Welcome to E-XFL.COM

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

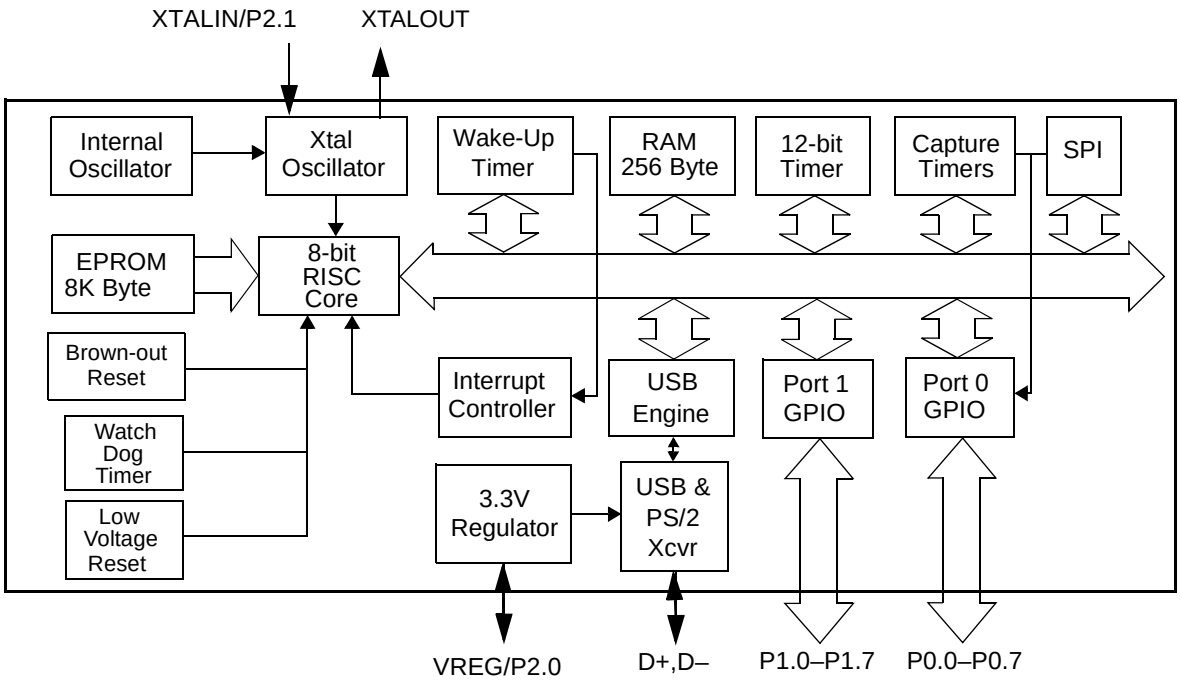
Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering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

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Microcontroller
Core Processor	M8B
Program Memory Type	OTP (8kB)
Controller Series	CY7C637xx
RAM Size	256 x 8
Interface	PS/2, USB
Number of I/O	16
Voltage - Supply	4V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	24-SOIC (0.295", 7.50mm Width)
Supplier Device Package	24-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c63743-sxc

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 breakthrough 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).

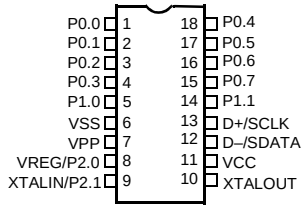
The CY7C637xx includes an integrated USB serial interface engine (SIE) that supports the integrated peripherals. The hardware supports one USB device address with three endpoints. The SIE allows the USB host to communicate with the function integrated into the microcontroller. A 3.3V regulated output 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.

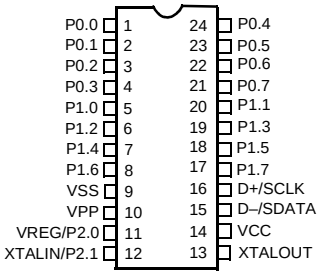
4.0 Pin Configurations

Top View

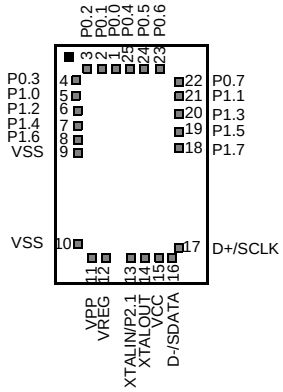
CY7C63723
18-pin SOIC/PDIP



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.

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

8.2 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.

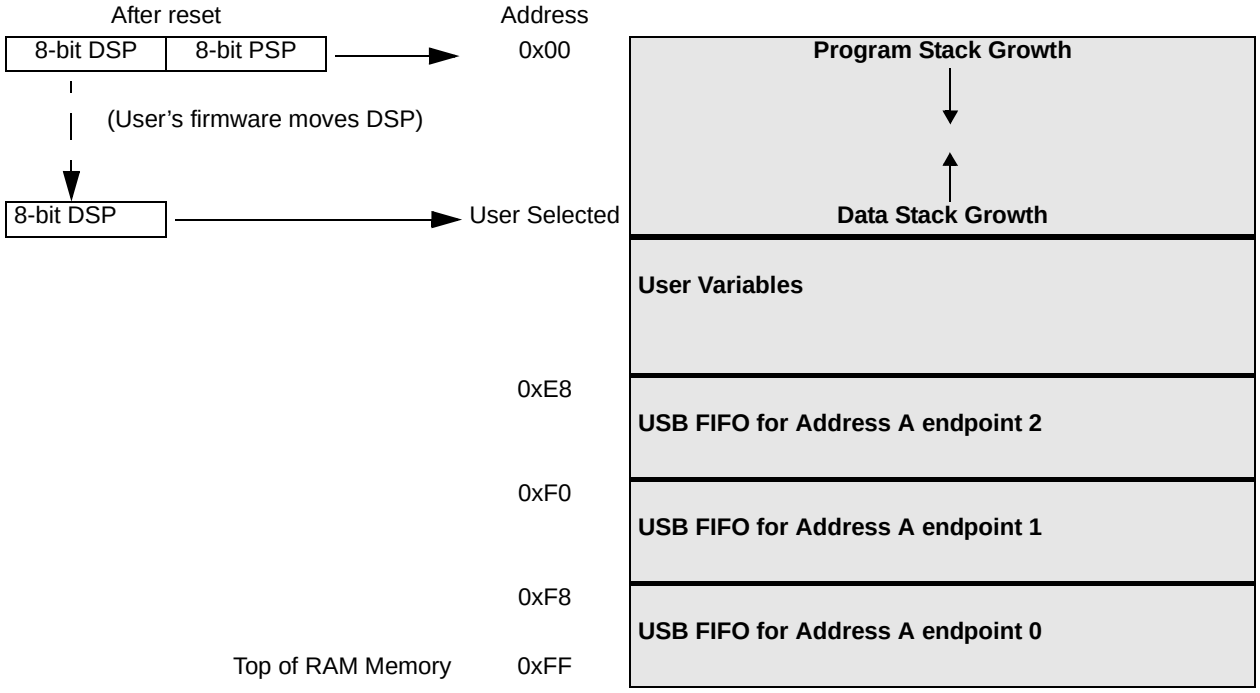


Figure 8-2. Data Memory Organization

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

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.

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.

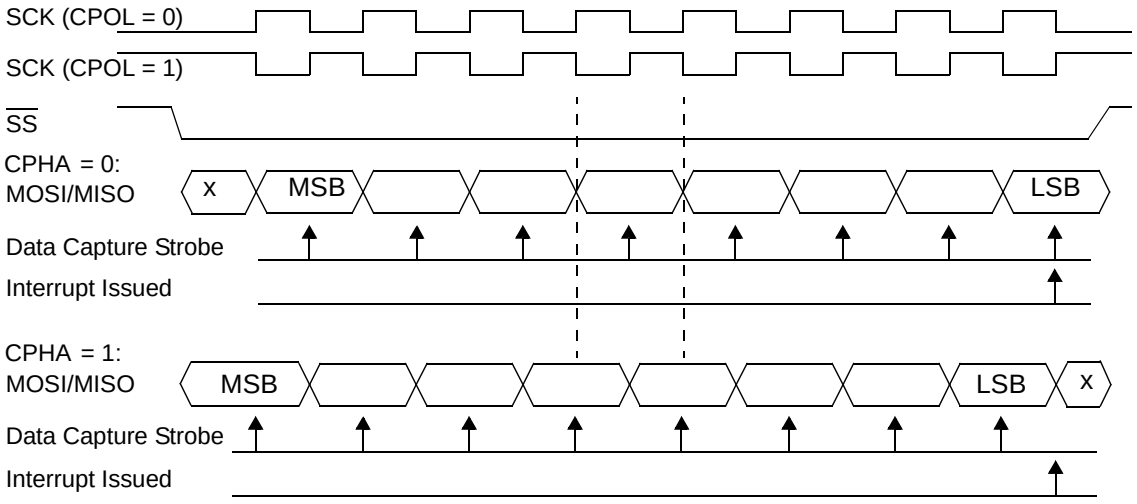


Figure 17-4. SPI Data Timing

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.

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*.

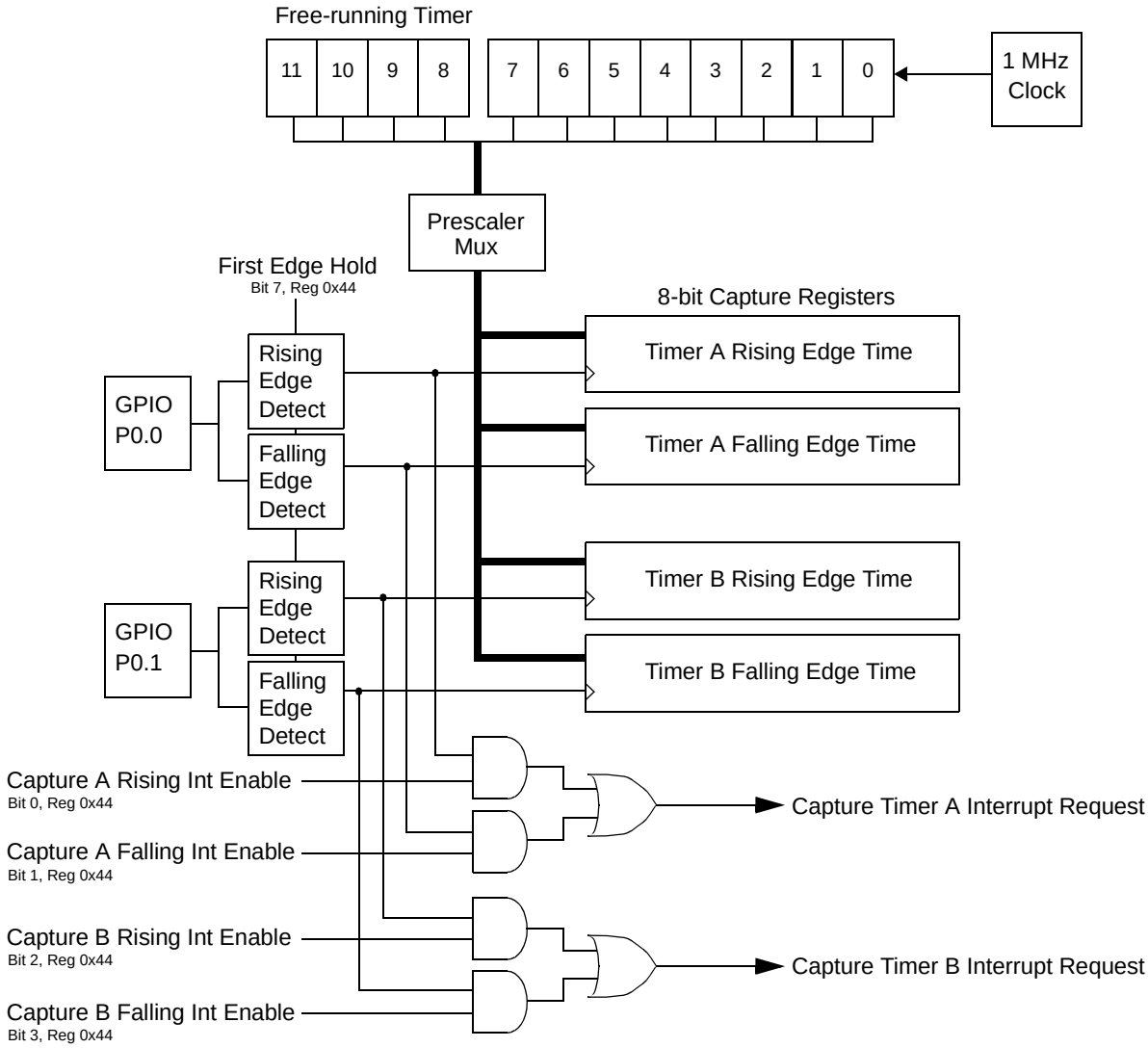


Figure 19-1. Capture Timers Block Diagram

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

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.

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

1 = Rising GPIO edge

0 = Falling GPIO edge

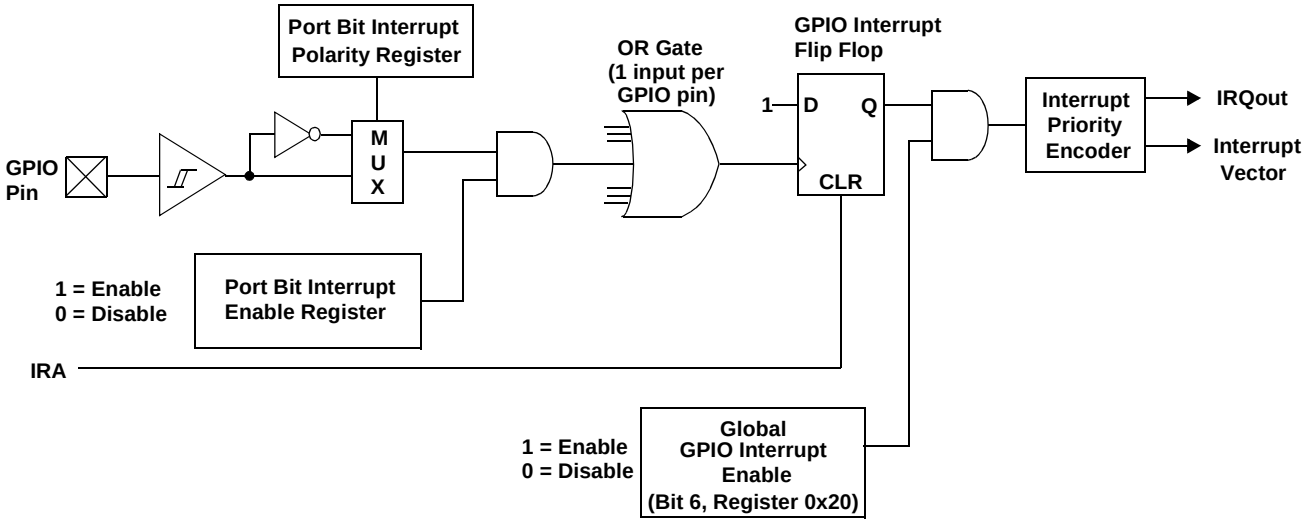


Figure 21-8. GPIO Interrupt Diagram

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:

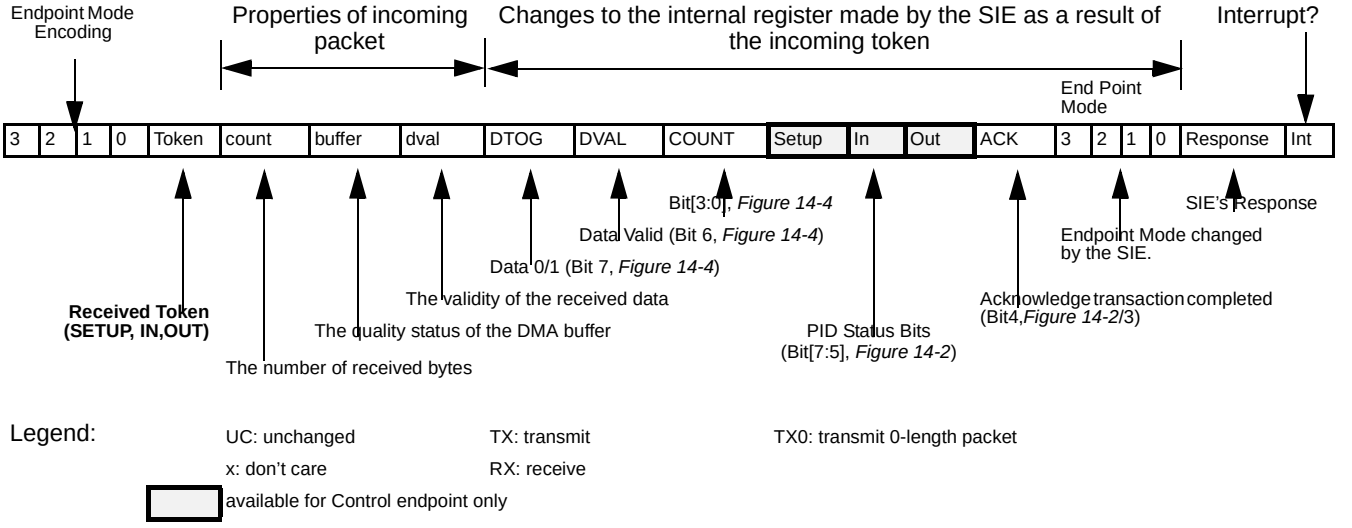
Some Mode Bits are automatically changed by the SIE in response to many USB transactions. For example, if the Mode Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode as shown in *Table 22-1*, the SIE will change the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACKing a valid 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"



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.

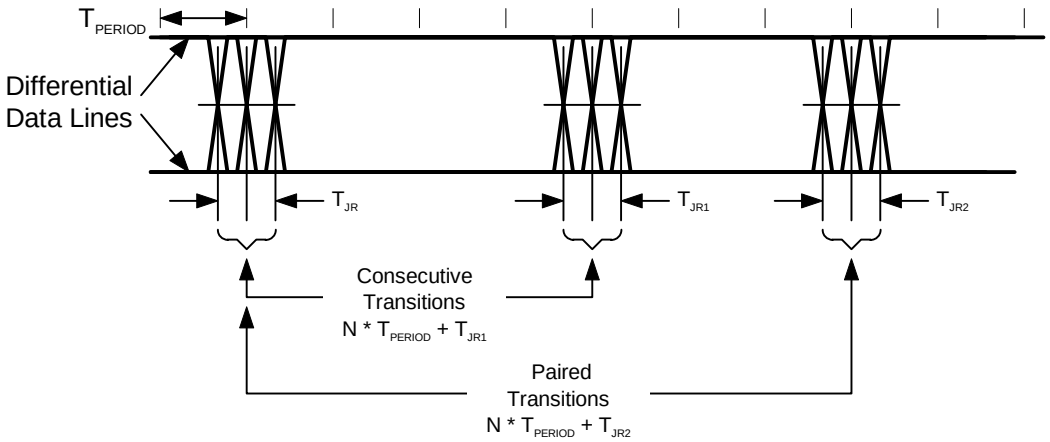


Figure 26-3. Receiver Jitter Tolerance

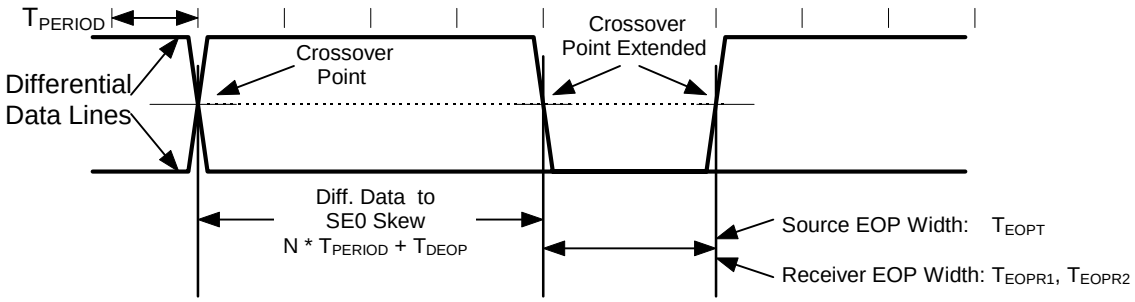


Figure 26-4. Differential to EOP Transition Skew and EOP Width

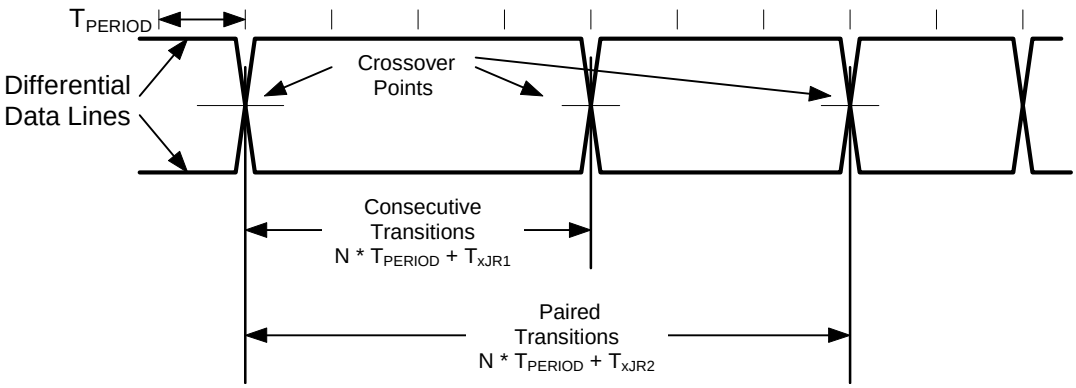


Figure 26-5. Differential Data Jitter

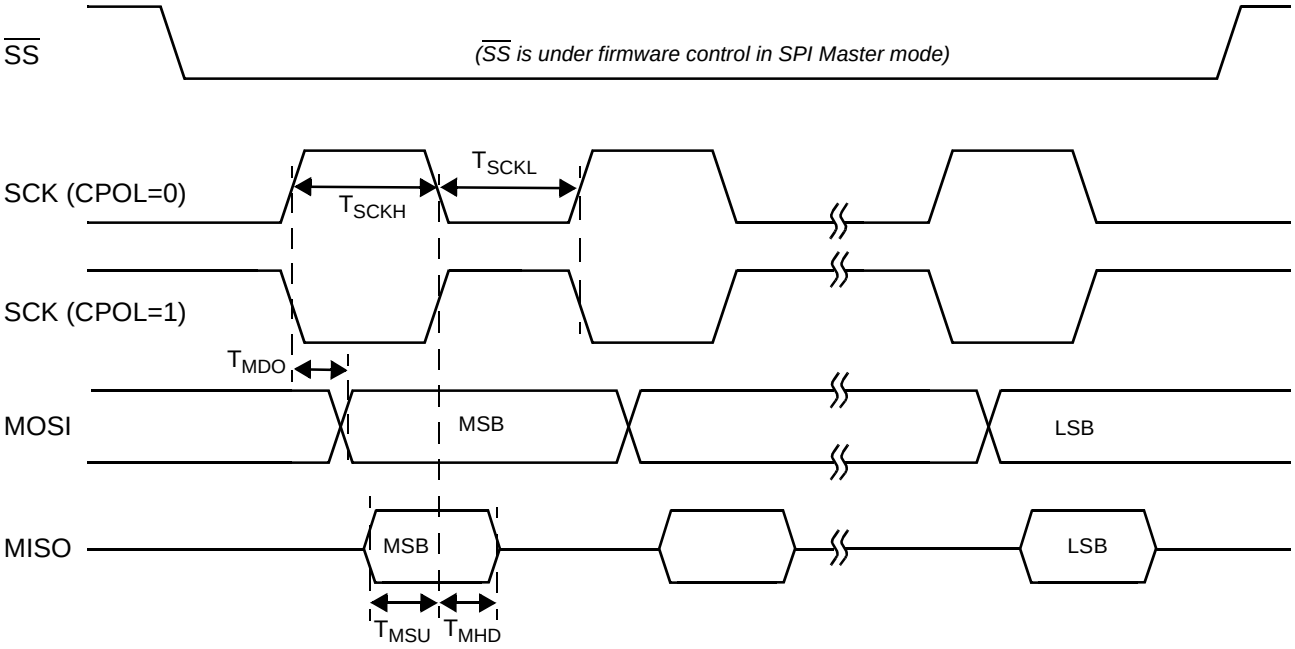


Figure 26-6. SPI Master Timing, CPHA = 0

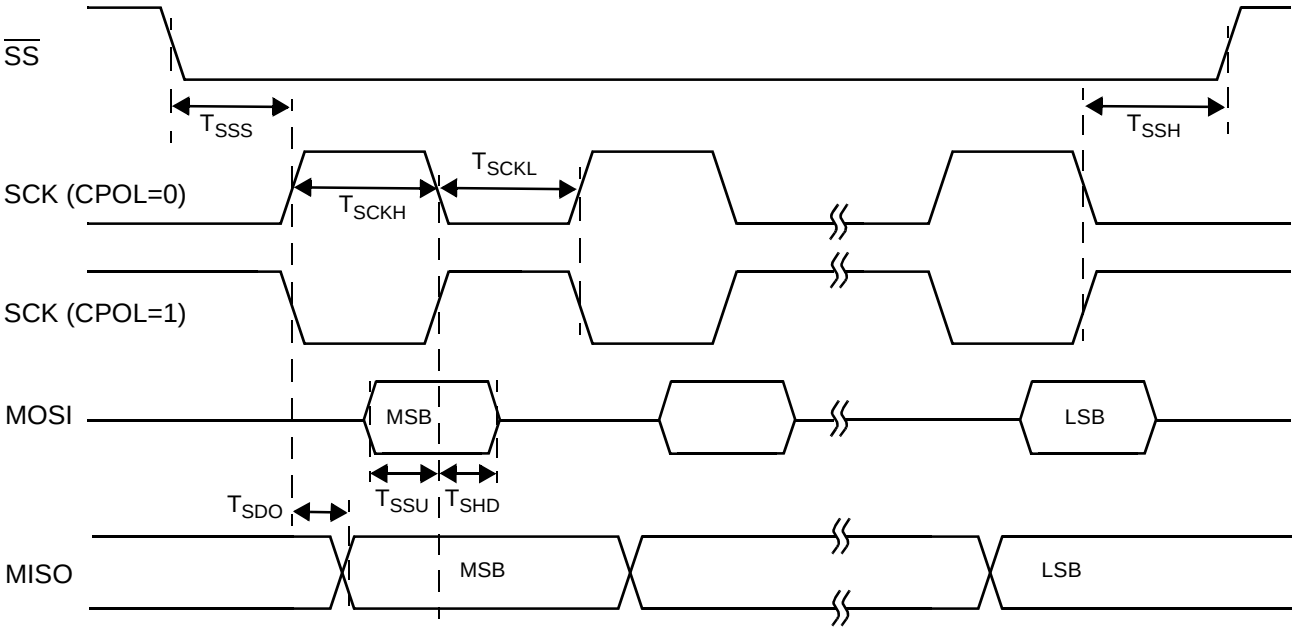


Figure 26-7. SPI Slave Timing, CPHA = 0

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

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