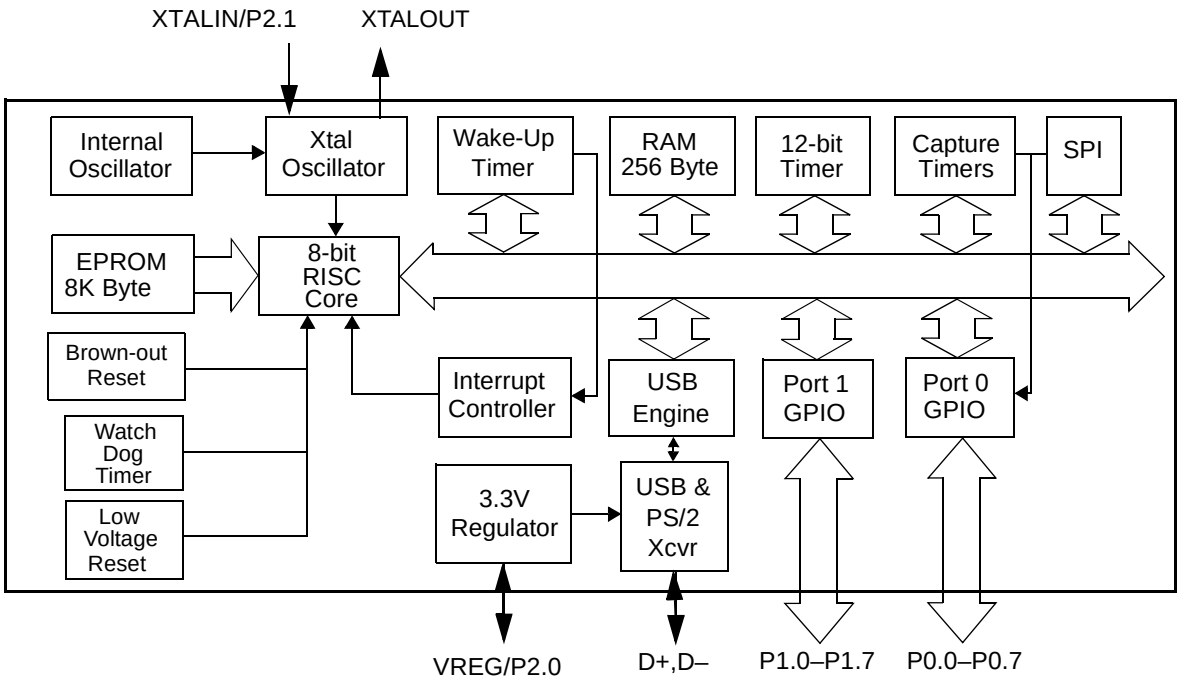
CY7C63722
CY7C63723
CY7C63743

enCoRe™ USB Combination Low-Speed USB and PS/2 Peripheral Controller

1.0 Features

- enCoRe™ USB - enhanced Component Reduction
  - Internal oscillator eliminates the need for an external crystal or resonator
  - Interface can auto-configure to operate as PS/2 or USB without the need for external components to switch between modes (no General Purpose I/O [GPIO] pins needed to manage dual mode capability)
  - Internal 3.3V regulator for USB pull-up resistor
  - Configurable GPIO for real-world interface without external components
- Flexible, cost-effective solution for applications that combine PS/2 and low-speed USB, such as mice, game-pads, joysticks, and many others.
- USB Specification Compliance
  - Conforms to USB Specification, Version 2.0
  - Conforms to USB HID Specification, Version 1.1
  - Supports one low-speed USB device address and three data endpoints
  - Integrated USB transceiver
  - 3.3V regulated output for USB pull-up resistor
- 8-bit RISC microcontroller
  - Harvard architecture
  - 6-MHz external ceramic resonator or internal clock mode
  - 12-MHz internal CPU clock
  - Internal memory
  - 256 bytes of RAM
  - 8 Kbytes of EPROM
  - Interface can auto-configure to operate as PS/2 or USB
  - No external components for switching between PS/2 and USB modes
  - No GPIO pins needed to manage dual mode capability
- I/O ports
  - Up to 16 versatile GPIO pins, individually configurable
  - High current drive on any GPIO pin: 50 mA/pin current sink
  - Each GPIO pin supports high-impedance inputs, internal pull-ups, open drain outputs or traditional CMOS outputs
  - Maskable interrupts on all I/O pins
- SPI serial communication block
  - Master or slave operation
  - 2 Mbit/s transfers
- Four 8-bit Input Capture registers
  - Two registers each for two input pins
  - Capture timer setting with five prescaler settings
  - Separate registers for rising and falling edge capture
  - Simplifies interface to RF inputs for wireless applications
- Internal low-power wake-up timer during suspend mode
  - Periodic wake-up with no external components
- Optional 6-MHz internal oscillator mode
  - Allows fast start-up from suspend mode
- Watchdog Reset (WDR)
- Low-voltage Reset at 3.75V
- Internal brown-out reset for suspend mode
- Improved output drivers to reduce EMI
- Operating voltage from 4.0V to 5.5VDC
- Operating temperature from 0°C to 70°C
- CY7C63723 available in 18-pin SOIC, 18-pin PDIP
- CY7C63743 available in 24-pin SOIC, 24-pin PDIP, 24-pin QSOP
- CY7C63722 available in DIE form
- Industry standard programmer support

2.0 Logic Block Diagram



3.0 Functional Overview

3.1 enCoRe USB—The New USB Standard

Cypress has reinvented its leadership position in the low-speed USB market with a new family of innovative microcontrollers. Introducing...enCoRe USB—"enhanced Component Reduction." Cypress has leveraged its design expertise in USB solutions to create a new family of low-speed USB microcontrollers that enables peripheral developers to design new products with a minimum number of components. At the heart of the enCoRe USB technology is the break-through design of a crystalless oscillator. By integrating the oscillator into our chip, an external crystal or resonator is no longer needed. We have also integrated other external components commonly found in low-speed USB applications such as pull-up resistors, wake-up circuitry, and a 3.3V regulator. All of this adds up to a lower system cost.

The CY7C637xx is an 8-bit RISC one-time-programmable (OTP) microcontroller. The instruction set has been optimized specifically for USB and PS/2 operations, although the microcontrollers can be used for a variety of other embedded applications.

The CY7C637xx features up to 16 GPIO pins to support USB, PS/2 and other applications. The I/O pins are grouped into two ports (Port 0 to 1) where each pin can be individually configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs with programmable drive strength of up to 50 mA output drive. 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 CY7C637xx microcontrollers feature an internal oscillator. With the presence of USB traffic, the internal oscillator can be set to precisely tune to USB timing requirements (6 MHz ±1.5%). Optionally, an external 6-MHz ceramic resonator can be used to provide a higher precision reference for USB operation. 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 CY7C637xx has 8 Kbytes of EPROM and 256 bytes of data RAM for stack space, user variables, and USB FIFOs.

These parts include low-voltage reset logic, a Watchdog timer, a vectored interrupt controller, a 12-bit free-running timer, and capture timers. The low-voltage reset (LVR) logic detects when power is applied to the device, resets the logic to a known state, and begins executing instructions at EPROM address 0x0000. LVR will also reset the part when V_{CC} drops below the operating voltage range. The Watchdog timer can be used to ensure the firmware never gets stalled for more than approximately 8 ms.

The microcontroller supports 10 maskable interrupts in the vectored interrupt controller. Interrupt sources include the USB Bus-Reset, the 128-µs and 1.024-ms outputs from the free-running timer, three USB endpoints, two capture timers, an internal wake-up timer and the GPIO ports. The timers bits cause periodic interrupts when enabled. The USB endpoints interrupt after USB transactions complete on the bus. The capture timers interrupt whenever a new timer value is saved due to a selected GPIO edge event. The GPIO ports 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 pin. The interrupt polarity can be either rising or falling edge.

The free-running 12-bit timer clocked at 1 MHz provides two interrupt sources as noted above (128 µs and 1.024 ms). The timer can be used to measure the duration of an event under firmware control by reading the timer at the start and end of an

event, and subtracting the two values. The four capture timers save a programmable 8 bit range of the free-running timer when a GPIO edge occurs on the two capture pins (P0.0, P0.1).

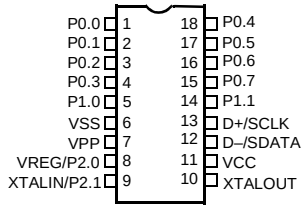
The CY7C637xx includes an integrated USB serial interface engine (SIE) that supports the integrated peripherals. The hardware supports one USB device address with three endpoints. The SIE allows the USB host to communicate with the function integrated into the microcontroller. A 3.3V regulated output pin provides a pull-up source for the external USB resistor on the D– pin.

The USB D+ and D– USB pins can alternately be used as PS/2 SCLK and SDATA signals, so that products can be designed to respond to either USB or PS/2 modes of operation. PS/2 operation is supported with internal pull-up resistors on SCLK and SDATA, the ability to disable the regulator output pin, and an interrupt to signal the start of PS/2 activity. No external components are necessary for dual USB and PS/2 systems, and no GPIO pins need to be dedicated to switching between modes. Slow edge rates operate in both modes to reduce EMI.

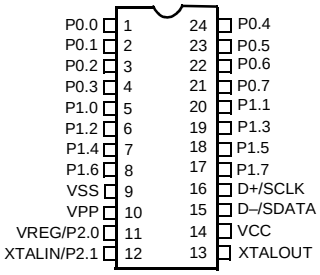
4.0 Pin Configurations

Top View

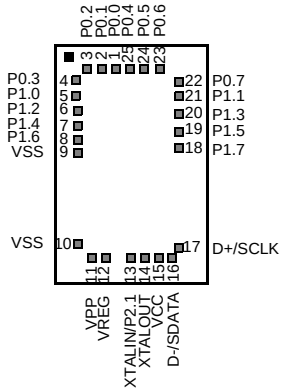
CY7C63723
18-pin SOIC/PDIP



CY7C63743
24-pin SOIC/PDIP/QSOP



CY7C63722-XC
DIE



5.0 Pin Definitions

Name	I/O	CY7C63723 18-Pin	CY7C63743 24-Pin	CY7C63722 25-Pad	Description
D–/SDATA, D+/SCLK	I/O	12 13	15 16	16 17	USB differential data lines (D– and D+), or PS/2 clock and data signals (SDATA and SCLK)
P0[7:0]	I/O	1, 2, 3, 4, 15, 16, 17, 18	1, 2, 3, 4, 21, 22, 23, 24	1, 2, 3, 4, 22, 23, 24, 25	GPIO Port 0 capable of sinking up to 50 mA/pin, or sinking controlled low or high programmable current. Can also source 2 mA current, provide a resistive pull-up, or serve as a high-impedance input. P0.0 and P0.1 provide inputs to Capture Timers A and B, respectively.
P1[7:0]	I/O	5, 14	5, 6, 7, 8, 17, 18, 19, 20	5, 6, 7, 8, 18, 19, 20, 21	IO Port 1 capable of sinking up to 50 mA/pin, or sinking controlled low or high programmable current. Can also source 2 mA current, provide a resistive pull-up, or serve as a high-impedance input.
XTALIN/P2.1	IN	9	12	13	6-MHz ceramic resonator or external clock input, or P2.1 input
XTALOUT	OUT	10	13	14	6-MHz ceramic resonator return pin or internal oscillator output
V_{PP}		7	10	11	Programming voltage supply, ground for normal operation
V_{CC}		11	14	15	Voltage supply
VREG/P2.0		8	11	12	Voltage supply for 1.3-kΩ USB pull-up resistor (3.3V nominal). Also serves as P2.0 input.
V_{SS}		6	9	9, 10	Ground

6.0 Programming Model

Refer to the *CYASM Assembler User's Guide* for more details on firmware operation with the CY7C637xx microcontrollers.

6.1 Program Counter (PC)

The 14-bit program counter (PC) allows access for up to 8 Kbytes of EPROM using the CY7C637xx architecture. The program counter is cleared during reset, such that the first instruction executed after a reset is at address 0x0000. This instruction 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 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.

6.2 8-bit Accumulator (A)

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

6.3 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.

6.4 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 that 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 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 reenable 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.

Note that there are restrictions in using the JMP, CALL, and INDEX instructions across the 4-KByte boundary of the program memory. Refer to the *CYASM Assembler User's Guide* for a detailed description.

6.5 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 equals 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, the firmware should set the DSP to an appropriate location to avoid a memory conflict with RAM dedicated to USB FIFOs. The memory requirements for the USB endpoints are shown in Section 8.2. For example, assembly instructions to set the DSP to 20h (giving 32 bytes for program and data stack combined) are shown below.

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

6.6 Address Modes

The CY7C637xx microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

6.6.1 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 0x30:

- MOV A, 30h

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 "0xE8h". 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 30h
- MOV A,DSPINIT

6.6.2 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]

6.6.3 Indexed

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

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

This would have the effect of loading A with the fourth element of the SRAM "array" that begins at address 0x10h. The fourth element would be at address 0x13h.

7.0 Instruction Set Summary

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

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

MNEMONIC	Operand	Opcode	Cycles		MNEMONIC	Operand	Opcode	Cycles
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 (or 4)
CALL	addr	90-9F	10		JNC	addr	D0-DF	5 (or 4)
JZ	addr	A0-AF	5 (or 4)		JACC	addr	E0-EF	7
JNZ	addr	B0-BF	5 (or 4)		INDEX	addr	F0-FF	14

8.0 Memory Organization

8.1 Program Memory Organization[1]

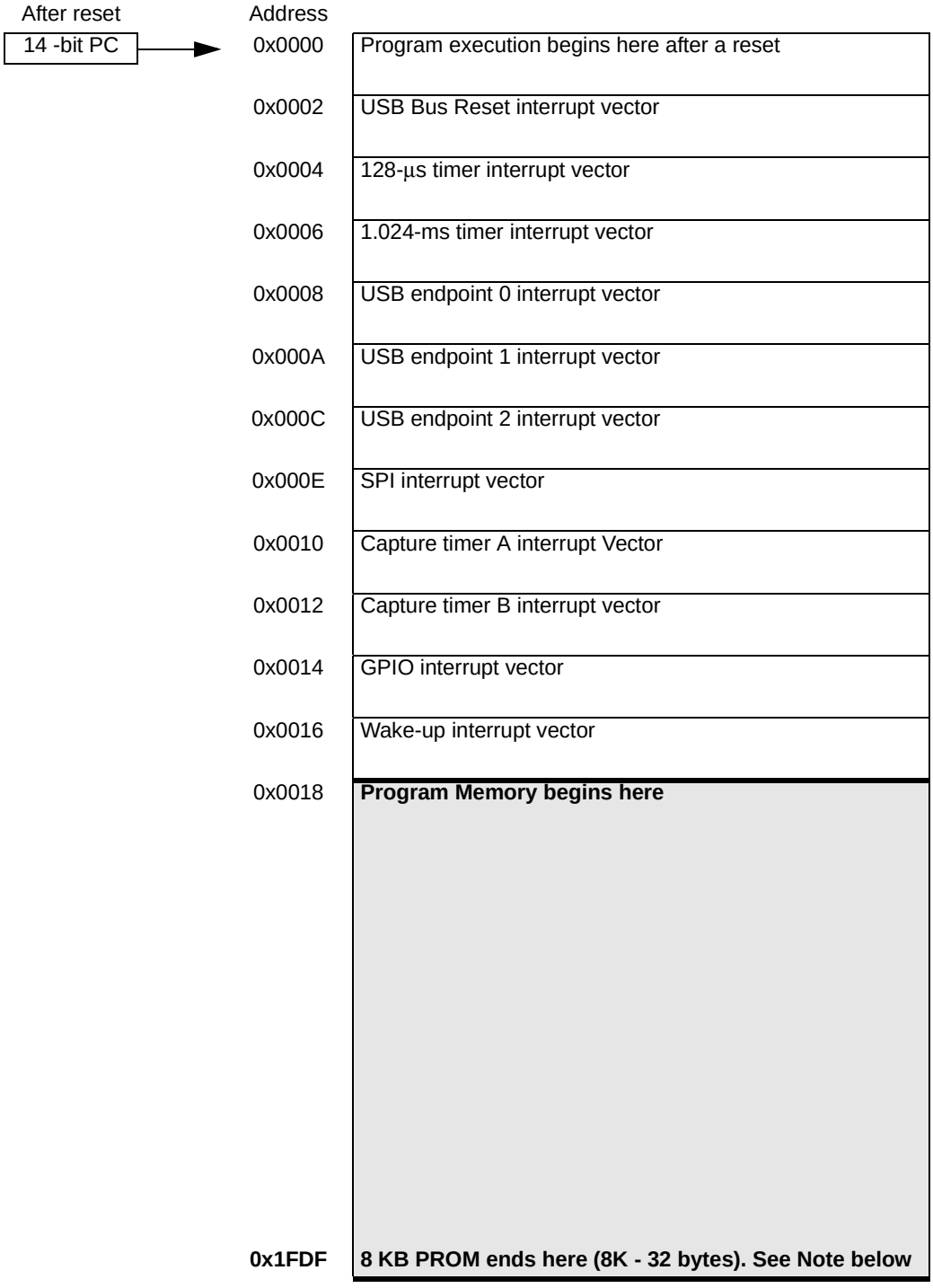


Figure 8-1. Program Memory Space with Interrupt Vector Table

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

8.2 Data Memory Organization

The CY7C637xx 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.

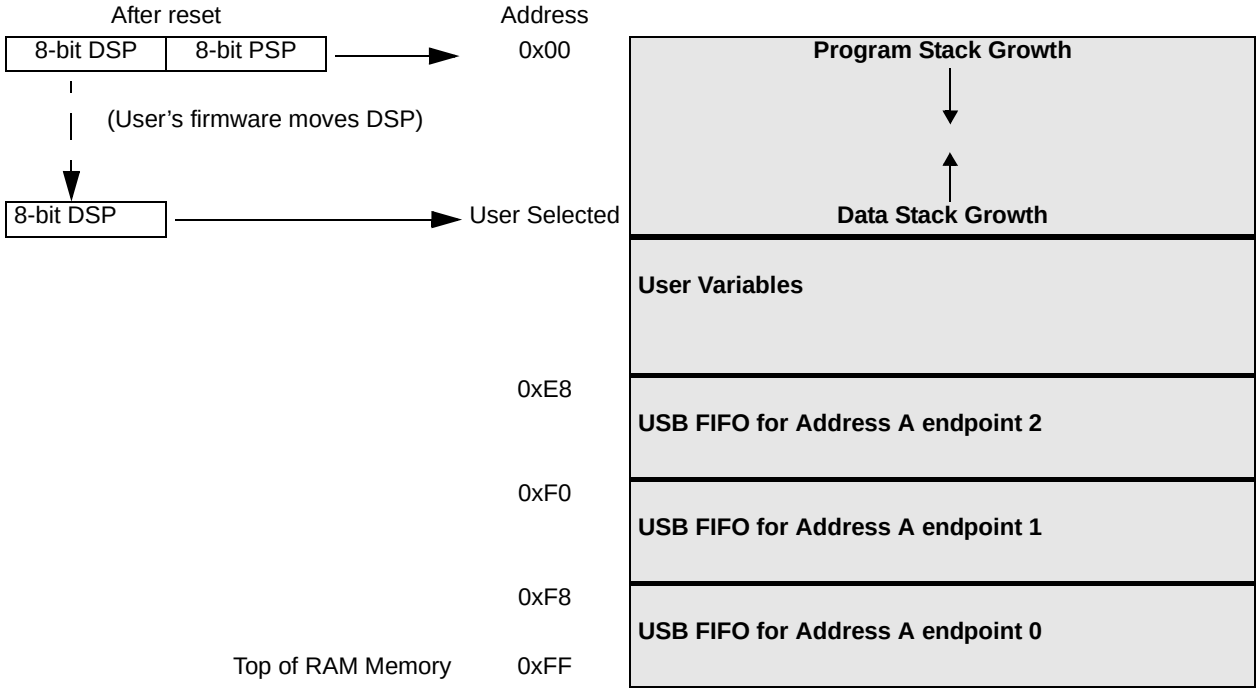


Figure 8-2. Data Memory Organization

8.3 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 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 with IOWX (e.g., IOWX 0h) means the I/O port is selected solely by the contents of X.

Note: All bits of all registers are cleared to all zeros on reset, except the Processor Status and Control Register (*Figure 20-1*). *All registers not listed are reserved, and should never be written by firmware. All bits marked as reserved should always be written as 0 and be treated as undefined by reads.*

Table 8-1. I/O Register Summary

Register Name	I/O Address	Read/Write	Function	Fig.
Port 0 Data	0x00	R/W	GPIO Port 0	12-2
Port 1 Data	0x01	R/W	GPIO Port 1	12-3
Port 2 Data	0x02	R	Auxiliary input register for D+, D–, VREG, XTALIN	12-8
Port 0 Interrupt Enable	0x04	W	Interrupt enable for pins in Port 0	21-4
Port 1 Interrupt Enable	0x05	W	Interrupt enable for pins in Port 1	21-5
Port 0 Interrupt Polarity	0x06	W	Interrupt polarity for pins in Port 0	21-6
Port 1 Interrupt Polarity	0x07	W	Interrupt polarity for pins in Port 1	21-7
Port 0 Mode0	0x0A	W	Controls output configuration for Port 0	12-4
Port 0 Mode1	0x0B	W		12-5
Port 1 Mode0	0x0C	W	Controls output configuration for Port 1	12-6
Port 1 Mode1	0x0D	W		12-7

Table 8-1. I/O Register Summary (continued)

Register Name	I/O Address	Read/Write	Function	Fig.
USB Device Address	0x10	R/W	USB Device Address register	14-1
EP0 Counter Register	0x11	R/W	USB Endpoint 0 counter register	14-4
EP0 Mode Register	0x12	R/W	USB Endpoint 0 configuration register	14-2
EP1 Counter Register	0x13	R/W	USB Endpoint 1 counter register	14-4
EP1 Mode Register	0x14	R/W	USB Endpoint 1 configuration register	14-3
EP2 Counter Register	0x15	R/W	USB Endpoint 2 counter register	14-4
EP2 Mode Register	0x16	R/W	USB Endpoint 2 configuration register	14-3
USB Status & Control	0x1F	R/W	USB status and control register	13-1
Global Interrupt Enable	0x20	R/W	Global interrupt enable register	21-1
Endpoint Interrupt Enable	0x21	R/W	USB endpoint interrupt enables	21-2
Timer (LSB)	0x24	R	Lower 8 bits of free-running timer (1 MHz)	18-1
Timer (MSB)	0x25	R	Upper 4 bits of free-running timer	18-2
WDR Clear	0x26	W	Watchdog Reset clear	-
Capture Timer A Rising	0x40	R	Rising edge Capture Timer A data register	19-2
Capture Timer A Falling	0x41	R	Falling edge Capture Timer A data register	19-3
Capture Timer B Rising	0x42	R	Rising edge Capture Timer B data register	19-4
Capture Timer B Falling	0x43	R	Falling edge Capture Timer B data register	19-5
Capture TImer Configuration	0x44	R/W	Capture Timer configuration register	19-7
Capture Timer Status	0x45	R	Capture Timer status register	19-6
SPI Data	0x60	R/W	SPI read and write data register	17-2
SPI Control	0x61	R/W	SPI status and control register	17-3
Clock Configuration	0xF8	R/W	Internal / External Clock configuration register	9-2
Processor Status & Control	0xFF	R/W	Processor status and control	20-1

9.0 Clocking

The chip can be clocked from either the internal on-chip clock, or from an oscillator based on an external resonator/crystal, as shown in *Figure 9-1*. No additional capacitance is included on chip at the XTALIN/OUT pins. Operation is controlled by the Clock Configuration Register, *Figure 9-2*.

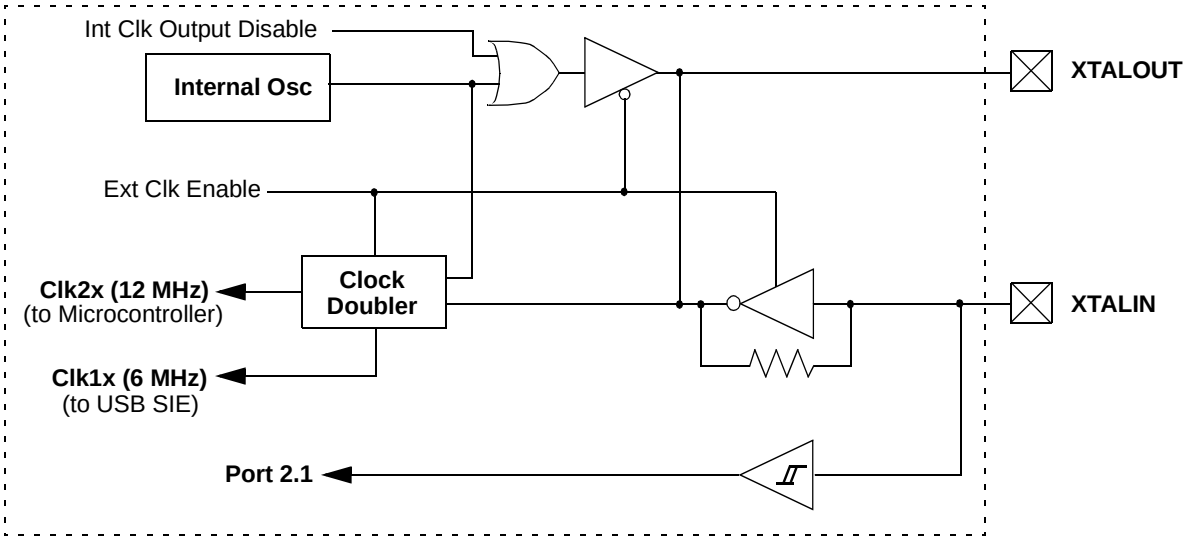


Figure 9-1. Clock Oscillator On-chip Circuit

Bit #	7	6	5	4	3	2	1	0
Bit Name	Ext. Clock Resume Delay	Wake-up Timer Adjust Bit [2:0]			Low-voltage Reset Disable	Precision USB Clocking Enable	Internal Clock Output Disable	External Oscillator Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 9-2. Clock Configuration Register (Address 0xF8)

Bit 7: Ext. Clock Resume Delay

External Clock Resume Delay bit selects the delay time when switching to the external oscillator from the internal oscillator mode, or when waking from suspend mode with the external oscillator enabled.

1 = 4 ms delay.

0 = 128 µs delay.

The delay gives the oscillator time to start up. The shorter time is adequate for operation with ceramic resonators, while the longer time is preferred for start-up with a crystal. (These times **do not include** an initial oscillator start-up time which depends on the resonating element. This time is typically 50–100 µs for ceramic resonators and 1–10 ms for crystals). Note that this bit only selects the delay time for the external clock mode. When waking from suspend mode with the internal oscillator (Bit 0 is LOW), the delay time is only 8 µs in addition to a delay of approximately 1 µs for the oscillator to start.

Bit [6:4]: Wake-up Timer Adjust Bit [2:0]

The Wake-up Timer Adjust Bits are used to adjust the Wake-up timer period.

If the Wake-up interrupt is enabled in the Global Interrupt Enable Register, the microcontroller will generate wake-up interrupts periodically. The frequency of these periodical wake-up interrupts is adjusted by setting the Wake-up Timer Adjust Bit [2:0], as described in Section 11.2. One common use of the wake-up interrupts is to generate periodical wake-up events during suspend mode to check for changes, such as looking for movement in a mouse, while maintaining a low average power.

Bit 3: Low-voltage Reset Disable

When V_{CC} drops below V_{LVR} (see Section 25.0 for the value of V_{LVR}) and the Low-voltage Reset circuit is enabled, the microcontroller enters a partial suspend state for a period of t_{START} (see Section 26.0 for the value of t_{START}). Program execution begins from address 0x0000 after this t_{START} delay period. This provides time for V_{CC} to stabilize

before the part executes code. See Section 10.1 for more details.

1 = Disables the LVR circuit.

0 = Enables the LVR circuit.

Bit 2: Precision USB Clocking Enable

The Precision USB Clocking Enable only affects operation in internal oscillator mode. **In that mode, this bit must be set to 1 to cause the internal clock to automatically precisely tune to USB timing requirements (6 MHz ±1.5%)**. The frequency may have a looser initial tolerance at power-up, but all USB transmissions from the chip will meet the USB specification.

1 = Enabled. The internal clock accuracy is **6 MHz ±1.5%** after USB traffic is received**.**

0 = Disabled. The internal clock accuracy is 6 MHz ±5%.

Bit 1: Internal Clock Output Disable

The Internal Clock Output Disable is used to keep the internal clock from driving out to the XTALOUT pin. This bit has no effect in the external oscillator mode.

1 = Disable internal clock output. XTALOUT pin will drive HIGH.

0 = Enable the internal clock output. The internal clock is driven out to the XTALOUT pin.

Bit 0: External Oscillator Enable

At power-up, the chip operates from the internal clock by default. Setting the External Oscillator Enable bit HIGH disables the internal clock, and halts the part while the external resonator/crystal oscillator is started. Clearing this bit has no immediate effect, although the state of this bit is used when waking out of suspend mode to select between internal and external clock. In internal clock mode, XTALIN pin will be configured as an input with a weak pull-down and can be used as a GPIO input (P2.1).

1 = Enable the external oscillator. The clock is switched to external clock mode, as described in Section 9.1.

0 = Enable the internal oscillator.

9.1 Internal/External Oscillator Operation

The internal oscillator provides an operating clock, factory set to a nominal frequency of 6 MHz. This clock requires no external components. At power-up, the chip operates from the internal clock. In this mode, the internal clock is buffered and driven to the XTALOUT pin by default, and the state of the XTALIN pin can be read at Port 2.1. While the internal clock is enabled, its output can be disabled at the XTALOUT pin by setting the Internal Clock Output Disable bit of the Clock Configuration Register.

Setting the External Oscillator Enable bit of the Clock Configuration Register HIGH disables the internal clock, and halts the part while the external resonator/crystal oscillator is started. The steps involved in switching from Internal to External Clock mode are as follows:

1. At reset, chip begins operation using the internal clock.
2. Firmware sets Bit 0 of the Clock Configuration Register. For example,

```
mov A, 1h      ; Set Bit 0 HIGH (External Oscillator Enable bit). Bit 7 cleared gives faster start-up
iowr F8h       ; Write to Clock Configuration Register
```

3. Internal clocking is halted, the internal oscillator is disabled, and the external clock oscillator is enabled.
4. After the external clock becomes stable, chip clocks are re-enabled using the external clock signal. (Note that the time for the external clock to become stable depends on the external resonating device; see next section.)
5. After an additional delay the CPU is released to run. This delay depends on the state of the Ext. Clock Resume Delay bit of the Clock Configuration Register. The time is 128 µs if the bit is 0, or 4 ms if the bit is 1.
6. Once the chip has been set to external oscillator, it can only return to internal clock when waking from suspend mode. Clearing bit 0 of the Clock Configuration Register will not re-enable internal clock mode until suspend mode is entered. See Section 11.0 for more details on suspend mode operation.

If the Internal Clock is enabled, the XTALIN pin can serve as a general purpose input, and its state can be read at Port 2, Bit 1 (P2.1). Refer to *Figure 12-8* for the Port 2 Data Register. In this mode, there is a weak pull-down at the XTALIN pin. This input cannot provide an interrupt source to the CPU.

9.2 External Oscillator

The user can connect a low-cost ceramic resonator or an external oscillator to the XTALIN/XTALOUT pins to provide a precise reference frequency for the chip clock, as shown in *Figure 9-1*. The external components required are a ceramic resonator or crystal and any associated capacitors. To run from the external resonator, the External Oscillator Enable bit of the Clock Configuration Register must be set to 1, as explained in the previous section.

Start-up times for the external oscillator depend on the resonating device. Ceramic resonator based oscillators typically start in less than 100 µs, while crystal based oscillators take longer, typically 1 to 10 ms. Board capacitance should be minimized on the XTALIN and XTALOUT pins by keeping the traces as short as possible.

An external 6-MHz clock can be applied to the XTALIN pin if the XTALOUT pin is left open.

10.0 Reset

The USB Controller supports three types of resets. The effects of the reset are listed below. The reset types are:

1. Low-voltage Reset (LVR)
2. Brown Out Reset (BOR)
3. Watchdog Reset (WDR)

The occurrence of a reset is recorded in the Processor Status and Control Register (*Figure 20-1*). Bits 4 (Low-voltage or Brown-out Reset bit) and 6 (Watchdog Reset bit) are used to record the occurrence of LVR/BOR 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 LVR, BOR, 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. Attempting to execute either a RET or RETI in the reset handler will cause unpredictable execution results.

The following events take place on reset. More details on the various resets are given in the following sections.

1. All registers are reset to their default states (all bits cleared, except in Processor Status and Control Register).
2. GPIO and USB pins are set to high-impedance state.
3. The VREG pin is set to high-impedance state.
4. Interrupts are disabled.
5. USB operation is disabled and must be enabled by firmware if desired, as explained in Section 14.1.
6. For a BOR or LVR, the external oscillator is disabled and Internal Clock mode is activated, followed by a time-out period t_{START} for V_{CC} to stabilize. A WDR does not change the clock mode, and there is no delay for V_{CC} stabilization on a WDR. Note that the External Oscillator Enable (Bit 0, *Figure 9-2*) will be cleared by a WDR, but it does not take effect until suspend mode is entered.
7. The Program Stack Pointer (PSP) and Data Stack Pointer (DSP) reset to address 0x00. Firmware should move the DSP for USB applications, as explained in Section 6.5.
8. Program execution begins at address 0x0000 after the appropriate time-out period.

10.1 Low-voltage Reset (LVR)

When V_{CC} is first applied to the chip, the internal oscillator is started and the Low-voltage Reset is initially enabled by default. At the point where V_{CC} has risen above V_{LVR} (see Section 25.0 for the value of V_{LVR}), an internal counter starts counting for a period of t_{START} (see Section 26.0 for the value of t_{START}). During this t_{START} time, the microcontroller enters a partial suspend state to wait for V_{CC} to stabilize before it begins executing code from address 0x0000.

As long as the LVR circuit is enabled, this reset sequence repeats whenever the V_{CC} pin voltage drops below V_{LVR}. The LVR can be disabled by firmware by setting the Low-voltage Reset Disable bit in the Clock Configuration Register (*Figure 9-2*). In addition, the LVR is automatically disabled in suspend mode to save power. If the LVR was enabled before entering suspend mode, it becomes active again once the suspend mode ends.

When LVR is disabled during normal operation (i.e., by writing '0' to the Low-voltage Reset Disable bit in the Clock Configuration Register), the chip may enter an unknown state if V_{CC} drops below V_{LVR}. Therefore, LVR should be enabled at all times during normal operation. If LVR is disabled (i.e., by firmware or during suspend mode), a secondary low-voltage monitor, BOR, becomes active, as described in the next section. The LVR/BOR Reset bit of the Processor Status and Control Register (*Figure 20-1*), is set to '1' if either a LVR or BOR has occurred.

10.2 Brown Out Reset (BOR)

The Brown Out Reset (BOR) circuit is always active and behaves like the POR. BOR is asserted whenever the V_{CC} voltage to the device is below an internally defined trip voltage of approximately 2.5V. The BOR re-enables LVR. That is, once V_{CC} drops and trips BOR, the part remains in reset until V_{CC} rises above V_{LVR}. At that point, the t_{START} delay occurs before normal operation resumes, and the microcontroller starts executing code from address 0x00 after the t_{START} delay.

In suspend mode, only the BOR detection is active, giving a reset if V_{CC} drops below approximately 2.5V. Since the device is suspended and code is not executing, this lower reset voltage is safe for retaining the state of all registers and memory. Note that in suspend mode, LVR is disabled as discussed in Section 10.1.

10.3 Watchdog Reset (WDR)

The Watchdog Timer Reset (WDR) occurs when the internal Watchdog timer rolls over. Writing any value to the write-only Watchdog Reset 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} (see *Figure 10-1*) of the last clear. Bit 6 (Watchdog Reset bit) of the Processor Status and Control Register is set to record this event (see Section 20.0 for more details). A Watchdog Timer Reset typically lasts for 2–4 ms, after which the microcontroller begins execution at ROM address 0x0000.

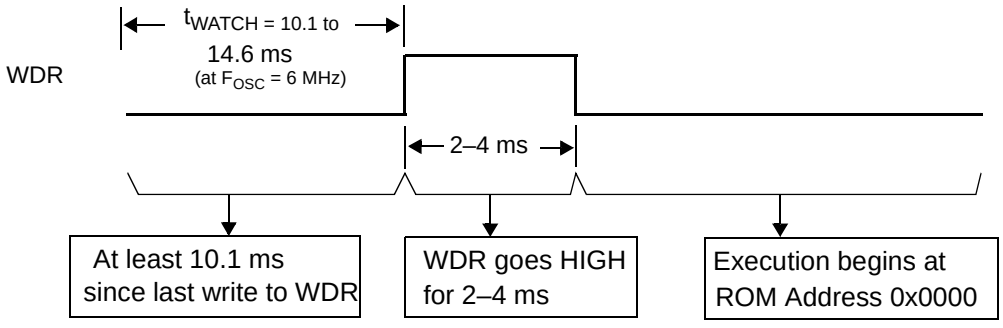


Figure 10-1. Watchdog Reset (WDR, Address 0x26)

11.0 Suspend Mode

The CY7C637xx parts support a versatile low-power suspend mode. In suspend mode, only an enabled interrupt or a LOW state on the D–/SDATA pin will wake the part. Two options are available. For lowest power, all internal circuits can be disabled, so only an external event will resume operation. Alternatively, a low-power internal wake-up timer can be used to trigger the wake-up interrupt. This timer is described in Section 11.2, and can be used to periodically poll the system to check for changes, such as looking for movement in a mouse, while maintaining a low average power.

The CY7C637xx is placed into a low-power state by setting the Suspend bit of the Processor Status and Control Register (*Figure 20-1*). All logic blocks in the device are turned off except the GPIO interrupt logic, the D–/SDATA pin input receiver, and (optionally) the wake-up timer. The clock oscillators, as well as the free-running and Watchdog timers are shut down. Only the occurrence of an enabled GPIO interrupt, wake-up interrupt, SPI slave interrupt, or a LOW state on the D–/SDATA pin will wake the part from suspend (D– LOW indicates non-idle USB activity). Once one of these resuming conditions occurs, clocks will be restarted and the device returns to full operation after the oscillator is stable and the selected delay period expires. This delay period is determined by selection of internal vs. external clock, and by the state of the Ext. Clock Resume Delay as explained in Section 9.0.

In suspend mode, any enabled and pending interrupt will wake the part up. The state of the Interrupt Enable Sense bit (Bit 2, *Figure 20-1*) does not have any effect. As a result, any interrupts not intended for waking from suspend should be disabled through the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register (Section 21.0).

If a resuming condition exists when the suspend bit is set, the part will still go into suspend and then awake after the appropriate delay time. The Run bit in the Processor Status and Control Register must be set for the part to resume out of suspend.

Once the clock is stable and the delay time has expired, the microcontroller will execute the instruction following the I/O write that placed the device into suspend mode before servicing any interrupt requests.

To achieve the lowest possible current during suspend mode, all I/O should be held at either V_{CC} or ground. In addition, the GPIO *bit* interrupts (*Figure 21-4* and *Figure 21-5*) should be disabled for any pins that are not being used for a wake-up interrupt. This should be done even if the main GPIO Interrupt Enable (*Figure 21-1*) is off.

Typical code for entering suspend is shown below:

```
...          ; All GPIO set to low-power state (no floating
               pins, and bit interrupts disabled unless
               using for wake-up)
...          ; Enable GPIO and/or wake-up timer
               interrupts if desired for wake-up
...          ; Select clock mode for wake-up (see
               Section 11.1)
mov a, 09h   ; Set suspend and run bits
iowr FFh     ; Write to Status and Control Register –
               Enter suspend, wait for GPIO/wake-up
               interrupt or USB activity
nop          ; This executes before any ISR
...          ; Remaining code for exiting suspend
               routine
```

11.1 Clocking Mode on Wake-up from Suspend

When exiting suspend on a wake-up event, the device can be configured to run in either Internal or External Clock mode. The mode is selected by the state of the External Oscillator Enable bit in the Clock Configuration Register (*Figure 9-2*). Using the Internal Clock saves the external oscillator start-up time and keeps that oscillator off for additional power savings. The external oscillator mode can be activated when desired, similar to operation at power-up.

The sequence of events for these modes is as follows:

Wake in Internal Clock Mode:

1. Before entering suspend, clear bit 0 of the Clock Configuration Register. This selects Internal clock mode after suspend.
2. Enter suspend mode by setting the suspend bit of the Processor Status and Control Register.
3. After a wake-up event, the internal clock starts immediately (within 2 µs).
4. A time-out period of 8 µs passes, and then firmware execution begins.
5. At some later point, to activate External Clock mode, set bit 0 of the Clock Configuration Register. This halts the internal clocks while the external clock becomes stable. After an additional time-out (128 µs or 4 ms, see Section 9.0), firmware execution resumes.

Wake in External Clock Mode:

1. Before entering suspend, the external clock must be selected by setting bit 0 of the Clock Configuration Register. Make sure this bit is still set when suspend mode is entered. This selects External clock mode after suspend.
2. Enter suspend mode by setting the suspend bit of the Processor Status and Control Register.
3. After a wake-up event, the external oscillator is started. The clock is monitored for stability (this takes approximately 50–100 µs with a ceramic resonator).
4. After an additional time-out period (128 µs or 4 ms, see Section 9.0), firmware execution resumes.

11.2 Wake-up Timer

The wake-up timer runs whenever the wake-up interrupt is enabled, and is turned off whenever that interrupt is disabled. Operation is independent of whether the device is in suspend mode or if the global interrupt bit is enabled. Only the Wake-up Timer Interrupt Enable bit (*Figure 21-1*) controls the wake-up timer.

Once this timer is activated, it will give interrupts after its time-out period (see below). These interrupts continue periodically until the interrupt is disabled. Whenever the interrupt is disabled, the wake-up timer is reset, so that a subsequent enable always results in a full wake-up time.

The wake-up timer can be adjusted by the user through the Wake-up Timer Adjust bits in the Clock Configuration Register (*Figure 9-2*). These bits clear on reset. In addition to allowing the user to select a range for the wake-up time, a firmware algorithm can be used to tune out initial process and operating condition variations in this wake-up time. This can be done by timing the wake-up interrupt time with the accurate 1.024-ms timer interrupt, and adjusting the Timer Adjust bits accordingly to approximate the desired wake-up time.

Table 11-1. Wake-up Timer Adjust Settings

Adjust Bits [2:0] (Bits [6:4] in *Figure 9-2*)	Wake-up Time
000 (reset state)	1 * t_{WAKE}
001	2 * t_{WAKE}
010	4 * t_{WAKE}
011	8 * t_{WAKE}
100	16 * t_{WAKE}
101	32 * t_{WAKE}
110	64 * t_{WAKE}
111	128 * t_{WAKE}
See Section 26.0 for the value of t_{WAKE}	

12.0 General Purpose I/O Ports

Ports 0 and 1 provide up to 16 versatile GPIO pins that can be read or written (the number of pins depends on package type). *Figure 12-1* shows a diagram of a GPIO port pin.

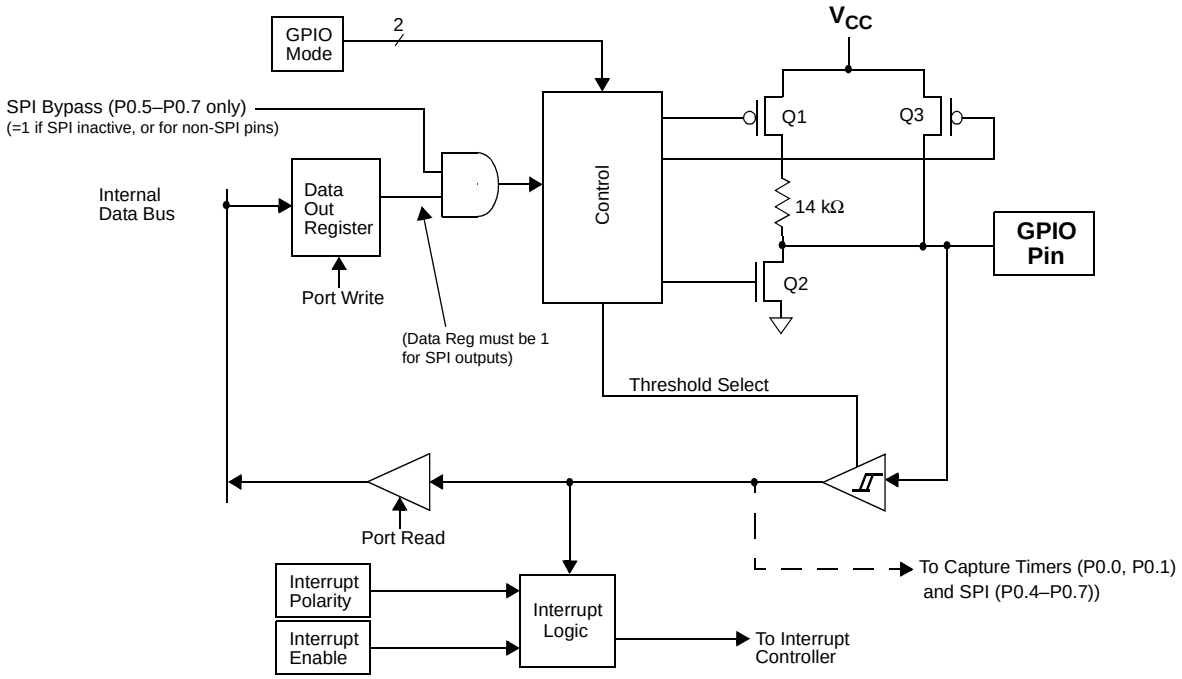


Figure 12-1. Block Diagram of GPIO Port (one pin shown)

Port 0 is an 8-bit port; Port 1 contains either 2 bits, P1.1–P1.0 in the CY7C63723, or all 8 bits, P1.7–P1.0 in the CY7C63743 parts. Each bit can also be selected as an interrupt source for the microcontroller, as explained in Section 21.0.

The data for each GPIO pin is accessible through the Port Data register. Writes to the Port Data register store outgoing data state for the port pins, while reads from the Port Data register return the actual logic value on the port pins, not the Port Data register contents.

Each GPIO pin is configured independently. The driving state of each GPIO pin is determined by the value written to the pin's Data Register and by two associated pin's Mode0 and Mode1 bits.

The Port 0 Data Register is shown in *Figure 12-2*, and the Port 1 Data Register is shown in *Figure 12-3*. The Mode0 and Mode1 bits for the two GPIO ports are given in *Figure 12-4* through *Figure 12-7*.

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

Figure 12-2. Port 0 Data (Address 0x00)

Bit [7:0]: P0[7:0]

1 = Port Pin is logic HIGH

0 = Port Pin is logic LOW

Bit #	7	6	5	4	3	2	1	0
Bit Name	P1							
Notes	Pins 7:2 only in CY7C63743						Pins 1:0 in all parts	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 12-3. Port 1 Data (Address 0x01)

Bit [7:0]: P1[7:0]

1 = Port Pin is logic HIGH

0 = Port Pin is logic LOW

Bit #	7	6	5	4	3	2	1	0
Bit Name	P0[7:0] Mode0							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 12-4. GPIO Port 0 Mode0 Register (Address 0x0A)

Bit [7:0]: P0[7:0] Mode 0

1 = Port 0 Mode 0 is logic HIGH

0 = Port 0 Mode 0 is logic LOW

Bit #	7	6	5	4	3	2	1	0
Bit Name	P0[7:0] Mode1							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 12-5. GPIO Port 0 Mode1 Register (Address 0x0B)

Bit [7:0]: P0[7:0] Mode 1

1 = Port Pin Mode 1 is logic HIGH

0 = Port Pin Mode 1 is logic LOW

Bit #	7	6	5	4	3	2	1	0
Bit Name	P1[7:0] Mode0							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 12-6. GPIO Port 1 Mode0 Register (Address 0x0C)

Bit [7:0]: P1[7:0] Mode 0

1 = Port Pin Mode 0 is logic HIGH

0 = Port Pin Mode 0 is logic LOW

Bit #	7	6	5	4	3	2	1	0
Bit Name	P1[7:0] Mode1							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 12-7. GPIO Port 1 Mode1 Register (Address 0x0D)

Bit [7:0]: P1[7:0] Mode 1

1 = Port Pin Mode 1 is logic HIGH

0 = Port Pin Mode 1 is logic LOW

Each pin can be independently configured as high-impedance inputs, inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs with selectable drive strengths.

The driving state of each GPIO pin is determined by the value written to the pin's Data Register and by its associated Mode0 and Mode1 bits. *Table 12-1* lists the configuration states based on these bits. The GPIO ports default on reset to all Data and Mode Registers cleared, so the pins are all in a high-impedance state. The available GPIO output drive strength are:

- **Hi-Z Mode** (Mode1 = 0 and Mode0 = 0)

  Q1, Q2, and Q3 (*Figure 12-1*) are OFF. The GPIO pin is not driven internally. Performing a read from the Port Data Register return the actual logic value on the port pins.
- **Low Sink Mode** (Mode1 = 1, Mode0 = 0, and the pin's Data Register = 0)

  Q1 and Q3 are OFF. Q2 is ON. The GPIO pin is capable of sinking 2 mA of current.
- **Medium Sink Mode** (Mode1 = 0, Mode0 = 1, and the pin's Data Register = 0)

  Q1 and Q3 are OFF. Q2 is ON. The GPIO pin is capable of sinking 8 mA of current.
- **High Sink Mode** (Mode1 = 1, Mode0 = 1, and the pin's Data Register = 0)

  Q1 and Q3 are OFF. Q2 is ON. The GPIO pin is capable of sinking 50 mA of current.
- **High Drive Mode** (Mode1 = 0 or 1, Mode0 = 1, and the pin's Data Register = 1)

  Q1 and Q2 are OFF. Q3 is ON. The GPIO pin is capable of sourcing 2 mA of current.
- **Resistive Mode** (Mode1 = 1, Mode0 = 0, and the pin's Data Register = 1)

  Q2 and Q3 are OFF. Q1 is ON. The GPIO pin is pulled up with an internal 14-kΩ resistor.

Note that open drain mode can be achieved by fixing the Data and Mode1 Registers LOW, and switching the Mode0 register.

Input thresholds are CMOS, or TTL as shown in the table (See Section 25.0 for the input threshold voltage in TTL or CMOS modes). Both input modes include hysteresis to minimize noise sensitivity. In suspend mode, if a pin is used for a wake-up interrupt using an external R-C circuit, CMOS mode is preferred for lowest power.

Table 12-1. Ports 0 and 1 Output Control Truth Table

Data Register	Mode1	Mode0	Output Drive Strength	Input Threshold
0	0	0	Hi-Z	CMOS
1			Hi-Z	TTL
0	0	1	Medium (8 mA) Sink	CMOS
1			High Drive	CMOS
0	1	0	Low (2 mA) Sink	CMOS
1			Resistive	CMOS
0	1	1	High (50 mA) Sink	CMOS
1			High Drive	CMOS

12.1 Auxiliary Input Port

Port 2 serves as an auxiliary input port as shown in *Figure 12-8*. The Port 2 inputs all have TTL input thresholds.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved		D+ (SCLK) State	D– (SDATA) State	Reserved		P2.1 (Internal Clock Mode Only)	P2.0 VREG Pin State
Read/ Write	-	-	R	R	-	-	R	R
Reset	0	0	0	0	0	0	0	0

Figure 12-8. Port 2 Data Register (Address 0x02)

Bit [7:6]: Reserved

Bit [5:4]: D+ (SCLK) and D– (SDATA) States

The state of the D+ and D– pins can be read at Port 2 Data Register. Performing a read from the port pins returns their logic values.

1 = Port Pin is logic HIGH

0 = Port Pin is logic LOW

Bit [3:2]: Reserved

Bit 1: P2.1 (Internal Clock Mode Only)

In the Internal Clock mode, the XTALIN pin can serve as a general purpose input, and its state can be read at Port 2, Bit 1 (P2.1). See Section 9.1 for more details.

1 = Port Pin is logic HIGH

0 = Port Pin is logic LOW

Bit 0: P2.0/VREG Pin State

In PS/2 mode, the VREG pin can be used as an input and its state can be read at port P2.0. Section 15.0 for more details.

1 = Port Pin is logic HIGH

0 = Port Pin is logic LOW

13.0 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:

- Translate the encoded received data and format the data to be transmitted on the bus.
- CRC checking and generation. Flag the microcontroller if errors exist during transmission.
- Address checking. Ignore the transactions not addressed to the device.
- Send appropriate ACK/NAK/STALL handshakes.
- Token type identification (SETUP, IN, or OUT). Set the appropriate token bit once a valid token is received.
- Place valid received data in the appropriate endpoint FIFOs.
- Send and update the data toggle bit (Data1/0).
- Bit stuffing/unstuffing.

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

- Coordinate enumeration by decoding USB device requests.
- Fill and empty the FIFOs.
- Suspend/Resume coordination.
- Verify and select Data toggle values.

13.1 USB Enumeration

A typical USB enumeration sequence is shown below. In this description, ‘Firmware’ refers to embedded firmware in the CY7C637xx controller.

1. The host computer sends a SETUP packet followed by a DATA packet to USB address 0 requesting the Device descriptor.
2. Firmware decodes the request and retrieves its Device descriptor from the program memory tables.
3. The host computer performs a control read sequence and Firmware responds by sending the Device descriptor over the USB bus, via the on-chip FIFO.
4. After receiving the descriptor, the host sends a SETUP packet followed by a DATA packet to address 0 assigning a new USB address to the device.
5. Firmware stores the new address in its USB Device Address Register after the no-data control sequence completes.
6. The host sends a request for the Device descriptor using the new USB address.
7. Firmware decodes the request and retrieves the Device descriptor from program memory tables.
8. The host performs a control read sequence and Firmware responds by sending its Device descriptor over the USB bus.
9. The host generates control reads from the device to request the Configuration and Report descriptors.
10. Once the device receives a Set Configuration request, its functions may now be used.
11. Firmware should take appropriate action for Endpoint 1 and/or 2 transactions, which may occur from this point.

13.2 USB Port Status and Control

USB status and control is regulated by the USB Status and Control Register as shown in *Figure 13-1*.

Bit #	7	6	5	4	3	2:0		
Bit Name	PS/2 Pull-up Enable	VREG Enable	USB Reset-PS/2 Activity Interrupt Mode	Reserved	USB Bus Activity	D+/D– Forcing Bit		
Read/ Write	R/W	R/W	R/W	-	R/W	R/W		
Reset	0	0	0	0	0	0	0	0

Figure 13-1. USB Status and Control Register (Address 0x1F)

Bit 7: PS/2 Pull-up Enable

This bit is used to enable the internal PS/2 pull-up resistors on the SDATA and SCLK pins. Normally the output high level on these pins is V_{CC}, but note that the output will be clamped to approximately 1 Volt above V_{REG} if the VREG Enable bit is set, or if the Device Address is enabled (bit 7 of the USB Device Address Register, *Figure 14-1*).

1 = Enable PS/2 Pull-up resistors. The SDATA and SCLK pins are pulled up internally to V_{CC} with two resistors of approximately 5 kΩ (see Section 25.0 for the value of R_{PS2}).

0 = Disable PS/2 Pull-up resistors.

Bit 6: V_{REG} Enable

A 3.3V voltage regulator is integrated on chip to provide a voltage source for a 1.5-kΩ pull-up resistor connected to the D– pin as required by the USB Specification. Note that the VREG output has an internal series resistance of approximately 200Ω, the external pull-up resistor required is approximately 1.3-kΩ (see *Figure 16-1*).

1 = Enable the 3.3V output voltage on the VREG pin.

0 = Disable. The VREG pin can be configured as an input.

Bit 5: USB-PS/2 Interrupt Select

This bit allows the user to select whether an USB bus reset interrupt or a PS/2 activity interrupt will be generated when the interrupt conditions are detected.

1 = PS/2 interrupt mode. A PS/2 activity interrupt will occur if the SDATA pin is continuously LOW for 128 to 256 µs.

0 = USB interrupt mode (default state). In this mode, a USB bus reset interrupt will occur if the single ended zero (SE0, D– and D+ are LOW) exists for 128 to 256 µs.

See Section 21.3 for more details.

Bit 4: Reserved. Must be written as a '0'.

Bit 3: USB Bus Activity

The Bus Activity bit is a "sticky" bit that detects any non-idle USB event has occurred on the USB bus. Once set to HIGH by the SIE to indicate the bus activity, this bit retains its logical HIGH value until firmware clears it. Writing a '0' to this bit clears it; writing a '1' preserves its value. The user firmware should check and clear this bit periodically to detect any loss of bus activity. 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.

1 = There has been bus activity since the last time this bit was cleared. This bit is set by the SIE.

0 = No bus activity since last time this bit was cleared (by firmware).

Bit [2:0]: D+/D– Forcing Bit [2:0]

Forcing bits allow firmware to directly drive the D+ and D– pins, as shown in *Table 13-1*. Outputs are driven with controlled edge rates in these modes for low EMI. For forcing the D+ and D– pins in USB mode, D+/D– Forcing Bit 2 should be 0. Setting D+/D– Forcing Bit 2 to '1' puts both pins in an open-drain mode, preferred for applications such as PS/2 or LED driving.

Table 13-1. Control Modes to Force D+/D– Outputs

D+/D– Forcing Bit [2:0]	Control Action	Application
000	Not forcing (SIE controls driver)	Any Mode
001	Force K (D+ HIGH, D– LOW)	USB Mode
010	Force J (D+ LOW, D– HIGH)	
011	Force SE0 (D– LOW, D+ LOW)	
100	Force D– LOW, D+ LOW	PS/2 Mode[2]
101	Force D– LOW, D+ HiZ	
110	Force D– HiZ, D+ LOW	
111	Force D– HiZ, D+ HiZ	

Note:
2. For PS/2 operation, the D+/D– Forcing Bit [2:0] = 111b mode must be set initially (one time only) before using the other PS/2 force modes.

14.0 USB Device

The CY7C637xx supports one USB Device Address with three endpoints: EP0, EP1, and EP2.

14.1 USB Address Register

The USB Device Address Register contains a 7-bit USB address and one bit to enable USB communication. This register is cleared during a reset, setting the USB device address to zero and marking this address as disabled. *Figure 14-1* shows the format of the USB Address Register.

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

Figure 14-1. USB Device Address Register (Address 0x10)

In either USB or PS/2 mode, this register is cleared by both hardware resets and the USB bus reset. See Section 21.3 for more information on the USB Bus Reset – PS/2 interrupt.

Bit 7: Device Address Enable

This bit must be enabled by firmware before the serial interface engine (SIE) will respond to USB traffic at the address specified in Bit [6:0].

1 = Enable USB device address.

0 = Disable USB device address.

Bit [6:0]: Device Address Bit [6:0]

These bits must be set by firmware during the USB enumeration process (i.e., SetAddress) to the non-zero address assigned by the USB host.

14.2 USB Control Endpoint

All USB devices are required to have an endpoint number 0 (EP0) that is used to initialize and control the USB device. EP0 provides access to the device configuration information and allows generic USB status and control accesses. EP0 is bidirectional as the device can both receive and transmit data. EP0 uses an 8-byte FIFO at SRAM locations 0xF8-0xFF, as shown in Section 8.2.

The EP0 endpoint mode register uses the format shown in *Figure 14-2*.

Bit #	7	6	5	4	3:0
Bit Name	SETUP Received	IN Received	OUT Received	ACKed Transaction	Mode Bit
Read/ Write	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0 0 0 0

Figure 14-2. Endpoint 0 Mode Register (Address 0x12)

The SIE provides a locking feature to prevent firmware from overwriting bits in the USB Endpoint 0 Mode Register. Writes to the register have no effect from the point that Bit[6:0] of the register are updated (by the SIE) until the firmware reads this register. The CPU can unlock this register by reading it.

Because of these hardware-locking features, firmware should perform an read after a write to the USB Endpoint 0 Mode Register and USB Endpoint 0 Count Register (*Figure 14-4*) to verify that the contents have changed as desired, and that the SIE has not updated these values.

Bit [7:4] of this register are cleared by any non-locked write to this register, regardless of the value written.

Bit 7: SETUP Received

1 = A valid SETUP packet has been received. This bit is forced HIGH from the start of the data packet phase of the SETUP transaction until the start of the ACK packet returned by the SIE. The CPU is prevented from clearing this bit during this interval. While this bit is set to ‘1’, the CPU cannot write to the EP0 FIFO. This prevents firmware from overwriting an incoming SETUP transaction before firmware has a chance to read the SETUP data.

0 = No SETUP received. This bit is cleared by any non-locked writes to the register.

Bit 6: IN Received

1 = A valid IN packet has been received. This bit is updated to ‘1’ after the last received packet in an IN transaction. This bit is cleared by any non-locked writes to the register.

0 = No IN received. This bit is cleared by any non-locked writes to the register.

Bit 5: OUT Received

1 = A valid OUT packet has been received. This bit is updated to ‘1’ after the last received packet in an OUT transaction. This bit is cleared by any non-locked writes to the register.

0 = No OUT received. This bit is cleared by any non-locked writes to the register.

Bit 4: ACKed Transaction

The ACKed Transaction bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet.

1 = The transaction completes with an ACK.

0 = The transaction does not complete with an ACK.

Bit [3:0]: Mode Bit[3:0]

The endpoint modes determine how the SIE responds to USB traffic that the host sends to the endpoint. For example, if the endpoint Mode Bits [3:0] are set to 0001 which is NAK IN/OUT mode as shown in *Table 22-1*, the SIE will send NAK handshakes in response to any IN or OUT token sent to this endpoint. In this NAK IN/OUT mode, the SIE will send an ACK handshake when the host sends a SETUP token to this endpoint. The mode encoding is shown in *Table 22-1*. Additional information on the mode bits can be found in *Table 22-2* and *Table 22-3*. These modes give the firmware total control on how to respond to different tokens sent to the endpoints from the host.

In addition, the Mode Bits are automatically changed by the SIE in response to many USB transactions. For example, if the Mode Bit [3:0] are set to 1011 which is ACK OUT-NAK IN mode as shown in *Table 22-1*, the SIE will change the endpoint Mode Bit [3:0] to NAK IN/OUT (0001) mode after issuing an ACK handshake in response to an OUT token. Firmware needs to update the mode for the SIE to respond appropriately.

14.3 USB Non-control Endpoints

The CY7C637xx feature two non-control endpoints, endpoint 1 (EP1) and endpoint 2 (EP2). The EP1 and EP2 Mode Registers do not have the locking mechanism of the EP0 Mode Register. The EP1 and EP2 Mode Registers use the format shown in *Figure 14-3*. EP1 uses an 8-byte FIFO at SRAM locations 0xF0–0xF7, EP2 uses an 8-byte FIFO at SRAM locations 0xE8–0xEF as shown in Section 8.2.

Bit #	7	6	5	4	3	2	1	0
Bit Name	STALL	Reserved		ACKed Transaction	Mode Bit			
Read/ Write	R/W	-	-	R/C	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 14-3. USB Endpoint EP1, EP2 Mode Registers (Addresses 0x14 and 0x16)

Bit 7: STALL

1 = The SIE will stall an OUT packet if the Mode Bits are set to ACK-OUT, and the SIE will stall an IN packet if the mode bits are set to ACK-IN. See Section 22.0 for the available modes.

0 = This bit must be set to LOW for all other modes.

Bit [6:5]: Reserved. Must be written to zero during register writes.

Bit 4: ACKed Transaction

The ACKed transaction bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet.

1 = The transaction completes with an ACK.

0 = The transaction does not complete with an ACK.

Bit [3:0]: Mode Bit [3:0]

The EP1 and EP2 Mode Bits operate in the same manner as the EP0 Mode Bits (see Section 14.2).

14.4 USB Endpoint Counter Registers

There are three Endpoint Counter registers, with identical formats for both control and non-control endpoints. These registers contain byte count information for USB transactions, as well as bits for data packet status. The format of these registers is shown in *Figure 14-4*.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Data Toggle	Data Valid	Reserved		Byte Count			
Read/Write	R/W	R/W	-	-	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 14-4. Endpoint 0,1,2 Counter Registers (Addresses 0x11, 0x13 and 0x15)

Bit 7: Data Toggle

This bit selects the DATA packet's toggle state. For IN transactions, firmware must set this bit to the select the transmitted Data Toggle. For OUT or SETUP transactions, the hardware sets this bit to the state of the received Data Toggle bit.

1 = DATA1

0 = DATA0

Bit 6: Data Valid

This bit is used for OUT and SETUP tokens only. This bit is cleared to ‘0’ if CRC, bitstuff, or PID errors have occurred. This bit does not update for some endpoint mode settings. Refer to *Table 22-3* for more details.

1 = Data is valid.

0 = Data is invalid. If enabled, the endpoint interrupt will occur even if invalid data is received.

Bit [5:4]: Reserved

Bit [3:0]: Byte Count Bit [3:0]

Byte Count Bits indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint FIFO. Valid values are 0 to 8 inclusive. For OUT or SETUP transactions, the count is updated by hardware to the number of data bytes received, plus 2 for the CRC bytes. Valid values are 2 to 10 inclusive.

For Endpoint 0 Count Register, whenever the count updates from a SETUP or OUT transaction, the count register locks and cannot be written by the CPU. Reading the register unlocks it. This prevents firmware from overwriting a status update on incoming SETUP or OUT transactions before firmware has a chance to read the data.

15.0 USB Regulator Output

The VREG pin provides a regulated output for connecting the pull-up resistor required for USB operation. For USB, a 1.5-kΩ resistor is connected between the D– pin and the V_{REG} voltage, to indicate low-speed USB operation. Since the VREG output has an internal series resistance of approximately 200Ω, the external pull-up resistor required is R_{PU} (see Section 25.0).

The regulator output is placed in a high-impedance state at reset, and must be enabled by firmware by setting the VREG Enable bit in the USB Status and Control Register (*Figure 13-1*). This simplifies the design of a combination PS/2-USB device, since the USB pull-up resistor can be left in place during PS/2 operation without loading the PS/2 line. In this mode, the V_{REG} pin can be used as an input and its state can be read at port P2.0. Refer to *Figure 12-8* for the Port 2 data register. This input has a TTL threshold.

In suspend mode, the regulator is automatically disabled. If VREG Enable bit is set (*Figure 13-1*), the VREG pin is pulled up to V_{CC} with an internal 6.2-kΩ resistor. This holds the proper V_{OH} state in suspend mode

Note that enabling the device for USB (by setting the Device Address Enable bit, *Figure 14-1*) activates the internal regulator, even if the VREG Enable bit is cleared to 0. This insures proper USB signaling in the case where the VREG pin is used as an input, and an external regulator is provided for the USB pull-up resistor. This also limits the swing on the D– and D+ pins to about 1V above the internal regulator voltage, so the Device Address Enable bit normally should only be set for USB operating modes.

The regulator output is only designed to provide current for the USB pull-up resistor. In addition, the output voltage at the VREG pin is effectively disconnected when the CY7C637xx device transmits USB from the internal SIE. This means that the VREG pin does not provide a stable voltage during transmits, although this does not affect USB signaling.

16.0 PS/2 Operation

The CY7C637xx parts are optimized for combination USB or PS/2 devices, through the following features:

1. USB D+ and D– lines can also be used for PS/2 SCLK and SDATA pins, respectively. With USB disabled, these lines can be placed in a high-impedance state that will pull up to V_{CC}. (Disable USB by clearing the Address Enable bit of the USB Device Address Register, *Figure 14-1*).
2. An interrupt is provided to indicate a long LOW state on the SDATA pin. This eliminates the need to poll this pin to check for PS/2 activity. Refer to Section 21.3 for more details.
3. Internal PS/2 pull-up resistors can be enabled on the SCLK and SDATA lines, so no GPIO pins are required for this task (bit 7, USB Status and Control Register, *Figure 13-1*).
4. The controlled slew rate outputs from these pins apply to both USB and PS/2 modes to minimize EMI.
5. The state of the SCLK and SDATA pins can be read, and can be individually driven LOW in an open drain mode. The pins are read at bits [5:4] of Port 2, and are driven with the Control Bits [2:0] of the USB Status and Control Register.
6. The V_{REG} pin can be placed into a high-impedance state, so that a USB pull-up resistor on the D–/SDATA pin will not interfere with PS/2 operation (bit 6, USB Status and Control Register).

The PS/2 on-chip support circuitry is illustrated in *Figure 16-1*.

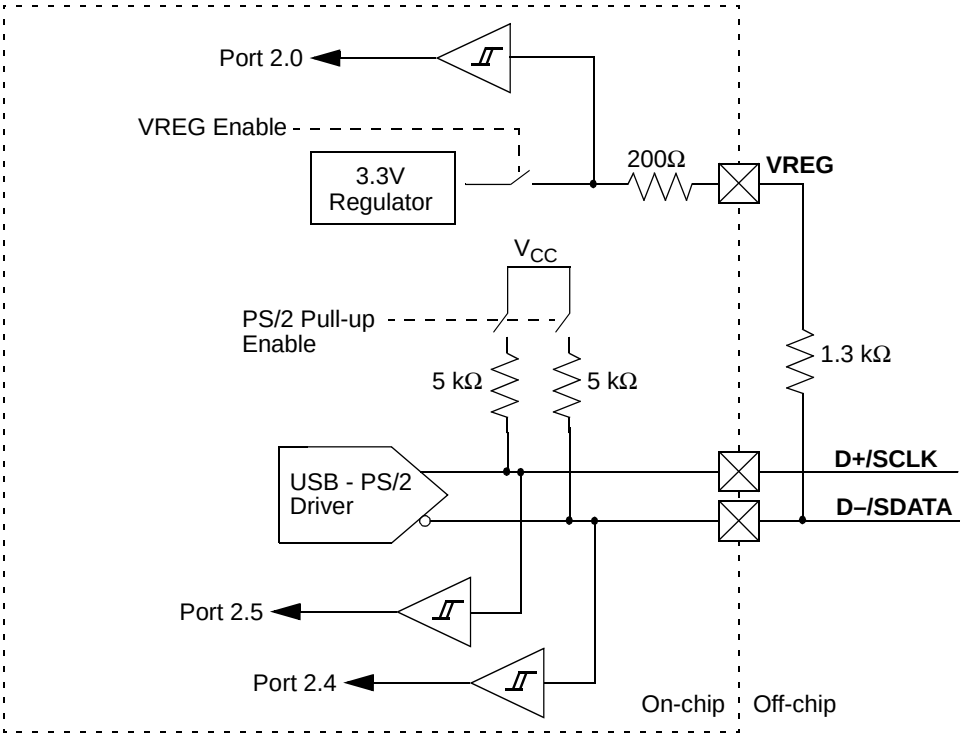


Figure 16-1. Diagram of USB-PS/2 System Connections

17.0 Serial Peripheral Interface (SPI)

SPI is a four-wire, full-duplex serial communication interface between a master device and one or more slave devices. The CY7C637xx SPI circuit supports byte serial transfers in either Master or Slave modes. The block diagram of the SPI circuit is shown in *Figure 17-1*. The block contains buffers for both transmit and receive data for maximum flexibility and throughput. The CY7C637xx can be configured as either an SPI Master or Slave. The external interface consists of Master-Out/Slave-In (MOSI), Master-In/Slave-Out (MISO), Serial Clock (SCK), and Slave Select ($\overline{SS}$).

SPI modes are activated by setting the appropriate bits in the SPI Control Register, as described below.

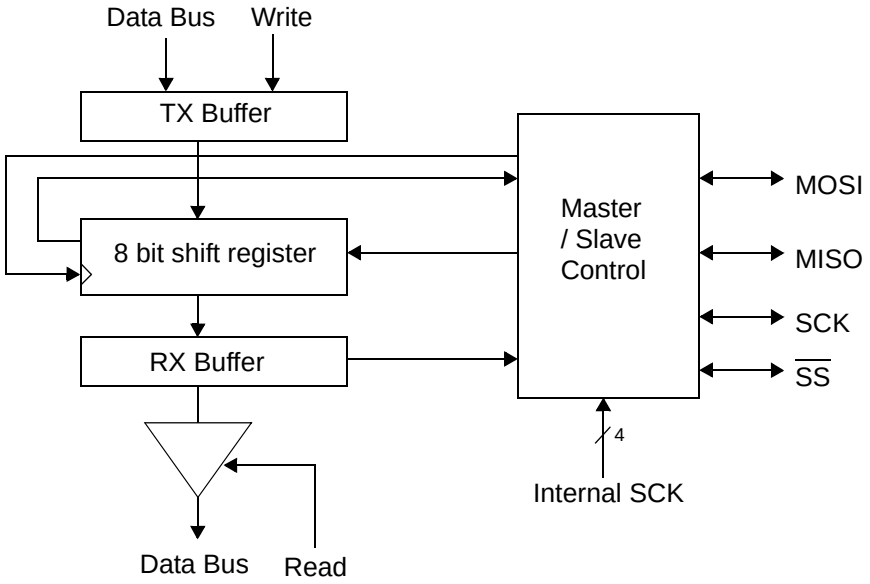


Figure 17-1. SPI Block Diagram

The SPI Data Register below serves as a transmit and receive buffer.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Data I/O							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 17-2. SPI Data Register (Address 0x60)

Bit [7:0]: Data I/O[7:0]

Writes to the SPI Data Register load the transmit buffer, while reads from this register read the receive buffer contents.

1 = Logic HIGH

0 = Logic LOW

17.1 Operation as an SPI Master

Only an SPI Master can initiate a byte/data transfer. This is done by the Master writing to the SPI Data Register. The Master shifts out 8 bits of data (MSB first) along with the serial clock SCK for the Slave. The Master's outgoing byte is replaced with an incoming one from a Slave device. When the last bit is received, the shift register contents are transferred to the receive buffer and an interrupt is generated. The receive data must be read from the SPI Data Register before the next byte of data is transferred to the receive buffer, or the data will be lost.

When operating as a Master, an active LOW Slave Select ($\overline{SS}$) must be generated to enable a Slave for a byte transfer. This Slave Select is generated under firmware control, and is not part of the SPI internal hardware. Any available GPIO can be used for the Master's Slave Select output.

When the Master writes to the SPI Data Register, the data is loaded into the transmit buffer. If the shift register is not busy shifting a previous byte, the TX buffer contents will be automatically transferred into the shift register and shifting will begin. If the shift register is busy, the new byte will be loaded into the shift register only after the active byte has finished and is transferred to the receive buffer. The new byte will then be shifted out. The Transmit Buffer Full (TBF) bit will be set HIGH until the transmit buffer's data-byte is transferred to the shift register. Writing to the transmit buffer while the TBF bit is HIGH will overwrite the old byte in the transmit buffer.

The byte shifting and SCK generation are handled by the hardware (based on firmware selection of the clock source). Data is shifted out on the MOSI pin (P0.5) and the serial clock is output on the SCK pin (P0.7). Data is received from the slave on the MISO pin (P0.6). The output pins must be set to the desired drive strength, and the GPIO data register must be set to 1 to enable a bypass mode for these pins. The MISO pin must be configured in the desired GPIO input mode. See Section 12.0 for GPIO configuration details.

17.2 Master SCK Selection

The Master's SCK is programmable to one of four clock settings, as shown in *Figure 17-1*. The frequency is selected with the Clock Select Bits of the SPI control register. The

hardware provides 8 output clocks on the SCK pin (P0.7) for each byte transfer. Clock phase and polarity are selected by the CPHA and CPOL control bits (see *Figure 17-1* and *17-4*).

The master SCK duty cycle is nominally 33% in the fastest (2 Mbps) mode, and 50% in all other modes.

17.3 Operation as an SPI Slave

In slave mode, the chip receives SCK from an external master on pin P0.7. Data from the master is shifted in on the MOSI pin (P0.5), while data is being shifted out of the slave on the MISO pin (P0.6). In addition, the active LOW Slave Select must be asserted to enable the slave for transmit. The Slave Select pin is P0.4. These pins must be configured in appropriate GPIO modes, with the GPIO data register set to 1 to enable bypass mode selected for the MISO pin.

In Slave mode, writes to the SPI Data Register load the Transmit buffer. If the Slave Select is asserted ($\overline{SS}$ LOW) and the shift register is not busy shifting a previous byte, the transmit buffer contents will be automatically transferred into the shift register. If the shift register is busy, the new byte will be loaded into the shift register only after the active byte has finished and is transferred to the receive buffer. The new byte is then ready to be shifted out (shifting waits for SCK from the Master). If the Slave Select is not active when the transmit buffer is loaded, data is not transferred to the shift register until Slave Select is asserted. The Transmit Buffer Full (TBF) bit will be set to '1' until the transmit buffer's data-byte is transferred to the shift register. Writing to the transmit buffer while the TBF bit is HIGH will overwrite the old byte in the Transmit Buffer.

If the Slave Select is deasserted before a byte transfer is complete, the transfer is aborted and no interrupt is generated. Whenever Slave Select is asserted, the transmit buffer is automatically reloaded into the shift register.

Clock phase and polarity must be selected to match the SPI master, using the CPHA and CPOL control bits (see *Figure 17-3* and *Figure 17-4*).

The SPI slave logic continues to operate in suspend, so if the SPI interrupt is enabled, the device can go into suspend during a SPI slave transaction, and it will wake up at the interrupt that signals the end of the byte transfer.

17.4 SPI Status and Control

The SPI Control Register is shown in *Figure 17-3*. The timing diagram in *Figure 17-4* shows the clock and data states for the various SPI modes.

Bit #	7	6	5	4	3	2	1	0
Bit Name	TCMP	TBF	Comm Mode[1:0]		CPOL	CPHA	SCK Select	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 17-3. SPI Control Register (Address 0x61)

Bit 7: TCMP

1 = TCMP is set to 1 by the hardware when 8-bit transfer is complete. The SPI interrupt is asserted at the same time TCMP is set to 1.

0 = This bit is only cleared by firmware.

Bit 6: TBF

Transmit Buffer Full bit.

1 = Indicates data in the transmit buffer has not transferred to the shift register.

0 = Indicates data in the transmit buffer has transferred to the shift register.

Bit [5:4] Comm Mode[1:0]

00 = All communications functions disabled (default).

01 = SPI Master Mode.

10 = SPI Slave Mode.

11 = Reserved.

Bit 3: CPOL

SPI Clock Polarity bit.

1 = SCK idles HIGH.

0 = SCK idles LOW.

Bit 2: CPHA

SPI Clock Phase bit (see *Figure 17-4*)

Bit [1:0]: SCK Select

Master mode SCK frequency selection (no effect in Slave Mode):

00 = 2 Mbit/s

01 = 1 Mbit/s

10 = 0.5 Mbit/s

11 = 0.0625 Mbit/s

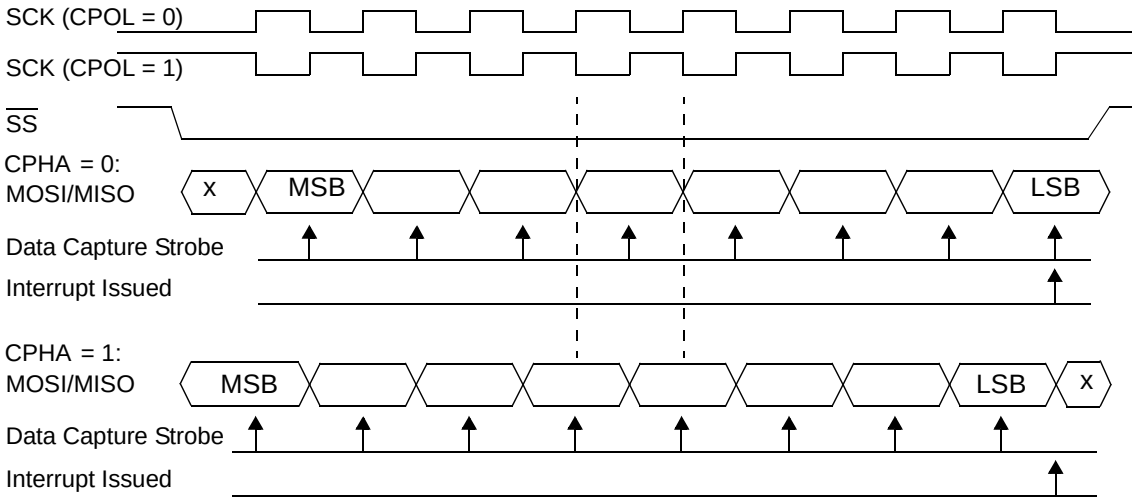


Figure 17-4. SPI Data Timing

17.5 SPI Interrupt

For SPI, an interrupt request is generated after a byte is received or transmitted. See Section 21.3 for details on the SPI interrupt.

17.6 SPI Modes for GPIO Pins

The GPIO pins used for SPI outputs (P0.5–P0.7) contain a bypass mode, as shown in the GPIO block diagram (*Figure 12-1*). Whenever the SPI block is inactive (Mode[5:4] = 00), the bypass value is 1, which enables normal GPIO operation. When SPI master or slave modes are activated, the appropriate bypass signals are driven by the hardware for outputs, and are held at 1 for inputs. **Note that the corresponding data bits in the Port 0 Data Register must be set to 1 for each pin being used for an SPI output.** In addition, the GPIO modes are not affected by operation of the SPI block, so each pin must be programmed by firmware to the desired drive strength mode.

For GPIO pins that are not used for SPI outputs, the SPI bypass value in *Figure 12-1* is always 1, for normal GPIO operation.

Table 17-1. SPI Pin Assignments

SPI Function	GPIO Pin	Comment
Slave Select ($\overline{SS}$)	P0.4	For Master Mode, Firmware sets $\overline{SS}$, may use any GPIO pin. For Slave Mode, $\overline{SS}$ is an active LOW input.
Master Out, Slave In (MOSI)	P0.5	Data output for master, data input for slave.
Master In, Slave Out (MISO)	P0.6	Data input for master, data output for slave.
SCK	P0.7	SPI Clock: Output for master, input for slave.

18.0 12-bit Free-running Timer

The 12-bit timer operates with a 1-µs tick, provides two interrupts (128-µs and 1.024-ms) and allows the firmware to directly time events that are up to 4 ms in duration. The lower eight bits of the timer can be read directly by the firmware. Reading the lower eight bits latches the upper four bits into a temporary register. When the firmware reads the upper four bits of the timer, it is actually reading the count stored in the temporary register. The effect of this is to ensure a stable 12-bit timer value can be read, even when the two reads are separated in time.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Timer [7:0]							
Read/Write	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Figure 18-1. Timer LSB Register (Address 0x24)

Bit [7:0]: Timer lower eight bits

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved				Timer [11:8]			
Read/Write	-	-	-	-	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Figure 18-2. Timer MSB Register (Address 0x25)

Bit [7:4]: Reserved

Bit [3:0]: Timer upper four bits

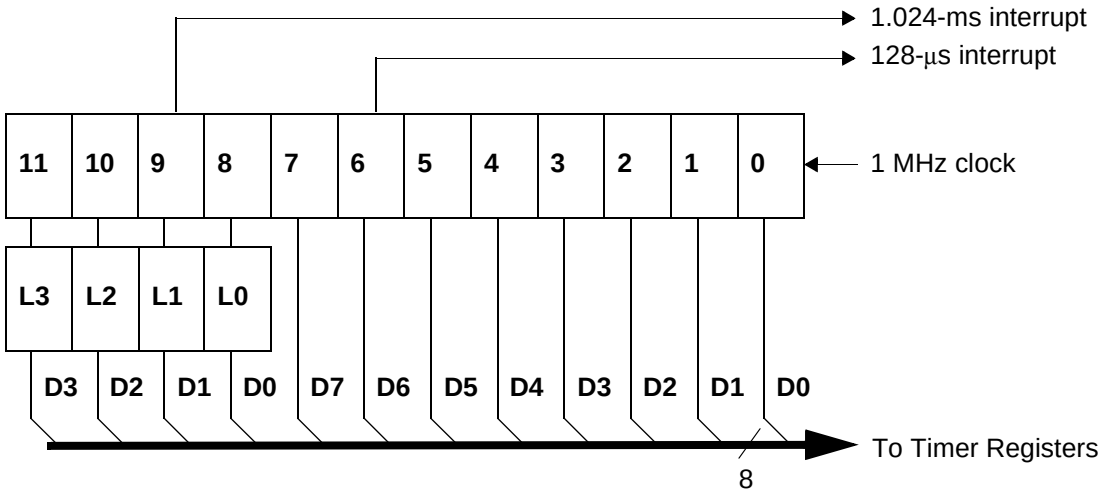


Figure 18-3. Timer Block Diagram

19.0 Timer Capture Registers

Four 8-bit capture timer registers provide both rising- and falling-edge event timing capture on two pins. Capture Timer A is connected to Pin 0.0, and Capture Timer B is connected to Pin 0.1. These can be used to mark the time at which a rising or falling event occurs at the two GPIO pins. Each timer will capture eight bits of the free-running timer into its Capture Timer Data Register if a rising or falling edge event that matches the specified rising or falling edge condition at the pin. A prescaler allows selection of the capture timer tick size. Interrupts can be individually enabled for the four capture registers. A block diagram is shown in *Figure 19-1*.

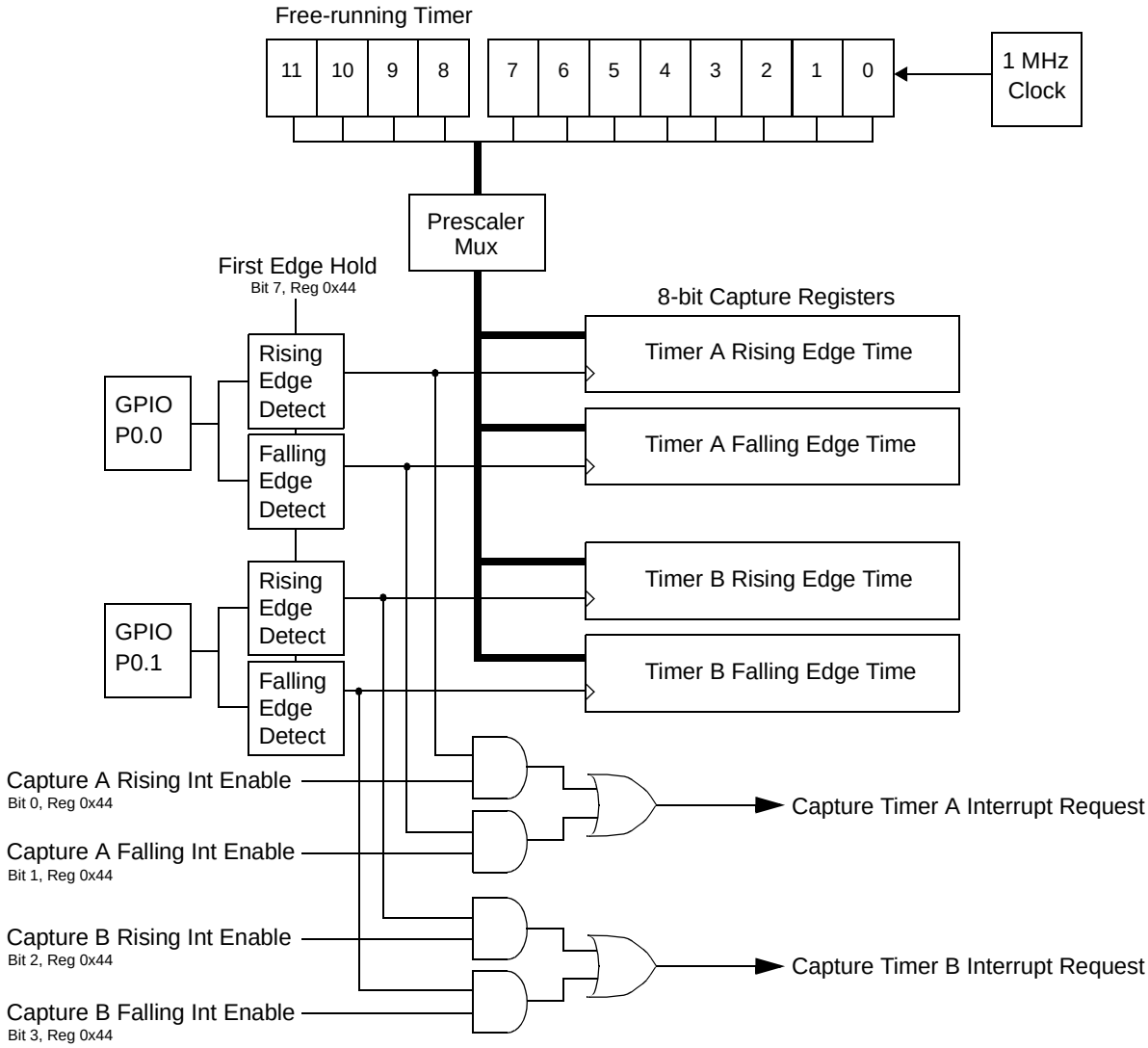


Figure 19-1. Capture Timers Block Diagram

The four Capture Timer Data Registers are read-only, and are shown in *Figure 19-2* through *Figure 19-5*.

Out of the 12-bit free running timer, the 8-bit captured in the Capture Timer Data Registers are determined by the Prescale Bit [2:0] in the Capture Timer Configuration Register (*Figure 19-7*).

Bit #	7	6	5	4	3	2	1	0
Bit Name	Capture A Rising Data							
Read/Write	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Figure 19-2. Capture Timer A-Rising, Data Register (Address 0x40)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Capture A Falling Data							
Read/Write	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Figure 19-3. Capture Timer A-Falling, Data Register (Address 0x41)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Capture B Rising Data							
Read/Write	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Figure 19-4. Capture Timer B-Rising, Data Register (Address 0x42)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Capture B Falling Data							
Read/Write	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Figure 19-5. Capture Timer B-Falling, Data Register (Address 0x43)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved				Capture B Falling Event	Capture B Rising Event	Capture A Falling Event	Capture A Rising Event
Read/ Write	-	-	-	-	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Figure 19-6. Capture Timer Status Register (Address 0x45)

Bit [7:4]: Reserved.

Bit [3:0]: Capture A/B, Falling/Rising Event

These bits record the occurrence of any rising or falling edges on the capture GPIO pins. Bits in this register are cleared by reading the corresponding data register.

1 = A rising or falling event that matches the pin's rising/falling condition has occurred.

0 = No event that matches the pin's rising or falling edge condition.

Because both Capture A events (rising and falling) share an interrupt, user's firmware needs to check the status of both Capture A Falling and Rising Event bits to determine what caused the interrupt. This is also true for Capture B events.

Bit #	7	6	5	4	3	2	1	0
Bit Name	First Edge Hold	Prescale Bit [2:0]			Capture B Falling Int Enable	Capture B Rising Int Enable	Capture A Falling Int Enable	Capture A Rising Int Enable
Read/ Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 19-7. Capture Timer Configuration Register (Address 0x44)

Bit 7: First Edge Hold

1 = The time of the first occurrence of an edge is held in the Capture Timer Data Register until the data is read. Subsequent edges are ignored until the Capture Timer Data Register is read.

0 = The time of the most recent edge is held in the Capture Timer Data Register. That is, if multiple edges have occurred before reading the capture timer, the time for the last one will be read (default state).

The First Edge Hold function applies globally to all four capture timers.

Bit [6:4]: Prescale Bit [2:0]

Three prescaler bits allow the capture timer clock rate to be selected among 5 choices, as shown in *Table 19-1* below.

Bit [3:0]: Capture A/B, Rising/Falling Interrupt Enable

Each of the four Capture Timer registers can be individually enabled to provide interrupts.

Both Capture A events share a common interrupt request, as do the two Capture B events. In addition to the event enables, the main Capture Interrupt Enables bit in the Global Interrupt Enable register (Section 21.0) must be set to activate a capture interrupt.

1 = Enable interrupt

0 = Disable interrupt

Table 19-1. Capture Timer Prescalar Settings (Step size and range for F_{CLK} = 6 MHz)

Prescale 2:0	Captured Bits	LSB Step Size	Range
000	Bits 7:0 of free-running timer	1 µs	256 µs
001	Bits 8:1 of free-running timer	2 µs	512 µs
010	Bits 9:2 of free-running timer	4 µs	1.024 ms
011	Bits 10:3 of free-running timer	8 µs	2.048 ms
100	Bits 11:4 of free-running timer	16 µs	4.096 ms

20.0 Processor Status and Control Register

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

Figure 20-1. Processor Status and Control Register (Address 0xFF)

Bit 7: IRQ Pending

When an interrupt is generated, it is registered as a pending interrupt. The interrupt will remain pending until its interrupt enable bit is set (*Figure 21-1* and *Figure 21-2*) and interrupts are globally enabled (Bit 2, Processor Status and Control Register). At that point the internal interrupt handling sequence will clear the IRQ Pending bit until another interrupt is detected as pending. This bit is only valid if the Global Interrupt Enable bit is disabled.

1 = There are pending interrupts.

0 = No pending interrupts.

Bit 6: Watchdog Reset

The Watchdog Timer Reset (WDR) occurs when the internal Watchdog timer rolls over. The timer will roll over and WDR will occur if it is not cleared within t_{WATCH} (see Section 26.0 for the value of t_{WATCH}). This bit is cleared by an LVR/BOR. Note that a Watchdog reset can occur with a POR/LVR/BOR event, as discussed at the end of this section.

1 = A Watchdog reset occurs.

0 = No Watchdog reset

Bit 5: Bus Interrupt Event

The Bus Reset Status is set whenever the event for the USB Bus Reset or PS/2 Activity interrupt occurs. The event type (USB or PS/2) is selected by the state of the USB-PS/2 Interrupt Mode bit in the USB Status and Control Register (see *Figure 13-1*). The details on the event conditions that set this bit are given in Section 21.3. In either mode, this bit is set as soon as the event has lasted for 128–256 µs, and the bit will be set even if the interrupt is not enabled. The bit is only cleared by firmware or LVR/WDR.

1 = A USB reset occurred or PS/2 Activity is detected, depending on USB-PS/2 Interrupt Select bit.

0 = No event detected since last cleared by firmware or LVR/WDR.

Bit 4: LVR/BOR Reset

The Low-voltage or Brown-out Reset is set to ‘1’ during a power-on reset. Firmware can check bits 4 and 6 in the reset handler to determine whether a reset was caused by a LVR/BOR condition or a Watchdog timeout. This bit is not affected by WDR. Note that a LVR/BOR event may be followed by a Watchdog reset before firmware begins executing, as explained at the end of this section.

1 = A POR or LVR has occurred.

0 = No POR nor LVR since this bit last cleared.

Bit 3: Suspend

Writing a '1' to the Suspend bit will halt the processor and cause the microcontroller to enter the suspend mode that significantly reduces power consumption. An interrupt or USB 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. When writing the suspend bit with a resume condition present (such as non-idle USB activity), the suspend state will still be entered, followed immediately by the wake-up process (with appropriate delays for the clock start-up). See Section 11.0 for more details on suspend mode operation.

1 = Suspend the processor.

0 = Not in suspend mode. Cleared by the hardware when resuming from suspend.

Bit 2: Interrupt Enable Sense

This bit shows whether interrupts are enabled or disabled. Firmware has no direct control over this bit as writing a zero or one to this bit position will have no effect on interrupts. This bit is further gated with the bit settings of the Global Interrupt Enable Register (*Figure 21-1*) and USB Endpoint Interrupt Enable Register (*Figure 21-2*). Instructions DI, EI, and RETI manipulate the state of this bit.

1 = Interrupts are enabled.

0 = Interrupts are masked off.

Bit 1: Reserved. Must be written as a 0.

Bit 0: Run

This bit 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 occurs (low-voltage, brown-out, or Watchdog). This bit should normally be written as a ‘1’.

During power-up, or during a low-voltage reset, the Processor Status and Control Register is set to 00010001, which indicates a LVR/BOR (bit 4 set) has occurred and no interrupts are pending (bit 7 clear). Note that during the t_{START} ms partial suspend at start-up (explained in Section 10.1), a Watchdog Reset will also occur. When a WDR occurs during the power-up suspend interval, firmware would read 01010001 from the Status and Control Register after power-up. Normally the LVR/BOR bit should be cleared so that a subsequent WDR can be clearly identified. *Note that if a USB bus reset (long SE0) is received before firmware examines this register, the Bus Interrupt Event bit would also be set.*

During a Watchdog Reset, the Processor Status and Control Register is set to 01XX0001, which indicates a Watchdog Reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear).

21.0 Interrupts

Interrupts can be generated by the GPIO lines, the internal free-running timer, the SPI block, the capture timers, on various USB events, PS/2 activity, or by the wake-up timer. 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 of the interrupt enable registers are cleared, along with the Global Interrupt enable bit of the CPU, effectively disabling all interrupts.

The interrupt controller contains a separate flip-flop for each interrupt. See *Figure 21-3* for the logic block diagram of the interrupt controller. When an interrupt is generated it is first registered as a pending interrupt. It will stay pending until it is serviced or a reset occurs. A pending interrupt will only generate an interrupt request if it is enabled by the corresponding bit in the interrupt enable registers. The highest priority interrupt request will be serviced following the completion of the currently executing instruction.

When servicing an interrupt, the hardware will first disable all interrupts by clearing the Global Interrupt Enable bit in the CPU (the state of this bit can be read at Bit 2 of the Processor Status and Control Register). Next, the flip-flop of the current interrupt is cleared. This is followed by an automatic CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector, see Section 21.1). 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 stored onto the Program Stack by the automatic CALL instruction generated as part of the interrupt acknowledge process. The user firmware is responsible for ensuring that the processor state is preserved and restored during an interrupt. The PUSH A instruction should typically be used as the first command in the ISR to save the accumulator value and the POP A instruction should be used 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.

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

21.1 Interrupt Vectors

The Interrupt Vectors supported by the device are listed in *Table 21-1*. The highest priority interrupt is #1 (USB Bus Reset / PS/2 activity), and the lowest priority interrupt is #11 (Wake-up Timer). Although Reset is not an interrupt, the first instruction executed after a reset is at ROM address 0x0000, which corresponds to the first entry in the Interrupt Vector Table. Interrupt vectors occupy two bytes to allow for a two-byte JMP instruction to the appropriate Interrupt Service Routine (ISR).

Table 21-1. Interrupt Vector Assignments

Interrupt Vector Number	ROM Address	Function
not applicable	0x0000	Execution after Reset begins here
1	0x0002	USB Bus Reset or PS/2 Activity interrupt
2	0x0004	128-µs timer interrupt
3	0x0006	1.024-ms timer interrupt
4	0x0008	USB Endpoint 0 interrupt
5	0x000A	USB Endpoint 1 interrupt
6	0x000C	USB Endpoint 2 interrupt
7	0x000E	SPI Interrupt
8	0x0010	Capture Timer A interrupt
9	0x0012	Capture Timer B interrupt
10	0x0014	GPIO interrupt
11	0x0016	Wake-up Timer interrupt

21.2 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 minimum of 16 clocks (1+10+5) or a maximum of 20 clocks (5+10+5) after the interrupt is issued. With a 6-MHz external resonator, internal CPU clock speed is 12 MHz, so 20 clocks take 20/12 MHz = 1.67 µs.

21.3 Interrupt Sources

The following sections provide details on the different types of interrupt sources.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Wake-up Interrupt Enable	GPIO Interrupt Enable	Capture Timer B Intr. Enable	Capture Timer A Intr. Enable	SPI Interrupt Enable	1.024-ms Interrupt Enable	128-µs Interrupt Enable	USB Bus Reset / PS/2 Activity Intr. Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 21-1. Global Interrupt Enable Register (Address 0x20)

Bit 7: Wake-up Interrupt Enable

The internal wake-up timer is normally used to wake the part from suspend mode, but it can also provide an interrupt when the part is awake. The wake-up timer is cleared whenever the Wake-up Interrupt Enable bit is written to a 0, and runs whenever that bit is written to a 1. When the interrupt is enabled, the wake-up timer provides periodic interrupts at multiples of period, as described in Section 11.2.

1 = Enable wake-up timer for periodic wake-up.

0 = Disable and power-off wake-up timer.

Bit 6: GPIO Interrupt Enable

Each GPIO pin can serve as an interrupt input. During a reset, GPIO interrupts are disabled by clearing all GPIO interrupt enable registers. Writing a '1' to a GPIO Interrupt Enable bit enables GPIO interrupts from the corresponding input pin. These registers are shown in *Figure 21-4* for Port 0 and *Figure 21-5* for Port 1. In addition to enabling the desired individual pins for interrupt, the main GPIO interrupt must be enabled, as explained in Section 21.0.

The polarity that triggers an interrupt is controlled independently for each GPIO pin by the GPIO Interrupt Polarity Registers. Setting a Polarity bit to '0' allows an interrupt on a falling GPIO edge, while setting a Polarity bit to '1' allows an interrupt on a rising GPIO edge. The Polarity Registers reset to 0 and are shown in *Figure 21-6* for Port 0 and *Figure 21-7* for Port 1.

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.The GPIO interrupt structure is illustrated in *Figure 21-8*.

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 CY7C637xx does not assign interrupt priority to different port pins and the Port Interrupt Enable Registers are not affected by the interrupt acknowledge process.

1 = Enable

0 = Disable

Bit [5:4]: Capture Timer A and B Interrupts

There are two capture timer interrupts, one for each associated pin. Each of these interrupts occurs on an enabled edge of the selected GPIO pin(s). For each pin, rising and/or falling edge capture interrupts can be in selected. Refer to Section 19.0. These interrupts are independent of the GPIO interrupt, described in the next section.

1 = Enable

0 = Disable

Bit 3: SPI Interrupt Enable

The SPI interrupt occurs at the end of each SPI byte transaction, at the final clock edge, as shown in *Figure 17-4*. After the interrupt, the received data byte can be read from the SPI Data Register, and the TCMP control bit will be high

1 = Enable

0 = Disable

Bit 2: 1.024-ms Interrupt Enable

The 1.024-ms interrupts are periodic timer interrupts from the free-running timer (based on the 6-MHz clock). The user should disable this interrupt before going into the suspend mode to avoid possible conflicts between servicing the timer interrupts (128-µs interrupt and 1.024-ms interrupt) first or the suspend request first when waking up.

1 = Enable. Periodic interrupts will be generated approximately every 1.024 ms.

0 = Disable.

Bit 1: 128-µs Interrupt Enable

The 128-µs interrupt is another source of timer interrupt from the free-running timer. The user should disable both timer interrupts (128-µs and 1.024-ms) before going into the suspend mode to avoid possible conflicts between servicing the timer interrupts first or the suspend request first when waking up.

1 = Enable. Periodic interrupts will be generated approximately every 128 µs.

0 = Disable.

Bit 0: USB Bus Reset - PS/2 Interrupt Enable

The function of this interrupt is selectable between detection of either a USB bus reset condition, or PS/2 activity. The selection is made with the USB-PS/2 Interrupt Mode bit in the USB Status and Control Register (*Figure 13-1*). In either case, the interrupt will occur if the selected condition exists for 256 µs, and may occur as early as 128 µs.

A USB bus reset is indicated by a single ended zero (SE0) on the USB D+ and D– pins. The USB Bus Reset interrupt occurs when the SE0 condition ends. PS/2 activity is indicated by a continuous LOW on the SDATA pin. The PS/2 interrupt occurs as soon as the long LOW state is detected.

During the entire interval of a USB Bus Reset or PS/2 interrupt event, the USB Device Address register is cleared.

The Bus Reset/PS/2 interrupt may occur 128 µs after the bus condition is removed.

1 = Enable

0 = Disable

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved					EP2 Interrupt Enable	EP1 Interrupt Enable	EP0 Interrupt Enable
Read/Write	-	-	-	-	-	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 21-2. Endpoint Interrupt Enable Register (Address 0x21)

Bit [7:3]: Reserved.

Bit [2:1]: EP2,1 Interrupt Enable

There are two non-control endpoint (EP2 and EP1) interrupts. If enabled, a non-control endpoint interrupt is generated when:

- The USB host writes valid data to an endpoint FIFO. However, if the endpoint is in ACK OUT modes, an interrupt is generated regardless of data packet validity (i.e., good CRC). Firmware must check for data validity.
- The device SIE sends a NAK or STALL handshake packet to the USB host during the host attempts to read data from the endpoint (INs).
- The device receives an ACK handshake after a successful read transaction (IN) from the host.
- The device SIE sends a NAK or STALL handshake packet to the USB host during the host attempts to write data (OUTs) to the endpoint FIFO.

1 = Enable

0 = Disable

Refer to *Table 22-1* for more information.

Bit 0: EP0 Interrupt Enable

If enabled, a control endpoint interrupt is generated when:

- The endpoint 0 mode is set to accept a SETUP token.
- After the SIE sends a 0-byte packet in the status stage of a control transfer.
- The USB host writes valid data to an endpoint FIFO. However, if the endpoint is in ACK OUT modes, an interrupt is generated regardless of what data is received. Firmware must check for data validity.
- The device SIE sends a NAK or STALL handshake packet to the USB host during the host attempts to read data from the endpoint (INs).
- The device SIE sends a NAK or STALL handshake packet to the USB host during the host attempts to write data (OUTs) to the endpoint FIFO.

1 = Enable EP0 interrupt

0 = Disable EP0 interrupt

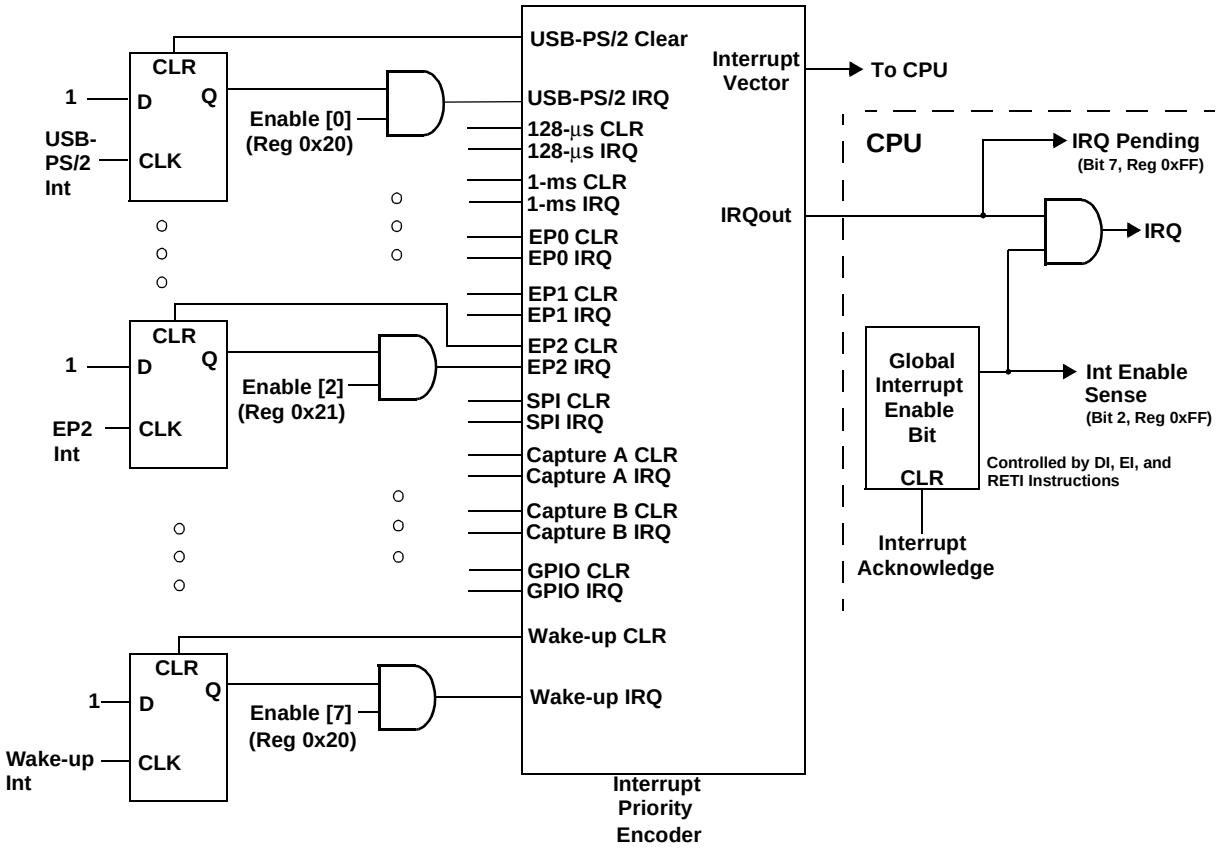


Figure 21-3. Interrupt Controller Logic Block Diagram

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

Figure 21-4. Port 0 Interrupt Enable Register (Address 0x04)

Bit [7:0]: P0 [7:0] Interrupt Enable

1 = Enables GPIO interrupts from the corresponding input pin.

0 = Disables GPIO interrupts from the corresponding input pin.

Bit #	7	6	5	4	3	2	1	0
Bit Name	P1 Interrupt Enable							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 21-5. Port 1 Interrupt Enable Register (Address 0x05)

Bit [7:0]: P1 [7:0] Interrupt Enable

1 = Enables GPIO interrupts from the corresponding input pin.

0 = Disables GPIO interrupts from the corresponding input pin.

The polarity that triggers an interrupt is controlled independently for each GPIO pin by the GPIO Interrupt Polarity Registers. *Figure 21-6* and *Figure 21-7* control the interrupt polarity of each GPIO pin.

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

Figure 21-6. Port 0 Interrupt Polarity Register (Address 0x06)

Bit [7:0]: P0[7:0] Interrupt Polarity

1 = Rising GPIO edge

0 = Falling GPIO edge

Bit #	7	6	5	4	3	2	1	0
Bit Name	P1 Interrupt Polarity							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 21-7. Port 1 Interrupt Polarity Register (Address 0x07)

Bit [7:0]: P1[7:0] Interrupt Polarity

1 = Rising GPIO edge

0 = Falling GPIO edge

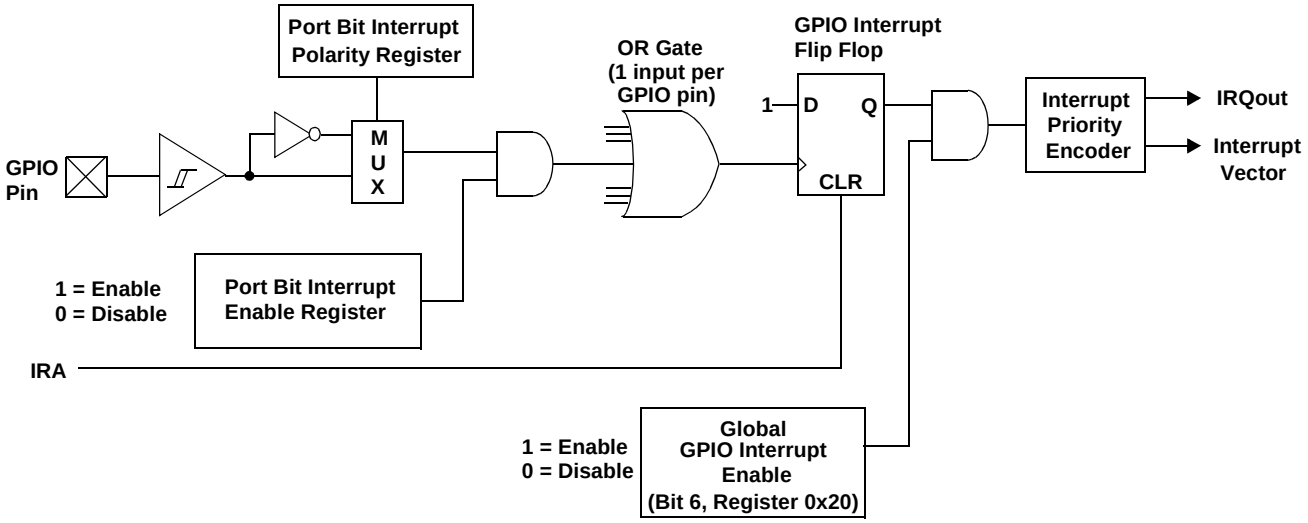


Figure 21-8. GPIO Interrupt Diagram

22.0 USB Mode Tables

The following tables give details on mode setting for the USB Serial Interface Engine (SIE) for both the control endpoint (EP0) and non-control endpoints (EP1 and EP2).

Table 22-1. USB Register Mode Encoding for Control and Non-Control Endpoints

Mode	Encoding	SETUP	IN	OUT	Comments
Disable	**0000**	Ignore	Ignore	Ignore	Ignore all USB traffic to this endpoint
NAK IN/OUT	**0001**	Accept	NAK	NAK	On Control endpoint, after successfully sending an ACK handshake to a SETUP packet, the SIE forces the endpoint mode (from modes other than 0000) to 0001. The mode is also changed by the SIE to 0001 from mode 1011 on issuance of ACK handshake to an OUT.
Status OUT Only	**0010**	Accept	STALL	Check	For Control endpoints
STALL IN/OUT	**0011**	Accept	STALL	STALL	For Control endpoints
Ignore IN/OUT	**0100**	Accept	Ignore	Ignore	For Control endpoints
Reserved	**0101**	Ignore	Ignore	Always	Reserved
Status IN Only	**0110**	Accept	TX 0 Byte	STALL	For Control Endpoints
Reserved	**0111**	Ignore	TX Count	Ignore	Reserved
NAK OUT	**1000**	Ignore	Ignore	NAK	In mode 1001, after sending an ACK handshake to an OUT, the SIE changes the mode to 1000
ACK OUT(STALL[3]=0) ACK OUT(STALL[3]=1)	**1001** **1001**	Ignore Ignore	Ignore Ignore	ACK STALL	This mode is changed by the SIE to mode 1000 on issuance of ACK handshake to an OUT
NAK OUT - Status IN	**1010**	Accept	TX 0 Byte	NAK	
ACK OUT - NAK IN	**1011**	Accept	NAK	ACK	This mode is changed by the SIE to mode 0001 on issuance of ACK handshake to an OUT
NAK IN	**1100**	Ignore	NAK	Ignore	An ACK from mode 1101 changes the mode to 1100
ACK IN(STALL[3]=0) ACK IN(STALL[3]=1)	**1101** **1101**	Ignore Ignore	TX Count STALL	Ignore Ignore	This mode is changed by the SIE to mode 1100 on issuance of ACK handshake to an IN
NAK IN - Status OUT	**1110**	Accept	NAK	Check	An ACK from mode 1111 changes the mode to 1110
ACK IN - Status OUT	**1111**	Accept	TX Count	Check	This mode is changed by the SIE to mode 1110 on issuance of ACK handshake to an IN

Note:

3. STALL bit is the bit 7 of the USB Non-Control Device Endpoint Mode registers. Refer to Section 14.3 for more explanation.

Mode Column:

The 'Mode' column contains the mnemonic names given to the modes of the endpoint. The mode of the endpoint is determined by the four-bit binaries in the 'Encoding' column as discussed below. The Status IN and Status OUT modes represent the status IN or OUT stage of the control transfer.

Encoding Column:

The contents of the 'Encoding' column represent the Mode Bits [3:0] of the Endpoint Mode Registers (*Figure 14-2* and *Figure 14-3*). The endpoint modes determine how the SIE responds to different tokens that the host sends to the endpoints. For example, if the Mode Bits [3:0] of the Endpoint 0 Mode Register (*Figure 14-2*) are set to '0001', which is NAK IN/OUT mode as shown in *Table 22-1* above, the SIE of the part will send an ACK handshake in response to SETUP tokens and NAK any IN or OUT tokens. For more information on the functionality of the Serial Interface Engine (SIE), see Section 13.0.

SETUP, IN, and OUT Columns:

Depending on the mode specified in the 'Encoding' column, the 'SETUP', 'IN', and 'OUT' columns contain the device SIE's responses when the endpoint receives SETUP, IN, and OUT tokens respectively.

A 'Check' in the Out column means that upon receiving an OUT token the SIE checks to see whether the OUT is of zero length and has a Data Toggle (Data1/0) of 1. If these conditions are true, the SIE responds with an ACK. If any of the above conditions is not met, the SIE will respond with either a STALL or Ignore. *Table 22-3* gives a detailed analysis of all possible cases.

A 'TX Count' entry in the IN column means that the SIE will transmit the number of bytes specified in the Byte Count Bit [3:0] of the Endpoint Count Register (*Figure 14-4*) in response to any IN token.

A 'TX 0 Byte' entry in the IN column means that the SIE will transmit a zero byte packet in response to any IN sent to the endpoint. Sending a 0 byte packet is to complete the status stage of a control transfer.

An 'Ignore' means that the device sends no handshake tokens.

An 'Accept' means that the SIE will respond with an ACK to a valid SETUP transaction.

Comments Column:

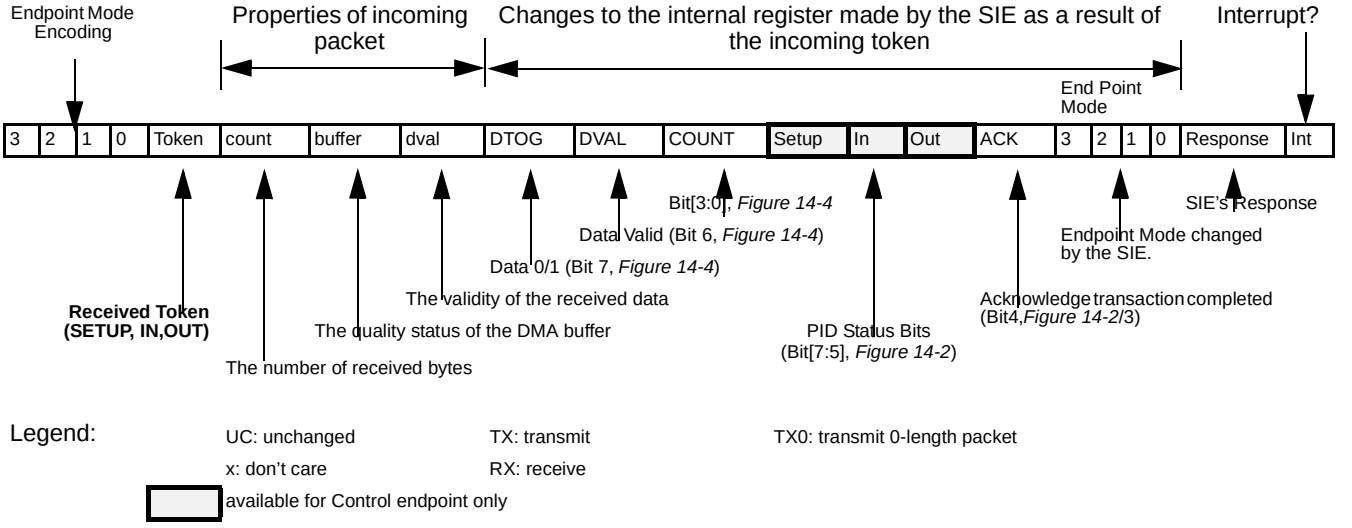
Some Mode Bits are automatically changed by the SIE in response to many USB transactions. For example, if the Mode Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode as shown in *Table 22-1*, the SIE will change the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACKing a valid status stage OUT token. The firmware needs to update the mode for the SIE to respond appropriately. See *Table 22-1* for more details on what modes will be changed by the SIE.

Any SETUP packet to an enabled endpoint with mode set to accept SETUPs will be changed by the SIE to 0001 (NAKing). Any mode set to accept a SETUP will send an ACK handshake to a valid SETUP token.

A disabled endpoint will remain disabled until changed by firmware, and all endpoints reset to the Disabled mode (0000). Firmware normally enables the endpoint mode after a SetConfiguration request.

The control endpoint has three status bits for identifying the token type received (SETUP, IN, or OUT), but the endpoint must be placed in the correct mode to function as such. Non-control endpoints should not be placed into modes that accept SETUPs.

Table 22-2. Decode table for *Table 22-3*: "Details of Modes for Differing Traffic Conditions"



The response of the SIE can be summarized as follows:

1. The SIE will only respond to valid transactions, and will ignore non-valid ones.
2. The SIE will generate an interrupt when a valid transaction is completed or when the FIFO is corrupted. FIFO corruption occurs during an OUT or SETUP transaction to a valid internal address, that ends with a non-valid CRC.
3. An incoming Data packet is valid if the count is ≤ Endpoint Size + 2 (includes CRC) and passes all error checking;
4. An IN will be ignored by an OUT configured endpoint and visa versa.
5. The IN and OUT PID status is updated at the end of a transaction.
6. The SETUP PID status is updated at the beginning of the Data packet phase.
7. The entire Endpoint 0 mode register and the Count register are locked to CPU writes at the end of any transaction to that endpoint in which an ACK is transferred. These registers are only unlocked by a CPU read of these registers, and only if that read happens after the transaction completes. This represents about a 1-µs window in which the CPU is locked from register writes to these USB registers. Normally the firmware should perform a register read at the beginning of the Endpoint ISRs to unlock and get the mode register information. The interlock on the Mode and Count registers ensures that the firmware recognizes the changes that the SIE might have made during the previous transaction.

Table 22-3. Details of Modes for Differing Traffic Conditions

End Point Mode											**PID**				Set End Point Mode					
3	2	1	0	Rcved Token	Count	Buffer	Dval	DTOG	DVAL	COUNT	SETUP	IN	OUT	ACK	3	2	1	0	Response	Int
SETUP Packet (if accepting)																				
See22-1				SETUP	<= 10	data	valid	updates	1	updates	1	UC	UC	1	0	0	0	1	ACK	yes
See22-1				SETUP	> 10	junk	x	updates	updates	updates	1	UC	UC	UC	NoChange				Ignore	yes
See 22-1				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	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
0	0	0	1	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
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	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
0	0	1	1	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
0	0	1	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				STALL	yes
Control Write																				
ACK OUT/NAK IN																				
1	0	1	1	OUT	<= 10	data	valid	updates	1	updates	UC	UC	1	1	0	0	0	1	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	UC	NoChange				NAK	yes
NAK OUT/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 Byte	yes
Status IN Only																				
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 Byte	yes

Table 22-3. Details of Modes for Differing Traffic Conditions (continued)

Control Read																		
ACK IN/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/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
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	SETUP	IN	OUT	ACK	3	2	1	0	response	int
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 Only																		
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
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	NoChange				Ignore	no
0	0	1	0	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
0	0	1	0	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	0	0	1	1	STALL	yes
OUT Endpoint																		
ACK OUT, STALL Bit = 0 (*Figure 14-3*)																		
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
ACK OUT, STALL Bit = 1 (*Figure 14-3*)																		
1	0	0	1	OUT	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	NoChange				STALL	yes
1	0	0	1	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
1	0	0	1	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
1	0	0	1	IN	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				Ignore	no
NAK OUT																		
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
Reserved																		
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
IN Endpoint																		
ACK IN, STALL Bit = 0 (*Figure 14-3*)																		
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
ACK IN, STALL Bit = 1 (*Figure 14-3*)																		
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	UC	NoChange				STALL	yes
NAK IN																		

Table 22-3. Details of Modes for Differing Traffic Conditions (continued)

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
Reserved																	
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

23.0 Register Summary

	Address	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Read/Write/ Both/	Default/ Reset
GPIO CONFIGURATION PORTS 0, 1 AND 2	0x00	Port 0 Data	P0								BBBBBBBB	00000000
	0x01	Port 1 Data	P1								BBBBBBBB	00000000
	0x02	Port 2 Data	Reserved		D+(SCLK) State	D- (SDATA) State	Reserved		P2.1 (Int Clk Mode Only	VREG Pin State	--RR--RR	00000000
	0x0A	GPIO Port 0 Mode 0	P0[7:0] Mode0								WWWWWWWW	00000000
	0x0B	GPIO Port 0 Mode 1	P0[7:0] Mode1								WWWWWWWW	00000000
	0x0C	GPIO Port 1 Mode 0	P1[7:0] Mode0								WWWWWWWW	00000000
	0x0D	GPIO Port 1 Mode 1	P1[7:0] Mode1								WWWWWWWW	00000000
	0x04	Port 0 Interrupt Enable	P0[7:0] Interrupt Enable								WWWWWWWW	00000000
	0x05	Port 1 Interrupt Enable	P1[7:0] Interrupt Enable								WWWWWWWW	00000000
	0x06	Port 0 Interrupt Polarity	P0[7:0] Interrupt Polarity								WWWWWWWW	00000000
	0x07	Port 1 Interrupt Polarity	P1[7:0] Interrupt Polarity								WWWWWWWW	00000000
Clock Config.	0xF8	Clock Configuration	Ext. Clock Resume Delay	Wake-up Timer Adjust Bit [2:0]			Low-voltage Reset Disable	Precision USB Clocking Enable	Internal Clock Output Disable	External Oscillator Enable	BBBBBBBB	00000000
ENDPOINT 0, I AND 2 CONFIGURATION	0x10	USB Device Address	Device Address Enable	Device Address							BBBBBBBB	00000000
	0x12	EP0 Mode	SETUP Received	IN Received	OUT Received	ACKed Transaction	Mode Bit				BBBBBBBB	00000000
	0x14, 0x16	EP1, EP2 Mode Register	STALL	Reserved		ACKed Transaction	Mode Bit				B--BBBBB	00000000
	0x11, 0x13, and 0x15	EP0,1, and 2 Counter	Data 0/1 Toggle	Data Valid	Reserved		Byte Count				BB--BBBB	00000000
USB-SC	0x1F	USB Status and Control	PS/2 Pull-up Enable	VREG Enable	USB Reset-PS/2 Activity Interrupt Mode	Reserved	USB Bus Activity	D+/D- Forcing Bit			BBB-BBBB	00000000
INTERRUPT	0x20	Global Interrupt Enable	Wake-up Interrupt Enable	GPIO Interrupt Enable	Capture Timer B Intr. Enable	Capture Timer A Intr. Enable	SPI Interrupt Enable	1.024 ms Interrupt Enable	128 µs Interrupt Enable	USB Bus Reset-PS/2 Activity Intr. Enable	BBBBBBBB	00000000
	0x21	Endpoint Interrupt Enable	Reserved					EP2 Interrupt Enable	EP1 Interrupt Enable	EP0 Interrupt Enable	-----BBB	00000000
TIMER	0x24	Timer LSB	Timer Bit [7:0]								RRRRRRRR	00000000
	0x25	Timer (MSB)	Reserved				Timer Bit [11:8]				----RRRR	00000000
SPI	0x60	SPI Data	Data I/O								BBBBBBBB	00000000
	0x61	SPI Control	TCMP	TBF	Comm Mode [1:0]		CPOL	CPHA	SCK Select		BBBBBBBB	00000000
CAPTURE TIMER	0x40	Capture Timer A-Rising, Data Register	Capture A Rising Data								RRRRRRRR	00000000
	0x41	Capture Timer A-Falling, Data Register	Capture A Falling Data								RRRRRRRR	00000000
	0x42	Capture Timer B-Rising, Data Register	Capture B Rising Data								RRRRRRRR	00000000
	0x43	Capture Timer B-Falling, Data Register	Capture B Falling Data								RRRRRRRR	00000000
	0x44	Capture Timer Configuration	First Edge Hold	Prescale Bit [2:0]			Capture B Falling Intr Enable	Capture B Rising Intr Enable	Capture A Falling Intr Enable	Capture A Rising Intr Enable	BBBBBBBB	00000000
	0x45	Capture Timer Status	Reserved				Capture B Falling Event	Capture B Rising Event	Capture A Falling Event	Capture A Rising Event	----BBBB	00000000
PROC SC.	0xFF	Process Status & Control	IRQ Pending	Watch Dog Reset	Bus Interrupt Event	LVR/BOR Reset	Suspend	Interrupt Enable Sense	Reserved	Run	RBBBBR-B	See Section 20.0

24.0 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
Maximum Total Sink Output Current into Port 0 and 1 and Pins..... 70 mA
Maximum Total Source Output Current into Port 0 and 1 and Pins ..... 30 mA
Maximum On-chip Power Dissipation on any GPIO Pin .....50 mW
Power Dissipation .....300 mW
Static Discharge Voltage ..... >2000V
Latch-up Current ..... > 200 mA

25.0 DC Characteristics FOSC = 6 MHz; Operating Temperature = 0 to 70°C

	Parameter	Conditions	Min.	Max.	Unit
	General				
V_{CC1}	Operating Voltage	Note 4	V_{LVR}	5.5	V
V_{CC2}	Operating Voltage	Note 4	4.35	5.25	V
I_{CC1}	V_{CC} Operating Supply Current – Internal Oscillator Mode Typical I_{CC1} = 16 mA[5]	V_{CC} = 5.5V, no GPIO loading V_{CC} = 5.0V. T = Room Temperature		20	mA
I_{CC2}	V_{CC} Operating Supply Current – External Oscillator Mode Typical I_{CC2}= 13 mA[5]	V_{CC} = 5.5V, no GPIO loading V_{CC} = 5.0V. T = Room Temperature		17	mA
I_{SB1}	Standby Current – No Wake-up Osc	Oscillator off, D– > 2.7V		25	µA
I_{SB2}	Standby Current – With Wake-up Osc	Oscillator off, D– > 2.7V		75	µA
V_{PP}	Programming Voltage (disabled)		–0.4	0.4	V
T_{RSNTR}	Resonator Start-up Interval	V_{CC} = 5.0V, ceramic resonator		256	µs
I_{IL}	Input Leakage Current	Any I/O pin		1	µA
I_{SNK}	Max I_{SS} GPIO Sink Current	Cumulative across all ports[6]		70	mA
I_{SRC}	Max I_{CC} GPIO Source Current	Cumulative across all ports[6]		30	mA
	Low-Voltage and Power-on Reset				
V_{LVR}	Low-Voltage Reset Trip Voltage	V_{CC} below V_{LVR} for >100 ns[7]	3.5	4.0	V
t_{VCCS}	V_{CC} Power-on Slew Time	linear ramp: 0 to 4V[8]		100	ms
	USB Interface				
V_{REG}	VREG Regulator Output Voltage	Load = R_{PU} +R_{PD}[9, 10]	3.0	3.6	V
C_{REG}	Capacitance on VREG Pin	External cap not required		300	pF
V_{OHU}	Static Output High, driven	R_{PD} to Gnd[4]	2.8	3.6	V

Notes:
4. Full functionality is guaranteed in V_{CC1} range, except USB transmitter specifications and GPIO output currents are guaranteed for V_{CC2} range.
5. Bench measurements taken under nominal operating conditions. Spec cannot be guaranteed at final test.
6. Total current cumulative across all Port pins, limited to minimize Power and Ground-Drop noise effects.
7. LVR is automatically disabled during suspend mode.
8. LVR will re-occur whenever V_{CC} drops below V_{LVR}. In suspend or with LVR disabled, BOR occurs whenever V_{CC} drops below approximately 2.5V.
9. V_{REG} specified for regulator enabled, idle conditions (i.e., no USB traffic), with load resistors listed. During USB transmits from the internal SIE, the VREG output is not regulated, and should not be used as a general source of regulated voltage in that case. During receive of USB data, the VREG output drops when D– is LOW due to internal series resistance of approximately 200Ω at the VREG pin.
10. In suspend mode, V_{REG} is only valid if R_{PU} is connected from D– to VREG pin, and R_{PD} is connected from D– to ground.

25.0 DC Characteristics FOSC = 6 MHz; Operating Temperature = 0 to 70°C (continued)

	Parameter	Conditions	Min.	Max.	Unit
V_{OLU}	Static Output Low	With R_{PU} to VREG pin		0.3	V
V_{OHZ}	Static Output High, idle or suspend	R_{PD} connected D– to Gnd, R_{PU} connected D– to VREG pin[4]	2.7	3.6	V
V_{DI}	Differential Input Sensitivity	|(D+)–(D–)|	0.2		V
V_{CM}	Differential Input Common Mode Range		0.8	2.5	V
V_{SE}	Single Ended Receiver Threshold		0.8	2.0	V
C_{IN}	Transceiver Capacitance			20	pF
I_{LO}	Hi-Z State Data Line Leakage	0 V < V_{in}<3.3 V (D+ or D– pins)	–10	10	µA
R_{PU}	External Bus Pull-up resistance (D–)	1.3 kΩ ±2% to V_{REG}[11]	1.274	1.326	kΩ
R_{PD}	External Bus Pull-down resistance	15 kΩ ±5% to Gnd	14.25	15.75	kΩ
	PS/2 Interface				
V_{OLP}	Static Output Low	Isink = 5 mA, SDATA or SCLK pins		0.4	V
R_{PS2}	Internal PS/2 Pull-up Resistance	SDATA, SCLK pins, PS/2 Enabled	3	7	kΩ
	General Purpose I/O Interface				
R_{UP}	Pull-up Resistance		8	24	kΩ
V_{ICR}	Input Threshold Voltage, CMOS mode	Low to high edge, Port 0 or 1	40%	60%	V_{CC}
V_{ICF}	Input Threshold Voltage, CMOS mode	High to low edge, Port 0 or 1	35%	55%	V_{CC}
V_{HC}	Input Hysteresis Voltage, CMOS mode	High to low edge, Port 0 or 1	3%	10%	V_{CC}
V_{ITTL}	Input Threshold Voltage, TTL mode	Ports 0, 1, and 2	0.8	2.0	V
V_{OL1A}	Output Low Voltage, high drive mode	I_{OL1} = 50 mA, Ports 0 or 1[4]		0.8	V
V_{OL1B}		I_{OL1} = 25 mA, Ports 0 or 1[4]		0.4	V
V_{OL2}	Output Low Voltage, medium drive mode	I_{OL2} = 8 mA, Ports 0 or 1[4]		0.4	V
V_{OL3}	Output Low Voltage, low drive mode	I_{OL3} = 2 mA, Ports 0 or 1[4]		0.4	V
V_{OH}	Output High Voltage, strong drive mode	Port 0 or 1, I_{OH} = 2 mA[4]	V_{CC}–2		V
R_{XIN}	Pull-down resistance, XTALIN pin	Internal Clock Mode only	50		kΩ

Note:
11. The 200Ω internal resistance at the VREG pin gives a standard USB pull-up using this value. Alternately, a 1.5 kΩ,5%pull-up from D– to an external 3.3V supply can be used.

26.0 Switching Characteristics

Parameter	Description	Conditions	Min.	Max.	Unit
	Internal Clock Mode				
F_{ICLK}	Internal Clock Frequency	Internal Clock Mode enabled	5.7	6.3	MHz
F_{ICLK2}	Internal Clock Frequency, USB mode	Internal Clock Mode enabled, Bit 2 of register 0xF8h is set (Precision USB Clocking)[12]	5.91	6.09	MHz
	External Oscillator Mode				
T_{CYC}	Input Clock Cycle Time	USB Operation, with External ±1.5% Ceramic Resonator or Crystal	164.2	169.2	ns
T_{CH}	Clock HIGH Time		0.45 t_{CYC}		ns
T_{CL}	Clock LOW Time		0.45 t_{CYC}		ns
	Reset Timing				
t_{START}	Time-out Delay after LVR/BOR		24	60	ms
t_{WAKE}	Internal Wake-up Period	Enabled Wake-up Interrupt[13]	1	5	ms
t_{WATCH}	WatchDog Timer Period	F_{OSC} = 6 MHz	10.1	14.6	ms
	USB Driver Characteristics				
T_R	Transition Rise Time	CLoad = 200 pF (10% to 90%[4])	75		ns
T_R	Transition Rise Time	CLoad = 600 pF (10% to 90%[4])		300	ns
T_F	Transition Fall Time	CLoad = 200 pF (10% to 90%[4])	75		ns
T_F	Transition Fall Time	CLoad = 600 pF (10% to 90%[4])		300	ns
T_{RFM}	Rise/Fall Time Matching	t_r/t_f[4, 14]	80	125	%
V_{CRS}	Output Signal Crossover Voltage[18]	CLoad = 200 to 600 pF[4]	1.3	2.0	V
	USB Data Timing				
T_{DRATE}	Low Speed Data Rate	Ave. Bit Rate (1.5 Mb/s ±1.5%)	1.4775	1.5225	Mb/s
T_{DJR1}	Receiver Data Jitter Tolerance	To Next Transition[15]	–75	75	ns
T_{DJR2}	Receiver Data Jitter Tolerance	For Paired Transitions[15]	–45	45	ns
T_{DEOP}	Differential to EOP transition Skew	Note 15	–40	100	ns
T_{EOPR2}	EOP Width at Receiver	Accepts as EOP[15]	670		ns
T_{EOPT}	Source EOP Width		1.25	1.50	µs
T_{UDJ1}	Differential Driver Jitter	To next transition, *Figure 26-5*	–95	95	ns
T_{UDJ2}	Differential Driver Jitter	To paired transition, *Figure 26-5*	–150	150	ns
T_{LST}	Width of SE0 during Diff. Transition			210	ns
	Non-USB Mode Driver Characteristics	Note 16			
T_{FPS2}	SDATA/SCK Transition Fall Time	CLoad = 150 pF to 600 pF	50	300	ns
	SPI Timing	See *Figures 26-6 to 26-9*[17]			
T_{SMCK}	SPI Master Clock Rate	F_{CLK}/3; see *Figure 17-1*		2	MHz
T_{SSCK}	SPI Slave Clock Rate			2.2	MHz

Notes:
12. Initially $F_{ICLK2} = F_{ICLK}$ until a USB packet is received.
13. Wake-up time for Wake-up Adjust Bits cleared to 000b (minimum setting)
14. Tested at 200 pF.
15. Measured at cross-over point of differential data signals.
16. Non-USB Mode refers to driving the D–/SDATA and/or D+/SCLK pins with the Control Bits of the USB Status and Control Register, with Control Bit 2 HIGH.
17. SPI timing specified for capacitive load of 50 pF, with GPIO output mode = 01 (medium low drive, strong high drive).
18. Per the USB 2.0 Specification, Table 7.7, Note 10, the first transition from the Idle state is excluded.

26.0 Switching Characteristics (continued)

Parameter	Description	Conditions	Min.	Max.	Unit
T_{SCKH}	SPI Clock High Time	High for CPOL = 0, Low for CPOL = 1	125		ns
T_{SCKL}	SPI Clock Low Time	Low for CPOL = 0, High for CPOL = 1	125		ns
T_{MDO}	Master Data Output Time	SCK to data valid	–25	50	ns
T_{MDO1}	Master Data Output Time, First bit with CPHA = 1	Time before leading SCK edge	100		ns
T_{MSU}	Master Input Data Set-up time		50		ns
T_{MHD}	Master Input Data Hold time		50		ns
T_{SSU}	Slave Input Data Set-up Time		50		ns
T_{SHD}	Slave Input Data Hold Time		50		ns
T_{SDO}	Slave Data Output Time	SCK to data valid		100	ns
T_{SDO1}	Slave Data Output Time, First bit with CPHA = 1	Time after $\overline{SS}$ LOW to data valid		100	ns
T_{SSS}	Slave Select Set-up Time	Before first SCK edge	150		ns
T_{SSH}	Slave Select Hold Time	After last SCK edge	150		ns

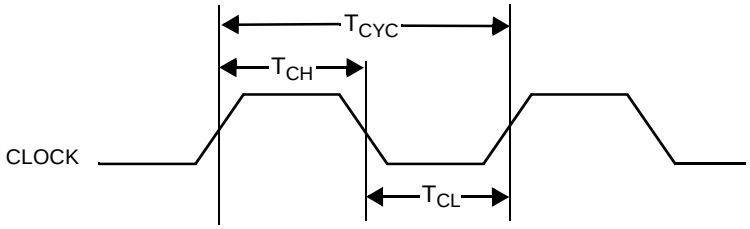


Figure 26-1. Clock Timing

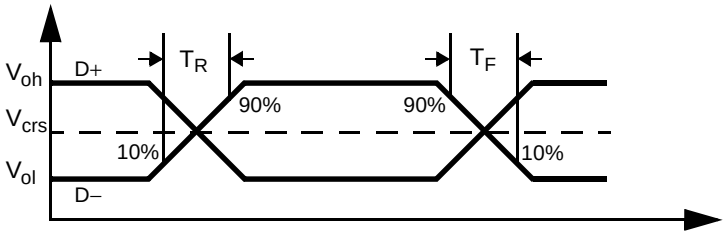


Figure 26-2. USB Data Signal Timing

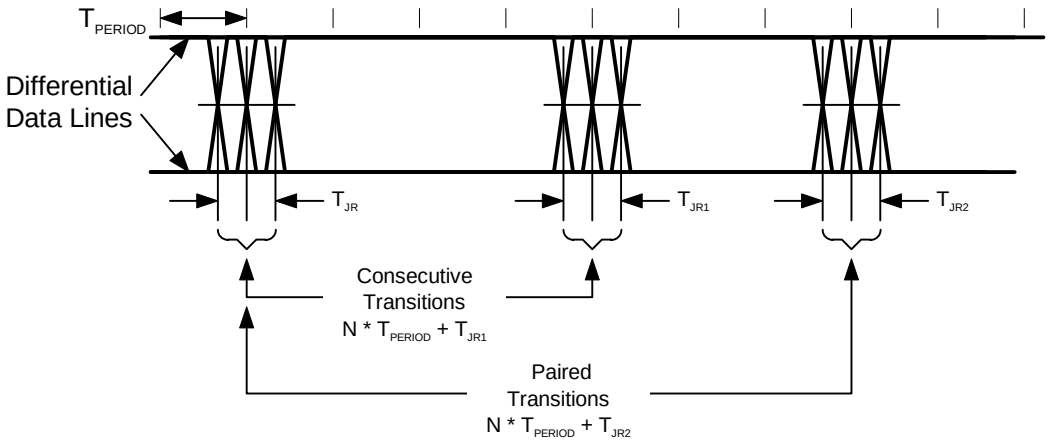


Figure 26-3. Receiver Jitter Tolerance

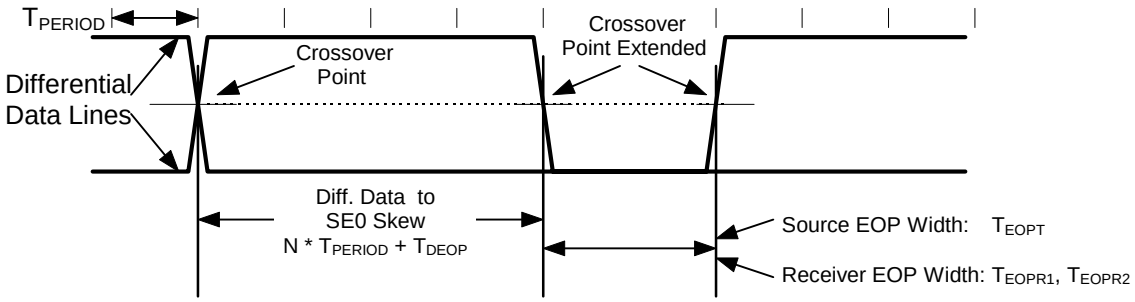


Figure 26-4. Differential to EOP Transition Skew and EOP Width

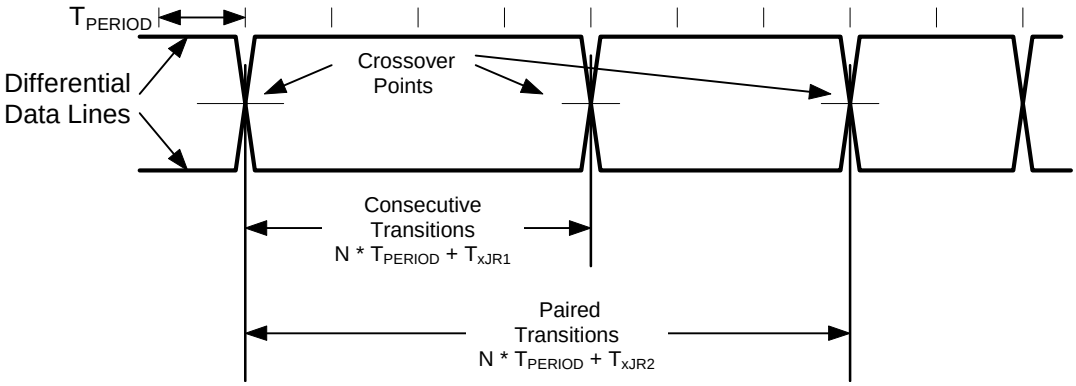


Figure 26-5. Differential Data Jitter

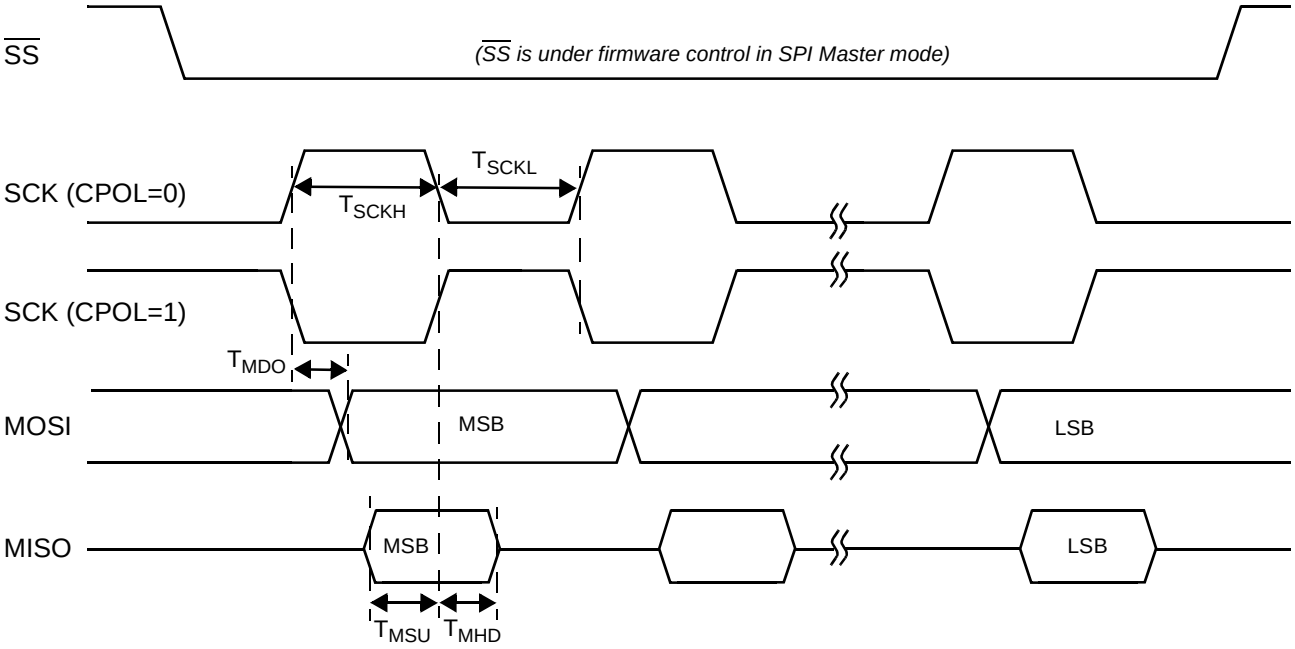


Figure 26-6. SPI Master Timing, CPHA = 0

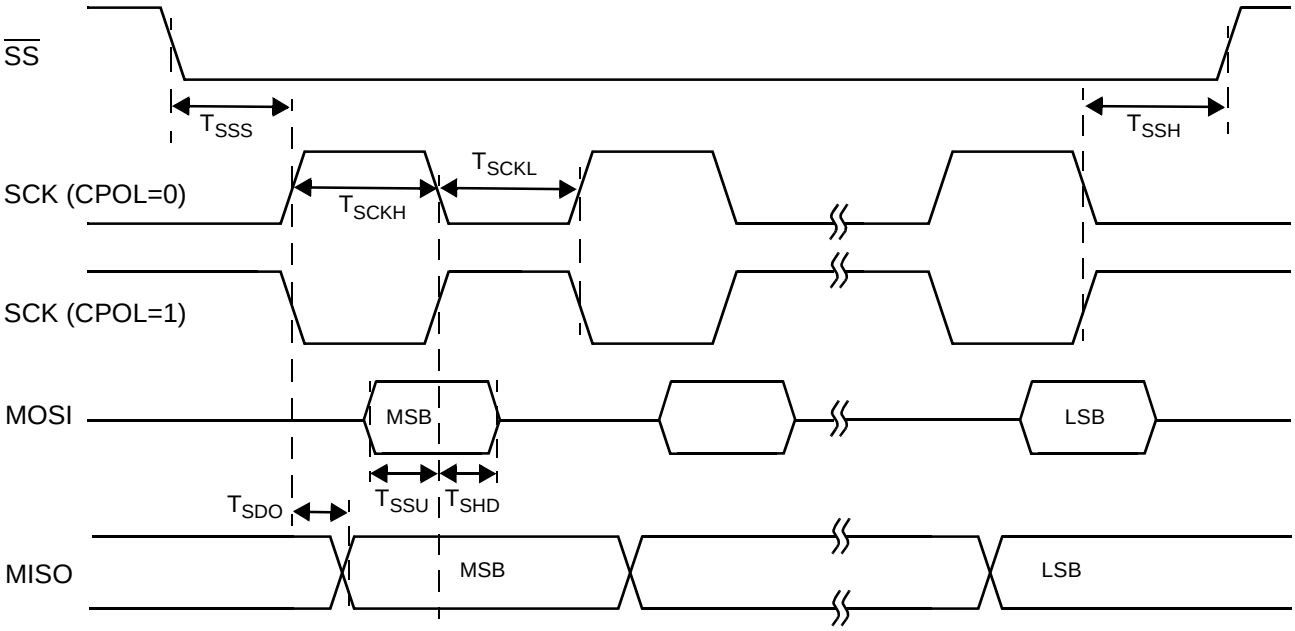


Figure 26-7. SPI Slave Timing, CPHA = 0

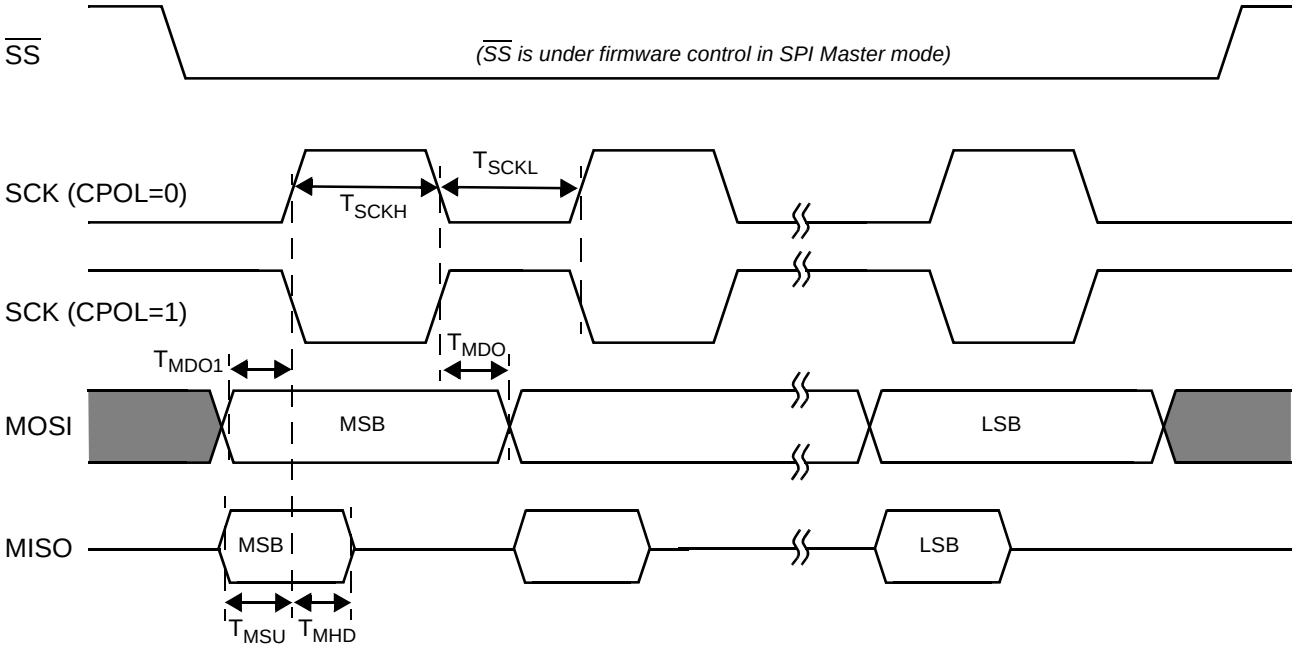


Figure 26-8. SPI Master Timing, CPHA = 1

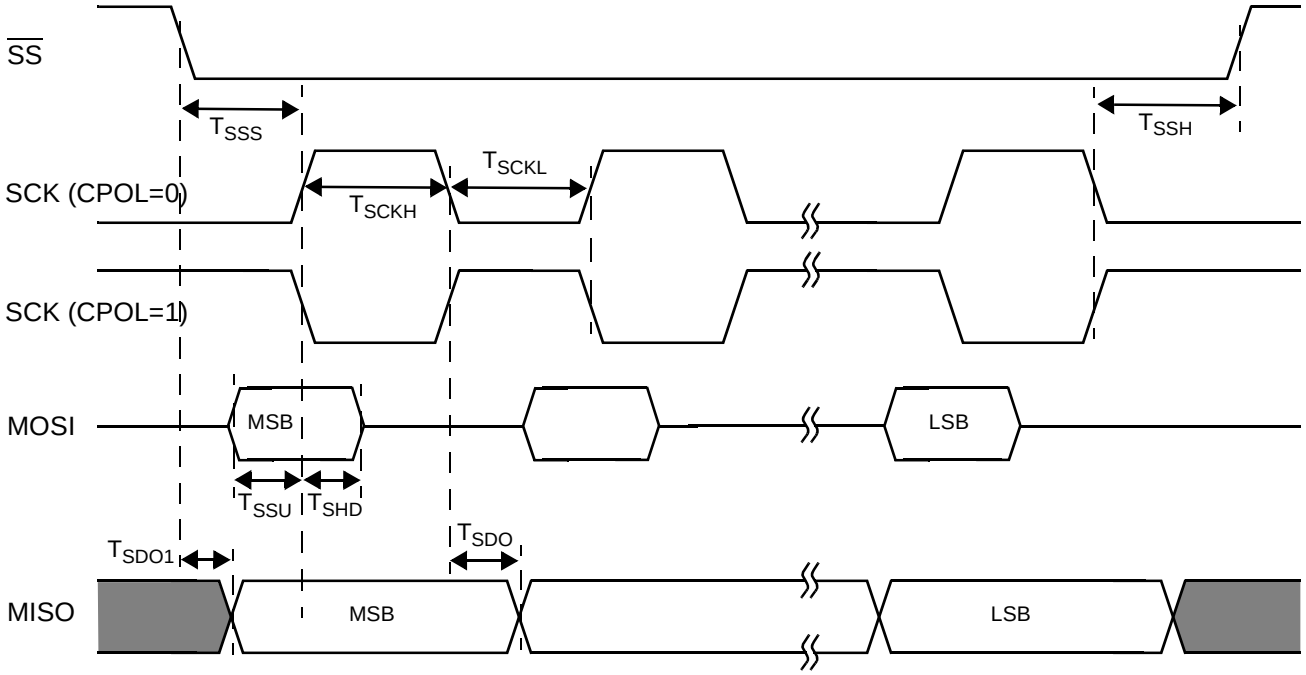


Figure 26-9. SPI Slave Timing, CPHA = 1

27.0 Ordering Information

Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C63723-PC	8 KB	P3	18-Pin (300-Mil) PDIP	Commercial
CY7C63723-PXC	8 KB	P3	18-Pin (300-Mil) Lead-free PDIP	Commercial
CY7C63723-SC	8 KB	S3	18-Pin Small Outline Package	Commercial
CY7C63723-SXC	8 KB	S3	18-Pin Small Outline Lead-free Package	Commercial
CY7C63743-QXC	8 KB	Q13	24 QSOP Lead-free Package	Commercial
CY7C63743-PC	8 KB	P13	24-Pin (300-Mil) PDIP	Commercial
CY7C63743-PXC	8 KB	P13	24-Pin (300-Mil) Lead-free PDIP	Commercial
CY7C63743-SC	8 KB	S13	24-Pin Small Outline Package	Commercial
CY7C63743-SXC	8 KB	S13	24-Pin Small Outline Lead-free Package	Commercial
CY7C63722-XC	8 KB	–	25-Pad DIE Form	Commercial
CY7C63722-XWC	8 KB	–	25-Pad DIE Form Lead-free	Commercial

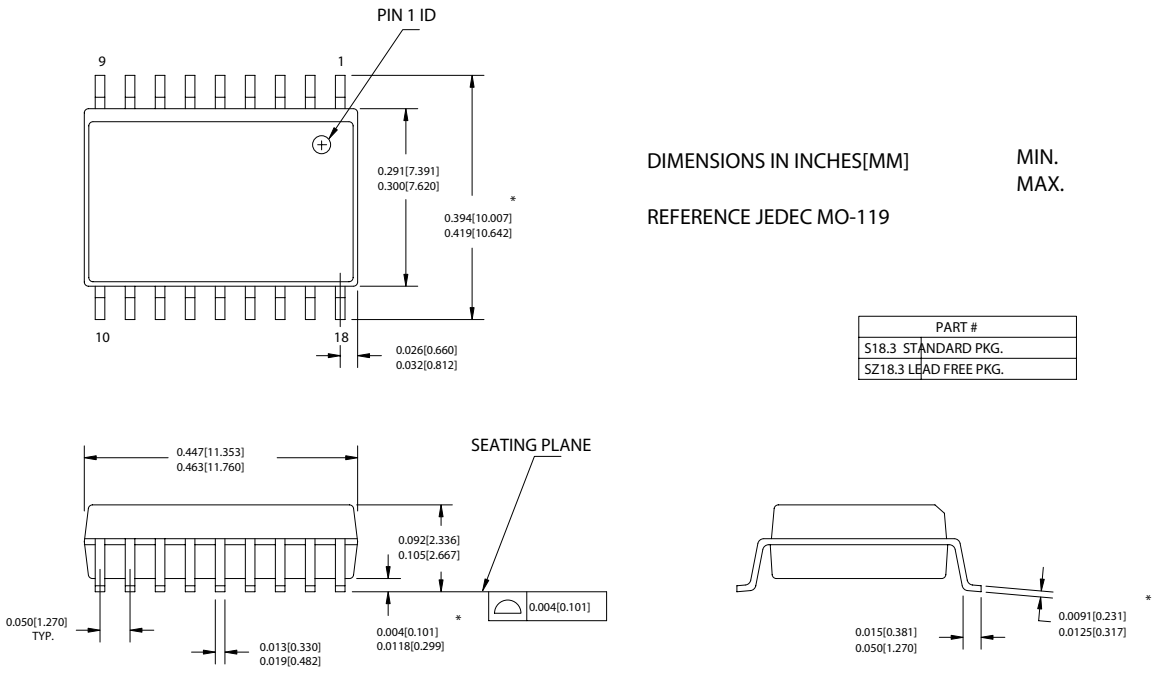
28.0 Package Diagrams

18-Lead (300-Mil) Molded DIP P3

51-85010-*A

28.0 Package Diagrams (continued)

18-Lead (300-Mil) Molded SOIC S3



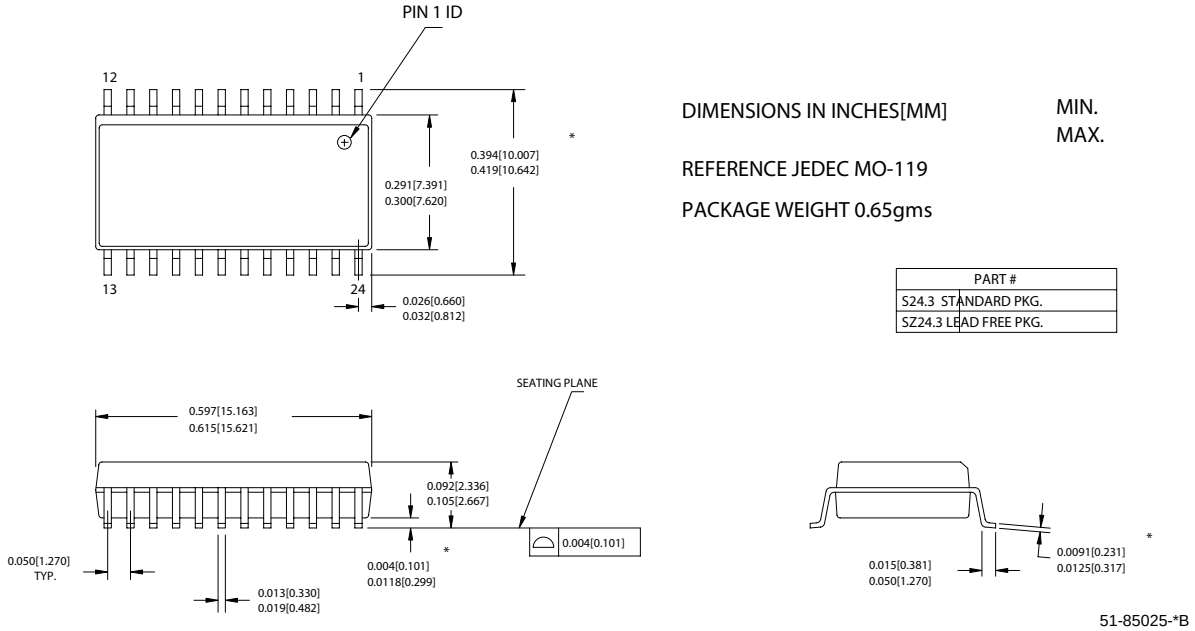
DIMENSIONS IN INCHES[MM] MIN.
MAX.

REFERENCE JEDEC MO-119

PART #
S18.3 STANDARD PKG.
SZ18.3 LEAD FREE PKG.

51-85023-*B

24-Lead (300-Mil) SOIC S13



DIMENSIONS IN INCHES[MM] MIN.
MAX.

REFERENCE JEDEC MO-119

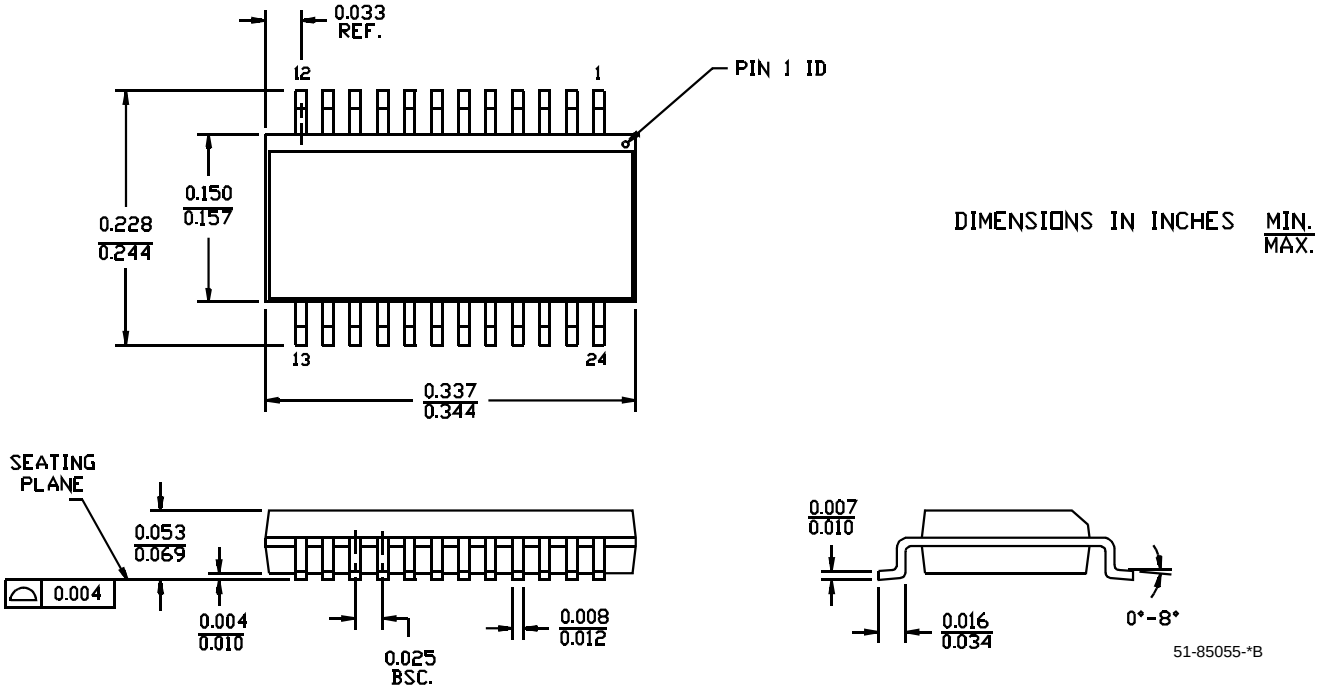
PACKAGE WEIGHT 0.65gms

PART #
S24.3 STANDARD PKG.
SZ24.3 LEAD FREE PKG.

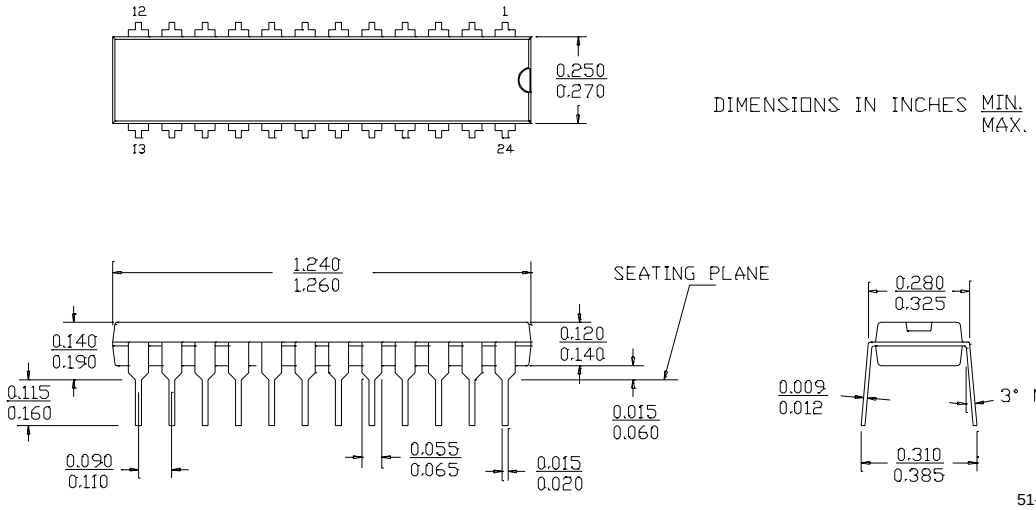
51-85025-*B

28.0 Package Diagrams (continued)

24-Lead Quarter Size Outline Q13



24-Lead (300-Mil) PDIP P13



28.0 Package Diagrams (continued)

DIE FORM

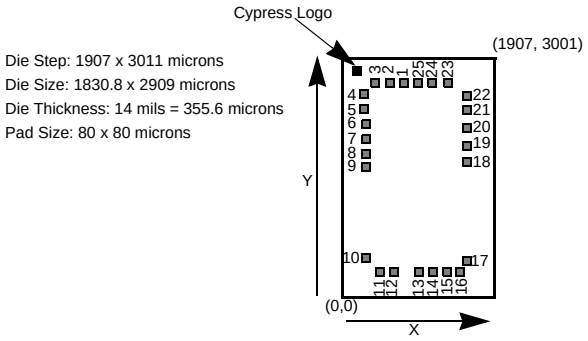


Table 28-1 below shows the die pad coordinates for the CY7C63722-XC. The center location of each bond pad is relative to the bottom left corner of the die which has coordinate (0,0).

Table 28-1. CY7C63722-XC Probe Pad Coordinates in microns ((0,0) to bond pad centers)

Pad Number	Pin Name	X (microns)	Y (microns)
1	P0.0	788.95	2843.15
2	P0.1	597.45	2843.15
3	P0.2	406.00	2843.15
4	P0.3	154.95	2687.95
5	P1.0	154.95	2496.45
6	P1.2	154.95	2305.05
7	P1.4	154.95	2113.60
8	P1.6	154.95	1922.05
9	Vss	154.95	1730.90
10	Vss	154.95	312.50
11	Vpp	363.90	184.85
12	VREG	531.70	184.85
13	XTALIN	1066.55	184.85
14	XTALOUT	1210.75	184.85
15	Vcc	1449.75	184.85
16	D–	1662.35	184.85
17	D+	1735.35	289.85
18	P1.7	1752.05	1832.75
19	P1.5	1752.05	2024.30
20	P1.3	1752.05	2215.75
21	P1.1	1752.05	2407.15
22	P0.7	1752.05	2598.65
23	P0.6	1393.25	2843.15
24	P0.5	1171.80	2843.15
25	P0.4	980.35	2843.15

enCoRe is a trademark of Cypress Semiconductor Corporation. All product and company names mentioned in this document may be the trademarks of their respective holders.

© Cypress Semiconductor Corporation, 2004. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress

Document History Page

Document Title: CY7C63722, CY7C63723, CY7C63743 enCoRe™ USB Combination Low-Speed USB and PS/2 Peripheral Controller Document Number: 38-08022				
REV.	**ECN NO.**	**Issue Date**	**Orig. of Change**	**Description of Change**
**	118643	10/22/02	BON	Converted from Spec 38-00944 to Spec 38-08022. Added notes 17, 18 to section 26 Removed obsolete parts (63722-PC and 63742) Added die sale Added section 23 (Register Summary)
*A	243308	SEE ECN	KKU	Added 24 QSOP package Added Lead-free packages to section 27 Reformatted to update format
*B	267229	See ECN	ARI	Corrected part number in the Ordering Information section