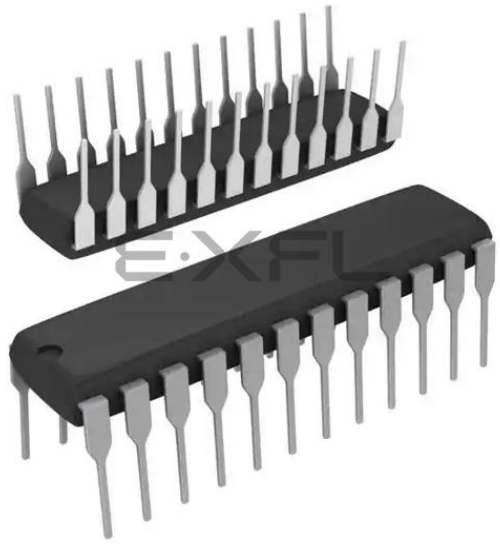


Welcome to E-XFL.COM

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	3.5V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Through Hole
Package / Case	24-DIP (0.300", 7.62mm)
Supplier Device Package	24-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c63743c-pxc

Contents

Functional Overview

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 3.3 V regulator. All of this adds up to a lower system cost.

The CY7C637xxC 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 CY7C637xxC 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 CY7C637xxC 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 CY7C637xxC 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 [1].

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 CY7C637xxC 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.

Note
1. Errata: When a falling edge interrupt is enabled for a GPIO pin, reading the GPIO Port 1 coincident to a rising edge of that GPIO signal may generate a false GPIO interrupt. In similar manner when a rising edge interrupt is enabled for a GPIO pin, reading the GPIO Port 1 coincident to a falling edge of that GPIO signal may generated a false GPIO interrupt. For more information, see the “Errata” on page 53.

Programming Model

Refer to the *CYASM Assembler User's Guide* for more details on firmware operation with the CY7C637xxC microcontrollers.

Program Counter (PC)

The 14-bit program counter (PC) allows access for up to 8 Kbytes of EPROM using the CY7C637xxC 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.

8-bit Accumulator (A)

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

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.

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 re-enable interrupts.

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

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

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.

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 Data Memory Organization on page 10 . 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
```

Address Modes

The CY7C637xxC microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

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

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]

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.

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

page 44 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 Low-voltage Reset (LVR) on page 14 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 Internal/External Oscillator Operation on page 13.

0 = Enable the internal oscillator.

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 Suspend Mode on page 15 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 13 on page 19 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.

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 3 on page 12. 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.

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 34 on page 30). Bits 4 (Low-voltage or Brown-out Reset bit) and 6 (Watchdog Reset bit) are used to

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 Clocking Mode on Wake-up from Suspend on page 15), 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 Clocking Mode on Wake-up from Suspend on page 15), firmware execution resumes.

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 35 on page 32) 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 4 on page 12). 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 2. Wake-up Timer Adjust Settings

Adjust Bits [2:0] (Bits [6:4] in Figure 4 on page 12)	Wakeup 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 Switching Characteristics on page 44 for the value of t_{WAKE}	

Figure 13. Port 2 Data Register (Address 0x02)

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

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 Internal/External Oscillator Operation on page 13 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. USB Regulator Output on page 22 for more details.

1 = Port Pin is logic HIGH

0 = Port Pin is logic LOW

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.

USB Enumeration

A typical USB enumeration sequence is shown below. In this description, ‘Firmware’ refers to embedded firmware in the CY7C637xxC 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.

USB Port Status and Control

USB status and control is regulated by the USB Status and Control Register as shown in Figure 14.

Figure 14. USB Status and Control Register (Address 0x1F)

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

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 15 on page 21).

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 DC Characteristics on page 42 for the value of R_{PS2}).

0 = Disable PS/2 Pull-up resistors.

Bit 6: V_{REG} Enable

A 3.3 V 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 19 on page 23).

1 = Enable the 3.3 V 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 Interrupt Sources on page 32 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 4. 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 4. 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[3]
101	Force D– LOW, D+ HiZ	
110	Force D– HiZ, D+ LOW	
111	Force D– HiZ, D+ HiZ	

USB Device

The CY7C637xxC supports one USB Device Address with three endpoints: EP0, EP1, and EP2.

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 15 shows the format of the USB Address Register.

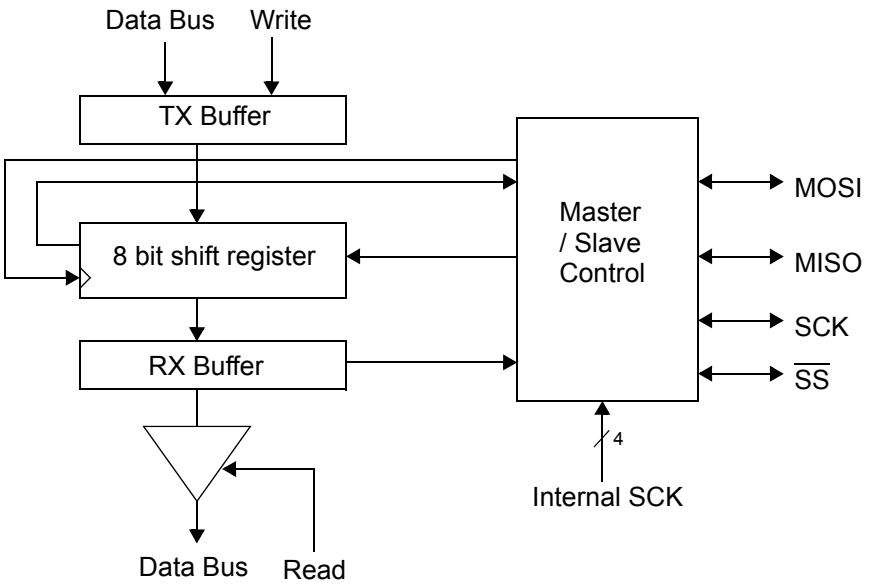
Note
3. For PS/2 operation, the D+/D– Forcing Bit [2:0] = 111b mode must be set initially (one time only) before using the other PS/2 force modes

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 CY7C637xxC SPI circuit supports byte serial transfers in either Master or Slave modes. The block diagram of the SPI circuit is shown in Figure 20. The block contains buffers for both transmit and receive data for maximum flexibility and throughput. The CY7C637xxC 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 20. SPI Block Diagram



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

Figure 21. SPI Data Register (Address 0x60)

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

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

Operation as an SPI Master

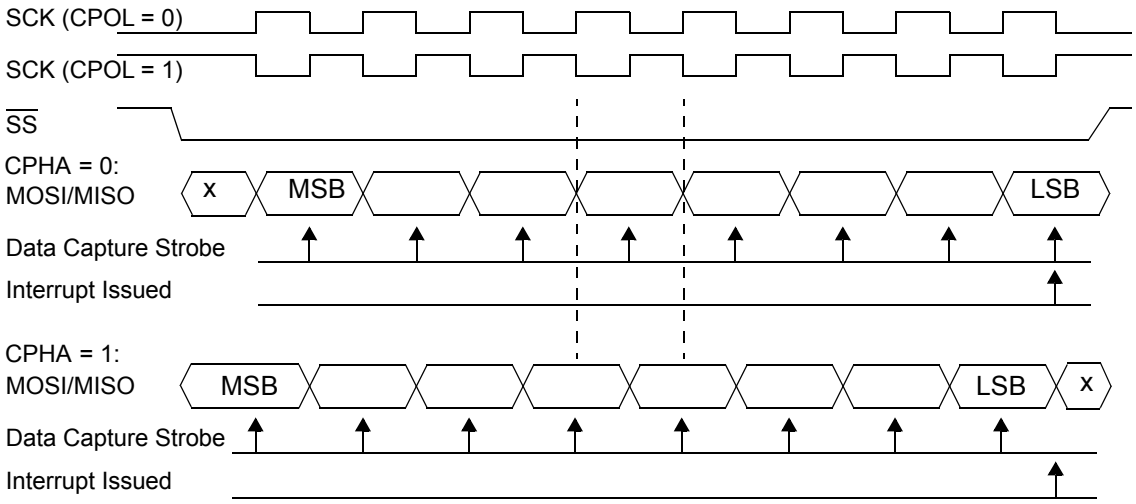
Only an SPI Master can initiate a byte/data transfer. This is done by the Master writing to the SPI Data Register. The Master shifts out 8 bits of data (MSB first) along with the serial clock SCK for the Slave. The Master's outgoing byte is replaced with an incoming one from a Slave device. When the 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 General Purpose I/O Ports on page 17 for GPIO configuration details.

Figure 23. SPI Data Timing



SPI Interrupt

For SPI, an interrupt request is generated after a byte is received or transmitted. See Interrupt Sources on page 32 for details on the SPI interrupt.

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 6 on page 17). 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 6 on page 17 is always 1, for normal GPIO operation.

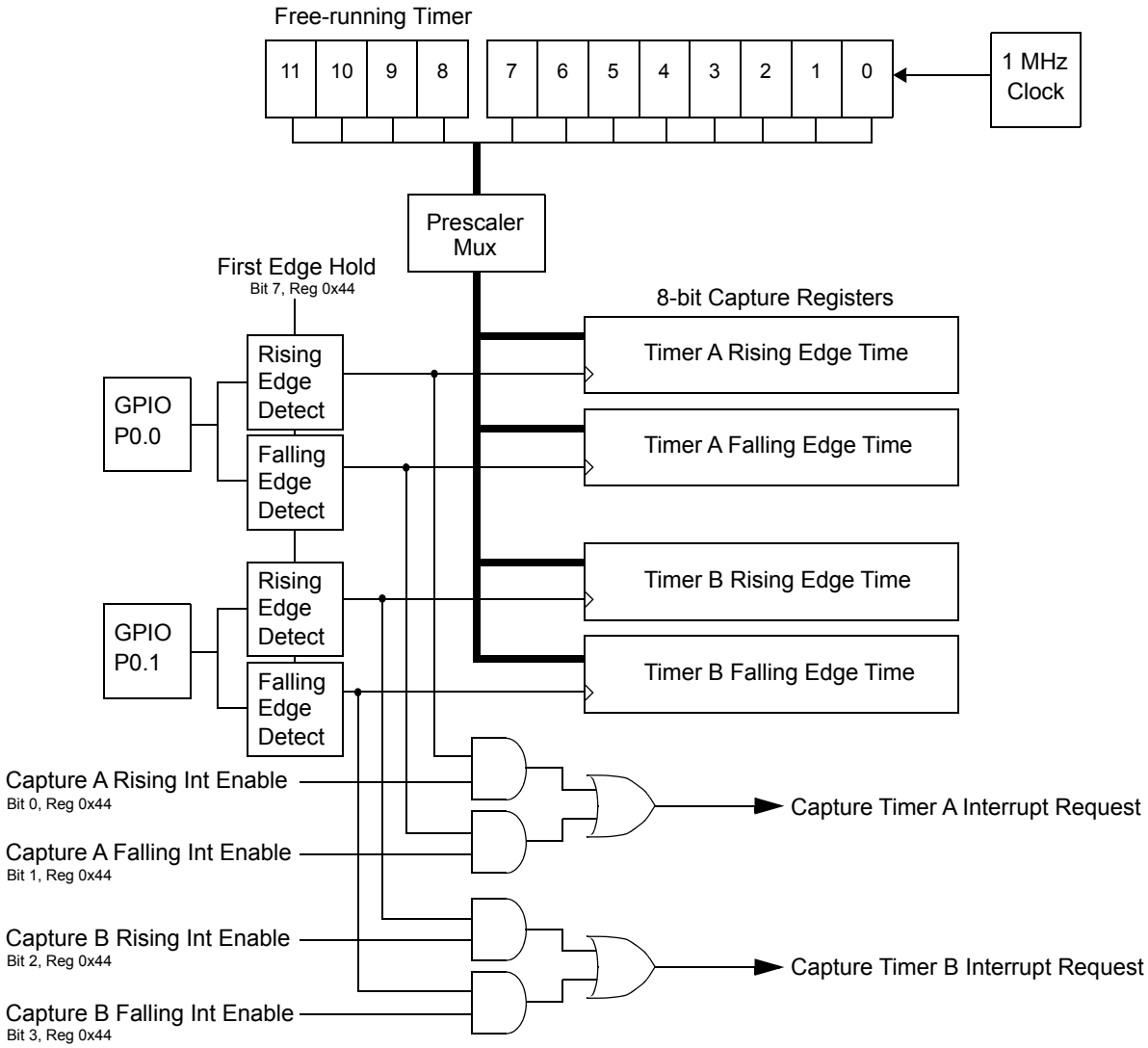
Table 5. SPI Pin Assignments

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.

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 27.

Figure 27. Capture Timers Block Diagram



0 = Disable interrupt

Table 6. 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

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

Processor Status and Control Register

Figure 34. Processor Status and Control Register (Address 0xFF)

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

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 35 and Figure 36) 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 Switching Characteristics on page 44 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 14). The details on the event conditions that set this bit are given in Interrupt Sources on page 32. 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 Suspend Mode on page 15 for more details on suspend mode operation.

Table 10. Details of Modes for Differing Traffic Conditions (continued)

OUT Endpoint																	
ACK OUT, STALL Bit = 0 (Figure 17)																	
1	0	0	1	OUT	<= 10	data	valid	updates	1	updates	UC	UC	1	1	1000	ACK	yes
1	0	0	1	OUT	> 10	junk	x	updates	updates	updates	UC	UC	1	UC	No-Change	Ignore	yes
1	0	0	1	OUT	x	junk	invalid	updates	0	updates	UC	UC	1	UC	No-Change	Ignore	yes
1	0	0	1	IN	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
ACK OUT, STALL Bit = 1 (Figure 17)																	
1	0	0	1	OUT	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	No-Change	STALL	yes
1	0	0	1	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	0	0	1	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	0	0	1	IN	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
NAK OUT																	
1	0	0	0	OUT	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	No-Change	NAK	yes
1	0	0	0	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	0	0	0	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	0	0	0	IN	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
Reserved																	
0	1	0	1	OUT	x	updates	updates	updates	updates	updates	UC	UC	1	1	No-Change	RX	yes
0	1	0	1	IN	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
IN Endpoint																	
ACK IN, STALL Bit = 0 (Figure 17)																	
1	1	0	1	OUT	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	1	0	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	1	1100	ACK (back)	yes
ACK IN, STALL Bit = 1 (Figure 17)																	
1	1	0	1	OUT	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	1	0	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	No-Change	STALL	yes
NAK IN																	
1	1	0	0	OUT	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	1	0	0	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	No-Change	NAK	yes
Reserved																	
0	1	1	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	1	1	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	No-Change	TX	yes

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)[13]	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[14]	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%[5])	75		ns
T_R	Transition Rise Time	CLoad = 600 pF (10% to 90%[5])		300	ns
T_F	Transition Fall Time	CLoad = 200 pF (10% to 90%[5])	75		ns
T_F	Transition Fall Time	CLoad = 600 pF (10% to 90%[5])		300	ns
T_{RFM}	Rise/Fall Time Matching	t_r/t_f[5, 15]	80	125	%
V_{CRS}	Output Signal Crossover Voltage[19]	CLoad = 200 to 600 pF[5]	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[16]	–75	75	ns
T_{DJR2}	Receiver Data Jitter Tolerance	For Paired Transitions[16]	–45	45	ns
T_{DEOP}	Differential to EOP transition Skew	Note 16	–40	100	ns
T_{EOPR2}	EOP Width at Receiver	Accepts as EOP[16]	670		ns
T_{EOPT}	Source EOP Width		1.25	1.50	µs
T_{UDJ1}	Differential Driver Jitter	To next transition, Figure 46	–95	95	ns
T_{UDJ2}	Differential Driver Jitter	To paired transition, Figure 46	–150	150	ns
T_{LST}	Width of SE0 during Diff. Transition			210	ns
	Non-USB Mode Driver Characteristics	Note 17			
T_{FPS2}	SDATA/SCK Transition Fall Time	CLoad = 150 pF to 600 pF	50	300	ns
	SPI Timing	See Figures 47 to 50[18]			
T_{SMCK}	SPI Master Clock Rate	F_{CLK}/3; see Figure 20		2	MHz
T_{SSCK}	SPI Slave Clock Rate			2.2	MHz

Notes
13. Initially $F_{ICLK2} = F_{ICLK}$ until a USB packet is received.
14. Wake-up time for Wake-up Adjust Bits cleared to 000b (minimum setting)
15. Tested at 200 pF.
16. Measured at cross-over point of differential data signals.
17. 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.
18. SPI timing specified for capacitive load of 50 pF, with GPIO output mode = 01 (medium low drive, strong high drive).
19. Per the USB 2.0 Specification, Table 7.7, Note 10, the first transition from the Idle state is excluded.

Figure 49. SPI Master Timing, CPHA = 1

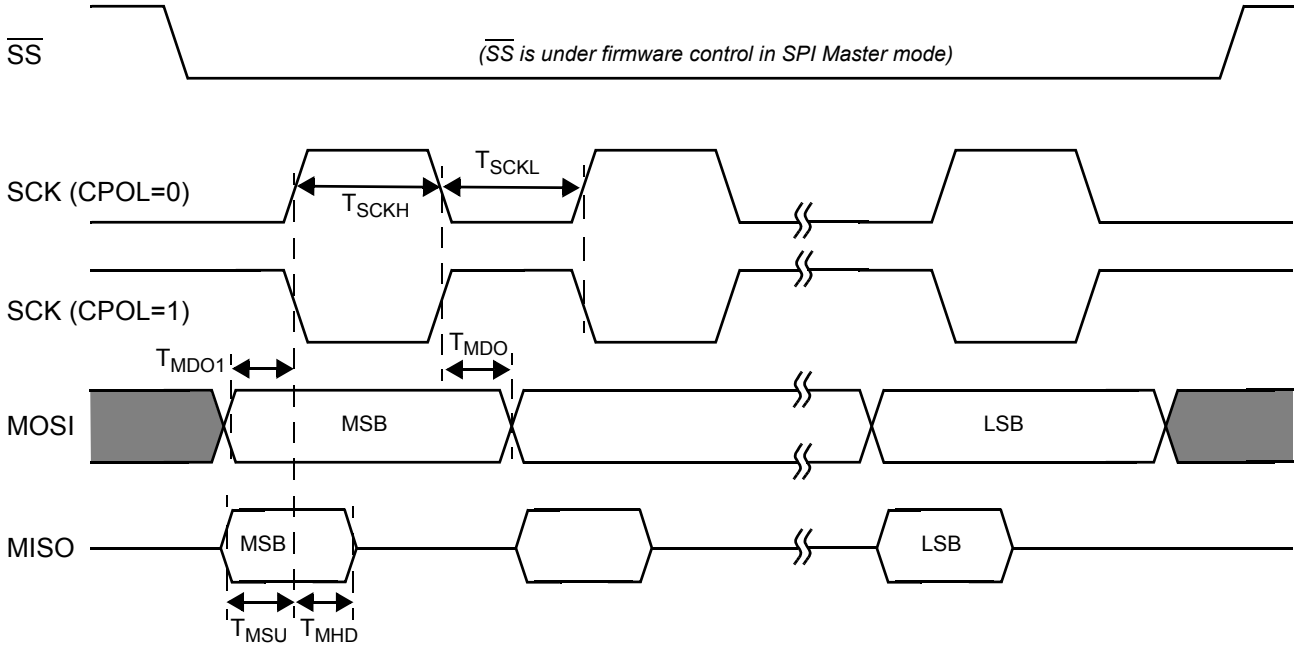
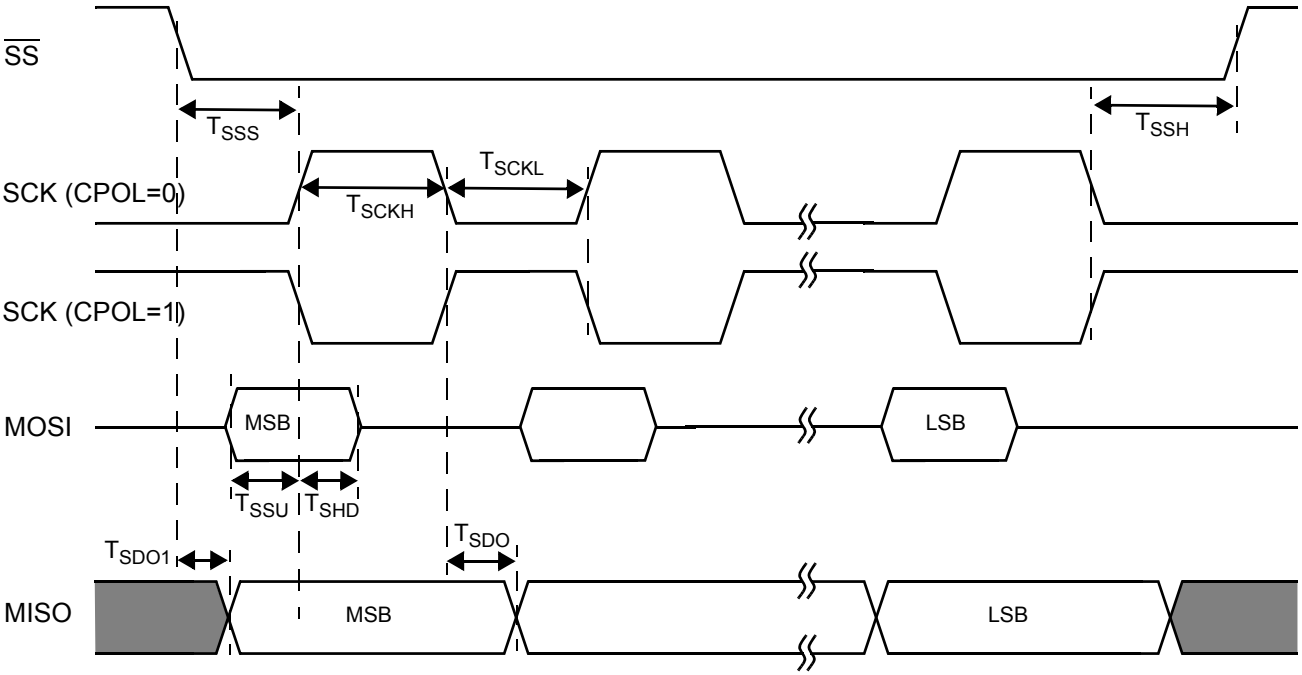


Figure 50. SPI Slave Timing, CPHA = 1

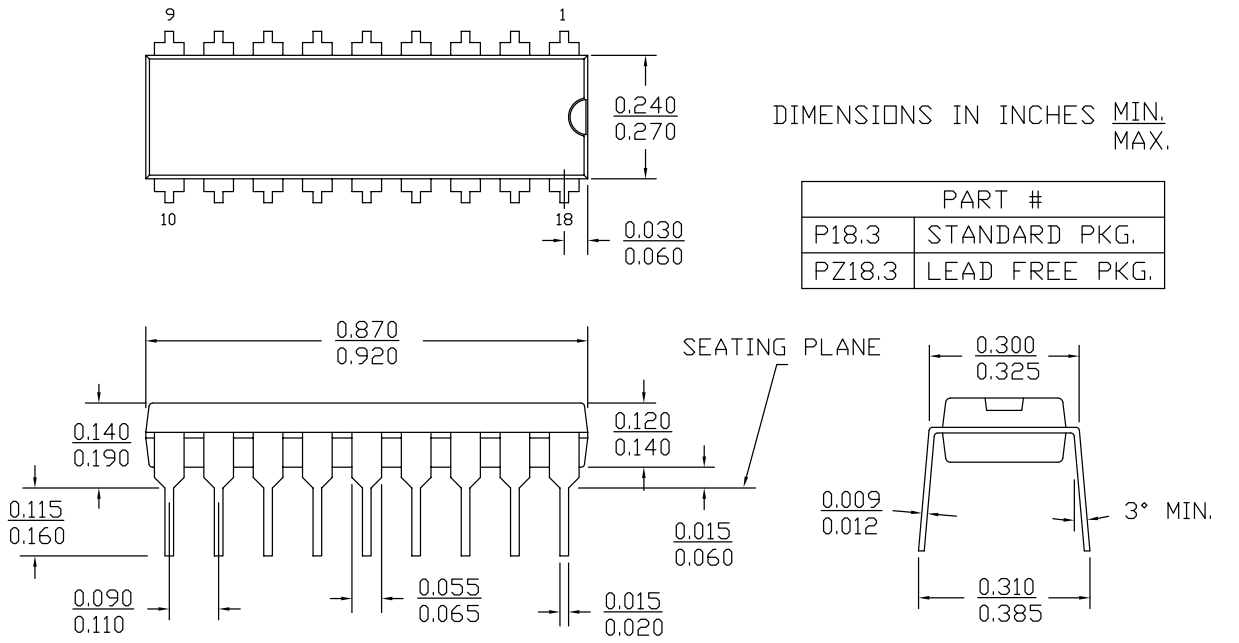


Ordering Information

Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C63723C-PXC	8 KB	P3	18-Pin (300-Mil) Pb-free PDIP	Commercial
CY7C63723C-SXC	8 KB	S3	18-Pin Small Outline Pb-free Package	Commercial
CY7C63743C-PXC	8 KB	P13	24-Pin (300-Mil) Pb-free PDIP	Commercial
CY7C63743C-SXC	8 KB	S13	24-Pin Small Outline Pb-free Package	Commercial
CY7C63743C-QXC	8 KB	Q13	24-Pin QSOP Pb-free Package	Commercial
CY7C63722C-XC	8 KB	–	25-Pad Die Form	Commercial
CY7C63743C-SXCT	8 KB	S13	24-Pin Small Outline Pb-free Package Tape-reel	Commercial
CY7C63723C-SXCT	8 KB	S3	18-Pin Small Outline Pb-free Package Tape-reel	Commercial

Package Diagrams

Figure 51. 18-pin PDIP (300-Mil) Molded DIP



51-85010 *E

Errata

This section describes the errata for the enCoRe™ USB Combination Low-speed USB & PS/2 Periheral Controller / CY7C637xx. The details include errata trigger conditions, available workaround, and silicon revision applicability.

Please contact your local Cypress Sales Representative if you have further questions.

Part Numbers Affected

Part Number	Device Characteristics
CY7C63722	All packages
CY7C63723	All packages
CY7C63743	All packages

enCoRe™ USB Combination Low-speed USB & PS/2 Peripheral Controller Qualification Status

Product status: In Production - Qual report: 001406

enCoRe™ USB Combination Low-speed USB & PS/2 Peripheral Controller Errata Summary

The following table defines the errata applicability to available enCoRe™ USB Combination Low-speed USB & PS/2 Peripheral Controller family devices. An "X" indicates that the errata pertains to the selected device.

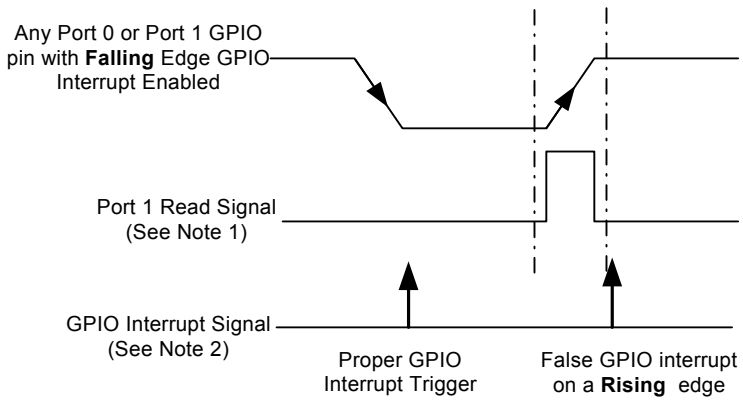
Note: Errata titles are hyperlinked. Click on table entry to jump to description.

Items	CY7C637xx	Rev Letter	Fix Status
1. Faulty GPIO Interrupt	X	A	No silicon fix planned.

1. Faulty GPIO Interrupt

■ **Problem Definition**

When a falling edge interrupt is enabled for a GPIO pin, reading the GPIO Port 1 coincident to a rising edge of that GPIO signal may generate a false GPIO interrupt.



When a rising edge interrupt is enabled for a GPIO pin, reading the GPIO Port 1 coincident to a falling edge of that GPIO signal may generated a false GPIO intterupt.

Document History Page

Document Title: CY7C63722C/CY7C63723C/CY7C63743C, 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
*C	429169	See ECN	TYJ	Updated part numbers with 'C' part numbers Changed to 'Cypress Perform' logo Added the 24-QSOP part offering
*D	3057657	10/13/2010	AJHA	Added "Not recommended for new designs" watermark in the PDF. Updated package diagrams. Updated template.
*E	3229083	04/15/2011	NXZ	Package diagram updated 51-85025 *E Completing Sunset Review.
*F	3593602	04/20/2012	ANTG	Added Not recommended for New Designs" watermark in the PDF. Updated cross-references for figures and sections throughout the datasheet. Updated Package Diagrams 51-85010, 51-85023, 51-85013, and 51-85055 (from Rev *C to *D). Added Table of Contents.
*G	3997628	05/11/2013	SELV	Updated Package Diagrams: spec 51-85010 – Changed revision from *D to *E. Added Errata.
*H	4072605	07/22/2013	SELV	Added Errata footnote (Note 1). Updated Functional Overview: Updated enCoRe USB—The New USB Standard: Added Note 1 and referred the same note at the end of the sentence "The interrupt polarity can be either rising or falling edge". Updated to new template.
*I	4313900	03/21/2014	AKSL	Added CY7C63743C-SXCT and CY7C63723C-SXCT in Ordering Information. Updated 24-pin SOIC package diagram. Removed "Not recommended for new designs" watermark.
*J	4910453	09/07/2015	KISB	Updated DC Characteristics: Changed maximum value of V_{OHZ} parameter from 3.6 V to 3.8 V. Updated Package Diagrams: spec 51-85023 – Changed revision from *D to *E. spec 51-85013 – Changed revision from *D to *E. spec 51-85055 – Changed revision from *D to *E. Updated to new template.
*K	5705418	04/21/2017	AESATMP8	Updated logo and Copyright.

Sales, Solutions, and Legal Information

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support