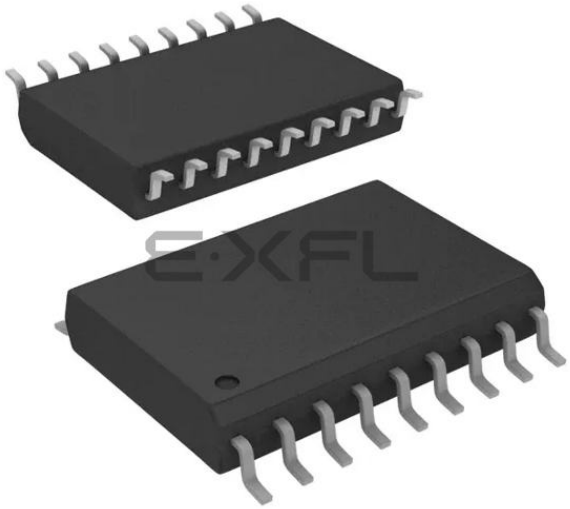


Welcome to E-XFL.COM

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	10
Voltage - Supply	3.5V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	18-SOIC (0.295", 7.50mm Width)
Supplier Device Package	18-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/flip-electronics/cy7c63723c-sxc

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.

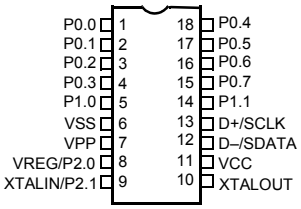
The USB D+ and D– USB pins can alternately be used as PS/2 SCLK and SDATA signals, so that products can be designed to respond to either USB or PS/2 modes of operation. PS/2 operation is supported with internal pull-up resistors on SCLK and SDATA, the ability to disable the regulator output pin, and 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.

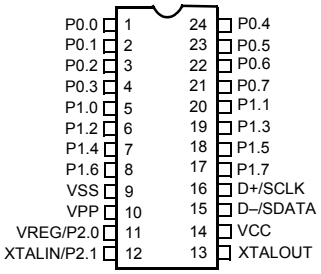
Pin Configurations

Top View

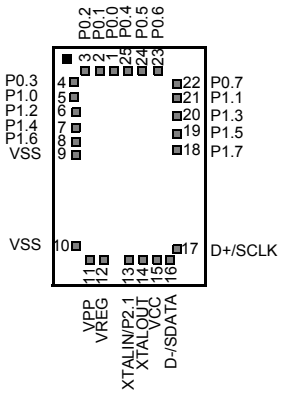
CY7C63723C
18-pin SOIC/PDIP



CY7C63743C
24-pin SOIC/PDIP/QSOP



CY7C63722C-XC
DIE



Pin Definitions

Name	I/O	CY7C63723C	CY7C63743C	CY7C63722C	Description
		18-Pin	24-Pin	25-Pad	
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

■ 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

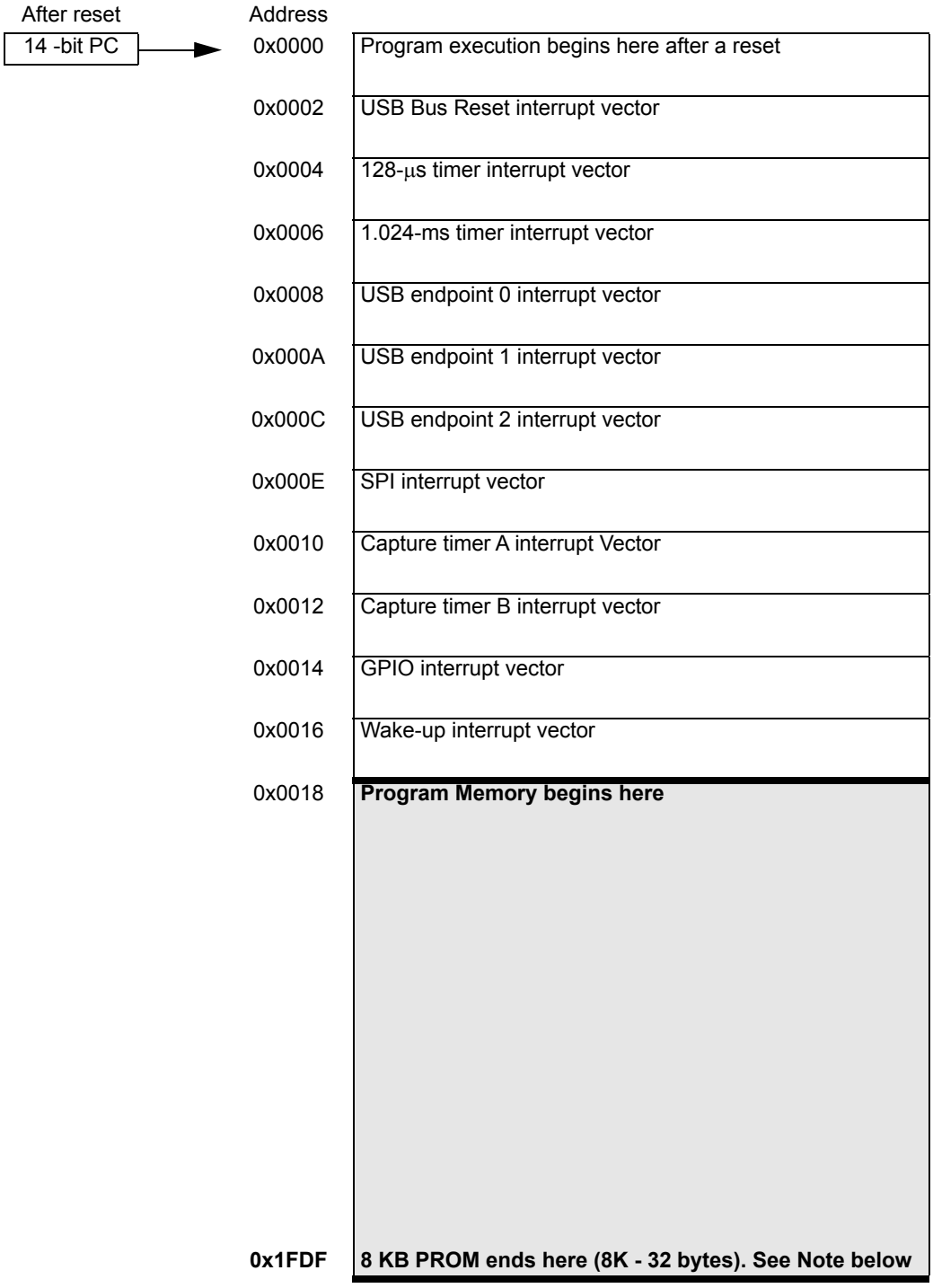
Refer to the *CYASM Assembler User’s Guide* for detailed information on these instructions. Note that conditional jump instructions (i.e., JC, JNC, JZ, JNZ) take five cycles if jump is taken, four cycles if no jump.

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

Memory Organization

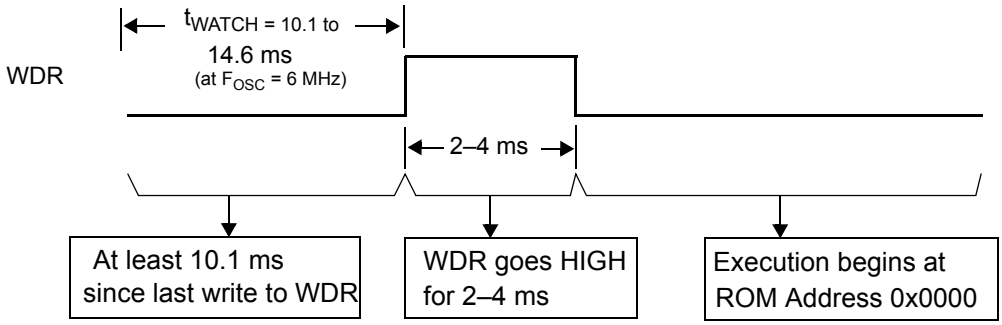
Program Memory Organization[2]

Figure 1. Program Memory Space with Interrupt Vector Table



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

Figure 5. Watchdog Reset (WDR, Address 0x26)



Suspend Mode

The CY7C637xxC parts support a versatile low-power suspend mode. In suspend mode, only an enabled interrupt or a LOW state on the D–/SDATA pin will wake the part. Two options are available. For lowest power, all internal circuits can be disabled, so only an external event will resume operation. Alternatively, a low-power internal wake-up timer can be used to trigger the wake-up interrupt. This timer is described in Wake-up Timer on page 16, and can be used to periodically poll the system to check for changes, such as looking for movement in a mouse, while maintaining a low average power.

The CY7C637xxC is placed into a low-power state by setting the Suspend bit of the Processor Status and Control Register (Figure 34). All logic blocks in the device are turned off except the GPIO interrupt logic, the D–/SDATA pin input receiver, and (optionally) the wake-up timer. The clock oscillators, as well as the free-running and Watchdog timers are shut down. Only the occurrence of an enabled GPIO interrupt, wake-up interrupt, SPI slave interrupt, or a LOW state on the D–/SDATA pin will wake the part from suspend (D– LOW indicates non-idle USB activity). Once one of these resuming conditions occurs, clocks will be restarted and the device returns to full operation after the oscillator is stable and the selected delay period expires. This delay period is determined by selection of internal versus external clock, and by the state of the Ext. Clock Resume Delay as explained in Clocking Mode on Wake-up from Suspend on page 15.

In suspend mode, any enabled and pending interrupt will wake the part up. The state of the Interrupt Enable Sense bit (Bit 2, Figure 34 on page 30) does not have any effect. As a result, any interrupts not intended for waking from suspend should be disabled through the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register (Interrupts).

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

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

To achieve the lowest possible current during suspend mode, all I/O should be held at either V_{CC} or ground. In addition, the GPIO *bit* interrupts (Figure on page 34 and Figure on page 34) should be disabled for any pins that are not being used for a wake-up interrupt. This should be done even if the main GPIO Interrupt Enable (Figure 35 on page 32) is off.

Typical code for entering suspend is shown below:

```
...             ; All GPIO set to low-power state (no floating
                  pins, and bit interrupts disabled unless using
                  for wake-up)
...             ; Enable GPIO and/or wake-up timer
                  interrupts if desired for wake-up
...             ; Select clock mode for wake-up (see Clocking
                  Mode on Wake-up from Suspend on page
                  15)
mov a, 09h      ; Set suspend and run bits
iowr FFh        ; Write to Status and Control Register – Enter
                  suspend, wait for GPIO/wake-up interrupt or
                  USB activity
nop             ; This executes before any ISR
...             ; Remaining code for exiting suspend routine
```

Clocking Mode on Wake-up from Suspend

When exiting suspend on a wake-up event, the device can be configured to run in either Internal or External Clock mode. The mode is selected by the state of the External Oscillator Enable bit in the Clock Configuration Register (Figure 4 on page 12). Using the Internal Clock saves the external oscillator start-up time and keeps that oscillator off for additional power savings. The external oscillator mode can be activated when desired, similar to operation at power-up.

The sequence of events for these modes is as follows:

Wake in Internal Clock Mode:

1. Before entering suspend, clear bit 0 of the Clock Configuration Register. This selects Internal clock mode after suspend.
2. Enter suspend mode by setting the suspend bit of the Processor Status and Control Register.
3. After a wake-up event, the internal clock starts immediately (within 2 µs).
4. A time-out period of 8 µs passes, and then firmware execution begins.

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.

USB signaling in the case where the VREG pin is used as an input, and an external regulator is provided for the USB pull-up resistor. This also limits the swing on the D– and D+ pins to about 1V above the internal regulator voltage, so the Device Address Enable bit normally should only be set for USB operating modes.

The regulator output is only designed to provide current for the USB pull-up resistor. In addition, the output voltage at the VREG pin is effectively disconnected when the CY7C637xxC device transmits USB from the internal SIE. This means that the VREG pin does not provide a stable voltage during transmits, although this does not affect USB signaling.

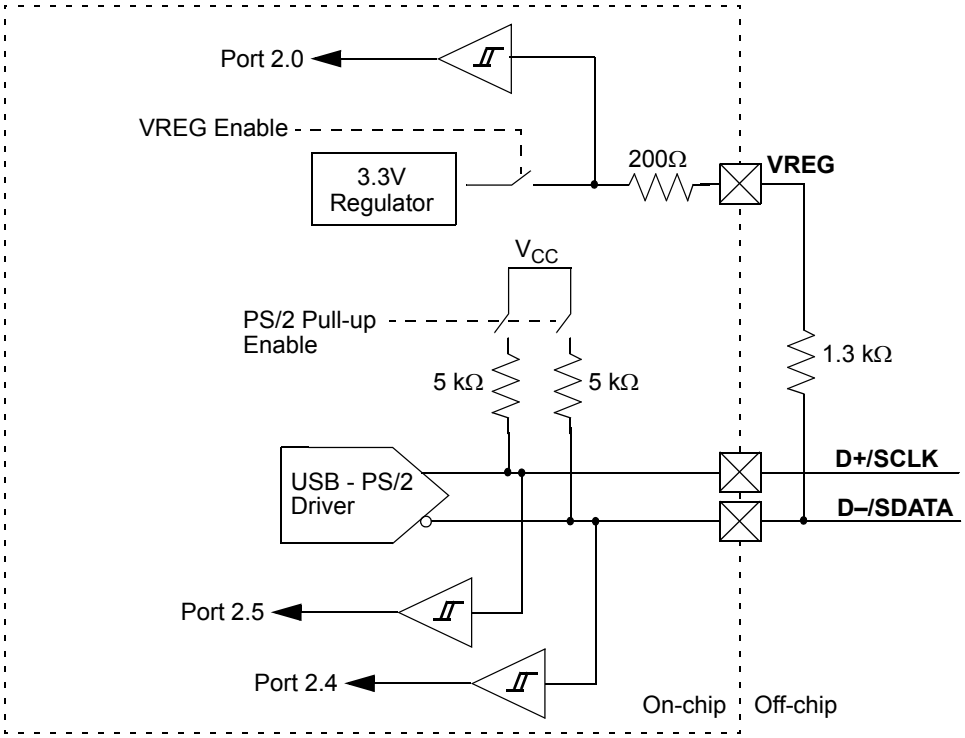
PS/2 Operation

The CY7C637xxC parts are optimized for combination USB or PS/2 devices, through the following features:

1. USB D+ and D– lines can also be used for PS/2 SCLK and SDATA pins, respectively. With USB disabled, these lines can be placed in a high-impedance state that will pull up to V_{CC}. (Disable USB by clearing the Address Enable bit of the USB Device Address Register, Figure 15).
2. An interrupt is provided to indicate a long LOW state on the SDATA pin. This eliminates the need to poll this pin to check for PS/2 activity. Refer to USB Port Status and Control on page 19 for more details.
3. Internal PS/2 pull-up resistors can be enabled on the SCLK and SDATA lines, so no GPIO pins are required for this task (bit 7, USB Status and Control Register, *Figure 14*).
4. The controlled slew rate outputs from these pins apply to both USB and PS/2 modes to minimize EMI.
5. The state of the SCLK and SDATA pins can be read, and can be individually driven LOW in an open drain mode. The pins are read at bits [5:4] of Port 2, and are driven with the Control Bits [2:0] of the USB Status and Control Register.
6. The V_{REG} pin can be placed into a high-impedance state, so that a USB pull-up resistor on the D–/SDATA pin will not interfere with PS/2 operation (bit 6, USB Status and Control Register).

The PS/2 on-chip support circuitry is illustrated in Figure 19.

Figure 19. Diagram of USB-PS/2 System Connections

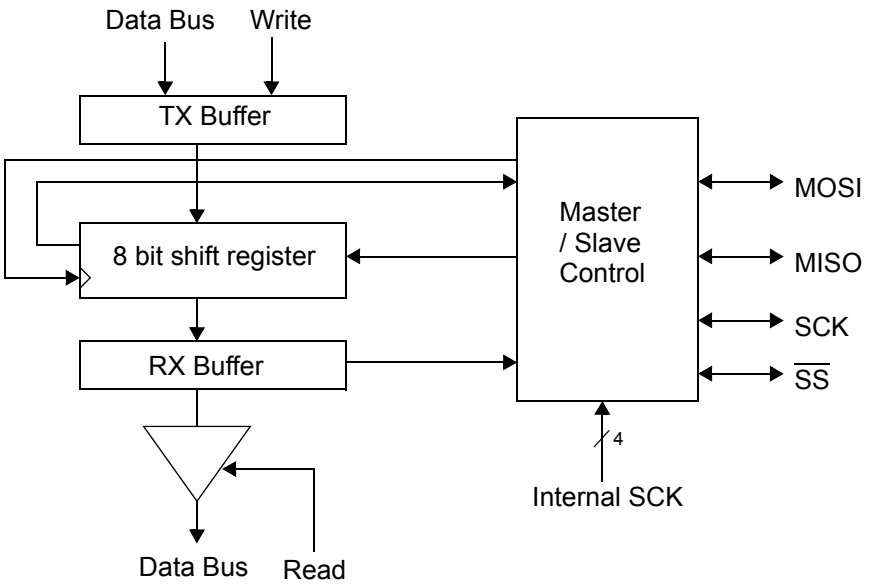


Serial Peripheral Interface (SPI)

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.

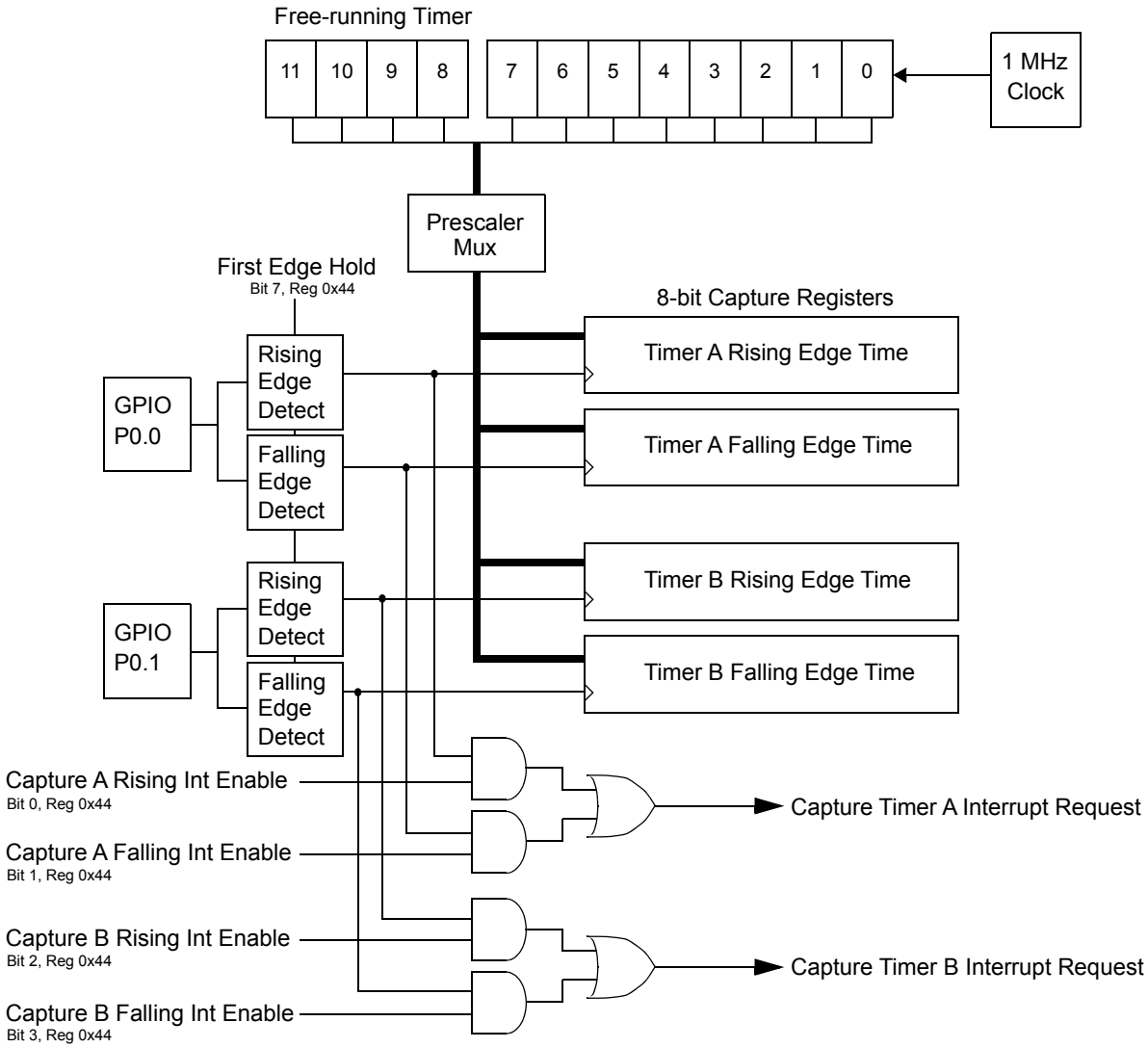
When the Master writes to the SPI Data Register, the data is loaded into the transmit buffer. If the shift register is not busy shifting a previous byte, the TX buffer contents will be automatically transferred into the shift register and shifting will begin. If the shift register is busy, the new byte 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.

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.

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 14). In either case, the interrupt will occur if the selected condition exists for 256 µs, and may occur as early as 128 µs.

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

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

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

1 = Enable

0 = Disable

Figure 36. Endpoint Interrupt Enable Register (Address 0x21)

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

Bit [7:3]: Reserved.

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

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

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

1 = Enable

0 = Disable

Refer to Table 8 for more information.

Bit 0: EP0 Interrupt Enable

If enabled, a control endpoint interrupt is generated when:

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

1 = Enable EP0 interrupt

0 = Disable EP0 interrupt

Figure 40. Port 1 Interrupt Polarity Register (Address 0x07)

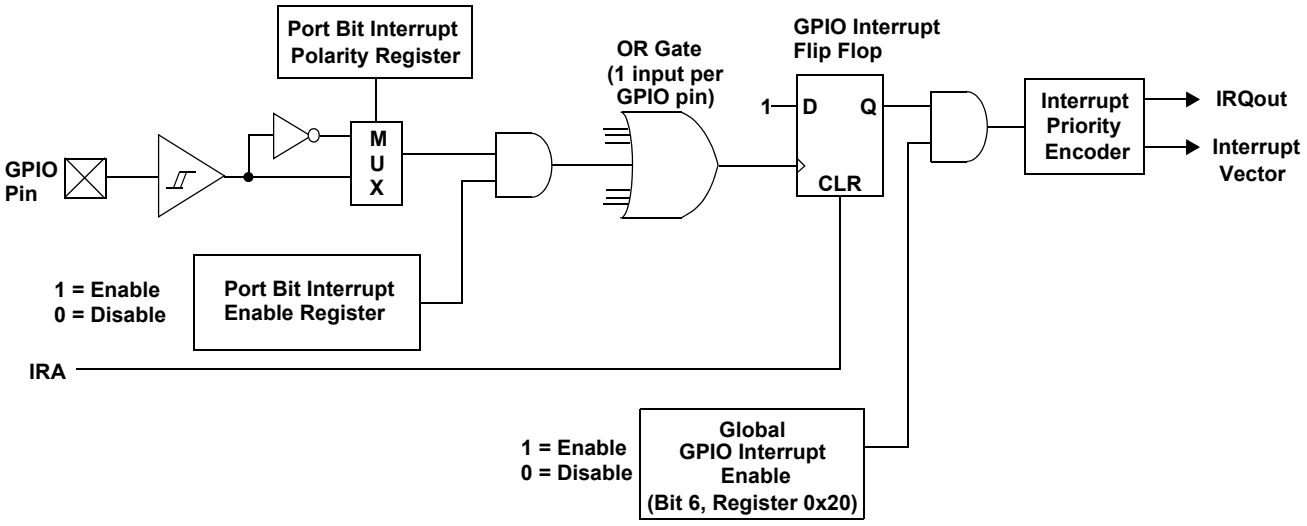
Bit #	7	6	5	4	3	2	1	0
Bit Name	P1 Interrupt Polarity							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

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

1 = Rising GPIO edge

0 = Falling GPIO edge

Figure 41. GPIO Interrupt Diagram



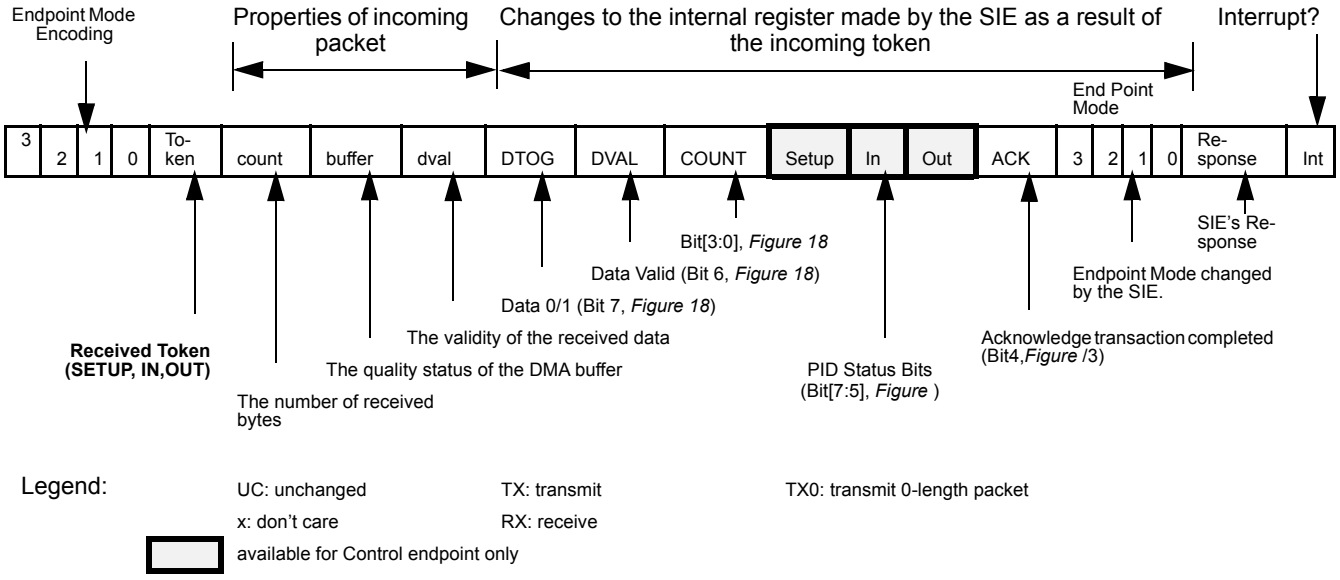
Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode as shown in Table 8, the SIE will change the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACKing a valid status stage OUT token. The firmware needs to update the mode for the SIE to respond appropriately. See Table 8 for 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 9. Decode table for Table 10: "Details of Modes for Differing Traffic Conditions"



The response of the SIE can be summarized as follows:

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

Table 10. Details of Modes for Differing Traffic Conditions

End Point Mode										PID				Set End Point Mode		
3	2	1	0	Rcved Token	Count	Buffer	Dval	DTOG	DVAL	COUNT	SET-UP	IN	OUT	ACK	3 2 1 0	Response	Int
SETUP Packet (if accepting)																	
See8				SETUP	<= 10	data	valid	updates	1	updates	1	UC	UC	1	0 0 0 1	ACK	yes
See8				SETUP	> 10	junk	x	updates	updates	updates	1	UC	UC	UC	No-Change	Ignore	yes
See 8				SETUP	x	junk	invalid	updates	0	updates	1	UC	UC	UC	No-Change	Ignore	yes
Disabled																	
0	0	0	0	x	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
NAK IN/OUT																	
0	0	0	1	OUT	x	UC	x	UC	UC	UC	UC	UC	1	UC	No-Change	NAK	yes
0	0	0	1	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	0	0	1	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	0	0	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	No-Change	NAK	yes
Ignore IN/OUT																	
0	1	0	0	OUT	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	1	0	0	IN	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
STALL IN/OUT																	
0	0	1	1	OUT	x	UC	x	UC	UC	UC	UC	UC	1	UC	No-Change	STALL	yes
0	0	1	1	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	0	1	1	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	0	1	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	No-Change	STALL	yes
Control Write																	
ACK OUT/NAK IN																	
1	0	1	1	OUT	<= 10	data	valid	updates	1	updates	UC	UC	1	1	0 0 0 1	ACK	yes
1	0	1	1	OUT	> 10	junk	x	updates	updates	updates	UC	UC	1	UC	No-Change	Ignore	yes
1	0	1	1	OUT	x	junk	invalid	updates	0	updates	UC	UC	1	UC	No-Change	Ignore	yes

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

1	0	1	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	No-Change	NAK	yes
NAK OUT/Status IN																	
1	0	1	0	OUT	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	No-Change	NAK	yes
1	0	1	0	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	0	1	0	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	0	1	0	IN	x	UC	x	UC	UC	UC	UC	1	UC	1	No-Change	TX 0 Byte	yes
Status IN Only																	
0	1	1	0	OUT	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	0	0	1	1	STALL	yes
0	1	1	0	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	1	1	0	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	1	1	0	IN	x	UC	x	UC	UC	UC	UC	1	UC	1	No-Change	TX 0 Byte	yes
Control Read																	
ACK IN/Status OUT																	
1	1	1	1	OUT	2	UC	valid	1	1	updates	UC	UC	1	1	No-Change	ACK	yes
1	1	1	1	OUT	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	STALL	yes
1	1	1	1	OUT	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	STALL	yes
1	1	1	1	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	1	1	1	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	1	1	1	IN	x	UC	x	UC	UC	UC	UC	1	UC	1	1	1	1	0	ACK (back)	yes
NAK IN/Status OUT																	
1	1	1	0	OUT	2	UC	valid	1	1	updates	UC	UC	1	1	No-Change	ACK	yes
1	1	1	0	OUT	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	STALL	yes
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	SETUP	IN	OUT	ACK	3	2	1	0	response	int
1	1	1	0	OUT	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	STALL	yes
1	1	1	0	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	1	1	0	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
1	1	1	0	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	No-Change	NAK	yes
Status OUT Only																	
0	0	1	0	OUT	2	UC	valid	1	1	updates	UC	UC	1	1	No-Change	ACK	yes
0	0	1	0	OUT	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	STALL	yes
0	0	1	0	OUT	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	STALL	yes
0	0	1	0	OUT	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	0	1	0	OUT	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No-Change	Ignore	no
0	0	1	0	IN	x	UC	x	UC	UC	UC	UC	1	UC	UC	0	0	1	1	STALL	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.

Switching Characteristics (continued)

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

Figure 42. Clock Timing

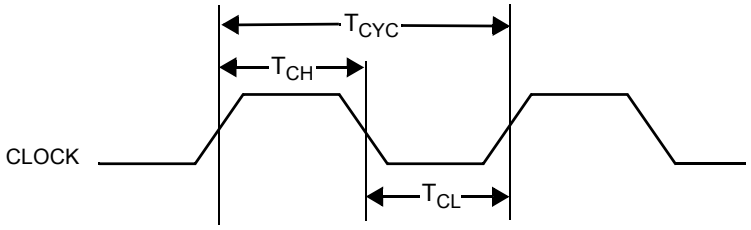


Figure 43. USB Data Signal Timing

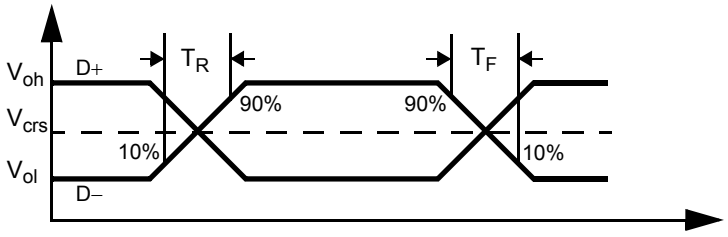


Figure 49. SPI Master Timing, CPHA = 1

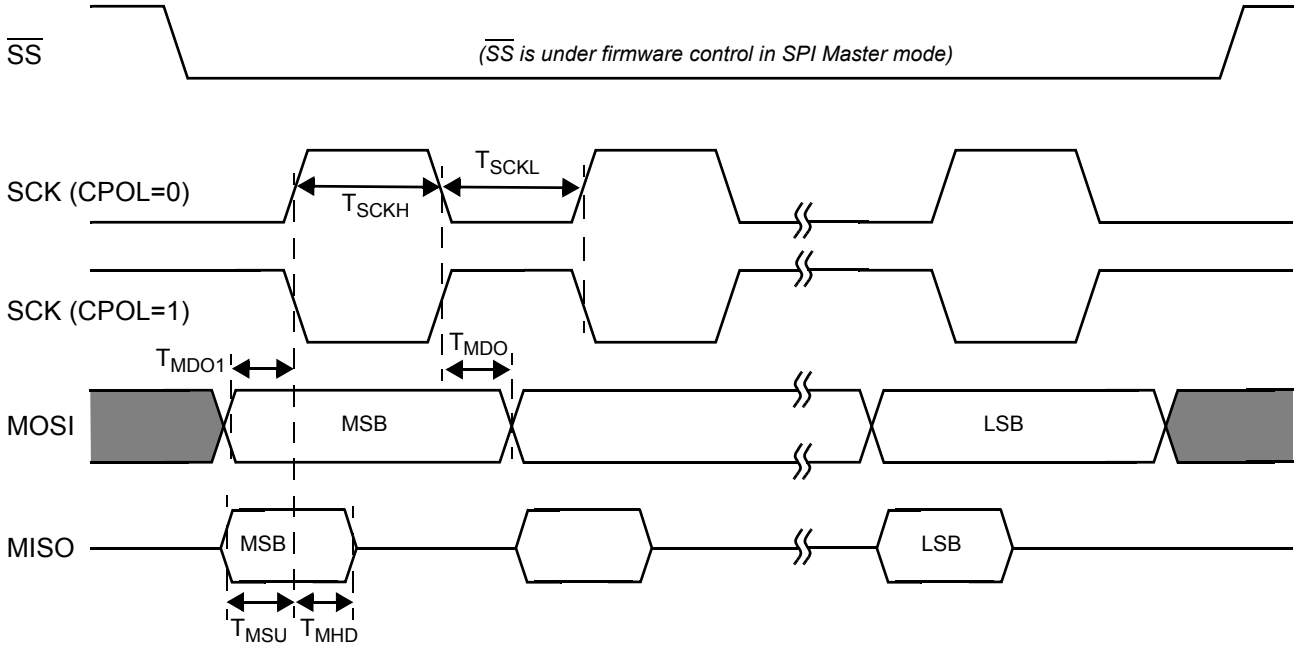
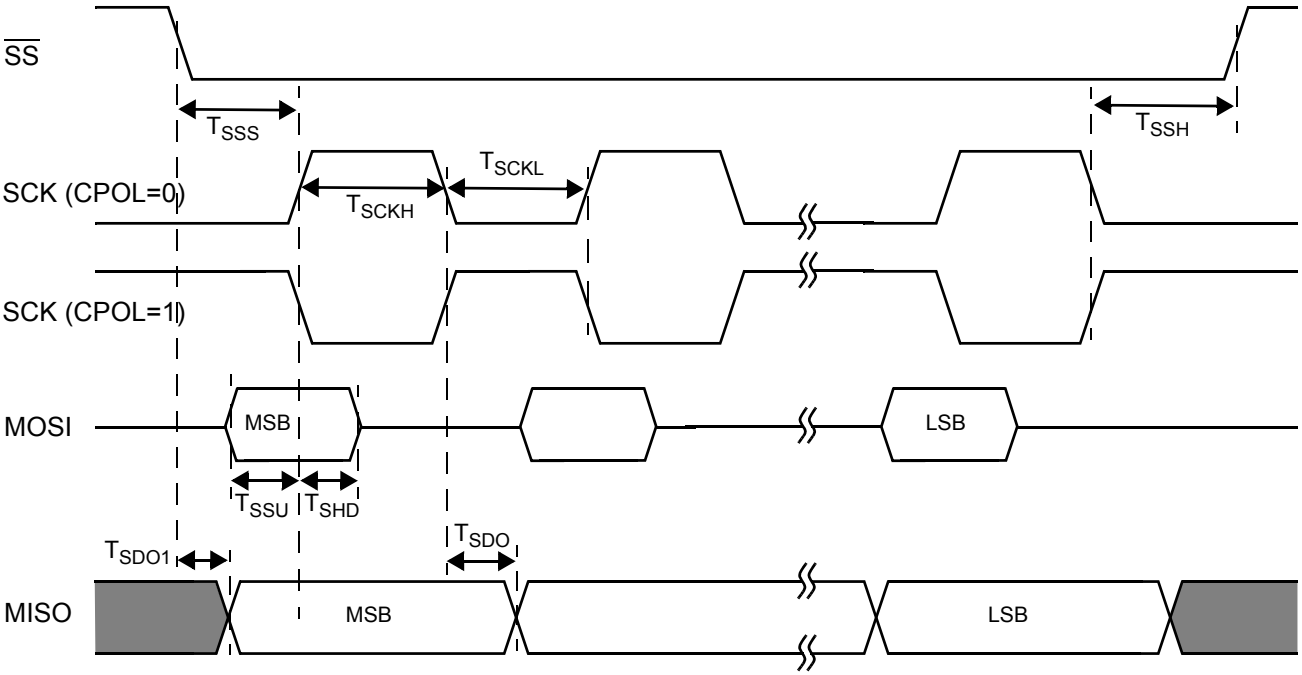


Figure 50. SPI Slave Timing, CPHA = 1



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

© Cypress Semiconductor Corporation, 2004-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.