CY7C64213
CY7C64313
Full-Speed USB (12 Mbps) Microcontroller

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 [IEEE 1284], modems, hand-held scanners, serial devices, USB bridge, etc.)
- USB Specification Compliance
  - —Conforms to USB Specification, Version 1.1
  - —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 1023 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
  - —8 KB of EPROM (CY7C64213, CY7C64313)
  - —1 KB of Endpoint FIFO RAM
- 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 Transfers among external BUS interfaces (SPI and BHHI), 1K FIFO and USB SIE interface through one DMA channel, without intervention of the microcontroller
- Bidirectional Hardware Handshaking Interface (BHHI) - shared with GPIOs
  - —Flexible microprocessor 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; can also 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 Mbit/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-μs 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
- CY7C64213 available in 28-pin SOIC package
- CY7C64313 available in 48-pin SSOP package
- Industry standard programmer support

2.0 Functional Overview

The CY7C64213 and CY7C64313 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 CY7C64213 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]). Each port can be configured as inputs with internal pull-ups (14-KΩ), 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 and 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 CY7C64313.

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 CY7C64313 have separate address and data lines, whereas the CY7C64213 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 CY7C64313 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, $\overline{\text{CTS}}$, $\overline{\text{RTS}}$. It supports a baud rate from 2400 to 1M bits/s, parity (odd, even, or none), and error detection (framing, parity, and over-run). Quadrature Baud Rate Clock for the UART interface can be generated from either Timer1 or Timer2.

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

There are seven Special-Purpose I/O (SPIO) in the CY7C64313 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 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 clock signals for distribution within the microcontroller.

The CY7C64213 and CY7C64313 have 8 KB of EPROM. All parts have 256 bytes of internal RAM (includes Endpoint 0 DMA buffer) and 1 KB of additional Endpoint FIFO memory. The 1K FIFO space support up to 8 endpoints (Endpoints 1 to 8). The size of each endpoint is configurable from 128 bytes to 1023 bytes.

The CY7C64213/313 include power-on reset logic, an optional external reset pin (SPIO[4]), and four timers: a 16-bit 1-MHz 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 CY7C64213 and CY7C64313, allowing external asynchronous reset of the microcontrollers.

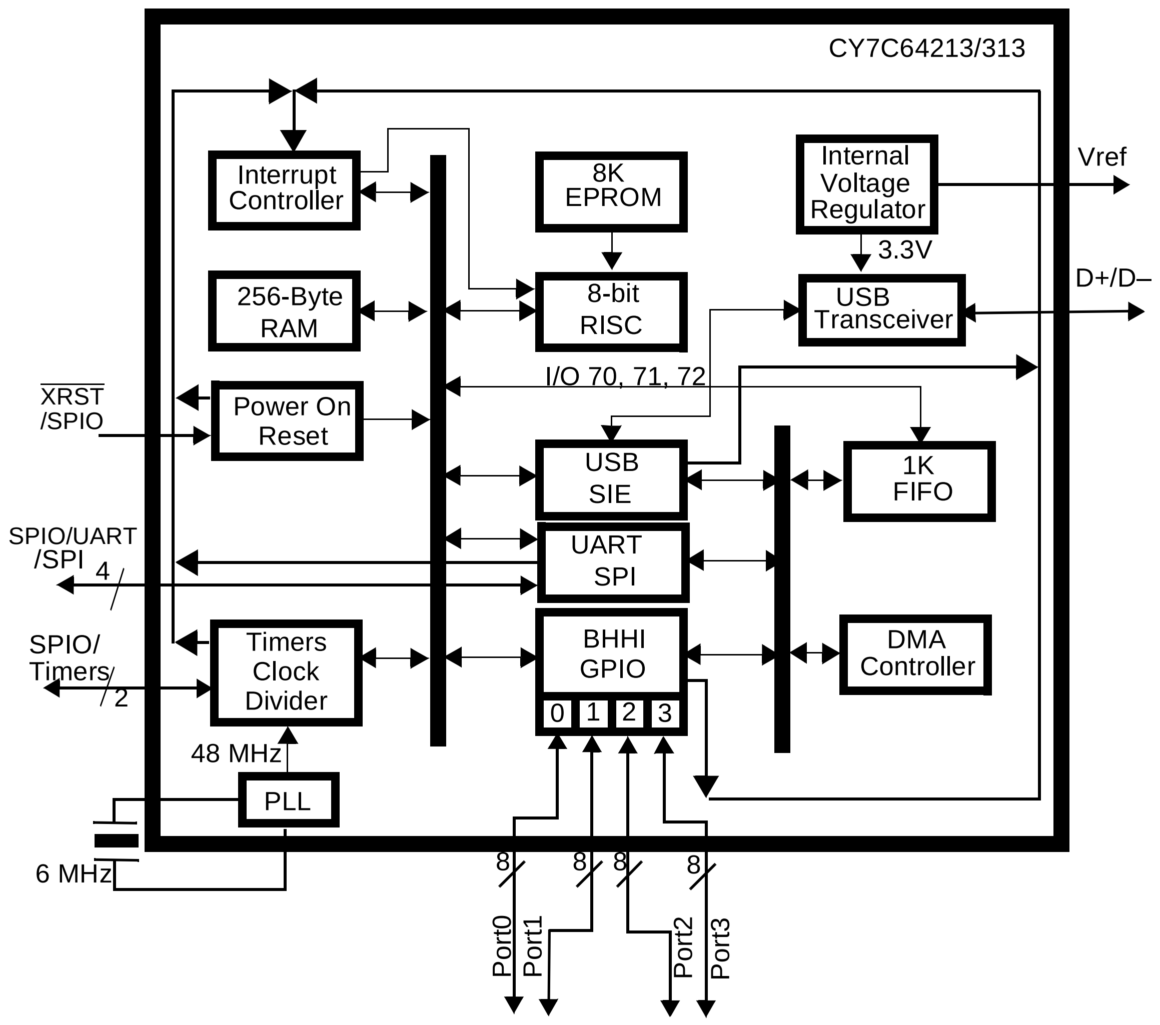
Timer0 is a 16-bit, free running timer, clocked at 1 MHz. It provides the interrupt source for the 1.024-ms interrupt. The upper 8 bits of the timer are latched into an internal register when the firmware reads the lower 8 bits. A read from the upper 8 bits actually reads data from the internal register, 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.

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 The watchdog timer is used to ensure the microcontroller can recover if it is stalled for more than approximately 8 ms. 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-ms interrupt service routine). If the watchdog timer is not cleared for approximately 8 ms, then a watchdog reset is generated that will return the microcontroller to a known state.

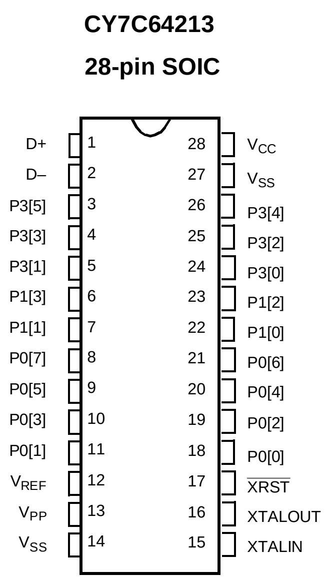
The microcontroller supports 10 maskable interrupts in the vectored interrupt controller. Interrupt sources include the 1.024-ms timer, 2 USB endpoint interrupts (Endpoint 0 and Endpoint i, i=1..8), the GPIO/SPIO ports, BHHI, UART, Timer1 and Timer2, USB Bus Reset, USB Start of Frame (SOF).

The GPIO 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 CY7C64213 and CY7C64313 include an integrated USB serial interface engine (SIE). The hardware supports one USB device addresses with up to nine endpoints. The SIE provides an interface to communicate with the functions integrated into the microcontroller.


CY7C64213/313
Interrupt Controller
8K EPROM
Internal Voltage Regulator
Vref
3.3V
256-Byte RAM
8-bit RISC
USB Transceiver
D+/D–
I/O 70, 71, 72
XRST /SPIO
Power On Reset
USB SIE
1K FIFO
SPIO/UART /SPI
4
UART SPI
SPIO/ Timers
2
Timers Clock Divider
BHHI GPIO
0
1
2
3
DMA Controller
48 MHz
PLL
6 MHz
8
8
8
8
Port0
Port1
Port2
Port3

TOP VIEW



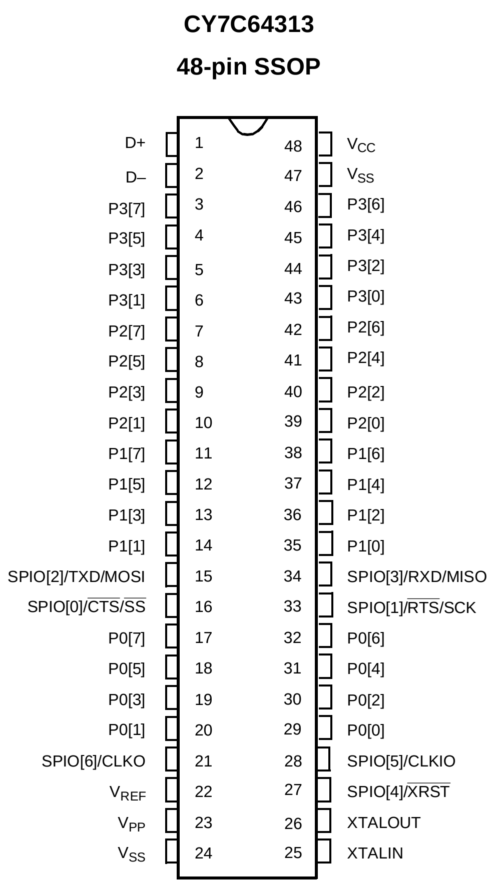


Figure 2-1. Pin Configurations

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). BHHI 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). BHHI control signals
P2	I/O		39,10,40,9, 41,8,42,7	GPIO Port 2 capable of sinking 12 mA (typical). BHHI 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). BHHI 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 to Timer1; Timer1 Output; Hardware Trigger Enable for Timer1; 12-MHz Clock Output
SPIO[6]/ CLKO/ T2O/HTI2	I/O		21	Special Purpose I/O; Timer2 Output; Hardware Trigger Enable for Timer2; 12-MHz Clock Output
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
V_{SS}	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 V_{SS} 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
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
B2B Starting Address	0x17	R/W	Back-to-Back Starting Address Register
EP1–8 USB ACK Status	0x18	R	USB Endpoint 1–8 ACK Status Register
EP1–8 Buffer Control	0x19	R/W	USB Endpoint FIFO Control Register
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	0x23	R/W	Allows Read and Write of Interrupt Vector
Timer0 (LSB)	0x24	R	Free-running Timer Lower 8 bits (1 MHz)
Timer0 (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
Timer1	0x29	R/W	Timer for generating clock or interrupt or PWM
Timer2	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

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

Register Name	I/O Address	Read/ Write	Function
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
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 FIFO	0x70	R/W	LSB 8-bit Address for accessing 1K FIFO
Address 1 for 1K FIFO	0x71	R/W	MSB 2-bit Address for accessing 1K FIFO
Data for 1K FIFO	0x72	R/W	1K FIFO Data Register
Data for 1K FIFO Through DMA	0x73	R/W	1K FIFO Data Register with Auto-Incrementing of 70h
DMA Stop Pointer	0x74	R/W	DMA Stop Pointer of DMA controller
POR Configuration	0xF1	R	POR Configuration
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
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) addresses up to 8 KB of EPROM available with the CY7C64213/313 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

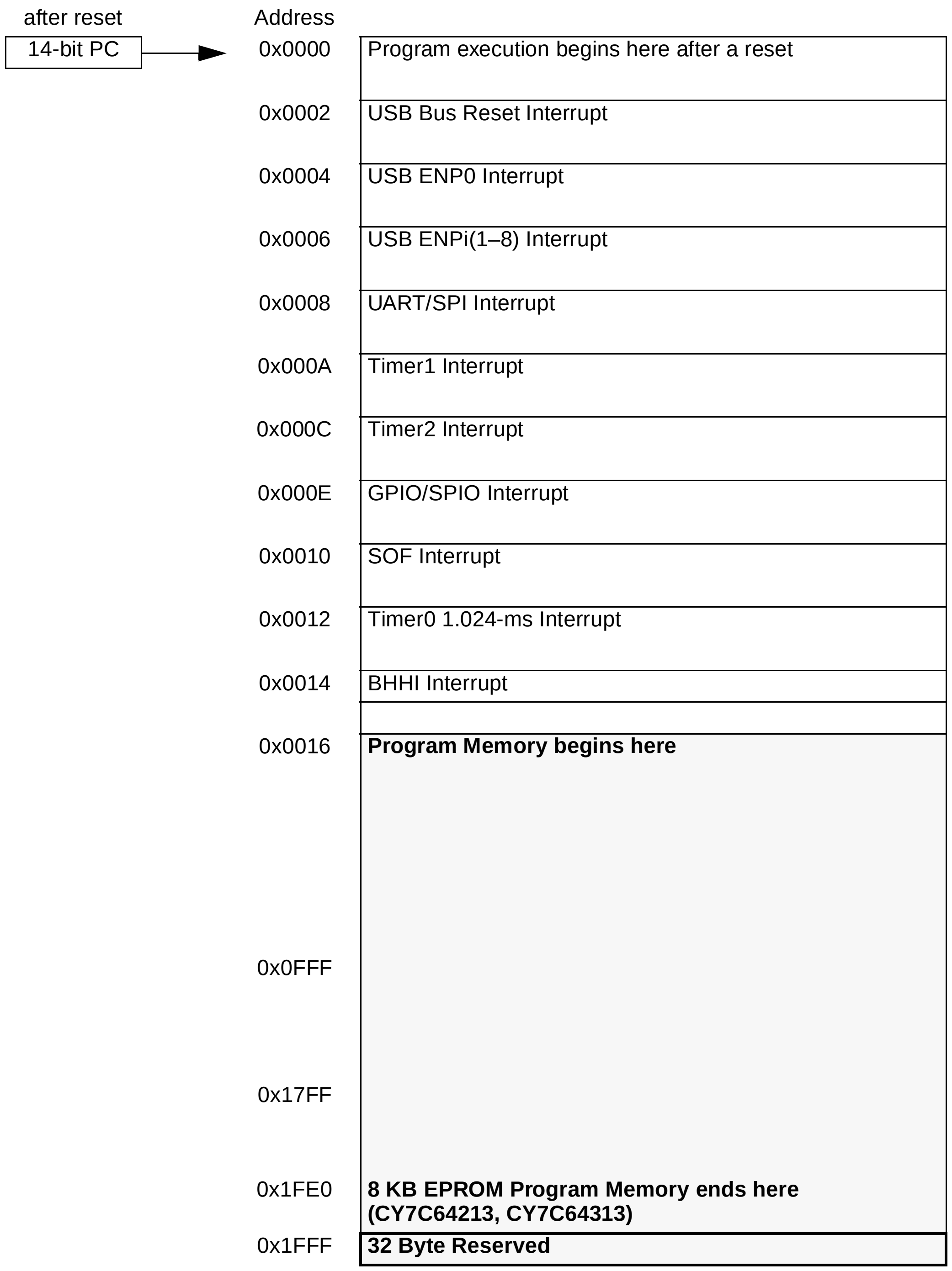


Figure 4-1. Program Memory Space with Interrupt Vector Table

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 using a value loaded into the X register.

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 pushed onto the stack 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.

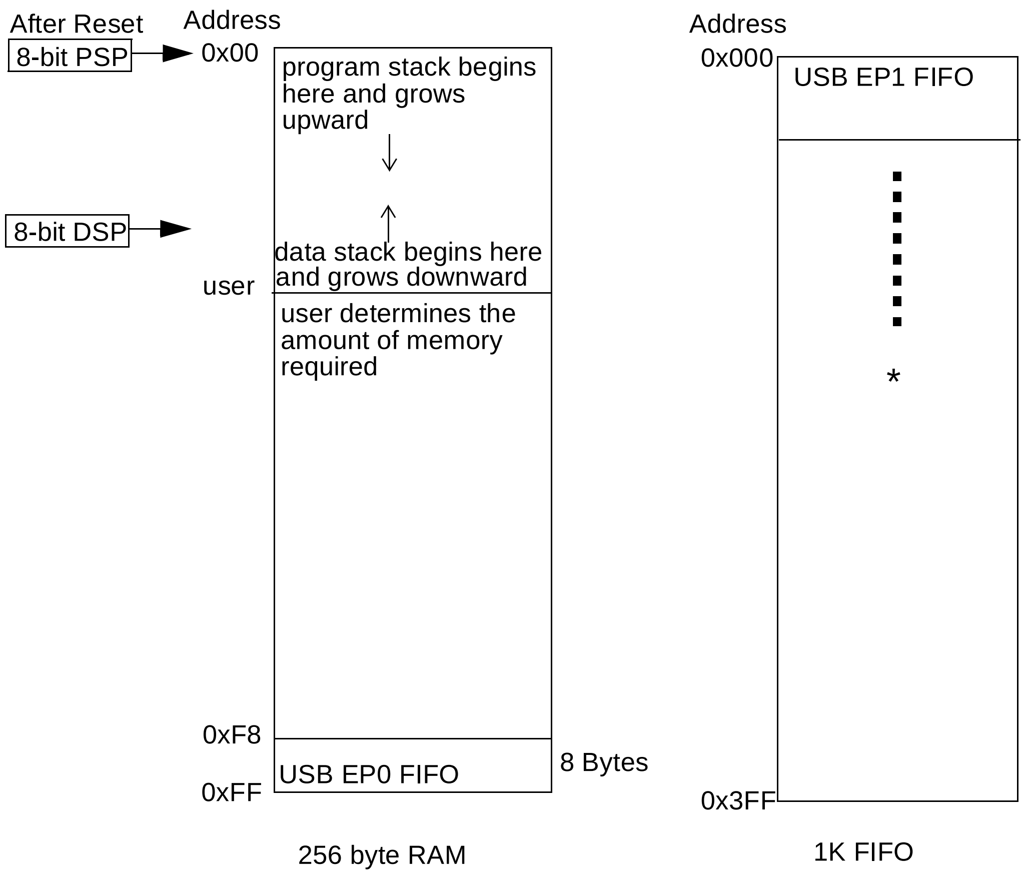
The return from interrupt (RETI) instruction decrements the program stack pointer, then restores the second byte from memory addressed by the PSP. The program stack pointer is decremented again and the first byte is restored from memory addressed by the PSP. After the program counter and flags have been restored from the 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 CY7C64213 and CY7C64313 microcontrollers provide 256 bytes of data RAM and 1 Kbyte 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 1K FIFO RAM. The CY7C64213/313 supports up to 8 endpoints. FIFO sizes are selected through the USB Endpoint FIFO Control Register (0x19h), and can be set between 128 and 1023 bytes. The number of endpoints will be restricted if large FIFOs are selected.


After Reset
Address
8-bit PSP
0x00
program stack begins here and grows upward
8-bit DSP
data stack begins here and grows downward
user
user determines the amount of memory required
0xF8
USB EP0 FIFO
8 Bytes
0xFF
256 byte RAM
Address
0x000
USB EP1 FIFO
*
0x3FF
1K FIFO

* Total of 8 endpoints can be supported, refer to USB Endpoint FIFO Control Register (0x19h) 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 will 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 program stack (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 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 CY7C64213 and CY7C64313 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 requires two bytes of code where the first byte identifies the “MOV A” instruction with a data operand as the second byte. The second byte of the instruction 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 has the effect of loading A with the fourth element of the SRAM “array” that begins at address 0x10h. The fourth element is at address 0x13h.

5.0 Clocking

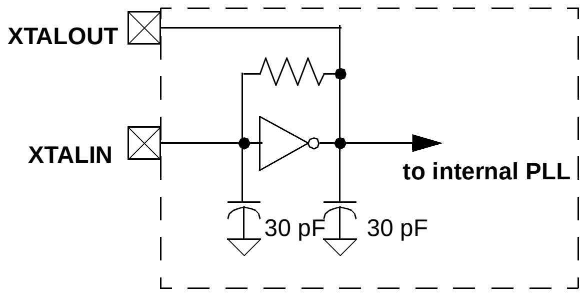


Figure 5-1. Clock Oscillator On-chip Circuit

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 USB D+ / D–

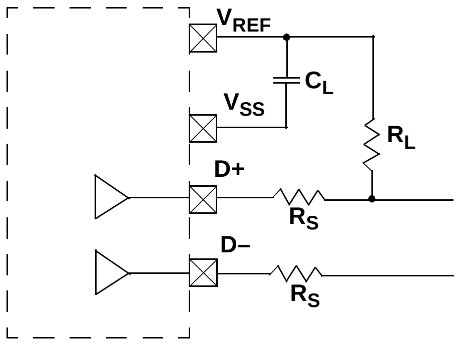


Figure 6-1. D+ and D– Termination Resistors

External resistors R_S of 22 Ohm $\pm$ 5% must be connected in series to D+ and D– lines to meet the USB 1.1 specification.

An external pull-up resistor R_L of 1.5 Kohm $\pm$ 5% is required.

A bypass capacitor C_L of 100 nF between V_{REF} and V_{SS} is also required.

7.0 Reset

The CY7C64213/313 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 WDR, USB Bus Reset[1], and POR 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.

Note:
1. USB Bus Reset is not a hardware reset in this device. See section 18.3, USB Bus Reset Interrupt.

7.1 Power-On Reset (POR)

The CY7C64213/313 enters a suspend state when V_{CC} is first applied to the chip. The Power-On Reset (POR) signal is asserted during this suspend time and ensures that both a valid V_{CC} level is reached and that the internal PLL has time to stabilize. When the V_{CC} has risen above approximately 2.5V, and the oscillator is stable, the POR is deasserted and the on-chip timer will start counting. The first 1 ms of suspend time is not interruptible, and the suspend state will continue for an additional 95 ms unless the count is bypassed by a USB Bus Reset on the upstream port. The 96 ms provides time for V_{CC} to stabilize at a valid operating voltage before the chip executes code.

If a USB Bus Reset occurs during the 96 ms suspend time, the suspend will be aborted and program execution will begin immediately from address 0x0000. In this case, the Bus Reset interrupt will be pending but will not be serviced until firmware sets the USB Bus Reset Interrupt Enable bit (bit 0 of register 0x20) and enables interrupts with the EI command.

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

7.2 External Reset

An optional external reset pin is provided (otherwise used as SPIO[4]). The external reset is active LOW and must 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 enabled.

7.3 Watch Dog Reset (WDR)

The Watch Dog Timer Reset (WDR) occurs when the internal Watch Dog timer rolls over. Writing any value to the write-only Watch Dog Restart Register at address 0x26 will clear the timer. The timer will roll over and WDR will occur if it is not cleared within t_{WATCH} (8 ms minimum) of the last clear. Bit 6 of the Processor Status and Control Register is set to record this event (the register contents are set to 010X0001 by the WDR). A Watch Dog Timer Reset lasts for 2 to 4 ms after which the microcontroller begins execution at ROM address 0x0000.

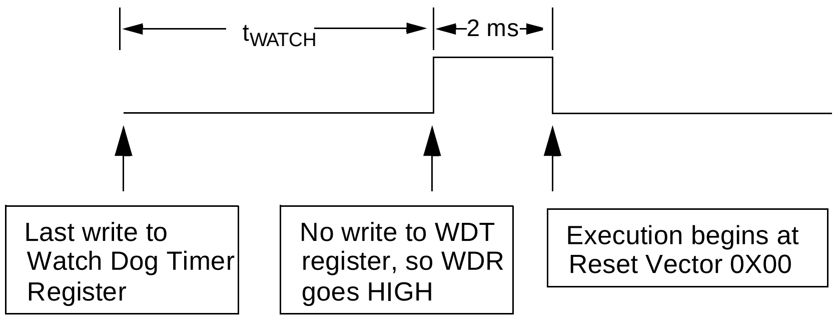

tWATCH
2 ms
Last write to Watch Dog Timer Register
No write to WDT register, so WDR goes HIGH
Execution begins at Reset Vector 0X00

Figure 7-1. Watch Dog Reset (WDR)

The USB transmitter is disabled by a Watch Dog Reset because the USB Device Address Registers are cleared. Otherwise, the USB Controller would respond to all address 0 transactions.

8.0 Suspend Mode

The CY7C64213/313 can be placed into a low-power state by setting the Suspend bit (bit 3) of the Processor Status and Control Register. All logic blocks in the device are turned off except the GPIO interrupt logic and the USB transceiver. The clock oscillator and PLL, as well as the free-running and watch dog timers are shut down. Only the occurrence of an enabled GPIO interrupt or non-idle bus activity at a USB port will wake the part from suspend. The Run bit must be set for a part to resume out of suspend.

The clock oscillator restarts immediately after exiting suspend mode. The microcontroller returns to a fully functional state 1 ms after the oscillator is stable. The microcontroller will execute the instruction following the I/O write that placed the device into suspend mode before servicing any interrupt requests.

The GPIO interrupt allows the controller to wake-up periodically and poll system components while maintaining a very low average power consumption. To achieve the lowest possible current during suspend mode, all I/O should be held at either V_{SS} or V_{SS}.

Special care should be exercised with any unused GPIO data bits. An unused GPIO data bit, whether it represents 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 mode, 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 CY7C64213 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.

Typical code for entering suspend is shown below:

```
...             ; Enable GPIO interrupts if desired for wake-up and disable Timer Interrupts (Timer1, Timer2 and 1.024-ms).
mov a, 09h      ; Set suspend and run bits
iowr FFh        ; Write to Status and Control Register - Enter suspend
nop             ; This executes before any ISR
..              ; Remaining code for exiting suspend routine
```

9.0 General Purpose I/O Ports

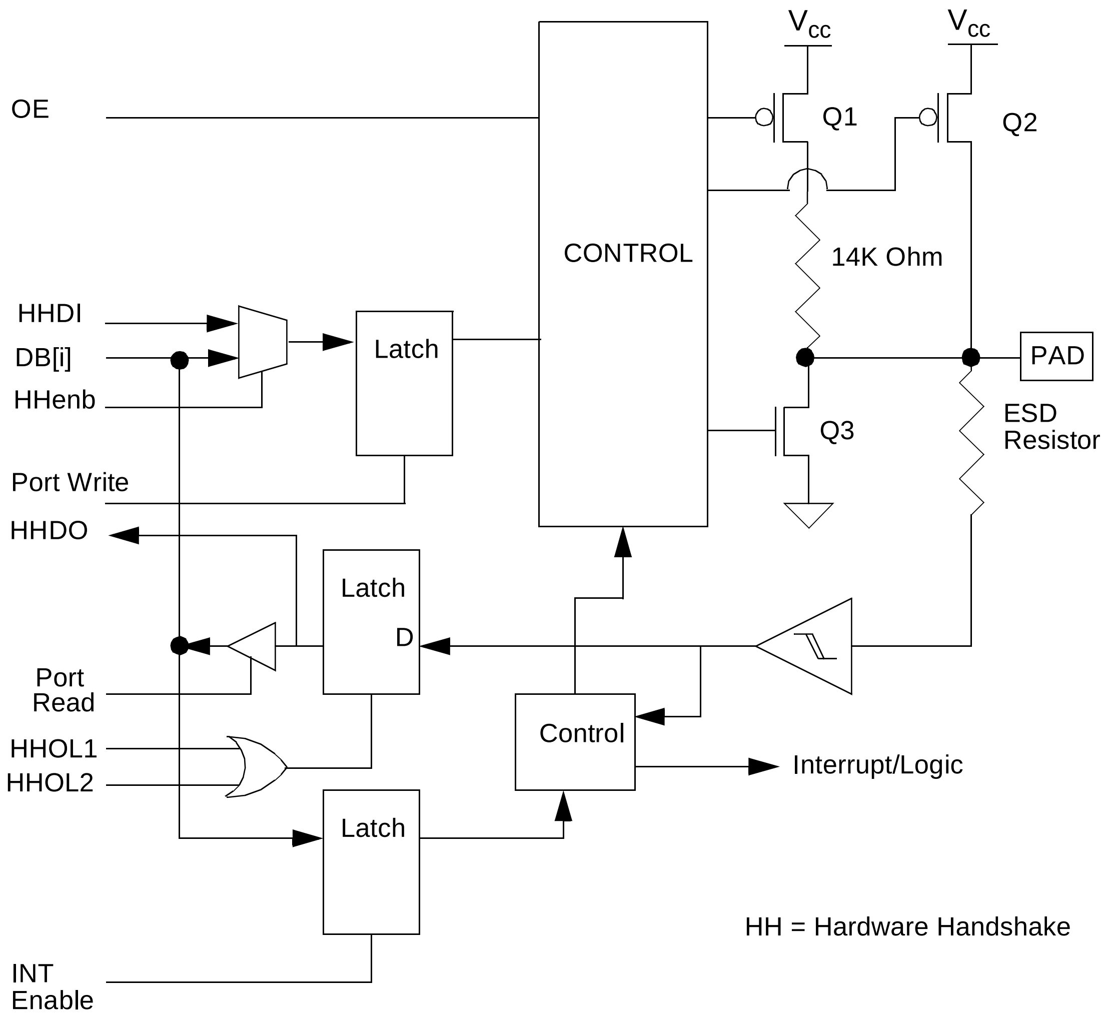


Figure 9-1. Block Diagram of a General Purpose I/O Line

There are 32 GPIO pins (P0, P1, P2, and P3) available in the CY7C64313, and 18 GPIO pins (P0, P1, and P3) available in the CY7C64213. 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 Bidirectional Hardware Handshaking Interface (BHHI). The data for each GPIO port is accessible through the data registers.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]

Figure 9-2. Port 0 Data 0x00h

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]

Figure 9-3. Port 1 Data 0x01h

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]

Figure 9-4. Port 2 Data 0x02h

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
P3[7]	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]

Figure 9-5. Port 3 Data 0x03h

During reset, all of the GPIO pins are configured in open drain mode, with data bit set to 1 (high impedance). 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.

9.1 GPIO Configuration Port

Every GPIO port can be programmed as inputs with internal pull-ups (resistive), open drain, or as a CMOS output. In addition, the interrupt polarity for each port can be programmed. With positive interrupt polarity, a rising edge on an input pin causes an interrupt. With negative polarity, a falling edge on an input pin causes an interrupt. As shown in the table below, when a GPIO port is configured as CMOS, interrupts from that port are disabled. The GPIO Configuration register provides two bits per port to program these features. The possible port configurations are listed in *Table 9-1*.

Table 9-1. Port Configurations

Port Configuration Bits	Interrupt Enable Bit	Driver Mode	Interrupt Polarity
11	X	Resistive	–
10	0	CMOS Output	disabled
10	1	Open Drain	disabled
01	X	Open Drain	–
00	X	Open Drain	+

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, positive interrupt polarity for all GPIO ports as the default configuration.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
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 9-6. GPIO Configuration Register 0x08h

9.2 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. In the “10” mode (Configuration bits) these registers are used to select CMOS Output or Open Drain modality (refer to *Table 9-1)*.

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

Figure 9-7. Port 0 Interrupt Enable 0x04h

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

Figure 9-8. Port 1 Interrupt Enable 0x05h

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

Figure 9-9. Port 2 Interrupt Enable 0x06h

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

Figure 9-10. Port 3 Interrupt Enable 0x07h

10.0 Timers

10.1 16-bit Free-running Timer (Timer0)

The 16-bit timer (Timer0) provides one interrupt (1.024 ms) and allows the firmware to directly time events that are up to 65 ms 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.

10.1.1 Timer0 (LSB)

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
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 10-1. Timer0 Register 0x24h

10.1.2 Timer0 (MSB)

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
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 10-2. Timer0 Register 0x25h

After reset the values of bits[7:0] in the Timer0 registers (0x24h and 0x25h) are “unknown”.

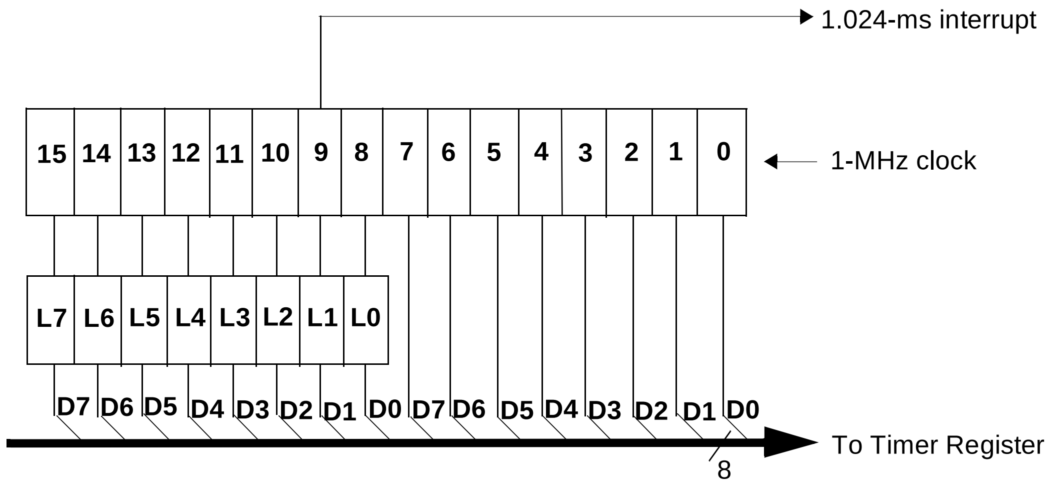


Figure 10-3. Timer0 Block Diagram

10.2 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 Timer Control 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 counting down to zero, and automatically reload the count value (in Timer1 and Timer2 Control registers 0x29h, 0x2Ah).

Pulse width modulation is programmed by writing into Timer1 Control Register (0x29h) or Timer2 Control Register (0x2Ah) the number of clocks the output will be at logic 1 (register value + 1); for the rest of clocks, while a timer is counting down to the terminal count, the output 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 Prescaler Output Frequency Control Register (0x28h) and can be used as an input to Timer1 and Timer2 for further scale down.

The Prescaler is a down counter that starts when the Prescaler Output Frequency Control Register (0x28h) is loaded. When the counter is down to 0, it generates 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 10-4. 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 10-5. 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 10-6. 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 10-7. 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.

After reset the values of bits[7:0] in the registers 0x28h, 0x29h, 0x2Ah, 0x2Ch are at logic “0”.

When Timer1 and Timer2 are in non_trigger_enable mode and operating as terminal counters, their outputs are as shown in *Figure 10-8*.

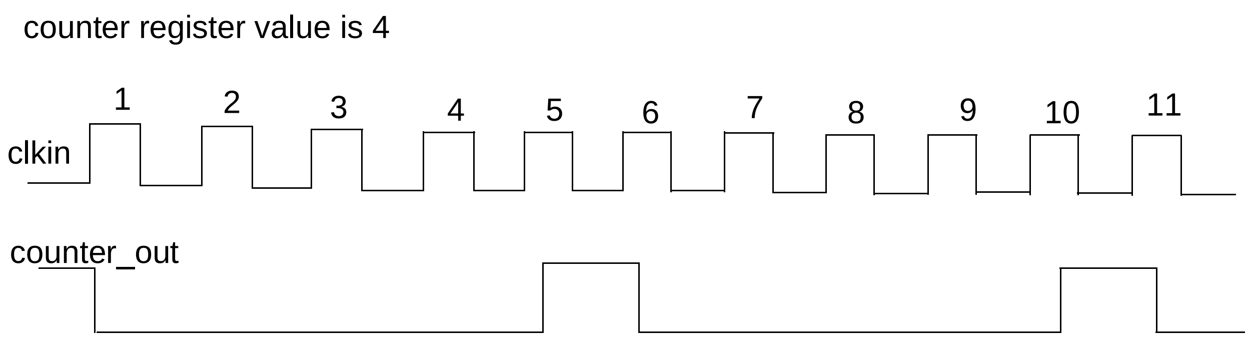


Figure 10-8. Timing Diagram for Prescaler or Timer1/Timer2 Terminal Counters (non_trigger_enable mode)

When Timer1 and Timer2 are in trigger_enable mode and operating as terminal counters, their outputs are as shown in *Figure 10-9*.

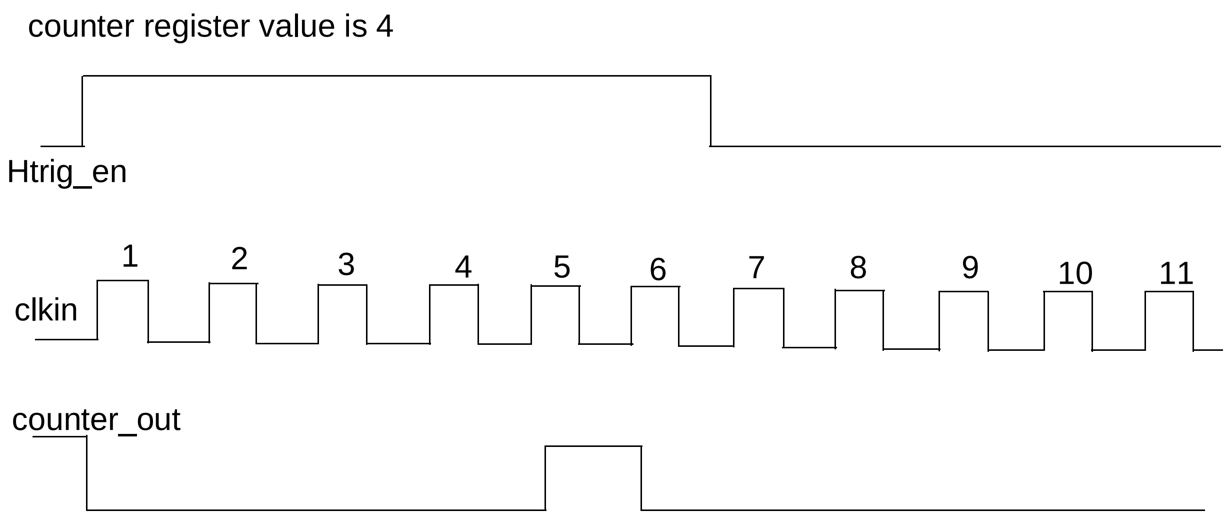


Figure 10-9. Timing Diagram for Timer1/Timer2 Terminal Counters (trigger_enable mode)

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 Timer1 or Timer2 Control Registers (0x29h, 0x2Ah) does not take effect until the end of the current cycle. See *Figure 10-10*.

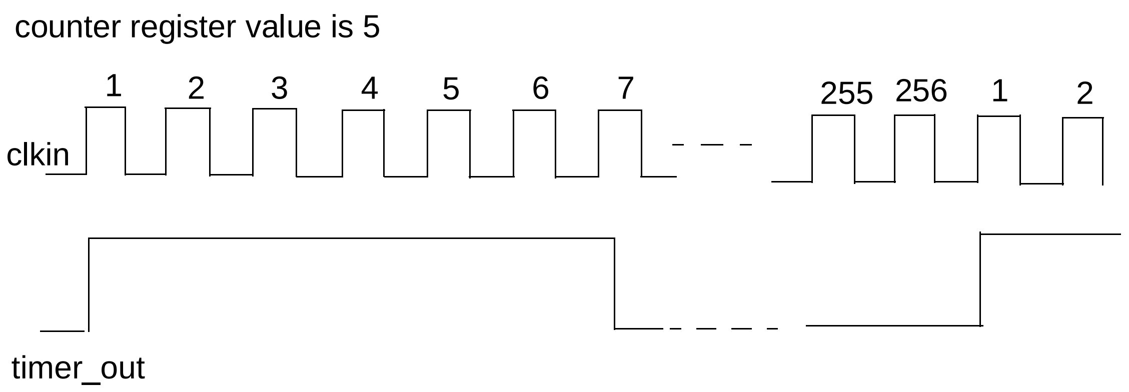


Figure 10-10. Timing Diagram for Timer1/Timer2 PWM (non_trigger_enable mode)

When Timer1 or Timer2 is set for PWM in trigger_enable mode, the output is as shown in *Figure 10-11*.

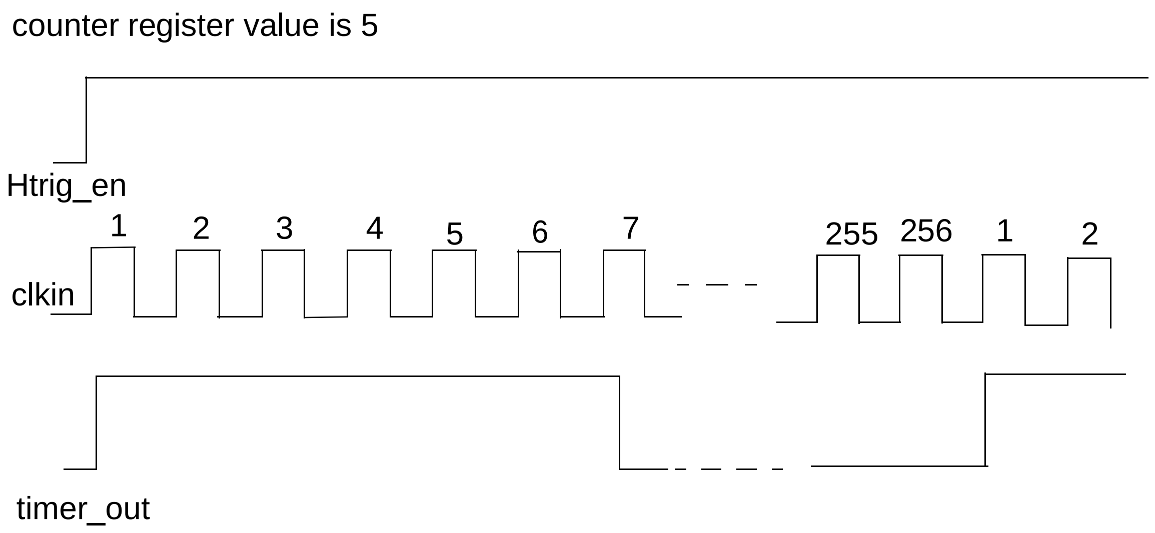
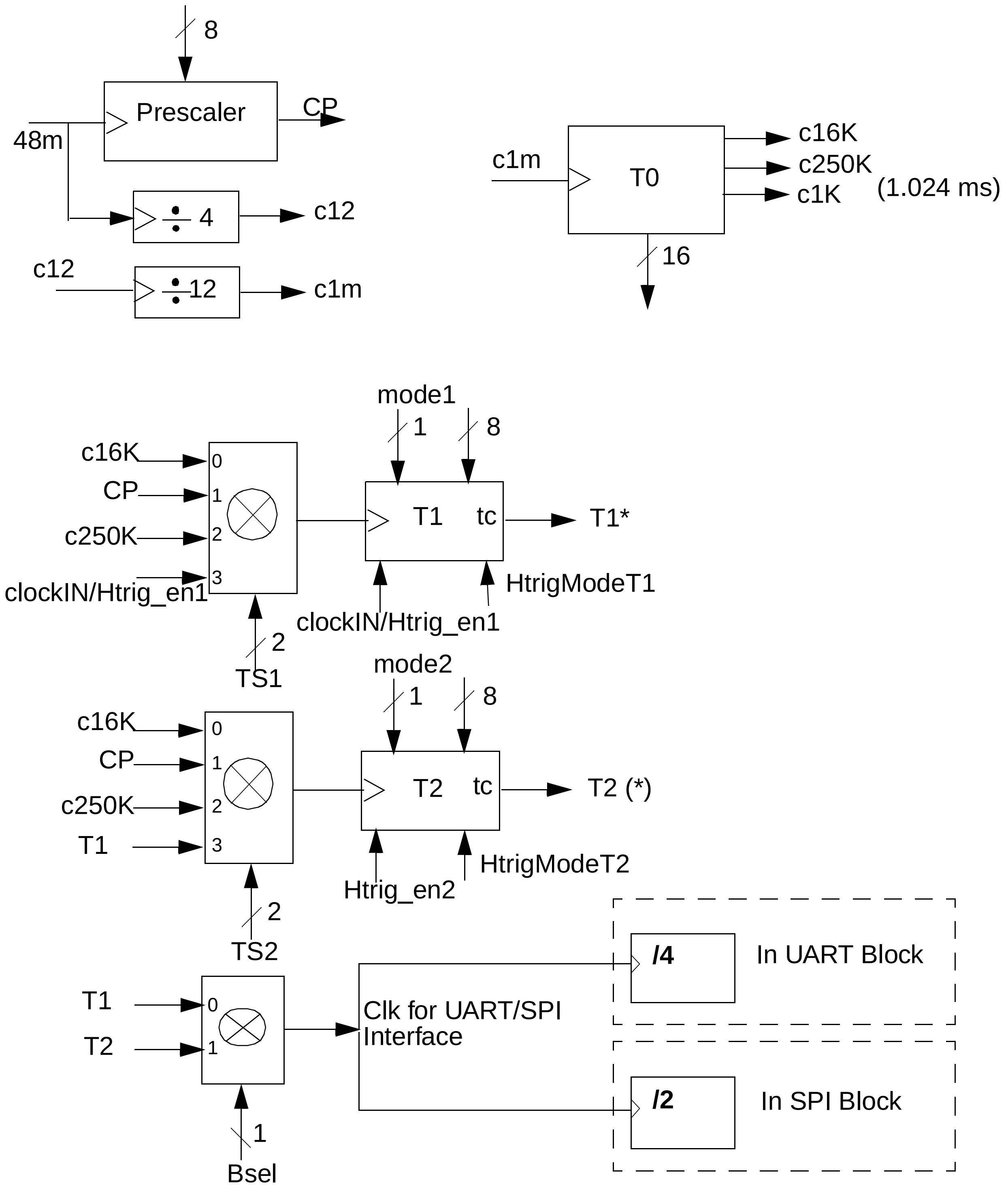


Figure 10-11. Timing Diagram for Timer1/Timer2 PWM (trigger_enable mode)



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

Figure 10-12. Timer Block Diagram

11.0 DMA Controller

The DMA Controller in the CY7C64213/313 allows high-speed data transfers without burdening the microcontroller. It is implemented to achieve a high throughput from/to the 1K FIFO and external package pins.

The DMA channel is shared among the SPI/UART, BHHI, SIE, and internal microcontroller. Therefore, only one of these interfaces 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 separate from the internal microcontroller bus to allow their concurrent operation: the internal microcontroller would not be stalled while the 1K FIFO is being accessed.

The internal microcontroller can access the 1K FIFO through registers 0x70h, 0x71h, 0x72h, and 0x73h. The SIE uses the DMA channel to access the 1K FIFO but does not use the DMA controller to generate addresses and bytes count; these are generated by the SIE itself. The external microcontroller has two choices to access the internal 1K FIFO: one is to use the 0x70h, 0x71h, 0x72h, 0x73h registers, and the other is to directly access it through the internal DMA bus.

12.0 1K FIFO for Endpoint 1–8

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 FIFO which resides on the internal DMA bus.

The 1K FIFO Address Registers (0x70h and 0x71h) store the address where the next read/write data will be accessed through 1K FIFO Data 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 register 0x072h will not increment register 0x70, accessing 0x73h will automatically increment 0x70h.

When the internal microcontroller performs a read or write to register 0x72h or 0x73h, a "Bus request" signal will be immediately asserted by the DMA controller to back off the internal microcontroller. In this way the DMA controller can arbitrate the internal microcontroller request. Once the 1K FIFO registers (0x70h, 0x71h, 0x72h, and 0x73h) have the control of the internal DMA bus, the DMA controller will release the "Bus Request" signal allowing the internal microcontroller to finish the cycle.

When an external microcontroller accesses the 1K FIFO Data Registers (0x72 or 0x73h), through BHHI, the transaction happens on the internal DMA bus, not the internal microcontroller 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
Addr7	Addr6	Addr5	Addr4	Addr3	Addr2	Addr1	Addr0

Figure 12-1. 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 12-2. 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 be able to access the 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 12-3. 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 12-4. 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 FIFO directly.

Accessing 1K FIFO DMA Data Register (0x73h) does not increment register 0x71h, so when register 0x70h reaches FFh, firmware has to update 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 12-5. DMA Stop Pointer Register 0x74h

In DMA mode, for UART/SPI and BHHI pushing mode operation, this register is used to compare with register 0x70 to stop the DMA operation, the DMA starts at the address in register {0x71h,0x70h} and stops at address in register {0x71h, 0x74h}.

The value of this register is not used for any DMA operations related to the SIE, internal microcontroller or BHHI in pulling mode.

After reset the values of bits[7:0] in the registers 0x70h, 0x71h, 0x72h, 0x73h and 0x74h are unknown.

13.0 GPIO Bidirectional Hardware Handshaking Interface (BHHI)

BHHI is a hardware asynchronous parallel interface that can transfer data to or from the internal 1K FIFO at up to 15 Mbytes/s using the DMA controller.

BHHI has two modes of operability: pulling mode, which is used to interface external microprocessor and the pushing mode that is used to support the PC parallel port 1284 modes. In pulling mode, the external microprocessor initiates a cycle with address and “strobe”, CY7C64213/313 functions as a slave. In pushing mode, the CY7C64213/313 is a master on the interface and controls the arbitration to allow another master to do limited accesses to 1K FIFO.

The pushing mode can be programmed for forwarding or reversing operation. The forwarding operation is for 1284 EPP/ECP-Forwarding mode, in this case CY7C64213/313 drives “strobe”, samples “RDY” and acts like a master on the bus. In reversing operation, the CY7C64213/313 samples “strobe” and drives “RDY”.

When a BHHI mode is enabled, the data movement is performed by internal DMA, the internal microcontroller is removed from the burden of moving data between the 1K FIFO and GPIO port. The CY7C64213/313 can be configured for one the following functions through BHHI:

Table 13-1. BHHI Function-Mode

CY7C64213/313 Function	Modes	
1. As a Microcontroller	Pushing Mode, Forwarding Operation	The CY7C64213/313 can be configured as a microcontroller that supports 8-bit data transfer, the address has to be generated by firmware.
2. As a USB controller that interfaces an external microprocessor	Pulling Mode	The operation is asynchronous and maximum bandwidth is 15 MByte/s. The external microprocessor supplies the address and initiates the cycle.
3. As a 1284 EPP/ECP Compatible Host	Pushing Mode, Forwarding/ Reversing Operation	The CY7C64213/313 is a master with capability of providing arbitration to support another master. The maximum bandwidth is 15 MByte/s. The internal DMA controller is used to generate the address for 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
DMA Start	Forward/ Reverse	Out/In	RDY Polarity Control	Enable $\overline{\text{FSTRB}}$ on Port1[2]	HHC2	HHC1	HHC0

Figure 13-1. 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 4 sets the polarity of RDY signal (Port1[2]). 0 is active HIGH, 1 is active LOW.

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

Bit 5 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.

After reset the values of bits[7:0] in the register 0x09h are at logic “0”.

Table 13-2. Bidirectional Hardware Handshake Interface Configuration

HHC[2:0]	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

HHC[2] is 0 for pulling mode (external microcontroller); HHC[2] is 1 for the pushing modes (1284 EPP/ECP 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.

When the BHHI is enable the GPIO control pins related to the BHHI interface will overwrite the previous GPIO port configurations and will be configured in the following way:

$\overline{\text{STRB}}$ and FRDY are in Open Drain mode.

FSTRB is in CMOS output mode

RDY is in CMOS output mode when active, left in three-state when not active. For this reason a pull-up or pull-down resistor is required, depending on the polarity chosen for RDY.

HHC	Accessing	Mode	AX[9]	AX[8]	AX[7:0]
001, 010	1K 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]

HHC	Accessing	Mode [1:0]
011	Data Transfer	0x
011	I/O Reg. Add.	10
011	256 RAM Add.	11

Figure 13-2. Address Mapping for BHHI in Pulling Mode

In BHHI Pulling Modes (HHC=001,010), address is directly sent into the device from the external micro; in 28-pin pulling mode (HHC=011), address is multiplexed on the data bus; in this mode an external microcontroller can access the internal 1K FIFO through 70h–73h registers. The mode[1:0] bits control the multiplexing.

ACK(s) are used in pulling modes (HHC=001,010,011) to inform the external micro that one of the USB Endpoint [1–8] has received or transmitted an ACK.

In isochronous data transfer, the ACK is also set if one of the Endpoint [1–8] receives/transfers 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 t_{CP} and t_{CR} reflects the 1K SRAM’s characteristic.

14.0 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 14-1. SPIO Data Register 0x0Ah

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

After reset the values of bits[6:0] in the register 0x0Ah are at logic “1”.

Notice that the CY7C64213 will always require that the data bits SPIO[6,5,3:0], related to unconnected pins, be written with a 0.

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 14-2. SPIO Interrupt Enable Register 0x0Bh

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

In the “10” mode (Configuration bits in the SPIO Configuration Register) this register is used to select CMOS Output or Open Drain modality (refer to *Table 14-2)*.

After reset the values of bits[6:0] in the register 0x0Bh are at logic “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 14-3. SPIO Configuration Register 0x0Ch

Bit 7 and 6 set to 1 enables the trigger for Timer1 and 2 and changes the definition of SPIO [6:5] in *Table 14-1* (refer to section 8.2). An external reset pin, XRST, can be provided as an option instead of SPIO[4] by setting bit 7 of the POR Configuration Register (0xF1h) to 0, in this case all SPIO[4] related functions defined in the table below are disabled.

TBit [5:3] select the mode configuration (see *Table 14-1*)

Table 14-1. 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	Reserved	Reserved
SPIO[6]	G	T2	T2	CO	CO	T2	Reserved	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 14-2*.

After reset the values of bits[7:0] in the register 0x0Ch are at logic “0” (setting the SPIO in GPIO mode).

.

Table 14-2. Port Configurations

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

For the explanation of the different SPIO Driver Mode listed in the table above refer to *“GPIO Configuration Port” on page 22.*

7	6	5	4	3	2	1	0
R/W							
External Reset Disable	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

Figure 14-4. POR Configuration Register 0xF1h

Bit 7 set to logic “1” will disable the external reset $\overline{\text{XRST}}$, to enable the $\overline{\text{XRST}}$ this bit has to be programmed with “0”.

After reset the value of bit[7] of register 0x1Fh is at logic “1” (External Reset disable). All the reserved bits should be written as “0”.

The external reset $\overline{\text{XRST}}$ is active LOW and it has the same effect as the Power-On Reset.

15.0 UART Interface

The Universal Asynchronous Receiver/Transmitter (UART) is a RS-232-compatible interface available in the CY7C64313. 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 15-1. UART Transmit Data Register 0x30h

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

After reset the values of bits[7:0] in the register 0x30h are “unknown”.

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 15-2. UART Receive Data Register 0x31h

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

After reset the values of bits[7:0] in the register 0x31h are unknown.

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 15-3. 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 UART 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 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 [1:0]	DMA Mode Selection
00	Disabled
01	Read Only
10	Write Only
11	Reserved

After reset the values of bits[5:0] in the register 0x32h are at logic “0”.

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 15-4. UART Status Register 0x33h

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

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

When bit 5 is set to 1 by hardware, it indicates that the Transmitter Data Register and the Transmit Shift register are both empty. This bit is set to logic 0 whenever the Transmit Data Register (0x30h) or the Transmit 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. When this bit is set to 1 a UART interrupt is generated (if enabled). This bit is cleared when the Transmitter Data Register (0x30h) is written by the CPU.

When bit 3 is set by hardware, it indicates that the newly received data byte has an invalid stop bit. When this bit is set to 1 an UART interrupt is generated (if enabled). The bit is cleared upon reading UART Status Register (0x33h).

When bit 2 is set by hardware, it indicates that the newly received data byte has an incorrect parity bit. When this bit is set to 1 an UART interrupt is generated (if enabled). The bit is cleared upon reading UART Status 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. When this bit is set to 1 a UART interrupt is generated (if enabled). The bit is cleared upon reading UART Status Register (0x33h).

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

After reset the values of bits[7,6,3:0] in the UART Status Register (0x33h) are at logic “0”, bits[5,4] are at logic “1”.

15.1 Non-DMA UART Mode

In Non-DMA mode the UART controller starts to transmit data when the firmware writes a data byte in the Transmit Data Register (0x30), when the transmission is finished the Transmit Empty Bit is set to 1.

When a byte of data is received by the UART, the Receiver Data Ready bit is set to 1, the firmware can read the byte from the Receive Data Register.

In non-DMA mode, the UART generates the interrupt in the following condition:

- When either the Transmitter Empty or Transmitter Holding Empty bit is set, depending on the state of the Transmitter Interrupt Source Selection.
- When the Receiver buffer is full.
- If an Error condition is detected.

15.2 DMA UART Mode

In DMA mode, the UART generates DMA read and write requests to the DMA controller. While transmitting, the UART will generate a DMA write request if the Transmitter Holding Register is empty. While receiving, the UART will generate a DMA read request if the receiver buffer is full. After the DMA has finished transferring all bytes in the UART the DMA controller will generate a UART interrupt to the internal microcontroller. A UART interrupt is also generated if an error condition is generated.

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 compared to 1K FIFO Address Register (0x70h) to determine where the DMA should stop. 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 will start the operation. If DMA is enabled, the firmware should ensure that registers 0x70h, 0x71h, 0x72h, 0x73h and 74h are not changed until the DMA operation is finished.

15.3 UART Baud Rate

The UART in the CY7C64313 supports 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=1 or 2, bit 2 of Register 0x2C is use to select Timer1or 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 15-1* lists the common baud rates supported in the CY7C64313, with the recommended combinations for the Prescaler and TimerX.

Table 15-1. 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%

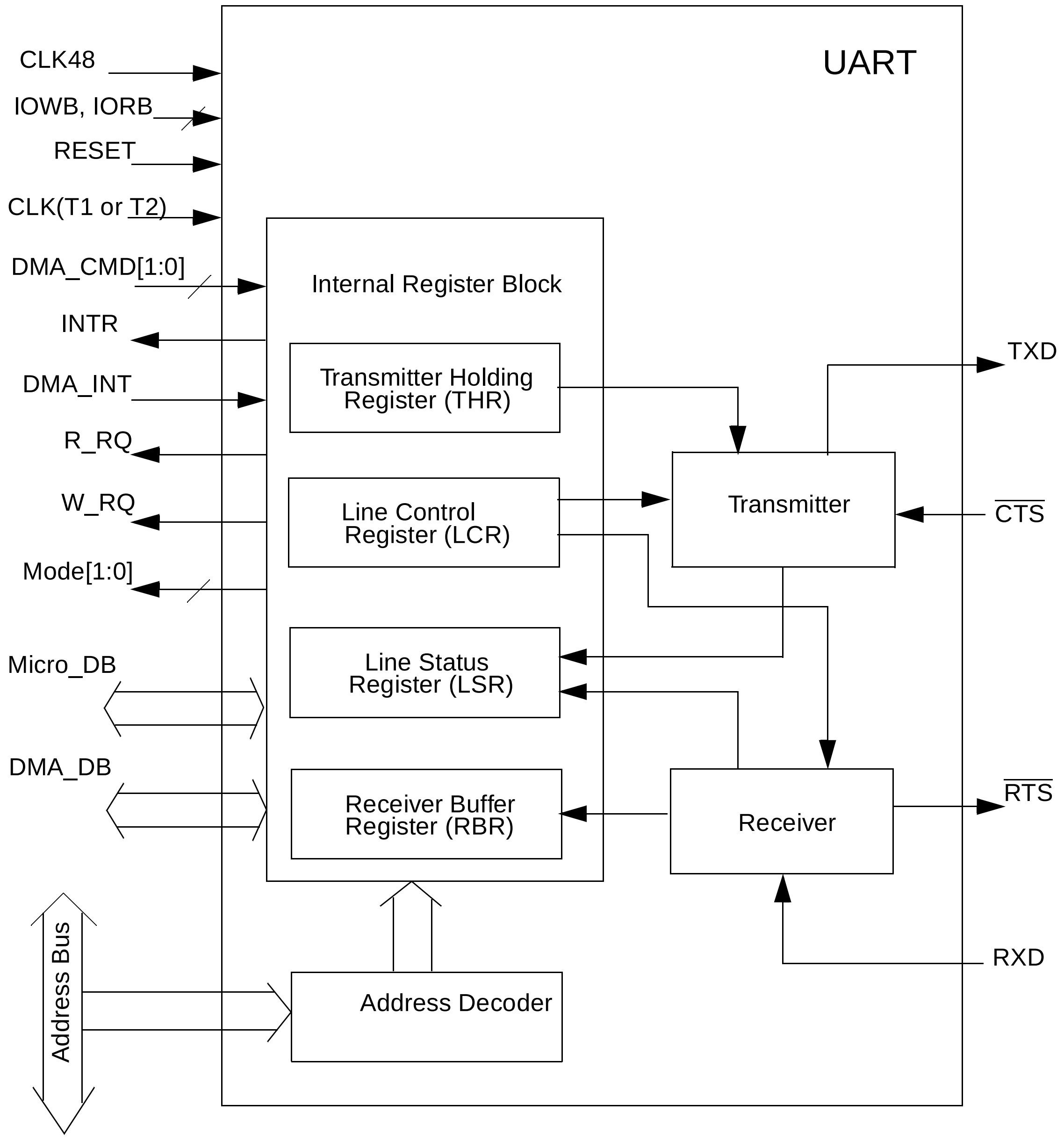


Figure 15-5. UART Block Diagram

16.0 SPI Interface

The Serial Peripheral Interface (SPI) is a synchronous serial communication protocol. It allows the microcontroller unit in the CY7C64313 to communicate with serial peripheral devices. The system can be used as a master or a slave device. In addition, the CY7C64313 can be configured in 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 16-1. SPI Transmit Data Register 0x30h

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

After reset the values of bits[7:0] in the register 0x30h are “unknown”.

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

Figure 16-2. SPI Receive Data Register 0x31h

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

After reset the values of bits[7:0] in the register 0x31h are “unknown”.

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

Figure 16-3. SPI Control Register 0x32h

Bit 6 set to 1 is used to drive the SPI clock in the correct polarity (according with the setting on bit 3) before the firmware generates $\overline{\text{SS}}$ through the GPIO (Non-DMA SPI Mode).

Bit 5 set to 1 will enable the SPI interface.

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

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

After reset the values of bits[6:0] in the register 0x32h are at logic “0”.

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

Figure 16-4. SPI Status Register 0x33h

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

Figure 16-5. SPI Status Register 0x33h

Bit 4 of register 0x33h is set each time the SPI Transmit Data Register (0x30h) transfers a byte to the Shift Register. An interrupt request to the CPU is generated (in Non-DMA mode) 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 the CY7C64213/313 SPI is configured as a master device, in a multimaster and multislave 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 the CY7C64213/313 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 the CY7C64213/313 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.

When bit 3 is set to 1 an interrupt will generate to the CPU. This bit is cleared by reading the 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 the SPI Status Register (0x33h) and then reading the SPI Receive Data Register (0x31h).

Bit 0 is set each time a byte is moved from the Shift Register to the Receive Data Register. An interrupt request to the CPU will be generated (in Non-DMA mode) if the SPI interrupt enable bit in the Global Interrupt Enable Register (0x20h) is set. Any read of the SPI Receive Data Register clears this bit.

After reset the values of bits[4:0] in the register 0x33h are at logic “0”.

16.1 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 and 0x71h). 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 operation will start. If DMA is enabled, the firmware should ensure that registers 0x70h, 0x71h, 0x72h, 0x73h and 0x74 are not changed until the DMA is done.

For a DMA SPI Read, when the DMA “read only mode” is enabled, the SPI interface starts to transmit the SPI clock (master mode) or receive the SPI clock (slave mode). After the SPI receives one byte, it moves the data from the Shift Register to the Receive Data Register. Then, the DMA moves the data to the proper 1K FIFO location. After the last byte is moved to the FIFO, the DMA will disable itself.

For a DMA SPI Write, when the DMA “write only mode” is enabled, the DMA controller 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 into the transmit data register, the DMA will disable itself.

After the DMA has finished transferring all bytes, the SPI controller will generate the SPI interrupt to the internal microcontroller. An SPI interrupt is also generated if an error condition is generated.

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

16.2 Non-DMA SPI Mode

In master mode, the transmission begins by writing a byte to the Transmit Data Register (0x30h). If the Transmit Shift Register is empty, data is parallel loaded into the 8-bit Transmit Shift Register. After data is moved from the Transmit Data Register (0x30h), the SPI controller sets the SPI Transmit Empty (bit 4) of the SPI Status Register (0x33h). If the SPI Interrupt Enable (bit 3) of the Global Interrupt Enable Register (0xFFh) is set, an interrupt will be generated to the microcontroller. The byte begins serially shifted out via the MOSI pin to the slave device.

Every time a bit is transmitted through the Transmit Shift register a bit gets inside the Receive Shift register through the MISO pin. This causes the SPI Receive Full (bit 0) of the SPI Status Register (0x33h) to be set to 1 at the end of transmission; therefore the firmware has to perform a “dummy” read to the SPI Receive Data Register (0x31h) to avoid an overflow condition in the next transmission.

In order to receive a data byte from the slave the firmware must load “dummy” data in the SPI Transmit Data Register (0x30h) to enable the SPI clock; after the Receive Shift Register is full the data byte will be parallel transferred to the SPI Receive Data Register (0x31h) and the SPI Receive Full (bit 0) of the SPI Status Register (0x33h) will be set. If the SPI Interrupt Enable (bit 3) of the Global Interrupt Enable Register (0xFFh) is set, a interrupt will be generated to the microcontroller.

In slave mode, the CY7C64213/313 slave start logic receives a logic LOW at the $\overline{\text{SS}}$ 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 Receive Shift Register. After a byte enters the Receive Shift Register of the slave, it is transferred to the Receive Data Register (0x31h), and the SPI Receive Full (bit 0) of the SPI Status Register (0x33h) will be set.

When the master of the SPI bus starts a transmission, the data in the slave Transmit Shift Register begins shifting out on the MISO pin to the master SPI. Data that is transferred from slave to master SPI must be loaded from the Transmit Data Register (0x30h) to the Transmit Shift Register prior to the master starting the transmission. After data is loaded from the Transmit Data Register (0x30h) to the Transmit Shift Register, SPI Transmit Empty (bit 4) of the SPI Status Register (0x33h) will be set. A new byte can be written to the Transmit Data Register (0x30h) for the next transmission.

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

16.3 SPI Baud Rate

The UART in the CY7C64313 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 16-1* lists some of the baud rates supported in the CY7C64313, with the recommended combinations for the Prescaler and TimerX.

Table 16-1. SPI Baud Rate Table

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

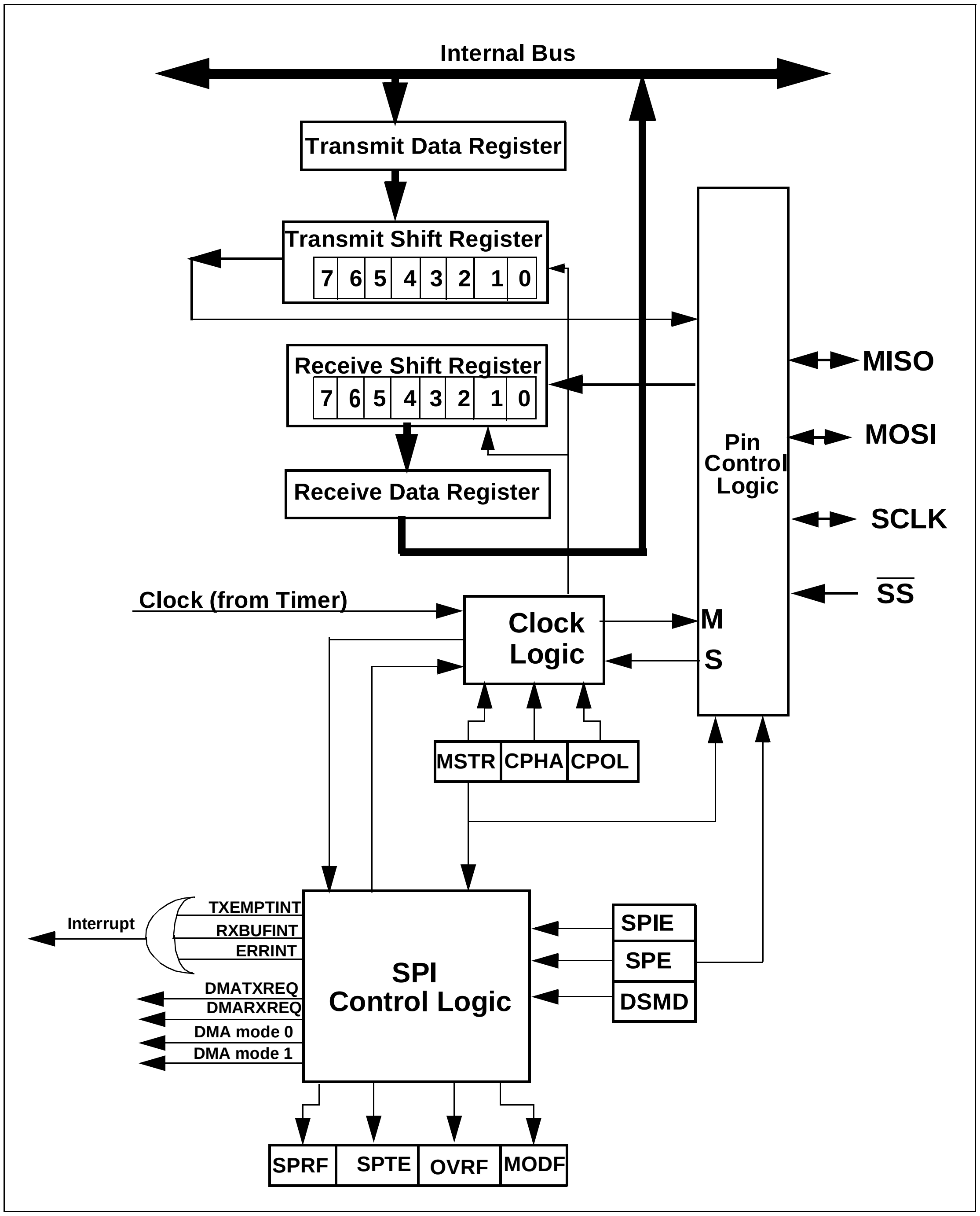


Figure 16-6. SPI Block Diagram

17.0 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	Interrupt Mask	Reserved	Run

Figure 17-1. 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 until reset (Power-on or Watch Dog).

Bit 1 is reserved and must be set to zero.

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” (bit 3) will halt the processor and cause the microcontroller to enter the “suspend” mode that significantly reduces power consumption. Refer to section 8.0 “Suspend Mode” for more details on suspend mode operation.

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. 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 to 1 during a reset initiated by the watchdog timer. See section 7.3 Watch Dog Reset (WDR).

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, the Processor Status and Control Register is set to 0x010001, which indicates a Power-on Reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear); the processor is running (bit 0 set). Note that during the 96-ms suspend at start-up (explained in Section 7.1), a Watch Dog Reset will also occur unless this suspend is aborted by an USB Bus Reset.

During a Watch Dog Reset, the Processor Status and Control Register is set to 010x0001, which indicates a Watch Dog Reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear); the processor is running (bit 0 set).

18.0 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 18-1. Global Interrupt Enable Register 0x20h

After reset the values of bits[6:0] in the register 0x20h are at logic “0” (interrupts disabled).

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 18-2. USB End Point Interrupt Enable Register 0x21h

After reset the values of bits[1:0] in the register 0x21h are at logic “0” (interrupt disabled).

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 interrupt controller contains a separate latch for each interrupt. See *Figure 18-3* for the logic block diagram for the interrupt controller. When an interrupt is generated it is latched as a pending interrupt. It will stay as a pending interrupt until it is serviced or a reset occurs. A pending interrupt will only generate an interrupt request if it is enabled in the Global Interrupt Enable Registers (0x20h,0x21h).

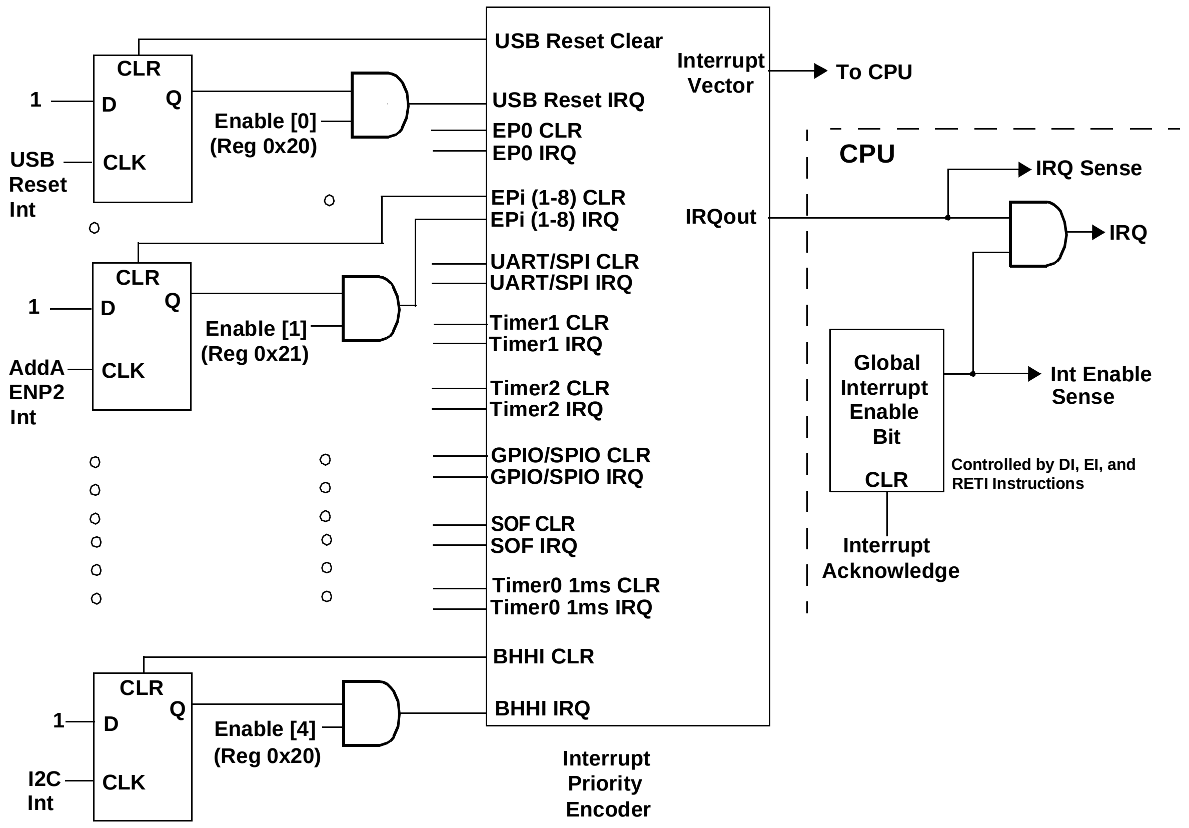


Figure 18-3. Interrupt Controller Logic Block Diagram

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.

18.1 Interrupt Vectors

The Interrupt Vectors supported by the USB Controller are listed in *Table 18-1*. Although Reset is not an interrupt, per se, the first instruction executed after a reset is at ROM 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 18-1. 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

7	6	5	4	3	2	1	0
			R/W	R/W	R/W	R/W	R/W
Reserved	Reserved	Reserved	Interrupt Vector Bit 4	Interrupt Vector Bit 3	Interrupt Vector Bit 2	Interrupt Vector Bit 1	Always “0”

Figure 18-4. Interrupt Vector Register 0x23h

This register can be used to read the Interrupt Vector of the next pending interrupt when interrupts are disabled (Global Interrupt Enable Register 0x20h is cleared).

Any write to this register will clear all the pending interrupts.

18.2 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. With a 6-MHz crystal, the instruction clock is 12 MHz so 16 clocks is 1.333 µs and 20 clocks 1.666 µs.

18.3 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 ends. The USB Controller recognizes a USB Bus Reset when a Single Ended Zero (SE0) condition persists for at least 12–16 µs (the Reset may be recognized for an SE0 as short as 12 µs, but will always be recognized for an SE0 longer than 16 µs). 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.

18.4 BHHI Interrupt

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

18.5 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.

18.6 UART/SPI Interrupt

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

18.7 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.

18.8 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.

A block diagram of the GPIO interrupt logic is shown in *Figure 18-5* below. The interrupt polarity is selected through the GPIO Configuration Register (0x08h). If the selected signal polarity is detected on the I/O pin a HIGH signal is generated. If the Port Interrupt Enable bit for this pin is HIGH and no other ports pins are requesting interrupts, the OR gate will issue a LOW-to-HIGH signal to clock the GPIO interrupt flip-flop. The output of the flip-flop is further qualified by the Global GPIO Interrupt Enable bit before it is processed by the Interrupt Priority Encoder. Both the GPIO interrupt flip-flop and the Global GPIO Enable bit are cleared during GPIO interrupt acknowledge by on-chip hardware.

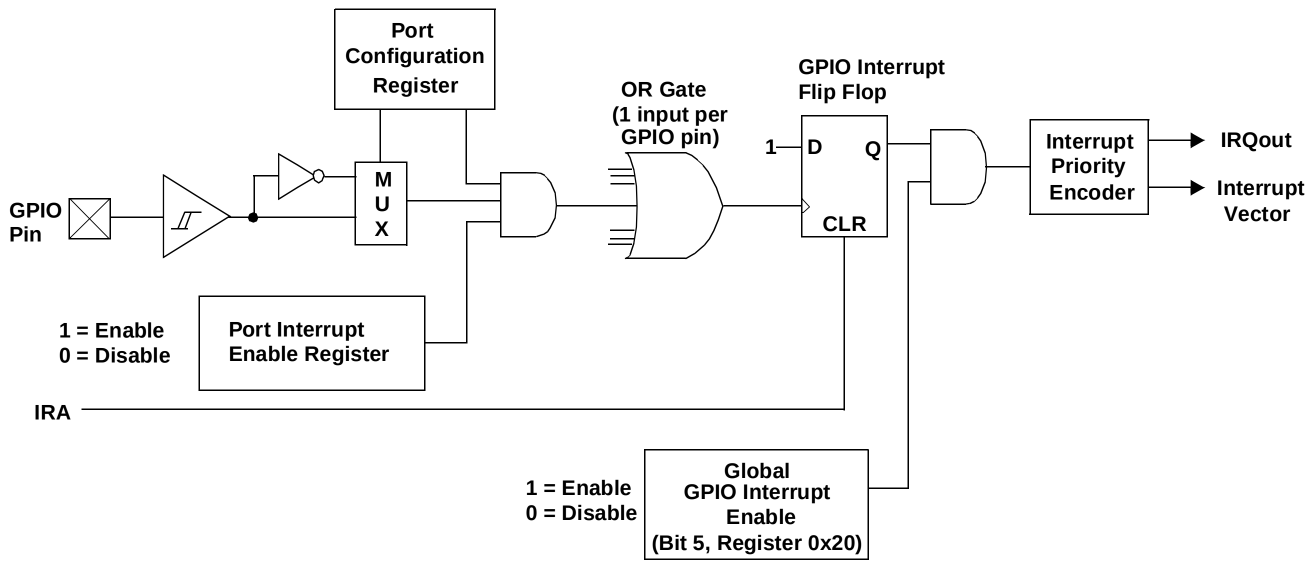

Port Configuration Register
OR Gate (1 input per GPIO pin)
GPIO Interrupt Flip Flop
1
D
Q
CLR
Interrupt Priority Encoder
IRQout
Interrupt Vector
GPIO Pin
M U X
1 = Enable 0 = Disable
Port Interrupt Enable Register
IRA
1 = Enable 0 = Disable
Global GPIO Interrupt Enable (Bit 5, Register 0x20)

Figure 18-5. GPIO Interrupt Logic Block Diagram

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.

18.9 Start Of Frame Interrupt (SOF)

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

19.0 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).

19.1 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

19.2 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 19-1*. 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
Reserved	Reserved	D+	D–	Bus Activity	Control Bit 2	Control Bit 1	Control Bit 0

Figure 19-1. USB Status and Control Register 0x1Fh

Bit [5:4] can be used by firmware to monitor the state of D+ and D– lines

Bit 3, 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. 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 [2:0]	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

Reserved bits [7,6] must to be written with “0”.

After reset the values of bits[7,6,3:0] in the register 0x1Fh are at logic “0”.

20.0 USB Device

The USB device includes up to nine endpoints: EP0, and EP1 to EP8. Endpoint 0 (EP0) is always used as the “control” Endpoint. Endpoint 1–8 can be configured for “Interrupt”, “Isochronous”, or “Bulk” transactions.

The USB Controller provides one USB Device Address Register. The USB Device Address Register contents are cleared during a reset and also during USB Bus Reset, setting the USB device address to zero and marking this address as disabled. *Figure 20-1* 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 20-1. 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.

20.1 USB Device Endpoints

In the CY7C64213/313, EP0 (Endpoint 0) has its FIFO in the 256 byte data RAM space. 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 1K FIFO RAM space. The data movement is done by the 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 20-2. USB Endpoint FIFO Control Register 0x19h

Bits [2:0] select the Endpoint Configuration according to *Figure 20-3*, below.

Bits [7:3] are the extension of the USB Device EP 1–8 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.

After reset the values of bits[7:0] in the register 0x19 are at logic “0”.

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 20-3. 1K FIFO Mapping

The FIFO size of each Endpoint (1–8) is determined by 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. The maximum FIFO size supported for one Endpoint is 1023 bytes.

The addressing of the 1K FIFO is done based on the mapping selected in the *Figure 20-3*. 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 0x07Fh. The Endpoint 2 FIFO starts at address 0x080h and ends at 0x0FFh. The Endpoint 8 FIFO starts at 0x380h and ends at 0x3FFh. In the mapping of FFC=110 or 111, the Endpoint 1 FIFO starts at 000x and ends at 0x3FEh. 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 0x3FFh, and in the second case (*128), the Endpoint 2 FIFO starts at 0x380h 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 USB Device EPi (1–8) Mode Register of each Endpoint, there is an enable bit for back-to-back mode (bit 5). When this is enabled, the next USB transmission/reception will not start at location 0 of the Endpoint’s FIFO. Instead, it will start from the value (address) loaded on the Back-to-back Starting Address Register (0x17h). Before enabling the back-to-back feature the firmware has to write the correct “starting address” on register 0x17h and the correct “ending address + 1” in the respective EP count register (refer to USB Device EPi (1–8) Counter Registers section). This allows the FIFO to be used more efficiently for bulk or isochronous transactions where back-to-back is 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. In addition, if the packet 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 bit of the Endpoint Mode Register to find out if the assertion of ACK is a 128-byte early interrupt or an end-of-packet ACK interrupt. If an ACK on the pin is asserted and the corresponding ACK bit of the Endpoint Mode 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 20-4*.

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 SETUP	ENDP0 IN	ENDP0 OUT	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0

Figure 20-4. 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.

Bit 4, the ACK bit, is set whenever the SIE engages in a transaction that completes with an ‘ACK’ packet. The ACK bit must be cleared by firmware

The ‘SETUP’ 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 (SETUP... 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 ‘SETUP’ 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 20-1*.

After reset the values of bits[7:0] in the register 0x12 are at logic “0”

Table 20-1. 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 20-5. USB Device EPi (1–8) Stall Mode Register 0x13h

The USB Device EPi (1–8) 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.

After reset the values of bits[7:0] in the register 0x13 are at logic “0”.

7	6	5	4	3	2	1	0
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Data Toggle	Data Valid	B2B Enable	ACK	Mode3	Mode2	Mode1	Mode0

Figure 20-6. USB Device EPi (1–8) Mode Registers 0x41h, 0x43h, 0x45h, 0x47h, 0x49h, 0x4Bh, 0x4Dh, 0x4Fh

Bit [3:0] of these registers have the same meaning of the Mode bits in the USB Device EP0 Mode Register (0x12h) and control how the Endpoints respond to USB traffic (refer to Endpoint Mode Encoding Definition *Table 20-1*).

Bit 4, the ACK bit, is set whenever the SIE engages in a transaction that completes with an ‘ACK’ packet. When this bit is set to 1 the related Endpoint ACK bit in the USB Device EPi ACK Status Register (0x18h) is also set to 1. The ACK bit must be cleared by firmware

Bit 5 set to “1” enables the back-to-back transactions for the corresponding endpoint.

The Data Valid bit (bit 6) is used for OUT Data tokens only.

Data 0/1 Toggle bit (bit 7) selects the DATA packet’s toggle state: 0 for DATA0, 1 for DATA1.

After reset the values of bits[7:0] in the registers 0x41h, 0x43h, 0x45h, 0x47h, 0x49h, 0x4Bh, 0x4Dh, 0x4Fh are at logic “0”.

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 20-7. 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 plus 2 CRC bytes (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.

After reset the values of bits[7:0] in the register 0x11 are at logic "0".

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 20-8. USB Device EPi (1–8) Counter Registers 0x40h, 0x42h, 0x44h, 0x46h, 0x48h, 0x4Ah, 0x4Ch, 0x4Eh

Bits[7:0] of the USB Device EPi (1–8) Counter Registers indicate the number of data bytes to be transmitted during IN transfers or received during OUT transfer. For Endpoint sizes greater than 128 bytes, the EP(1–4)X0/1 bits [7:3] in register 0x19h provide the additional count bits.

If the back-to-back is enabled this register has the following meaning:

- for OUT transfer this register contains the number of bytes received.
- for IN transfer this register has to be loaded with the "ending address + 1" by firmware (i.e., if in the previous IN transfer 8 bytes were transmitted starting from address 0, of the specific endpoint, and we want to transmit another 8 bytes in the new IN transfer, this register has to be loaded with the value 11h).

After reset the values of bits[7:0] in the registers 0x40h, 0x42h, 0x44h, 0x46h, 0x48h, 0x4Ah, 0x4Ch, 0x4Eh are at logic "0".

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 20-9. USB Device EPi ACK Status Register 0x18h

The USB Device EPi ACK Status Register register is used to inform an external microcontroller that a transaction in a specific endpoint is done and an ACK is received/sent. The bits in this register are also set to "1" every 128 bytes of transfer if the packet is bigger than 128 bytes.

To clear this register the firmware has to read this register and clear the ACK bit (bit 4) in the USB Device EPi (1-8) Mode Register for the related Endpoint.

After reset the values of bits[7:0] in the register 0x11 are at logic "0".

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
B2B Starting Address 7	B2B Starting Address 6	B2B Starting Address 5	B2B Starting Address 4	B2B Starting Address 3	B2B Starting Address 2	B2B Starting Address 1	B2B Starting Address 0

Figure 20-10. Back-to-back Starting Address Register 0x17h

Bits [7:0] are the Starting Address for the new transfer when back-to-back feature is enabled.

21.0 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

22.0 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	V_{CC}=5.25V
V_{REF}	Voltage reference Output	3	3.6	V	I_{REF}<=500uA
I_{SB1}	Supply Current - Suspend Mode		40	µA	Oscillator off, D– min $\geq$ 2.8V
V_{pp}	Programming Voltage (disabled)	–0.4	0.4	V	
C_{xtal}	Internal Crystal Capacitance/pin		30	pF	V_{CC} = 5.0V, XTALIN,XTALOUT
I_{il}	Input Leakage Current		1	µA	Any pin
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		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 23-2*
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_{OH}	Static Output HIGH	2.8	3.6	V	R_L of 1.5 kΩ to 3.6V
V_{OL}	Static Output LOW		0.3	V	R_L of 1.5 kΩ to V_{SS}
Z_{DRV}	Driver Output Resistance (Steady State Drive)	6	22	Ω	Z_{DRV} (refer to section 6.0)
C_{IN}	Transceiver Capacitance		20	pF	Pin to V_{SS}
	General Purpose I/O and Special Purpose I/O				
R_{up}	Pull-up resistance	8K	22K	Ohms	All ports (defines Voh) in resistive mode
V_{IH}	Input High Voltage	2		V	
V_{IL}	Input Low Voltage		0.8	V	
V_{OH1}	Output High Voltage	2.4		V	I_{OH}= –4 mA CMOS output mode
V_{OL1}	Output Low Voltage		0.4	V	I_{OL}= 4 mA CMOS output mode
	Clock Input				
V_{IHX}	Input High Voltage	V_{CC}–0.8		V	XTALIN pin only
V_{ILX}	Input Low Voltage		0.8	V	XTALIN pin only

23.0 Switching Characteristics

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

Parameter	Description	Min.	Max.	Unit
	Clock			
f_{OSC}	Clock Rate	6 $\pm$ 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 for High-speed Device[2]			
t_{FR}	Transition Rise Time	4	20	ns
t_{FF}	Transition Fall Time	4	20	ns
t_{FRFM}	Rise / Fall Time Matching; (t_{FR}/t_{FF})	90	111.11	%
V_{CRS}	Output Signal Crossover Voltage	1.3	2.0	V
	USB Data Signal Timing for High-speed Device			
t_{FDRATE}	Full Speed Date Rate	11.9700	12.0300	Mb/s
t_{FRAME}	Frame Interval	0.9995	1.0005	ms
t_{RFI}	Consecutive Frame Interval Jitter (No clock adjustment)		42	ns
t_{RFIADJ}	Consecutive Frame Interval Jitter (With clock adjustment)		126	ns
T_{DJ1}	Source Differential Driver Jitter To Next Transition (Notes 3 and 4 and *Figure 23-6*)	–3.5	3.5	ns
T_{DJ2}	Source Differential Driver Jitter For Paired Transitions (Notes 3 and 4 and *Figure 23-6*)	–4.0	4.0	ns
T_{FEOPT}	Source SE0 interval of EOP (Note [4] and *Figure 23-5*)	160	175	ns
T_{FDEOP}	Source Jitter for Differential Transition to SE0 Transition (Note 4 and *Figure 23-5*)	–2	5	ns
T_{JR1}	Receiver Data Jitter Tolerance To Next Transition (Note 4 and *Figure 23-4*)	–18.5	18.5	ns
T_{JR2}	Receiver Data Jitter Tolerance For Paired Transitions (Note 4 and *Figure 23-4*)	–9	9	ns
T_{FEOPR}	Receiver SE0 interval of EOP (Note 4 and *Figure 23-5*)	82		ns
T_{FST}	Width of SE0 Interval During Differential Transition		14	ns
	Bidirectional Hardware Handshake Interface (BHHI) Timing[5]			
	BHHI (Pulling Mode, Read from 1K FIFO)[6]			
t_{CP}	Command Pulse Width	46		ns
t_{CR}	Command Recovery Time	21		ns
t_{AS}	Address Set-up	5		ns
t_{AH}	Address Hold	5		ns
t_{RDYS}[7]	RDY Assertion from the Assertion of $\overline{STRB}$		45	ns
t_{RDYS}[8]	RDY Assertion from the Assertion of $\overline{STRB}$		170	ns
t_{RDYH}	RDY Deassertion from the Deassertion of $\overline{STRB}$	5	22	ns
t_{RDS}	RDY Assertion to Data Valid		0	ns
t_{RDH}	$\overline{STRB}$ Deassertion to Data Not Valid	5	22	ns
	BHHI (Pulling Mode, Read from 256 Bytes RAM or I/O Register)			
t_{CP}	Command Pulse Width	192		ns

Parameter	Description	Min.	Max.	Unit
t_{CR}	Command Recovery Time	21		ns
t_{AS}	Address Set-up	5		ns
t_{AH}	Address Hold	5		ns
t_{RDYS}[9]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		182	ns
t_{RDYS}[10]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		385	ns
t_{RDYH}	RDY Deassertion from the Deassertion of $\overline{\text{STRB}}$	5	22	ns
t_{RDS}	RDY Assertion to Data Valid		0	ns
t_{RDH}	$\overline{\text{STRB}}$ Deassertion to Data Not Valid	5	22	ns
	BHHI (Pulling Mode, Write to 1K FIFO)[11]			
t_{CP}	Command Pulse Width	46		ns
t_{CR}	Command Recovery Time	21		ns
t_{AS}	Address Set-up	5		ns
t_{AH}	Address Hold	5		ns
t_{RDYS}[7]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		36	ns
t_{RDYS}[8]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		182	ns
t_{RDYH}	RDY Deassertion from the Deassertion of $\overline{\text{STRB}}$	5	22	ns
t_{WDS}	Data Valid to $\overline{\text{STRB}}$ Deassertion	20		ns
t_{WDH}	$\overline{\text{STRB}}$ Deassertion to Data Not Valid	5	22	ns
	BHHI (Pulling Mode, Write to 256 Bytes RAM or I/O Register)			
t_{CP}	Command Pulse Width	192		ns
t_{CR}	Command Recovery Time	21		ns
t_{AS}	Address Set-up	5		ns
t_{AH}	Address Hold	5		ns
t_{RDYS}[9]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		182	ns
t_{RDYS}[10]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		406	ns
t_{RDYH}	RDY Deassertion from the Deassertion of $\overline{\text{STRB}}$	5	22	ns
t_{WDS}	Data Valid to $\overline{\text{STRB}}$ Deassertion	20		ns
t_{WDH}	$\overline{\text{STRB}}$ Deassertion to Data Not Valid	5	22	ns
	BHHI (Pushing Mode, Forwarding Operation, Write/Read)			
t_{CP}	Command Pulse Width	42		ns
t_{CR}	Command Recovery Time	21		ns
t_{CH}	$\overline{\text{FSTRB}}$ Assertion from the Deassertion of RDY	5		ns
t_{RDYR}[12]	$\overline{\text{FSTRB}}$ Deassertion from the Assertion of RDY		45	ns
t_{RDYR}[13]	$\overline{\text{FSTRB}}$ Deassertion from the Assertion of RDY		182	ns
t_{RDYH}	FRDY Deassertion from the Deassertion of $\overline{\text{FSTRB}}$	5		ns
t_{WDS}	Write Data Set-up to the Assertion of $\overline{\text{FSTRB}}$	0		ns
t_{WDH}	Write Data Hold after the Deassertion of $\overline{\text{FSTRB}}$	5	22	ns
t_{RDS}	Read Data Ready to the Assertion of FRDY	0		ns
t_{RDH}	Read Data Hold after Deassertion of $\overline{\text{FSTRB}}$	5	22	ns
	BHHI (Pushing Mode, Reversing Operation, Read)			
t_{CP}	Command Pulse Width	46		ns

Parameter	Description	Min.	Max.	Unit
t_{CR}	Command Recovery Time	21		ns
t_{CH}	$\overline{\text{STRB}}$ Assertion from the Deassertion of RDY	5		ns
t_{RDYS}[7]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		36	ns
t_{RDYS}[8]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		161	ns
t_{RDYH}	RDY Deassertion from the Deassertion of $\overline{\text{STRB}}$	5	22	ns
t_{RDS}	RDY Assertion to Data Valid		0	ns
t_{RDH}	$\overline{\text{STRB}}$ Deassertion to Data Not Valid	5	22	ns
	BHHI (Pushing Mode, Reversing Operation, Write)			
t_{CP}	Command Pulse Width	46		ns
t_{CR}	Command Recovery Time	21		ns
t_{CH}	$\overline{\text{STRB}}$ Assertion from the Deassertion of RDY	5		ns
t_{RDYS}[7]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		36	ns
t_{RDYS}[8]	RDY Assertion from the Assertion of $\overline{\text{STRB}}$		182	ns
t_{RDYH}	RDY Deassertion from the Deassertion of $\overline{\text{STRB}}$	5	22	ns
t_{WDS}	Data Valid to $\overline{\text{STRB}}$ Deassertion	20		ns
t_{WDH}	$\overline{\text{STRB}}$ Deassertion to Data Not Valid	5	22	ns
	SPI Timing [Master Mode][14]			
$t_{cyc(M)}$	Cycle Time [15] (max. 6 MHz)	166.6		ns
$t_{w(SCKH)M}$	Clock (SCK) High Time	75		ns
$t_{w(SCKL)M}$	Clock (SCK) Low Time	75		ns
$t_{SU(M)}$	Data Set-up Time, Inputs	30		ns
$t_{H(M)}$	Data Hold Time, Inputs	5		ns
$t_{V(M)}$	Enable Assertion to Output Data Valid		15	ns
$t_{HO(M)}$	Output Data Hold after Enable Deassertion	2		ns
	SPI Timing [Slave Mode][14]			
$t_{cyc(S)}$	Cycle Time[15](max. 6 MHz)	166.6		ns
$t_{w(SCKH)S}$	Clock (SCK) HIGH Time	75		ns
$t_{w(SCKL)S}$	Clock (SCK) LOW Time	75		ns
t_{LEAD}	$\overline{\text{SS}}$ Assertion to First Clock Edge	15		ns
t_{LAG}	Last Clock Edge to Deassertion of $\overline{\text{SS}}$	80		ns
$t_{SU(S)}$	Data Set-up Time, Inputs	5		ns
$t_{H(S)}$	Data Hold Time, Inputs	10		ns
$t_{A(CP0)}$	$\overline{\text{SS}}$ Assertion to Valid Data Out (Clock Phase = 0) [16]		40	ns
t_{DIS}	Slave Disable Time [17]		30	ns
$t_{V(S)}$	Enable Edge to Valid Data Out		40	ns
$t_{HO(S)}$	Output Data Hold after Enable Deassertion	2		ns
	Timer1 Input Clock (CLKIO pin)			
t_{1CLK}	Input Clock Rate (max. 48 MHz)	20.83		ns

Notes:

2. Per Table 7-6 of revision 1.1 of USB specification, for C_{LOAD} of 50 pF.
3. Timing difference between the differential data signals.
4. Measured at crossover point of differential data signals.
5. BHHI operation is guaranteed => 4.75V
6. Read from 1K FIFO can be achieved by directly read with a 1K FIFO mapped address or read from register 72-73h.
7. This timing is without any internal arbitration, BHHI is parking on the internal DMA bus. The polarity of RDY can be programmed through bit 4 of register 0x09h; RDY active HIGH is shown in the diagram.
8. This timing is with internal arbitration (SIE, SPI/UART, Port7x). The polarity of RDY can be programmed through bit 4 of register 0x09h; RDY active HIGH is shown in the diagram.
9. This timing is without any internal arbitration, BHHI wins the internal Microcontroller Bus immediately. The polarity of RDY can be programmed through bit 4 of register 0x09h; RDY active HIGH is shown in the diagram.
10. This timing is with internal arbitration (SIE, Internal Microcontroller). The polarity of RDY can be programmed through bit 4 of register 0x09h; RDY active HIGH is shown in the diagram.
11. Write to 1K FIFO can be achieved by directly write with a 1K FIFO mapped address or write to register 72-73h.
12. This timing is without any internal arbitration, BHHI is parking on the internal DMA bus. RDY active HIGH is shown in the diagram.
13. This timing is with internal arbitration (SIE, SPI/UART, Port7x). RDY active HIGH IS shown in the diagram.
14. With 25 pF on all SPI pins
15. Frequency programmed in Timer1 or 2 divided by 2 is the SPI clock frequency
16. Time to data active from high-impedance state
17. Hold time to high-impedance state

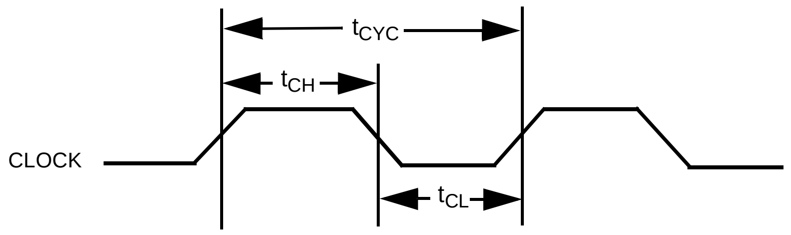


Figure 23-1. Clock Timing

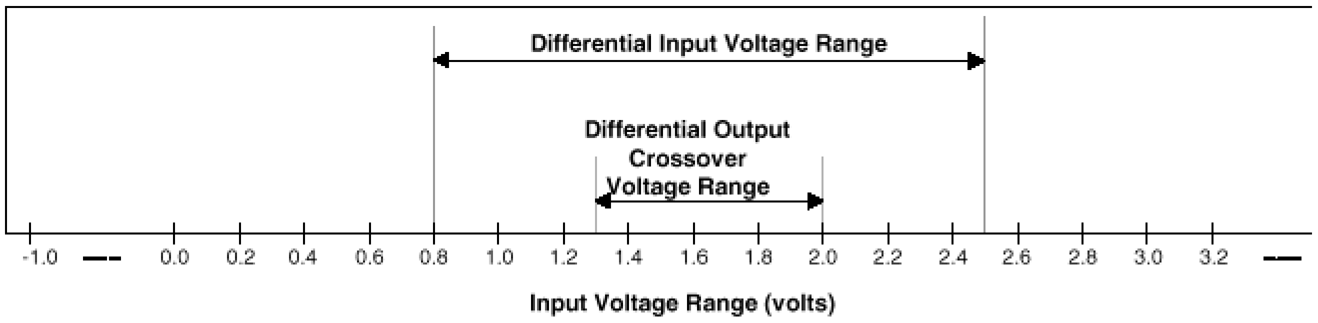

Differential Input Voltage Range
Differential Output Crossover Voltage Range
-1.0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2
2.4
2.6
2.8
3.0
3.2
Input Voltage Range (volts)

Figure 23-2. Differential Input Sensitivity Over Entire Common Mode Range (USB)

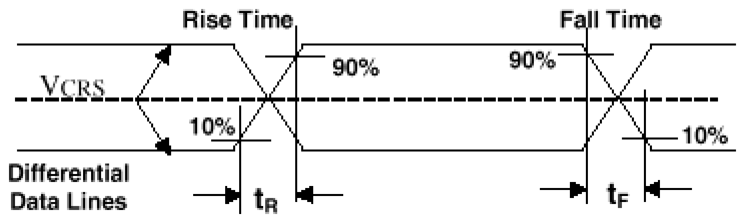


Figure 23-3. USB Data Signal Rise and Fall Time

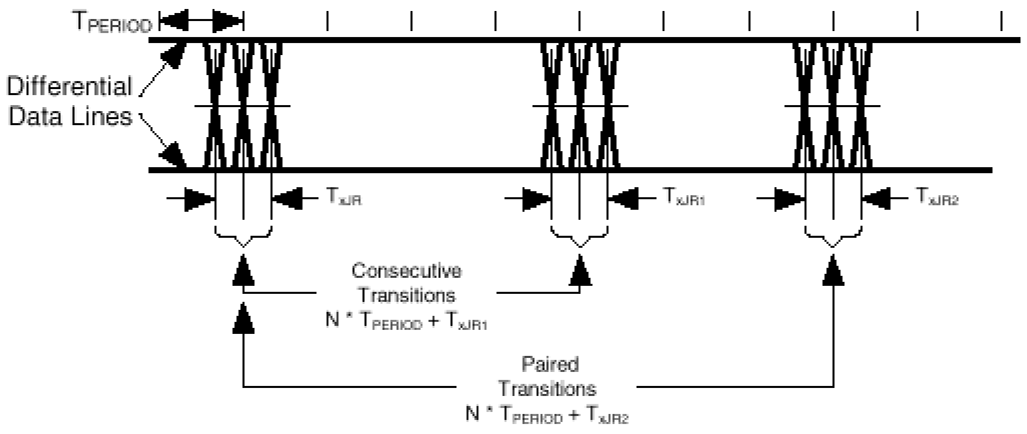

T_PERIOD
Differential Data Lines
T_xJR
T_xJR1
T_xJR2
Consecutive Transitions N * T_PERIOD + T_xJR1
Paired Transitions N * T_PERIOD + T_xJR2

Figure 23-4. USB Receiver Jitter Tolerance

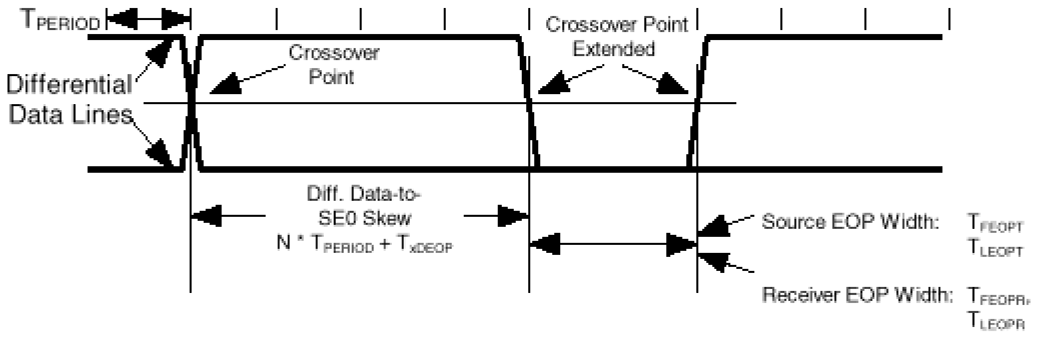

T_PERIOD
Differential Data Lines
Crossover Point
Crossover Point Extended
Diff. Data-to-SE0 Skew N * T_PERIOD + T_xDEOP
Source EOP Width: T_FEOPT T_LEOPT
Receiver EOP Width: T_FEOPR, T_LEOPR

Figure 23-5. Differential to EOP Transition Skew and EOP Width (USB)

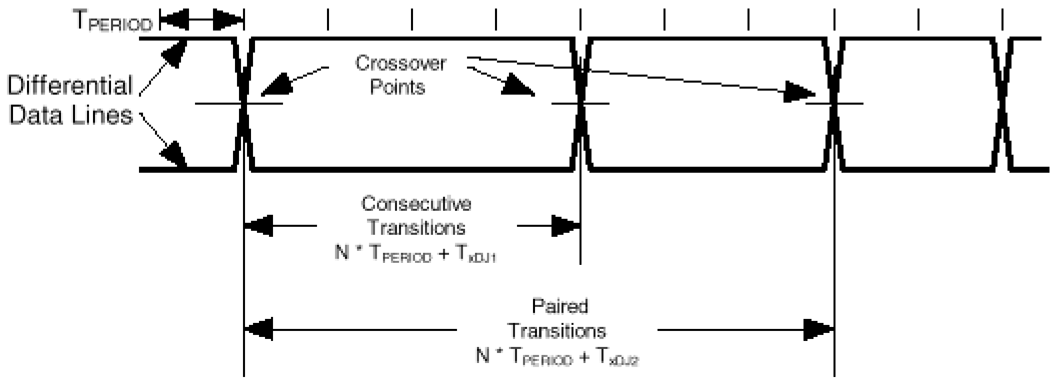

T_PERIOD
Differential Data Lines
Crossover Points
Consecutive Transitions N * T_PERIOD + T_xDJ1
Paired Transitions N * T_PERIOD + T_xDJ2

Figure 23-6. Differential Data Jitter

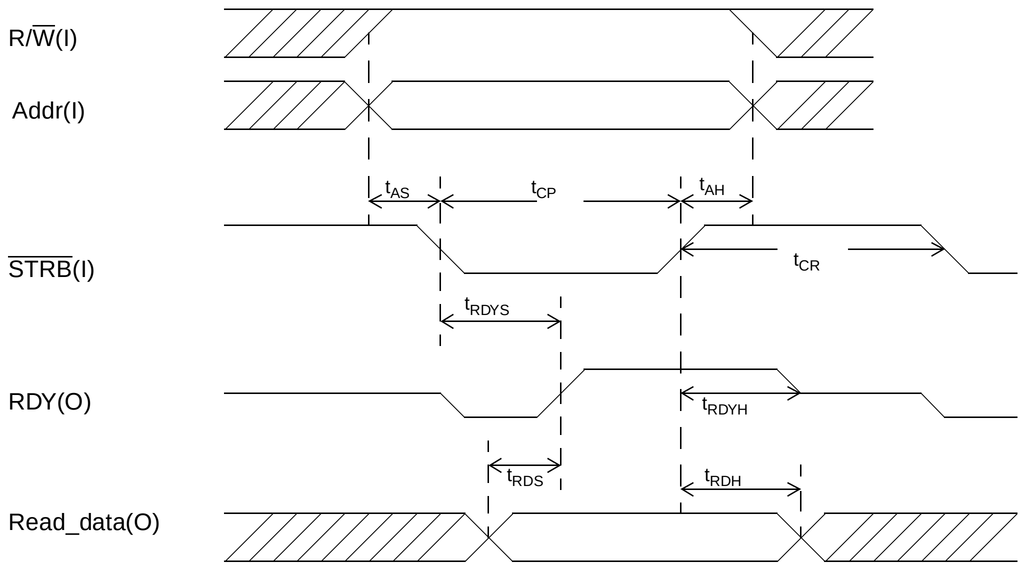


Figure 23-7. BHHI Timing Diagram (Pulling Mode, Read)

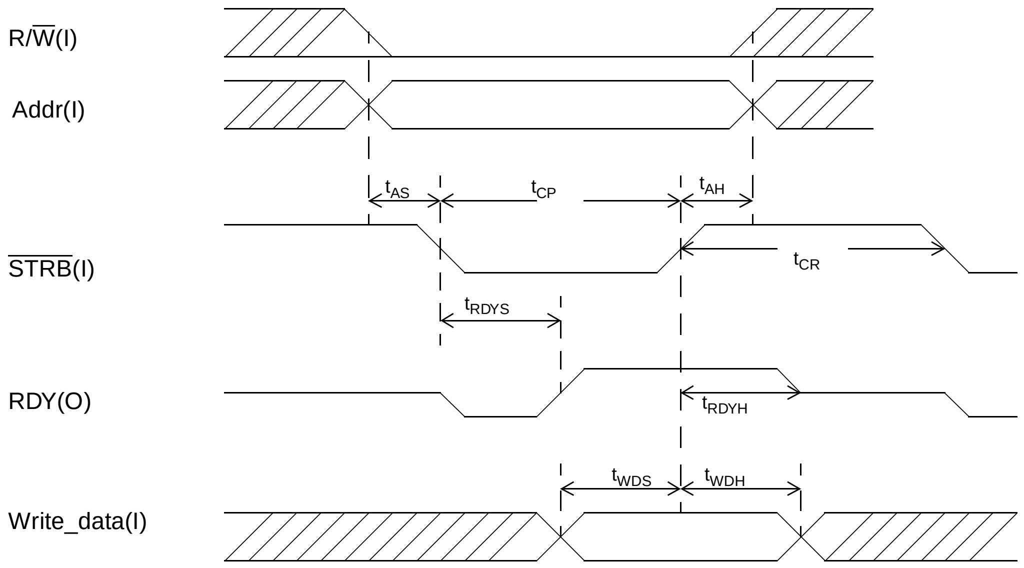


Figure 23-8. BHHI Timing Diagram (Pulling Mode, Write)

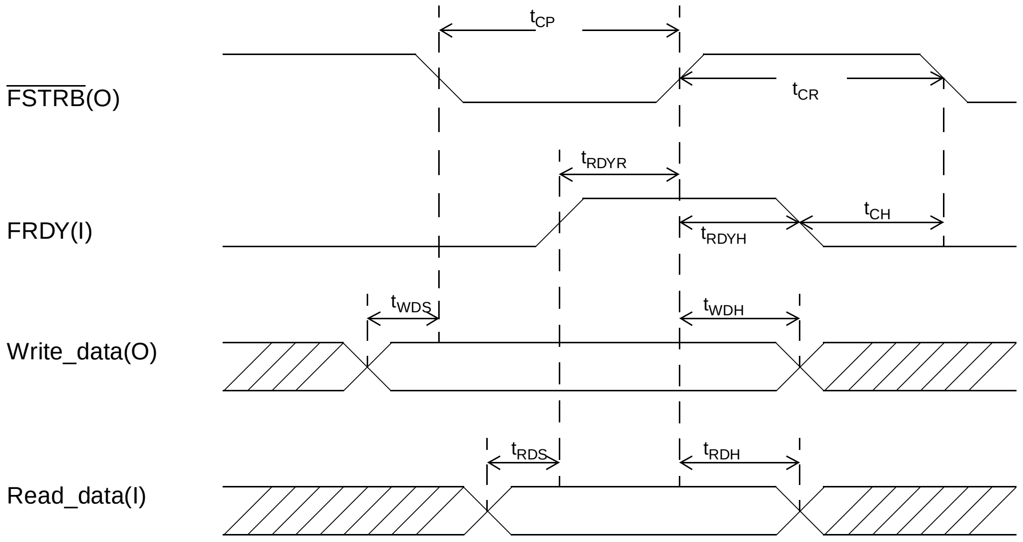


Figure 23-9. BHHI Timing Diagram (Pushing Mode, Forwarding Operation, Write/Read)

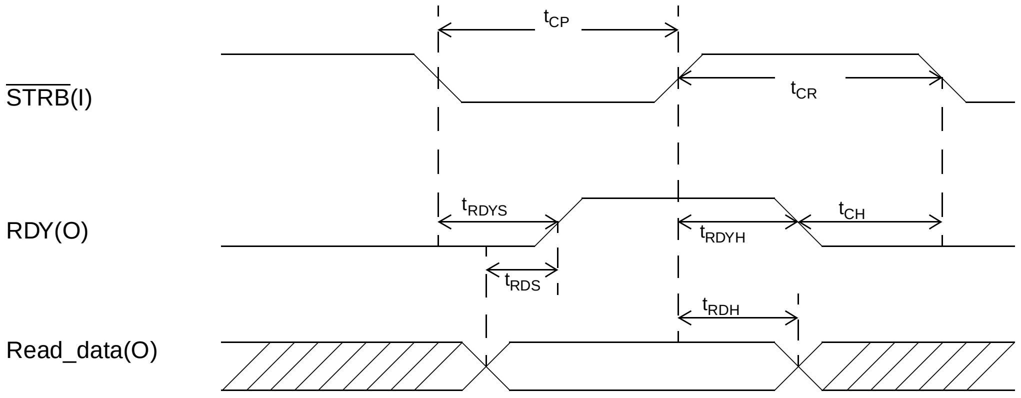


Figure 23-10. BHHI Timing Diagram (Pushing Mode, Reversing Operation, Read from CY7C64213/313)

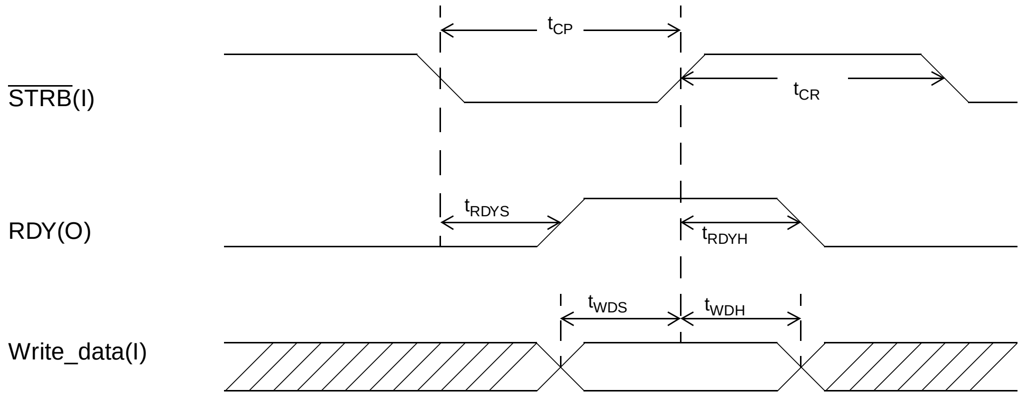


Figure 23-11. BHHI Timing Diagram (Pushing Mode, Reversing Operation, Write to CY7C64213/313)

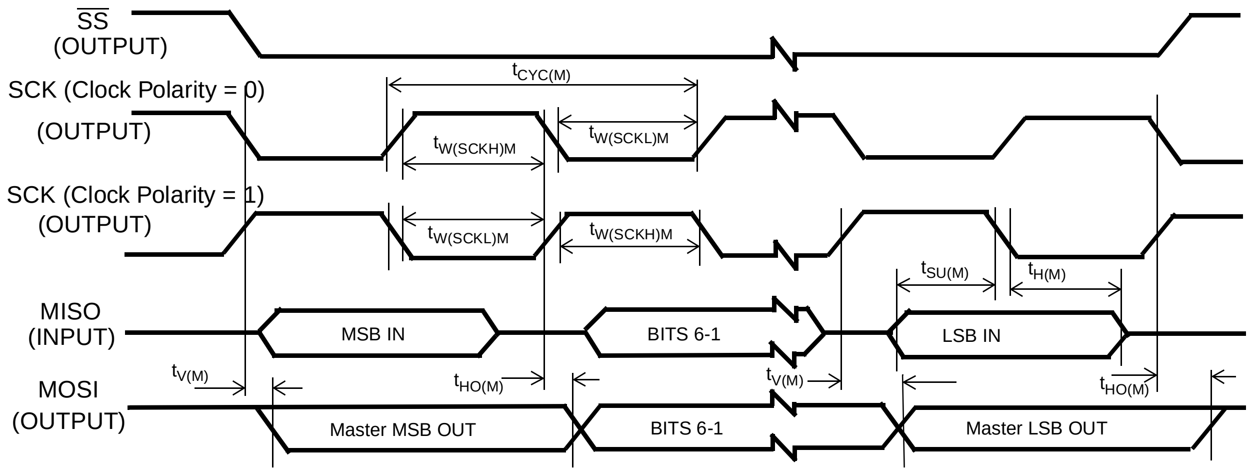


Figure 23-12. SPI Master Timing (Clock Phase = 0)

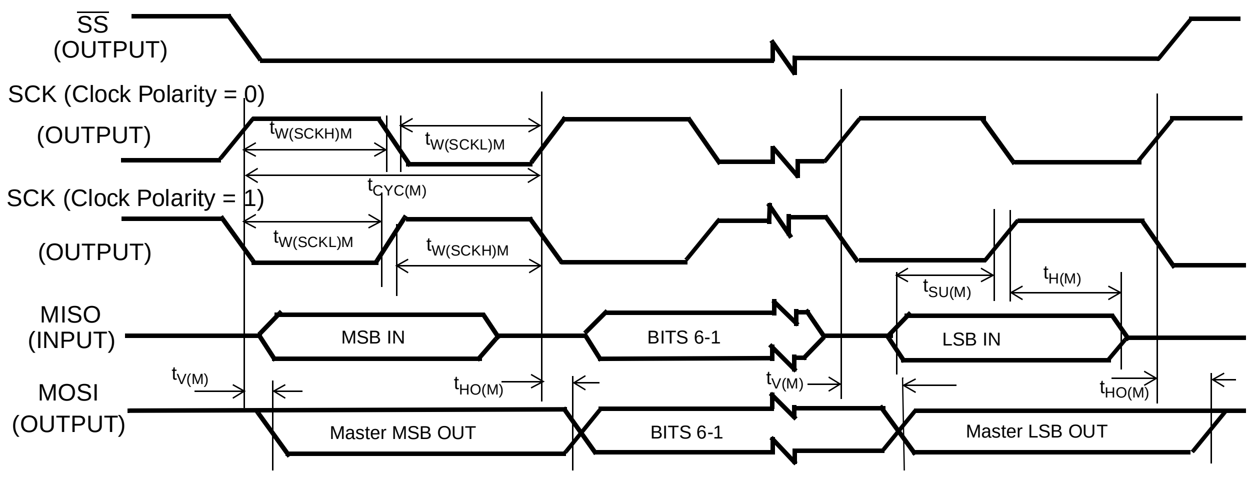
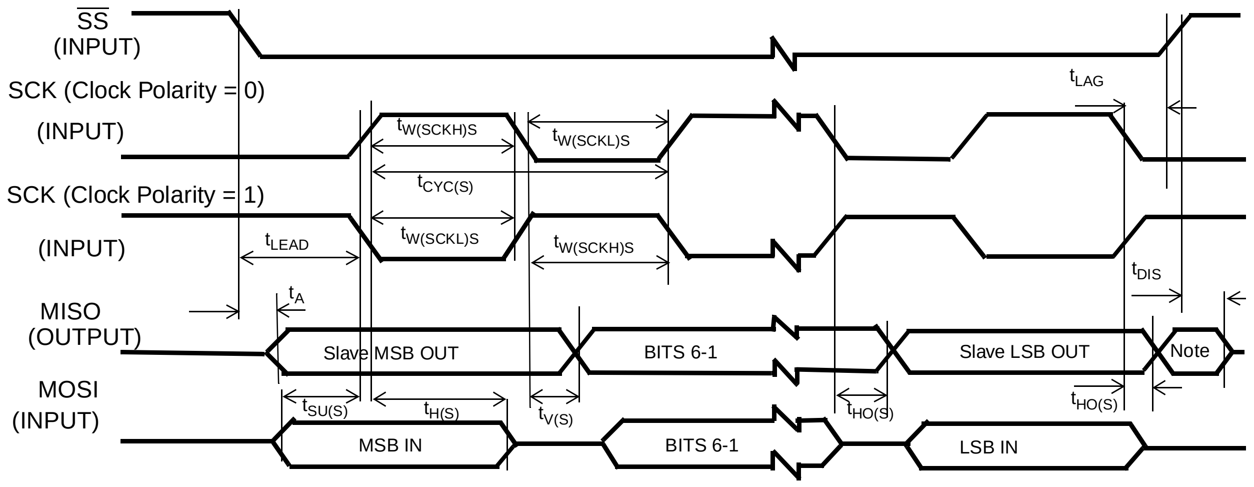
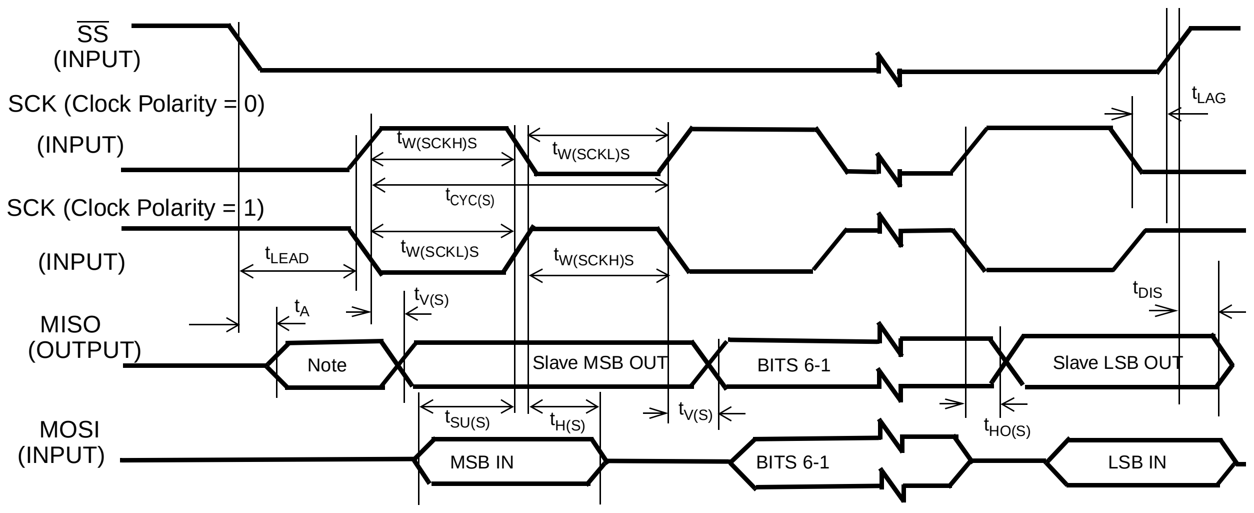


Figure 23-13. SPI Master Timing (Clock Phase = 1)



Note. Not defined but normally MSB of character just received

Figure 23-14. SPI Slave Timing (Clock Phase = 0)



Note. Not defined but normally LSB of character previously transmitted

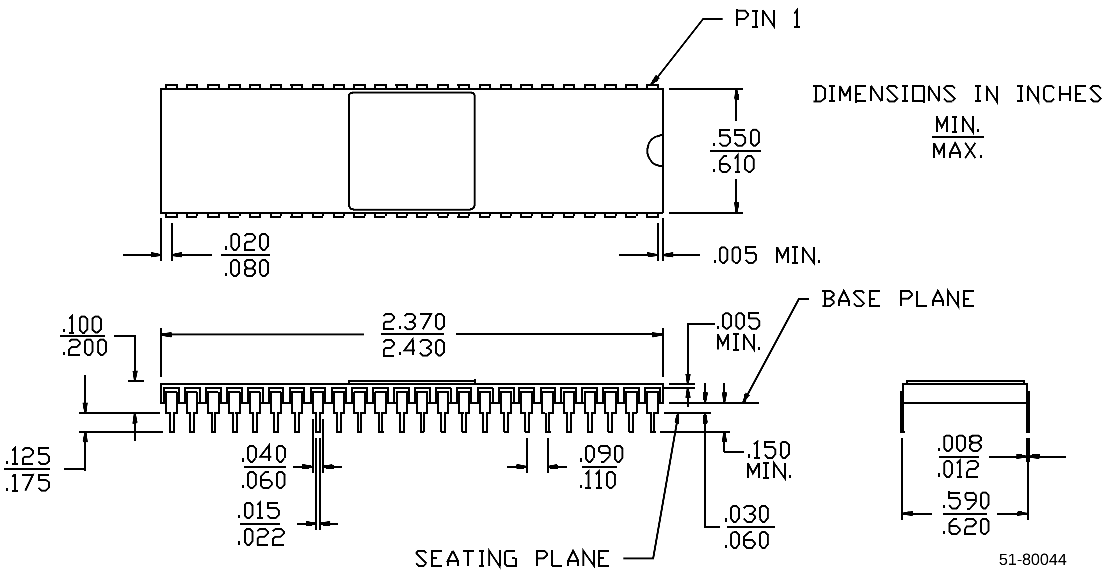
Figure 23-15. SPI Slave Timing (Clock Phase = 1)

24.0 Ordering Information

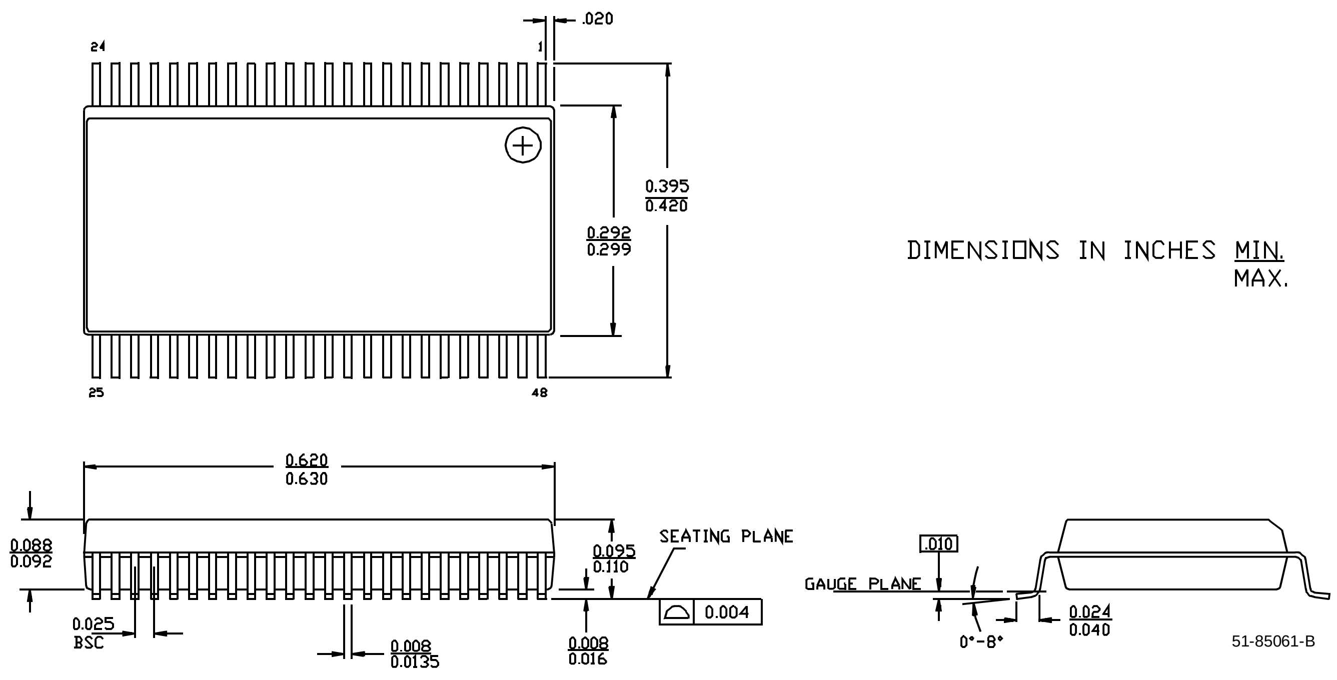
Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C64213-SC	8 KB	S21	28-Pin (300-Mil) SOIC	Commercial
CY7C64213-WC	8 KB	W22	28-Pin Windowed CerDIP	Commercial
CY7C64313-PVC	8 KB	O48	48-Pin (300-Mil) SSOP	Commercial
CY7C64313-WVC	8 KB	D26	48-Pin Windowed SideBraze	Commercial

25.0 Package Diagrams

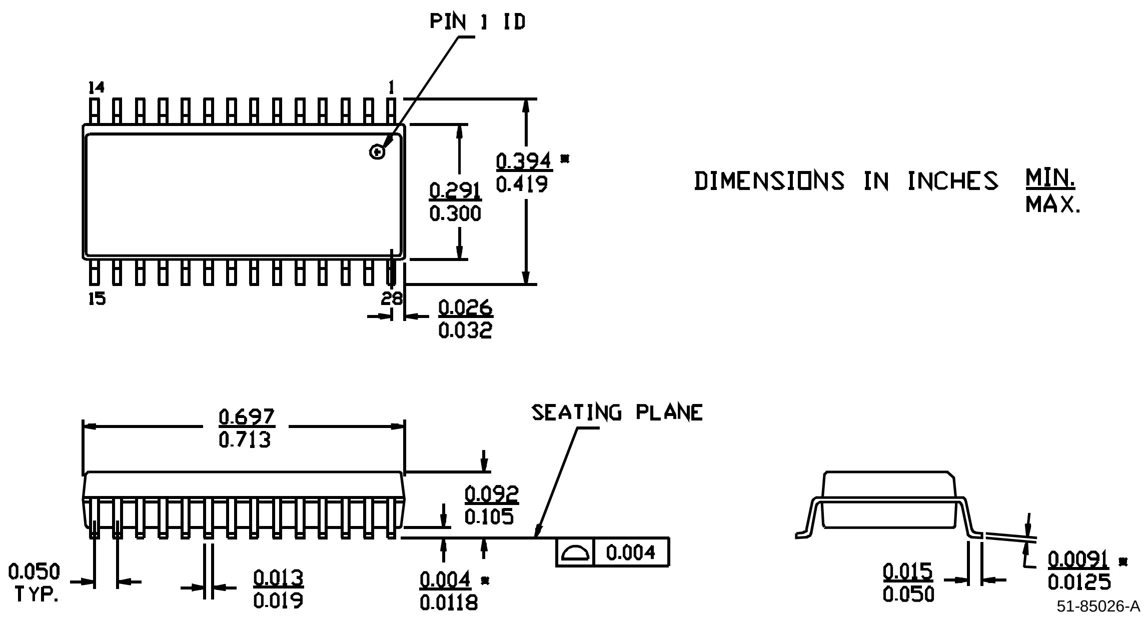
48-Lead (600-Mil) Sidebraze DIP D26
MIL-STD-1835 D-14 Config. C



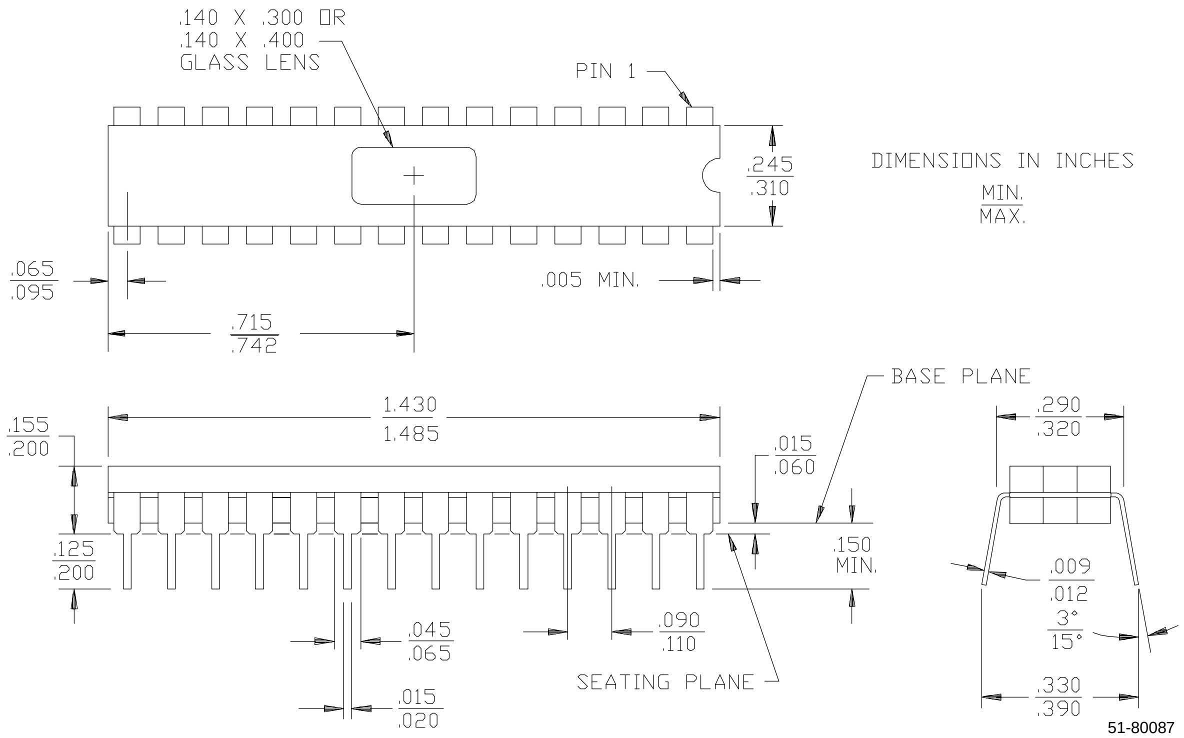
48-Lead Shrunk Small Outline Package O48



28-Lead (300-Mil) Molded SOIC S21



28-Lead (300-Mil) Windowed CerDIP W22
MIL-STD-1835 D-15 Config. A


.140 X .300 OR
.140 X .400
GLASS LENS
PIN 1
.245
.310
DIMENSIONS IN INCHES
MIN.
MAX.
.065
.095
.005 MIN.
.715
.742
BASE PLANE
1.430
1.485
.155
.200
.015
.060
.290
.320
.125
.200
.150
MIN.
.009
.012
3°
15°
.045
.065
.090
.110
SEATING PLANE
.015
.020
.330
.390
51-80087