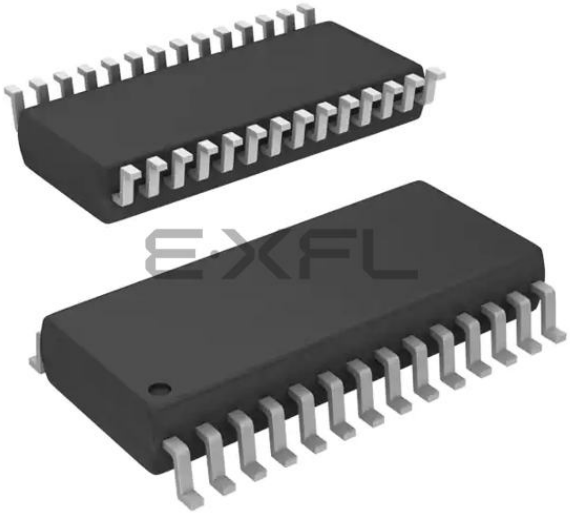


Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Hub/Microcontroller
Core Processor	M8
Program Memory Type	OTP (8kB)
Controller Series	USB Hub
RAM Size	256 x 8
Interface	I²C, USB
Number of I/O	11
Voltage - Supply	4V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c65113c-sxct

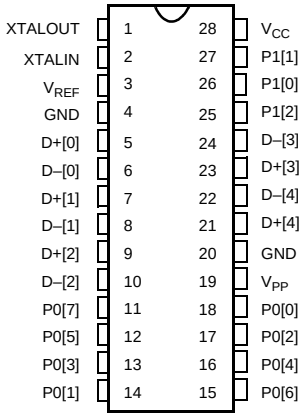
Contents

Pin Configurations **5**
Product Summary Tables **5**
Programming Model **8**
Clocking **11**
Reset **12**
Suspend Mode **13**
General-purpose I/O Ports **14**
12-bit Free-Running Timer **17**
I^2C Configuration Register **18**
I2C-compatible Controller **18**
Processor Status and Control Register **20**
Interrupts **21**
USB Overview **26**
USB Hub **26**
USB Mode Tables **35**
Register Summary **39**
Sample Schematic **41**
Absolute Maximum Ratings **41**
Electrical Characteristics **42**
Switching Characteristics 43
Ordering Information **44**
Ordering Code Definitions 44
Package Diagram **45**
Acronyms **46**
Document Conventions **46**
Units of Measure 46
Document History Page **47**
Sales, Solutions, and Legal Information **48**
Worldwide Sales and Design Support 48
Products 48
PSoC Solutions 48

Pin Configurations

Figure 1. CY7C65113C 28-Pin SOIC

Top View



Product Summary Tables

Pin Assignments

Table 1. Pin Assignments

Name	I/O	28-pin	Description
D+[0], D–[0]	I/O	5, 6	Upstream port, USB differential data.
D+[1], D–[1]	I/O	7, 8	Downstream Port 1, USB differential data.
D+[2], D–[2]	I/O	9, 10	Downstream Port 2, USB differential data.
D+[3], D–[3]	I/O	23, 24	Downstream Port 3, USB differential data.
D+[4], D–[4]	I/O	21, 22	Downstream Port 4, USB differential data.
P0	I/O	P1[7:0] 11, 15, 12, 16, 13, 17, 14, 18	GPIO Port 0 capable of sinking 7 mA (typical).
P1	I/O	P1[2:0] 25, 27, 26	GPIO Port 1 capable of sinking 7 mA (typical).
$XTAL_{IN}$	IN	2	6-MHz crystal or external clock input.
$XTAL_{OUT}$	OUT	1	6-MHz crystal out.
V_{PP}		19	Programming voltage supply, tie to ground during normal operation.
V_{CC}		28	Voltage supply.
GND		4, 20	Ground.
V_{REF}	IN	3	External 3.3V supply voltage for the downstream differential data output buffers and the D+ pull-up.

Table 2. I/O Register Summary (continued)

Register Name	I/O Address	Read/Write	Function	Page
Hub Port Speed	0x4A	R/W	Hub Downstream Ports Speed	27
Hub Port Control (Ports [4:1])	0x4B	R/W	Hub Downstream Ports Control (Ports [4:1])	28
Hub Port Suspend	0x4D	R/W	Hub Downstream Port Suspend Control	30
Hub Port Resume Status	0x4E	R	Hub Downstream Ports Resume Status	30
Hub Ports SE0 Status	0x4F	R	Hub Downstream Ports SE0 Status	29
Hub Ports Data	0x50	R	Hub Downstream Ports Differential Data	29
Hub Downstream Force Low	0x51	R/W	Hub Downstream Ports Force LOW (Ports [1:4])	28
Processor Status & Control	0xFF	R/W	Microprocessor Status and Control Register	20

Instruction Set Summary

Refer to the *CYASM Assembler User's Guide* for more details. Note that conditional jump instructions (i.e., JC, JNC, JZ, JNZ) take five cycles if jump is taken, four cycles if no jump.

Table 3. Instruction Set Summary

MNEMONIC	operand	opcode	cycles	MNEMONIC	operand	opcode	cycles
HALT		00	7	NOP		20	4
ADD A,expr	data	01	4	INC A	acc	21	4
ADD A,[expr]	direct	02	6	INC X	x	22	4
ADD A,[X+expr]	index	03	7	INC [expr]	direct	23	7
ADC A,expr	data	04	4	INC [X+expr]	index	24	8
ADC A,[expr]	direct	05	6	DEC A	acc	25	4
ADC A,[X+expr]	index	06	7	DEC X	x	26	4
SUB A,expr	data	07	4	DEC [expr]	direct	27	7
SUB A,[expr]	direct	08	6	DEC [X+expr]	index	28	8
SUB A,[X+expr]	index	09	7	IORD expr	address	29	5
SBB A,expr	data	0A	4	IOWR expr	address	2A	5
SBB A,[expr]	direct	0B	6	POP A		2B	4
SBB A,[X+expr]	index	0C	7	POP X		2C	4
OR A,expr	data	0D	4	PUSH A		2D	5
OR A,[expr]	direct	0E	6	PUSH X		2E	5
OR A,[X+expr]	index	0F	7	SWAP A,X		2F	5
AND A,expr	data	10	4	SWAP A,DSP		30	5
AND A,[expr]	direct	11	6	MOV [expr],A	direct	31	5
AND A,[X+expr]	index	12	7	MOV [X+expr],A	index	32	6
XOR A,expr	data	13	4	OR [expr],A	direct	33	7
XOR A,[expr]	direct	14	6	OR [X+expr],A	index	34	8
XOR A,[X+expr]	index	15	7	AND [expr],A	direct	35	7
CMP A,expr	data	16	5	AND [X+expr],A	index	36	8
CMP A,[expr]	direct	17	7	XOR [expr],A	direct	37	7
CMP A,[X+expr]	index	18	8	XOR [X+expr],A	index	38	8
MOV A,expr	data	19	4	IOWX [X+expr]	index	39	6
MOV A,[expr]	direct	1A	5	CPL		3A	4
MOV A,[X+expr]	index	1B	6	ASL		3B	4

Table 3. Instruction Set Summary (continued)

MNEMONIC	operand	opcode	cycles		MNEMONIC	operand	opcode	cycles
MOV X,expr	data	1C	4		ASR		3C	4
MOV X,[expr]	direct	1D	5		RLC		3D	4
reserved		1E			RRC		3E	4
XPAGE		1F	4		RET		3F	8
MOV A,X		40	4		DI		70	4
MOV X,A		41	4		EI		72	4
MOV PSP,A		60	4		RETI		73	8
CALL	addr	50-5F	10		JC	addr	C0-CF	5 (or 4)
JMP	addr	80-8F	5		JNC	addr	D0-DF	5 (or 4)
CALL	addr	90-9F	10		JACC	addr	E0-EF	7
JZ	addr	A0-AF	5 (or 4)		INDEX	addr	F0-FF	14
JNZ	addr	B0-BF	5 (or 4)					

Programming Model

14-bit Program Counter

The 14-bit Program Counter (PC) allows access to up to 8 KB of PROM available with the CY7C65113C architecture. The top 32 bytes of the ROM in the 8K part are reserved for testing purposes. The program counter is cleared during reset, such that the first instruction executed after a reset is at address 0x0000h. Typically, this is a jump instruction to a reset handler that initializes the application (see Interrupt Vectors on page 23).

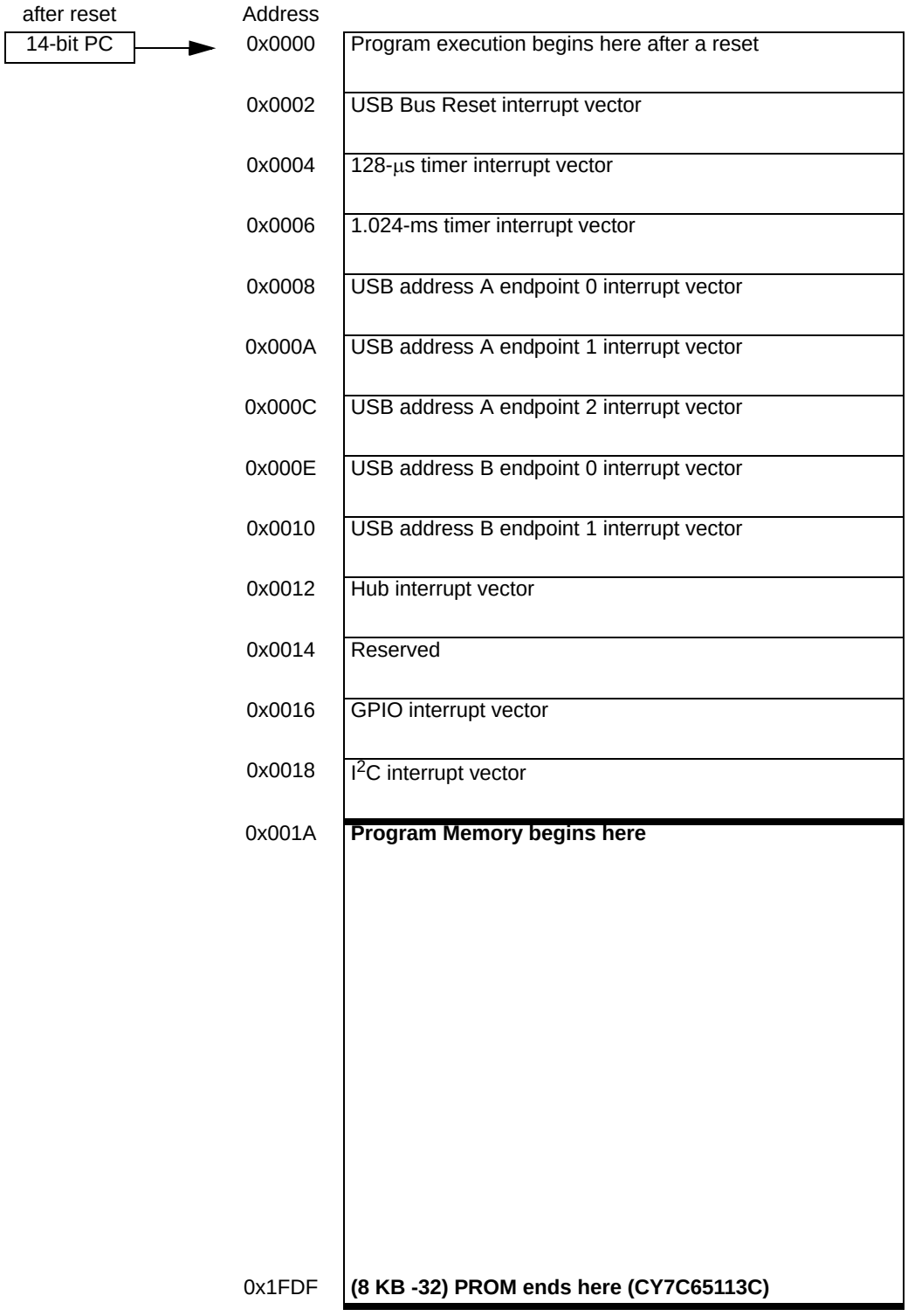
The lower eight bits of the program counter are incremented as instructions are loaded and executed. The upper six bits of the program counter are incremented by executing an XPAGE instruction. As a result, the last instruction executed within a 256-byte “page” of sequential code should be an XPAGE instruction. The assembler directive “XPAGEON” causes the assembler to insert XPAGE instructions automatically. Because instructions can be either one or two bytes long, the assembler may occasionally need to insert a NOP followed by an XPAGE to execute correctly.

The address of the next instruction to be executed, the carry flag, and the zero flag are saved as two bytes on the program stack during an interrupt acknowledge or a CALL instruction. The program counter, carry flag, and zero flag are restored from the program stack during a RETI instruction. Only the program counter is restored during a RET instruction.

The program counter cannot be accessed directly by the firmware. The program stack can be examined by reading SRAM from location 0x00 and up.

Program Memory Organization

Figure 2. Program Memory Space with Interrupt Vector Table



Note that the upper 32 bytes of the 8K PROM are reserved. Therefore, user's program must not overwrite this space.

8-bit Data Stack Pointer (DSP)

The Data Stack Pointer (DSP) supports PUSH and POP instructions that use the data stack for temporary storage. A PUSH instruction pre-decrements the DSP, then writes data to the memory location addressed by the DSP. A POP instruction reads data from the memory location addressed by the DSP, then post-increments the DSP.

During a reset, the DSP is reset to 0x00. A PUSH instruction when DSP equals 0x00 writes data at the top of the data RAM (address 0xFF). This writes data to the memory area reserved for USB endpoint FIFOs. Therefore, the DSP should be indexed at an appropriate memory location that does not compromise the Program Stack, user-defined memory (variables), or the USB endpoint FIFOs.

For USB applications, the firmware should set the DSP to an appropriate location to avoid a memory conflict with RAM dedicated to USB FIFOs. The memory requirements for the USB endpoints are described in Section 17.2. Example assembly instructions to do this with two device addresses (FIFOs begin at 0xD8) are shown below:

```
MOV A,20h     ; Move 20 hex into Accumulator (must be D8h or less)
SWAP A,DSP    ; swap accumulator value into DSP register.
```

Address Modes

The CY7C65113 microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

Data (Immediate)

“Data” address mode refers to a data operand that is actually a constant encoded in the instruction. As an example, consider the instruction that loads A with the constant 0xD8:

- MOV A, 0D8h.

This instruction requires two bytes of code where the first byte identifies the “MOV A” instruction with a data operand as the second byte. The second byte of the instruction is the constant “0xD8.” A constant may be referred to by name if a prior “EQU” statement assigns the constant value to the name. For example, the following code is equivalent to the example shown above:

- DSPINIT: EQU 0D8h
- MOV A, DSPINIT.

Direct

“Direct” address mode is used when the data operand is a variable stored in SRAM. In that case, the one byte address of the variable is encoded in the instruction. As an example, consider an instruction that loads A with the contents of memory address location 0x10:

- MOV A, [10h].

Normally, variable names are assigned to variable addresses using “EQU” statements to improve the readability of the assembler source code. As an example, the following code is equivalent to the example shown above:

- buttons: EQU 10h
- MOV A, [buttons].

Indexed

“Indexed” address mode allows the firmware to manipulate arrays of data stored in SRAM. The address of the data operand is the sum of a constant encoded in the instruction and the contents of the “X” register. Normally, the constant is the “base” address of an array of data and the X register contains an index that indicates which element of the array is actually addressed:

- array: EQU 10h
- MOV X, 3
- MOV A, [X+array].

This would have the effect of loading A with the fourth element of the SRAM “array” that begins at address 0x10. The fourth element would be at address 0x13.

Clocking

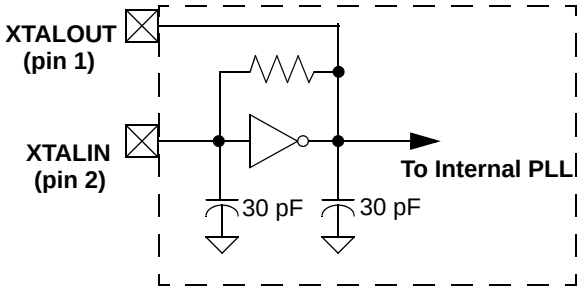


Figure 3. Clock Oscillator On-Chip Circuit

The XTALIN and XTALOUT are the clock pins to the microcontroller. The user can connect an external oscillator or a crystal to these pins. When using an external crystal, keep PCB traces between the chip leads and crystal as short as possible (less than 2 cm). A 6-MHz fundamental frequency parallel resonant crystal can be connected to these pins to provide a reference frequency for the internal PLL. The two internal 30-pF load caps appear in series to the external crystal and would be equivalent to a 15-pF load. Therefore, the crystal must have a required load capacitance of about 15–18 pF. A ceramic resonator does not allow the microcontroller to meet the timing specifications of full speed USB and therefore a ceramic resonator is not recommended with these parts.

An external 6-MHz clock can be applied to the XTALIN pin if the XTALOUT pin is left open. Grounding the XTALOUT pin when driving XTALIN with an oscillator does not work because the internal clock is effectively shorted to ground.

Reset

The CY7C65113C supports two resets: POR and WDR. Each of these resets causes:

- all registers to be restored to their default states
- the USB device addresses to be set to 0
- all interrupts to be disabled
- the PSP and DSP to be set to memory address 0x00.

The occurrence of a reset is recorded in the Processor Status and Control Register, as described in Section. Bits 4 and 6 are used to record the occurrence of POR and WDR respectively. Firmware can interrogate these bits to determine the cause of a reset.

Program execution starts at ROM address 0x0000 after a reset. Although this looks like interrupt vector 0, there is an important difference. Reset processing does NOT push the program counter, carry flag, and zero flag onto program stack. The firmware reset handler should configure the hardware before the “main” loop of code. Attempting to execute a RET or RETI in the firmware reset handler causes unpredictable execution results.

Power-on Reset

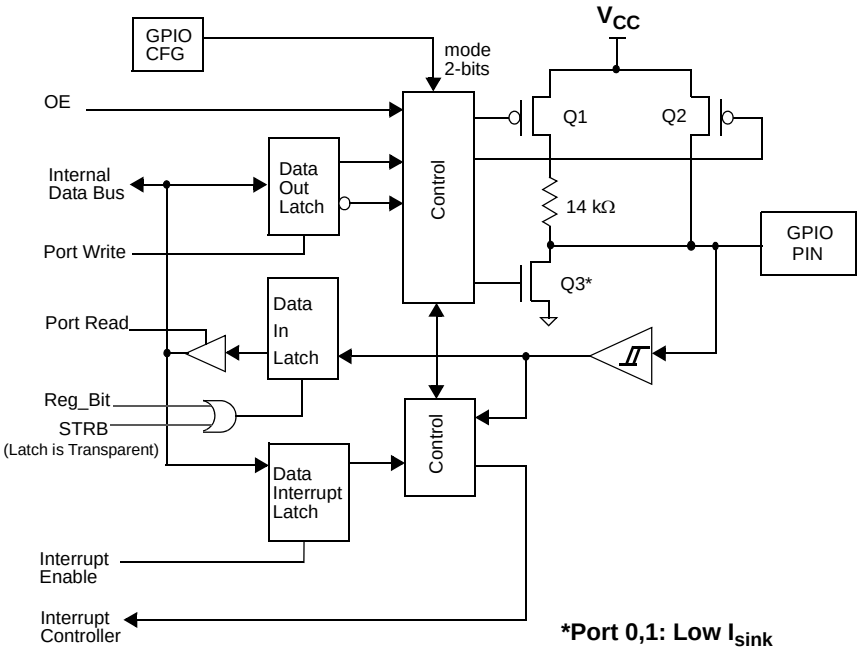
When V_{CC} is first applied to the chip, the POR signal is asserted and the CY7C65113C enters a “semi-suspend” state. During the semi-suspend state, which is different from the suspend state defined in the USB specification, the oscillator and all other blocks of the part are functional, except for the CPU. This semi-suspend time ensures that both a valid V_{CC} level is reached and that the internal PLL has time to stabilize before full operation begins. When the V_{CC} has risen above approximately 2.5V, and the oscillator is stable, the POR is deasserted and the on-chip timer starts counting. The first 1 ms of suspend time is not interruptible, and the semi-suspend state continues for an additional 95 ms unless the count is bypassed by a USB Bus Reset on the upstream port. The 95 ms provides time for V_{CC} to stabilize at a valid operating voltage before the chip executes code.

If a USB Bus Reset occurs on the upstream port during the 95 ms semi-suspend time, the semi-suspend state is aborted and program execution begins immediately from address 0x0000. In this case, the Bus Reset interrupt is pending but not serviced until firmware sets the USB Bus Reset Interrupt Enable bit (Bit 0, *Figure 18*) and enables interrupts with the EI command.

The POR signal is asserted whenever V_{CC} drops below approximately 2.5V, and remains asserted until V_{CC} rises above this level again. Behavior is the same as described above.

General-purpose I/O Ports

Figure 5. Block Diagram of a GPIO Pin



There are 11 GPIO pins (P0[7:0] and P1[2:0]) for the hardware interface. Each port can be configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs. The data for each GPIO port is accessible through the data registers. Port data registers are shown in *Figure 6* through Figure , and are set to 1 on reset.

Figure 6. Port 0 Data.

Port 0 Data **Address 0x00**

Bit #	7	6	5	4	3	2	1	0
Bit Name	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Figure 7. Port1 Data

Port 1 Data **Address 0x01**

Bit #	-	-	-	-	-	2	1	0
Bit Name	-	-	-	-	-	P1.2	P1.1	P1.0
Read/Write	-	-	-	-	-	R/W	R/W	R/W
Reset	-	-	-	-	-	1	1	1

Special care should be taken with any unused GPIO data bits. An unused GPIO data bit, either a pin on the chip or a port bit that is not bonded on a particular package, must not be left floating when the device enters the suspend state. If a GPIO data bit is left floating, the leakage current caused by the floating bit may violate the suspend current limitation specified by the USB Specifications. If a '1' is written to the unused data bit and the port is configured with open drain outputs, the unused data bit remains in an indeterminate state. Therefore, if an unused port bit is programmed in open-drain mode, it must be written with a '0.'

A read from a GPIO port always returns the present state of the voltage at the pin, independent of the settings in the Port Data Registers. During reset, all of the GPIO pins are set to a high-impedance input state. Writing a ‘0’ to a GPIO pin drives the pin LOW. In this state, a ‘0’ is always read on that GPIO pin unless an external source overdrives the internal pull-down device.

GPIO Configuration Port

Every GPIO port can be programmed as inputs with internal pull-ups, outputs LOW or HIGH, or Hi-Z (floating, the pin is not driven internally). In addition, the interrupt polarity for each port can be programmed. The Port Configuration bits (Figure) and the Interrupt Enable bit (Figure 10 through Figure 10) determine the interrupt polarity of the port pins

Figure 8. GPIO Configuration Register.

GPIO Configuration **Address 0x08**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Port 1 Config Bit 1	Port 1 Config Bit 0	Port 0 Config Bit 1	Port 0 Config Bit 0
Read/Write	-	-	-	-	R/W	R/W	R/W	R/W
Reset	-	-	-	-	0	0	0	0

As shown in Table 4 below, a positive polarity on an input pin represents a rising edge interrupt (LOW to HIGH), and a negative polarity on an input pin represents a falling edge interrupt (HIGH to LOW).

The GPIO interrupt is generated when all of the following conditions are met: the Interrupt Enable bit of the associated Port Interrupt Enable Register is enabled, the GPIO Interrupt Enable bit of the Global Interrupt Enable Register (*Figure 18*) is enabled, the Interrupt Enable Sense (bit 2, *Figure 17*) is set, and the GPIO pin of the port sees an event matching the interrupt polarity.

The driving state of each GPIO pin is determined by the value written to the pin’s Data Register (*Figure 6* through Figure) and by its associated Port Configuration bits as shown in the GPIO Configuration Register (*Figure*). These ports are configured on a per-port basis, so all pins in a given port are configured together. The possible port configurations are detailed in Table 4. As shown in this table below, when a GPIO port is configured with CMOS outputs, interrupts from that port are disabled.

During reset, all of the bits in the GPIO Configuration Register are written with ‘0’ to select Hi-Z mode for all GPIO ports as the default configuration.

Bits [7..0]: I²C Data

Contains the 8-bit data on the I²C Bus

Figure 16. I²C Status and Control Register.

I²C Status and Control **Address 0x28**

Bit #	7	6	5	4	3	2	1	0
Bit Name	MSTR Mode	Continue/Busy	Xmit Mode	ACK	Addr	ARB Lost/Restart	Received Stop	I²C Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

The I²C Status and Control register bits are defined in Table 6, with a more detailed description following.

Table 6. I²C Status and Control Register Bit Definitions

Bit	Name	Description
0	I²C Enable	When set to '1', the I²C-compatible function is enabled. When cleared, I²C GPIO pins operate normally.
1	Received Stop	Reads 1 only in slave receive mode, when I²C Stop bit detected (unless firmware did not ACK the last transaction).
2	ARB Lost/Restart	Reads 1 to indicate master has lost arbitration. Reads 0 otherwise. Write to 1 in master mode to perform a restart sequence (also set Continue bit).
3	Addr	Reads 1 during first byte after start/restart in slave mode, or if master loses arbitration. Reads 0 otherwise. This bit should always be written as 0.
4	ACK	In receive mode, write 1 to generate ACK, 0 for no ACK. In transmit mode, reads 1 if ACK was received, 0 if no ACK received.
5	Xmit Mode	Write to 1 for transmit mode, 0 for receive mode.
6	Continue/Busy	Write 1 to indicate ready for next transaction. Reads 1 when I²C-compatible block is busy with a transaction, 0 when transaction is complete.
7	MSTR Mode	Write to 1 for master mode, 0 for slave mode. This bit is cleared if master loses arbitration. Clearing from 1 to 0 generates Stop bit.

Bit 7: MSTR Mode

Setting this bit to 1 causes the I²C-compatible block to initiate a master mode transaction by sending a start bit and transmitting the first data byte from the data register (this typically holds the target address and R/W bit). Subsequent bytes are initiated by setting the Continue bit, as described below.

Clearing this bit (set to 0) causes the GPIO pins to operate normally.

In master mode, the I²C-compatible block generates the clock (SCK), and drives the data line as required depending on transmit or receive state. The I²C-compatible block performs any required arbitration and clock synchronization. IN the event of a loss of arbitration, this MSTR bit is cleared, the ARB Lost bit is set, and an interrupt is generated by the microcontroller. If the chip is the target of an external master that wins arbitration, then the interrupt is held off until the transaction from the external master is completed.

When MSTR Mode is cleared from 1 to 0 by a firmware write, an I²C Stop bit is generated.

Bit 6: Continue/Busy

This bit is written by the firmware to indicate that the firmware is ready for the next byte transaction to begin. In other words, the bit has responded to an interrupt request and has completed the required update or read of the data register. During a read this bit indicates if the hardware is busy and is locking out additional writes to the I²C Status and Control register. This locking allows the hardware to complete certain operations that may require an extended period of time. Following an I²C interrupt, the I²C-compatible block does not return to the Busy state until firmware sets the Continue bit. This allows the firmware to make one control register write without the need to check the Busy bit.

Bit 5: Xmit Mode

This bit is set by firmware to enter transmit mode and perform a data transmit in master or slave mode. Clearing this bit sets the part in receive mode. Firmware generally determines the value of this bit from the R/W bit associated with the I²C address packet. The Xmit Mode bit state is ignored when initially writing the MSTR Mode or the Re-

start bits, as these cases always cause transmit mode for the first byte.

Bit 4: ACK

This bit is set or cleared by firmware during receive operation to indicate if the hardware should generate an ACK signal on the I^2C-compatible bus. Writing a 1 to this bit generates an ACK (SDA LOW) on the I2C-compatible bus at the ACK bit time. During transmits (Xmit Mode = 1), this bit should be cleared.

Bit 3: Addr

This bit is set by the I^2C-compatible block during the first byte of a slave receive transaction, after an I^2C start or restart. The Addr bit is cleared when the firmware sets the Continue bit. This bit allows the firmware to recognize when the master has lost arbitration, and in slave mode it allows the firmware to recognize that a start or restart has occurred.

Bit 2: ARB Lost/Restart

This bit is valid as a status bit (ARB Lost) after master mode transactions. In master mode, set this bit (along with the Continue and MSTR Mode bits) to perform an I^2C restart sequence. The I^2C target address for the restart must be written to the data register before setting the Continue bit. To prevent false ARB Lost signals, the Restart bit is cleared by hardware during the restart sequence.

Bit 1: Receive Stop

This bit is set when the slave is in receive mode and detects a stop bit on the bus. The Receive Stop bit is not set if the firmware terminates the I^2C transaction by not acknowledging the previous byte transmitted on the I^2C-compatible bus, e.g., in receive mode if firmware sets the Continue bit and clears the ACK bit.

Bit 0: I^2C Enable

Set this bit to override GPIO definition with I^2C-compatible function on the two I^2C-compatible pins. When this bit is cleared, these pins are free to function as GPIOs. In I^2C-compatible mode, the two pins operate in open drain mode, independent of the GPIO configuration setting.

Processor Status and Control Register

Figure 17. Processor Status and Control Register

Processor Status and Control **Address 0xFF**

Bit #	7	6	5	4	3	2	1	0
Bit Name	IRQ Pending	Watchdog Reset	USB Bus Reset Interrupt	Power-on Reset	Suspend	Interrupt Enable Sense	Reserved	Run
Read/Write	R	R/W	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	1	0	0	0	1

Bit 0: Run

This bit is manipulated by the HALT instruction. When Halt is executed, all the bits of the Processor Status and Control Register are cleared to 0. Since the run bit is cleared, the processor stops at the end of the current instruction. The processor remains halted until an appropriate reset occurs (power-on or Watchdog). This bit should normally be written as a '1.'

Bit 1: Reserved

Bit 1 is reserved and must be written as a zero.

Bit 2: Interrupt Enable Sense

This bit indicates whether interrupts are enabled or disabled. Firmware has no direct control over this bit as writing a zero or one to this bit position has no effect on interrupts. A '0' indicates that interrupts are masked off and a '1' indicates that the interrupts are enabled. This bit is further gated with the bit settings of the Global Interrupt Enable Register (*Figure 18*) and USB End Point Interrupt Enable Register (*Figure 19*). Instructions DI, EI, and RETI manipulate the state of this bit.

Bit 3: Suspend

Writing a '1' to the Suspend bit halts the processor and cause the microcontroller to enter the suspend mode that significantly reduces power consumption. A pending, enabled interrupt or USB bus activity causes the device to come out of suspend. After coming out of suspend, the device resumes firmware execution at the instruction following the IOWR which put the part into suspend. An IOWR attempting to put the part into suspend is ignored if USB bus activity is present. See Section for more details on suspend mode operation.

Bit 4: Power-on Reset

The Power-on Reset is set to '1' during a power-on reset. The firmware can check bits 4 and 6 in the reset handler to determine whether a reset was caused by a power-on condition or a Watchdog timeout. A POR event may be followed by a Watchdog reset before firmware begins executing, as explained below.

Bit 5: USB Bus Reset Interrupt

The USB Bus Reset Interrupt bit is set when the USB Bus Reset is detected on receiving a USB Bus Reset signal on the upstream port. The USB Bus Reset signal is a single-ended zero (SE0) that lasts from 12 to 16 μs. An SE0

is defined as the condition in which both the D+ line and the D– line are LOW at the same time.

Bit 6: Watchdog Reset

The Watchdog Reset is set during a reset initiated by the Watchdog Timer. This indicates the Watchdog Timer went for more than t_{WATCH} (8 ms minimum) between Watchdog clears. This can occur with a POR event, as noted below.

Bit 7: IRQ Pending

The IRQ pending, when set, indicates that one or more of the interrupts has been recognized as active. An interrupt remains pending until its interrupt enable bit is set (*Figure 18*, *Figure 19*) and interrupts are globally enabled. At that point, the internal interrupt handling sequence clears this bit until another interrupt is detected as pending.

During power-up, the Processor Status and Control Register is set to 00010001, which indicates a POR (bit 4 set) has occurred and no interrupts are pending (bit 7 clear). During the 96-ms suspend at start-up (explained in Section), a Watchdog Reset also occurs unless this suspend is aborted by an upstream SE0 before 8 ms. If a WDR occurs during the power-up suspend interval, firmware reads 01010001 from the Status and Control Register after power-up. Normally, the POR bit should be cleared so a subsequent WDR can be clearly identified. If an upstream bus reset is received before firmware examines this register, the Bus Reset bit may also be set.

During a Watchdog Reset, the Processor Status and Control Register is set to 01XX0001, which indicates a Watchdog Reset (bit 6 set) has occurred and no interrupts are pending (bit 7 clear). The Watchdog Reset does not effect the state of the POR and the Bus Reset Interrupt bits.

Interrupts

Interrupts are generated by GPIO pins, internal timers, I^2C-compatible operation, internal USB hub and USB traffic conditions. All interrupts are maskable by the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register. Writing a ‘1’ to a bit position enables the interrupt associated with that bit position.

Figure 18. Global Interrupt Enable Register

Global Interrupt Enable Register **Address 0X20**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	I^2C Interrupt Enable	GPIO Interrupt Enable	Reserved	USB Hub Interrupt Enable	1.024-ms Interrupt Enable	128-µs Interrupt Enable	USB Bus RST Interrupt Enable
Read/Write	–	R/W	R/W	-	R/W	R/W	R/W	R/W
Reset	–	0	0	X	0	0	0	0

Bit 0: USB Bus RST Interrupt Enable

1 = Enable Interrupt on a USB Bus Reset; 0 = Disable interrupt on a USB Bus Reset (Refer to section).

Bit 1:128-µs Interrupt Enable

1 = Enable Timer interrupt every 128 µs; 0 = Disable Timer Interrupt for every 128 µs.

Bit 2: 1.024-ms Interrupt Enable

1 = Enable Timer interrupt every 1.024 ms; 0 = Disable Timer Interrupt every 1.024 ms.

Bit 3: USB Hub Interrupt Enable

1 = Enable Interrupt on a Hub status change; 0 = Disable interrupt due to hub status change. (Refer to section .)

Bit 4: Reserved.

Bit 5: GPIO Interrupt Enable

1 = Enable Interrupt on falling/rising edge on any GPIO; 0 = Disable Interrupt on falling/rising edge on any GPIO (Refer to section , and .).

Bit 6: I^2C Interrupt Enable

1 = Enable Interrupt on I2C related activity; 0 = Disable I2C related activity interrupt. (Refer to section .)

Bit 7: Reserved

Figure 19. USB Endpoint Interrupt Enable Register.

USB Endpoint Interrupt Enable **Address 0X21**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	EPB1 Interrupt Enable	EPB0 Interrupt Enable	EPA2 Interrupt Enable	EPA1 Interrupt Enable	EPA0 Interrupt Enable
Read/Write	–	–	–	R/W	R/W	R/W	R/W	R/W
Reset	–	–	–	0	0	0	0	0

Bit 0: EPA0 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint A0;
0 = Disable Interrupt on data activity through endpoint A0

Bit 1: EPA1 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint A1;
0 = Disable Interrupt on data activity through endpoint A1

Bit 2: EPA2 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint A2;
0 = Disable Interrupt on data activity through endpoint A2.

Bit 3: EPB0 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint B0;
0 = Disable Interrupt on data activity through endpoint B0

Bit 4: EPB1 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint B1;
0 = Disable Interrupt on data activity through endpoint B1

Bit [7..5]: Reserved

During a reset, the contents of the Global Interrupt Enable Register and USB End Point Interrupt Enable Register are cleared, effectively disabling all interrupts,

The interrupt controller contains a separate flip-flop for each interrupt. See Figure 20 for the logic block diagram of the interrupt controller. When an interrupt is generated, it is first registered as a pending interrupt. It stays pending until it is serviced or a reset occurs. A pending interrupt only generates an interrupt request if it is enabled by the corresponding bit in the interrupt enable registers. The highest priority interrupt request is serviced following the completion of the currently executing instruction.

When servicing an interrupt, the hardware does the following:

1. Disables all interrupts by clearing the Global Interrupt Enable bit in the CPU (the state of this bit can be read at Bit 2 of the Processor Status and Control Register, *Figure 17*).
2. Clears the flip-flop of the current interrupt.
3. Generates an automatic CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector, see Section).

The instruction in the interrupt table is typically a JMP instruction to the address of the Interrupt Service Routine (ISR). The user can reenable interrupts in the interrupt service routine by executing an EI instruction. Interrupts can be nested to a level limited only by the available stack space.

The Program Counter value as well as the Carry and Zero flags (CF, ZF) are stored onto the Program Stack by the automatic CALL instruction generated as part of the interrupt acknowledge process. The user firmware is responsible for ensuring that the processor state is preserved and restored during an interrupt. The PUSH A instruction should typically be used as the first command in the ISR to save the accumulator value and the POP A instruction should be used to restore the accumulator value just before the RETI instruction. The program counters CF and ZF are restored and interrupts are enabled when the RETI instruction is executed.

The IDI and EI instruction can be used to disable and enable interrupts, respectively. These instruction affect only the Global Interrupt Enable bit of the CPU. If desired, EI can be used to re-enable interrupts while inside an ISR, instead of waiting for the RETI that exits the ISR. While the global interrupt enable bit is cleared, the presence of a pending interrupt can be detected by examining the IRQ Sense bit (Bit 7 in the Processor Status and Control Register).

USB Overview

The USB hardware includes a USB Hub repeater with one upstream and up to seven downstream ports. The USB Hub repeater interfaces to the microcontroller through a full-speed serial interface engine (SIE). An external series resistor of R_{ext} must be placed in series with all upstream and downstream USB outputs in order to meet the USB driver requirements of the USB specification. The CY7C65113C microcontroller can provide the functionality of a compound device consisting of a USB hub and permanently attached functions.

USB Serial Interface Engine (SIE)

The SIE allows the CY7C65113C microcontroller to communicate with the USB host through the USB repeater portion of the hub. The SIE simplifies the interface between the microcontroller and USB by incorporating hardware that handles the following USB bus activity independently of the microcontroller:

- Bit stuffing/unstuffing
- Checksum generation/checking
- ACK/NAK/STALL
- Token type identification
- Address checking.

Firmware is required to handle the following USB interface tasks:

- Coordinate enumeration by responding to SETUP packets
- Fill and empty the FIFOs
- Suspend/Resume coordination
- Verify and select DATA toggle values.

USB Enumeration

The internal hub and any compound device function are enumerated under firmware control. The hub is enumerated first, followed by any integrated compound function. After the hub is enumerated, the USB host can read hub connection status to determine which (if any) of the downstream ports need to be enumerated. The following is a brief summary of the typical enumeration process of the CY7C65113C by the USB host. For a detailed description of the enumeration process, refer to the USB specification.

In this description, ‘Firmware’ refers to embedded firmware in the CY7C65113C controller.

1. The host computer sends a SETUP packet followed by a DATA packet to USB address 0 requesting the Device descriptor.
2. Firmware decodes the request and retrieves its Device descriptor from the program memory tables.
3. The host computer performs a control read sequence and Firmware responds by sending the Device descriptor over the USB bus, via the on-chip FIFOs.
4. After receiving the descriptor, the host sends a SETUP packet followed by a DATA packet to address 0 assigning a new USB address to the device.
5. Firmware stores the new address in its USB Device Address Register (for example, as Address B) after the no-data control sequence completes.
6. The host sends a request for the Device descriptor using the new USB address.
7. Firmware decodes the request and retrieves the Device descriptor from program memory tables.
8. The host performs a control read sequence and Firmware responds by sending its Device descriptor over the USB bus.
9. The host generates control reads from the device to request the Configuration and Report descriptors.
10. Once the device receives a Set Configuration request, its functions may now be used.
11. Following enumeration as a hub, Firmware can optionally indicate to the host that a compound device exists (for example, the keyboard in a keyboard/hub device).
12. The host carries out the enumeration process with this additional function as though it were attached downstream from the hub.
13. When the host assigns an address to this device, it is stored as the other USB address (for example, Address A).

USB Hub

A USB hub is required to support:

- Connectivity behavior: service connect/disconnect detection
- Bus fault detection and recovery
- Full-/Low-speed device support

These features are mapped onto a hub repeater and a hub controller. The hub controller is supported by the processor integrated into the CY7C65113C microcontroller. The hardware in the hub repeater detects whether a USB device is connected to a downstream port. The connection to a downstream port is through a differential signal pair (D+ and D–). Each downstream port provided by the hub requires external R_{UDN} resistors from each signal line to ground, so that when a downstream port has no device connected, the hub reads a LOW (zero) on both D+ and D–. This condition is used to identify the “no connect” state.

The hub must have a resistor R_{UUP} connected between its upstream D+ line and V_{REG} to indicate it is a full speed USB device.

The hub generates an EOP at EOF1, in accordance with the USB 1.1 Specification (section 11.2.2, page 234) as well as USB 2.0 specification (section 11.2.5, page 304).

Connecting/Disconnecting a USB Device

A low-speed (1.5 Mbps) USB device has a pull-up resistor on the D– pin. At connect time, the bias resistors set the signal levels on the D+ and D– lines. When a low-speed device is connected to a hub port, the hub sees a LOW on D+ and a HIGH on D–. This causes the hub repeater to set a connect bit in the Hub Ports Connect Status register for the downstream port (see *Figure 22*). Then the hub repeater generates a Hub Interrupt to notify the microcontroller that there has been a change in the Hub downstream status. The firmware sets the speed of this port in the Hub Ports Speed Register (see *Figure*).

A full-speed (12 Mbps) USB device has a pull-up resistor from the D+ pin, so the hub sees a HIGH on D+ and a LOW on D–. In this case, the hub repeater sets a connect bit in the Hub Ports Connect Status register and generates a Hub Interrupt to notify the microcontroller of the change in Hub status. The firmware sets the speed of this port in the Hub Ports Speed Register (see *Figure*)

Connects are recorded by the time a non-SE0 state lasts for more than 2.5 μs on a downstream port.

When a USB device is disconnected from the Hub, the downstream signal pair eventually floats to a single-ended zero state. The hub repeater recognizes a disconnect once the SE0 state on a downstream port lasts from 2.0 to 2.5 μs. On a disconnect, the corresponding bit in the Hub Ports Connect Status register is cleared, and the Hub Interrupt is generated.

Figure 22. Hub Ports Connect Status

Hub Ports Connect Status **Address 0x48**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Port 4 Connect Status	Port 3 Connect Status	Port 2 Connect Status	Port 1 Connect Status
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit [0..3]: Port x Connect Status (where x = 1..4).

When set to 1, Port x is connected; When set to 0, Port x is disconnected.

Bit [4..7]: Reserved.

Set to 0.

The Hub Ports Connect Status register is cleared to zero by reset or USB bus reset, then set to match the hardware configuration by the hub repeater hardware. The Reserved bits [4..7] should always read as '0' to indicate no connection.

Figure 23. Hub Ports Speed

Hub Ports Speed **Address 0x4A**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Port 4 Speed	Port 3 Speed	Port 2 Speed	Port 1 Speed
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit [0..3]: Port x Speed (where x = 1..4).

Set to 1 if the device plugged in to Port x is Low Speed; Set to 0 if the device plugged in to Port x is Full Speed.

Bit [4..7]: Reserved.

Set to 0.

The Hub Ports Speed register is cleared to zero by reset or bus reset. This must be set by the firmware on issuing a port reset. The Reserved bits [4..7] should always read as '0.'

Enabling/Disabling a USB Device

After a USB device connection has been detected, firmware must update status change bits in the hub status change data structure that is polled periodically by the USB host. The host responds by sending a packet that instructs the hub to reset and enable the downstream port. Firmware then sets the bit in the Hub Ports Enable register (*Figure 24*), for the downstream port. The hub repeater hardware responds to an enable bit in the Hub Ports Enable register (*Figure 24*) by enabling the downstream port, so that USB traffic can flow to and from that port.

If a port is marked enabled and is not suspended, it receives all USB traffic from the upstream port, and USB traffic from the downstream port is passed to the upstream port (unless babble is detected). Low-speed ports do not receive full-speed traffic from the upstream port.

When firmware writes to the Hub Ports Enable register (*Figure 24*) to enable a port, the port is not enabled until the end of any packet currently being transmitted. If there is no USB traffic, the port is enabled immediately.

When a USB device disconnection has been detected, firmware must update status bits in the hub change status data structure that is polled periodically by the USB host. In suspended mode, a connect or disconnect event generates an interrupt (if the hub interrupt is enabled) even if the port is disabled.

Figure 24. Hub Ports Enable Register

Hub Ports Enable Register **Address 0x49**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Port 4 Enable	Port 3 Enable	Port 2 Enable	Port 1 Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 29. Hub Ports Suspend Register

Hub Ports Suspend **Address 0x4D**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Device Remote Wakeup	Reserved	Reserved	Reserved	Port 4 Selective Suspend	Port 3 Selective Suspend	Port 2 Selective Suspend	Port 1 Selective Suspend
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit [0..3]: Port x Selective Suspend (where x = 1..4).

Set to 1 if Port x is Selectively Suspended; Set to 0 if Port x Do not suspend.

Bit 7: Device Remote Wakeup.

When set to 1, Enable hardware upstream resume signaling for connect/disconnect events during global resume.

When set to 0, Disable hardware upstream resume signaling for connect/disconnect events during global resume.

Figure 30. Hub Ports Resume Status Register

Hub Ports Resume **Address 0x4E**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Resume 4	Resume 3	Resume 2	Resume 1
Read/Write	-	-	-	-	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit [0..3] - Resume x (where x = 1..4).

When set to 1 Port x requesting to be resumed (set by hardware); default state is 0.

Bit [4..7]: Reserved.

Set to 0.

Resume from a selectively suspended port, with the hub not in suspend, typically involves the following actions:

1. Hardware detects the Resume, drives a K to the port, and generates the hub interrupt. The corresponding bit in the Resume Status Register (0x4E) reads ‘1’ in this case.
2. Firmware responds to hub interrupt, and reads register 0x4E to determine the source of the Resume.
3. Firmware begins driving K on the port for 10 ms or more through register 0x4B.
4. Firmware clears the Selective Suspend bit for the port (0x4D), which clears the Resume bit (0x4E). This ends the hardware-driven Resume, but the firmware-driven Resume continues. To prevent traffic being fed by the hub repeater to the port during or just after the Resume, firmware should disable this port.
5. Firmware drives a timed SE0 on the port for two low-speed bit times as appropriate. Firmware must disable interrupts during this SE0 so the SE0 pulse isn’t inadvertently lengthened, and appear as a bus reset to the downstream device.
6. Firmware drives a J on the port for one low-speed bit time, then it idles the port.
7. Firmware re-enables the port.

Resume when the hub is suspended typically involves these actions:

1. Hardware detects the Resume, drives a K on the upstream (which is then reflected to all downstream enabled ports), and generates the hub interrupt.
2. The part comes out of suspend and the clocks start.
3. Once the clocks are stable, firmware execution resumes. An internal counter ensures that this takes at least 1 ms. Firmware should check for Resume from any selectively suspended ports. If found, the Selective Suspend bit for the port should be cleared; no other action is necessary.
4. The Resume ends when the host stops sending K from upstream. Firmware should check for changes to the Enable and Connect Registers. If a port has become disabled but is still connected, an SE0 has been detected on the port. The port should be treated as having been reset, and should be reported to the host as newly connected.

Firmware can choose to clear the Device Remote Wake-up bit (if set) to implement firmware timed states for port changes. All allowed port changes wake the part. Then, the part can use internal timing to determine whether to take action or return to suspend. If Device Remote Wake-up is set, automatic hardware assertions take place on Resume events.

USB Upstream Port Status and Control

USB status and control is regulated by the USB Status and Control Register, as shown in Figure . All bits in the register are cleared during reset.

Electrical Characteristics

f_{OSC} = 6 MHz; Operating Temperature = 0 to 70°C, V_{CC} = 4.0V to 5.25V

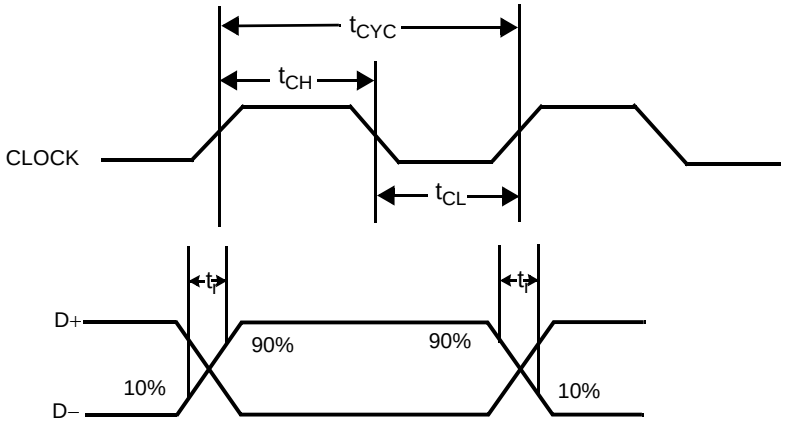
Parameter	Description	Conditions	Min.	Max.	Unit
		General			
V_{REF}	Reference Voltage	3.3V ±5%	3.15	3.45	V
V_{pp}	Programming Voltage (disabled)		–0.4	0.4	V
I_{CC}	V_{CC} Operating Current	No GPIO source current		50	mA
I_{SB1}	Supply Current—Suspend Mode			50	µA
I_{ref}	V_{REF} Operating Current	No USB Traffic[8]		10	mA
I_{il}	Input Leakage Current	Any pin		1	µA
		USB Interface			
V_{di}	Differential Input Sensitivity	| (D+)–(D–) |	0.2		V
V_{cm}	Differential Input Common Mode Range		0.8	2.5	V
V_{se}	Single Ended Receiver Threshold		0.8	2.0	V
C_{in}	Transceiver Capacitance			20	pF
I_{lo}	Hi-Z State Data Line Leakage	0V < V_{in} < 3.3V	–10	10	µA
R_{ext}	External USB Series Resistor	In series with each USB pin	19	21	Ω
R_{UUP}	External Upstream USB Pull-up Resistor	1.5 kΩ ±5%, D+ to V_{REG}	1.425	1.575	kΩ
R_{UDN}	External Downstream Pull-down Resistors	15 kΩ ±5%, downstream USB pins	14.25	15.75	kΩ
		Power-on Reset			
t_{vccs}	V_{CC} Ramp Rate	Linear ramp 0V to V_{CC}[9]	0	100	ms
		USB Upstream/Downstream Port			
V_{UOH}	Static Output High	15 kΩ ±5% to Gnd	2.8	3.6	V
V_{UOL}	Static Output Low	1.5 kΩ ±5% to V_{REF}		0.3	V
Z_O	USB Driver Output Impedance	Including R_{ext} Resistor	28	44	Ω
		General Purpose I/O (GPIO)			
R_{up}	Pull-up Resistance (typical 14 kΩ)		8.0	24.0	kΩ
V_{ITH}	Input Threshold Voltage	All ports, low-to-high edge	20%	40%	V_{CC}
V_H	Input Hysteresis Voltage	All ports, high-to-low edge	2%	8%	V_{CC}
V_{OL}	Port 0,1 Output Low Voltage	I_{OL} = 3 mA I_{OL} = 8 mA		0.4 2.0	V V
V_{OH}	Output High Voltage	I_{OH} = 1.9 mA (all ports 0,1)	2.4		V

Notes
8. Add 18 mA per driven USB cable (upstream or downstream. This is based on transitions every 2 full-speed bit times on average.
9. Power-on Reset occurs whenever the voltage on V_{CC} is below approximately 2.5V.

Switching Characteristics (f_{OSC} = 6.0 MHz)

Parameter	Description	Min.	Max.	Unit
Clock Source				
f_{OSC}	Clock Rate	6 ±0.25%		MHz
t_{cyc}	Clock Period	166.25	167.08	ns
t_{CH}	Clock HIGH time	0.45 t_{CYC}		ns
t_{CL}	Clock LOW time	0.45 t_{CYC}		ns
USB Full-speed Signaling[10]				
t_{rfs}	Transition Rise Time	4	20	ns
t_{ffs}	Transition Fall Time	4	20	ns
t_{rfmfs}	Rise/Fall Time Matching; (t_r/t_f)	90	111	%
$t_{dratefs}$	Full Speed Date Rate	12 ±0.25%		Mb/s
Timer Signals				
t_{watch}	Watchdog Timer Period	8.192	14.336	ms

Note
10. Per Table 7-6 of revision 1.1 of USB specification.



Acronyms

Acronym	Description
CMOS	complementary metal oxide semiconductor
CPU	central processing unit
DSP	data stack pointer
EMI	electro magnetic interference
GPIO	general purpose I/O
HID	human interface device
I2C	inter integrated circuit
LSB	least-significant byte
MSB	most-significant byte
PC	program counter
PLL	phase-locked loop
POR	power on reset
PROM	precision power on reset
PSP	program stack pointer
RAM	random access memory
SIE	serial interface engine
SOIC	small outlined integrated circuit
SRAM	standard random access memory
USB	universal serial bus
WDT	watchdog timer

Document Conventions

Units of Measure

Convention	Description
DC	Direct current
KB	1024 bytes
Kbit	1024 bits
kHz	kilohertz
kΩ	kilohm
mA	milli-ampere
Mbps	megabits per second
ms	milli seconds
pF	picofarad
µs	microsecond
V	volts

Document History Page

Document Title: CY7C65113C USB Hub with Microcontroller Document Number: 38-08002				
REV.	**ECN NO.**	**Issue Date**	**Orig. of Change**	**Description of Change**
**	109965	02/22/02	SZV	Change from Spec number: 38-00590 to 38-08002
*A	120372	12/17/02	MON	Added register bit definitions. Added default bit state of each register. Corrected the Schematic (location of the pull-up on D+). Corrected the Logical Diagram (removed the extra GPIO Port 1). Added register summary. Modified Figure 17, more labeling. Removed information on the availability of the part in PDIP package. Modified Table 11 and provided more explanation regarding locking/unlocking mechanism of the mode register. Removed any information regarding the speed detect bit in Hub Port Speed register being set by hardware.
*B	124522	03/13/03	MON	Fixed the figure on page 42 regarding the update of mode registers. The arrows in the figure were misplaced and the figure was unreadable. This is an important figure for understanding mode register functioning.
*C	368601	See ECN	BHA	Added Lead-free Package Information. Removed CY7C65013 Information. Updated Package Drawing.
*D	429098	See ECN	TYJ	Part numbers changed to ‘C’ types Cypress Perform logo added Part numbers updated in the ordering section
*E	3057657	10/13/10	AJHA	Added “Not recommended for new designs” watermark in the PDF. Updated package diagrams. Updated template.
*F	3207401	03/28/2011	ODC	Added Ordering Code Definitions, Acronyms, and Document Conventions. Updated package diagram.
*G	4313900	03/21/2014	AKSL	Removed "Not recommended for new designs" watermark. Updated package diagram to current revision.