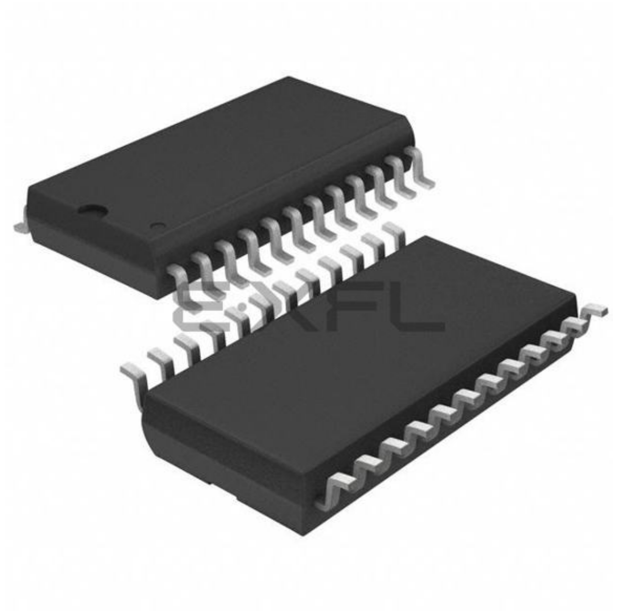
Infineon Technologies - CY7C60223-SXC Datasheet



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-SOIC (0.295", 7.50mm Width)
Supplier Device Package	24-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c60223-sxc

3. Contents

4. Applications

The CY7C601xx and CY7C602xx are targeted for the following applications:

- PC wireless human interface devices (HID)
  - Mice (optomechanical, optical, trackball)
  - Keyboards
  - Presenter tools
- Gaming
  - Joysticks
  - Gamepad
- General-purpose wireless applications
  - Remote controls
  - Barcode scanners
  - POS terminal
  - Consumer electronics
  - Toys

5. Introduction

The enCoRe II LV 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. Other external components, such as wakeup circuitry, are also integrated into this chip.

The enCoRe II LV is a low-voltage, low-cost 8-bit flash-programmable microcontroller.

The enCoRe II LV features up to 36 GPIO pins. The I/O pins are grouped into five ports (Port 0 to 4). The pins on ports 0 and 1 are configured individually, when the pins on ports 2, 3, and 4 are only configured as a group. Each GPIO port supports high-impedance inputs, configurable pull-up, open-drain output, CMOS, and 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 is 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, in addition to the port interrupt vector, three dedicated pins that have independent interrupt vectors (P0.2–P0.4).

The enCoRe II LV features an internal oscillator. Optionally, an external 1-MHz to 24-MHz crystal is used to provide a higher precision reference. The enCoRe II LV also supports external clock.

The enCoRe II LV has 8 KB of flash for user code and 256 bytes of RAM for stack space and user variables.

In addition, enCoRe II LV includes a WDT, a vectored interrupt controller, a 16-bit free-running timer with capture registers, and a 12-bit programmable interval timer. The power on reset (POR) circuit detects when power is applied to the device, resets the logic to a known state, and executes instructions at flash address 0x0000. When power falls below a programmable trip voltage, it generates a reset or is configured to generate an interrupt. There is a LVD circuit that detects when V_{CC} drops below a programmable trip voltage. This is 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 WDT ensures 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 or 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 wakeup timer interrupt. The wakeup 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 also generates an interrupt when the free-running counter overflow occurs – every 16.384 ms. The duration of an event under firmware control is measured by reading the timer at the start and end of an event, then calculating the difference between the two values. The two 8-bit capture timer registers save a programmable 8-bit range of the free-running timer when a GPIO edge occurs on the two capture pins (P0.5 and P0.6). The two 8-bit capture registers are ganged into a single 16-bit capture register.

The enCoRe II LV supports in-system programming by using the P1.0 and P1.1 pins as the serial programming mode interface.

6. Conventions

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

□	Available in all enCoRe II LV family members
■	CY7C601xx only

7. Pinouts

Figure 7-1. Package Configurations

Top View

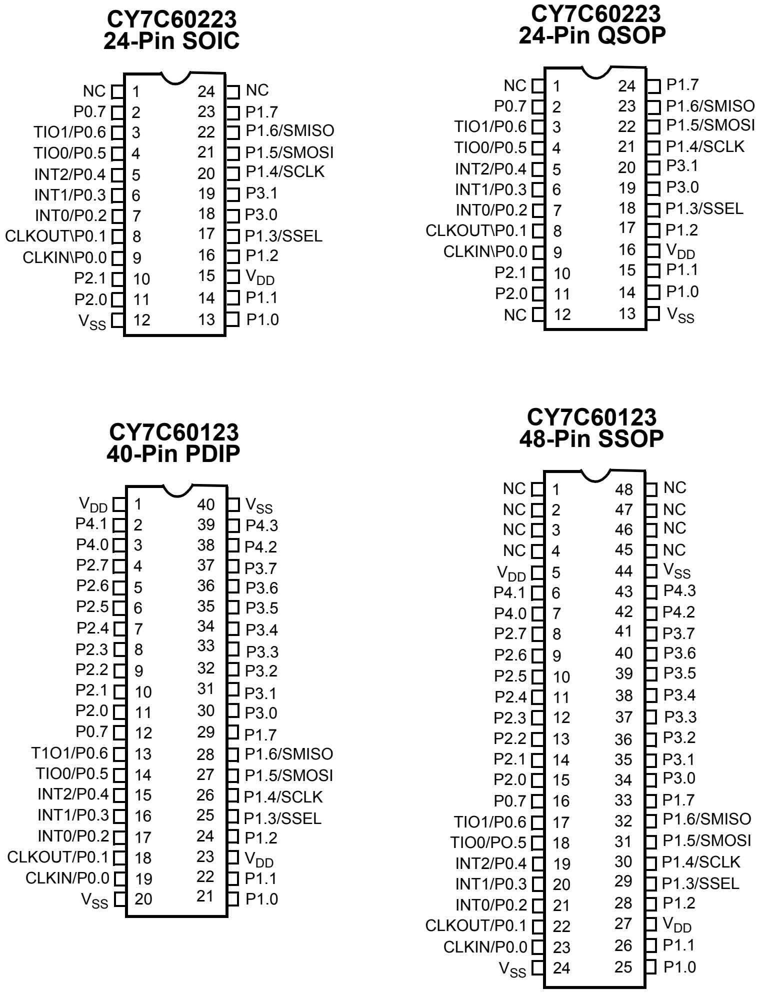


Table 8-1. enCoRe II LV Register Summary (continued)

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
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
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	00001000
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 or clear

d = Calibration value. Must not change during normal use

9. 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 9-1. 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) directly addresses the full 8 KB of program memory space.

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

The index register (CPU_X) holds an offset value 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 is also 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 (AND, OR, XOR). See Table 11-1 on page 13.

10. CPU Registers

10.1 Flags Register

The flags register is only set or reset with logical instruction.

Table 10-1. 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	R/W	R/W	R/W
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 is a carry in the previous logical or arithmetic operation.

0 = No carry

1 = Carry

Bit 1: Zero

Set by CPU to indicate whether there is a zero result in the previous logical or 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* are used to set and clear the CPU_F bits.

10.2 Addressing Modes

10.2.1 Source Immediate

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

Table 10-7. 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.

10.2.2 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 10-8. 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.

10.2.3 Source Indexed

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

Table 10-9. 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.

10.2.4 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 10-10. Destination Direct

Opcode	Operand 1
Instruction	Destination address

Examples

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

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.

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

Opcode Hex	Cycles	Bytes	Instruction Format[1, 2]	Flags
12	6	2	SUB A, [expr]	C, Z
13	7	2	SUB A, [X+expr]	C, Z
14	7	2	SUB [expr], A	C, Z
15	8	2	SUB [X+expr], A	C, Z
16	9	3	SUB [expr], expr	C, Z
17	10	3	SUB [X+expr], expr	C, Z
18	5	1	POP A	Z
19	4	2	SBB A, expr	C, Z
1A	6	2	SBB A, [expr]	C, Z
1B	7	2	SBB A, [X+expr]	C, Z
1C	7	2	SBB [expr], A	C, Z
1D	8	2	SBB [X+expr], A	C, Z
1E	9	3	SBB [expr], expr	C, Z
1F	10	3	SBB [X+expr], expr	C, Z
20	5	1	POP X	
21	4	2	AND A, expr	Z
22	6	2	AND A, [expr]	Z
23	7	2	AND A, [X+expr]	Z
24	7	2	AND [expr], A	Z
25	8	2	AND [X+expr], A	Z
26	9	3	AND [expr], expr	Z
27	10	3	AND [X+expr], expr	Z
28	11	1	ROMX	Z
29	4	2	OR A, expr	Z
2A	6	2	OR A, [expr]	Z
2B	7	2	OR A, [X+expr]	Z
2C	7	2	OR [expr], A	Z
3F	10	2	MVI [[expr]++], A	
40	4	1	NOP	
41	9	3	AND reg[expr], expr	Z
42	10	3	AND reg[X+expr], expr	Z
43	9	3	OR reg[expr], expr	Z
44	10	3	OR reg[X+expr], expr	Z
45	9	3	XOR reg[expr], expr	Z
46	10	3	XOR reg[X+expr], expr	Z
47	8	3	TST [expr], expr	Z
48	9	3	TST [X+expr], expr	Z
49	9	3	TST reg[expr], expr	Z
4A	10	3	TST reg[X+expr], expr	Z
4B	5	1	SWAP A, X	Z
4C	7	2	SWAP A, [expr]	Z
4D	7	2	SWAP X, [expr]	
4E	5	1	SWAP A, SP	Z
4F	4	1	MOV X, SP	
50	4	2	MOV A, expr	Z
51	5	2	MOV A, [expr]	Z
52	6	2	MOV A, [X+expr]	Z
53	5	2	MOV [expr], A	
54	6	2	MOV [X+expr], A	
55	8	3	MOV [expr], expr	
56	9	3	MOV [X+expr], expr	
57	4	2	MOV X, expr	
58	6	2	MOV X, [expr]	
59	7	2	MOV X, [X+expr]	
6C	8	2	RLC [X+expr]	C, Z
6D	4	1	RRC A	C, Z
6E	7	2	RRC [expr]	C, Z
6F	8	2	RRC [X+expr]	C, Z
70	4	2	AND F, expr	C, Z
71	4	2	OR F, expr	C, Z
72	4	2	XOR F, expr	C, Z
73	4	1	CPL A	Z
74	4	1	INC A	C, Z
75	4	1	INC X	C, Z
76	7	2	INC [expr]	C, Z
77	8	2	INC [X+expr]	C, Z
78	4	1	DEC A	C, Z
79	4	1	DEC X	C, Z
7A	7	2	DEC [expr]	C, Z
7B	8	2	DEC [X+expr]	C, Z
7C	13	3	LCALL	
7D	7	3	LJMP	
7E	10	1	RETI	C, Z
7F	8	1	RET	
8x	5	2	JMP	
9x	11	2	CALL	
Ax	5	2	JZ	
Bx	5	2	JNZ	
Cx	5	2	JC	
Dx	5	2	JNC	
Ex	7	2	JACC	
Fx	13	2	INDEX	Z

12. Memory Organization

12.1 Flash Program Memory Organization

Figure 12-1. Program Memory Space with Interrupt Vector Table

after reset

16-bit PC →

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	

Table 12-8. 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

12.5.6 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 8 KB 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 previous contents. The bulk program is followed by a second erase that leaves the flash macro 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 erase of the user space, the protection block for the flash macro with the highest address range is erased. Following 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 result of the EraseAll function is that all user data in 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.

Besides the keys, the CLOCK and DELAY parameter block values are also set.

Table 12-9. 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

12.5.7 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 12-10. Table Read Parameters

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

The table space for the enCoRe II LV is simply a 64-byte row broken up into eight tables of eight bytes (see Figure 12-3 on page 21). The tables are numbered zero through seven. All user and hidden blocks in the CY7C601xx/CY7C602xx parts consist of 64 bytes.

An internal table (Table 0) holds the silicon ID and returns the revision ID. The silicon ID is returned in SRAM, while the revision and family IDs are 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 and also redundantly placed in SROM Table 1. This is discussed in detail later in this section.

SROM Table 1 holds Family/Die ID and revision ID values for the device and returns a one-byte internal revision counter. The internal revision counter starts with a value of zero and is incremented when one of the other revision numbers is not incremented. It is reset to zero when one of the other revision numbers is incremented. The internal revision count is returned in the CPU_A register. The CPU_X register is always set to FFh when Table 1 is read. The CPU_A and CPU_X registers always return a value of FFh when Tables 2 to 7 are read. The BLOCKID value, in the parameter block, indicates which table must be returned to the user. Only the three least significant bits of the BLOCKID parameter are used by TableRead function for enCoRe II LV devices. 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.

The return values for the corresponding table calls are tabulated as shown in Table 12-11.

Table 12-11. Return Values for Table Read

Table Number	Return Value	
	A	X
0	Revision ID	Family ID
1	Internal revision counter	0xFF
2-7	0xFF	0xFF

14.1 Power On Reset

POR occurs every time the power to the device is switched on. POR is released when the supply is typically 2.6 V 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 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 less than 200 ms.

Note The PORS status bit is set at POR and is only cleared by the user; it cannot be set by firmware.

14.2 Watchdog Timer Reset

The user has the option to enable the WDT. The WDT is enabled by clearing the PORS bit. When the PORS bit is cleared, the WDT cannot be disabled. The only exception to this is if a POR event takes place, which disables 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 programs the sleep time period using the sleep timer bits of the OSC_CR0 register (Table 13-2 on page 25). When the sleep time elapses (sleep timer overflows), an interrupt to the sleep timer interrupt vector is generated.

The watchdog timer period is automatically set to be three counts of the sleep timer overflow. 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 either clears the WDT, or the WDT and the sleep timer. Whenever the user writes to the reset WDT register (RES_WDT), the WDT is cleared. If the data written is the hex value 0x38, the sleep timer is also cleared at the same time.

Table 14-2. 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 clears the watchdog timer, a write of 0x38 also clears the sleep timer.

Bit [7:0]: Reset watchdog timer [7:0]

15. Sleep Mode

The CPU is put to sleep only 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 remains asleep until an interrupt is pending, or there is a reset event (either a POR or a WDT reset).

The low-voltage detection (LVD) circuit drops into fully functional power reduced states, and the latency for the LVD is increased. The actual latency is traded against power consumption by changing sleep duty cycle field of the ECO_TR register.

The internal 32-kHz low-speed oscillator remains running. Before entering the suspend mode, firmware optionally configures the 32-kHz low-speed oscillator to operate in a low-power mode to help reduce the overall power consumption (using the 32-kHz low-power bit, as shown in Table 13-4 on page 30). This helps to save approximately 5 µA; however, the trade off is that the 32-kHz low-speed oscillator is less accurate (–53.12% to +56.25% deviation).

All interrupts remain active. Only the occurrence of an interrupt wakes the part from sleep. The stop bit in the system status and control register (CPU_SCR) is 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 wakes the system. As a result, any interrupt not intended for waking is disabled through the interrupt mask registers.

When the CPU enters sleep mode the CPUCLK select (Bit 1, Table 13-1 on page 24) 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, the firmware needs 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 must disable the external oscillator before entering sleep mode. In systems where the CPU runs off the external oscillator, the firmware needs to switch the CPU to the internal oscillator before disabling the external oscillator.

On exiting sleep mode, after 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 is also used to provide periodic interrupts during non-sleep modes.

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.

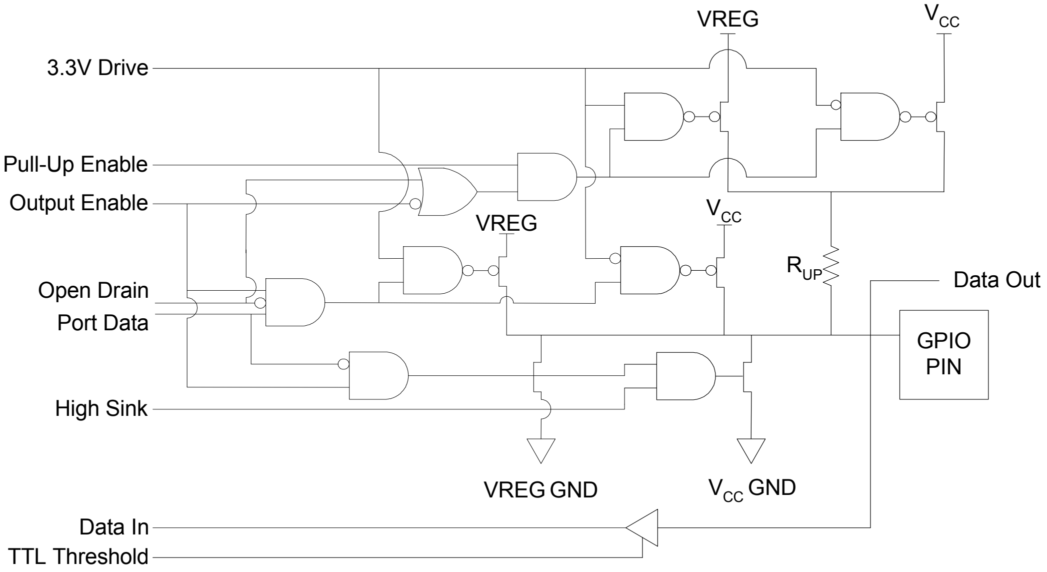
For pins with shared functions there are some special cases.

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.

17.2.15 P1.1 Configuration

Table 17-7. 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} (see DC Characteristics on page 60)

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

17.2.16 P1.2 Configuration

Table 17-8. 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.

Note: Table 13-3 on page 26 is used to select the external or internal clock in enCoRe II devices

17.2.17 P1.3 Configuration (SSEL)

Table 17-9. 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. This register exists in all enCoRe II LVparts.

The P1.3 GPIO's threshold is always set to TTL.

When the SPI hardware is enabled or 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.

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 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.

20.1 Architectural Description

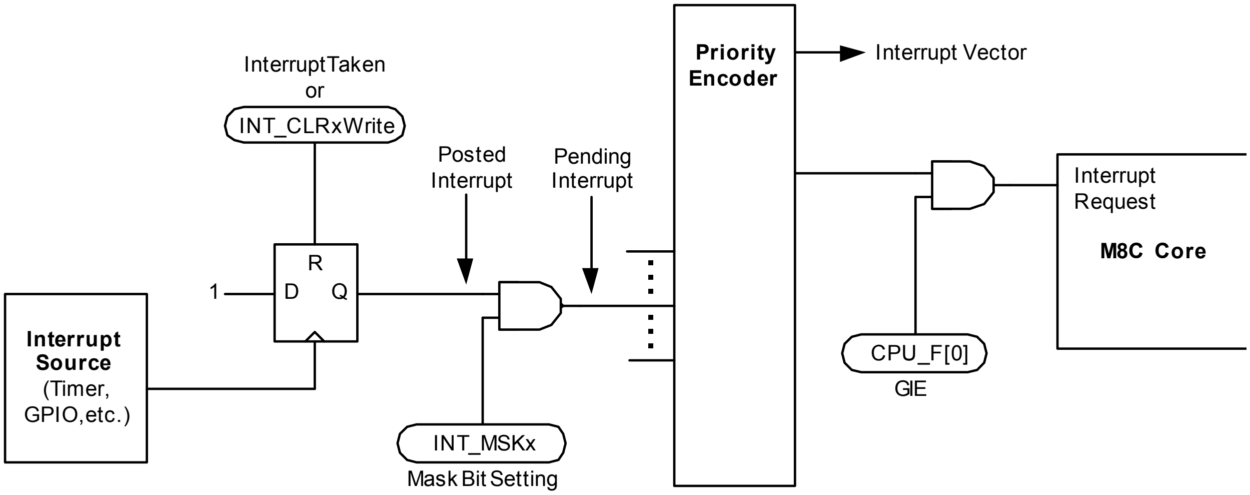
An interrupt is posted when its interrupt conditions occur. This results in the flip-flop in Figure 20-1 clocking in a ‘1’. The interrupt remains posted until the interrupt is taken or until it is cleared by writing to the appropriate INT_CLRx register.

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 is 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 are 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 enCoRe II LV interrupt controller is shown in Figure 20-1.

Figure 20-1. Interrupt Controller Block Diagram



20.2 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 (for example, 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.
1. The current executing instruction finishes.
2. 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. Because 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 (for example, 0004h for the POR and LVD interrupt).
1. Program execution vectors to the interrupt table. Typically, a LJMP instruction in the interrupt table sends execution to the user's interrupt service routine (ISR) for this interrupt.
2. The ISR executes. Note that interrupts are disabled because GIE =0. In the ISR, interrupts are re-enabled if desired, by setting GIE = 1 (avoid stack overflow).
3. 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, because GIE = 1 again.
4. Execution resumes at the next instruction, after the one that occurred before the interrupt. However, if there are more pending interrupts, the subsequent interrupts are processed before the next normal program instruction.

20.3 Interrupt Latency

The time between the assertion of an enabled interrupt and the start of its ISR is 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 is 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.

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 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 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.

Figure 21-2. GPIO Timing Diagram

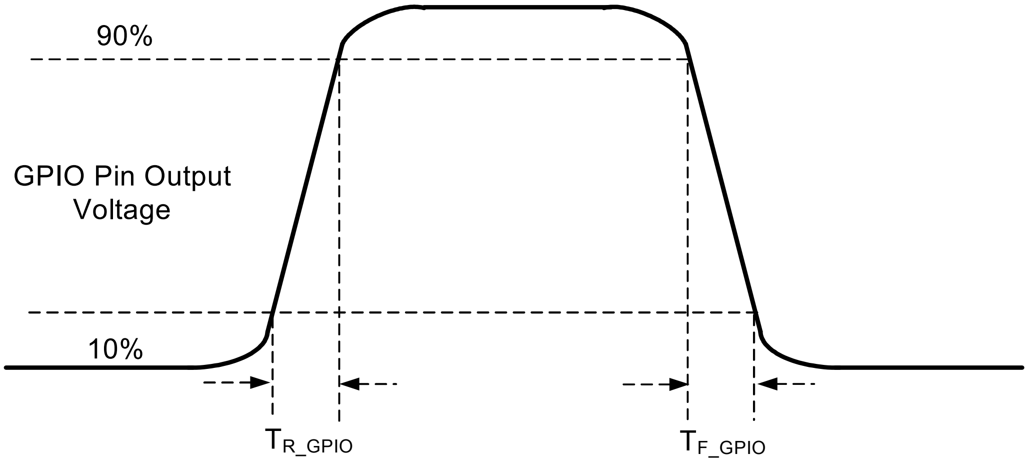
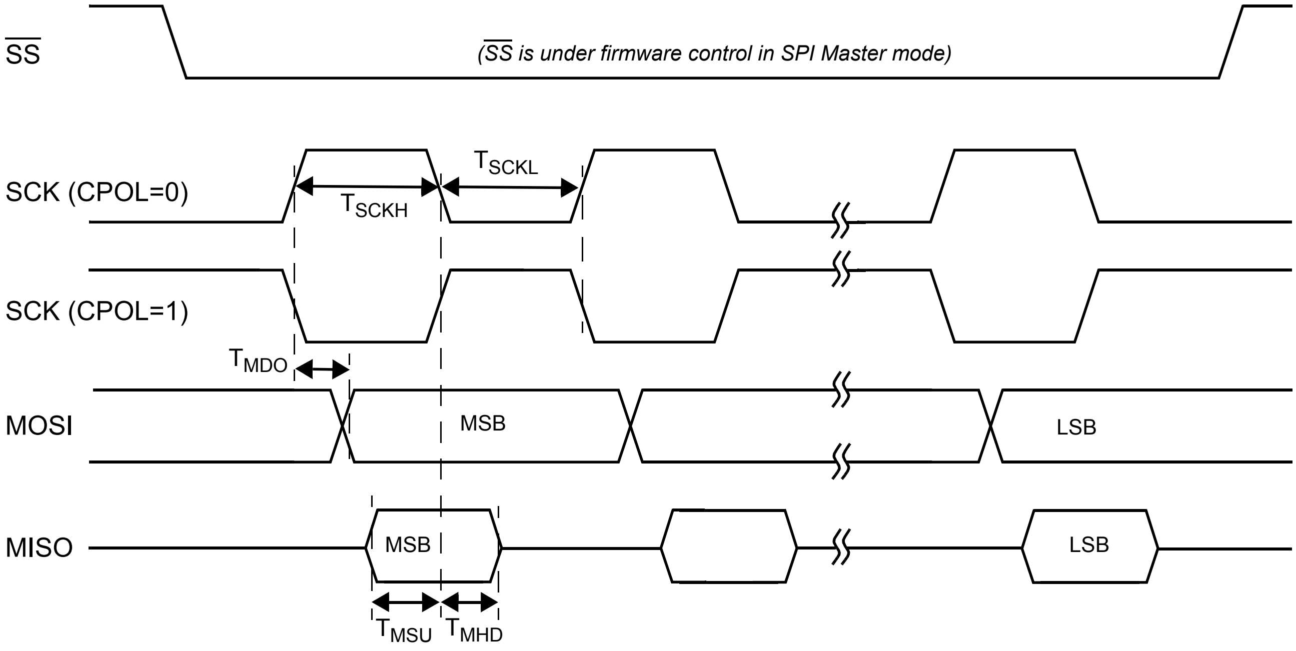


Figure 21-3. SPI Master Timing, CPHA = 1



25. Document History Page

Document Title: CY7C601xx/CY7C602xx, enCoRe™ II Low-Voltage Microcontroller
Document Number: 38-16016

Rev.	ECN	Orig. of Change	Submission Date	Description of Change
**	327601	BON	See ECN	New data sheet
*A	400134	BHA	See ECN	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 24 MHz oscillator accuracy information Added information on sending/receiving data when using 32 KHz oscillator
*B	505222	TYJ	See ECN	Minor text changes GPIO capacitance and timing diagram included Method to clear Capture Interrupt Status bit discussed Sleep and Wakeup sequence documented PIT Timer registers' R/W capability corrected to read only Modified Free-running Counter text in section 17.1.1
*C	524104	KKVTMP	See ECN	Change title from Wireless enCoRe II to enCoRe II Low Voltage
*D	1821746	VGT / FSU / AESA	See ECN	Changed "High current drive" on GPIO pins to "2 mA source current on all GPIO pins". Changed the storage temperature from -40C to 90C in "Absolute Maximum ratings" section. Added the line "The GPIOs interrupts are edge-triggered." in Tables 19-2 and 19-6. Made timing changes in Table 43. Added Figure 12-1 (SROM Table) and text after it. Also modified Table 12-1 based on Figure 12-1 (SROM Table). Changed "CAPx" to "TIOx" in Tables 18-8 and 18-9. Changed "Capturex" to "TIOx" in Figure 18-3.
*E	2620679	CMCC / PYRS	12/12/08	Added Package Handling information Formatted code in Clocking section, Removed reference to external crystal oscillator in Tables 12-2 and 12-4
*F	2761532	DVJA	09/09/2009	Changed default value of the Sleep Timer from 00(512 Hz) to 01(64 Hz) in the OSC_CR0 [0x1E0] register.
*G	2899862	XUT	03/26/10	Removed obsolete parts from the ordering information table Updated package diagrams
*H	2978027	DATT	07/12/2010	Sunset review; no technical updates. Updated content to meet style guide and template requirements.
*I	2999570	MLIM	08/03/2010	Minor change to correct revision in the document footer.
*J	3275367	NXZ	06/06/2011	Removed "CY7C60223 24-pin PDIP and CY7C60113 28-pin SSOP" from Figure 7-1. Removed "28 SSOP" and "24 PDIP" columns from Table 7-1. Removed Figure 24-2 (24-pin PDIP) and Figure 24-4 (28-pin SSOP) Updated description field of P1.0 and P1.1 in Table 7-1 on page 5

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

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.