Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

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

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	M8C
Core Size	8-Bit
Speed	12MHz
Connectivity	SPI
Peripherals	LVD, POR, WDT
Number of I/O	20
Program Memory Size	8KB (8K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	-
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	24-SSOP (0.154", 3.90mm Width)
Supplier Device Package	24-QSOP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c60223-qxc

8. Register Summary

Table 8-1. enCoRe II LV Register Summary

The XIO bit in the CPU flags register must be set to access the extended register space for all registers above 0xFF.

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 16-bit 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

10.1.1 Accumulator Register

Table 10-2. 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 or arithmetic instruction that uses a source addressing mode.

10.1.2 Index Register

Table 10-3. 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.

10.1.3 Stack Pointer Register

Table 10-4. 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.

10.1.4 CPU Program Counter High Register

Table 10-5. 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.

10.1.5 CPU Program Counter Low Register

Table 10-6. 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.

10.2.5 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 10-11. 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.
```

10.2.6 Destination Direct Source Immediate

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

Table 10-12. Destination Direct Source Immediate

Opcode	Operand 1	Operand 2
Instruction	Destination address	Immediate value

Examples

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

10.2.7 Destination Indexed Source Immediate

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

Table 10-13. Destination Indexed Source Immediate

Opcode	Operand 1	Operand 2
Instruction	Destination index	Immediate value

Examples

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

10.2.8 Destination Direct Source Direct

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

Table 10-14. Destination Direct Source Direct

Opcode	Operand 1	Operand 2
Instruction	Destination address	Source address

Example

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

10.2.9 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 10-15. Source Indirect Post Increment

Opcode	Operand 1
Instruction	Source 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 Indirect
                 address is moved into the accumulator. The
                 indirect address is then incremented.
```

10.2.10 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 10-16. Destination Indirect Post Increment

Opcode	Operand 1
Instruction	Destination address

Example

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

11. Instruction Set Summary

The instruction set is summarized in Table 11-1 numerically and serves as a quick reference. The instruction set summary tables are described in detail in the *PSoC Designer: Assembly Language User Guide*.

Table 11-1. Instruction Set Summary Sorted Numerically by Opcode Order

Opcode Hex	Cycles	Bytes	Instruction Format[1, 2]	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

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.

12.6 SROM Table Read Description

The silicon IDs for enCoRe II LV devices are stored in the SROM tables in the part, as shown in Figure 12-3 on page 21.

The silicon ID can be read out from the part using SROM table reads. This is demonstrated in the following pseudo code. As mentioned in the section SROM on page 16, the SROM variables occupy address F8h through FFh in the SRAM. Each of the variables and their definitions are given in the section SROM on page 16.

```
AREA SSCParmBlkA(RAM,ABS)

    org  F8h // Variables are defined starting at address F8h

SSC_KEY1:                  ; F8h  supervisory key
SSC_RETURNCODE:     blk 1 ; F8h  result code
SSC_KEY2 :          blk 1  ;F9h  supervisory stack ptr key
SSC_BLOCKID:        blk 1 ; FAh  block ID
SSC_POINTER:        blk 1 ; FBh  pointer to data buffer
SSC_CLOCK:          blk 1 ; FCh  Clock
SSC_MODE:           blk 1 ; FDh  ClockW ClockE multiplier
SSC_DELAY:          blk 1 ; FEh  flash macro sequence delay count
SSC_WRITE_ResultCode: blk 1 ; FFh  temporary result code

_main:
          mov   A, 2
          mov   [SSC_BLOCKID], A// To read from Table 2 - trim values for the IMO are stored in table 2
          mov   X, SP      ; copy SP into X
          mov   A, X       ; A temp stored in X
          add   A, 3               ; create 3 byte stack frame (2 + pushed A)
          mov   [SSC_KEY2], A      ; save stack frame for supervisory code

   ; load the supervisory code for flash operations
          mov   [SSC_KEY1], 3Ah  ;FLASH_OPER_KEY - 3Ah

          mov   A,6        ; load A with specific operation. 06h is the code for Table (read Table
12-1 on page 16)
          SSC              ; SSC call the supervisory ROM

// At the end of the SSC command the silicon ID is stored in F8 (MSB) and F9(LSB) of the SRAM

.terminate:
    jmp .terminate
```

Figure 12-3. SROM Table

	F8h	F9h	FAh	FBh	FCh	FDh	FEh	FFh
Table 0	Silicon ID [15-8]	Silicon ID [7-0]						
Table 1	Family / Die ID	Revision ID						
Table 2					24 MHz IOSCTR at 3.30V	24 MHz IOSCTR at 3.00V	24 MHz IOSCTR at 2.85V	24 MHz IOSCTR at 2.70V
Table 3	32 kHz LPOSCTR at 3.30V	32 kHz LPOSCTR at 3.00V	32 kHz LPOSCTR at 2.85V	32 kHz LPOSCTR at 2.70V				
Table 4								
Table 5								
Table 6								
Table 7								

12.6.1 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' calculates the checksum of only block 0, while a BLOCKID value of '0' calculates 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.

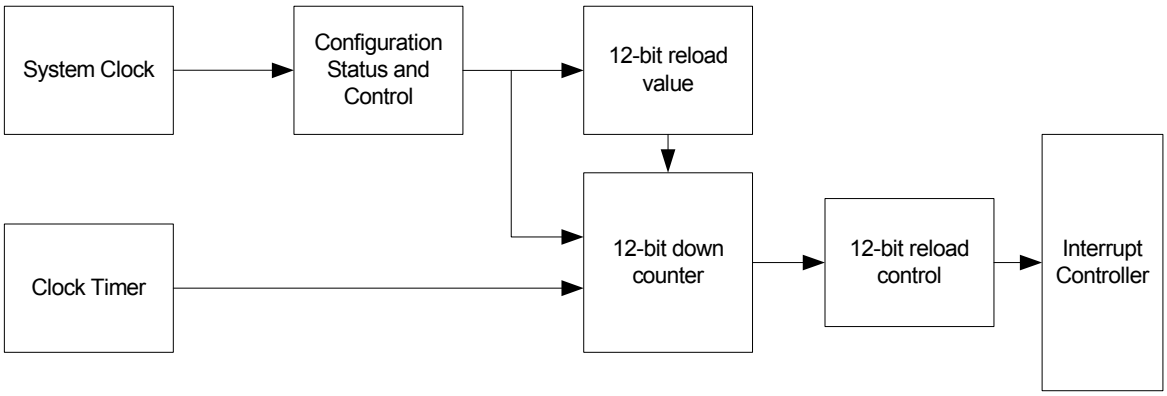
The checksum algorithm executes the following sequence of three instructions over the number of blocks times 64 to be checksummed.

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

Table 12-1. 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

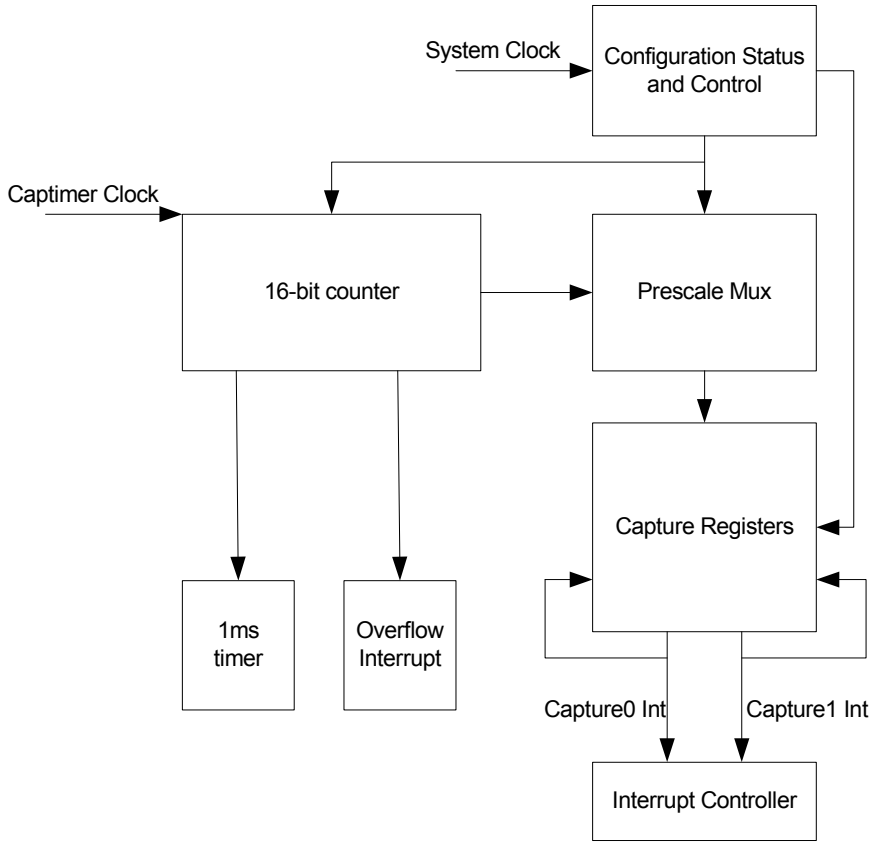
Figure 13-2. Programmable Interval Timer Block Diagram



13.2.3 Timer Capture Clock (TCAPCLK)

The TCAPCLK is sourced from the external crystal oscillator, the 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 13-3. Timer Capture Block Diagram

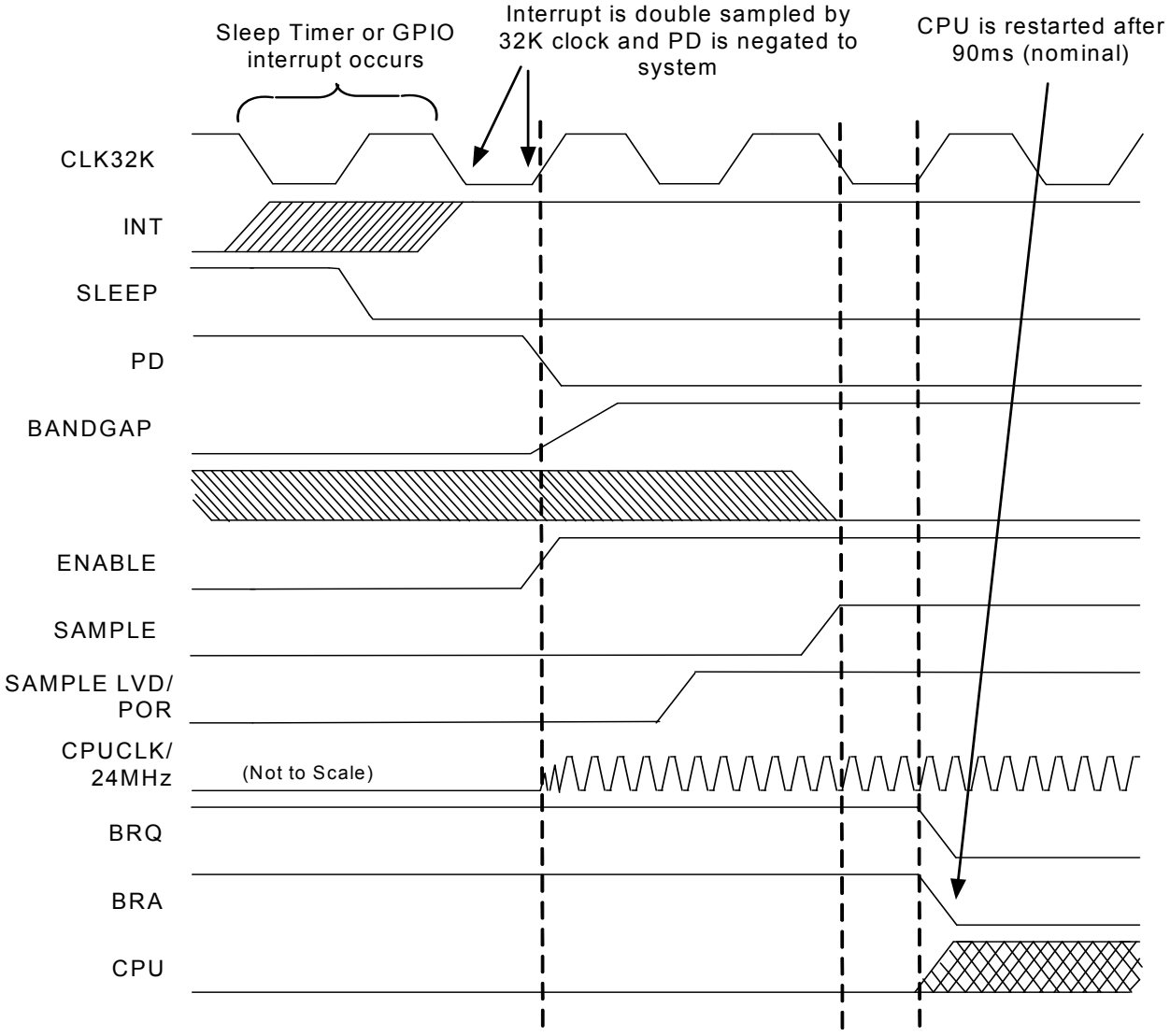


15.2 Wakeup Sequence

When asleep, the only event that wakes the system up is an interrupt. The global interrupt enable of the CPU flag register need not be set. Any unmasked interrupt wakes the system up. It is optional for the CPU to actually take the interrupt after the wakeup sequence. The wakeup sequence is synchronized to the 32-kHz clock. This is done to sequence a startup delay and allow the flash memory module enough time to power up before the CPU asserts the first read access. Another reason for the delay is to enable the oscillator, bandgap, and LVD and POR circuits time to settle before actually being used in the system. As shown in Figure 15-2, the wakeup sequence is as follows:

1. The wakeup interrupt occurs and is synchronized by the negative edge of the 32-kHz clock.
2. At the following positive edge of the 32-kHz clock, the system wide PD signal is negated. The flash memory module, internal oscillator, EFTB, and bandgap circuit are all powered up to a normal operating state.
3. At the following positive edge of the 32-kHz clock, the current values for the precision POR and LVD have settled and are sampled.
4. At the following negative edge of the 32-kHz clock (after about 15 µs nominal), the BRQ signal is negated by the sleep logic circuit. On the following CPUCLK, BRA is negated by the CPU and instruction execution resumes. Note that in Figure 15-2 fixed function blocks, such as flash, internal oscillator, EFTB, and bandgap, have about 15 µs start-up. The wakeup times (interrupt to CPU operational) range from 75 µs to 105 µs.

Figure 15-2. Wakeup Timing



17.1.3 P2 Data

Table 17-3. 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]

17.1.4 P3 Data

Table 17-4. 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]

17.1.5 P4 Data

Table 17-5. 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.

17.2 GPIO Port Configuration

All GPIO configuration registers have common configuration controls. By default all GPIOs are configured as inputs. To prevent the inputs from floating, pull-up resistors are enabled. Firmware configures each of the GPIOs before use. The following are bit definitions of the GPIO configuration registers.

17.2.1 Int Enable

When set, the Int Enable bit allows the GPIO to generate interrupts. Interrupt generate occurs regardless of whether the pin is configured for input or output. All interrupts are edge-sensitive. However, for interrupts that are shared by multiple sources (ports 2, 3, and 4), all inputs are deasserted before a new interrupt occurs.

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 generate the interrupt by driving the appropriate pin state. This is useful in test and may find value in applications too.

17.2.2 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 makes interrupts active on the rising edge. Int Act Low set makes interrupts active on the falling edge.

17.2.3 TTL Thresh

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

Note The GPIOs default to CMOS threshold. The user's firmware must configure the threshold to TTL mode if necessary.

17.2.4 High Sink

When set, the output sinks up to 50 mA.

When clear, the output sinks up to 8 mA.

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

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

17.2.5 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 it is clear, the pin is driven low.

When clear, the output is driven low or high.

17.2.6 Pull-up Enable

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

When clear, the pull-up is disabled.

17.2.7 Output Enable

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

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

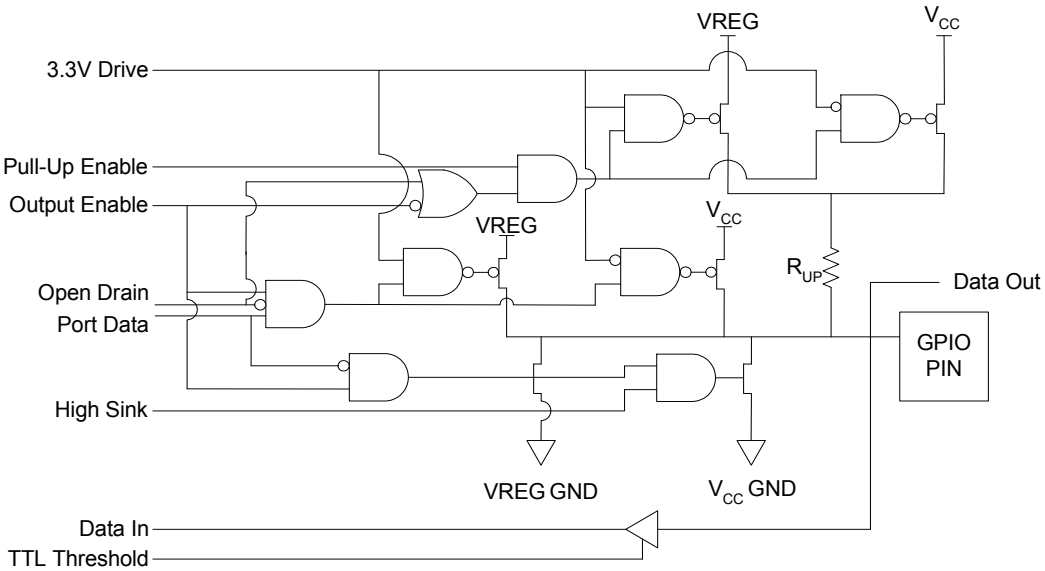
For pins with shared functions there are some special cases.

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

17.2.8 SPI Use

The P1.3(SSEL), P1.4(SCLK), P1.5(SMOSI), and P1.6(SMISO) pins are 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 17-1. GPIO Block Diagram



17.2.9 P0.0/CLKIN Configuration

Table 17-1. 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 13-3 on page 26), the GPIO function of the pin is disabled.

The 50-mA sink drive capability is only available in the CY7C601xx. In the CY7C602xx, only an 8-mA sink drive capability is available on this pin regardless of the setting of the high sink bit.

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

Table 17-4. 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, the firmware controls 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. The pulse is active for one cycle of the capture timer clock.

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 pulse is active for one cycle of the interval timer clock.

The P0.5/TIO0 and P0.6/TIO1 pins are individually configured with the P05CR (0x0A) and P06CR (0x0B), respectively.

17.2.13 P0.7 Configuration

Table 17-5. 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.

17.2.14 P1.0 Configuration

Table 17-6. 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} (see DC Characteristics on page 60).

Note There is no 2 mA sourcing capability on this pin. The pin can only sink 5 mA at V_{OL3} (see DC Characteristics on page 60)

The P1.0 is an open drain only output. It actively drives a signal low, but cannot actively drive a signal high.

If this pin is used as a general purpose output, it draws current. It is therefore configured as an input to reduce current draw.

17.2.21 P3 Configuration

Table 17-13. 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 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 CY7C601xx. In CY7C602xx, only an 8-mA sink drive capability is available on this pin regardless of the setting of the high sink bit.

17.2.22 P4 Configuration

Table 17-14. 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 CY7C601xx. This register controls the operation of pins P4.0–P4.3.

18.1 SPI Data Register

Table 18-1. SPI Data Register (SPIDATA) [0x3C] [R/W]

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

When read, this register returns the contents of the receive buffer. When written, it loads the transmit holding register.

Bit [7:0]: SPI Data [7:0]

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

18.2 SPI Configure Register

Table 18-2. 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 is useful to implement single wire communications similar to SPI.

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 18-3 on page 47 shows the timing for 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 17-10 on page 43), the input and output direction of pins P1.3, P1.5, and P1.6 is set automatically by the SPI logic. However, pin P1.4's input and output direction is NOT automatically set; it must be explicitly set by firmware. For SPI Master mode, pin P1.4 is configured as an output; for SPI Slave mode, pin P1.4 is configured as an input.

Table 18-3. SPI Mode Timing vs. LSB First, CPOL, and CPHA

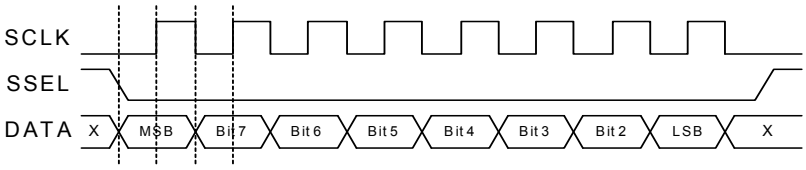
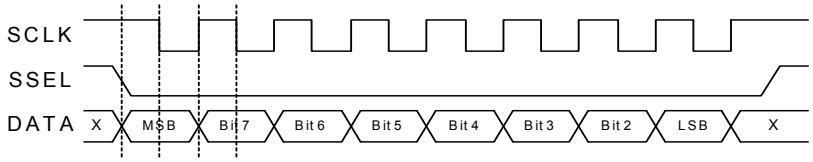
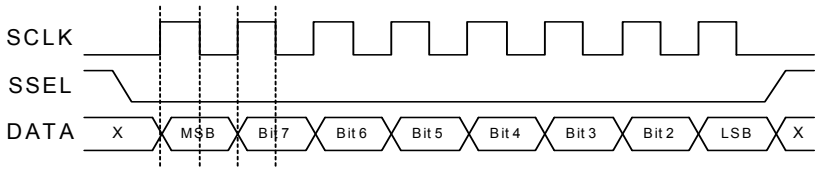
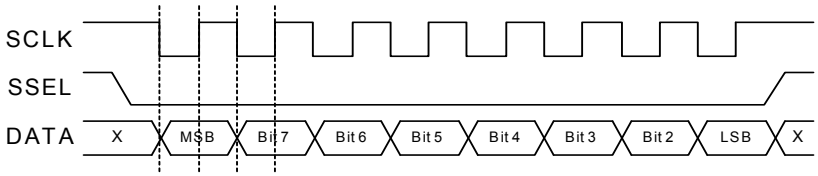
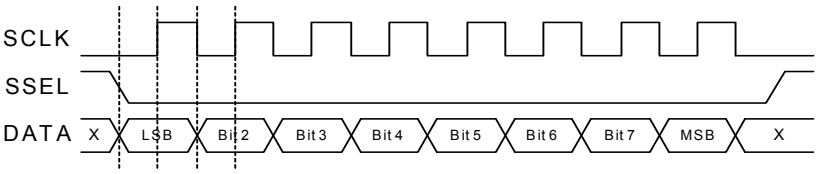
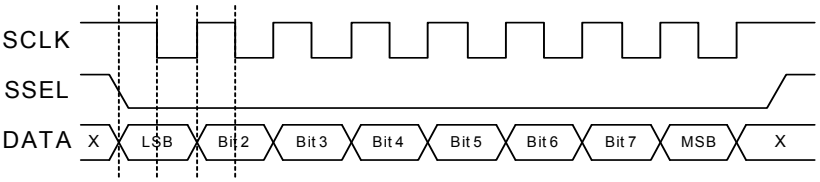
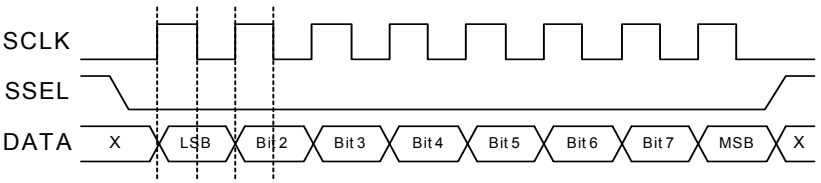
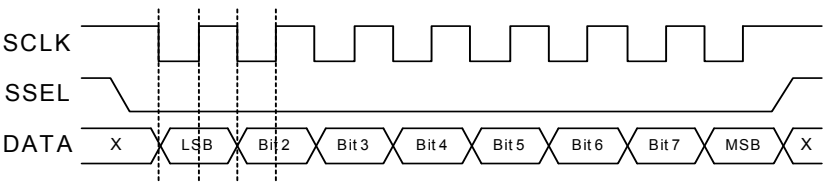
LSB First	CPHA	CPOL	Diagram
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Table 19-6. Timer Capture 1 Falling (TCAP1F) [0x25] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Capture 1 Falling [7:0]							
Read/Write	R	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 TIO1 input. The bits 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 eight bits of the 16-bit timer from the last TIO0 falling edge.

Table 19-7. 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 TIO1

Bit 2: Cap1 rise active

0 = No event

1 = A rising edge has occurred on TIO1

Bit 1: Cap0 Fall Active

0 = No event

1 = A falling edge has occurred on TIO0

Bit 0: Cap0 Rise Active

0 = No event

1 = A rising edge has occurred on TIO0

Note The interrupt status bits are cleared by firmware to enable subsequent interrupts. This is achieved by writing a ‘1’ to the corresponding Interrupt status bit.

19.1.3 Programmable Interval Timer

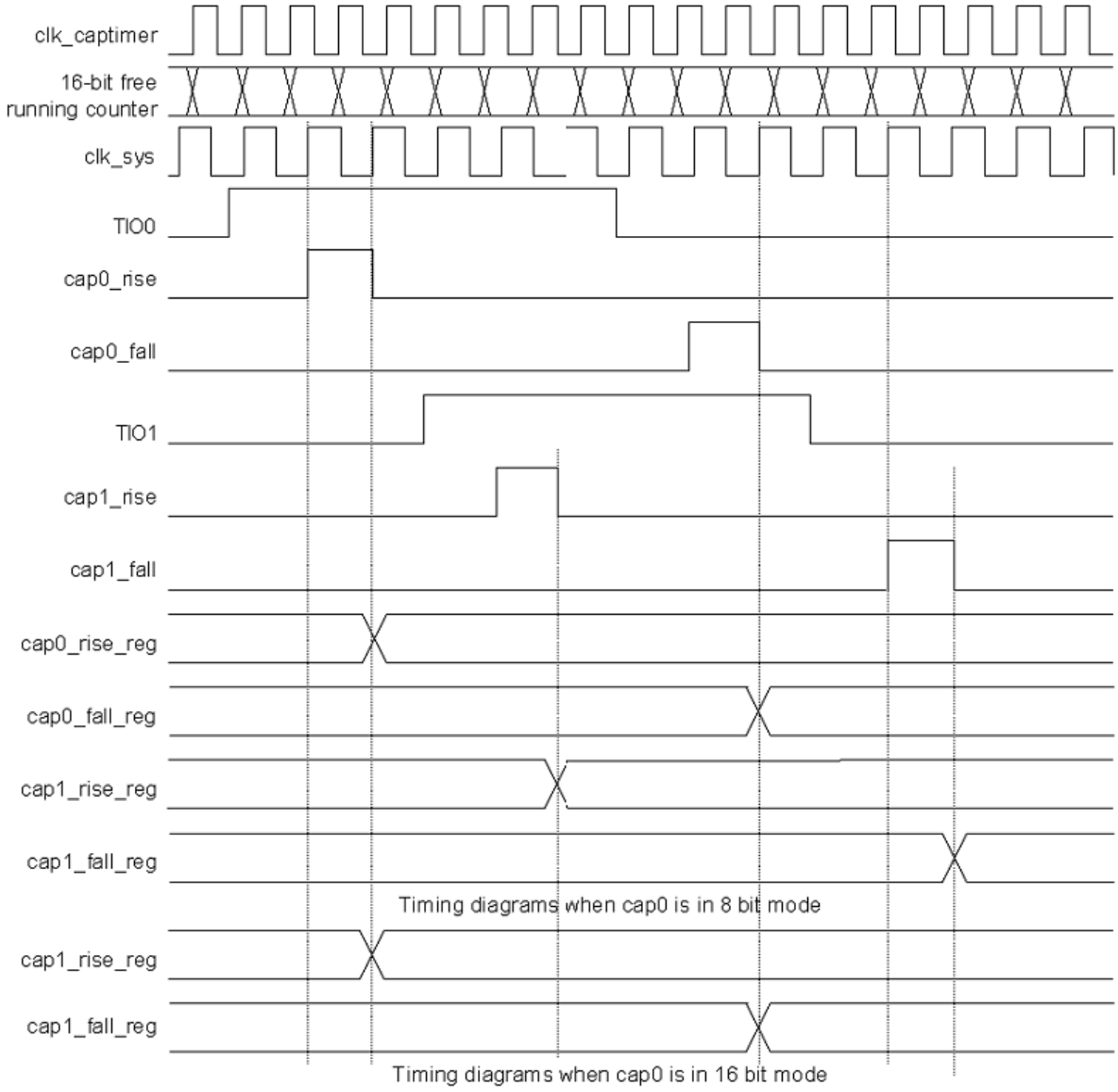
Table 19-8. Programmable Interval Timer Low (PITMRL) [0x26] [R]

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

Bit [7:0]: Prog Interval Timer [7:0]

This register holds the low-order byte of the 12-bit programmable interval timer. Reading this register moves the high-order byte into a holding register allowing an automatic read of all 12 bits simultaneously.

Figure 19-3. Timer Functional Sequence Diagram



20. Interrupt Controller

The interrupt controller and its associated registers allow the user's code to respond to an interrupt from almost every functional block in the enCoRe II LV devices. The registers associated with the interrupt controller are disabled either globally or individually. The registers also provide a mechanism for users to clear all pending and posted interrupts or clear individual posted or pending interrupts.

Table 20-1 lists all interrupts and the priorities that are available in the enCoRe II LV devices.

Table 20-1. Interrupt Priorities, Address, and 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

20.4 Interrupt Registers

20.4.1 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 enables the user to determine all posted interrupts.

Table 20-1. 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 is no posted interrupt for the corresponding hardware.

1 = There is a posted interrupt for the corresponding hardware.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits and to the ENSWINT (Bit 7 of the INT_MSK3 Register) posts the corresponding hardware interrupt.

The GPIO interrupts are edge-triggered.

Table 20-2. Interrupt Clear 1 (INT_CLR1) [0xDB] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	TCAP0	Prog Interval Timer	1-ms Program-mable 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 is no posted interrupt for the corresponding hardware.

1 = There is a posted interrupt for the corresponding hardware.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT (Bit 7 of the INT_MSK3 Register) posts the corresponding hardware interrupt.

Table 20-3. 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 is no posted interrupt for the corresponding hardware.

1 = There is a posted interrupt for the corresponding hardware.

Writing a '0' to the bits clears the posted interrupts for the corresponding hardware. Writing a '1' to the bits AND to the ENSWINT (Bit 7 of the INT_MSK3 Register) posts the corresponding hardware interrupt.

Document Title: CY7C601xx/CY7C602xx, enCoRe™ II Low-Voltage Microcontroller Document Number: 38-16016				
Rev.	**ECN**	**Orig. of Change**	**Submission Date**	**Description of Change**
*K	4499620	SELV	09/11/2014	Updated Package Diagrams: spec 51-85025 – Changed revision from *E to *F. spec 51-85055 – Changed revision from *C to *D. spec 51-85019 – Changed revision from *B to *C. spec 51-85061 – Changed revision from *D to *F. Updated in new template. Completing Sunset Review.

Sales, Solutions, and Legal Information

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/USB
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support

© Cypress Semiconductor Corporation, 2006-2014. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Any Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.

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