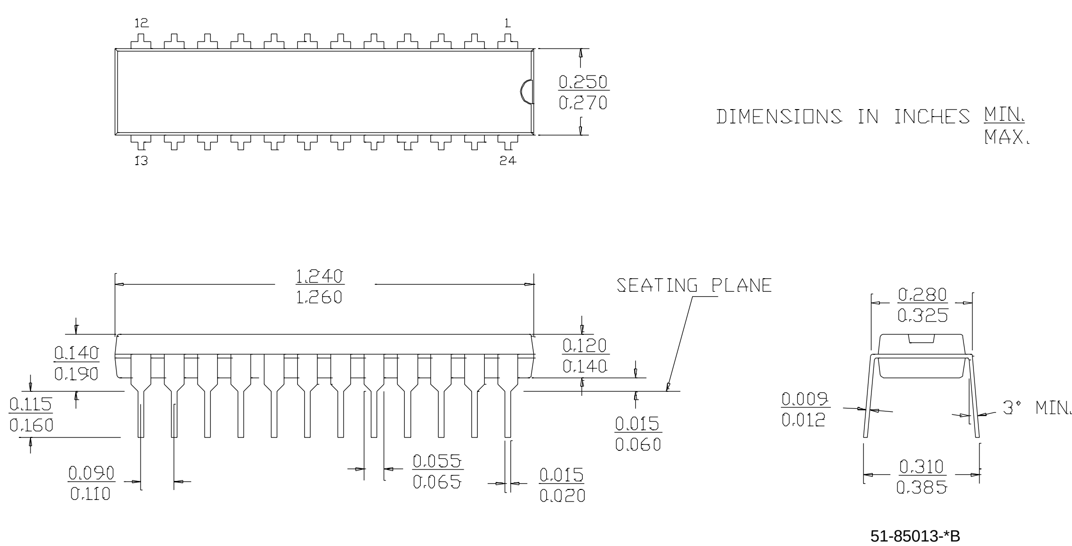
Ordering Code	FLASH Size	RAM Size	Package Type
CY7C60123-PVXC	8K	256	48-SSOP
CY7C60123-PXC	8K	256	40-PDIP
CY7C60113-PVXC	8K	256	28-SSOP
CY7C60223-PXC	8K	256	24-PDIP
CY7C60223-SXC	8K	256	24-SOIC
CY7C60223-QXC	8K	256	24-QSOP

23.0 Package Diagrams

24-Lead (300-Mil) SOIC S13

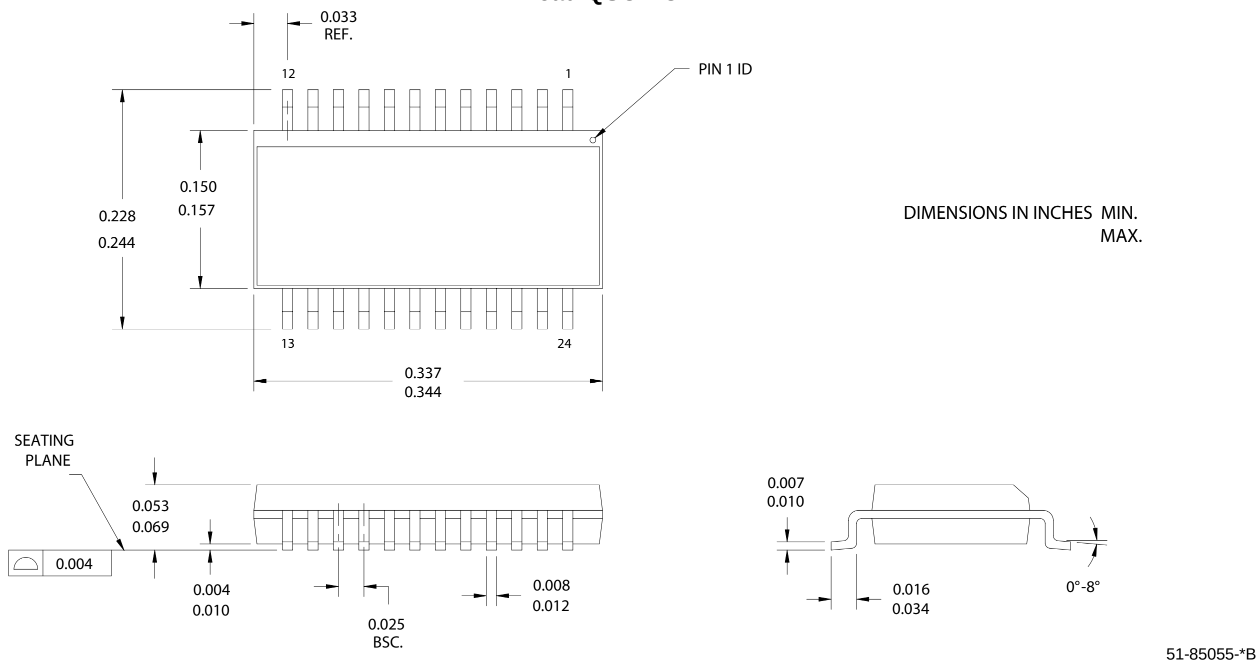


24-Lead (300-Mil) PDIP P13

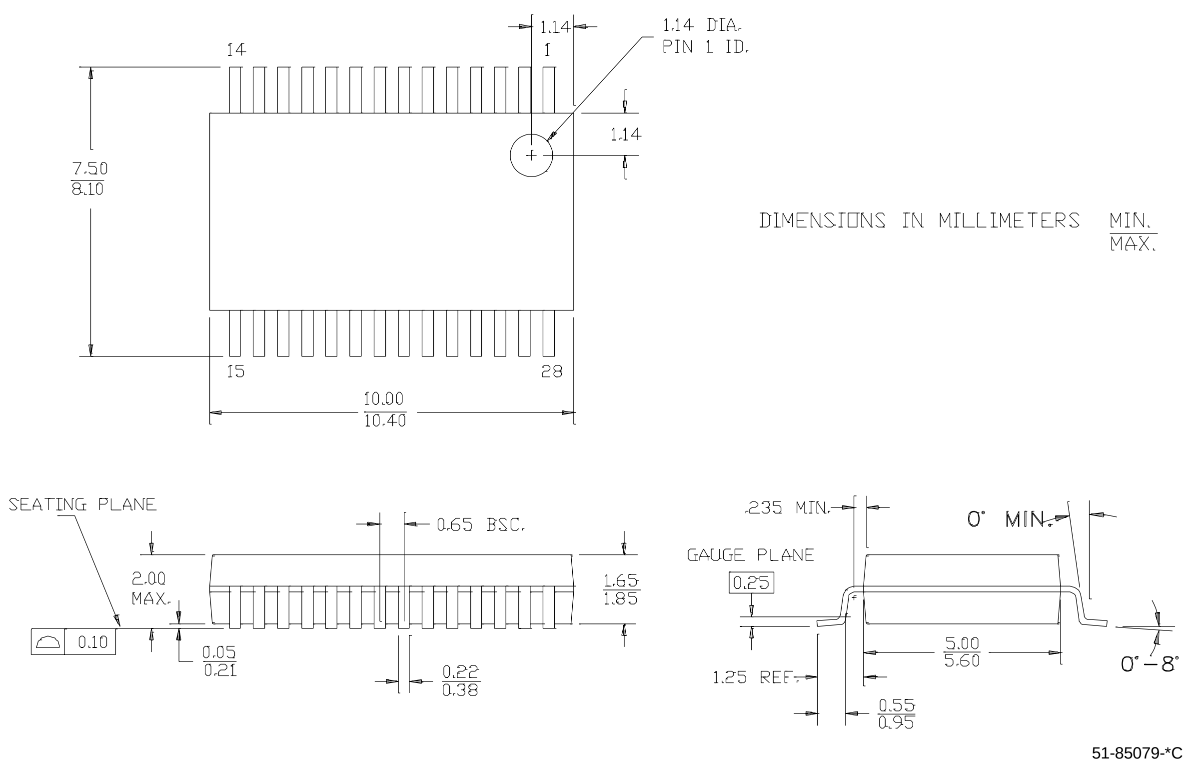


23.0 Package Diagrams (continued)

24-lead QSOP O241

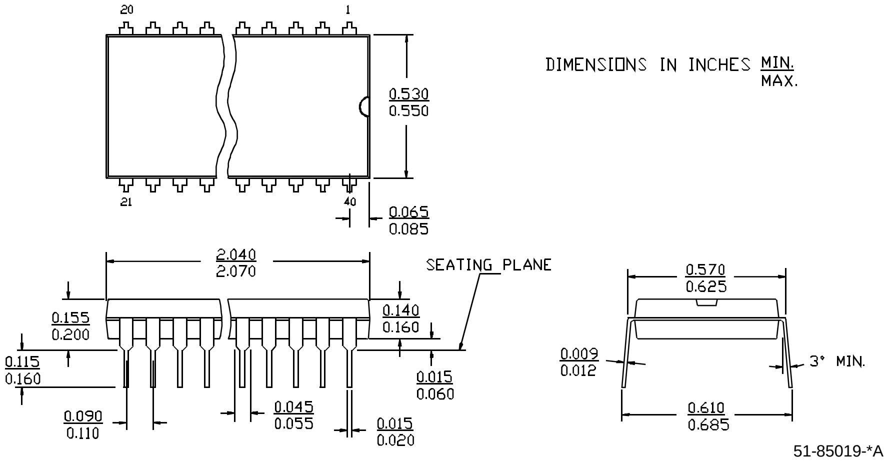


28-Lead (5.3 mm) Shrunk Small Outline Package O28



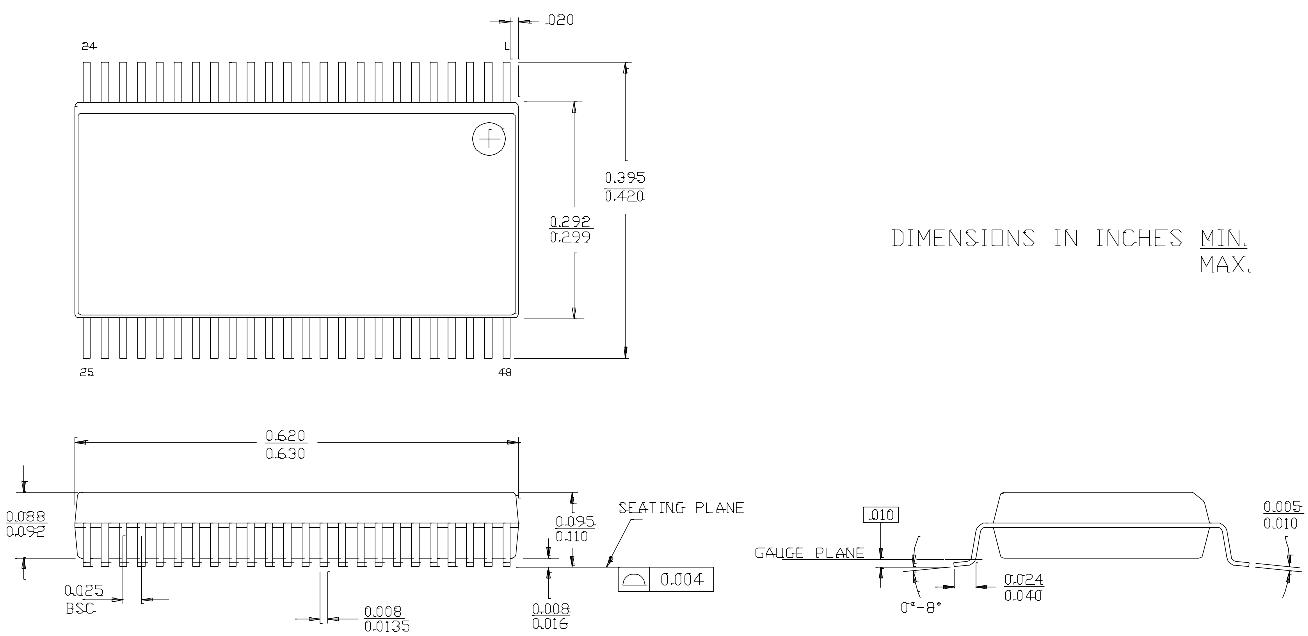
23.0 Package Diagrams (continued)

40-Lead (600-Mil) Molded DIP P17



51-85019-*A

48-Lead Shrunk Small Outline Package O48



51-85061-*C

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.

Document History Page

Document Title: CY7C601/2xx Wireless enCoRe™ II Microcontroller Document Number: 38-16016				
Rev.	**ECN No.**	**Issue Date**	**Orig. of Change**	**Description of Change**
**	327601	See ECN	BON	New data sheet
*A	400134	See ECN	BHA	Updated Power consumption values. Corrected Pin Assignment Table for 24 QSOP, 24 PDIP and 28 SSOP packages. Minor text changes for clarification purposes. Corrected INT_MSK0 and INT_MSK1 register address. Corrected register bit definitions. Corrected Protection Mode Settings in Table 10-7. Updated LVD Trip Point values. Added Block diagrams for Timer functional timing. Replaced TBD's with actual values. Added SPI Block Diagram. Added Timing Block Diagrams. Removed CY7C60123 DIE from Figure 5-1. Removed CY7C60123-WXC from Section 22.0 Ordering Information. Updated internal 24MHz Oscillator accuracy information. Added information on sending/receiving data when using 32Khz oscillator.