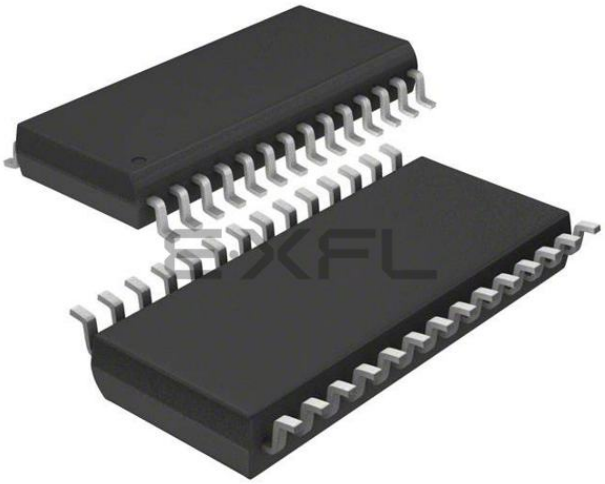


Welcome to E-XFL.COM

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	24
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	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c60113-pvxc

8. 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 8-1. 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 Kbytes 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 10-1 on page 12.

9. CPU Registers

9.1 Flags Register

The Flags Register is only set or reset with logical instruction.

Table 9-1. 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.

9.2 Addressing Modes

9.2.1 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 9-7. 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.
```

9.2.2 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 9-8. 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.
```

9.2.3 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 9-9. 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.
```

9.2.4 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 9-10. 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.
```

Table 10-1. Instruction Set Summary Sorted Numerically by Opcode Order (continued)

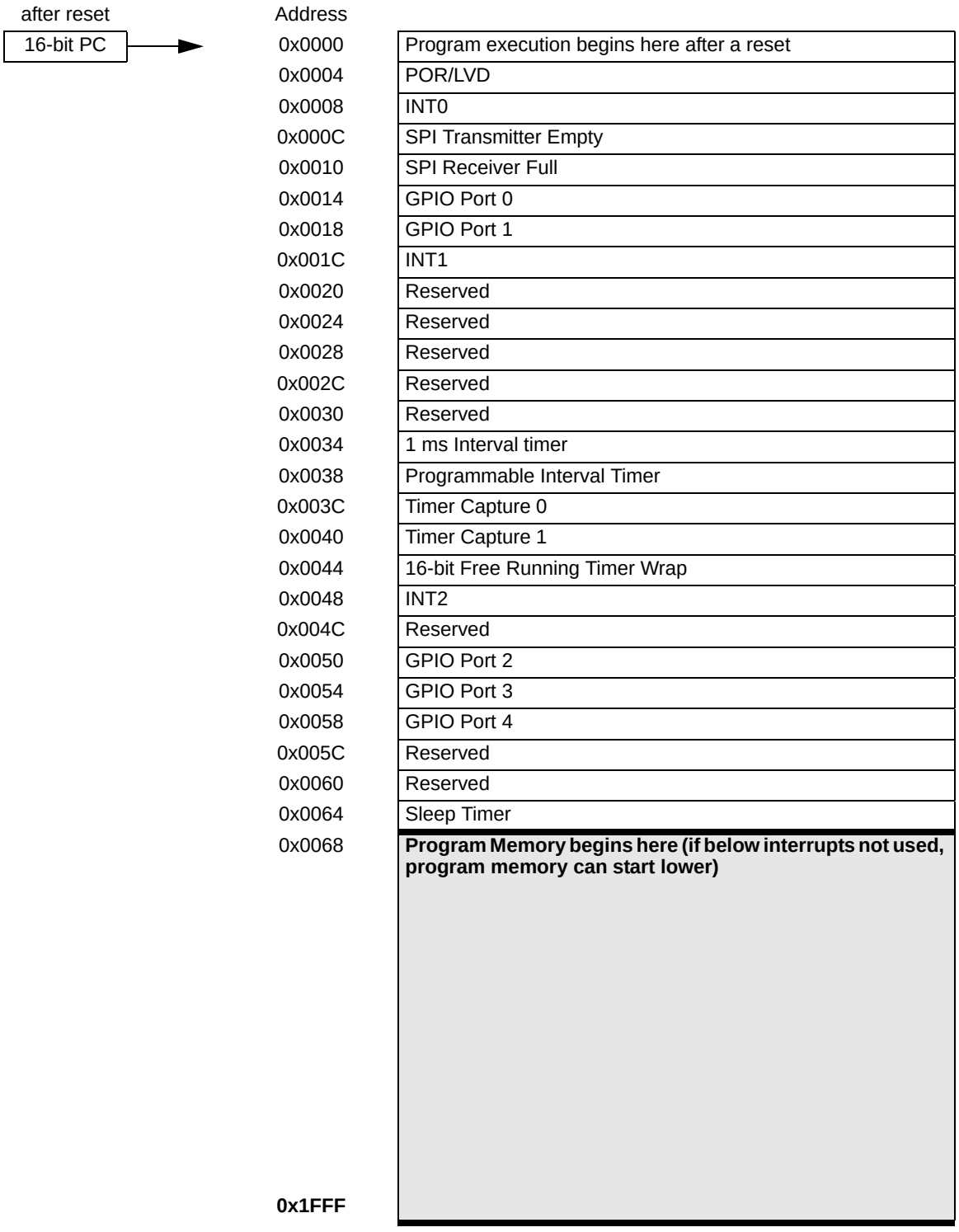
Opcode Hex	Cycles	Bytes	Instruction Format[1, 2]	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags	Opcode Hex	Cycles	Bytes	Instruction Format	Flags
15	8	2	SUB [X+expr], A	C, Z	42	10	3	AND reg[X+expr], expr	Z	6F	8	2	RRC [X+expr]	C, Z
16	9	3	SUB [expr], expr	C, Z	43	9	3	OR reg[expr], expr	Z	70	4	2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

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

11. Memory Organization

11.1 Flash Program Memory Organization

Figure 11-1. Program Memory Space with Interrupt Vector Table



11.5.3 WriteBlock Function

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

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

Table 11-5. WriteBlock Parameters

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

11.5.4 EraseBlock Function

The EraseBlock function is used to erase a block of 64 contiguous bytes in Flash. The EraseBlock function first checks the protection bits and determines if the desired BLOCKID is writable. If write protection is turned on, the EraseBlock function exits setting the accumulator and KEY2 back to 00h. KEY1 has a value of 01h, indicating a write failure. The EraseBlock function is only useful as the first step in programming. Erasing a block does not make data in a block fully unreadable. If the objective is to obliterate data in a block, the best method is to perform an EraseBlock followed by a WriteBlock of all zeros.

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

Table 11-6. EraseBlock Parameters

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

11.5.5 ProtectBlock Function

The enCoRe II LV devices offer Flash protection on a block-by-block basis. Table 11-7 lists the protection modes available. In the table, ER and EW indicate the ability to perform external reads and writes; IW is used for internal writes. Internal reading is always permitted using the ROMX instruction. The ability to read using the SROM ReadBlock function is indicated by SR. The protection level is stored in two bits according to Table 11-7. These bits are bit packed into 64 bytes of the protection block. Therefore, each protection block byte stores the protection level for four Flash blocks. The bits are packed into a byte, with the lowest numbered block's protection level stored in the lowest numbered bits in Table 11-7.

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

Table 11-7. Protection Modes

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

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

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

Table 11-8. 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

11.5.6 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 Kbytes of user data.

The EraseAll function begins by erasing the user space of the Flash macro with the highest address range. A bulk program of all zeros is then performed on the same Flash macro, to destroy all traces of 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 11-9. 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

11.5.7 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 11-10. 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. 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 corresponding Table calls are tabulated as shown in Table 11-11.

Table 11-11. 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

11.6 SROM Table Read Description

The Silicon IDs for enCoRe II LV devices are stored in SROM tables in the part, as shown in Figure 11-3.

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 15, the SROM variables occupy address F8h through FFh in the SRAM. Each of the variables and their definition are given in the section, SROM on page 15.

```
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
11-1 on page 15)
          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 11-3. 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								

11.6.1 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 11-1. 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

15.1 POR Compare State

Table 15-2. Voltage Monitor Comparators Register (VLTCMP) [0x1E4] [R]

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

This read-only register allows reading the current state of the LVD and PPOR comparators.

Bit [7:2]: Reserved

Bit 1: LVD

This bit is set to indicate that the LVD comparator has tripped, indicating that the supply voltage has gone below the trip point set by VM[2:0] (See Table 15-1 on page 34).

0 = No low-voltage-detect event

1 = A low-voltage-detect has tripped

Bit 0: PPOR

This bit is set to indicate that the PPOR comparator has tripped, indicating that the supply voltage is below the trip point set by PORLEV[1:0].

0 = No precision-power-on-reset event

1 = A precision-power-on-reset event has occurred

Note This register exists in the second bank of I/O space. This requires setting the XIO bit in the CPU flags register.

15.2 ECO Trim Register

Table 15-3. ECO (ECO_TR) [0x1EB] [R/W]

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

This register controls the ratios (in numbers of 32 kHz clock periods) of “on” time versus “off” time for LVD and POR detection circuit.

Bit [7:6]: Sleep Duty Cycle [1:0]

0 0 = 1/128 periods of the Internal 32 kHz low speed oscillator.

0 1 = 1/512 periods of the Internal 32 kHz low speed oscillator.

1 0 = 1/32 periods of the Internal 32 kHz low speed oscillator.

1 1 = 1/8 periods of the Internal 32 kHz low speed oscillator.

Note This register is only accessed in the second bank of I/O space. This requires setting the XIO bit in the CPU flags register.

16.1.3 P2 Data

Table 16-3. 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]

16.1.4 P3 Data

Table 16-4. 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]

16.1.5 P4 Data

Table 16-5. 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.

16.2 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.

16.2.1 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 as well.

16.2.2 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.

16.2.3 TTL Thresh

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

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

16.2.4 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 50 mA sink drive capability. Other pins have 8 mA sink drive capability.

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

16.2.5 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.

16.2.6 Pull Up Enable

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

When clear, the pull up is disabled.

16.2.7 Output Enable

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

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

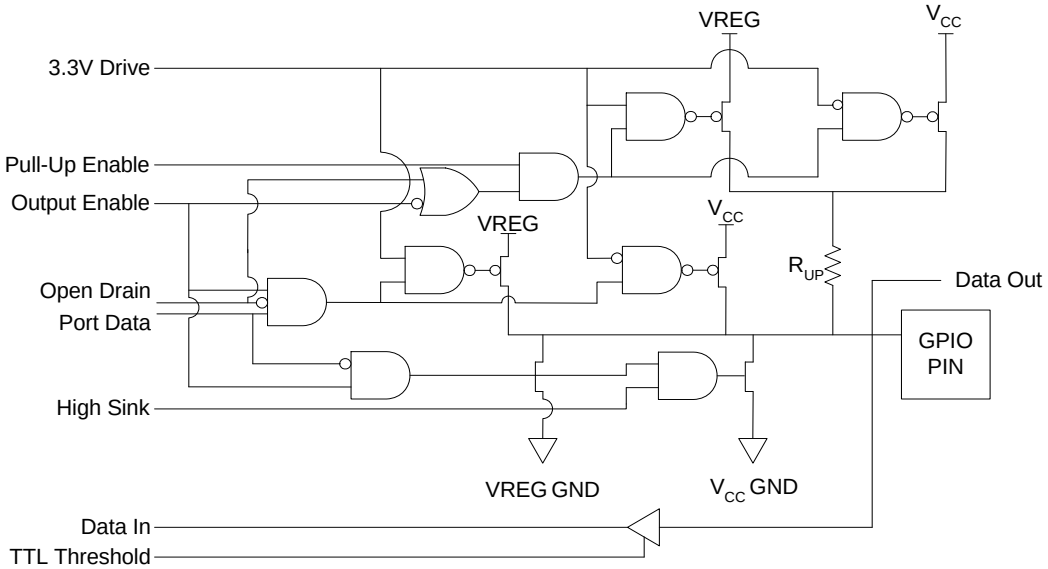
For pins with shared functions there are some special cases.

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.

16.2.8 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 16-1. GPIO Block Diagram



16.2.9 P0.0/CLKIN Configuration

Table 16-1. 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 12-4 on page 25), the GPIO function of the pin is disabled.

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

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

Table 16-10. P1.4–P1.6 Configuration (P14CR–P16CR) [0x11–0x13] [R/W]

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

These registers control the operation of pins P1.4–P1.6, respectively. These registers exist in all enCoRe II LV parts.

Bit 7: SPI Use

0 = Disable the SPI alternate function. The pin is used as a GPIO

1 = Enable the SPI function. The SPI circuitry controls the output of the pin

The P1.4–P1.6 GPIO's threshold is always set to TTL.

When the SPI hardware is enabled, pins that are configured as SPI Use have their output enable and output state controlled by the SPI circuitry. When the SPI hardware is disabled or a pin has its SPI Use bit clear, the pin is controlled by the Output Enable bit and the corresponding bit in the P1 data register.

Regardless of whether any pin is used as an SPI or GPIO pin the Int Enable, Int act Low, High Sink, Open Drain, and Pull up Enable control the behavior of the pin.

Note for Comm Modes 01 or 10 (SPI Master or SPI Slave, see Table 17-2 on page 45)

When configured for SPI (SPI Use = 1 and Comm Modes [1:0] = SPI Master or SPI Slave mode), the input and output direction of pins 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 must be configured as an output; for SPI Slave mode, pin P1.4 must be configured as an input.

16.2.19 P1.7 Configuration

Table 16-11. P1.7 Configuration (P17CR) [0x14] [R/W]

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

This register controls the operation of pin P1.7.

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

The P1.7 GPIO's threshold is always set to TTL.

16.2.20 P2 Configuration

Table 16-12. P2 Configuration (P2CR) [0x15] [R/W]

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

In CY7C602xx, this register controls the operation of pins P2.0–P2.1. In CY7C601xx, this register controls the operation of pins P2.0–P2.7.

The 50 mA sink drive capability is only available on pin P2.7 and only on CY7C601xx. In CY7C602xx, only 8 mA sink drive capability is available on this pin regardless of the setting of the High Sink bit.

Table 17-3. SPI Mode Timing vs. LSB First, CPOL, and CPHA

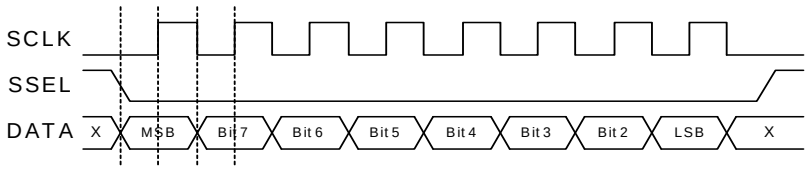
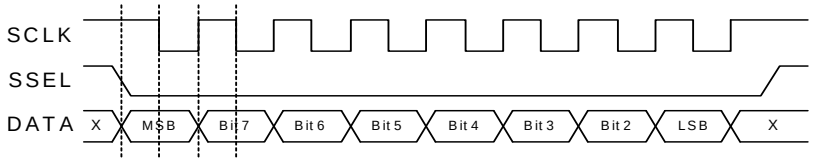
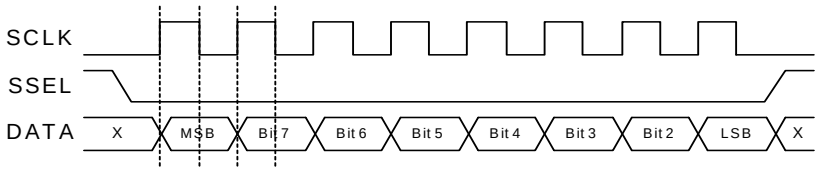
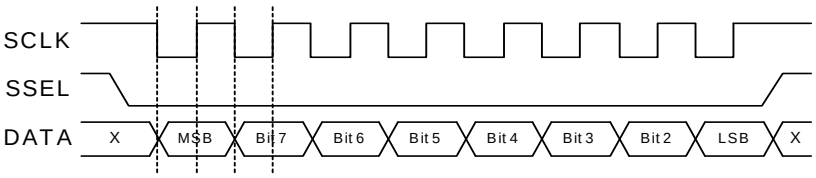
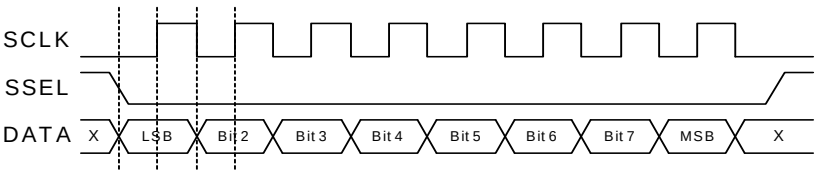
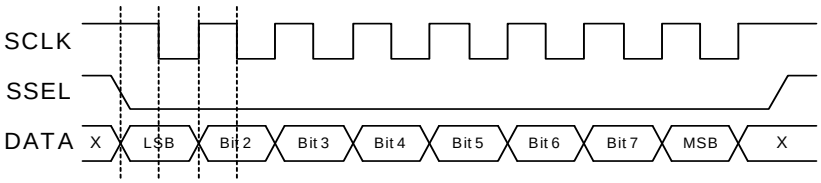
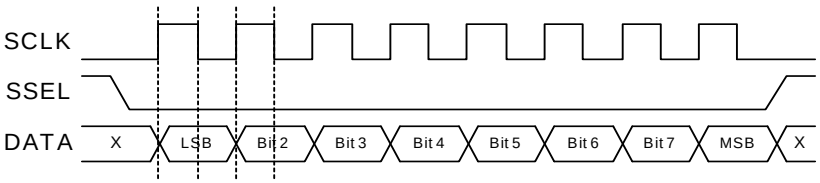
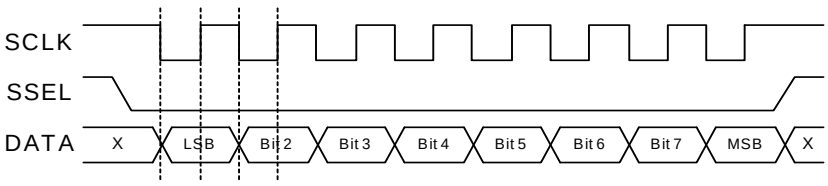
LSB First	CPHA	CPOL	Diagram
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Table 18-2. Free Running Timer High Order Byte (FRTMRH) [0x21] [R/W]

Bit #	7	6	5	4	3	2	1	0
Field	Free Running Timer [15:8]							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Default	0	0	0	0	0	0	0	0

Bit [7:0]: Free Running Timer [15:8]
When reading the free running timer, the low order byte is read first and the high order second. When writing, the low order byte is written first, then the high order byte.

18.1.2 Time Capture

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

Figure 18-2. Time Capture Block Diagram

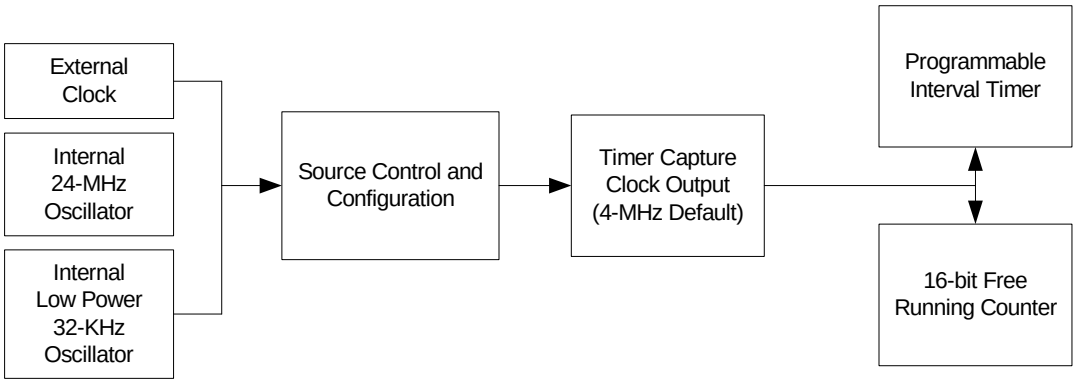
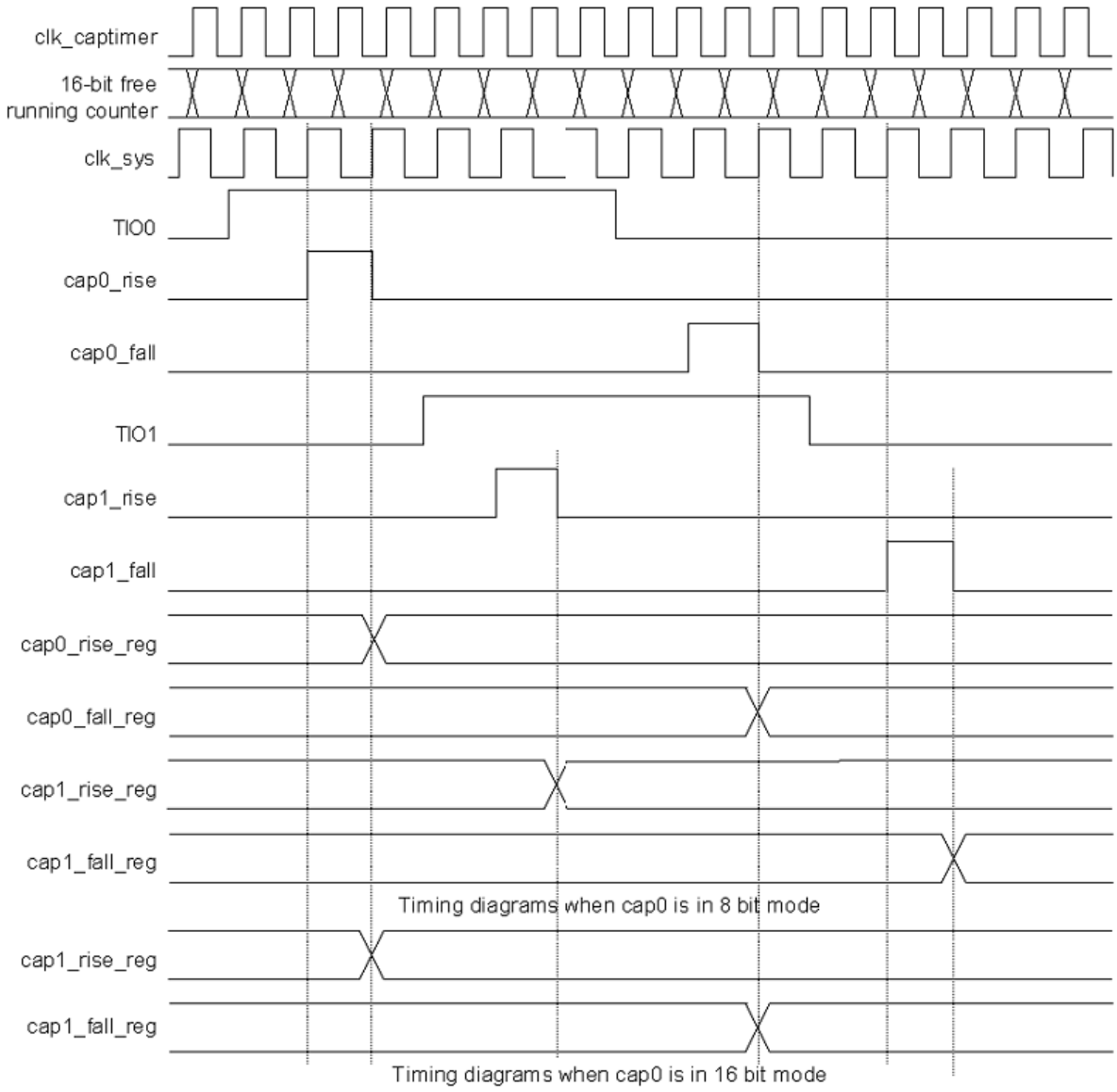


Table 18-1. Timer Configuration (TMRCR) [0x2A] [R/W]

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

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

Figure 18-3. Timer Functional Sequence Diagram



19. 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 19-1 lists all interrupts and the priorities that are available in the enCoRe II LV devices.

Table 19-1. 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. Absolute Maximum Ratings

Storage Temperature ..... –40°C to +90°C
Ambient Temperature with Power Applied..... –0°C to +70°C
Supply Voltage on V_{CC} Relative to V_{SS} .....–0.5V to +7.0V
DC Input Voltage ..... –0.5V to + V_{CC} + 0.5V
DC Voltage Applied to Outputs in High-Z State..... –0.5V to + V_{CC} + 0.5V

Maximum Total Sink Output Current into Port 0 and 1 and Pins..... 70 mA
Maximum Total Source Output Current into GPIO Pins..... 30 mA
Maximum On-chip Power Dissipation on any GPIO Pin..... 50 mW
Power Dissipation ..... 300 mW
Static Discharge Voltage 2200V
Latch up Current ..... 200 mA

20.1 DC Characteristics

Parameter	Description	Conditions	Min	Typical	Max	Unit
	General					
V_{CC1}	Operating Voltage	CPU speed <= 12 MHz	2.7		3.6	V
T_{FP}	Operating Temperature	Flash programming	0		70	°C
I_{CC1}	V_{CC} Operating Supply Current	CPU =12 MHz, Vdd = 3.3V, T = 75°C		4.25	11	mA
		CPU =12 MHz, Vdd = 2.7V, T = 25°C		3.25	-	mA
I_{CC2}	V_{CC} Operating Supply Current	CPU = 6 MHz, Vdd = 3.3V, T = 75°C		3.15	9	mA
		CPU = 6 MHz, Vdd = 3.3V, T = 25°C		2.45	-	mA
I_{CC3}	V_{CC} Operating Supply Current	CPU = 3 MHz, Vdd = 2.7V, T = 25°C		2.0	-	mA
I_{SB1}	Standby Current	Internal and external oscillators, Bandgap, Flash, CPU clock, timer clock all disabled			10	µA
Low Voltage Detect						
V_{LVD}	Low Voltage Detect Trip Voltage	LVDCR [2:0] set to 000	2.681		2.7	V
General Purpose I/O Interface						
R_{UP}	Pull Up Resistance		4		12	KΩ
V_{ICR}	Input Threshold Voltage Low, CMOS Mode	Low to high edge	40%		65%	V_{CC}
V_{ICF}	Input Threshold Voltage Low, CMOS Mode	High to low edge	30%		55%	V_{CC}
V_{HC}	Input Hysteresis Voltage, CMOS Mode	High to low edge	3%		10%	V_{CC}
V_{ILTTL}	Input Low Voltage, TTL Mode				0.72	V
V_{IHTTL}	Input HIGH Voltage, TTL Mode		1.6			V
V_{OL1}	Output Low Voltage, High Drive[4]	I_{OL1} = 50 mA			1.4	V
V_{OL2}	Output Low Voltage, High Drive[4]	I_{OL1} = 25 mA			0.4	V
V_{OL3}	Output Low Voltage, Low Drive	I_{OL2} = 8 mA			0.8	V
V_{OH}	Output High Voltage[4]	I_{OH} = 2 mA	V_{CC} – 0.5			V

Note
4. Available only on CY7C601xx P2.7, P3.7, P0.0, P0.1; CY7C602xx P1.3, P1.4, P1.5, P1.6, P1.7.

23. Package Diagrams

Figure 23-1. 24-Pin (300-Mil) SOIC S13

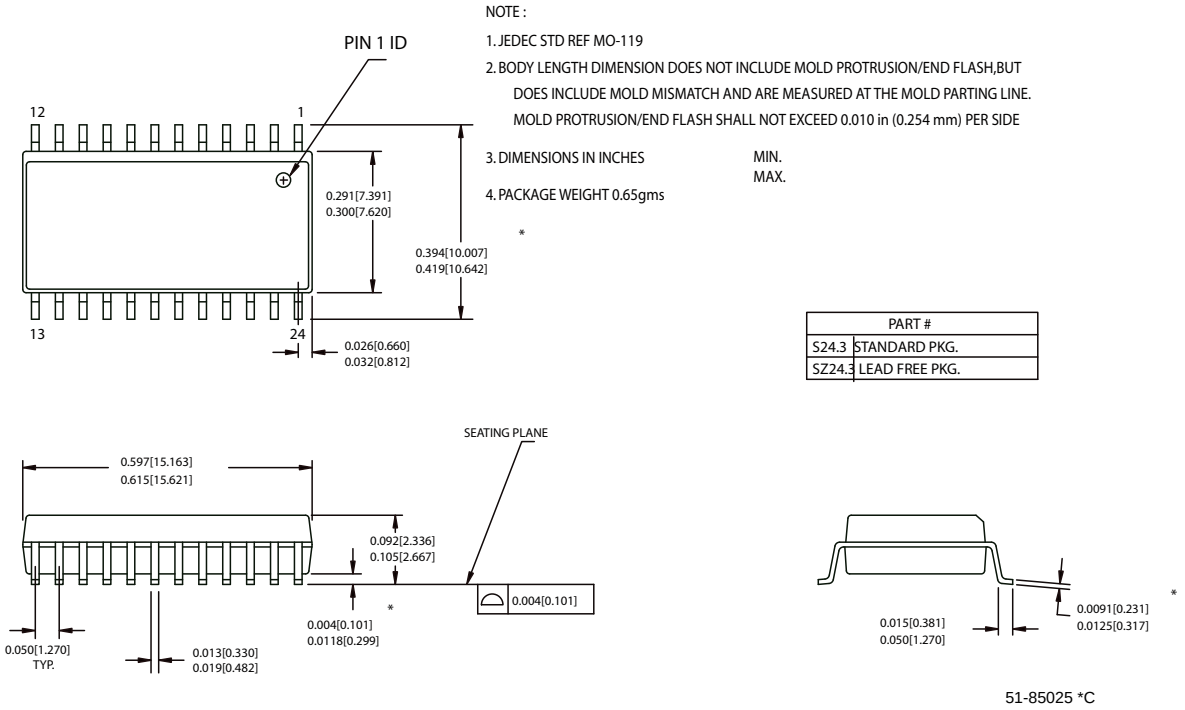


Figure 23-2. 24-Pin (300-Mil) PDIP P13

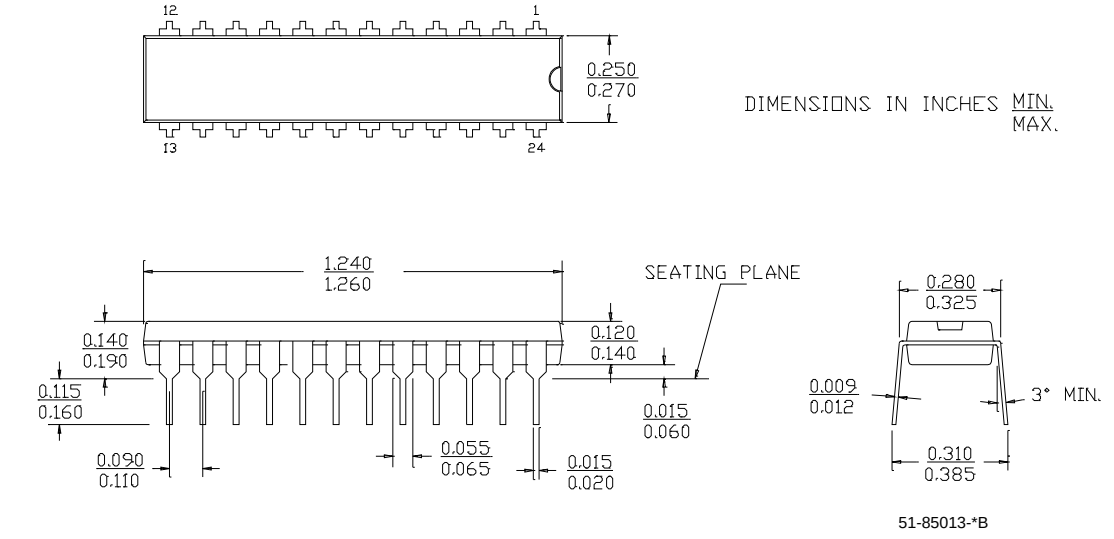


Figure 23-5. 40-Pin (600-Mil) Molded DIP P17

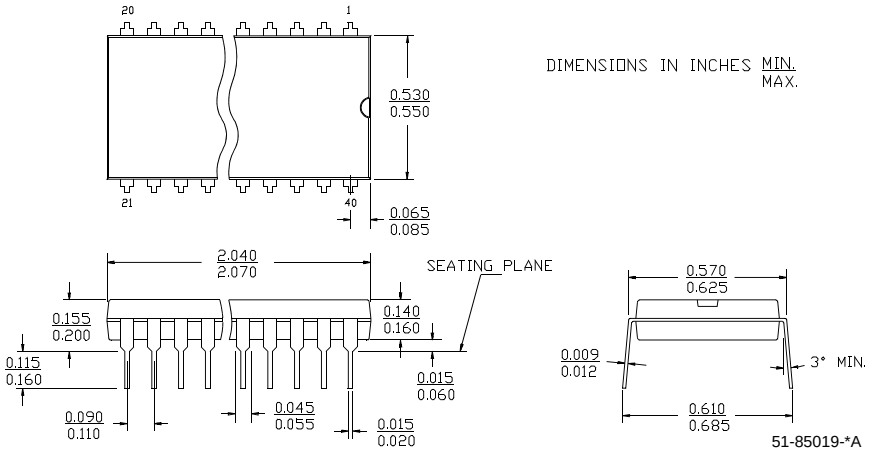
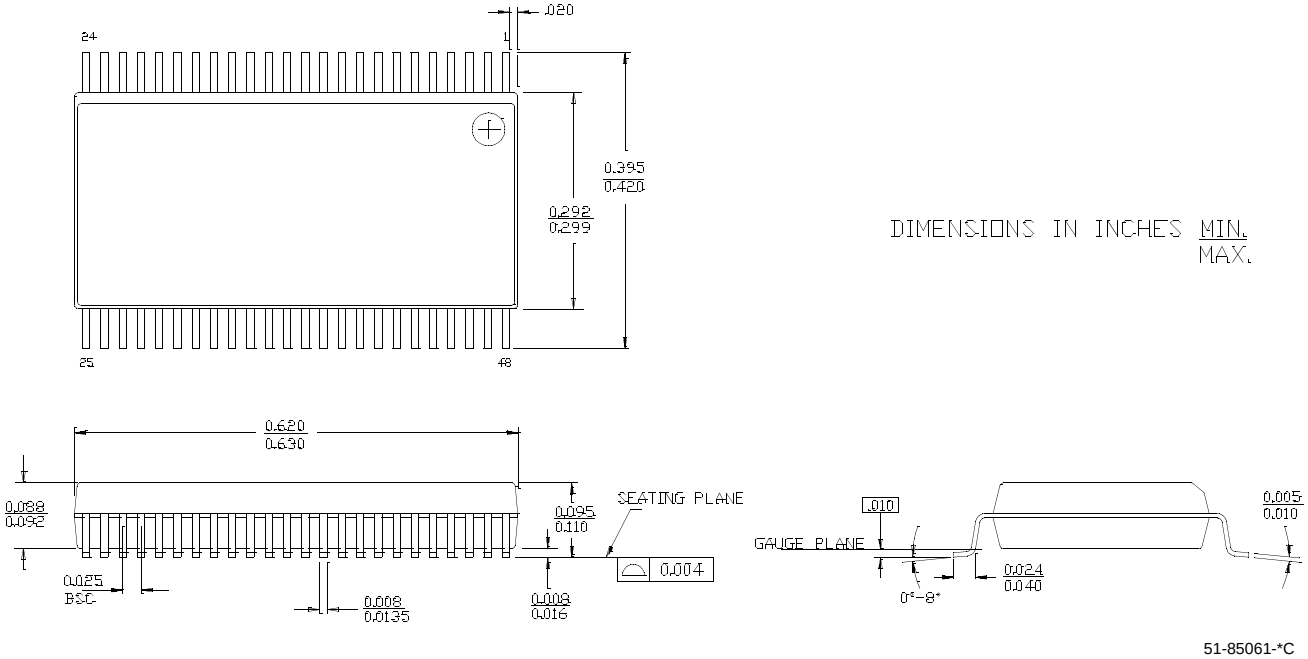


Figure 23-6. 48-Pin Shrunk Small Outline Package O48



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.com/sales.

Products

PSoC	psoc.cypress.com
Clocks & Buffers	clocks.cypress.com
Wireless	wireless.cypress.com
Memories	memory.cypress.com
Image Sensors	image.cypress.com