CY7C64211/12/13
CY7C64311/12/13
Full-Speed USB (12 Mbps) Microcontroller

Cypress Semiconductor Corporation • 3901 North First Street • San Jose • CA 95134 • 408-943-2600

November 11, 1998

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF TABLES

1.0 Features

- Low-cost solution for high-speed USB peripheral applications (phones, speakers, microphones, printers, modems, hand-held scanners, serial devices, etc.)
- USB Specification Compliance
  - Conforms to USB Specification, Version 1.0
  - Supports 1 device address and 9 endpoints

    One control endpoint & 8 data endpoints (interrupt, bulk, and isochronous)

    Software configurable endpoints. Each data endpoint can be configured as interrupt, bulk, or isochronous with sizes ranging from 128 bytes to 1K bytes.
  - Integrated USB transceiver with 3.3V regulator
- 8-bit RISC microcontroller
  - Harvard architecture
  - 6-MHz external clock source
  - 12-MHz internal CPU clock
- Internal memory
  - 256 bytes of RAM
  - 4 KB of EPROM (CY7C64211, CY7C64311)
  - 6 KB of EPROM (CY7C64212, CY7C64312)
  - 8 KB of EPROM (CY7C64213, CY7C64313)
  - 1 Kbytes of Endpoint FIFOs
- I/O ports
  - Up to four General Purpose I/O (GPIO) ports (Port 0 to 3) capable of sinking 12 mA per pin (typical)
  - Higher current drive achievable by connecting multiple GPIO pins together to drive a common output
  - Each GPIO port can be configured as inputs with internal pull-ups or open drain outputs or traditional CMOS outputs
  - Maskable interrupts on all I/O pins
- DMA Controller. Transfers data between the GPIOs/CPU bus/1K FIFO bus interfaces and the USB bus through one DMA channel, without intervention of the microcontroller
- Bidirectional Hardware Handshaking Interface (BHHI) - shared with GPIOs
  - Flexible MCU interface which can be operated in slave mode by control signals from Host CPU to transfer data to/from 1 Kbyte FIFO, RAM, or I/O registers; also can be configured into IEEE1284 compatible modes to support standard PC parallel interface.
- Special Purpose I/O (SPIO)
  - Up to 7 configurable I/O pins: GPIO function, Timer1 clock input, Timer1 & 2 clock outputs, 12-MHz clock output, external reset.
- Serial Control Block (only one serial interface is available at the same time)
  - One full-duplex UART interface (shared with SPIO pins)

    Baud Rate: Max 1 Mbits/sec

    Error Detection: Parity/Framing/Over-run

    Parity: Odd/even/none
  - Industry-standard, high-speed serial SPI interface - Max. 6 MHz (shared with SPIO pins)
- Four timers:

  Timer0: Free-running 16-bit timer with 1 microsecond resolution

  Timer1 & Timer2: Multi-function 8 bit timers with prescaler, programmable down counter, and baud rate generator

  Watchdog timer (WDT)
- Internal power-on reset (POR), Optional external reset pin (enabled by firmware and shared with SPIO pin)
- Improved output drivers to reduce EMI
- Operating voltage from 4.0V to 5.25V DC
- Operating temperature from 0 to 70 degrees Celsius
- CY7C64211/12/13 available in 28-pin SOIC package
- CY7C64311/12/13 available in 48-pin SSOP package
- Industry standard programmer support

2.0 Functional Overview

The CY7C64211/12/13 and CY7C64311/12/13 are 8-bit RISC One Time Programmable (OTP) microcontrollers which are designed for full speed USB peripherals. The instruction set has been optimized specifically for USB operations, although the microcontrollers can be used for a variety of non-USB embedded applications.

The CY7C64211/12/13 features 18 General Purpose I/O (GPIO) pins to support USB and other applications. The I/O pins are grouped into three ports (P0[7:0], P1[3:0], P3[5:0]) where each port can be configured as inputs with internal pull-ups (14-K Ohm), open drain outputs, or traditional CMOS outputs. Ports P0,P1,P3 pins are rated at 12 mA typical sink current, a current sufficient to drive LEDs. Multiple GPIO pins can be connected together to drive a single output for more drive current capacity. Additionally, each GPIO pin can be used to generate a GPIO interrupt to the microcontroller. All of the GPIO interrupts share the same "GPIO" interrupt vector.

Thirty-two GPIO pins (P0[7:0], P1[7:0], P2[7:0], P3[7:0]) are available on the CY7C64311/12/13.

A Bidirectional Hardware Handshaking Interface (BHHI) is available through the GPIO pins to provide an 8-bit parallel interface to an external MCU, including control signals, address and data lines. The CY7C64311/12/13 have separate address and data lines, whereas the CY7C64211/12/13 have shared address and data lines (muxed). The interface is extremely flexible, allowing an external microcontroller to access the internal USB Endpoint FIFOs, RAM, or I/O registers.

The CY7C64311/12/13 features a full-duplex UART, an industry-standard, serial SPI interface, and Special Purpose I/O (SPIO). The UART interface, the SPI interface and four SPIO share common pins of the microcontroller.

The UART interface consists of four lines for RS-232 communication: TXD, RXD, CTS, RTS. It supports a baud rate from 2400 to 1M bits/s, parity (odd, even, or none), and error detection (framing, parity, and over-run).

The SPI interface is an industry-standard serial interface consisting of four lines: MOSI, MISO, SCK, and SS. The SPI interface is capable of transfer rates up to 6 MHz.

There are 7 Special Purpose I/O (SPIO) in the CY7C64311/12/13 which can be configured for a variety of purposes. SPIO[3:0], which shares common pins with the UART and SPI interfaces, can be configured as General Purpose I/O (GPIO) pins. SPIO[6:4] can be configured as GPIO, programmable timer outputs, one programmable timer input, and a 12 MHz clock output.

The microcontrollers use an external 6-MHz crystal to provide a reference to an internal PLL-based clock generator. This technology allows the customer application to use an inexpensive 6 MHz fundamental crystal that reduces the clock-related noise emissions (EMI). A PLL clock generator provides the 6, 12, and 48 MHz clock signals for distribution within the microcontroller.

The CY7C64211/12/13 and CY7C64311/12/13 are offered with three EPROM options. The CY7C64211 and CY7C64311 have 4 KB of EPROM. The CY7C64212 and CY7C64312 have 6 KB of EPROM. The CY7C64213 and CY7C64313 have 8 KB of EPROM. All parts have 256 bytes of internal RAM (including Endpoint 0) and 1 Kbytes of additional Endpoint FIFO memory. The 1 Kbytes of Endpoint FIFO space support up to 8 endpoints (Endpoints 1 to 9). The size of each endpoint is configurable between 128 bytes all the way up to 1 Kbyte, for a total of 1 Kbyte.

The CY7C642/3xx include power-on reset logic, an optional external reset pin (SPIO[4]), and four timers: a 12-bit 1MHz free-running timer, two programmable 8-bit timers, and a watchdog timer.

The power-on reset (POR) logic detects when power is applied to the device, resets the logic to a known state, and begins executing instructions at EPROM address 0x0000. An optional external reset pin is a shared function with SPIO[4] on both the CY7C64211/12/13 and CY7C64311/12/13, allowing external asynchronous resetting of the parts.

There are two programmable, 8-bit timers: Timer1 and Timer2. Both timers have selectable inputs: a programmable 8-bit prescaled clock derived from the 48-MHz clock, a 16-kHz clock, or a 250-kHz clock. Additionally, Timer1 can select an external clock input (through SPIO), and Timer2 can select Timer1 as an input. Both Timer1 and Timer2 can generate an interrupt to the CPU. Finally, the Quadrature Baud Rate Clock for the UART interface can be generated from either Timer1 or Timer2.

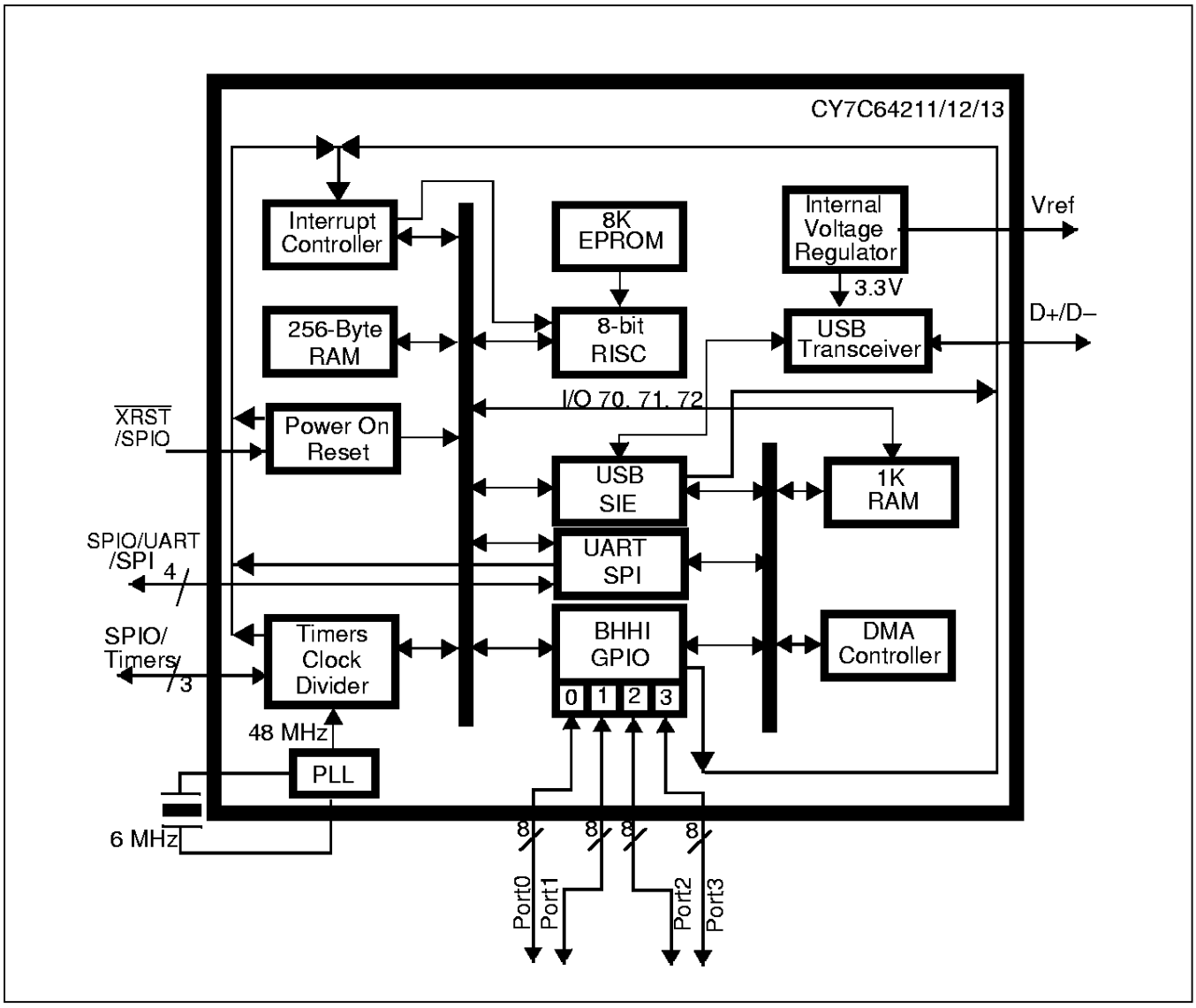
The watchdog timer is used to ensure the microcontroller can recover if it is stalled for more than approximately 8 msec. The firmware can get stalled for a variety of reasons, including errors in the code or a hardware failure such as waiting for an interrupt that never occurs. The firmware should clear the watchdog timer periodically (usually in the 1.024-msec interrupt service routine). If the watchdog timer is not cleared for approximately 8 msec, then a watchdog reset is generated that will return the microcontroller to a known state.

The free-running 16-bit timer, Timer0, clocked at 1 MHz, provides the interrupt source for the 1.024-msec interrupt, as noted above. Timer0 can be used to measure the duration of an event under firmware control by reading the timer twice: once at the start of the event, and once after the event is complete. The difference between the two readings indicates the duration of the event measured in microseconds. The upper 8 bits of the timer are latched into an internal register when the firmware reads the lower 8 bits. A read from the upper 8 bits actually reads data from the internal register, instead of the timer. This feature eliminates the need for firmware to attempt to compensate if the upper 8 bits happened to increment right after the lower 8 bits are read.

The microcontroller supports 10 maskable interrupts in the vectored interrupt controller. Interrupt sources include the 1.024 msec (bit 9) outputs from the free-running timer, 2 USB endpoint interrupts (Endpoint 0 and Endpoint i, i=1..8), the GPIO/SPIO ports, the BHH interface, the UART interface, Timer1 and Timer2, USB Bus Reset, SOF. The timer bits cause an interrupt (if enabled) when the bit toggles from low "0" to high "1". The USB endpoints interrupt after the USB host has written data to the endpoint FIFO or after the USB controller sends a packet to the USB host. The GPIO ports also have a level of masking to select which

GPIO inputs can cause a GPIO interrupt. For additional flexibility, the input transition polarity that causes an interrupt is programmable for each GPIO port. The interrupt polarity can be either rising edge ("0" to "1") or falling edge ("1" to "0").

The CY7C64211/12/13 and CY7C64311/12/13 include an integrated USB serial interface engine (SIE) that supports the integrated peripheral. The hardware supports one USB device addresses with up to nine endpoints. The SIE allows the USB host to communicate with the functions integrated into the microcontroller.

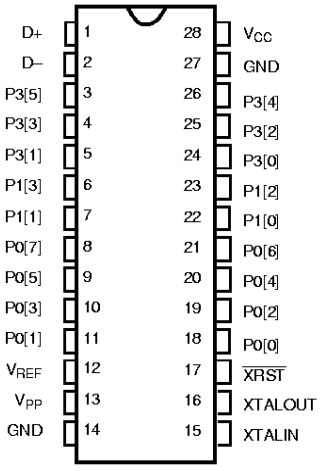


Pin Configurations

TOP VIEW

CY7C64211/12/13

28-pin SOIC



CY7C64311/12/13

48-pin SSOP

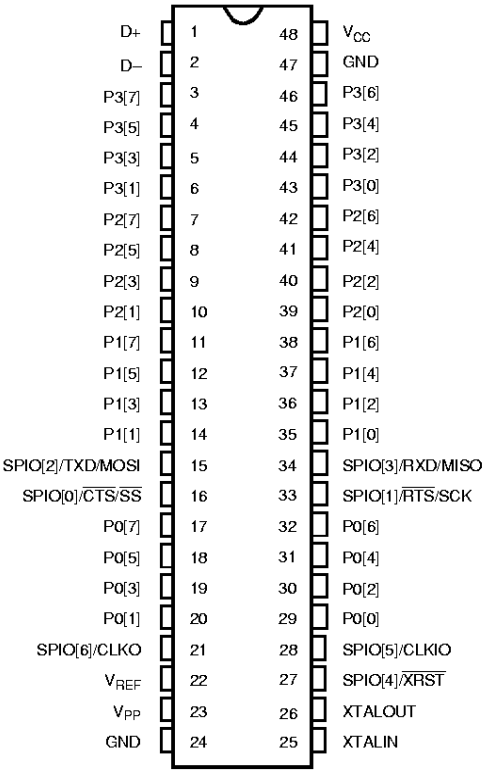


Figure 2-1. Pin Configurations

3.0 Product Summary Tables

3.1 Pin Assignments

Name	I/O	28-pin SOIC	48-Pin	Description
D+, D–	I/O	1,2	1,2	Upstream port, USB differential data
P0	I/O	18,11,19,10, 20,9,21,8	29,20,30,19, 31,18,32,17	GPIO Port 0 capable of sinking 12 mA (typical). BHH Interface control signals.
P1	I/O	22,7,23,6	35,14,36,13, 37,12,38,11	GPIO Port 1 capable of sinking 12 mA (typical) BHH Interface control signals.
P2	I/O		39,10,40,9, 41,8,42,7	GPIO Port 2 capable of sinking 12 mA (typical.) BHH Interface control signals.
P3	I/O	24,5,25,4, 26,3	43,6,44,5, 45,4,46,3	GPIO Port 3 capable of sinking 12 mA (typical). BHH Interface control signals.
SPIO[0]/ CTS/SS	I/O		16	Special Purpose I/O; UART Interface, Clear to Send; or SPI Interface, Slave Select
SPIO[1]/ RTS/SCK	I/O		33	Special Purpose I/O; UART Interface, Request to Send; or SPI Interface, Serial Clock
SPIO[2]/ TXD/MOSI	I/O		15	Special Purpose I/O; UART Interface, Transmit Data Line; or SPI Interface, Master Out/Slave In
SPIO[3]/ RXD/MISO	I/O		34	Special Purpose I/O; UART Interface, Receive Data Line; or SPI Interface, Master In/Slave Out
SPIO[4]/ XRST	I/O	17	27	Special Purpose I/O; External Reset
SPIO[5]/ CLKIO/ T1O/HTI1	I/O		28	Special Purpose I/O; Clock Input-Output Timer1; Timer1 Output/Hardware Trigger Enable for Timer1
SPIO[6]/ CLKO/ T2O/HTI2	I/O		21	Special Purpose I/O; Clock Output Timer2; Timer2 Output/Hardware Trigger Enable for Timer2
XTALIN	IN	15	25	6-MHz crystal or external clock input
XTALOUT	OUT	16	26	6-MHz crystal out
V_{PP}	IN	13	23	Programming voltage supply, tie to ground during normal operation
V_{CC}	Power	28	48	Voltage supply
GND	Ground	14,27	24,47	Ground
V_{REF}	OUT	12	22	3.3V reference voltage from internal voltage regulator. 100-nF capacitor between V_{REF} and GND is required.

3.2 I/O Register Summary

I/O registers are accessed via the I/O Read (IORD) and I/O Write (IOWR, IOWX) instructions. IORD reads the selected port into the accumulator. IOWR writes data from the accumulator to the selected port. Indexed I/O Write (IOWX) adds the contents of X to the address in the instruction to form the port address and writes data from the accumulator to the specified port. Note that specifying address 0 (e.g., IOWX 0h) means the I/O port is selected solely by the contents of X.

Table 3-1. I/O Register Summary

Register Name	I/O Address	Read/ Write	Function
Port 0 Data	0x00	R/W	General Purpose I/O Port 0
Port 1 Data	0x01	R/W	General Purpose I/O Port 1
Port 2 Data	0x02	R/W	General Purpose I/O Port 2
Port 3 Data	0x03	R/W	General Purpose I/O Port 3
Port 0 Interrupt Enable	0x04	W	Interrupt enable for pins in Port 0

Table 3-1. I/O Register Summary (continued)

Register Name	I/O Address	Read/ Write	Function
Port 1 Interrupt Enable	0x05	W	Interrupt enable for pins in Port 1
Port 2 Interrupt Enable	0x06	W	Interrupt enable for pins in Port 2
Port 3 Interrupt Enable	0x07	W	Interrupt enable for pins in Port 3
GPIO Configuration	0x08	R/W	General Purpose I/O Ports Configurations
GPIO BHH Control	0x09	R/W	GPIO Ports Bid. Hardware Handshake Control
SPIO Data	0x0A	R/W	Special Purpose I/O Port
SPIO Interrupt Enable	0x0B	W	Interrupt enable for pins in SPIO port
SPIO Configuration	0x0C	R/W	Special Purpose I/O Ports Configuration
USB Device Address	0x10	R/W	USB Device address
EP 0 Counter Register	0x11	R/W	USB Endpoint 0 counter register
EP 0 Mode Register	0x12	R/W	USB Endpoint 0 configuration register
EP1–8 Stall Mode Register	0x13	R/W	USB Endpoint 1–8 Stall Mode Control Register
EP1–8 USB ACK Status	0x18	R/W	USB Endpoint 1–8 ACK Status
EP1–8 Buffer Control	0x19	R/W	USB Endpoint 1–8 Buffer Control
USB Status & Control	0x1F	R/W	USB upstream port traffic status & control register
Global Interrupt Enable 1	0x20	R/W	Global interrupt enable 1 register
Global Interrupt Enable 2	0x21	R/W	Global interrupt enable 2 register
Interrupt Vector Read	0x23	R	Allows Read of Interrupt Vector in Test Mode Only
Timer 0 (LSB)	0x24	R	Free-running timer Lower 8 bits (1 MHz)
Timer 0 (MSB)	0x25	R	Free-running timer Upper 8 bits
Watch Dog Clear	0x26	W	Watch Dog Timer clear
Timer Prescaler	0x28	R/W	Prescaling Timer Control Register
Timer 1	0x29	R/W	Timer for generating clock or interrupt or PWM
Timer 2	0x2A	R/W	Timer for generating clock/interrupt/PWM
Timer Control	0x2C	R/W	Control the inputs and outputs of timer 1, 2
UART/SPI Transmit Data	0x30	W	UART/SPI Transmit Data Register
UART/SPI Receive Data	0x31	R	UART/SPI Receive Data Register
UART/SPI Control	0x32	R/W	UART/SPI Control Register
UART/SPI Status	0x33	R/W	UART/SPI Status Register
USB EP 1 Counter	0x40	R/W	USB EP 1 Counter Register
USB EP 1 Mode	0x41	R/W	USB EP 1 Mode Register
USB EP 2 Counter	0x42	R/W	USB EP 2 Counter Register
USB EP 2 Mode	0x43	R/W	USB EP 2 Mode Register
USB EP 3 Counter	0x44	R/W	USB EP 3 Counter Register
USB EP 3 Mode	0x45	R/W	USB EP 3 Mode Register
USB EP 4 Counter	0x46	R/W	USB EP 4 Counter Register
USB EP 4 Mode	0x47	R/W	USB EP 4 Mode Register
USB EP 5 Counter	0x48	R/W	USB EP 5 Counter Register
USB EP 5 Mode	0x49	R/W	USB EP 5 Mode Register
USB EP 6 Counter	0x4A	R/W	USB EP 6 Counter Register
USB EP 6 Mode	0x4B	R/W	USB EP 6 Mode Register

Table 3-1. I/O Register Summary (continued)

Register Name	I/O Address	Read/ Write	Function
USB EP 7 Counter	0x4C	R/W	USB EP 7 Counter Register
USB EP 7 Mode	0x4D	R/W	USB EP 7 Mode Register
USB EP 8 Counter	0x4E	R/W	USB EP 8 Counter Register
USB EP 8 Mode	0x4F	R/W	USB EP 8 Mode Register
Address 0 for 1K Buffer	0x70	R/W	LSB 8 bit Address for accessing 1K Buffer
Address 1 for 1K Buffer	0x71	R/W	MSB 2 bit Address for accessing 1K Buffer
Data for 1K Buffer	0x72	R/W	1K Buffer Data Register
Data for 1K Buffer Through DMA	0x73	R/W	1K Buffer Data Register with Auto-Incrementing of 70
DMA Stop Pointer	0x74	R/W	DMA Stop Pointer of DMA controller
Test Register Configuration	0xF0	R/W	Test Register for internal use only
POR Configuration	0xF1	R	POR Configuration
Device ID Register	0xF3	R	USB ID for Device Revision (0x40)
Manufacturer ID Register	0xF4	R	USB ID for Cypress (0x34)
Processor Status & Control	0xFF	R/W	Microprocessor status & control register

3.3 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	4
AND A,expr	data	10	4		SWAP A,DSP		30	4
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

MNEMONIC	operand	opcode	cycles		MNEMONIC	operand	opcode	cycles
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
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					
JMP	addr	80-8F	5		JC	addr	C0-CF	5
CALL	addr	90-9F	10		JNC	addr	D0-DF	5
JZ	addr	A0-AF	5		JACC	addr	E0-EF	7

4.0 Programming Model

4.1 14-bit Program Counter (PC)

The 14-bit program counter (PC) allows access to up to 8 KB of EPROM available with the CY7C642/3xx architecture. The program counter is cleared during reset, such that the first instruction executed after a reset is at address 0x0000h. This is typically a jump instruction to a reset handler that initializes the application.

The lower 8 bits of the program counter are incremented as instructions are loaded and executed. The upper 6 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" will cause the assembler to insert XPAGE instructions automatically. As instructions can be either one or two bytes long, the assembler may occasionally need to insert a NOP followed by an XPAGE for correct execution.

The program counter of the next instruction to be executed, carry flag, and 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. Please note the program counter cannot be accessed directly by the firmware. The program stack can be examined by reading SRAM from location 0x00 and up.

4.1.1 Program Memory Organization

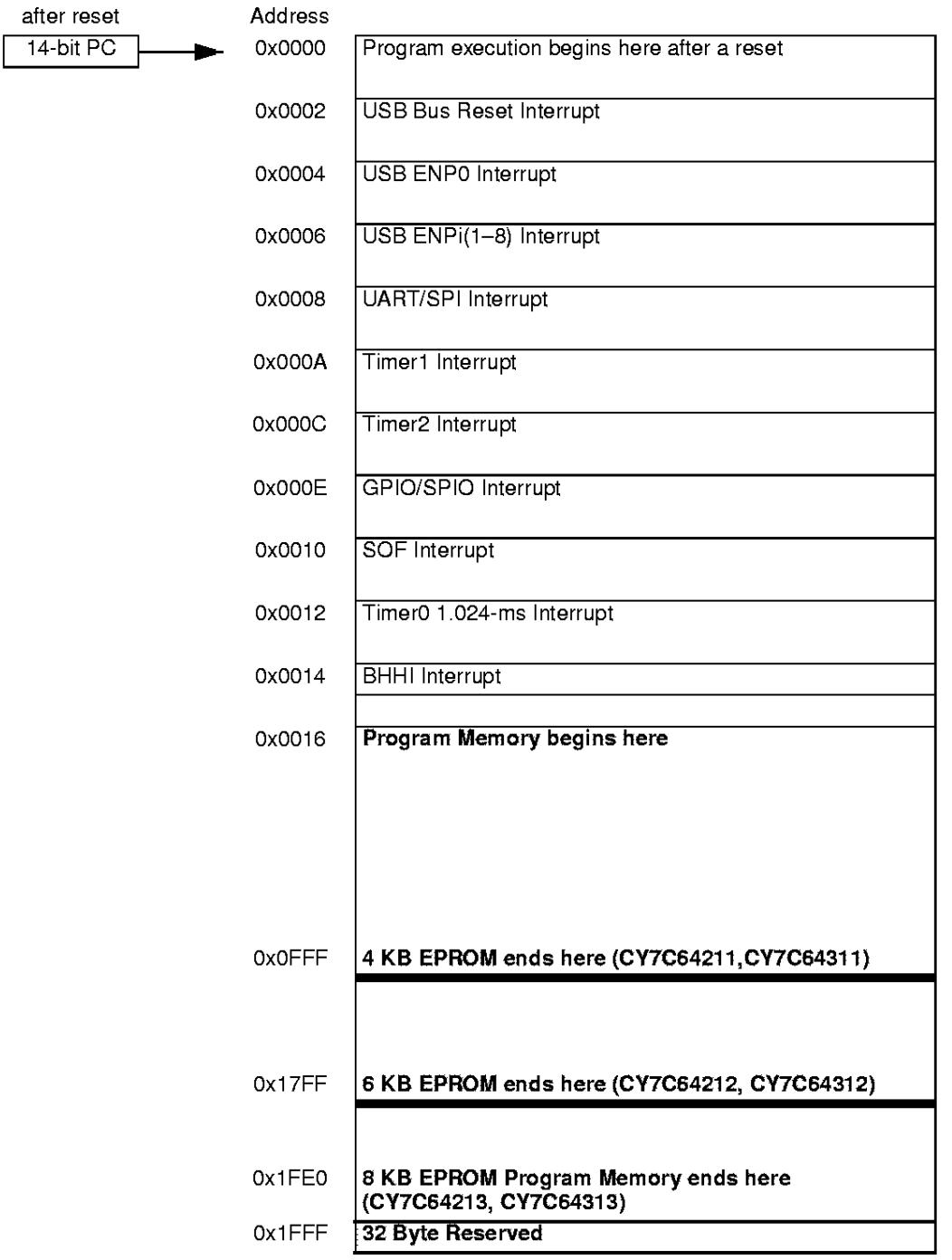


Figure 4-1. Program Memory Space with Interrupt Vector Table

4.2 8-bit Accumulator (A)

The accumulator is the general-purpose register for the microcontroller.

4.3 8-bit Temporary Register (X)

The "X" register is available to the firmware for temporary storage of intermediate results. The microcontroller can perform indexed operations dependent of the value in X.

4.4 8-bit Program Stack Pointer (PSP)

During a reset, the program stack pointer (PSP) is set to 0x00 and "grows" upward from this address. The program stack pointer is directly addressable under firmware control, using the MOV PSP,A instruction. The PSP supports interrupt service under hardware control and CALL, RET, and RETI instructions under firmware control.

During an interrupt acknowledge, interrupts are disabled and the 14-bit program counter, carry flag, and zero flag are written as two bytes of data memory. The first byte is stored in the memory addressed by the program stack pointer, then the PSP is incremented. The second byte is stored in memory addressed by the program stack pointer and the PSP is incremented again. The net effect is to store the program counter and flags on the program "stack" and increment the program stack pointer by two.

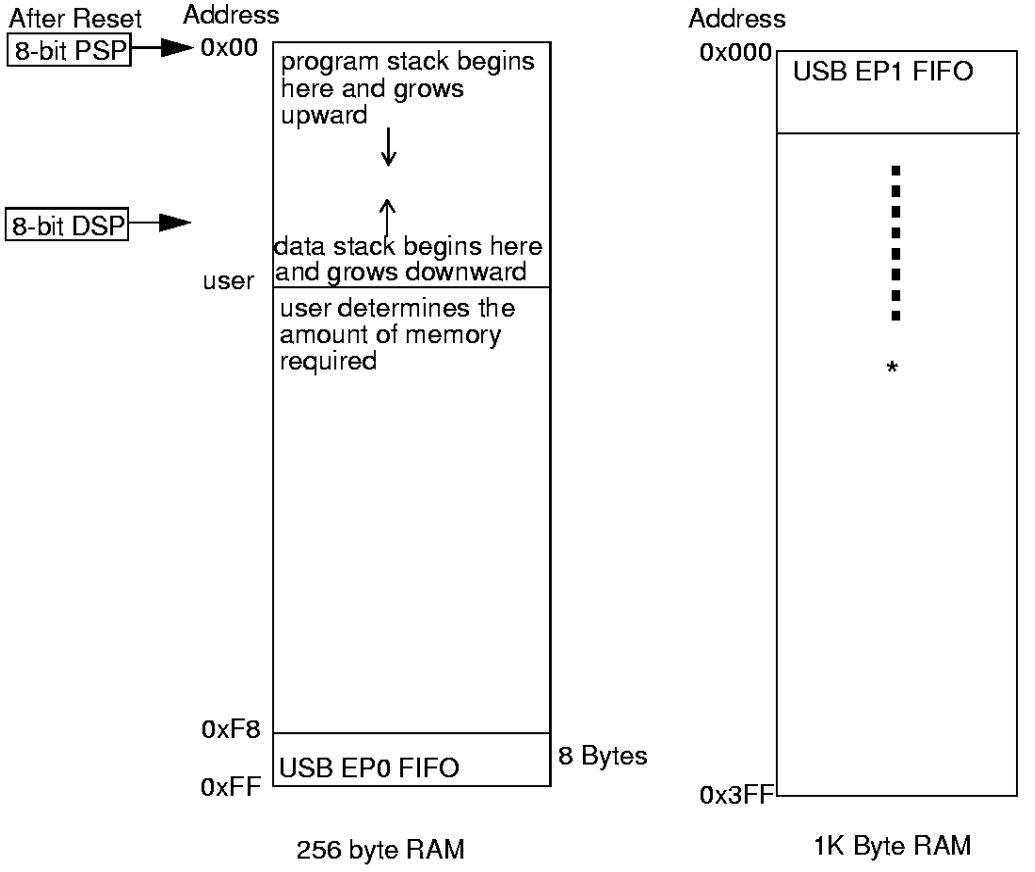
The return from interrupt (RETI) instruction decrements the program stack pointer, then restores the second byte from memory addressed by the PSP. The program stack pointer is decremented again and the first byte is restored from memory addressed by the PSP. After the program counter and flags have been restored from stack, the interrupts are enabled. The effect is to restore the program counter and flags from the program stack, decrement the program stack pointer by two, and re-enable interrupts.

The call subroutine (CALL) instruction stores the program counter and flags on the program stack and increments the PSP by two.

The return from subroutine (RET) instruction restores the program counter but not the flags from the program stack and decrements the PSP by two.

4.4.1 Data Memory Organization

The CY7C64211/12/13 and CY7C64311/12/13 microcontrollers provide 256 bytes of data RAM and 1 Kbytes of Endpoint FIFO RAM. The data RAM is partitioned into four areas: program stack, data stack, user variables and a USB endpoint FIFO. The USB Endpoint 0 FIFO resides in the 256 byte data RAM. All other USB Endpoint FIFOs are located in the 1-Kbyte RAM. The maximum number of endpoints supported is 8. The FIFO sizes are selected through register 0x19, and can be set between 128 to 1023 bytes. The number of endpoints will be restricted if large FIFOs are selected. See below:



* Total of 8 EP can be supported, refer to register 0x19 for more detail.

4.5 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 will pre-decrement the DSP, then write data to the memory location addressed by the DSP. A POP instruction will read data from the memory location addressed by the DSP, then post-increment the DSP.

During a reset, the DSP will be reset to 0x00. A PUSH instruction when DSP equals 0x00 will write data at the top of the data RAM (address 0xFF). This would write data to the memory area reserved for the USB endpoint 0 FIFO. Therefore, the DSP should be indexed at an appropriate memory location that will not compromise the PS, USER defined memory (variables), or the USB endpoint 0 FIFO.

For USB applications, the firmware should set the DSP to the appropriate location above the program stack (PS), without interfering with the USB endpoint FIFO. For instance, if the program and data stacks are each determined to be 30 bytes, the DSP should be initialized to 3Ch. The assembly instructions to do this are shown below:

```
Mov A, 3Ch    ; Move 0x3C hex into Accumulator
Swap A,dsp    ; swap accumulator value into dsp register
```

4.6 Address Modes

The CY7C64211/12/13 and CY7C64311/12/13 microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

4.6.1 Data

The "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 0x3Ch:

- MOV A,3Ch

This instruction will require 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 will be the constant "0x3Ch". 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 3Ch
- MOV A,DSPINIT

4.6.2 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 0x10h:

- MOV A, [10h]

In normal usage, 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]

4.6.3 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. In normal usage, the constant will be the "base" address of an array of data and the X register will contain 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 0x10h. The fourth element would be at address 0x13h.

5.0 Clocking

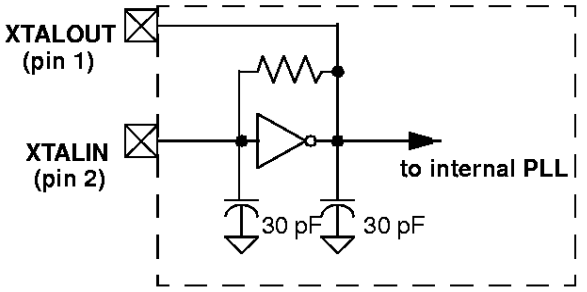


Figure 5-1. Clock Oscillator On-chip Circuit

The XTALIN and XTALOUT are the clock pins to the microcontroller. The user can either connect an external oscillator or a crystal to these pins. A 6-MHz fundamental crystal can be connected to these pins to provide a reference frequency for the internal PLL. A ceramic resonator will not allow the microcontroller to meet the timing specifications, 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. Please note that grounding the XTALOUT pin when driving XTALIN with an oscillator will not work as the internal clock is effectively shorted to ground.

6.0 Reset

The CY7C642/3xx supports three resets: Power on Reset (POR), External Reset (optional), and Watch Dog Reset (WDR). Upon reset:

- all registers will be restored to their default states,
- the USB Device Address is set to 0,
- all interrupts are disabled,
- the Program Stack Pointer (PSP) and Data Stack Pointer (DSP) are set to memory address 0x00.

The occurrence of a reset is recorded in the Processor Status and Control Register located at I/O address 0xFF. Bits [6:4] are used to record the occurrence of POR, USB Bus Reset[1], and WDR respectively. Firmware can interrogate these bits to determine the cause of a reset.

Program execution starts at ROM address 0x0000h 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 the program stack. The firmware reset handler should configure the hardware before the "main" loop of code. Attempting to execute either a RET or RETI in the firmware reset handler will cause unpredictable execution results.

6.1 Power-On Reset (POR)

Power-On Reset (POR) occurs every time the V_{CC} voltage to the device ramps from 0V to an internally defined trip voltage (Vrst), of approximately 1/2 full supply voltage. In addition to the normal reset initialization noted under "Reset," bit 4 (PORS) of the Processor Status and Control Register (Reg. 0xFF) is set to "1" to indicate to the firmware that a power on reset occurred. The POR event forces the GPIO ports into input mode (high impedance).

The device will be in the Suspend state after Power-On-Reset, and a 96-ms timer will start. After 96 ms, the device will come out of the Suspend state and be ready to run. During the 96 ms, any USB Bus Reset event will overwrite the timer and bring the device out of the Suspend state.

A brownout circuit resets the device during rapid disconnecting/connecting on the USB bus.

6.2 External Reset

An optional external reset pin is provided (otherwise used as SPIO[4]). The external reset is active LOW and has to be asserted for 16 µs to be recognized. The external reset is enabled through POR Configuration Register (0xF1), it is default disabled. The internal POR circuit is always used.

Note:
1. USB Bus Reset is not a hardware reset in this device. See section 16.3, USB Bus Reset Interrupt.

6.3 Watch Dog Reset (WDR)

The Watch Dog Timer Reset (WDR) occurs when the Most Significant Bit (MSB) of the 2-bit Watch Dog Timer Register transitions from LOW to HIGH. In addition to the normal reset initialization noted under "Reset", bit 6 of the Processor Status and Control Register is set to "1" to indicate to the firmware that a Watch Dog Reset occurred.

The Watch Dog Timer is a 2-bit timer clocked by a 4-ms clock from the free running timer. Writing any value to the write-only Watch Dog Clear I/O port (0x26h) will clear the Watch Dog Timer.

In some applications, the Watch Dog Timer may be cleared in the 1.024-ms timer interrupt service routine. If the 1.024-ms timer interrupt service routine does not get executed for 8 ms or more, a Watch Dog Timer Reset will occur. A Watch Dog Timer Reset lasts for 2.048 ms after which the microcontroller begins execution at ROM address 0x0000h. The USB transmitter is disabled by a Watch Dog Reset because the USB Device Address Register is cleared. Otherwise, the USB Controller would respond to all address 0 transaction. The USB SIE remains disabled until the MSB of the USB address register is set.

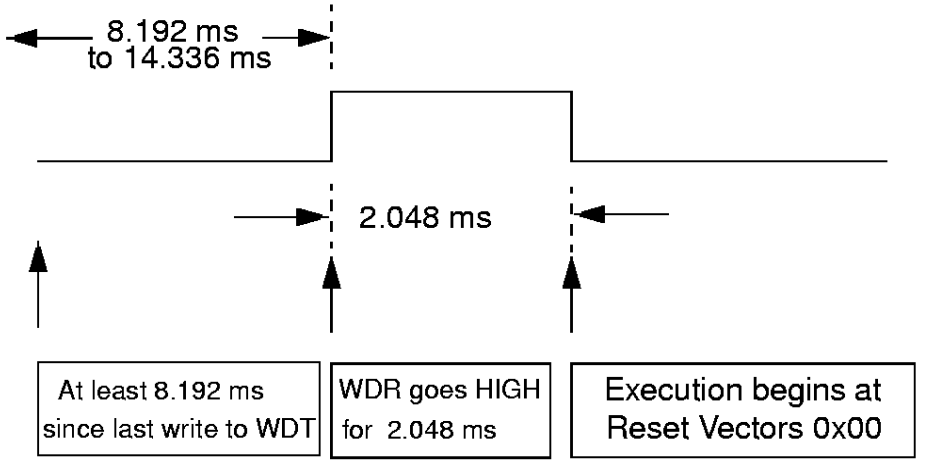


Figure 6-1. Watch Dog Reset (WDR)

7.0 General Purpose I/O Ports

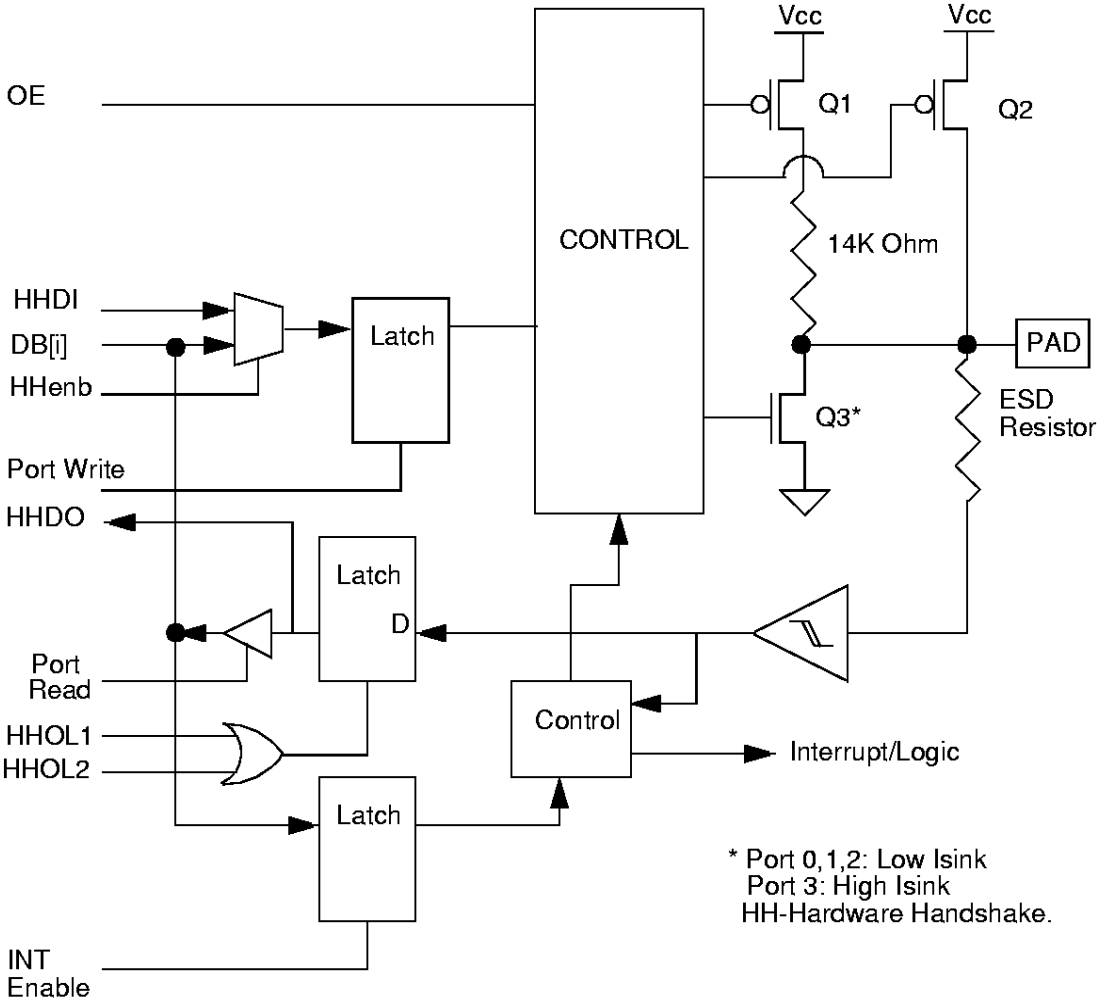


Figure 7-1. Block Diagram of a General Purpose I/O Line

There are 32 GPIO pins (P0, P1, P2, and P3) available in the CY7C64311/12/13, and 18 GPIO pins (P0, P1, and P3) available in the CY7C64211/12/13. Each port can be configured as resistive inputs with internal 14-KΩ pull-ups, open drain outputs, or traditional CMOS I/O. The interrupt polarity and interrupt enable are also user-programmable. Programmability is on a per port basis for drive and interrupt polarity, and on a per bit basis for interrupt enable. Ports 0 to 3 offer a typical current sink capability of 12 mA. Input data can be latched if desired using the external strobe signal (STRB) in the Hardware Handshaking (HH) interface. The data for each GPIO port is accessible through the data registers.

P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]

Figure 7-2. Port 0 Data 0x00h (read/write)

P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]

Figure 7-3. Port 1 Data 0x01h (read/write)

P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]

Figure 7-4. Port 2 Data 0x02h (read/write)

unused	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]

Figure 7-5. Port 3 Data 0x03h (read/write)

Special care should be exercised with any unused GPIO data bits. An unused GPIO data bit, whether it represent a pin on the chip or it is not bonded out to a pin on a particular package, can 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 could violate the suspend current limitation specified by the USB specification. If a 1 is written to the unused data bit and the port is configured with open drain outputs, the unused data bit will be in an indeterminate state. Therefore, if an unused port bit is programmed in open-drain mode, the GPIO data bits related to this port must be written with a 0 prior to entering the suspend state. Notice that the CY7C64211/12/13 will always require that the data bits P1[7:4], P2[7:0], P3[7:6], related to unconnected pins in this package, be written with a 0.

During reset, all of the GPIO pins are set to output 1 (input). Writing a 0 to a GPIO pin enables the output current sink to ground (LOW) and disables the internal pull-up for that pin. In this state, a 0 will always be read on that GPIO pin unless an external current source overdrives the output current sink.

7.1 GPIO Configuration Port

Every GPIO port can be programmed as inputs with internal pull-ups (resistive), open drain outputs, and traditional CMOS outputs. In addition, the interrupt polarity for each port can be programmed. With positive interrupt polarity, a rising edge (0 to 1) on an input pin causes an interrupt. With negative polarity, a falling edge (1 to‘0) on an input pin causes an interrupt. As shown in the table below, when a GPIO port is configured with CMOS outputs, interrupts from that port are disabled. The GPIO Configuration Port register provides two bits per port to program these features. The possible port configurations are detailed in *Table 7-1*.

Table 7-1. Port Configurations

Port Configuration bits	Pin Interrupt Bit	Driver Mode	Interrupt Polarity
11	X	Resistive	–
10	0	CMOS Output	disabled
10	1	CMOS Input	disabled
01	X	Open Drain	–
00	X	Open Drain	+ (default)

In “Resistive” mode, a 14-kΩ pull-up resistor is conditionally enabled for all pins of a GPIO port. The resistor is enabled for any pin that has been written as a 1. The resistor is disabled on any pin that has been written as a 0. An I/O pin will be driven HIGH through a 14-kΩ pull-up resistor when a 1 has been written to the pin. The output pin will be driven LOW with the pull-up disabled when a 0 has been written to the pin. An I/O pin that has been written as a 1 can be used as an input pin with an integrated 14-kΩ pull-up resistor. Resistive mode selects a negative (falling edge) interrupt polarity on all pins that have the GPIO interrupt enabled.

In “CMOS” mode, all pins of the GPIO pin are outputs that are actively driven. The current source and sink capacity are roughly the same (symmetric output drive). A CMOS port is not a possible source for interrupts.

In “Open Drain” mode the internal pull-up resistor and CMOS driver (HIGH) are both disabled. An I/O pin that has been written as a 1 can be used as either a high-impedance input or an open drain output. An I/O pin that has been written as a 0 will drive the output LOW. The interrupt polarity for an open drain GPIO port can be selected as either positive (rising edge) or negative (falling edge).

During reset, all of the bits in the GPIO Configuration Register are written with 0 to select Open Drain output, positive interrupt polarity for all GPIO ports as the default configuration.

7	6	5	4	3	2	1	0
Port 3 Config Bit 1	Port 3 Config Bit 0	Port 2 Config Bit 1	Port 2 Config Bit 0	Port 1 Config Bit 1	Port 1 Config Bit 0	Port 0 Config Bit 1	Port 0 Config Bit 0

Figure 7-6. GPIO Configuration Register 0x08h (write only)

7.2 GPIO Interrupt Enable Ports

During a reset, GPIO interrupts are disabled by clearing all of the GPIO interrupt enable ports. Writing a 1 to a GPIO Interrupt Enable bit enables GPIO interrupts from the corresponding input pin.

P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]

Figure 7-7. Port 0 Interrupt Enable 0x04h (write only)

P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]

Figure 7-8. Port 1 Interrupt Enable 0x05h (write only)

P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]

Figure 7-9. Port 2 Interrupt Enable 0x06h (write only)

P3[7]	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]

Figure 7-10. Port 3 Interrupt Enable 0x07h (write only)

8.0 Timers

8.1 16-bit Free-running Timer (Timer0)

The 16-bit timer (Timer0) provides one interrupt (1.024 msec) and allows the firmware to directly time events that are up to 65 msec in duration. The lower 8 bits of the timer can be read directly by the firmware. Reading the lower 8 bits latches the upper 8 bits into a temporary register. When the firmware reads the upper 8 bits of the timer, it is accessing the count stored in the temporary register. The effect of this logic is to ensure a stable 16-bit timer value can be read, even when the two reads are separated in time.

8.1.1 Timer0 (LSB)

Timer0 Bit 7	Timer0 Bit 6	Timer0 Bit 5	Timer0 Bit 4	Timer0 Bit 3	Timer0 Bit 2	Timer0 Bit 1	Timer0 Bit 0

Figure 8-1. Timer0 Register 0x24h (read only)

8.1.2 Timer0 (MSB)

Timer0 Bit 15	Timer0 Bit 14	TImer0 Bit 13	Timer0 Bit 12	Timer0 Bit 11	Timer0 Bit 10	Timer0 Bit 9	Timer0 Bit 8

Figure 8-2. Timer0 Register 0x25h (read only)

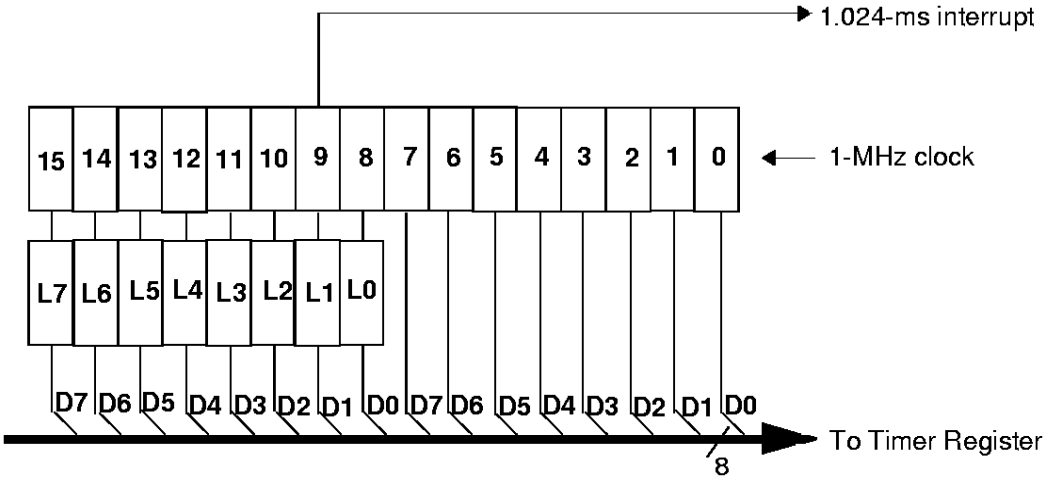


Figure 8-3. Timer0 Block Diagram

8.2 Timer1, Timer2 and Prescaler

Timer1 and Timer2 are programmable 8-bit timers that can be used as terminal counters, or pulse width modulation (PWM) counters (bit 0 and bit 1 of register 0x2Ch determine the timer function for Timer1 and Timer2, respectively). As terminal counters, the timers can be programmed to generate an interrupt to the CPU when the timer has completed down counting to the zero, and automatically reload the count value (in register 0x29h or 0x2Ah). Pulse width modulation can be accomplished by writing into

the registers 0x29h or register 0x2Ah the number of clocks the output will be at logic 1 (register value + 1), the rest of clocks, while a timer is down counting to the terminal count, will be at logic 0 (256 – register value – 1).

Additionally, there are two modes for Timer1 and Timer2: trigger_enable mode and non_trigger_enable mode. The control bits HtrigModeT1 and HtrigModeT2 (bits 7 and 6 of register 0x0Ch) determine the timer mode. In trigger_enable mode (HtrigModeT1 or HtrigModeT2 is 1), if either of the trigger enable signals, Htrig_en1 (SPIO[5]) or Htrig_en2 (SPIO[6]), are 1, then the corresponding timer, Timer1 or Timer2, works as a terminal counter or PWM counter. If either Htrig_en1 or Htrig_en2 are 0, Timer1 or Timer2 do not count, and the timer output is 0. In non_trigger_enable mode (HtrigModeT1 or HtrigModeT2 is 0), Timer1 and Timer2 work normally as terminal counters or PWM counters.

The input of the Prescaler is 48 MHz. The output of the Prescaler is controlled by Register 0x28h and can be used as an input to Timer1 and Timer2 for further scale down.

The Prescaler is a down counter. The counter loads in the Register 0x28h value. The counter then starts the down counting. When the counter is down to 0, it will generate an interrupt and reload the Register 0x28h value into the counter. The output of the prescaler is divided by (register value +1). For example, if the register value is 0, the output will be the input clock (i.e., divide by 1).

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Count7	Count6	Count5	Count4	Count3	Count2	Count1	Count0

Figure 8-4. Prescaler Output Frequency Control Register 0x28h

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Count7	Count6	Count5	Count4	Count3	Count2	Count1	Count0

Figure 8-5. Timer1 Control Register 0x29h

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Count7	Count6	Count5	Count4	Count3	Count2	Count1	Count0

Figure 8-6. Timer2 Control Register 0x2Ah

7	6	5	4	3	2	1	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reserved	TS2[1]	TS2[0]	TS1[1]	TS1[0]	BSel	Timer2 Rate/PWM	Timer1 Rate/PWM

Figure 8-7. Timer Control Register 0x2Ch

Bits[6:5] select the clock input for Timer2: 00 = 16 kHz; 01 = Prescaler output; 10 = 250 kHz; 11 = Timer1 output.

Bits[4:3] select the clock input for Timer1: 00 = 16 kHz; 01 = Prescaler output; 10 = 250 kHz; 11 = clockIN (SPIO clock input).

Bit 2 selects which timer output is used for the UART clock input: 0 = Timer1; 1 = Timer2.

Bits [1:0] select the timer function for Timer1 and Timer2, respectively: 0 = terminal counter; 1 = PWM.

When Timer1 and Timer2 are in non_trigger_enable mode and operating as terminal counters, their outputs are as shown in *Figure 8-8*.

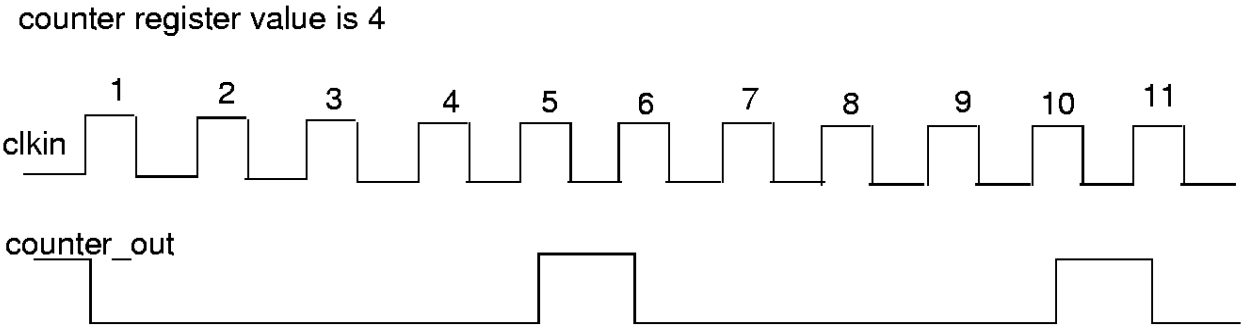


Figure 8-8. Timing Diagram for Prescaler or Timer1/Timer2 Terminal Counters (non_trigger_enable mode)

When Timer1 and Timer2 are in trigger_enable mode and operating as terminal counters, their outputs are as shown in *Figure 8-9*.

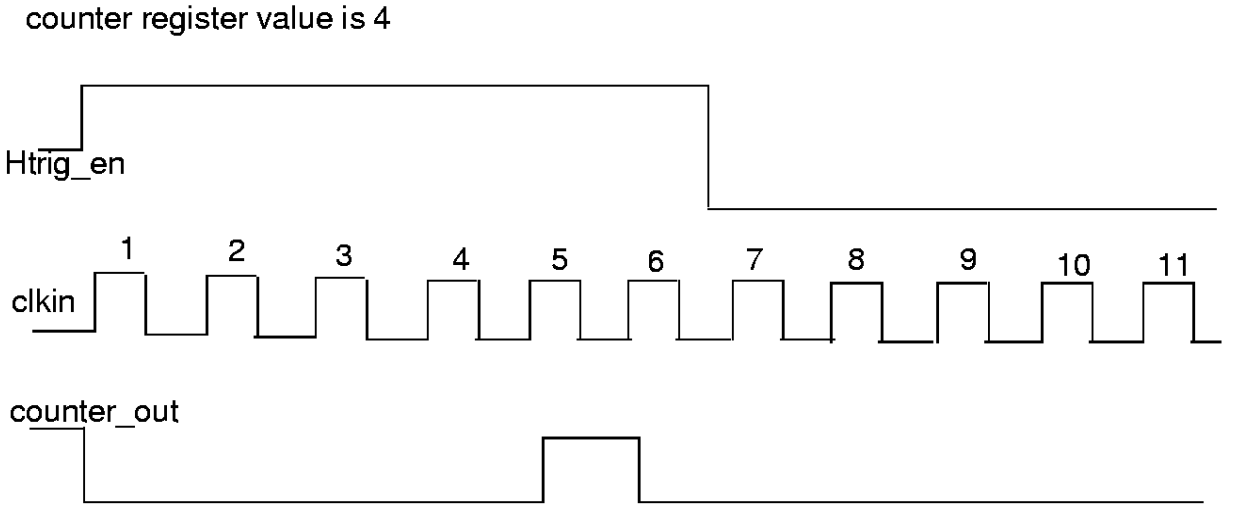


Figure 8-9. Timing Diagram for Timer1/Timer2 Terminal Counters (trigger_enable mode)

When Timer1 or Timer2 is set for pulse width modulation (PWM) in non_trigger_enable mode, it still operates as a down counter, but allows variation of the duty cycle. The output of Timer1 (or Timer2) is a clock which has (register value + 1) clocks at logic 1 and (256 – register value – 1) clocks at logic 0. The count value written to either register 0x29h or 0x2Ah does not take effect until the end of the current cycle. See *Figure 8-10*.

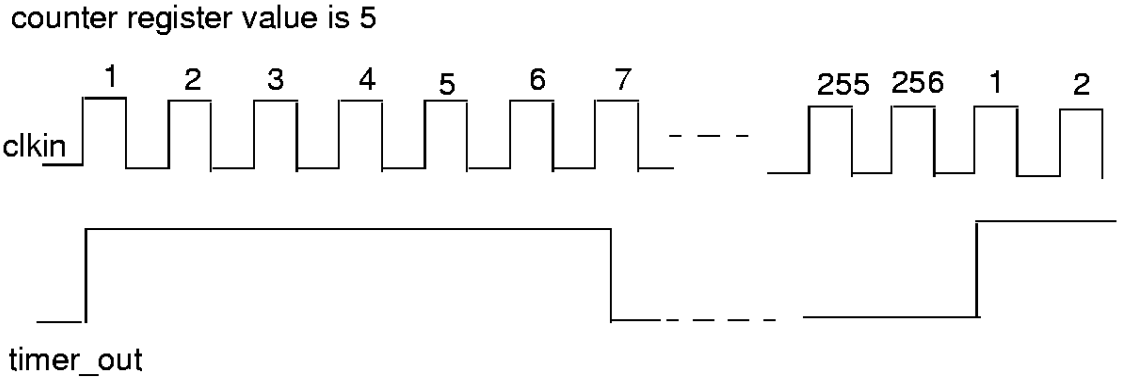


Figure 8-10. Timing Diagram for Timer1/Timer2 PWM (non_trigger_enable mode)

When Timer1 or Timer2 is set for PWM in trigger_enable mode, the output is as shown in *Figure 8-11*.

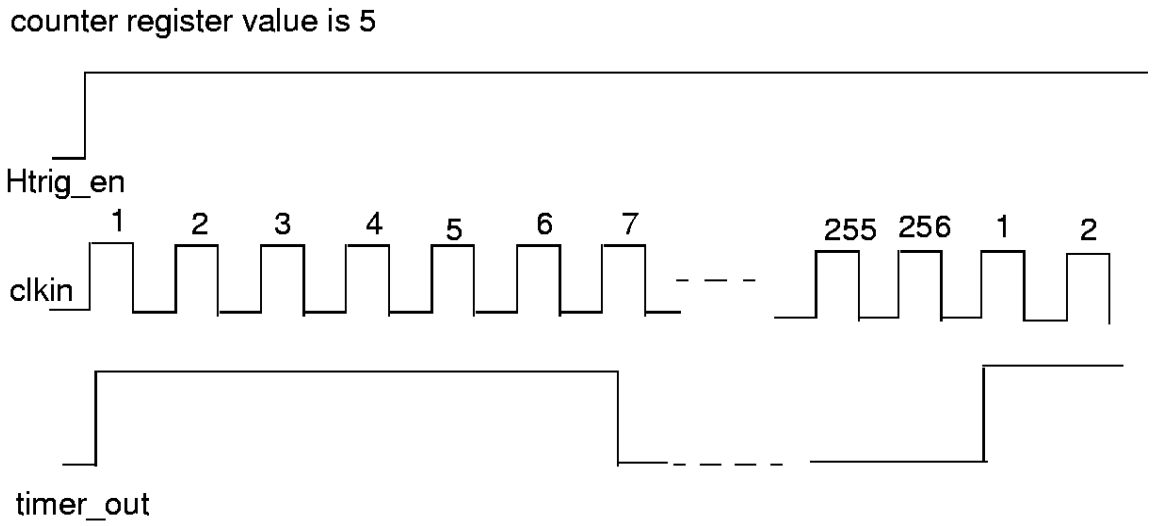
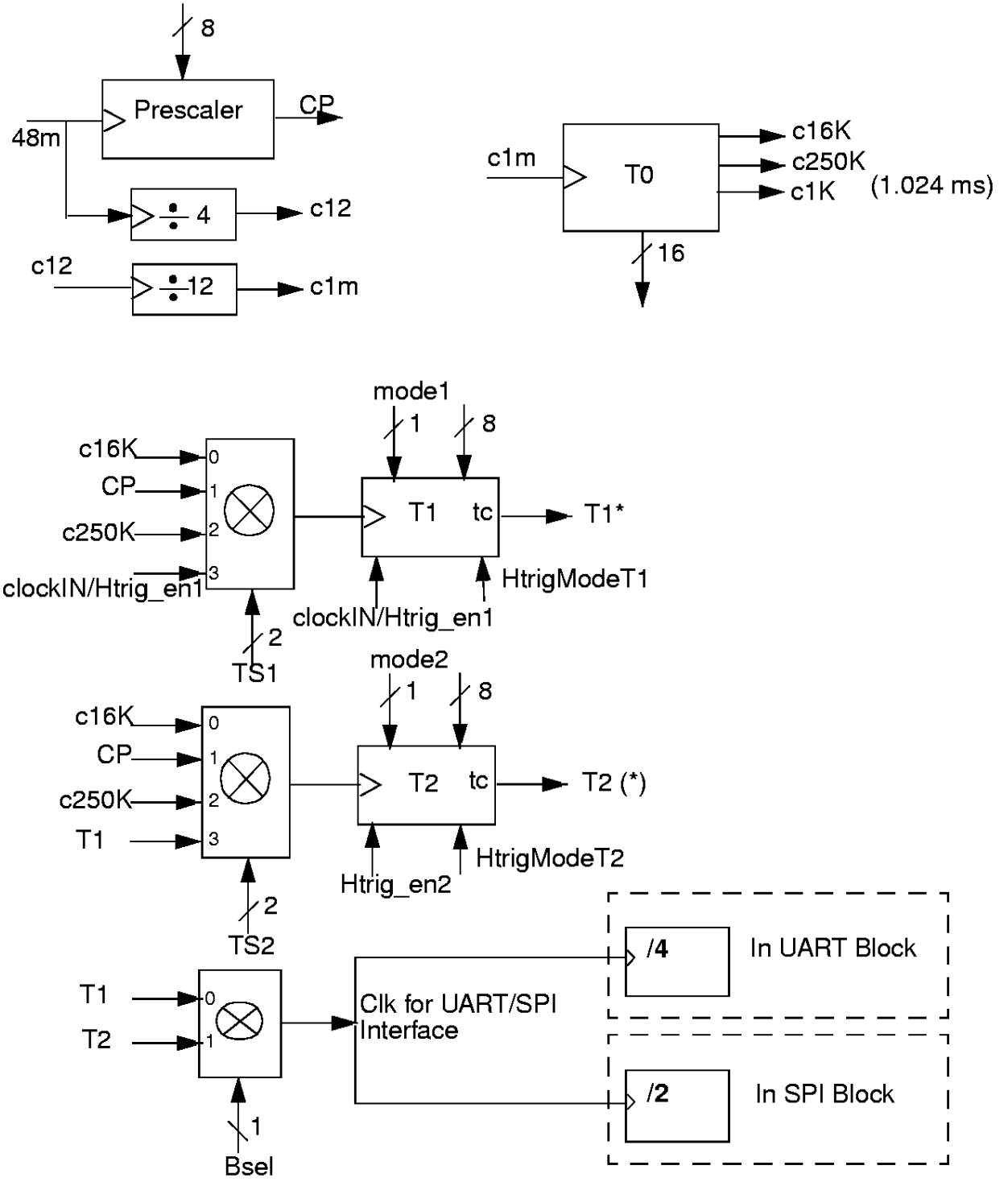


Figure 8-11. Timing Diagram for Timer1/Timer2 PWM (trigger_enable mode)



*Terminal Count T1 and T2 are available on the SPIO[6:4] pins related with the programmation of the bits[5:3] in the SPIO Configuration Register (0x0Ch).

Figure 8-12. Timer Block Diagram

9.0 DMA Controller

The DMA Controller in the CY7C642/3xx allows high speed data transfer without burdening the microcontroller. It is implemented to achieve a high throughput from/to the 1K Endpoint FIFO and external package pins. The DMA contains one channel for handling both the SPI/UART and the 1K FIFO registers (0x70h, 0x71h, 0x72, and 0x73h). Therefore, only one of them can use the DMA channel at a time, and it is the firmware's responsibility to manage this. As well, the DMA has one arbiter to handle the arbitration among SIE, 1K FIFO register access, SPI/UART, and BHHI (Bidirectional Hardware Handshake Interface). Who ever wins the arbitration would get the access to DMA bus (by default, the DMA bus is normally given to the BHHI). The internal DMA bus is totally separate from the internal microcontroller bus to allow their concurrent operation: the internal microcontroller would not be stalled while the 1K EP FIFO is being accessed. The SIE does not use the DMA channel, and generates its own address and byte count. The external microcontroller has two choices to access the internal 1 Kbyte EP FIFO: one is to use the 0x70h, 0x71h , 0x72h, 0x73h registers, and the other is to directly access it through the internal DMA bus.

10.0 1K Endpoint FIFO

Registers 0x70h, 0x71h, 0x72h, and 0x73h can be accessed by either the internal microcontroller or an external microcontroller (through BHHI) on the internal microcontroller bus. These registers are used to access the 1 Kbyte Endpoint FIFO which resides on the internal DMA bus. Registers 0x70h and 0x71h store the address where the next read/write data will be accessed through registers 0x72h and 0x73h. Register 0x70h holds the LSB and 0x71h has the MSB of the address. Registers 0x72h and 0x73h are the same with one difference: accessing 0x73h will automatically increment 0x70. When the internal microcontroller performs a read or write to register 0x72h or 0x73h, a "BRQ" will be immediately asserted to back off the microcontroller. The back off is only temporary stalling the internal microcontroller, but it does not stop the BHHI or SIE. If the BHHI or SIE want to access the internal microcontroller bus they can do that even if the "BRQ" is asserted. Once the 1K FIFO register (0x70h, 0x71h, 0x72h, and 0x73h) owns the internal DMA bus, it will release the "BRQ" and continue to hold the internal DMA bus until the internal microcontroller comes back and finishes the cycle. When an external microcontroller accesses register 0x73h through BHHI, the transaction happens on the internal DMA bus, not the internal microcontroller bus. In this way, the external microcontroller can continuously access the 1K FIFO without having to stall the internal microcontroller.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr7	Addr6	Addr5	Addr4	Addr3	Addr2	Addr1	Addr0

Figure 10-1. 1K FIFO Address 0 Register 0x70h

Bits [7:0] hold the LSB of the address accessed in the 1K FIFO.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
C2D Enable	Reserved	Reserved	Reserved	Reserved	Reserved	Addr9	Addr8

Figure 10-2. 1K FIFO Address 1 Register 0x71h

Bits [1:0] hold the MSB of the address accessed in the 1K FIFO.

When bit 7 is set to 1 the internal DMA Bus is parked on port 7x and no other masters will access DMA Bus. Firmware has to make sure not to stall other devices on DMA bus.

When bit 7 is set to 0 there is no parking operation, the DMA Bus is normally arbitrated.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0

Figure 10-3. 1K FIFO Data Register 0x72h

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0

Figure 10-4. 1K FIFO DMA Data Register 0x73h

Using the above set of registers, the internal microprocessor as well as an external microprocessor (through BHHI) can access the 1K FIFO1 directly.

Register 0x73h differs from 0x72h in two respects. First, accessing 0x73h will automatically increment address register 0x70h. Second, the data movement is done over internal DMA data bus by the DMA controller, not over internal microcontroller data bus as in the case of register 0x72h. This prevents the stall of the internal microcontroller during accesses to the 1K FIFO.

Register 0x73h does not increment register 0x71h, so when register 0x70 reaches FFh firmware has to update the register 0x71h.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr7	Addr6	Addr5	Addr4	Addr3	Addr2	Addr1	Addr0

Figure 10-5. DMA Stop Pointer Register 0x74h

In the DMA mode, for UART/SPI, this register is used to compare with register 0x70 to stop the DMA operation, the DMA starts at the address in register {0x71,0x70} and stops at address in register {0x71, 0x74}.

11.0 GPIO Bidirectional Hardware Handshaking Interface (BHHI)

The BHHI is a hardware asynchronous parallel interface that can transfer data to the internal FIFO, up to 15 Mbytes of transfer rate using the DMA controller.

The BHHI has two modes, one is the pulling mode which is used for supporting external micro. In the pulling mode, the external micro initiates a cycle with address and "strobe", CY7C642/3xx functions as a slave. In the second mode, the pushing mode, the CY7C642/3xx is a master on the interface and controls the arbitration to allow other master to do some limited accessing to CY7C642/3xx EP1–8's FIFO. In the pushing mode, the CY7C642/3xx's internal DMA engine generates address for accessing FIFO, register 0x71,70 contain the starting address, register70 increments after every accessing to FIFO. The DMA is stopped when register70 reaches the value stored in register 74. The pushing mode has two operations, forwarding and reversing, the forwarding operation is for 1284 EPP/ECP-Forwarding. The CY7C642/3xx drives "strobe" and samples "RDY" in forwarding operation, and acts like a master on the bus, while in reversing operation, the CY7C642/3xx samples "strobe" and drives "RDY" when the required transaction is done. When the BHHI modes are enabled, all the data movement is performed by the internal DMA engine, the internal micro is removed from the burden of moving data between FIFO and GPIO port. The CY7C642/3xx can be configured as one the following functions through BHHI:

Function	Modes	
1. As a Micro	Pushing Mode, Forwarding Operation	The CY7C642/3xx can be configured as a micro, which supports 8-bit I/O space and RAM space, though the access to the external RAM has to rely on firmware, no DMA operation.
2. As a USB controller with interface to an external micro	Pulling Mode	The operation is asynchronous and maximum bandwidth is 15 MByte/s. The external micro supplies the address and initiates the cycle.
3. As a 1284 EPP/ECP Compatible Host	Pushing Mode, Forwarding/ Reversing Operation	The CY7C642/3xx is a master with capability of providing arbitration to support another master. The maximum bandwidth is 15 MByte/s. The internal DMA engine is used to generate address for accessing the internal 1K FIFO.

Figure 11-1. BHHI Function-Mode

7	6	5	4	3	2	1	0
R/W	R/W	R/W		R/W	R/W	R/W	R/W
DMA Start	Forward/ Reverse	Out/In	Reserved	Enable $\overline{\text{FSTRB}}$ on Port1[2]	HHC2	HHC1	HHC0

Figure 11-2. GPIO BHHI Control Register 0x09h

Bit [2:0] define the different pin configuration on GPIO pin to support the BHHI.

Bit 3 enables the Forwarding Strobe ($\overline{\text{FSTRB}}$) on Port1[2] for 1284 compatibility.

Bit[7:5] is only effective when the BHHI is in pushing mode.

Bit5 defines the DMA operation's direction: 0 is input, 1 is output.

Bit 6 defines forwarding or reversing operation: 0 is reversing, 1 is forwarding.

Bit 7. The DMA would start only after bit 7 is set to 1, and bit 7 is only cleared (0) by hardware when DMA is done.

The GPIO BHHI Control Register will be at 00h after a reset.

HHC2,1,0	000	001	010	011(28-Pin)	100	101(28-Pin)
Port0[7:0]	GPIO	Data[7:0]	Data[7:0]	Data/Addr[7:0]	Data[7:0]	Data[7:0]
Port1[0]	GPIO	R$\overline{\text{W}}$	R$\overline{\text{W}}$	R$\overline{\text{W}}$	R$\overline{\text{W}}$/GPIO	R$\overline{\text{W}}$/GPIO
Port1[1]	GPIO	STRB	STRB	STRB	$\overline{\text{STRB}}$/GPIO	$\overline{\text{STRB}}$/GPIO
Port1[2]	GPIO	RDY	RDY	RDY	RDY/GPIO	RDY/GPIO
Port1[3]	GPIO	IRQ	IRQ	IRQ	IRQ	IRQ
Port1[7:4]	GPIO	GPIO	ACK[4:1]	X	GPIO	X
Port2[7:0]	GPIO	AX[7:0]	AX[7:0]	X	GPIO	X
Port3[0]	GPIO	IRA	IRA	IRA	IRA	IRA
Port3[1]	GPIO	Mode	Mode	Mode[0]	FSTRB	FSTRB
Port3[2]	GPIO	AX[8]	AX[8]	Mode[1]	FRDY	FRDY
Port3[3]	GPIO	AX[9]	AX[9]	GPIO	GPIO	GPIO
Port3[4]	GPIO	GPIO	ACK[5]	GPIO	GPIO	GPIO
Port3[5]	GPIO	ACK	ACK[6]	ACK	ACK	ACK
Port3[6]	GPIO	GPIO	ACK[7]	X	GPIO	X
Port3[7]	GPIO	GPIO	ACK[8]	X	GPIO	X

Figure 11-3. Bidirectional Hardware Handshake Interface Configuration

The pulling modes are those with HHC2=0; when HHC2=1, those are the pushing modes. Port1[2:1] is used as strobe and ready only in pulling or pushing (reversing operation) mode, the strobe will be ignored if in the wrong mode; Port3[2:1] are the strobe and ready for pushing (forwarding operation) mode.

HHC	Accessing	Mode	AX[9]	AX[8]	AX[7:0]
001, 010	1K EP FIFO	0	addr[9]	addr[8]	addr[7:0]
001, 010	256 RAM	1	0	x	addr[7:0]
001, 010	I/O Register	1	1	x	addr[7:0]
011	Data Transfer	0x	x	x	x
011	I/O Address	10	x	x	x
011	SRAM Address	11	x	x	x

Figure 11-4. Address Mapping for BHHI in Pulling Mode

In BHHI Pulling Modes (HHC=001,010), address is directly sent into the device from external micro; in 28-pin mode (HHC=011), address is multiplexed on the data bus, the mode[1:0] bits control the multiplexing. ACK(s) are used in pulling modes(HHC=001,001,011) to inform the external micro that a USB ACK is received or transmitted; ACK is set if any one of the 8 EPs has received data and returned an ACK packet; also, ACK is set if any one of the 8 EPs has transmitted data and received an ACK packet. Only in isochronous data transfer, the ACK is set if any one of the EPs has received 128 bytes or the last byte in the packet; the ACK is set if any one of the 8 EPs has transmitted 128 bytes or the last byte in the packet. This special feature could prevent FIFO overrun or underrun in a large-sized data transfer. The assertion of ACK(s) will remain until the external micro reads the ACK status register(0x18).

The maximum transfer rate through the Bidirectional Hardware Handshake Interface is 15 MByte/s; an asynchronous transfer technique is used. The Tcp and Tcr reflects the 1K SRAM's characteristic.

12.0 Special Purpose I/O (SPIO)

7	6	5	4	3	2	1	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reserved	Data Bit 6	Data Bit 5	Data Bit 4	Data Bit 3	Data Bit 2	Data Bit 1	Data Bit 0

Figure 12-1. SPIO Data Register 0x0Ah

Bits [6:0] are the data value of the 7 SPIO.

7	6	5	4	3	2	1	0
	W	W	W	W	W	W	W
Reserved	Enable Data Bit 6	Enable Data Bit 5	Enable Data Bit 4	Enable Data Bit 3	Enable Data Bit 2	Enable Data Bit 1	Enable Data Bit 0

Figure 12-2. SPIO Interrupt Enable Register 0x0Bh

Writing a logic 1 to a bit position enables the interrupt associated with that bit position.

Notice that the CY7C64211/12/13 will always require that the data bits SPIO[6,5,3:0], related to unconnected pin, be written with a 0.

7	6	5	4	3	2	1	0
		R/W	R/W	R/W	R/W	R/W	R/W
HtrigModeT2	HtrigModeT1	MC2	MC1	MC0	SPIO[3:0] Enable	SPIO Port Config. Bit 1	SPIO Port Config. Bit 0

Figure 12-3. SPIO Configuration Register 0x0Ch

Bit 7 and 6 set to 1 enable the trigger for Timer1 and 2 and overwrite the definition of SPIO [6:5] in *Table 12-1* (refer to section 8.2). Also an external reset pin, $\overline{\text{XRST}}$, can be provided as an option instead of SPIO[4] by setting bit 7 of the POR Configuration Register (0xF1h) to 1, in this case all SPIO[4] related functions defined in the table below are disabled.

TBit [5:3] select the mode configuration (see *Table 12-1*).

Table 12-1. SPIO[6:4] Configuration

MC2,MC1,MC0	000	001	010	011	100	101	110	111
SPIO[4]	G	CO	CO	G	G	G	T2	Reserved
SPIO[5]	G	CI	T1	CI	T1	CO	X	Reserved
SPIO[6]	G	T2	T2	CO	CO	T2	X	Reserved

In the above table, G stands for GPIO function, T1 is Timer1 output, T2 is Timer2 output, CI is clock input to Timer1, CO is a 12-MHz clock out.

Bit 2 set to 1 enables SPIO[3:0].

Bits [1:0] are the configuration bits for all the SPIO pins. See *Table 12-2*.

Table 12-2. Port Configurations

Port Configuration bits	Driver Mode	Interrupt Polarity
11	Resistive	falling edge
10	CMOS Output	disabled
01	Open Drain	falling edge
00	Open Drain	rising edge

7	6	5	4	3	2	1	0
R/W							
$\overline{\text{External}}$ Reset Enable	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Figure 12-4. POR Configuration Register 0xF1h

13.0 UART Interface

The Universal Asynchronous Receiver/Transmitter (UART) is a serial communications protocol available in the CY7C64311/12/13. It provides an asynchronous interface between a microcontroller and a serial communications channel. The UART receives parallel data from the microcontroller, converts it into serial data, and transmits the results to the send data output terminal TX. The UART receives serial data from an external source through the receiver data input RX, converts it into parallel data, and makes it available to the microcontroller. It receives and transmits data in a variety of configurations, including selective baud rates, 8 data bits, odd, even, or no parity, and 1 stop bit. In addition, the UART can be programmed to support DMA.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0

Figure 13-1. UART Transmit Data Register 0x30h

Register 0x30h is the transmitter holding register. It holds the next byte to be transmitted by the UART.

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R/W
Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0

Figure 13-2. UART Receive Data Register 0x31h

Register 0x31h contains the last complete data byte sample received by the UART.

7	6	5	4	3	2	1	0
		R/W	R/W	R/W	R/W	R/W	R/W
Reserved	Reserved	SPI Enable	Parity Even/Odd	Parity Enable	Transmit Interrupt Source	DMA UART Mode1	DMA UART Mode0

Figure 13-3. UART Control Register 0x32h

After system reset the UART interface is enabled. When bit 5 of register 0x32h is set to 1 the SPI interface is enabled.

Bit 4 selects the parity: 0 = Odd parity; 1 = Even parity.

Bit 3 selects the parity enable mode. When this bit is set to 1, parity generation in the transmitter and parity checking in the receiver are enabled. The parity bit is inserted between the last bit of the data byte and the stop bit.

Bit 2 selects the Transmitter Interrupt Source: 1 = transmitter interrupt occurs when the Transmitter Empty bit in UART Status Register (0x33h) is set; 0 = transmitter interrupt occurs when the Transmitter Holding Register Empty bit in register 0x33h is set.

Refer to the following table for the DMA modes of the UART (controlled through bits[1:0]):

DMA UART Mode bits	DMA Mode Selection
00	Disabled
01	Read Only
10	Write Only
11	Reserved

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
Clear to Send	Request to Send	Transmitter Empty	Transmitter Holding Empty	Framing Error	Parity Error	Overrun Error	Receiver Data Ready

Figure 13-4. UART Status Register 0x33h

Bit 7 reflects the status of the $\overline{CTS}$ signal: 1 = Clear to Send; 0 = Not Clear to Send.

Bit 6 reflects the status of the $\overline{RTS}$ signal: 1 = Request to Send; 0 = Not Request to Send.

When bit 5 is set to 1 by hardware, it indicates that the transmitter holding register and the transmitter shift register are both empty. This bit is set to logic 0 whenever the transmitter holding register (0x30h) or the transmitter shift register contains a data character.

When bit 4 is set to 1 by hardware, it indicates that the UART is ready to accept a new data byte from the microcontroller for transmission. An interrupt is generated when it is set to 1. This bit is cleared when the transmitter holding register (0x30h) is written to by the CPU.

When bit 3 is set by hardware, it indicates that the newly received data byte has an invalid stop bit. An interrupt is generated to the CPU when it is set to 1. The bit is cleared upon reading register 0x33h.

When bit 2 is set by hardware, it indicates that the newly received data byte has an incorrect parity bit. An interrupt is generated to the CPU when it is set to 1. The bit is cleared upon reading register 0x33h.

Bit 1 is set by hardware when the Receive Data Register (0x31h) is full and a new data byte is received in the shift register. An interrupt is generated to the CPU when it is set to 1. This bit is cleared upon reading register 0x33h.

Bit 0 is set by hardware when a data byte is received and transferred into the Receive Data Register (0x31h), and there is no error. An interrupt is generated when this bit is set to 1. This bit is cleared upon reading register 0x31h.

13.1 Non-DMA UART Mode

In non-DMA mode, the UART generates the Transmitter interrupt when either the Transmitter Empty or Transmitter Holding Empty bit is set, depending on the state of the Transmitter Interrupt Source Selection. The UART generates a Receiver Data Ready interrupt when the Receiver buffer is full. The Receive Error Interrupt is generated if the UART detects an error when receiving.

13.2 DMA UART Mode

In DMA mode, the UART generates DMA read and write requests to the DMA controller. When transmitting, the UART will generate a DMA write request if the Transmitter Holding Register is empty. When receiving, the UART will generate a DMA read request if the receiver buffer is full. The DMA done interrupt is ANDed with the UART transmitter interrupt or Receiver Data Ready interrupt to produce the interrupt to the microcontroller. The Receiver Error interrupt is sent directly to the microcontroller.

To begin a transfer, the firmware should first set up the DMA Stop Register (0x74h) and the1K FIFO Address Registers 0x70h, 0x71h to point to the Endpoint FIFO. The DMA Stop Register is used to compare with Register 0x70h to determine where the DMA should stop. The firmware writes to the UART Control register (0x32h) to set the DMA mode (read or write, read and write at the same time is not allowed for DMA). Once the DMA mode is set, the DMA is enabled. If the DMA is enabled, the firmware should ensure that registers 0x70h, 0x71h, 0x72h, and 0x73h should not be touched until the DMA is done.

13.3 UART Baud Rate

The UART in the CY7C64311/12/13 will support 300, 1200, 2400, 4800, 9600, 14.4k, 19.2k, 28.8k, 33.6k, 38.4k, 57.6k, 115k, 230k, 460k, 920k, and 1M baud rates. All baud rates are generated by pre-scaling the 48-MHz clock using the Prescaler and TimerX (X=Timer1, 2). Prescaler and TimerX are 8-bit, programmable registers. All baud rates above can be obtained by programming the Prescaler and TimerX. Since Prescaler and TimerX are programmable, different set of numbers can be programmed into the Prescaler and TimerX to have the same baud rate. *Table 13-1* lists the common baud rates supported in the CY7C64311/12/13, with the recommended combinations for the Prescaler and TimerX.

Table 13-1. UART Baud Rate Table

Baud Rate	Prescaler	TimerX	Sample-rate	Error
300	200	200	4	0%
1200	100	100	4	0%
2400	50	100	4	0%
4800	25	100	4	0%
9600	25	50	4	0%
14.4k	4	208	4	0.16%
19.2k	3	208	4	0.16%
28.8k	2	208	4	0.16%
33.6k	2	178	4	0.32%
38.4k	2	156	4	0.16%
57.6k	1	208	4	0.16%
115k	1	104	4	0.33%
230k	1	52	4	0.33%
460k	1	26	4	0.33%
920k	1	13	4	0.33%
1M	1	12	4	0%

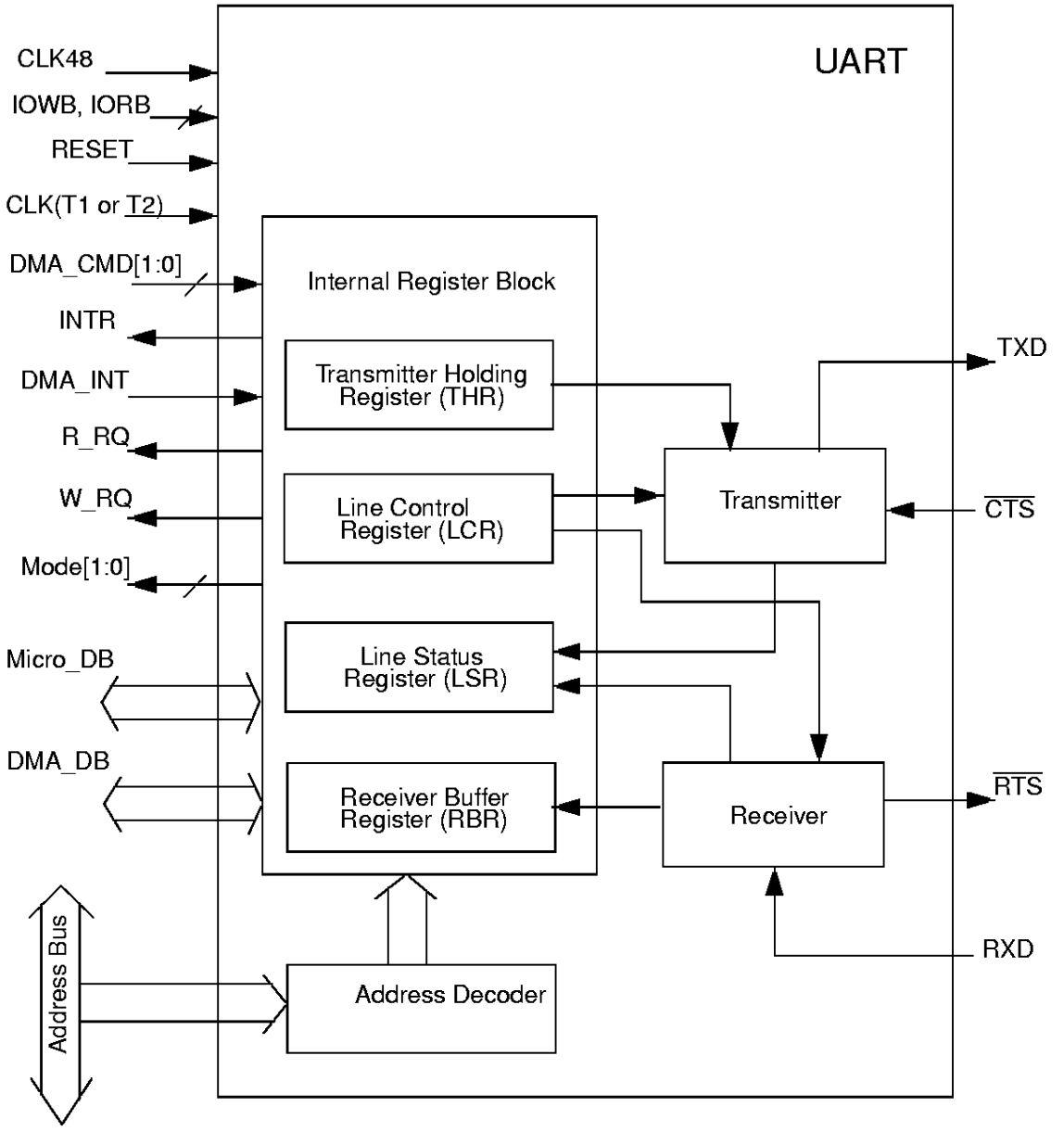


Figure 13-5. UART Block Diagram

14.0 SPI Interface

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol. It allows the microcontroller unit in the CY7C64311/12/13 to communicate with serial peripheral devices. The system can be used as a master or a slave device. In addition, the CY7C64311/12/13 can be configured as DMA SPI mode or Non-DMA SPI mode.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0

Figure 14-1. SPI Transmit Data Register 0x30h

Register 0x30h is the transmit data register. It holds the next byte to be transmitted by the SPI interface.

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R/W
Data7	Data6	Data5	Data4	Data3	Data2	Data1	Data0

Figure 14-2. SPI Receive Data Register 0x31h

Register 0x31h contains the last complete data byte sample received by the SPI interface.

7	6	5	4	3	2	1	0
		R/W	R/W	R/W	R/W	R/W	R/W
Reserved	CLK Enable for firmware $\overline{SS}$	SPI Enable	Master/Slave	Clock Polarity	Clock Phase	DMA SPI Mode1	DMA SPI Mode0

Figure 14-3. SPI Control Register 0x32h

Bit 6 set to 1 is used to enable the SPI clock before the firmware generates $\overline{SS}$ through the GPIO (Non-DMA SPI Mode).

When bit 5 of register 0x32 is set to 1, the SPI interface is enabled.

Bit 4 is used for the Master/Slave selection: 0 = Master mode; 1 = Slave mode.

Bit 3 selects the Clock Polarity: 1= Clock is active HIGH when SPI is idle; 0 = Clock is active LOW when SPI is idle.

Bit 2 selects the Clock Phase: 1 = Clock Phase 1; 0 = Clock Phase 0.

Refer to the following table for the DMA modes of the SPI Interface (controlled through bits[1:0] of register 0x32h):

DMA SPI Mode bits	DMA Mode Selection
00	Disabled
01	Read Only
10	Write Only
11	Reserved

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reserved	Reserved	Reserved	SPI Transmit empty	Mode Fault	Reserved	Overflow	SPI Receive Full

Figure 14-4. SPI Status Register 0x33h

Bit 4 of register 0x33h is set each time the SPI Transmit Data Register (0x30h) transfers a byte in the Shift Register. An interrupt request to the CPU will be generated if the SPI interrupt enable bit in the Global Interrupt Enable Register (0x20h) is set.

Bit 3 of register 0x33h is set to 1 by hardware if a logic zero occurs on the $\overline{SS}$ pin when SPI is configured as a master device, in a multimaster architecture, when Slave Select is generated through GPIO pin. Bit 3 is set to 1 also if a logic zero occurs on the $\overline{SS}$ pin before the data transmission when SPI is configured as a master and Slave Select is generated by hardware through $\overline{SS}$ pin. Again, bit 3 is set to 1 if $\overline{SS}$ goes HIGH during a transmission when SPI is configured as a slave device.

In DMA SPI mode Slave Select is always generated by hardware through $\overline{SS}$ pin.

In NON-DMA SPI mode Slave Select can be generated by firmware through the GPIO pin.

The setting of this bit will generate an interrupt to the CPU. This bit is cleared by reading the register and subsequently writing to the SPI Transmit Control Register.

Bit 1 is set if the firmware does not read the byte in the SPI Receive Data Register (0x31h) before the next bit[1] of data enters the Shift Register when SPI is configured as a slave. In an overflow condition, the byte already in the SPI Receive Data Register is unaffected, and the byte that is shifted-in will be lost (SPI configured as a slave device). The setting of this bit will generate an interrupt request to the CPU. This bit is cleared by reading this register and then reading the SPI Receive Data Register.

Bit 0 is set each time a byte transfers from the Shift Register to the Receive Data Register. An interrupt request is generated to the CPU. Any read of the SPI Receive Data Register clears this bit.

14.1 DMA SPI Mode

To begin a transfer, the firmware should first set up the DMA Stop Register (0x74h) and the 1K FIFO Address Registers 0x70h, 0x71h to point to the Endpoint FIFO. The DMA Stop Register is used to compare with Register 0x70h to determine where the DMA should stop. The firmware writes to the SPI Control register (0x32h) to set the DMA mode (read or write; read and write at the same time is not allowed for DMA). Once the DMA mode is set, the DMA is enabled. If the DMA is enabled, the firmware should ensure that registers 0x70h, 0x71h, 0x72h, and 0x73h should not be touched until the DMA is done.

For DMA SPI Read, when the DMA is enabled, the DMA asserts the "read_clock_enable" to SPI, and the SPI starts the transmission. The SPI only looks at "read_clock_enable" at the start of the receiving clock. Once the clock is started, the SPI ignores the "read_clock_enable" for the rest of the byte until the next byte starts. After the SPI receives one byte, it moves the data from the Shift Register to the Receive Data Register and asserts "DMA_receive_request" to the DMA. Then, the DMA moves the data to the proper Endpoint FIFO. When the receiving clock is started for the last byte, the DMA will deassert the "read_clock_enable". After the last byte is moved to the FIFO, the DMA will disable itself.

For DMA SPI Write, when the DMA is enabled, the DMA will check the "DMA_transmit_request" from SPI. If the "DMA_transmit_request" is HIGH, the DMA will move one byte of data to the SPI transmit data register. Then the SPI will load the data from the Transmit Data Register to the Shift Register to start the transmit. When the last byte is written in to the transmit data register, the DMA will disable itself.

The DMA does not support Read/Write duplex; the reason is that the DMA only has one pointer to the Endpoint FIFO. When the DMA is enabled, the interrupt will be asserted only when all bytes are received in the Endpoint FIFO or transferred. The interrupt will be asserted until the status bit is cleared. If the DMA is disabled, the interrupt is asserted by SPI when data is moved from the transmit data register to the shift register or when read data is moved from the Shift Register into the Receive Data Register.

The DMA operation is the same for both master and slave modes of SPI. Without proper enabling through the mode bits in the SPI control register (0x32h) and setting the proper stop address in register 0x74h, the DMA would not operate.

The maximum transfer rate in DMA SPI mode is 6 MHz.

14.2 Non-DMA SPI Mode

In the master mode, the transmission begins with the microcontroller writing a byte to the Transmit Data Register. If the Shift Register is empty, data is parallel loaded into the 8-bit Shift Register. After data is moved from the Transmit Data Register, the SPI sets the Transmitter Empty bit. If the Interrupt Enable bit is set, a interrupt will be generated to the microcontroller. The byte begins serially shifted out via the MOSI pin to the slave device. As the byte shifts out on the MOSI pin of the master, another byte shifts in from the slave on the master's MISO pin to the 8-bit Shift Register. Data is then parallel transferred to the Receive Data Register and the Receiver Full bit is set. If the Interrupt Enable bit is set, an interrupt will be generated to the microcontroller.

In the slave mode, the slave start logic receives a logic LOW at the $\overline{SS}$ pin and a clock input at the SCK pin from the master. This synchronizes the slave with the master. Data from the master is received serially at the slave MOSI pin and shifted into the 8-bit Shift Register. After a byte enters the Shift Register of the slave, it is transferred to the Receive Data Register, and the Receiver Full bit is set. When the master SPI starts a transmission, the data in the slave Shift Register begins shifting out on the MISO pin to the master SPI. Data to be transferred from the slave SPI to master SPI has to be loaded from its Transmit Data register to its Shift Register prior to the master starting the transmission. After data loads from the Transmit Data Register to the Shift Register, the Transmitter Empty bit is set. A new byte can be written to the Transmit Data Register for the next transmission.

The maximum transfer rate in non-DMA SPI mode is 0.5 MHz.

14.3 SPI Baud Rate

The UART in the CY7C64311/12/13 will support up to 6M baud rates. All baud rates are generated by prescaling the 48-MHz clock using the Prescaler and TimerX (X=Timer1, 2). Prescaler and TimerX are 8-bit, programmable registers. All baud rates above can be obtained by programming the Prescaler and TimerX. Since Prescaler and TimerX are programmable, different sets of numbers can be programmed into the Prescaler and TimerX to have the same baud rate. *Table 14-1* lists some of the baud rates supported in the CY7C64311/12/13, with the recommended combinations for the Prescaler and TimerX.

Table 14-1. SPI Baud Rate Table

Baud Rate	Prescaler	TimerX	Internal Divider
2M	6	2	2
4M	2	3	2
6M	1	4	2

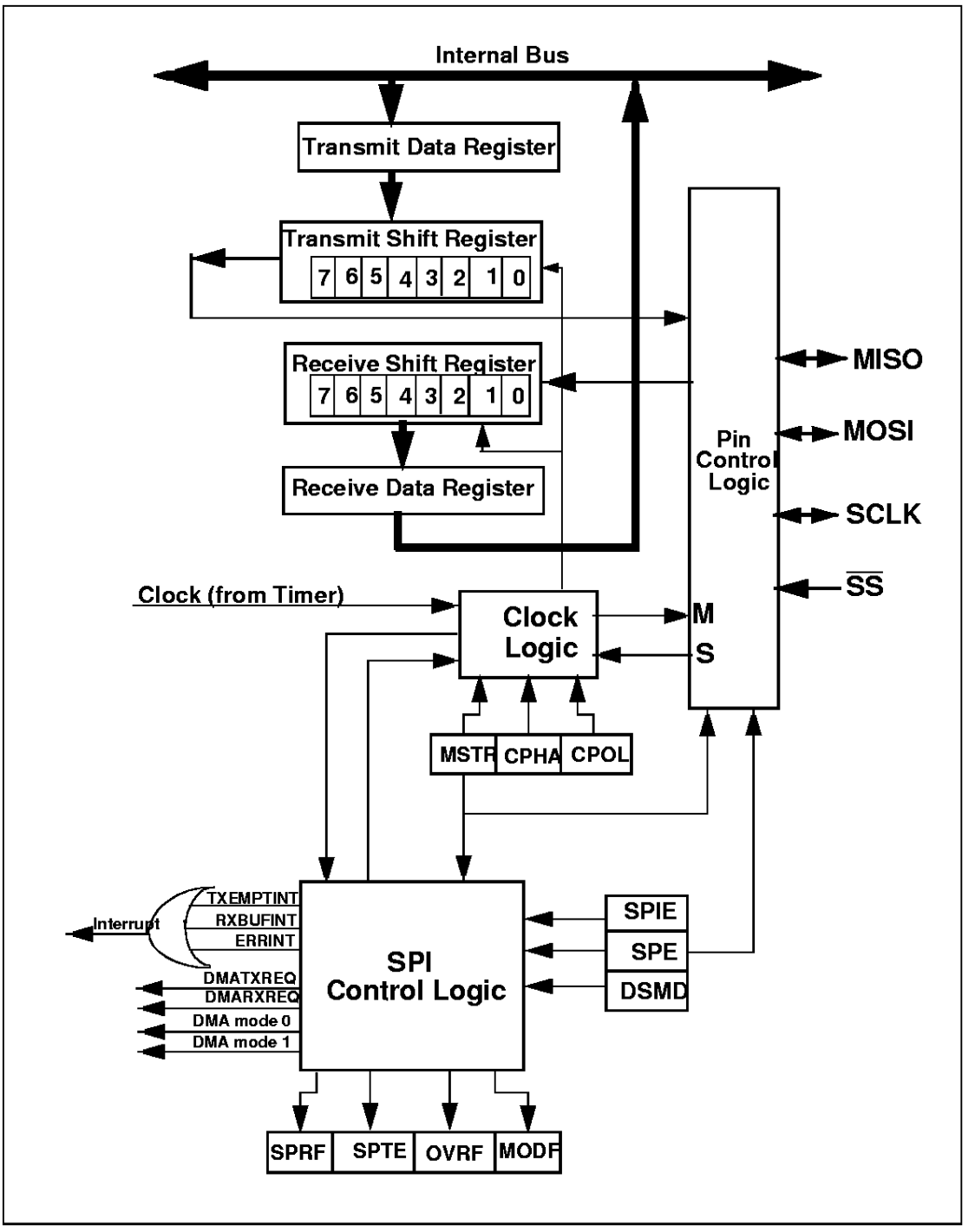


Figure 14-5. SPI Block Diagram

15.0 Processor Status and Control Register

7	6	5	4	3	2	1	0
R	R/W	R/W	R/W	R/W	R	R/W	R/W
IRQ Pending	Watch Dog Reset	USB Bus Reset	Power-on Reset	Suspend, Wait for Interrupt	Interrupt Mask	Single Step	Run

Figure 15-1. Processor Status and Control Register 0xFFh

The "run" (bit 0) is manipulated by the HALT instruction. When Halt is executed, the processor clears the run bit and halts at the end of the current instruction. The processor remains halted a reset (power on or watchdog).

The "single step" (bit 1) is provided to support a hardware debugger. When single step is set, the processor will execute one instruction and halt (clear the run bit).

The "Interrupt Mask" (bit 2) shows whether interrupts are enabled or disabled. The firmware has no direct control over this bit as writing a zero or one to this bit position will have no effect on interrupts. Instructions DI, EI, and RETI manipulate the internal hardware that controls the state of the interrupt mask bit in the Processor Status and Control Register.

Writing a 1 to "Suspend, wait for interrupts" (bit 3) will halt the processor and cause the microcontroller to enter the "suspend" mode that significantly reduces power consumption. The program counter that is pushed onto program stack by the interrupt service routine will be the instruction after the one that wrote 1 to bit 3 of the Processor Status and Control Register.

The "Power-on Reset" (bit 4) is only 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.

The "USB Bus Reset" (bit 5) will occur when a USB bus reset is received. The USB Bus Reset is a singled-ended zero (SE0) that lasts more than 8 microseconds. An SE0 is defined as the condition in which both the D+ line and the D– line are LOW at the same time. When the SIE detects this condition, the USB Bus Reset bit is set in the Processor Status and Control register and a USB Bus Reset interrupt is generated. Please note that this is an interrupt to the microcontroller and does not actually reset the processor, it only resets the USB Device Address Register 0x10h.

The "Watch Dog Reset" (bit 6) is set during a reset initiated by the watchdog timer. This indicates that the watchdog timer went for more than 8 msec between watch dog clears.

The "IRQ pending" (bit 7) indicates one or more of the interrupts has been recognized as active. The interrupt acknowledge sequence should clear this bit until the next interrupt is detected.

During power-on reset, the Processor Status and Control Register is set to 00010001, which indicates a power-on reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear).

During a watchdog reset, the Processor Status and Control Register is set to 01000001, which indicates a watchdog reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear).

16.0 Interrupts

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

7	6	5	4	3	2	1	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W
SOF Interrupt Enable	1.024-ms Timer0 Interrupt Enable	GPIO/SPIO Interrupt Enable	BHHI Interrupt Enable	UART/SPI Interrupt Enable	Timer2 Interrupt Enable	Timer1 Interrupt Enable	USB Bus Reset Interrupt Enable

Figure 16-1. Global Interrupt Enable Register 0x20h (read/write)

7	6	5	4	3	2	1	0
			R/W	R/W	R/W	R/W	R/W
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	EPi Interrupt Enable	EP0 Interrupt Enable

Figure 16-2. USB End Point Interrupt Enable Register 0x21h (read/write)

Pending interrupt requests are recognized during the last clock cycle of the current instruction. When servicing an interrupt, the hardware will first disable all interrupts by clearing the Interrupt Enable bit in the Processor Status and Control Register. Next, the interrupt latch of the current interrupt is cleared. This is followed by a CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector). The instruction in the interrupt table is typically a JMP instruction to the address of the Interrupt Service Routine (ISR). The user can re-enable 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 automatically stored onto the Program Stack by the CALL instruction as part of the interrupt acknowledge process. The user firmware is responsible for insuring that the processor state is preserved and restored during an interrupt. The PUSH A instruction should be used as the first command in the ISR to save the accumulator value and the POP A instruction should be used just before the RETI instruction to restore the accumulator value. The Program Counter, CF, and ZF are restored and interrupts are enabled when the RETI instruction is executed.

16.1 Interrupt Vectors

The Interrupt Vectors supported by the USB Controller are listed in *Table 16-1*. Although Reset is not an interrupt, per se, the first instruction executed after a reset is at PROM address 0x0000h—which corresponds to the first entry in the Interrupt Vector Table. Because the JMP instruction is 2 bytes long, the interrupt vectors occupy 2 bytes.

Table 16-1. Interrupt Vector Assignments

Interrupt Vector Number	ROM Address	Function
not applicable	0x0000	Execution after Reset begins here
1	0x0002h	USB Bus Reset
2	0x0004h	USB ENP0 Interrupt
3	0x0006h	USB ENPi(1–8) Interrupt
4	0x0008h	UART/SPI Interrupt
5	0x000Ah	Timer1 Interrupt
6	0x000Ch	Timer2 Interrupt
7	0x000Eh	GPIO/SPIO Interrupt
8	0x0010h	SOF Interrupt
9	0x0012h	Timer0 1.024-ms Interrupt
10	0x0014h	BHHI Interrupt

16.2 Interrupt Latency

Interrupt latency can be calculated from the following equation:

Interrupt Latency = (Number of clock cycles remaining in the current instruction) + (10 clock cycles for the CALL instruction) + (5 clock cycles for the JMP instruction)

For example, if a 5 clock cycle instruction such as JC is being executed when an interrupt occurs, the first instruction of the Interrupt Service Routine will execute a min. of 16 clocks (1+10+5) or a max. of 20 clocks (5+10+5) after the interrupt is issued. Remember that the interrupt latches are sampled at the rising edge of the last clock cycle in the current instruction.

16.3 USB Bus Reset Interrupt

The USB Bus Reset Interrupt is asserted after a USB bus reset condition is detected, specifically the interrupt is not logged until the USB Bus Reset event goes away. The USB Controller recognizes a USB Bus Reset when a Single Ended Zero (SE0) condition persists for at least 8–16 µs (the Reset may be recognized for an SE0 as short as 8 µs, but will always be recognized for an SE0 longer than 16 µs). SE0 is defined as the condition in which both the D+ line and the D– line are LOW. Bit 5 of the Status and Control Register will be set to record this event. If the USB Bus Reset happens (D+ and D– detected LOW for more than 8 µs) while the device is suspended the suspend condition will be cleared and the clock oscillator will be restarted. The USB Bus Reset will clear the USB Device Address Register (0x10h) after the 8- to 16-µs event.

16.4 BHHI Interrupt

The BHHI Interrupt is asserted when the IRQ line in the Bidirectional Hardware Handshaking Interface is asserted by the external microcontroller. The polarity of the BHHI interrupt can be selected programming the Port1 configuration bits in the GPIO Configuration Register (0x08h).

16.5 USB Endpoint Interrupts

There are two USB endpoint interrupts, one for Endpoint 0 and the other for Endpoint i (1–8). A USB Endpoint Interrupt is generated after the USB host writes to a USB Endpoint FIFO and an ACK packet is sent back to the USB host or after the USB controller sends a packet to the USB host and an ACK packet is received back from the USB host.

16.6 UART/SPI Interrupt

The UART/SPI interrupt is asserted after transmit/receive operations.

16.7 Timer Interrupts

There are three timer interrupts: Timer1 interrupt, Timer2 interrupt, and the 1.024-ms Timer0 interrupt. The user should disable timer interrupts before going into the suspend mode to avoid possible conflicts between servicing the interrupts first or the suspend request first.

16.8 GPIO/SPIO Interrupt

Each of the GPIO/SPIO pins can generate an interrupt, if enabled. The interrupt polarity can be programmed for each GPIO/SPIO port as part of the GPIO/SPIO configuration. All of the GPIO/SPIO pins share a single interrupt vector, which means the firmware will need to read the GPIO/SPIO ports with enabled interrupts to determine which pin or pins caused an interrupt.

Please note that if one port pin triggered an interrupt, no other port pins can cause a GPIO/SPIO interrupt until that port pin has returned to its inactive (non-trigger) state or its corresponding port interrupt enable bit is cleared. The USB Controller does not assign interrupt priority to different port pins and the Port Interrupt Enable Registers are not cleared during the interrupt acknowledge process.

16.9 Start Of Frame Interrupt (SOF)

SOF interrupt is generated by the SIE every time a SOF packet is received on the USB lines.

17.0 USB Overview

The USB hardware includes a USB function with one upstream port. The USB port interfaces to the microcontroller through a high-speed serial interface engine (SIE).

17.1 USB Serial Interface Engine (SIE)

The SIE allows the microcontroller to communicate with the USB host. 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
- TOKEN type identification
- Address checking

Firmware is required to handle the rest of the USB interface with the following tasks:

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

17.2 USB Enumeration

The enumeration sequence is shown below:

1. The host computer sends a SETUP packet followed by a DATA packet to USB address 0 requesting the Device descriptor.
2. The USB Controller decodes the request and retrieves its Device descriptor from the program memory space.
3. The host computer performs a control read sequence and the USB Controller responds by sending the Device descriptor over the USB bus.
4. After receiving the descriptor, the host computer sends a SETUP packet followed by a DATA packet to address 0 assigning a new USB address to the device.
5. The USB Controller stores the new address in its USB Device Address Register after the no-data control sequence is complete.

6. The host sends a request for the Device descriptor using the new USB address.
7. The USB Controller decodes the request and retrieves the Device descriptor from the program memory.
8. The host performs a control read sequence and the USB Controller responds by sending its Device descriptor over the USB bus.
9. The host generates control reads to the USB Controller to request the Configuration and Report descriptors.
10. The USB Controller retrieves the descriptors from its program space and returns the data to the host over the USB.
11. Enumeration is complete after the host has received all the descriptors.

17.3 USB Status and Control

USB status and control is regulated by the USB Status and Control Register located at I/O address 0x1Fh as shown in *Figure 17-1*. This is a read/write register. All reserved bits must be written to zero. All bits in the register are cleared during reset.

7	6	5	4	3	2	1	0
R/W	R/W	R	R	R/W	R/W	R/W	R/W
Speed	LSEnable	D+	D–	Bus Activity	Control Bit 2	Control Bit 1	Control Bit 0

Figure 17-1. USB Status and Control Register 0x1Fh

Bit 7 will be used to switch the CY7C642/3xx into/out of low speed or full speed device mode (0 = full speed, 1 = low speed).

Bit 6, LSEnable, will be set only when the CY7C642/3xx works as a low speed device and must to be cleared for full speed device operation.

Bit 4, Bus Activity, is a "sticky" bit that indicates if any non-idle USB event has occurred on the upstream USB port. The user firmware should check and clear this bit periodically to detect any loss of bus activity. Writing a 0 to the Bus Activity bit clears it while writing a 1 preserves the current value. In other words, the firmware can clear the Bus Activity bit, but only the SIE can set it. The 1.024-ms timer interrupt service routine is normally used to check and clear the Bus Activity bit.

The following table shows how the control bits 2:0 are encoded for this register.

Control Bits	Control action
000	Not forcing (SIE controls driver)
001	Force D+[0] HIGH, D–[0] LOW
010	Force D+[0] LOW, D–[0] HIGH
011	Force SE0 (D+[0] LOW, D–[0] LOW
100	Reserved
101	Force D+[0] LOW, D–[0] HiZ
110	Force D+[0] HiZ, D–[0] LOW
111	Force D+[0] HiZ, D–[0] HiZ

18.0 USB Device

The USB device includes up to nine endpoints: EP0, and EP1 to EP8. Endpoint 0 (EP0) allows the USB host to recognize, set up, and control the device. In particular, EP0 is used to receive and transmit control (including set-up) packets.

18.1 USB Address

The USB Controller provides one USB Device Address Register. The USB Device Address Register contents are cleared during a reset, setting the USB device address to zero and marking this address as disabled. *Figure 18-1* shows the format of the USB Address Register.

Device Address Enable	Device Address Bit 6	Device Address Bit 5	Device Address Bit 4	Device Address Bit 3	Device Address Bit 2	Device Address Bit 1	Device Address Bit 0

Figure 18-1. USB Device Address Registers 0x10h (read/write)

Bit 7 (Device Address Enable) in the USB Device Address Register must be set by firmware before the serial interface engine (SIE) will respond to USB traffic to this address. The Device Address in bits[6:0] are set during the USB enumeration process to an address assigned by the USB host that does not equal zero.

18.2 USB Device Endpoints

In the CY7C642/3xx, EP0 (Endpoint 0) has its FIFO in the 256 byte RAM. The data movement to and from this FIFO is done by the SIE on the internal microcontroller bus after some proper initialization by the firmware. The FIFOs of EP1 to EP8 are in the 1-Kbyte RAM. The data movement is done by SIE on the internal DMA bus. The SIE will auto detect the End Point number to direct the data to/from the microcontroller or the DMA bus.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
EP4X0	EP3X0/ EP2X1	EP2X0	EP1X1	EP1X0	FFC2	FFC1	FFC0

Figure 18-2. USB Endpoint i FIFO Control Register 0x19h

Bits [2:0] select the Endpoint Configuration according to *Figure 18-3*, below.

Bits [7:3] are the extension of the USB Device EP 1–4 Counter Register bits for Endpoint size greater than 128 bytes.

For example if the 1023 bytes size FIFO is chosen for EP1, 10 bits are requested to indicate to the number of bytes to be transmitted or received during IN/OUT transfer; bits 7 to 0 are available in the EP1 Counter Register (0x40h) bits 9 and 8 are respectively EP1X1 and EP1X0 of the USB Endpoint i FIFO Control Register.

FFC	000	001	010	011	100	101	110	111
EP1	128	256	256	256	256	Reserved	1023	1023
EP2	128	128	256	256	256	Reserved	(*512)	(*128)
EP3	128	128	128	256	256	Reserved	X	X
EP4	128	128	128	128	256	Reserved	X	X
EP5	128	128	128	128	X	Reserved	X	X
EP6	128	128	128	X	X	Reserved	X	X
EP7	128	128	X	X	X	Reserved	X	X
EP8	128	X	X	X	X	Reserved	X	X

Figure 18-3. 1K FIFO Mapping

The FIFO size of each Endpoint (1–8) is determined through the Endpoint i FIFO Control Register(0x19h). Once configured, the SIE will automatically divide the RAM into the number of FIFOs as programmed. When any USB transmission/reception occurs, the SIE starts at location 0 of the targeted EP's FIFO and goes up. Based on the mapping selected from above table, the number of Endpoints can be reduced in order to support larger Endpoint FIFO sizes. The total size of the FIFO for all of the Endpoints is 1 Kbyte (i.e., the sum of all Endpoint FIFO sizes can not exceed 1 Kbyte). The maximum FIFO size supported for one Endpoint is 1 Kbyte.

The addressing of the 1K FIFO is done based on the mapping selected in the above table. For example, if FFC is set to 000, 8 bidirectional Endpoints are supported, each with a 128-byte dedicated FIFO. The Endpoint 1 FIFO starts at address 0x000 and ends at 0x07F. The Endpoint 2 FIFO starts at address 0x080 and ends at 0x0FF. The Endpoint 8 FIFO starts at 0x380 and ends at 0x3FF. In the mapping of FFC=110 or 111, the Endpoint 1 FIFO starts at 000x and ends at 0x3FE. In these mappings, there is an option to use Endpoint 2. In the first case (*512), the Endpoint 2 FIFO starts at 0x200 and ends at 0x3FF, and in the second case (*128), the Endpoint 2 FIFO starts at 0x380 and ends at "0x3FF". In these two cases, the firmware has to make sure that the combined size of Endpoint 1 and Endpoint 2 is smaller than 1 Kbyte.

In the mode register of each Endpoint, there is an enable bit for back-to-back mode. When this is enabled, the next USB transmission/reception will not start at location 0 of the Endpoint's FIFO. Instead, it will start where last transmission/reception stopped. This allows the FIFO to be used more efficiently, especially for bulk or isochronous transactions where back-to-back would be an advantageous feature. The firmware has to take care of managing FIFO data in order to prevent an overflow condition.

Normally an interrupt will be generated after one transaction is done and an ACK is received/sent. When the back-to-back mode is enabled for an Endpoint, and FIFO size is larger than 128 bytes, an interrupt will be generated every 128 bytes to allow the external microcontroller to move data earlier. This interrupt will be sent to both the internal CPU and external microcontroller. The external microcontroller will receive this information on one of the ACK pins depending on the BHHI interface used. In the later

case (where one of the ACK pins is asserted), the external microcontroller has to poll the ACK Status Register (0x18h) to find out if the assertion of ACK is a 128-byte early interrupt or an end-of-packet ACK interrupt. If an ACK is asserted and the corresponding status bit of the ACK Status Register is not set, then an 128-byte early interrupt has occurred.

All USB devices are required to have an Endpoint 0 (EP0) that is used to initialize and control the USB device. Endpoint 0 provides access to the device configuration information and allows generic USB status and control accesses. Endpoint 0 is bidirectional as the USB controller can both receive and transmit data.

The Endpoint Mode Registers are cleared during reset. The EP0 Endpoint Mode Register uses the format shown in *Figure 18-4*.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ENDP0 SET-UP	ENDP0 IN	ENDP0 OUT	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0

Figure 18-4. USB Device EP0 Mode Register 0x12h

Bits[7:5] in the Endpoint 0 Mode Register are “sticky” status bits that are set by the SIE to report the type of token that was most recently received by to the corresponding device address. The sticky bits must be cleared by firmware as part of the USB processing.

The ‘Acknowledge’ bit is set whenever the SIE engages in a transaction that completes with an ‘ACK’ packet.

The ‘set-up’ PID status (bit[7]) is forced HIGH from the start of the data packet phase of the set-up transaction, until the start of the ACK packet returned by the SIE. The CPU is prevented from clearing this bit during this interval, and subsequently until the CPU first does a IORD to this endpoint 0 mode register.

Bits[6:0] of the Endpoint 0 Mode Register are locked from CPU IOWR operations only if the SIE has updated one of these bits, which the SIE does only at the end of a packet transaction (set-up... Data... ACK, or Out... Data... ACK, or In... Data... ACK). The CPU can unlock these bits by doing a subsequent I/O read of this register.

Firmware must do a IORD after an IOWR to an Endpoint 0 Register to verify that the contents have changed and that the SIE has not updated these values.

While the ‘set-up’ bit is set, the CPU cannot write to the endpoint zero DMA buffers. This prevents an incoming set-up transaction from conflicting with a previous In data buffer filling operation by firmware.

The mode bits (bits [3:0]) in an Endpoint Mode Register control how the endpoint responds to USB bus traffic. The mode bit encoding is shown in *Table 18-1*.

Table 18-1. Endpoint Mode Encoding Definitions

Mode	Encoding	Setup	IN	OUT
Disabled	0000	ignore	ignore	ignore
NAK both	0001	accept	NAK	NAK
Status Out Only	0010	accept	STALL	check
Stall	0011	accept	STALL	STALL
Ignore	0100	accept	ignore	ignore
Isochronous Out	0101	ignore	ignore	always
Status In Only	0110	accept	TX 0	STALL
Isochronous In	0111	ignore	TX cnt	ignore
NAK Out	1000	ignore	ignore	NAK
ACK Out	1001	ignore	ignore	ACK
NAK Out – Status In	1010	accept	TX 0	NAK
ACK Out – Status In	1011	accept	TX 0	ACK
NAK In	1100	ignore	NAK	ignore
ACK In	1101	ignore	TX cnt	ignore
NAK In – Status Out	1110	accept	NAK	check
ACK In – Status Out	1111	accept	TX cnt	check

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
EP8 Stall Mode	EP7 Stall Mode	EP6 Stall Mode	EP5 Stall Mode	EP4 Stall Mode	EP3 Stall Mode	EP2 Stall Mode	EP1 Stall Mode

Figure 18-5. USB Device EPi (1–8) Stall Mode Register 0x13h

The USB EPi Stall Mode Register controls how SIE will respond to an IN or OUT token. When a particular Endpoint is set in "ACK-In" mode, and the related stall mode bit is set to 1 (bits 0–7), the SIE will ignore SETUP and OUT towards to Endpoint, and instead of an ACK IN, it will stall in; this creates a "stall-IN-only mode".

Similarly, when the Endpoint is set in "ACK-Out" mode, and the stall mode bit is set to 1, the SIE will ignore SETUP and IN towards to Endpoint, and instead of ACK OUT, it will stall in; this creates a "stall-OUT-only" mode.

This register only affects Endpoint (1–8), it does NOT affect Endpoint0.

7	6	5	4	3	2	1	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
DTOG	DVAL	B2B Enable	ACK	Mode3	Mode2	Mode1	Mode0

Figure 18-6. USB Device EPi (1–8) Mode Registers 0x41h, 0x43h, 0x45h, 0x47h, 0x49h, 0x4Bh, 0x4Dh, 0x4Fh

The USB EPi Mode Registers are similar to the EP0 Mode Register, with the exception of the B2B Enable bit (bit 5). This bit enables back-to-back transactions for the corresponding endpoint.

The format of the Endpoint Device Counter Registers is shown below:

7	6	5	4	3	2	1	0
R/W	R/W			R/W	R/W	R/W	R/W
Data Toggle	Data Valid	Reserved	Reserved	Count 3	Count 2	Count 1	Count 0

Figure 18-7. USB Device EP0 Counter Register 0x11h

Bits 0 to 3 indicate the number of data bytes to be transmitted during an IN packet (valid values are 0 to 8) or received during an OUT or SETUP packet (valid values are 0 to 10). The Data Valid bit (bit 6) is used for OUT and SETUP tokens only. Data 0/1 Toggle bit (bit 7) selects the DATA packet's toggle state: 0 for DATA0, 1 for DATA1. This register is locked from CPU IOWR by the SIE whenever it is updated by the SIE during the EOP phase of a Setup or Out USB transaction, and is not unlocked until a subsequent IORD of this register by the CPU.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Count7	Count6	Count5	Count4	Count3	Count2	Count1	Count 0

Figure 18-8. USB Device EPi (1–8) Counter Registers 0x40h, 0x42h, 0x44h, 0x46h, 0x48h, 0x4Ah, 0x4Ch, 0x4Eh

Bits[7:0] of the EPi Counter Registers indicate the number of data bytes to be transmitted or received during IN/OUT transfers. For Endpoint sizes greater than 128 bytes, the EPiX0/1 bits in register 0x19h provide the additional count bits.

Below is the EPi (1–8) Acknowledge Status register:

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
EP8_ACK	EP7_ACK	EP6_ACK	EP5_ACK	EP4_ACK	EP3_ACK	EP2_ACK	EP1_ACK

Figure 18-9. USB Device EPi ACK Status Register 0x18h

19.0 Absolute Maximum Ratings

Storage Temperature ..... –65°C to +150°C
Ambient Temperature with Power Applied ..... –0°C to +70°C
Supply Voltage on V_{CC} Relative to V_{SS} ..... –0.5V to +7.0V
DC Input Voltage ..... –0.5V to +V_{CC}+0.5V
DC Voltage Applied to Outputs in High Z State ..... –0.5V to +V_{CC}+0.5V
Power Dissipation ..... 300 mW
Static Discharge Voltage ..... >2000V
Latch-up Current ..... >200 mA

20.0 DC Characteristics

Fosc = 6 MHz; Operating Temperature = 0 to 70° C, V_{CC} = 4.0 to 5.25 Volts

	Parameter	Min	Max	Units	Conditions
	General				
V_{CC}	Operating Voltage	4.0	5.25	V	
I_{CC}	V_{CC} Operating Supply Current		40	mA	
I_{SB1}	Supply Current - Suspend Mode		10	µA	Oscillator off, D– > Voh min
V_{pp}	Programming Voltage (disabled)	–0.4	0.4	V	
T_{pll}	PLL Start-up Interval		1.024	ms	V_{CC} = 5.0V
C_{xtal}	Internal Crystal Capacitance/pin		30	pF	V_{CC} = 5.0V, XTALIN,XTALOUT
I_{il}	Input Leakage Current		1000	nA	Any pin, V_{CC}, V_{SS}
I_{ref}	Max Load on Vref		500	µA	
t_{watch}	Watch Dog Timer Period	8.192	14.336	ms	
	Power On Reset				
t_{vccs}	V_{CC} Reset Slew	0.01	100	ms	Linear ramp V_{CC}: 0 to V_{CC}
	USB Interface				
I_{LO}	Hi-Z State Data Line Leakage	–10	+10	µA	0 V < Applied Voltage < 3.3 V
V_{DI}	Differential Input Sensitivity	0.2		V	|(D+)–(D–)| and *Figure 21-2*
V_{CM}	Differential Common Mode Range	0.8	2.5	V	Includes V_{DI} range
V_{SE}	Single-Ended Receiver Threshold	0.8	–2.0	V	
V_{OL}	Static Output LOW		0.3	V	R_L of 15 kΩ to 3.6V
V_{OH}	Static Output HIGH	2.8	3.6	V	R_L of 15 kΩ to GND
C_{IN}	Transceiver Capacitance		20	pF	Pin to GND
C_{RPB}	Root Port Bypass Capacitance	1.0	10.0	µF	Vbus to GND[6]
R_{PU}	Bus Pull-up Resistor on Root Port	1.425	1.575	kΩ	(1.5 kΩ ± 5%)
	General Purpose I/O and Special Purpose I/O				
Rup	Pull-up resistance	8K	20K	Ohms	All ports (defines Voh)
V_{IH}	Input High Voltage	2		V	
V_{IL}	Input Low Voltage		0.8	V	
V_{OH}	Output High Voltage	2.4		V	I_{OH}= –4 mA
V_{OL}	Output Low Voltage		0.4	V	I_{OL}= 4 mA

21.0 Switching Characteristics

Parameter	Description	Min.	Max.	Unit
	Clock			
f_{OSC}	Clock Rate	6 ± 0.25%		MHz
t_{CH}	Clock HIGH time	0.45 t_{CYC}		ns
t_{CL}	Clock LOW time	0.45 t_{CYC}		ns
	USB Driver Characteristics			
t_r	Transition Rise Time[2]	4	20	ns
t_f	Transition Fall Time[2]	4	20	ns
t_{rfm}	Rise / Fall Time Matching; (t_r/t_f)	90	110	%
V_{CRS}	Output Signal Crossover Voltage	1.3	2.0	V
Z_{DRV}	Driver Output Resistance (Steady State Drive)	6	22	Ω
	USB Data Signal Timing			
tdrate	Full Speed Date Rate	12 ± 0.25%		Mb/s
tframe	Frame Interval	1.0 ± 0.05%		ms
T_{DJ1}	Source Differential Driver Jitter To Next Transition (Notes 4 and 5 and *Figure 21-6*)	–3.5	3.5	ns
T_{DJ1}	Source Differential Driver Jitter For Paired Transitions (Notes 4 and 5 and *Figure 21-6*)	–4.0	4.0	ns
T_{EOPT}	Source EOP Width (Note 5 and *Figure 21-5*)	160	175	ns
T_{DEOP}	Differential to EOP Transition Skew (Note 5 and *Figure 21-5*)	–2	5	ns
T_{JR1}	Receiver Data Jitter Tolerance To Next Transition (Note 5 and *Figure 21-4*)	–18.5	18.5	ns
T_{JR2}	Receiver Data Jitter Tolerance For Paired Transitions (Note 5 and *Figure 21-4*)	–9	9	ns
T_{EOPR1}	EOP Width at Receiver Must Reject as EOP (Note 5 and *Figure 21-4*)	40		ns
T_{EOPR2}	EOP Width at Receiver Must Accept as EOP (Note 5 and *Figure 21-5*)	82		ns
	Bidirectional Hardware Handshake Interface (BHHI) Timing (Pulling Mode)			
t_{CP}	Command Pulse Width[9]	46	250	ns
t_{CR}	Command Recovery Time[9]	20		ns
t_{AS}	Address Set-up	5		ns
t_{AH}	Address Hold	5		ns
t_{WDS}	Write Data Set-up	0		ns
t_{WDH}	Write Data Hold	2		ns
t_{RDV}	RDY Assertion from the Assertion of $\overline{STRB}$	36		ns
t_{RDR}	Read Data Ready	0		ns
t_{RDH}	Read Data Hold	2		ns
t_{RDY}	Wait State Asserted by Microcontroller from $\overline{STRB}$[7]	22		ns
t_{RDY}	Wait State Asserted by Microcontroller from $\overline{STRB}$[8]	170		ns
	Bidirectional Hardware Handshake Interface (BHHI) Timing (Pushing Mode, Forwarding Operation)			
t_{CP}	Command Pulse Width[9]	21		ns

Parameter	Description	Min.	Max.	Unit
t_{CR}	Command Recovery Time[9]	42		ns
t_{CH}	$\overline{STRB}$ Assertion from the Deassertion of RDY	0		ns
t_{WDS}	Write Data Set-up	0		ns
t_{WDH}	Write Data Hold	2		ns
t_{RDV}	$\overline{STRB}$ Deassertion from the Assertion RDY	1	25	ns
t_{RDR}	Read Data Ready	0		ns
t_{RDH}	Read Data Hold	2		ns
	Bidirectional Hardware Handshake Interface (BHHI) Timing (Pushing Mode, Reversing Operation)			
t_{CP}	Command Pulse Width[9]	46	250	ns
t_{CR}	Command Recovery Time[9]	20		ns
t_{CH}	RDY Deassertion from the Deassertion of $\overline{STRB}$	5		ns
t_{AS}	Address Set-up	5		ns
t_{AH}	Address Hold	5		ns
t_{WDS}	Write Data Set-up	0		ns
t_{WDH}	Write Data Hold	2		ns
t_{RDV}	RDY Assertion from the Assertion of $\overline{STRB}$	36		ns
t_{RDR}	Read Data Ready	0		ns
t_{RDH}	Read Data Hold	2		ns
	SPI Timing[10]			
$t_{cyc(M)}$	Cycle Time - Master[11] (max. 6 MHz)	167		ns
$t_{cyc(S)}$	Cycle Time - Slave[11]	167		ns
t_{LEAD}	Enable Lead Time	15		ns
t_{LAG}	Enable Lag Time	15		ns
$t_{w(SCKH)M}$	Clock (SCK) HIGH Time - Master	$0.45t_{cyc(M)}$		ns
$t_{w(SCKH)S}$	Clock (SCK) HIGH Time - Slave	$0.45t_{cyc(S)}$		ns
$t_{w(SCKL)M}$	Clock (SCK) LOW Time - Master	$0.45t_{cyc(M)}$		ns
$t_{w(SCKL)S}$	Clock (SCK) LOW TIME - Slave	$0.45t_{cyc(S)}$		ns
$t_{SU(M)}$	Data Set-up Time, Inputs - Master	5		ns
$t_{SU(S)}$	Data Set-up Time, Inputs - Slave	5		ns
$t_{H(M)}$	Data Hold Time, Inputs - Master	5		ns
$t_{H(S)}$	Data Hold Time, Inputs - Slave	10		ns
$t_{A(CP0)}$	Access Time (Clock Phase = 0) - Slave[12]		20	ns
t_{DIS}	Slave Disable Time (Hold Time to High-Impedance State)[13]		20	ns
$t_{V(M)}$	Data Valid Time after Enable Edge - Master		15	ns
$t_{V(S)}$	Data Valid Time after Enable Edge - Slave		20	ns
$t_{HO(M)}$	Data Hold Time, Outputs, after Enable Edge - Master	2		ns
$t_{HO(S)}$	Data Hold Time, Outputs, after Enable Edge - Slave	2		ns
	Timer1 and Timer Clock Terminal Count Mode			
t_{1CLK}	Input Clock Rate (max. 62.5 MHz)	16		ns
t_{1H}	Timer1 Clock HIGH Time	16		ns
t_{1L}	Timer1 Clock LOW Time	4080		ns

Parameter	Description	Min.	Max.	Unit
t_{2CLK}	Input Clock Rate (max. 62.5 MHz)	16		ns
t_{2H}	Timer2 Clock HIGH Time	16		ns
t_{2L}	Timer2 Clock LOW Time	4080		ns
	Timer1 and Timer Clock PWM Mode			
t_{1CLK}	Input Clock Rate (max. 62.5 MHz)	16		ns
t_{1H}	Timer1 Clock HIGH Time	16	4096	ns
t_{1L}	Timer1 Clock LOW Time	0	4080	ns
t_{2CLK}	Input Clock Rate (max. 62.5 MHz)	16		ns
t_{2H}	Timer2 Clock HIGH Time	16	4096	ns
t_{2L}	Timer2 Clock LOW Time	0	4080	ns

Notes:
2. Per Table 7-6 of revision 1.0 of USB specification, for Cload of 100–350 pF.
3. Power-on Reset will occur until the voltage on V_{CC} increases above Vrst.
4. Timing difference between the differential data signals.
5. Measured at crossover point of differential data signals.
6. The maximum load specification is the maximum effective capacitive load allowed that meets the target hub VBUS droop of 330 mV.
7. Accesses (read, write) to 1-Kbyte FIFO, including through register 0x73h (all HHI control modes).
8. Accesses (read, write) to 256-byte data RAM and I/O registers (all HHI control modes).
9. Reflects 1K FIFO characteristic.
10. With 100 pF on all SPI pins.
11. Data programmed in Timer1 or 2 divided by 2 is the SPI clock frequency.
12. Time to data active from high-impedance state.
13. Hold time to high-impedance state.

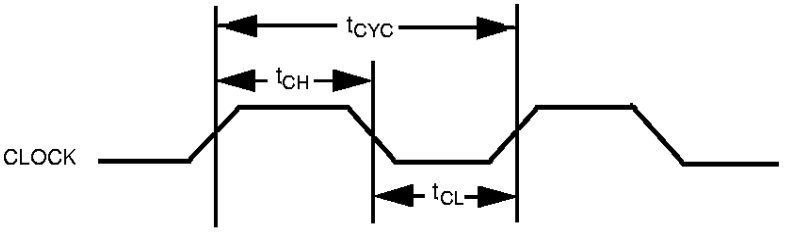


Figure 21-1. Clock Timing

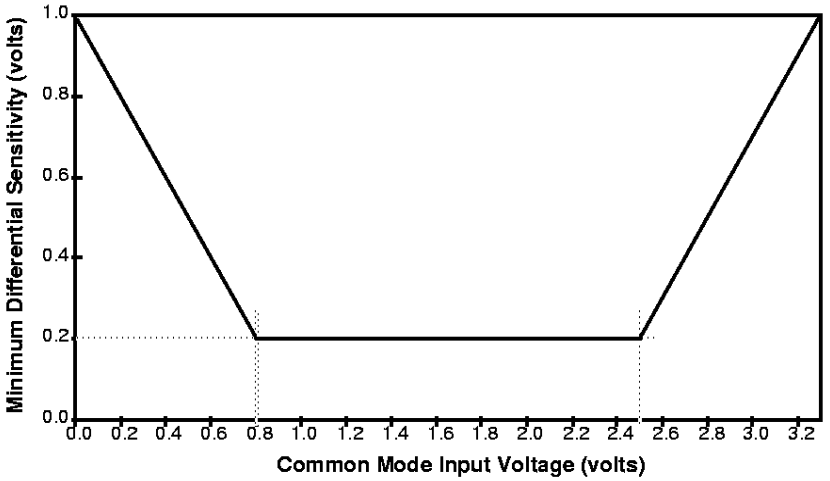


Figure 21-2. Differential Input Sensitivity Over Entire Common Mode Range (USB)

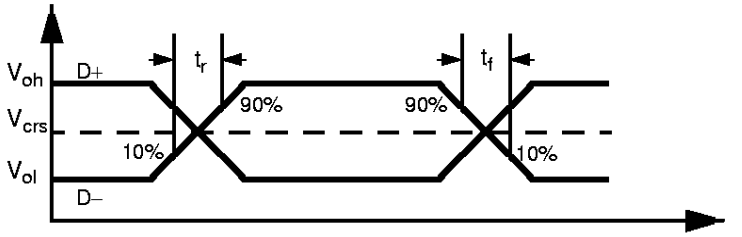


Figure 21-3. USB Data Signal Rise and Fall Time

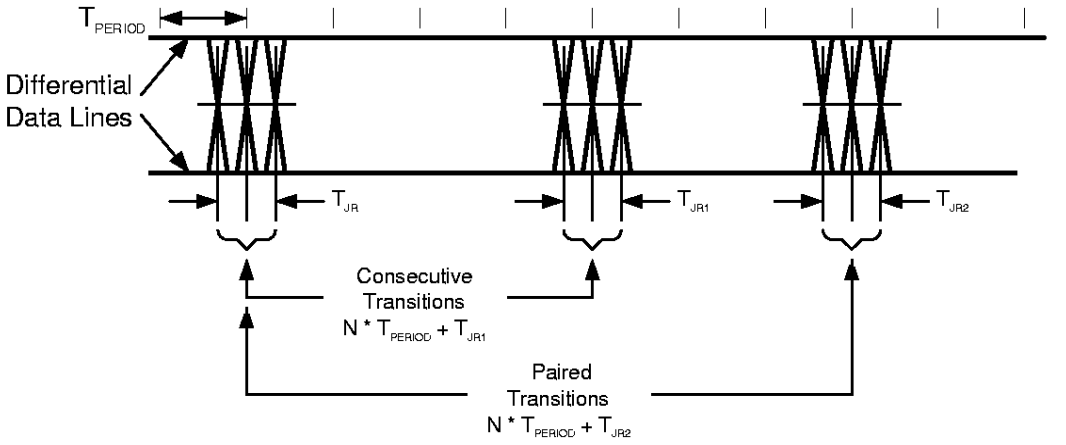


Figure 21-4. USB Receiver Jitter Tolerance

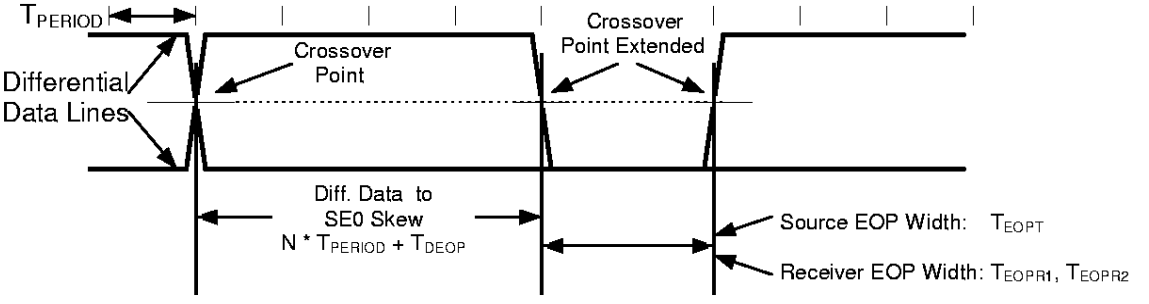


Figure 21-5. Differential to EOP Transition Skew and EOP Width (USB)

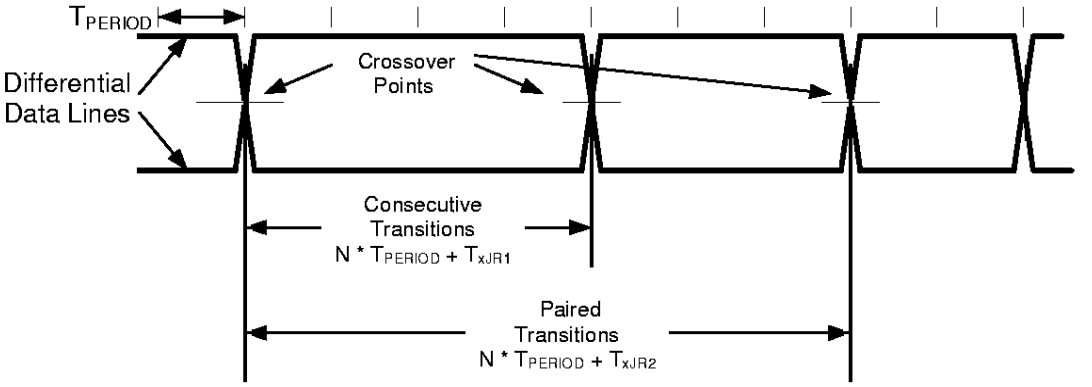


Figure 21-6. Differential Data Jitter

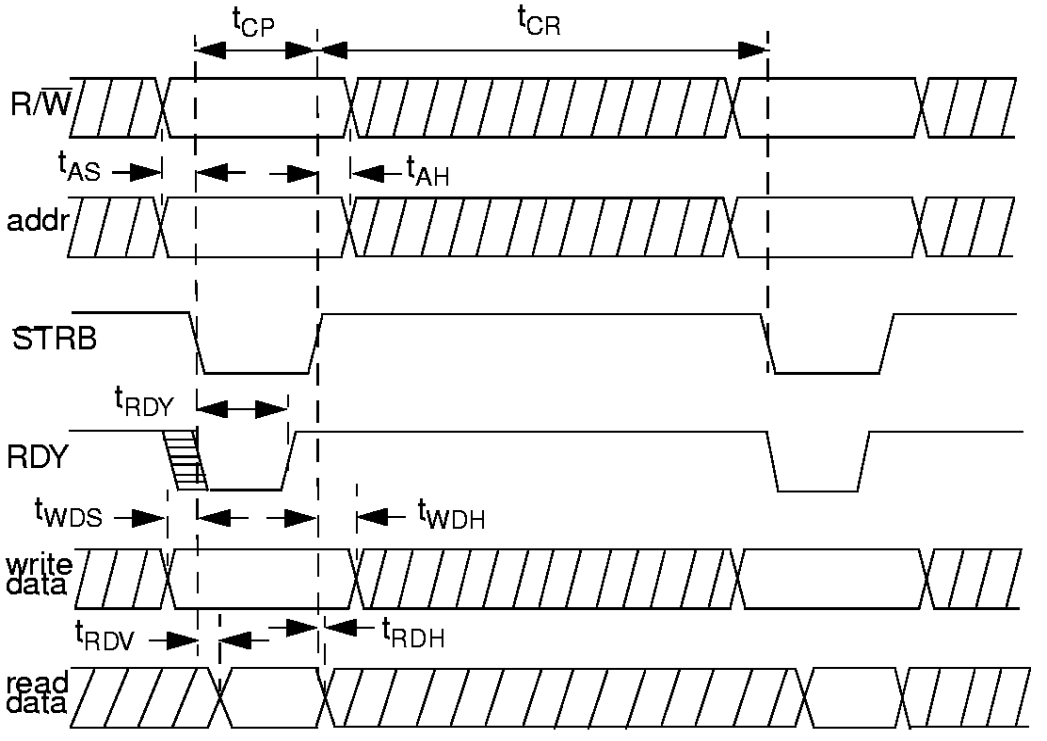


Figure 21-7. BHHI Timing Diagram (Pulling Mode)

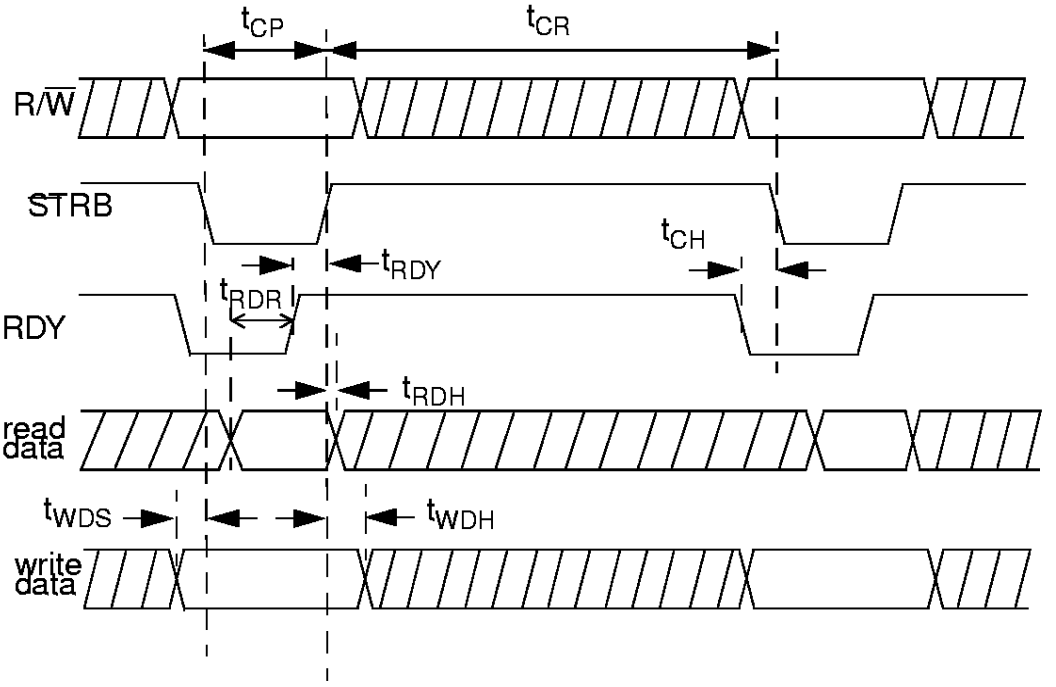


Figure 21-8. BHHI Timing Diagram (Pushing Mode, Forwarding Operation)

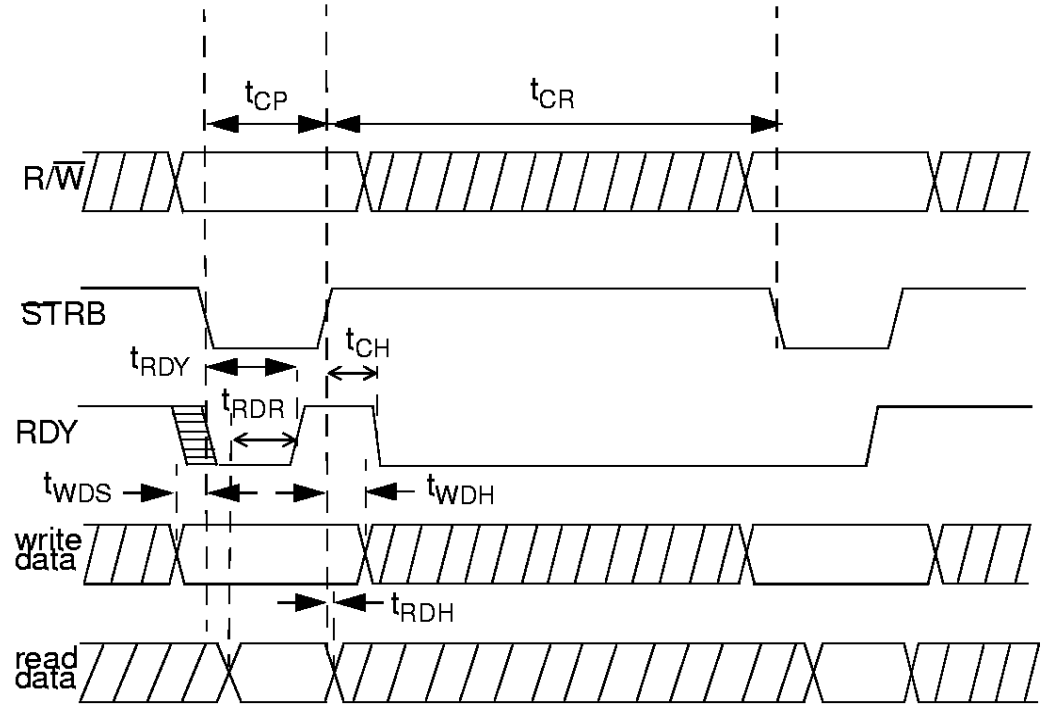


Figure 21-9. BHHI Timing Diagram (Pushing Mode, Reversing Operation)

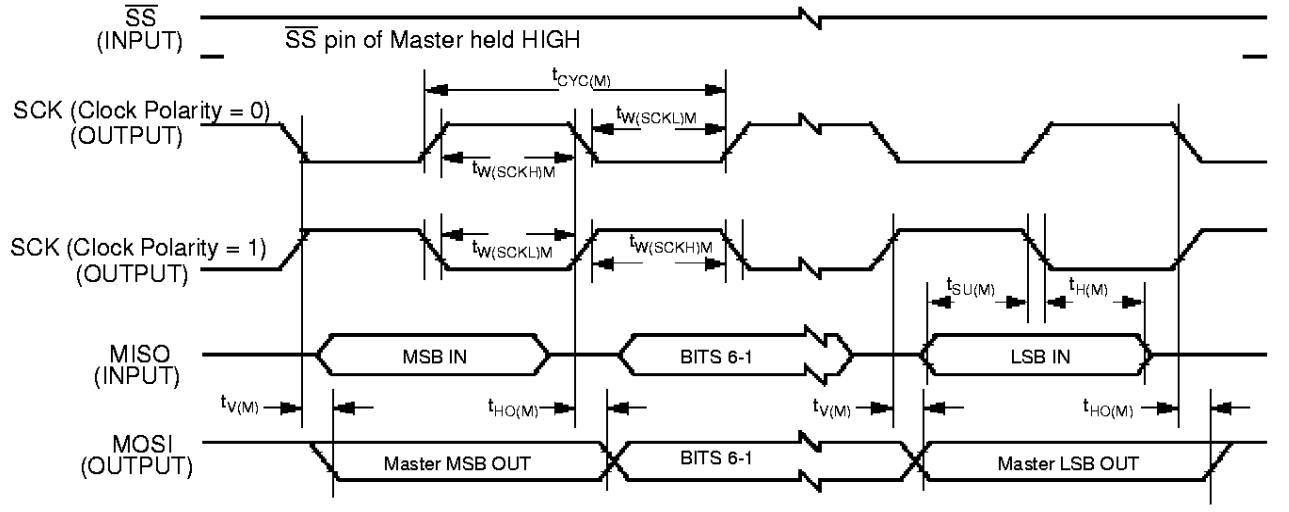


Figure 21-10. SPI Master Timing (Clock Phase = 0)

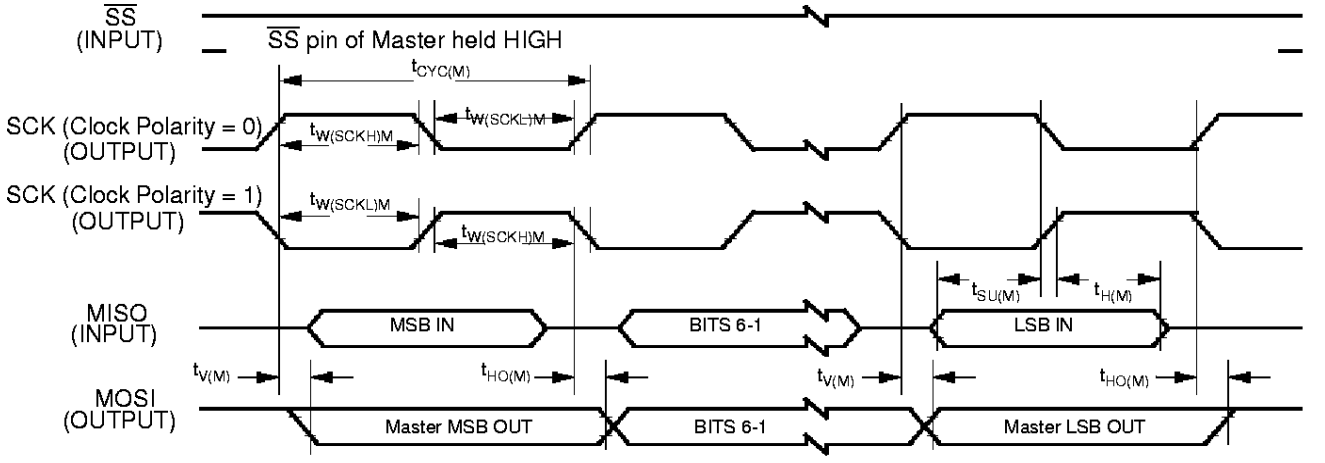
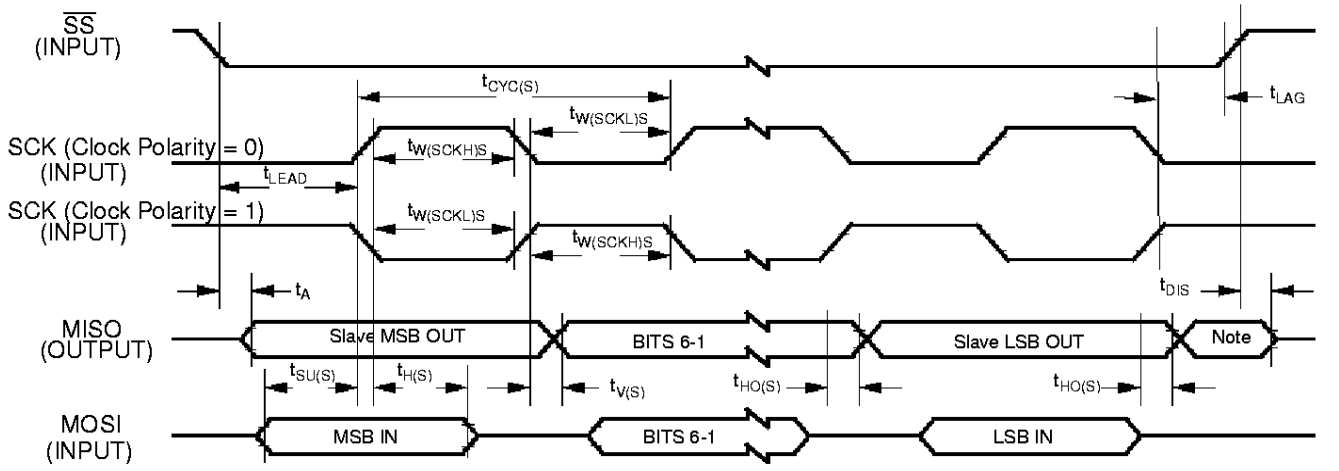
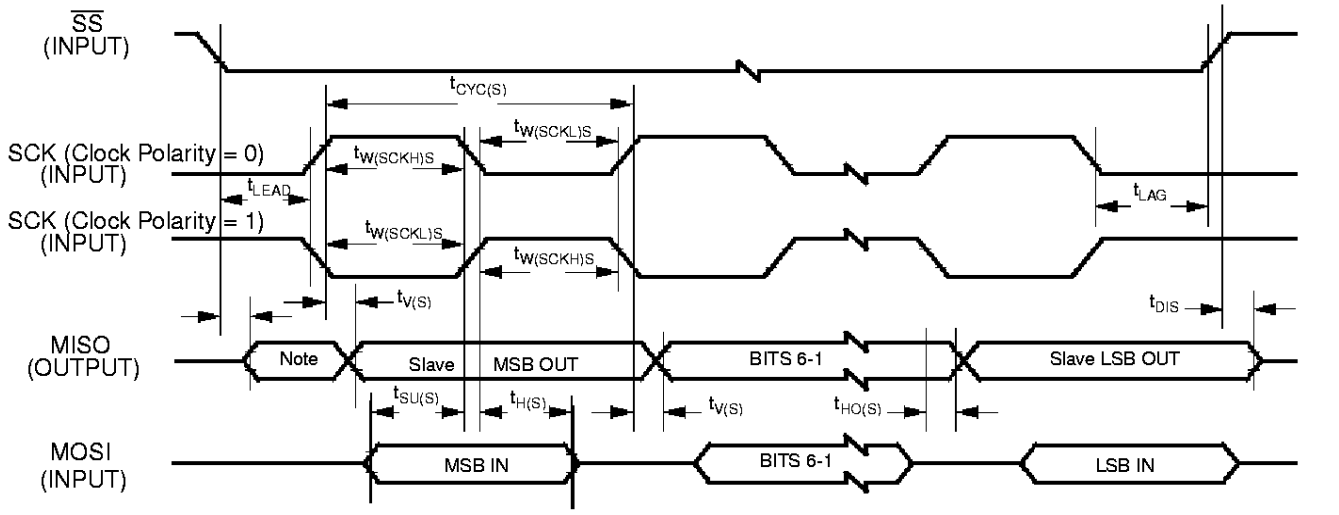


Figure 21-11. SPI Master Timing (Clock Phase = 1)



Note: Not defined but normally MSB of character just received

Figure 21-12. SPI Slave Timing (Clock Phase = 0)



Note: Not defined but normally LSB of character previously transmitted

Figure 21-13. SPI Slave Timing (Clock Phase = 1)

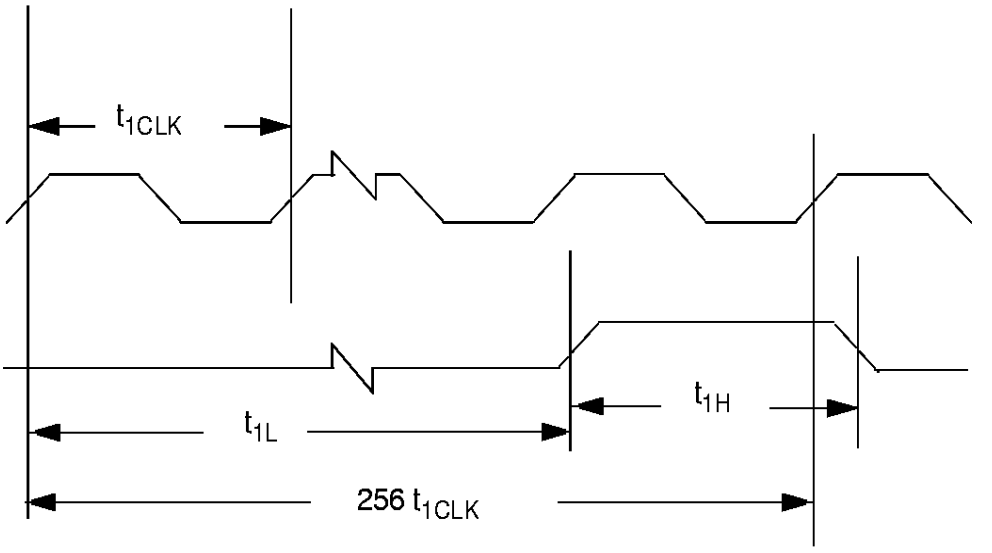


Figure 21-14. Timer1—Clock Terminal Count Mode

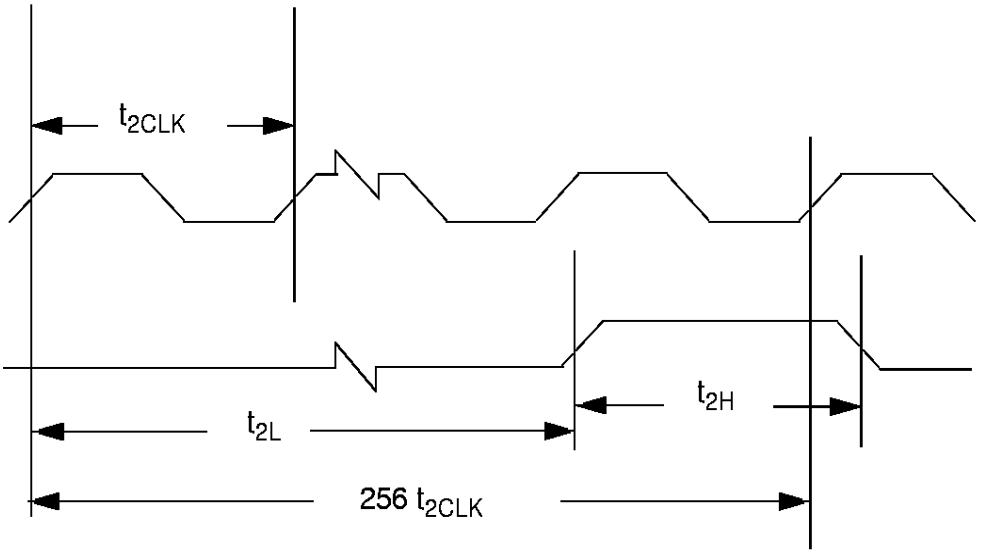


Figure 21-15. Timer2—Clock Terminal Count Mode

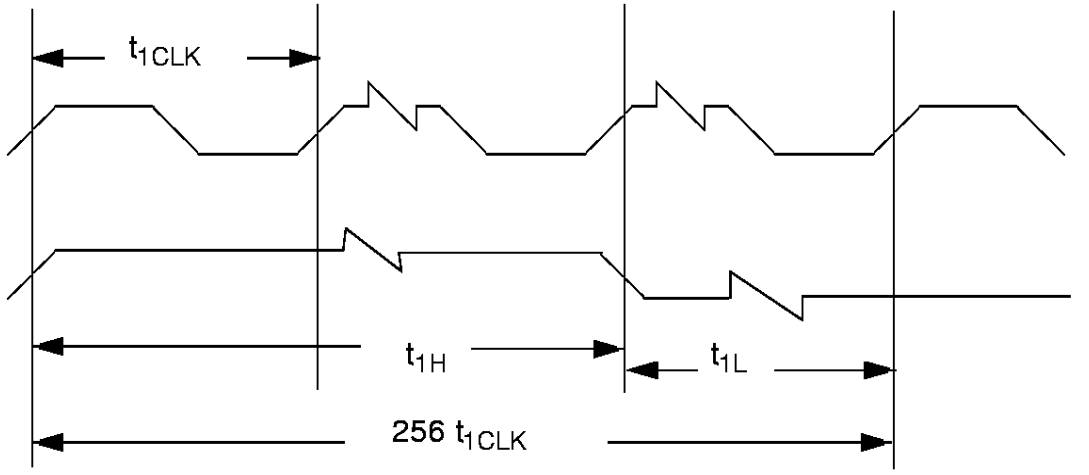


Figure 21-16. Timer1—Clock PWM Mode

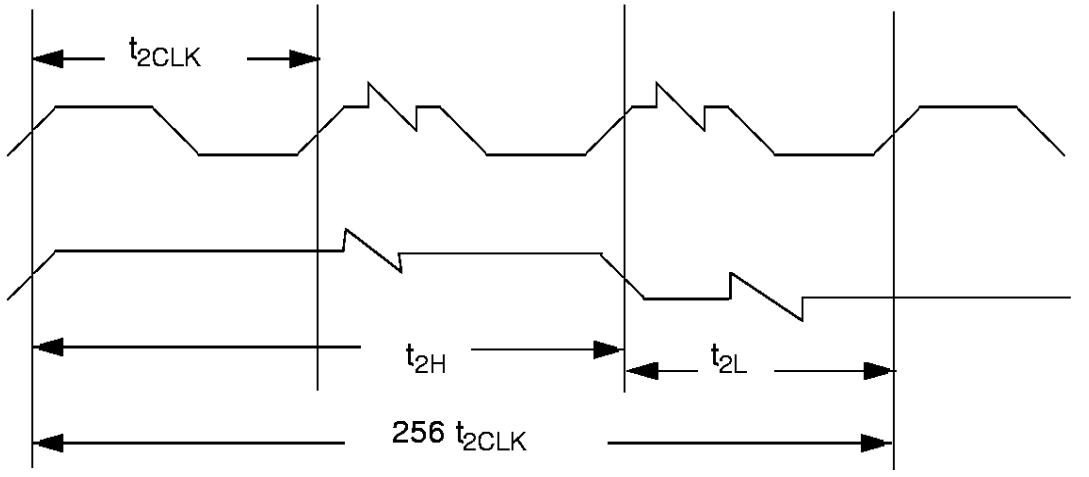


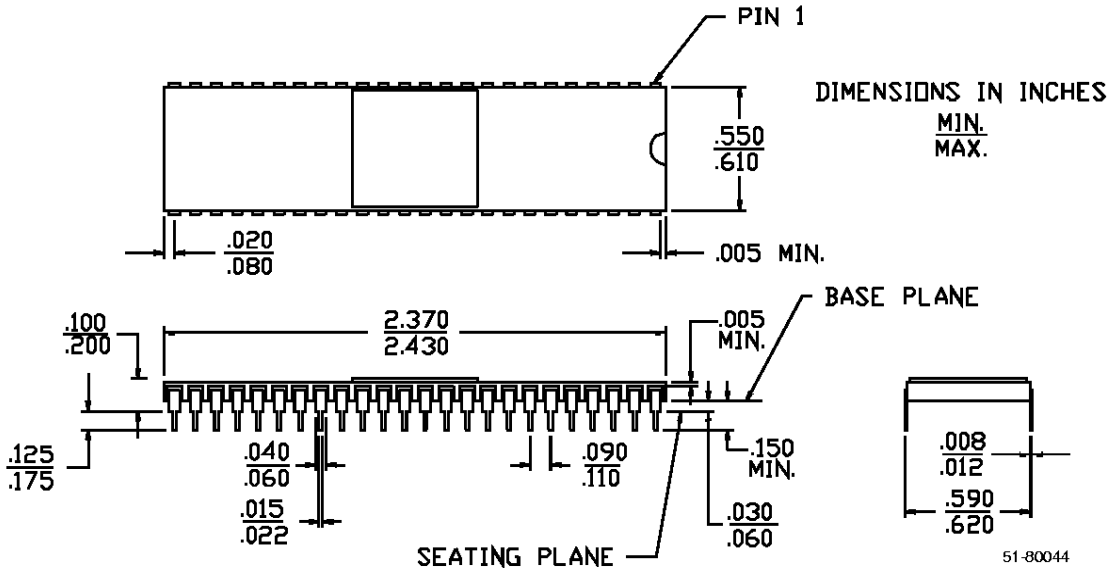
Figure 21-17. Timer2—Clock PWM Mode

22.0 Ordering Information

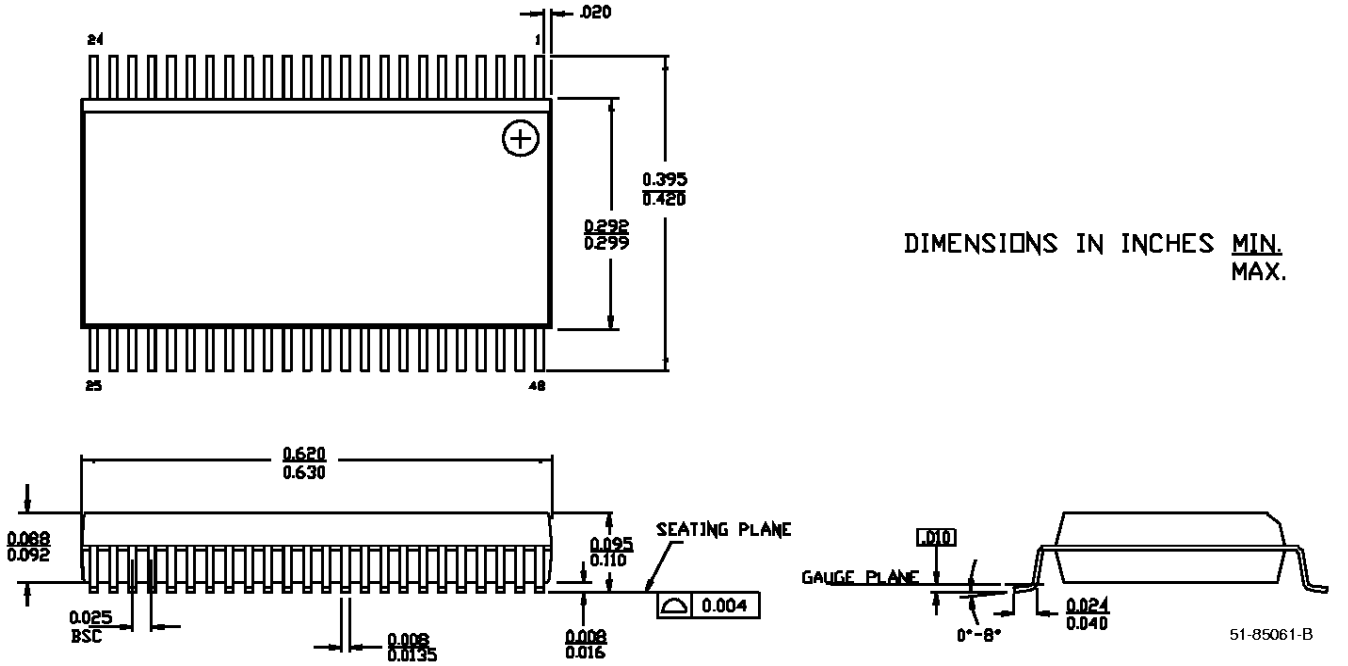
Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C64211-SC	4 KB	S21	28-Pin (300-Mil) SOIC	Commercial
CY7C64212-SC	6 KB	S21	28-Pin (300-Mil) SOIC	Commercial
CY7C64213-SC	8 KB	S21	28-Pin (300-Mil) SOIC	Commercial
CY7C64213-WC	8 KB	W22	28-Pin Windowed CerDIP	Commercial
CY7C64311-PVC	4 KB	O48	48-Pin (300-Mil) SSOP	Commercial
CY7C64312-PVC	6 KB	O48	48-Pin (300-Mil) SSOP	Commercial
CY7C64313-PVC	8 KB	O48	48-Pin (300-Mil) SSOP	Commercial
CY7C64313-WVC	8 KB	D26	48-Pin Windowed SideBraze	Commercial

23.0 Package Diagrams

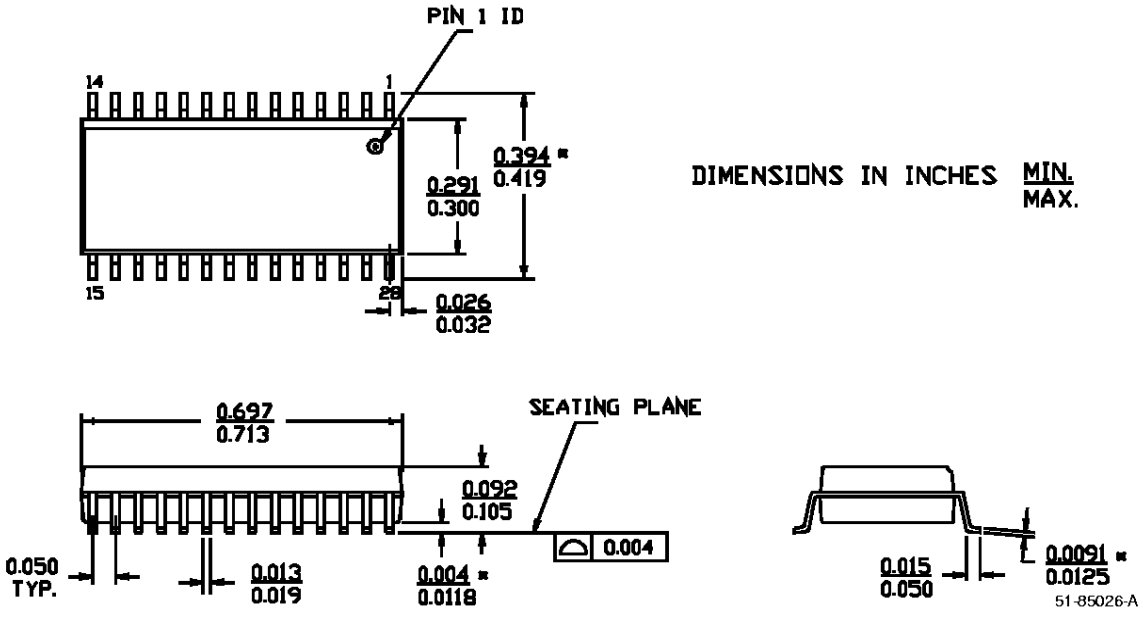
48-Lead (600-Mil) Sidebraze DIP D26
MIL-STD-1835 D-14 Config. C



48-Lead Shrunk Small Outline Package O48

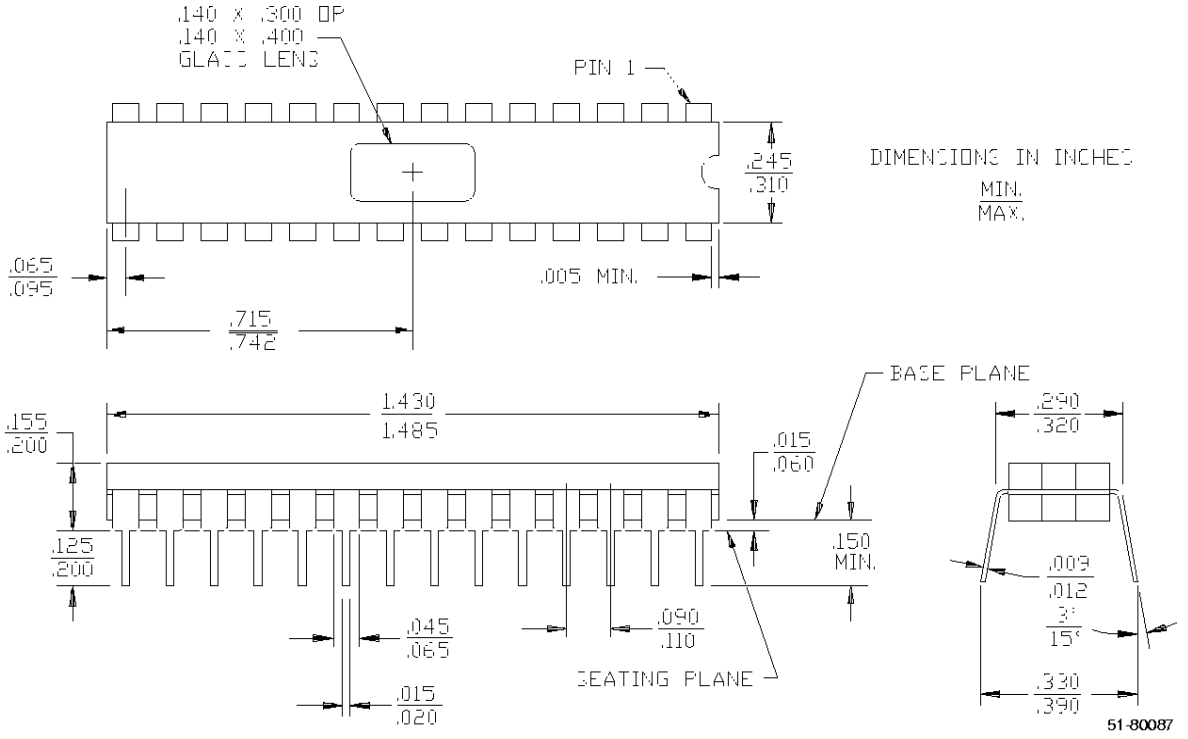


28-Lead (300-Mil) Molded SOIC S21



28-Lead (300-Mil) Windowed CerDIP W22

MIL-STD-1835 D-15 Config. A



© Cypress Semiconductor Corporation, 1998. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress Semiconductor product. Nor does it convey or imply any license under patent or other rights. Cypress Semiconductor does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress Semiconductor products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress Semiconductor against all charges.