



CYPRESS

## Low-Speed High I/O, 1.5-Mbps USB Controller

CY7C63413  
CY7C63513

### 1.0 Features

- **Low-cost solution for low-speed applications with high I/O requirements such as keyboards, keyboards with integrated pointing device, gamepads, and many others**
- **USB Specification Compliance**
  - Conforms to USB Specification, Versions 1.1 and 2.0
  - Conforms to USB HID Specification, Version 1.1
  - Supports 1 device address and 3 data endpoints
  - Integrated USB transceiver
- **8-bit RISC microcontroller**
  - Harvard architecture
  - 6-MHz external ceramic resonator
  - 12-MHz internal CPU clock
- **Internal memory**
  - 256 bytes of RAM
  - 8 Kbytes of EPROM (CY7C63413, CY7C63513)
- **Interface can auto-configure to operate as PS2 or USB**
- **I/O port**
  - The CY7C63413/513 have 24 General Purpose I/O (GPIO) pins (Port 0 to 2) capable of sinking 7 mA per pin (typical)
  - The CY7C63413/513 have eight GPIO pins (Port 3) capable of sinking 12 mA per pin (typical) which can drive LEDs
  - Higher current drive is available 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
  - The CY7C63513 has an additional eight I/O pins on a DAC port which has programmable current sink outputs
  - Maskable interrupts on all I/O pins
- **12-bit free-running timer with one microsecond clock ticks**
- **Watch Dog Timer (WDT)**
- **Internal Power-On Reset (POR)**
- **Improved output drivers to reduce EMI**
- **Operating voltage from 4.0V to 5.5V DC**
- **Operating temperature from 0 to 70 degrees Celsius**
- **CY7C63413 available in 40-pin PDIP, 48-pin SSOP for production**
- **CY7C63513 available in 48-pin SSOP packages for production**
- **Industry-standard programmer support**

### 2.0 Functional Overview

The CY7C63413/513 are 8-bit RISC One Time Programmable (OTP) microcontrollers. 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 CY7C63413/513 features 32 General-Purpose I/O (GPIO) pins to support USB and other applications. The I/O pins are grouped into four ports (Port 0 to 3) where each port can be configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs. The CY7C63413/513 have 24 GPIO pins (Ports 0 to 2) that are rated at 7 mA typical sink current. The CY7C63413/513 has 8 GPIO pins (Port 3) that are rated at 12 mA typical sink current, which allows these pins to drive LEDs. Multiple GPIO pins can be connected together to drive a single output for more drive current capacity. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. Note the GPIO interrupts all share the same "GPIO" interrupt vector.

The CY7C63513 features an additional 8 I/O pins in the DAC port. Every DAC pin includes an integrated 14-Kohm pull-up resistor. When a "1" is written to a DAC I/O pin, the output current sink is disabled and the output pin is driven high by the internal pull-up resistor. When a "0" is written to a DAC I/O pin, the internal pull-up is disabled and the output pin provides the programmed amount of sink current. A DAC I/O pin can be used as an input with an internal pull-up by writing a "1" to the pin.

The sink current for each DAC I/O pin can be individually programmed to one of sixteen values using dedicated Isink registers. DAC bits [1:0] can be used as high current outputs with a programmable sink current range of 3.2 to 16 mA (typical). DAC bits [7:2] have a programmable current sink range of 0.2 to 1.0 mA (typical). Again, multiple DAC pins can be connected together to drive a single output that requires more sink current capacity. Each I/O pin can be used to generate a DAC interrupt to the microcontroller and the interrupt polarity for each DAC I/O pin is individually programmable. The DAC port interrupts share a separate "DAC" interrupt vector.

The Cypress microcontrollers use an external 6-MHz ceramic resonator to provide a reference to an internal clock generator. This clock generator reduces the clock-related noise emissions (EMI). The clock generator provides the 6- and 12-MHz clocks that remain internal to the microcontroller.

The CY7C63413/513 are offered with single EPROM options. The CY7C63413 and CY7C63513 have 8 Kbytes of EPROM.



These parts include Power-on Reset logic, a Watch Dog Timer, a vectored interrupt controller, and a 12-bit free-running 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. The Watch Dog Timer can be used to ensure the firmware never gets 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 Watch Dog Timer periodically. If the Watch Dog Timer is not cleared for approximately 8 ms, the microcontroller will generate a hardware watch dog reset.

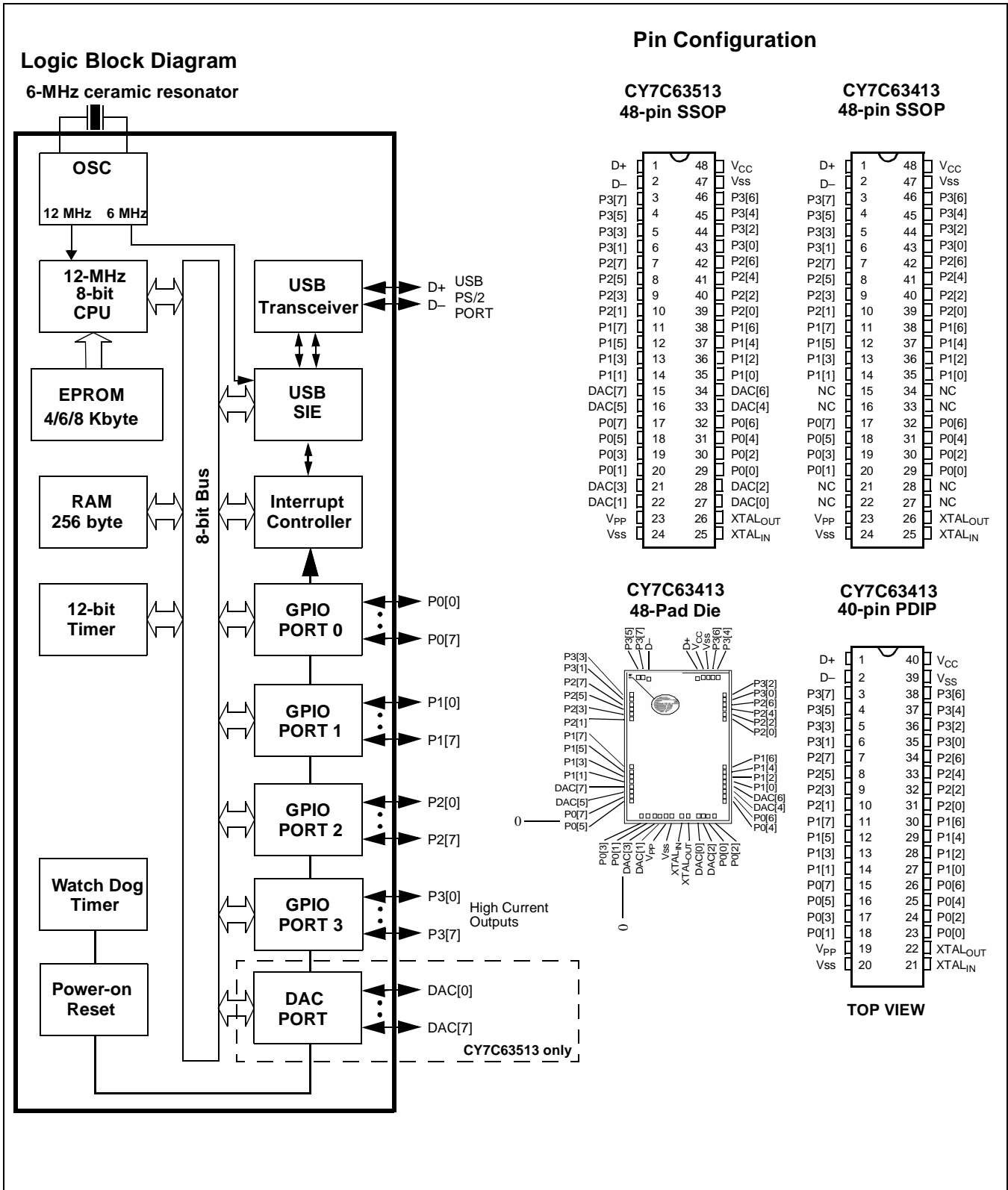
The microcontroller supports eight maskable interrupts in the vectored interrupt controller. Interrupt sources include the USB Bus-Reset, the 128- $\mu$ s and 1.024-ms outputs from the free-running timer, three USB endpoints, the DAC port, and the GPIO ports. The timer bits cause an interrupt (if enabled) when the bit toggles from LOW "0" to HIGH "1." The USB endpoints interrupt after either the USB host or the USB controller sends a packet to the USB. The DAC ports have an additional level of masking that allows the user to select which DAC inputs can cause a DAC interrupt. The GPIO ports also have a level of masking to select which GPIO inputs can cause a GPIO interrupt. For additional flexibility, the input transition polarity that causes an interrupt is programmable for each pin of the DAC port. Input transition polarity can be programmed for each

GPIO port as part of the port configuration. The interrupt polarity can be either rising edge ("0" to "1") or falling edge ("1" to "0").

The free-running 12-bit timer clocked at 1 MHz provides two interrupt sources as noted above (128- $\mu$ s and 1.024-ms). The timer can be used to measure the duration of an event under firmware control by reading the timer twice: once at the start of the event, and once after the event is complete. The difference between the two readings indicates the duration of the event measured in microseconds. The upper four bits of the timer are latched into an internal register when the firmware reads the lower eight bits. A read from the upper four 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 four bits happened to increment right after the lower 8 bits are read.

The CY7C63413/513 include an integrated USB serial interface engine (SIE) that supports the integrated peripherals. The hardware supports one USB device address with three endpoints. The SIE allows the USB host to communicate with the function integrated into the microcontroller.

Finally, the CY7C63413/513 support PS/2 operation. With appropriate firmware the D+ and D- USB pins can also be used as PS/2 clock and data signals. Products utilizing these devices can be used for USB and/or PS/2 operation with appropriate firmware.



### 3.0 Pin Definitions

Name	I/O	CY7C63413		Die	CY7C63513	Description
		40-Pin	48-Pin		48-Pin	
D+, D-	I/O	1,2	1,2	1,2	1,2	USB differential data; PS/2 clock and data signals
P0[7:0]	I/O	15,26,16, 25,17,24, 18,23	17,32,18, 31,19,30, 20,29	17,32,18, 31,19,30, 20,29	17,32,18, 31,19,30, 20,29	GPIO port 0 capable of sinking 7 mA (typical)
P1[3:0]	I/O	11,30,12, 29,13,28, 14,27	11,38,12, 37,13,36, 14,35	11,38,12, 37,13,36, 14,35	11,38,12, 37,13,36, 14,35	GPIO Port 1 capable of sinking 7 mA (typical).
P2	I/O	7,34,8, 33,9,32, 10,31	7,42,8, 41,9,40, 10,39	7,42,8, 41,9,40, 10,39	7,42,8, 41,9,40, 10,39	GPIO Port 2 capable of sinking 7 mA (typical).
P3[7:4]	I/O	3,38,4, 37,5,36, 6,35	3,46,4, 45,5,44, 6,43	3,46,4,45, 5,44,6,43	3,46,4, 45,5,44, 6,43	GPIO Port 3 capable of sinking 12 mA (typical).
DAC	I/O	n/a	n/a	15,34,16, 33,21,28, 22,27	15,34,16, 33,21,28, 22,27	DAC I/O Port with programmable current sink outputs. DAC[1:0] offer a programmable range of 3.2 to 16 mA typical. DAC[7:2] have a programmable sink current range of 0.2 to 1.0 mA typical.
XTAL <sub>IN</sub>	IN	21	25	25	25	6-MHz ceramic resonator or external clock input
XTAL <sub>OUT</sub>	OUT	22	26	26	26	6-MHz ceramic resonator
V <sub>PP</sub>		19	23	23	23	Programming voltage supply, ground during operation
V <sub>CC</sub>		40	48	48	48	Voltage supply
V <sub>SS</sub>		20,39	24,47	24,47	24,47	Ground

### 4.0 Programming Model

#### 4.1 14-bit Program Counter (PC)

The 14-bit Program Counter (PC) allows access for up to 8 kilobytes of EPROM using the CY7C63413/513 architecture. The program counter is cleared during reset, such that the first instruction executed after a reset is at address 0x0000. This is typically a jump instruction to a reset handler that initializes the application.

The lower eight bits of the program counter are incremented as instructions are loaded and executed. The upper six bits of the program counter are incremented by executing an XPAGE instruction. As a result, the last instruction executed within a 256-byte "page" of sequential code should be an XPAGE instruction. The assembler directive "XPAGEON" 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 only during a RETI 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.

#### 4.2 8-bit Accumulator (A)

The accumulator is the general purpose, do everything register in the architecture where results are usually calculated.

#### 4.3 8-bit Index Register (X)

The index register "X" is available to the firmware as an auxiliary accumulator. The X register also allows the processor to perform indexed operations by loading an index value into X.

#### 4.4 8-bit Program Stack Pointer (PSP)

During a reset, the Program Stack Pointer (PSP) is set to zero. This means the program "stack" starts at RAM address 0x00 and "grows" upward from there. Note the program stack pointer is directly addressable under firmware control, using the MOV PSP,A instruction. The PSP supports interrupt service under hardware control and CALL, RET, and RETI instructions under firmware control.

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

The Return From Interrupt (RETI) instruction decrements the program stack pointer, then restores the second byte from memory addressed by the PSP. The program stack pointer is

decremented again and the first byte is restored from memory addressed by the PSP. After the program counter and flags have been restored from stack, the interrupts are enabled. The effect is to restore the program counter and flags from the program stack, decrement the program stack pointer by two, and re-enable interrupts.

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

The Return From Subroutine (RET) instruction restores the program counter, but not the flags, from program stack and decrements the PSP by two.

#### 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 Data Stack Pointer will be set to zero. A PUSH instruction when DSP equal zero will write data at the top of the data RAM (address 0xFF). This would write data to the memory area reserved for a FIFO for USB endpoint 0. In non-USB applications, this works fine and is not a problem. For USB applications, it is strongly recommended that the DSP is loaded after reset just below the USB DMA buffers.

#### 4.6 Address Modes

The CY7C63413/513 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 0xE8:

- MOV A,0E8h

This instruction will require two bytes of code where the first byte identifies the "MOV A" instruction with a data operand as the second byte. The second byte of the instruction will be the constant "0xE8". 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 0E8h
- 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 0x10:

- MOV A,[10h]

In normal usage, variable names are assigned to variable addresses using "EQU" statements to improve the readability of the assembler source code. As an example, the following code is equivalent to the example shown above:

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

##### 4.6.3 Indexed

"Indexed" address mode allows the firmware to manipulate arrays of data stored in SRAM. The address of the data operand is the sum of a constant encoded in the instruction and the contents of the "X" register. In normal usage, the constant will be the "base" address of an array of data and the X register will contain an index that indicates which element of the array is actually addressed:

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

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

**5.0 Instruction Set Summary**

MNEMONIC	operand	opcode	cycles	MNEMONIC	operand	opcode	cycles
HALT		00	7	NOP		20	4
ADD A,expr	data	01	4	INC A	acc	21	4
ADD A,[expr]	direct	02	6	INC X	x	22	4
ADD A,[X+expr]	index	03	7	INC [expr]	direct	23	7
ADC A,expr	data	04	4	INC [X+expr]	index	24	8
ADC A,[expr]	direct	05	6	DEC A	acc	25	4
ADC A,[X+expr]	index	06	7	DEC X	x	26	4
SUB A,expr	data	07	4	DEC [expr]	direct	27	7
SUB A,[expr]	direct	08	6	DEC [X+expr]	index	28	8
SUB A,[X+expr]	index	09	7	IORD expr	address	29	5
SBB A,expr	data	0A	4	IOWR expr	address	2A	5
SBB A,[expr]	direct	0B	6	POP A		2B	4
SBB A,[X+expr]	index	0C	7	POP X		2C	4
OR A,expr	data	0D	4	PUSH A		2D	5
OR A,[expr]	direct	0E	6	PUSH X		2E	5
OR A,[X+expr]	index	0F	7	SWAP A,X		2F	5
AND A,expr	data	10	4	SWAP A,DSP		30	5
AND A,[expr]	direct	11	6	MOV [expr],A	direct	31	5
AND A,[X+expr]	index	12	7	MOV [X+expr],A	index	32	6
XOR A,expr	data	13	4	OR [expr],A	direct	33	7
XOR A,[expr]	direct	14	6	OR [X+expr],A	index	34	8
XOR A,[X+expr]	index	15	7	AND [expr],A	direct	35	7
CMP A,expr	data	16	5	AND [X+expr],A	index	36	8
CMP A,[expr]	direct	17	7	XOR [expr],A	direct	37	7
CMP A,[X+expr]	index	18	8	XOR [X+expr],A	index	38	8
MOV A,expr	data	19	4	IOWX [X+expr]	index	39	6
MOV A,[expr]	direct	1A	5	CPL		3A	4
MOV A,[X+expr]	index	1B	6	ASL		3B	4
MOV X,expr	data	1C	4	ASR		3C	4
MOV X,[expr]	direct	1D	5	RLC		3D	4
<i>reserved</i>		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
JNZ	addr	B0-BF	5	INDEX	addr	F0-FF	14

## 6.0 Memory Organization

### 6.1 Program Memory Organization

Address	Description
0x0000	Program execution begins here after a reset
0x0002	USB Bus Reset interrupt vector
0x0004	128- $\mu$ s timer interrupt vector
0x0006	1.024-ms timer interrupt vector
0x0008	USB address A endpoint 0 interrupt vector
0x000A	USB address A endpoint 1 interrupt vector
0x000C	USB address A endpoint 2 interrupt vector
0x000E	Reserved
0x0010	Reserved
0x0012	Reserved
0x0014	DAC interrupt vector
0x0016	GPIO interrupt vector
0x0018	Reserved
0x001A	<b>Program Memory begins here</b>
0x1FDF	<b>8-KB PROM ends here (CY7C63413, CY7C63513)</b>

after reset  
 14-bit PC  $\rightarrow$

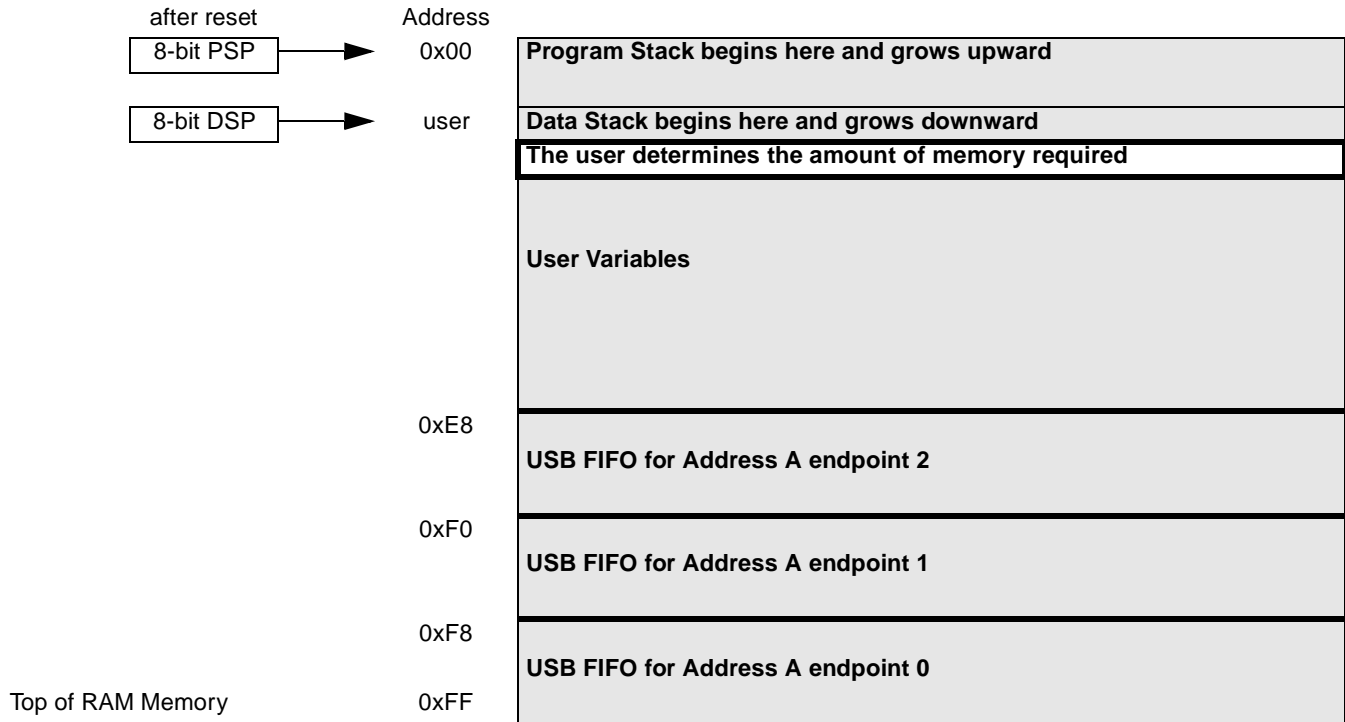
(8K - 32 bytes)

**Figure 6-1. Program Memory Space with Interrupt Vector Table**

## 6.2 Data Memory Organization

The CY7C63413/513 microcontrollers provide 256 bytes of data RAM. In normal usage, the SRAM is partitioned into four

areas: program stack, data stack, user variables and USB endpoint FIFOs as shown below:



### 6.3 I/O Register Summary

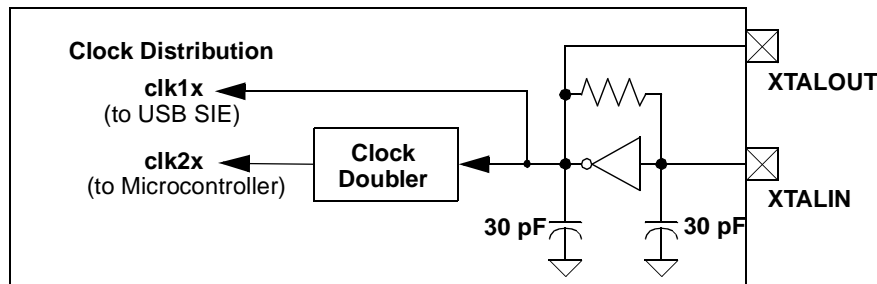
I/O registers are accessed via the I/O Read (IORD) and I/O Write (IOWR, IOWX) instructions. IORD reads the selected port into the accumulator. IOWR writes data from the accumu-

lator 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 6-1. I/O Register Summary**

Register Name	I/O Address	Read/Write	Function
Port 0 Data	0x00	R/W	GPIO Port 0
Port 1 Data	0x01	R/W	GPIO Port 1
Port 2 Data	0x02	R/W	GPIO Port 2
Port 3 Data	0x03	R/W	GPIO 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	GPIO Ports Configurations
USB Device Address A	0x10	R/W	USB Device Address A
EP A0 Counter Register	0x11	R/W	USB Address A, Endpoint 0 counter register
EP A0 Mode Register	0x12	R/W	USB Address A, Endpoint 0 configuration register
EP A1 Counter Register	0x13	R/W	USB Address A, Endpoint 1 counter register
EP A1 Mode Register	0x14	R/C	USB Address A, Endpoint 1 configuration register
EP A2 Counter Register	0x15	R/W	USB Address A, Endpoint 2 counter register
EP A2 Mode Register	0x16	R/C	USB Address A, Endpoint 2 configuration register
USB Status & Control	0x1F	R/W	USB upstream port traffic status and control register
Global Interrupt Enable	0x20	R/W	Global interrupt enable register
Endpoint Interrupt Enable	0x21	R/W	USB endpoint interrupt enables
Timer (LSB)	0x24	R	Lower eight bits of free-running timer (1 MHz)
Timer (MSB)	0x25	R	Upper four bits of free-running timer that are latched when the lower eight bits are read.
WDR Clear	0x26	W	Watch Dog Reset clear
DAC Data	0x30	R/W	DAC I/O
DAC Interrupt Enable	0x31	W	Interrupt enable for each DAC pin
DAC Interrupt Polarity	0x32	W	Interrupt polarity for each DAC pin
DAC Isink	0x38-0x3F	W	One four bit sink current register for each DAC pin
Processor Status & Control	0xFF	R/W	Microprocessor status and control

## 7.0 Clocking



**Figure 7-1. Clock Oscillator On-chip Circuit**

The XTAL<sub>IN</sub> and XTAL<sub>OUT</sub> are the clock pins to the microcontroller. The user can connect a low-cost ceramic resonator or an external oscillator can be connected to these pins to provide a reference frequency for the internal clock distribution and clock doubler.

An external 6 MHz clock can be applied to the XTAL<sub>IN</sub> pin if the XTAL<sub>OUT</sub> pin is left open. Please note that grounding the XTAL<sub>OUT</sub> pin is not permissible as the internal clock is effectively shorted to ground.

### 8.0 Reset

The USB Controller supports three types of resets. All registers are restored to their default states during a reset. The USB Device Addresses are set to 0 and all interrupts are disabled. In addition, the Program Stack Pointer (PSP) and Data Stack Pointer (DSP) are set to 0x00. For USB applications, the firmware should set the DSP below 0xE8 to avoid a memory conflict with RAM dedicated to USB FIFOs. The assembly instructions to do this are shown below:

```
Mov A, E8h ; Move 0xE8 hex into Accumulator
Swap A,dsp ; Swap accumulator value into dsp register
```

The three reset types are:

1. Power-On Reset (POR)
2. Watch Dog Reset (WDR)
3. USB Bus Reset (non hardware reset)

The occurrence of a reset is recorded in the Processor Status and Control Register located at I/O address 0xFF. Bits 4, 5, and 6 are used to record the occurrence of POR, USB Reset, and WDR respectively. The firmware can interrogate these bits to determine the cause of a reset.

The microcontroller begins execution from ROM address 0x0000 after a POR or WDR reset. Although this looks like interrupt vector 0, there is an important difference. Reset processing does NOT push the program counter, carry flag, and zero flag onto program stack. That means the reset handler in firmware should initialize the hardware and begin executing

the "main" loop of code. Attempting to execute either a RET or RETI in the reset handler will cause unpredictable execution results.

### 8.1 Power-On Reset (POR)

Power-On Reset (POR) occurs every time the V<sub>CC</sub> voltage to the device ramps from 0V to an internally defined trip voltage (V<sub>rst</sub>) of approximately 1/2 full supply voltage. In addition to the normal reset initialization noted under "Reset," bit 4 (PORS) of the Processor Status and Control Register is set to "1" to indicate to the firmware that a Power-On Reset occurred. The POR event forces the GPIO ports into input mode (high impedance), and the state of Port 3 bit 7 is used to control how the part will respond after the POR releases.

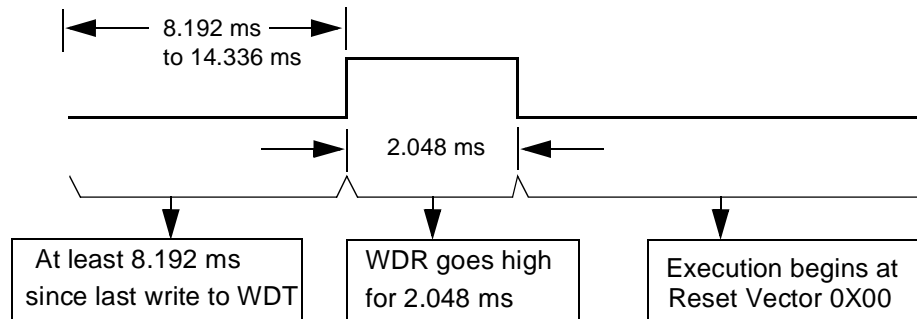
If Port 3 bit 7 is HIGH (pulled to V<sub>CC</sub>) and the USB IO are at the idle state (DM HIGH and DP LOW) the part will go into a semi-permanent power down/suspend mode, waiting for the USB IO to go to one of Bus Reset, K (resume) or SE0. If Port 3 bit 7 is still HIGH when the part comes out of suspend, then a 128- $\mu$ s timer starts, delaying CPU operation until the ceramic resonator has stabilized.

If Port 3 bit 7 was LOW (pulled to V<sub>SS</sub>) the part will start a 96-ms timer, delaying CPU operation until V<sub>CC</sub> has stabilized, then continuing to run as reset.

Firmware should clear the POR Status (PORS) bit in register 0xFF before going into suspend as this status bit selects the 128- $\mu$ s or 96-ms start-up timer value as follows: IF Port 3 bit 7 is HIGH then 128- $\mu$ s is always used; ELSE if PORS is HIGH then 96-ms is used; ELSE 128- $\mu$ s is used.

### 8.2 Watch Dog Reset (WDR)

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



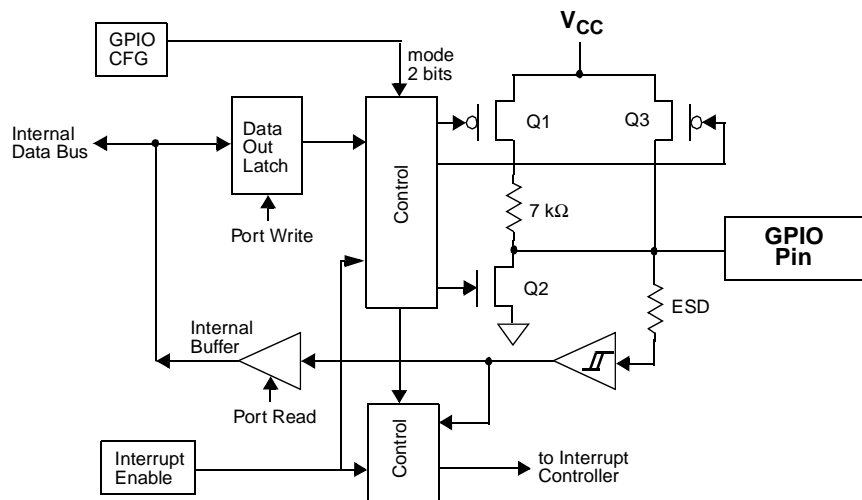
**Figure 8-1. Watch Dog Reset (WDR)**

The Watch Dog Timer is a 2-bit timer clocked by a 4.096-ms clock (bit 11) from the free-running timer. Writing any value to the write-only Watch Dog Clear I/O port (0x26) will clear the Watch Dog Timer.

In some applications, the Watch Dog Timer may be cleared in the 1.024-ms timer interrupt service routine. If the 1.024-ms timer interrupt service routine does not get executed for 8.192 ms or more, a Watch Dog Timer Reset will occur. A Watch Dog

Timer Reset lasts for 2.048 ms after which the microcontroller begins execution at ROM address 0x0000. The USB transmitter is disabled by a Watch Dog Reset because the USB Device Address Register is cleared. Otherwise, the USB Controller would respond to all address 0 transactions. The USB transmitter remains disabled until the MSB of the USB address register is set.

## 9.0 General Purpose I/O Ports



**Figure 9-1. Block Diagram of a GPIO Line**

Ports 0 to 2 provide 24 GPIO pins that can be read or written. Each port (8 bits) can be configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs. Please note an open drain output is also a high-impedance (no pull-up) input. All of the I/O pins within a given port have the same configuration. Ports 0 to 2 are considered low current drive with typical current sink capability of 7 mA.

The internal pull-up resistors are typically 7 kΩ. Two factors govern the enabling and disabling of the internal pull-up resistors: the port configuration selected in the GPIO Configuration

register and the state of the output data bit. If the GPIO Configuration selected is "Resistive" and the output data bit is "1", then the internal pull-up resistor is enabled for that GPIO pin. Otherwise, Q1 is turned off and the 7-kΩ pull-up is disabled. Q2 is "ON" to sink current whenever the output data bit is written as a "0". Q3 provides "HIGH" source current when the GPIO port is configured for CMOS outputs and the output data bit is written as a "1". Q2 and Q3 are sized to sink and source, respectively, roughly the same amount of current to support traditional CMOS outputs with symmetric drive

Addr: 0x00				Port 0 Data			
P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 9-1. Port 0 Data**

Addr: 0x01				Port 1 Data			
P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 9-2. Port 1 Data**

Addr: 0x02				Port 2 Data			
P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 9-3. Port 2 Data**

Addr: 0x03				Port 3 Data			
P3[7]	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 9-4. Port 3 Data**

Addr: 0x30				DAC Port Data			
Low current outputs 0.2 mA to 1.0 mA typical						High current outputs 3.2 mA to 16 mA typical	
DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 9-5. DAC Port Data**

Port 3 has eight GPIO pins. Port 3 (8 bits) can be configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs. An open drain output is also a high-impedance input. Port 3 offers high current drive with a typical current sink capability of 12 mA. The internal pull-up resistors are typically 7 k $\Omega$ .

**Note:** Special care should be exercised with any unused GPIO data bits. An unused GPIO data bit, either a pin on the chip or a port bit that is not bonded on a particular package, must not be left floating when the device enters the suspend state. If a GPIO data bit is left floating, the leakage current caused by the floating bit may violate the suspend current limitation specified by the USB Specification. If a '1' is written to the unused data bit and the port is configured with open drain outputs, the un-

used data bit will be in an indeterminate state. Therefore, if an unused port bit is programmed in open-drain mode, it must be written with a '0.'

During reset, all of the bits in the GPIO to a default configuration of Open Drain output, positive interrupt polarity for all GPIO ports.

### 9.1 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.

Addr: 0x04				Port 0 Interrupt Enable			
P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]
W	W	W	W	W	W	W	W

**Table 9-6. Port 0 Interrupt Enable**

Addr: 0x05				Port 1 Interrupt Enable			
P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]
W	W	W	W	W	W	W	W

**Table 9-7. Port 1 Interrupt Enable**

Addr: 0x06				Port 2 Interrupt Enable			
P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]
W	W	W	W	W	W	W	W

**Table 9-8. Port 2 Interrupt Enable**

Addr: 0x07				Port 3 Interrupt Enable			
P3[7]	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]
W	W	W	W	W	W	W	W

**Table 9-9. Port 3 Interrupt Enable**

## 9.2 GPIO Configuration Port

Every GPIO port can be programmed as inputs with internal pull-ups, open drain outputs, and traditional CMOS outputs. In addition, the interrupt polarity for each port can be programmed. With positive interrupt polarity, a rising edge (“0” to

“1”) on an input pin causes an interrupt. With negative polarity, a falling edge (“1” to “0”) on an input pin causes an interrupt. As shown in the table below, when a GPIO port is configured with CMOS outputs, interrupts from that port are disabled. The GPIO Configuration Port register provides two bits per port to program these features. The possible port configurations are:

Port Configuration bits	Pin Interrupt 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)

In “Resistive” mode, a 7-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 7-kΩ pull-up resistor when a “1” has been written to the pin. Or 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 7-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 port 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.

A port configured in CMOS mode has interrupt generation disabled, yet the interrupt mask bits serve to control port direction.

If a port's associated Interrupt Mask bits are cleared, those port bits are strictly outputs. If the Interrupt Mask bits are set then those bits will be open drain inputs. As open drain inputs, if their data output values are ‘1’ those port pins will be CMOS inputs (HIGH Z output).

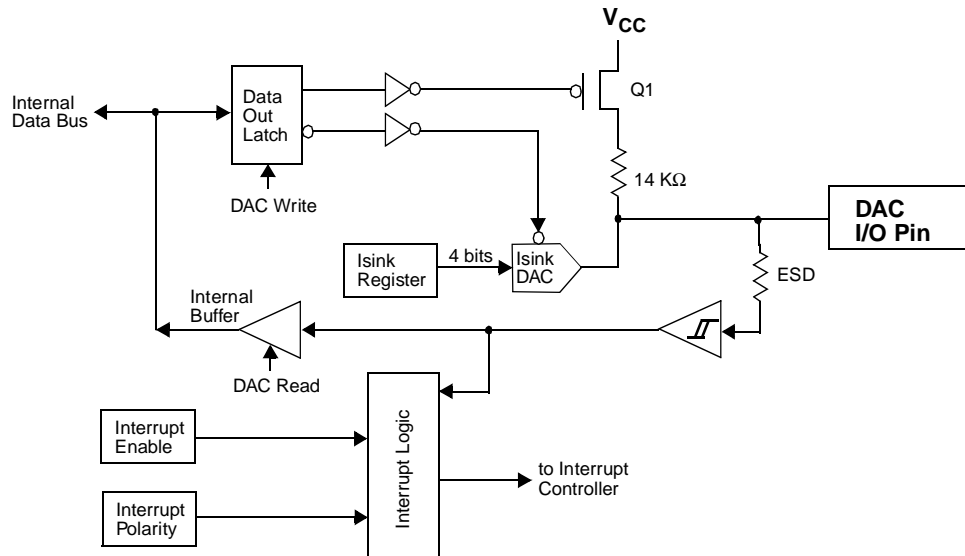
In “Open Drain” mode the internal pull-up resistor and CMOS driver (HIGH) are both disabled. An I/O pin that has been written as a “1” can be used as either a high-impedance input or a three-state output. An I/O pin that has been written as a “0” will drive the output LOW. The interrupt polarity for 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.” This selects the default configuration: Open Drain output, positive interrupt polarity for all GPIO ports.

Addr: 0x08		GPIO Configuration Register					
7	6	5	4	3	2	1	0
Port 3 Config Bit 1	Port 3 Config Bit 0	Port 2 Config Bit 1	Port 2 Config Bit 0	Port 1 Config Bit 1	Port 1 Config Bit 0	Port 0 Config Bit 1	Port 0 Config Bit 0
W	W	W	W	W	W	W	W

**Table 9-10. GPIO Configuration Register**

## 10.0 DAC Port


**Figure 10-1. Block Diagram of DAC Port**

The DAC port provides the CY7C63513 with 8 programmable current sink I/O pins. Writing a “1” to a DAC I/O pin disables the output current sink (Isink DAC) and drives the I/O pin HIGH through an integrated 14 Kohm resistor. When a “0” is written to a DAC I/O pin, the Isink DAC is enabled and the pull-up resistor is disabled. A “0” output will cause the Isink DAC to sink current to drive the output LOW. The amount of sink current for the DAC I/O pin is programmable over 16 values based on the contents of the DAC Isink Register for that output pin. DAC[1:0] are the two high current outputs that are program-

mable from a minimum of 3.2 mA to a maximum of 16 mA (typical). DAC[7:2] are low current outputs that are programmable from a minimum of 0.2 mA to a maximum of 1.0 mA (typical).

When a DAC I/O bit is written as a “1,” the I/O pin is either an output pulled high through the 14 Kohm resistor or an input with an internal 14 Kohm pull-up resistor. All DAC port data bits are set to “1” during reset.

Addr: 0x30		DAC Port Data					
Low current outputs 0.2 mA to 1.0 mA typical						High current outputs 3.2 mA to 16 mA typical	
DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 10-1. DAC Port Data**

### 10.1 DAC Port Interrupts

A DAC port interrupt can be enabled/disabled for each pin individually. The DAC Port Interrupt Enable register provides this feature with an interrupt mask bit for each DAC I/O pin. Writing a “1” to a bit in this register enables interrupts from the corre-

sponding bit position. Writing a “0” to a bit in the DAC Port Interrupt Enable register disables interrupts from the corresponding bit position. All of the DAC Port Interrupt Enable register bits are cleared to “0” during a reset.

Addr: 0x31				DAC Port Interrupt Enable			
DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]
W	W	W	W	W	W	W	W

**Table 10-2. DAC Port Interrupt Enable**

As an additional benefit, the interrupt polarity for each DAC pin is programmable with the DAC Port Interrupt Polarity register. Writing a “0” to a bit selects negative polarity (falling edge) that will cause an interrupt (if enabled) if a falling edge transition occurs on the corresponding input pin. Writing a “1” to a bit in

this register selects positive polarity (rising edge) that will cause an interrupt (if enabled) if a rising edge transition occurs on the corresponding input pin. All of the DAC Port Interrupt Polarity register bits are cleared during a reset.

Addr: 0x32				DAC Port Interrupt Polarity			
DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]
W	W	W	W	W	W	W	W

**Table 10-3. DAC Port Interrupt Polarity**

## 10.2 DAC Isink Registers

Each DAC I/O pin has an associated DAC Isink register to program the output sink current when the output is driven

LOW. The first Isink register (0x38) controls the current for DAC[0], the second (0x39) for DAC[1], and so on until the Isink register at 0x3F controls the current to DAC[7].

Addr: 0x38-0x3F				DAC Port Interrupt Polarity			
Reserved				Isink Value			
				Isink[3]	Isink[2]	Isink[1]	Isink[0]
				W	W	W	W

**Table 10-4. DAC Port Isink**

## 11.0 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 set-up packets
- Fill and empty the FIFOs
- Suspend/Resume coordination
- Verify and select Data toggle values

## 11.1 USB Enumeration

The enumeration sequence is shown below:

1. The host computer sends a **Setup** packet followed by a **Data** packet to USB address 0 requesting the Device descriptor.
2. The USB Controller decodes the request and retrieves its Device descriptor from the program memory space.
3. The host computer performs a control read sequence and the USB Controller responds by sending the Device descriptor over the USB bus.
4. After receiving the descriptor, the host computer sends a **Setup** packet followed by a **Data** packet to address 0 assigning a new USB address to the device.
5. The USB Controller stores the new address in its USB Device Address Register after the no-data control sequence is complete.
6. The host sends a request for the Device descriptor using the new USB address.
7. The USB Controller decodes the request and retrieves the Device descriptor from the program memory.
8. The host performs a control read sequence and the USB Controller responds by sending its Device descriptor over the USB bus.
9. The host generates control reads to the USB Controller to request the Configuration and Report descriptors.
10. The USB Controller retrieves the descriptors from its program space and returns the data to the host over the USB.

## 11.2 PS/2 Operation

PS/2 operation is possible with the CY7C63413/513 series through the use of firmware and several operating modes. The first enabling feature:

1. USB Bus reset on D+ and D- is an interrupt that can be disabled;
2. USB traffic can be disabled via bit 7 of the USB register;
3. D+ and D- can be monitored and driven via firmware as independent port bits.

Bits 5 and 4 of the Upstream Status and Control register are directly connected to the D+ and D- USB pins of the CY7C63413/513. These pins constantly monitor the levels of these signals with CMOS input thresholds. Firmware can poll and decode these signals as PS/2 clock and data.

Bits [2:0] defaults to '000' at reset which allows the USB SIE to control output on D+ and D-. Firmware can override the SIE and directly control the state of these pins via these 3 control bits. Since PS/2 is an open drain signaling protocol, these modes allow all 4 PS/2 states to be generated on the D+ and D- pins

## 11.3 USB Port Status and Control

USB status and control is regulated by the USB Status and Control Register located at I/O address 0x1F as shown in *Figure 11-1*. This is a read/write register. All reserved bits must be written to zero. All bits in the register are cleared during reset.

Addr:0x1F		USB Status and Control Register					
7	6	5	4	3	2	1	0
Reserved	Reserved	D+	D-	Bus Activity	Control Bit 2	Control Bit 1	Control Bit 0
		R	R	R/W	R/W	R/W	R/W

**Table 11-1. USB Status and Control Register**

The Bus Activity bit is a "sticky" bit that indicates if any non-idle USB event has occurred on the USB bus. The user firmware should check and clear this bit periodically to detect any loss of bus activity. Writing a "0" to the Bus Activity bit clears it while writing a "1" preserves the current value. In other words, the

firmware can clear the Bus Activity bit, but only the SIE can set it. The 1.024-ms timer interrupt service routine is normally used to check and clear the Bus Activity bit. The following table shows how the control bits are encoded for this register.

Control Bits	Control Action
000	Not forcing (SIE controls driver)
001	Force K (D+ HIGH, D- LOW)
010	Force J (D+ LOW, D- HIGH)
011	Force SE0 (D+ LOW, D- LOW)
100	Force SE0 (D- LOW, D+ LOW)
101	Force D- LOW, D+ HiZ
110	Force D- HiZ, D+ LOW
111	Force D- HiZ, D+ HiZ

## 12.0 USB Device

USB Device Address A includes three endpoints: EPA0, EPA1, and EPA2. End Point 0 (EPA0) allows the USB host to recognize, set up, and control the device. In particular, EPA0 is used to receive and transmit control (including set-up) packets.

## 12.1 USB Ports

The USB Controller provides one USB device address with three endpoints. The USB Device Address Register contents are cleared during a reset, setting the USB device address to zero and marking this address as disabled. *Figure 12-1* shows the format of the USB Address Register.

Addr:0x10		USB Device 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
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 12-1. USB Device Address Register**

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] must be set by firmware during the USB enumeration process to an address assigned by the USB host that does not equal zero. This register is cleared by a hardware reset or the USB bus reset.

## 12.2 Device Endpoints (3)

The USB controller communicates with the host using dedicated FIFOs, one per endpoint. Each endpoint FIFO is imple-

mented as 8 bytes of dedicated SRAM. There are three endpoints defined for Device “A” that are labeled “EPA0,” “EPA1,” and EPA2.”

All USB devices are required to have an endpoint number 0 (EPA0) that is used to initialize and control the USB device. End Point 0 provides access to the device configuration information and allows generic USB status and control accesses. End Point 0 is bidirectional as the USB controller can both receive and transmit data.

The endpoint mode registers are cleared during reset. The EPA0 endpoint mode register uses the format shown below:

Addr:0x12				USB Device EPA0, Mode Register			
Endpoint 0 Set-up Received	Endpoint 0 In Received	Endpoint 0 Out Received	Acknowledge	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 12-2. USB Device EPA0, Mode Register**

Bits[7:5] in the endpoint 0 mode registers (EPA0) are “sticky” status bits that are set by the SIE to report the type of token that was most recently received. The sticky bits must be cleared by firmware as part of the USB processing.

The endpoint mode registers for EPA1 and EPA2 do not use bits [7:5] as shown below:

Addr: 0x14, 0x16				USB Device Endpoint Mode Register			
Reserved	Reserved	Reserved	Acknowledge	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 12-3. USB Device Endpoint Mode Register**

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

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

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

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

While the ‘set-up’ bit is set, the CPU cannot write to the DMA buffers at memory locations 0xE0 through 0xE7 and 0xF8 through 0xFF. 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 Section 16.0.

The format of the endpoint Device counter registers is shown below:

Addr: 0x11, 0x13, 0x15				USB Device Counter Registers			
Data 0/1 Toggle	Data Valid	Reserved	Reserved	Byte count Bit 3	Byte count Bit 2	Byte count Bit 1	Byte count Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

**Table 12-4. USB Device Counter Registers**

Bits 0 to 3 indicate the number of data bytes to be transmitted during an IN packet, valid values are 0 to 8 inclusive. Data Valid bit 6 is used for OUT and set-up tokens only. Data 0/1

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

### 13.0 12-bit Free-running Timer

The 12-bit timer provides two interrupts (128  $\mu$ s and 1.024 ms) and allows the firmware to directly time events that are up to 4 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 4 bits into a temporary register. When the firmware reads the upper 4 bits of the timer, it is actually reading the count stored in the temporary register. The effect of this logic is to ensure a stable 12-bit timer value can be read, even when the two reads are separated in time.

#### 13.1 Timer (LSB)

Addr: 0x24								Timer Register (LSB)							
Timer Bit 7	Timer Bit 6	Timer Bit 5	Timer Bit 4	Timer Bit 3	Timer Bit 2	Timer Bit 1	Timer Bit 0	Timer Bit 7	Timer Bit 6	Timer Bit 5	Timer Bit 4	Timer Bit 3	Timer Bit 2	Timer Bit 1	Timer Bit 0
R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R

Table 13-1. Timer Register

#### 13.2 Timer (MSB)

Addr: 0x25								Timer Register (MSB)								
Reserved	Reserved	Reserved	Reserved	Reserved	Timer Bit 11	Timer Bit 10	Timer Bit 9	Timer Bit 8	Timer Bit 7	Timer Bit 6	Timer Bit 5	Timer Bit 4	Timer Bit 3	Timer Bit 2	Timer Bit 1	Timer Bit 0
					R	R	R	R								

Table 13-2. Timer Register

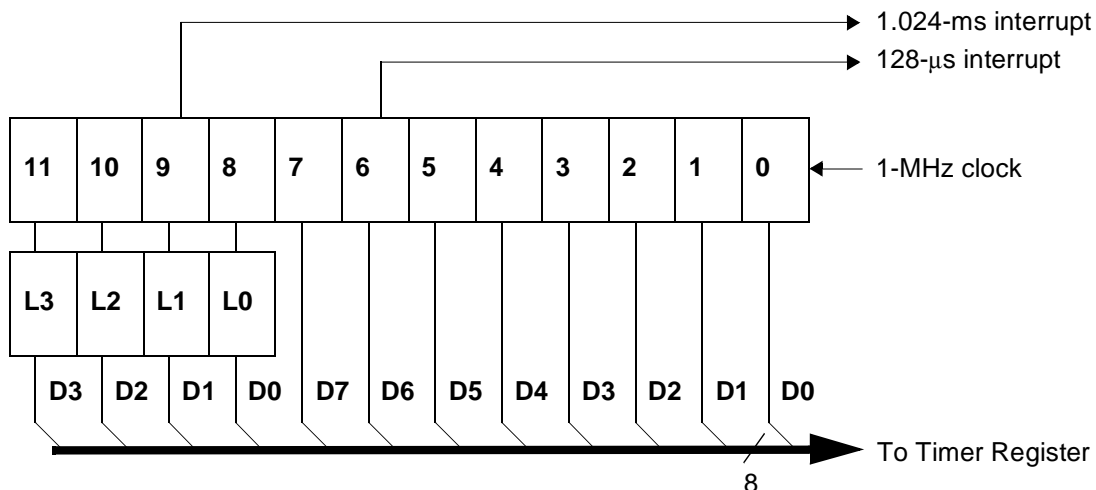


Figure 13-1. Timer Block Diagram

### 14.0 Processor Status and Control Register

Addr: 0xFF								Processor Status and Control Register								POR Default: 0x0101 WDC Reset: 0x41	
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6
IRQ Pending	Watch Dog Reset	USB Bus Reset	Power-on Reset	Suspend, Wait for Interrupt	Interrupt Mask	Single Step	Run	IRQ Pending	Watch Dog Reset	USB Bus Reset	Power-on Reset	Suspend, Wait for Interrupt	Interrupt Mask	Single Step	Run	IRQ Pending	Watch Dog Reset
R	R/W	R/W	R/W	R/W	R	R/W	R/W	R	R/W	R/W	R/W	R/W	R	R/W	R/W	R	R/W

Table 14-1. Processor Status and Control Register

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 a reset (Power On or Watch Dog). Notice, when writing

to the processor status and control register, the run bit should always be written as a "1."

The "Single Step" (bit 1) is provided to support a hardware debugger. When single step is set, the processor will execute one instruction and halt (clear the run bit). This bit must be cleared for normal operation.

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

Writing a "1" to "Suspend, Wait for Interrupts" (bit 3) will halt the processor and cause the microcontroller to enter the "suspend" mode that significantly reduces power consumption. A pending interrupt or bus activity will cause the device to come out of suspend. After coming out of suspend, the device will resume firmware execution at the instruction following the IOWR which put the part into suspend. An IOWR that attempts to put the part into suspend will be ignored if either bus activity or an interrupt is pending.

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 Watch Dog Timeout. PORS is used to determine suspend start-up timer value of 128  $\mu$ s or 96 ms.

The "USB Bus Reset" (bit 5) will occur when a USB bus reset is received. The USB Bus Reset is a singled-ended zero (SE0) that lasts more than 8 microseconds. An SE0 is defined as the condition in which both the D+ line and the D- line are LOW

at the same time. When the SIE detects this condition, the USB Bus Reset bit is set in the Processor Status and Control register and an USB Bus Reset interrupt is generated. Please note this is an interrupt to the microcontroller and does not actually reset the processor.

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

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

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

During a Watch Dog Reset, the Processor Status and Control Register is set to 01000001, which indicates a Watch Dog Reset (bit 6 set) has occurred and no interrupts are pending (bit 7 clear) yet.

## 15.0 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, effectively disabling all interrupts.

Addr: 0x20 Global Interrupt Enable Register							
7	6	5	4	3	2	1	0
Reserved	Reserved	GPIO Interrupt Enable	DAC Interrupt Enable	Reserved	1.024-ms Interrupt Enable	128- $\mu$ sec Interrupt Enable	USB Bus RST Interrupt Enable
		R/W	R/W		R/W	R/W	R/W

Table 15-1. Global Interrupt Enable Register

Addr: 0x21 USB End Point Interrupt Enable Register							
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	EPA2 Interrupt Enable	EPA1 Interrupt Enable	EPA0 Interrupt Enable
					R/W	R/W	R/W

Table 15-2. USB End Point Interrupt Enable Register

Pending interrupt requests are recognized during the last clock cycle of the current instruction. When servicing an interrupt, the hardware will first disable all interrupts by clearing the Interrupt Enable bit in the Processor Status and Control Register. Next, the interrupt latch of the current interrupt is cleared. This is followed by a CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector). The instruction in the interrupt table is typically a JMP instruction to the address of the Interrupt Service Routine (ISR). The user can re-enable interrupts in the interrupt service

routine by executing an EI instruction. Interrupts can be nested to a level limited only by the available stack space.

The Program Counter value as well as the Carry and Zero flags (CF, ZF) are automatically stored onto the Program Stack by the CALL instruction as part of the interrupt acknowledge process. The user firmware is responsible for insuring that the processor state is preserved and restored during an interrupt. The PUSH A instruction should be used as the first command in the ISR to save the accumulator value and the POP A in-

struction 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.

## 15.1 Interrupt Vectors

The Interrupt Vectors supported by the USB Controller are listed in *Table 15-1*. Although Reset is not an interrupt, per se, the first instruction executed after a reset is at PROM address 0x0000—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 15-3. Interrupt Vector Assignments**

Interrupt Vector Number	ROM Address	Function
not applicable	0x0000	Execution after Reset begins here
1	0x0002	USB Bus Reset interrupt
2	0x0004	128- $\mu$ s timer interrupt
3	0x0006	1.024-ms timer interrupt
4	0x0008	USB Address A Endpoint 0 interrupt
5	0x000A	USB Address A Endpoint 1 interrupt
6	0x000C	USB Address A Endpoint 2 interrupt
7	0x000E	Reserved
8	0x0010	Reserved
9	0x0012	Reserved
10	0x0014	DAC interrupt
11	0x0016	GPIO interrupt
12	0x0018	Reserved

## 15.2 Interrupt Latency

Interrupt latency can be calculated from the following equation:

$$\begin{aligned} \text{Interrupt Latency} = & (\text{Number of clock cycles remaining in the} \\ & \text{current instruction}) \\ & + (10 \text{ clock cycles for the CALL instruction}) \\ & + (5 \text{ clock cycles for the JMP instruction}) \end{aligned}$$

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

### 15.2.1 USB Bus Reset Interrupt

The USB Bus Reset interrupt is asserted when a USB bus reset condition is detected. A USB bus reset is indicated by a single ended zero (SE0) on the upstream port for more than 8 microseconds.

### 15.2.2 Timer Interrupt

There are two timer interrupts: the 128- $\mu$ s interrupt and the 1.024-ms interrupt. The user should disable both timer interrupts before going into the suspend mode to avoid possible conflicts between servicing the interrupts first or the suspend request first.

### 15.2.3 USB Endpoint Interrupts

There are three USB endpoint interrupts, one per endpoint. The USB endpoints interrupt after the either the USB host or the USB controller sends a packet to the USB.

### 15.2.4 DAC Interrupt

Each DAC I/O pin can generate an interrupt, if enabled. The interrupt polarity for each DAC I/O pin is programmable. A positive polarity is a rising edge input while a negative polarity is a falling edge input. All of the DAC pins share a single interrupt vector, which means the firmware will need to read the DAC port to determine which pin or pins caused an interrupt.

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

### 15.2.5 GPIO Interrupt

Each of the 32 GPIO pins can generate an interrupt, if enabled. The interrupt polarity can be programmed for each GPIO port as part of the GPIO configuration. All of the GPIO pins share a single interrupt vector, which means the firmware will need to read the GPIO ports with enabled interrupts to determine which pin or pins caused an interrupt.

Please note that if one port pin triggered an interrupt, no other port pins can cause a GPIO interrupt until that port pin has returned to its inactive (non-trigger) state or its corresponding

port interrupt enable bit is cleared. The USB Controller does not assign interrupt priority to different port pins and the Port

Interrupt Enable Registers are not cleared during the interrupt acknowledge process.

## 16.0 Truth Tables

**Table 16-1. USB Register Mode Encoding**

Mode	Encoding	Setup	In	Out	Comments
Disable	<b>0000</b>	ignore	ignore	ignore	Ignore all USB traffic to this endpoint
Nak In/Out	<b>0001</b>	accept	NAK	NAK	Forced from Set-up on Control endpoint, from modes other than 0000
Status Out Only	<b>0010</b>	accept	stall	check	For Control endpoints
Stall In/Out	<b>0011</b>	accept	stall	stall	For Control endpoints
Ignore In/Out	<b>0100</b>	accept	ignore	ignore	For Control endpoints
Isochronous Out	<b>0101</b>	ignore	ignore	always	Available to low speed devices, future USB spec enhancements
Status In Only	<b>0110</b>	accept	TX 0	stall	For Control Endpoints
Isochronous In	<b>0111</b>	ignore	TX cnt	ignore	Available to low speed devices, future USB spec enhancements
Nak Out	<b>1000</b>	ignore	ignore	NAK	An ACK from mode 1001 --> 1000
Ack Out	<b>1001</b>	ignore	ignore	ACK	This mode is changed by SIE on issuance of ACK --> 1000
Nak Out - Status In	<b>1010</b>	accept	TX 0	NAK	An ACK from mode 1011 --> 1010
Ack Out - Status In	<b>1011</b>	accept	TX 0	ACK	This mode is changed by SIE on issuance of ACK --> 1010
Nak In	<b>1100</b>	ignore	NAK	ignore	An ACK from mode 1101 --> 1100
Ack In	<b>1101</b>	ignore	TX cnt	ignore	This mode is changed by SIE on issuance of ACK --> 1100
Nak In - Status Out	<b>1110</b>	accept	NAK	check	An ACK from mode 1111 --> 1110 NAK In - Status Out
Ack In - Status Out	<b>1111</b>	accept	TX cnt	Check	This mode is changed by SIE on issuance of ACK -->1110

The 'In' column represents the SIE's response to the token type.

A disabled endpoint will remain such until firmware changes it, and all endpoints reset to disabled.

Any Setup packet to an enabled and accepting endpoint will be changed by the SIE to 0001 (NAKing). Any mode which indicates the acceptance of a Setup will acknowledge it.

Most modes that control transactions involving an ending ACK will be changed by the SIE to a corresponding mode which NAKs follow on packets.

A Control endpoint has three extra status bits for PID (Setup, In and Out), but must be placed in the correct mode to function as such. Also a non-Control endpoint can be made to act as a Control endpoint if it is placed in a non appropriate mode.

A 'check' on an Out token during a Status transaction checks to see that the Out is of zero length and has a Data Toggle (DTOG) of 1.



**Table 16-2. Details of Modes for Differing Traffic Conditions**

End Point Mode											PID				Set End Point Mode					
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	Setup	In	Out	ACK	3	2	1	0	response	int
<b>Setup Packet (if accepting)</b>																				
See Table 16-1				Setup	<= 10	data	valid	updates	1	updates	1	UC	UC	1	0	0	0	1	ACK	yes
See Table 16-1				Setup	> 10	junk	x	updates	updates	updates	1	UC	UC	UC	NoChange				ignore	yes
See Table 16-1				Setup	x	junk	invalid	updates	0	updates	1	UC	UC	UC	NoChange				ignore	yes
Disabled																				
0	0	0	0	x	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
Nak In/Out																				
0	0	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes
0	0	0	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				NAK	yes
Ignore In/Out																				
0	1	0	0	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	0	0	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
Stall In/Out																				
0	0	1	1	Out	x	UC	x	UC	UC	UC	UC	UC	1	UC	NoChange				Stall	yes
0	0	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				Stall	yes
<b>Control Write</b>																				
Normal Out/premature status In																				
1	0	1	1	Out	<= 10	data	valid	updates	1	updates	UC	UC	1	1	1	0	1	0	ACK	yes
1	0	1	1	Out	> 10	junk	x	updates	updates	updates	UC	UC	1	UC	NoChange				ignore	yes
1	0	1	1	Out	x	junk	invalid	updates	0	updates	UC	UC	1	UC	NoChange				ignore	yes
1	0	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	NoChange				TX 0	yes
NAK Out/premature status In																				
1	0	1	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes
1	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	1	NoChange				TX 0	yes
Status In/extra Out																				
0	1	1	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	0	0	1	1	Stall	yes
0	1	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	1	NoChange				TX 0	yes
<b>Control Read</b>																				
Normal In/premature status Out																				
1	1	1	1	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
1	1	1	1	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	1	1	1	0	ACK (back)	yes
Nak In/premature status Out																				
1	1	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
1	1	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	0	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				NAK	yes
Status Out/extra In																				
0	0	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
0	0	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes

**Table 16-2. Details of Modes for Differing Traffic Conditions (continued)**

End Point Mode												PID				Set End Point Mode					
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	Setup	In	Out	ACK	3	2	1	0	response	int	
0	0	1	0	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes	
0	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	U C	U C	U C	U C	ignore	no	
0	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	U C	U C	U C	U C	ignore	no	
0	0	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	0	0	1	1	Stall	yes	
<b>Out endpoint</b>																					
Normal Out/erroneous In																					
1	0	0	1	Out	<= 10	data	valid	updates	1	updates	UC	UC	1	1	1	0	0	0	ACK	yes	
1	0	0	1	Out	> 10	junk	x	updates	updates	updates	UC	UC	1	UC	NoChange				ignore	yes	
1	0	0	1	Out	x	junk	invalid	updates	0	updates	UC	UC	1	UC	NoChange				ignore	yes	
1	0	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
NAK Out/erroneous In																					
1	0	0	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes	
1	0	0	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
1	0	0	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
1	0	0	0	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
Isochronous endpoint (Out)																					
0	1	0	1	Out	x	updates	updates	updates	updates	updates	UC	UC	1	1	NoChange				RX	yes	
0	1	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
<b>In endpoint</b>																					
Normal In/erroneous Out																					
1	1	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
1	1	0	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	1	1	0	0	ACK (back)	yes	
NAK In/erroneous Out																					
1	1	0	0	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
1	1	0	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				NAK	yes	
Isochronous endpoint (In)																					
0	1	1	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no	
0	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				TX	yes	

### 17.0 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 + $V_{CC}+0.5V$
Max. Output Current into Port 0,1,2,3 and DAC[1:0] Pins .....	60 mA
Max. Output Current into DAC[7:2] Pins .....	10 mA
Power Dissipation .....	300 mW
Static Discharge Voltage .....	> 2000V
Latch-up Current .....	> 200 mA

### 18.0 DC Characteristics $F_{osc} = 6 \text{ MHz}$ ; Operating Temperature = 0 to 70°C

	Parameter	Min.	Max.	Unit	Conditions
<b>General</b>					
$V_{CC(1)}$	Operating Voltage	4.0	5.5	V	Non USB activity (note 1)
$V_{CC(2)}$	Operating Voltage	4.35	5.25	V	USB activity (note 2)
$I_{CC1}$	$V_{CC}$ Operating Supply Current		40	mA	$V_{CC} = 5.5V$
$I_{CC2}$	$V_{CC} = 4.35V$		15	mA	
$I_{SB1}$	Supply Current - Suspend Mode		30	$\mu A$	Oscillator off, D- > $V_{oh \text{ min}}$
$V_{PP}$	Programming Voltage (disabled)	-0.4	0.4	V	
$T_{start}$	Resonator Start-up Interval		256	$\mu s$	$V_{cc} = 5.0V$ , ceramic resonator
$t_{int1}$	Internal Timer #1 Interrupt Period	128	128	$\mu s$	
$t_{int2}$	Internal Timer #2 Interrupt Period	1.024	1.024	ms	
$t_{watch}$	Watch Dog Timer Period	8.192	14.33	ms	
$I_{il}$	Input Leakage Current		1	$\mu A$	Any pin
$I_{sm}$	Max $I_{SS}$ IO Sink Current		60	mA	Cumulative across all ports (note 8)
<b>Power-On Reset</b>					
$t_{vccs}$	$V_{CC}$ Reset Slew	0.001	200	ms	Linear ramp: 0 to 4.35V (notes 4,5)
<b>USB Interface</b>					
$V_{oh}$	Static Output HIGH	2.8	3.6	V	15k $\pm$ 5% ohms to Gnd (note 2)
$V_{ol}$	Static Output LOW		0.3	V	
$V_{di}$	Differential Input Sensitivity	0.2		V	$ (D+)-(D-) $
$V_{cm}$	Differential Input Common Mode Range	0.8	2.5	V	9-1
$V_{se}$	Single-Ended Receiver Threshold	0.8	2.0	V	
$C_{in}$	Transceiver Capacitance		20	pF	
$I_{lo}$	Hi-Z State Data Line Leakage	-10	10	$\mu A$	$0 \text{ V} < V_{in} < 3.3 \text{ V}$
$R_{pu}$	Bus Pull-up Resistance ( $V_{CC}$ option)	7.35K	7.65	k $\Omega$	7.5 k $\Omega$ $\pm$ 2% to $V_{CC}$
$R_{pu}$	Bus Pull-up Resistance (Ext. 3.3V option)	1.425	1.575	k $\Omega$	1.5 k $\Omega$ $\pm$ 5% to 3.0-3.6V
$R_{pd}$	Bus Pull-down Resistance	14.25	15.75	k $\Omega$	15 k $\Omega$ $\pm$ 5%
<b>General Purpose I/O Interface</b>					
$R_{up}$	Pull-up Resistance	4.9K	9.1K	Ohms	
$V_{ith}$	Input Threshold Voltage	45%	65%	$V_{CC}$	All ports, LOW to HIGH edge
$V_H$	Input Hysteresis Voltage	6%	12%	$V_{CC}$	All ports, HIGH to LOW edge
$I_{ol}$	Sink Current	7.2	16.5	mA	Port 3, $V_{out} = 1.0V$ (note 1)

**18.0 DC Characteristics** Fosc = 6 MHz; Operating Temperature = 0 to 70°C (continued)

	Parameter	Min.	Max.	Unit	Conditions
I <sub>ol</sub>	Sink Current	3.5	10.6	mA	Port 0,1,2, Vout = 2.0V (note 1)
I <sub>oh</sub>	Source Current	1.4	7.5	mA	Voh = 2.4V (all ports 0,1,2,3) (note 1)
<b>DAC Interface</b>					
R <sub>up</sub>	Pull-up Resistance	8.0K	20.0K	Ohms	(note 11)
I <sub>sink0(0)</sub>	DAC[7:2] Sink Current (0)	0.1	0.3	mA	Vout = 2.0 V DC (note 2)
I <sub>sink0(F)</sub>	DAC[7:2] Sink Current (F)	0.5	1.5	mA	Vout = 2.0 V DC (note 2)
I <sub>sink1(0)</sub>	DAC[1:0] Sink Current (0)	1.6	4.8	mA	Vout = 2.0 V DC (note 2)
I <sub>sink1(F)</sub>	DAC[1:0] Sink Current (F)	8	24	mA	Vout = 2.0 V DC (note 2)
I <sub>range</sub>	Programmed Isink Ratio: max/min	4	6		Vout = 2.0 V DC (notes 2,9)
I <sub>lin</sub>	Differential Nonlinearity		0.5	lsb	Any pin (note 6)
t <sub>sink</sub>	Current Sink Response Time		0.8	μs	Full scale transition
T <sub>ratio</sub>	Tracking Ratio DAC[1:0] to DAC[7:2]	14	21		Vout = 2.0V (note 7)

**19.0 Switching Characteristics**

Parameter	Description	Min.	Max.	Unit	Conditions
<b>Clock</b>					
t <sub>CYC</sub>	Input Clock Cycle Time	165.0	168.3	ns	
t <sub>CH</sub>	Clock HIGH Time	0.45 t <sub>CYC</sub>		ns	
t <sub>CL</sub>	Clock LOW Time	0.45 t <sub>CYC</sub>		ns	
<b>USB Driver Characteristics</b>					
t <sub>r</sub>	Transition Rise Time	75		ns	C <sub>Load</sub> = 50 pF [2, 3]
t <sub>r</sub>	Transition Rise Time		300	ns	C <sub>Load</sub> = 600 pF [2, 3]
t <sub>f</sub>	Transition Fall Time	75		ns	C <sub>Load</sub> = 50 pF [2, 3]
t <sub>f</sub>	Transition Fall Time		300	ns	C <sub>Load</sub> = 600 pF [2, 3]
t <sub>rfm</sub>	Rise/Fall Time Matching	80	125	%	t <sub>r</sub> /t <sub>f</sub> [2, 3]
V <sub>crs</sub>	Output Signal Crossover Voltage	1.3	2.0	V	[2, 3]
<b>USB Data Timing</b>					
t <sub>drate</sub>	Low Speed Data Rate	1.4775	1.5225	Mbs	Ave. Bit Rate (1.5 Mb/s ± 1.5%)
t <sub>djr1</sub>	Receiver Data Jitter Tolerance	-75	75	ns	To Next Transition [10]
t <sub>djr2</sub>	Receiver Data Jitter Tolerance	-45	45	ns	For Paired Transitions [10]
t <sub>deop</sub>	Differential to EOP Transition Skew	-40	100	ns	[8]
t <sub>eoпр1</sub>	EOP Width at Receiver	330		ns	Rejects as EOP [10]
t <sub>eoпр2</sub>	EOP Width at Receiver	675		ns	Accepts as EOP [10]
t <sub>eoрt</sub>	Source EOP Width	1.25	1.50	μs	
t <sub>udj1</sub>	Differential Driver Jitter	-95	95	ns	To next transition, Figure 19-5
t <sub>udj2</sub>	Differential Driver Jitter	-150	150	ns	To paired transition, Figure 19-5

**Notes:**

- Functionality is guaranteed of the V<sub>CC(1)</sub> range, except USB transmitter and DACs.
- USB transmitter functionality is guaranteed over the V<sub>CC(2)</sub> range, as well as DAC outputs.
- Per Table 7-7 of revision 1.1 of USB specification, for C<sub>LOAD</sub> of 50–600 pF.
- Port 3 bit 7 controls whether the parts goes into suspend after a POR event or waits 128 ms to begin running.
- POR will re-occur whenever V<sub>CC</sub> drops to approximately 2.5V.
- Measured as largest step size vs. nominal according to measured full scale and zero programmed values.
- T<sub>ratio</sub> = Isink1[1:0](n)/Isink0[7:2](n) for the same n, programmed.
- Total current cumulative across all Port pins flowing to V<sub>SS</sub> is limited to minimize Ground-Drop noise effects.
- I<sub>range</sub>: Isinkn(15)/ Isinkn(0) for the same pin.
- Measured at crossover point of differential data signals.
- Limits total bus capacitance loading (C<sub>LOAD</sub>) to 400 pF per section 7.1.5 of revision 1.1 of USB specification.

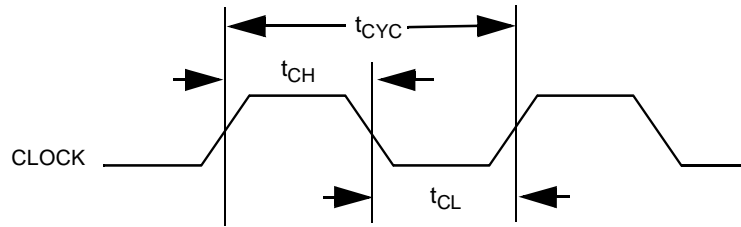


Figure 19-1. Clock Timing

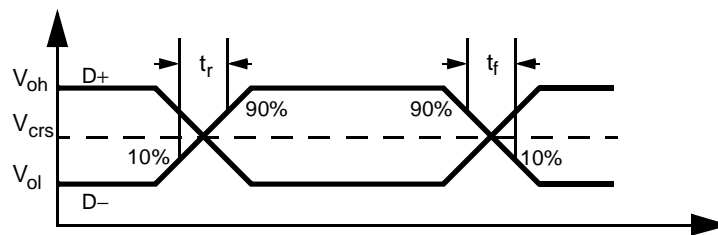


Figure 19-2. USB Data Signal Timing

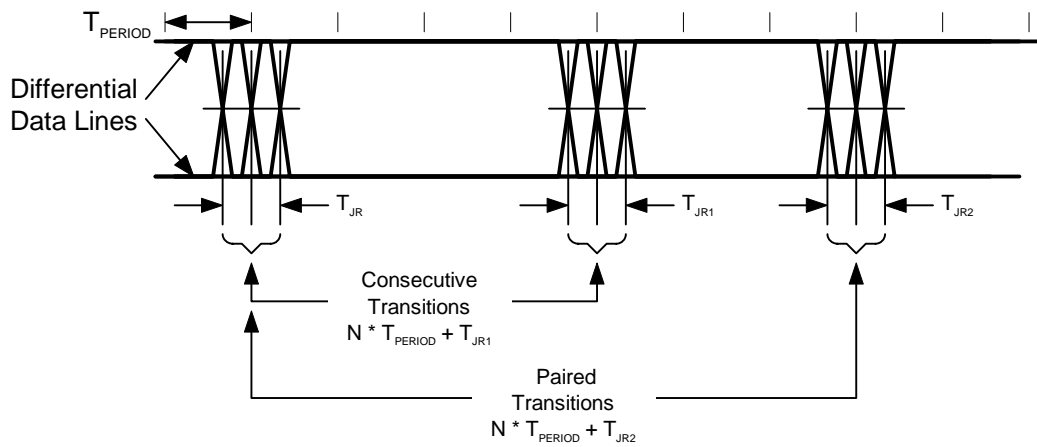
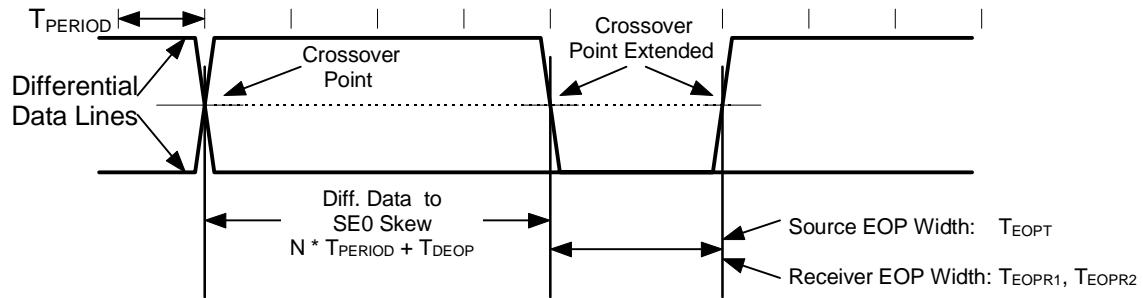
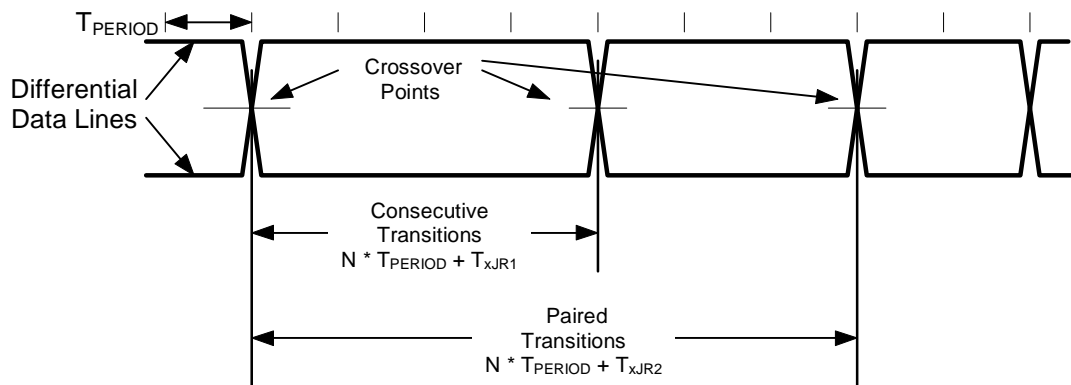


Figure 19-3. Receiver Jitter Tolerance

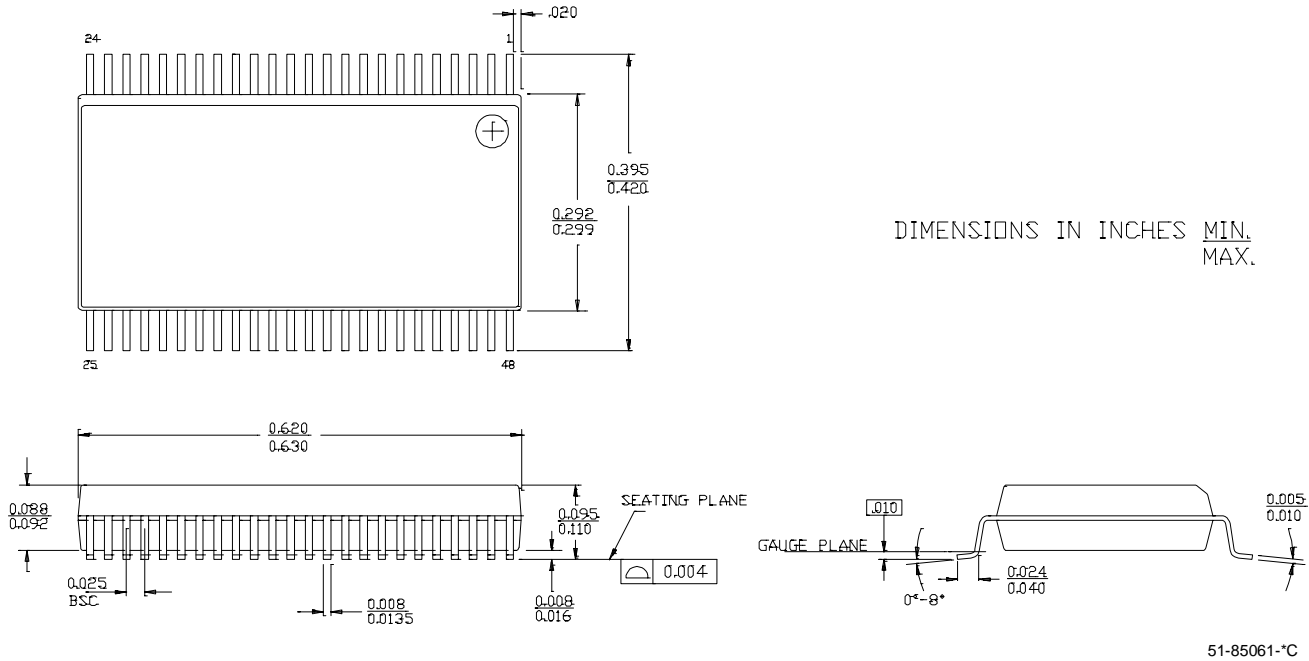
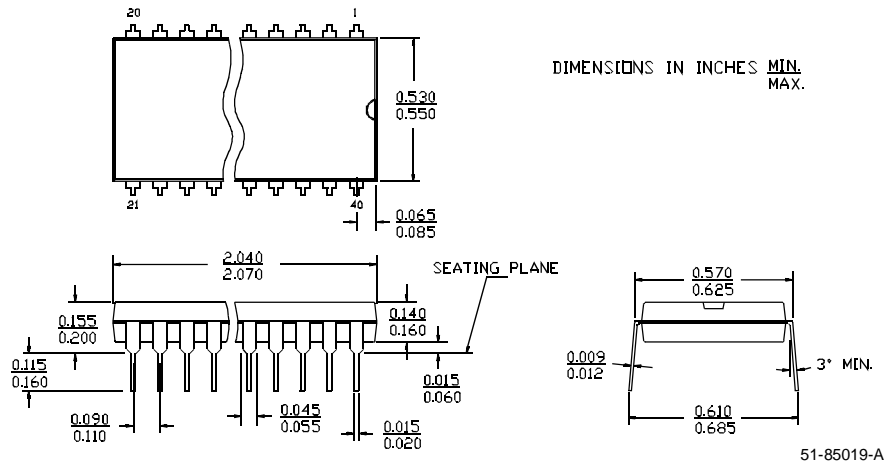

**Figure 19-4. Differential to EOP Transition Skew and EOP Width**

**Figure 19-5. Differential Data Jitter**

## 20.0 Ordering Information

Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C63413-PC	8 KB	P17	40-Pin (600-Mil) PDIP	Commercial
CY7C63413-PVC	8 KB	O48	48-Lead Shrunken Small Outline Package	Commercial
CY7C63513-PVC	8 KB	O48	48-Lead Shrunken Small Outline Package	Commercial
CY7C63413-PXC	8 KB	P2	40-Pin (600-Mil) PDIP Lead-free	Commercial
CY7C63413-PVXC	8 KB	SP48	48-Lead Shrunken Small Outline Package	Commercial
CY7C63513-PVXC	8 KB	SP48	48-Lead SSOP Lead-Free	Commercial
CY7C63413-XWC	8KB	XW	Die Lead-Free	Commercial

**21.0 Die Pad Locations**
**Table 21-1. Die Pad Locations (in microns)**

<b>Pad #</b>	<b>Pin Name</b>	<b>X</b>	<b>Y</b>	<b>Pad #</b>	<b>Pin Name</b>	<b>X</b>	<b>Y</b>
1	D+	1496.95	2995.00	48	V <sub>CC</sub>	1619.65	3023.60
2	D-	467.40	2995.00	47	V <sub>SS</sub>	1719.65	3023.60
3	Port3[7]	345.15	3023.60	46	Port3[6]	1823.10	3023.60
4	Port3[5]	242.15	3023.60	45	Port3[4]	1926.10	3023.60
5	Port3[3]	98.00	2661.25	44	Port3[2]	2066.30	2657.35
6	Port3[1]	98.00	2558.25	43	Port3[0]	2066.30	2554.35
7	Port2[7]	98.00	2455.25	42	Port2[6]	2066.30	2451.35
8	Port2[5]	98.00	2352.25	41	Port2[4]	2066.30	2348.35
9	Port2[3]	98.00	2249.25	40	Port2[2]	2066.30	2245.35
10	Port2[1]	98.00	2146.25	39	Port2[0]	2066.30	2142.35
11	Por1[7]	98.00	1134.25	38	Port1[6]	2066.30	1130.35
12	Por1[5]	98.00	1031.25	37	Port1[4]	2066.30	1027.35
13	Por1[3]	98.00	928.25	36	Port1[2]	2066.30	924.35
14	Por1[1]	98.00	825.25	35	Port1[0]	2066.30	821.35
15	DAC7	98.00	721.05	34	DAC6	2066.30	719.55
16	DAC5	98.00	618.05	33	DAC4	2066.30	616.55
17	Port0[7]	98.00	516.25	32	Port0[6]	2066.30	512.35
18	Port0[5]	98.00	413.25	31	Port0[4]	2066.30	409.35
19	Port0[3]	306.30	98.00	30	Port0[2]	1858.00	98.00
20	Port0[1]	442.15	98.00	29	Port0[0]	1718.30	98.00
21	DAC3	593.40	98.00	28	DAC2	1618.50	98.00
22	DAC1	696.40	98.00	27	DAC0	1513.50	98.00
23	V <sub>PP</sub>	824.25	98.00	26	XtalOut	1301.90	98.00
24	V <sub>SS</sub>	949.65	98.00	25	XtalIn	1160.50	98.00

**22.0 Package Diagrams**
**48-Lead Shrunken Small Outline Package O48**

**40-Lead (600-Mil) Molded DIP P17**


**23.0 Document History Page**

<b>Document Title: CY7C63413, CY7C63513 Low-speed High I/O, 1.5 -Mbps USB Controller</b>				
<b>Document Number: 38-08027</b>				
<b>REV.</b>	<b>ECN NO.</b>	<b>Issue Date</b>	<b>Orig. of Change</b>	<b>Description of Change</b>
**	116224	06/12/02	DSG	Change from Spec number: 38-00754 to 38-08027
*A	237148	SEE ECN	KKU	Removed 24 pin package, CY7C63411/12, CY7C63511/12 and CY7C636XX parts Added Lead-Free part numbers to section 20.0 Added USB Logo.



LittleDiode supplies new, hard to find or obsolete electronic components and semiconductors all over the world.

With over two million different components listed you are sure to find the part you need.

Feel free to visit us today at our online store:

[LittleDiode.com](http://LittleDiode.com)

Looking forward to providing you with the best possible service.