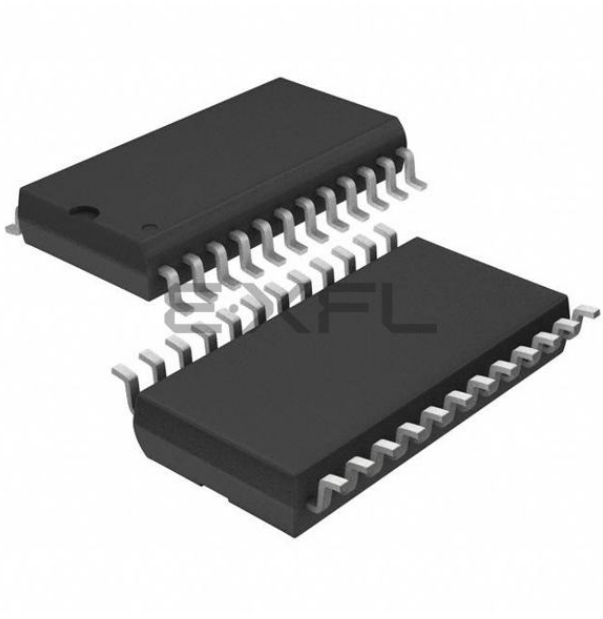


Welcome to E-XFL.COM

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	Surface Mount
Package / Case	24-SOIC (0.295", 7.50mm Width)
Supplier Device Package	24-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c63743c-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.

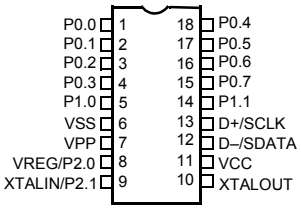
The USB D+ and D– USB pins can alternately be used as PS/2 SCLK and SDATA signals, so that products can be designed to respond to either USB or PS/2 modes of operation. PS/2 operation is supported with internal pull-up resistors on SCLK and SDATA, the ability to disable the regulator output pin, and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

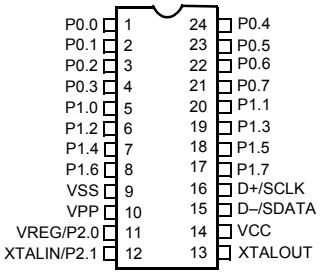
Pin Configurations

Top View

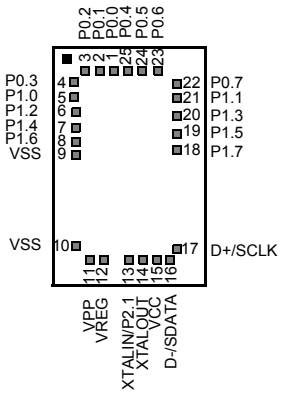
CY7C63723C
18-pin SOIC/PDIP



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

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

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.

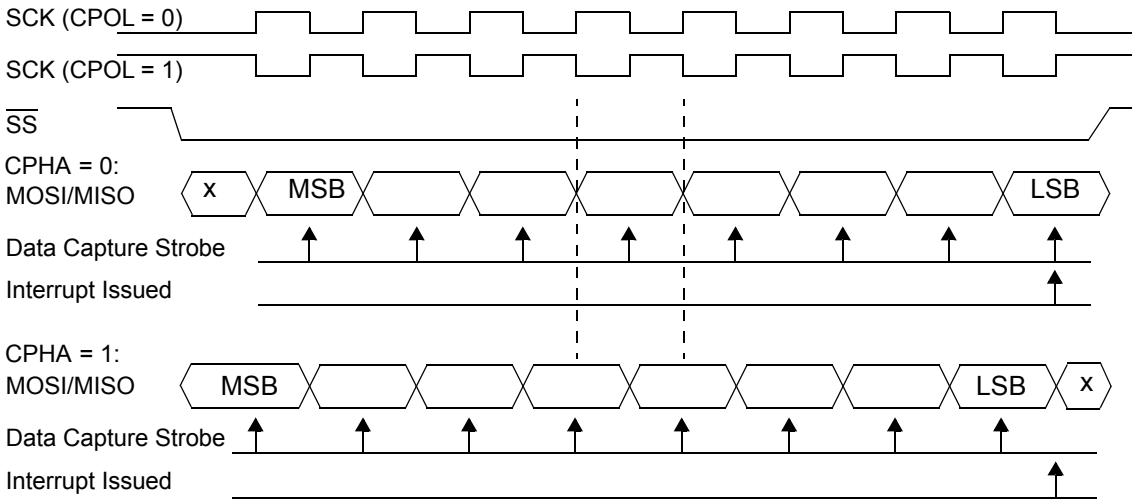
Once this timer is activated, it will give interrupts after its time-out period (see below). These interrupts continue periodically until the interrupt is disabled. Whenever the interrupt is disabled, the wake-up timer is reset, so that a subsequent enable always results in a full wake-up time.

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 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.

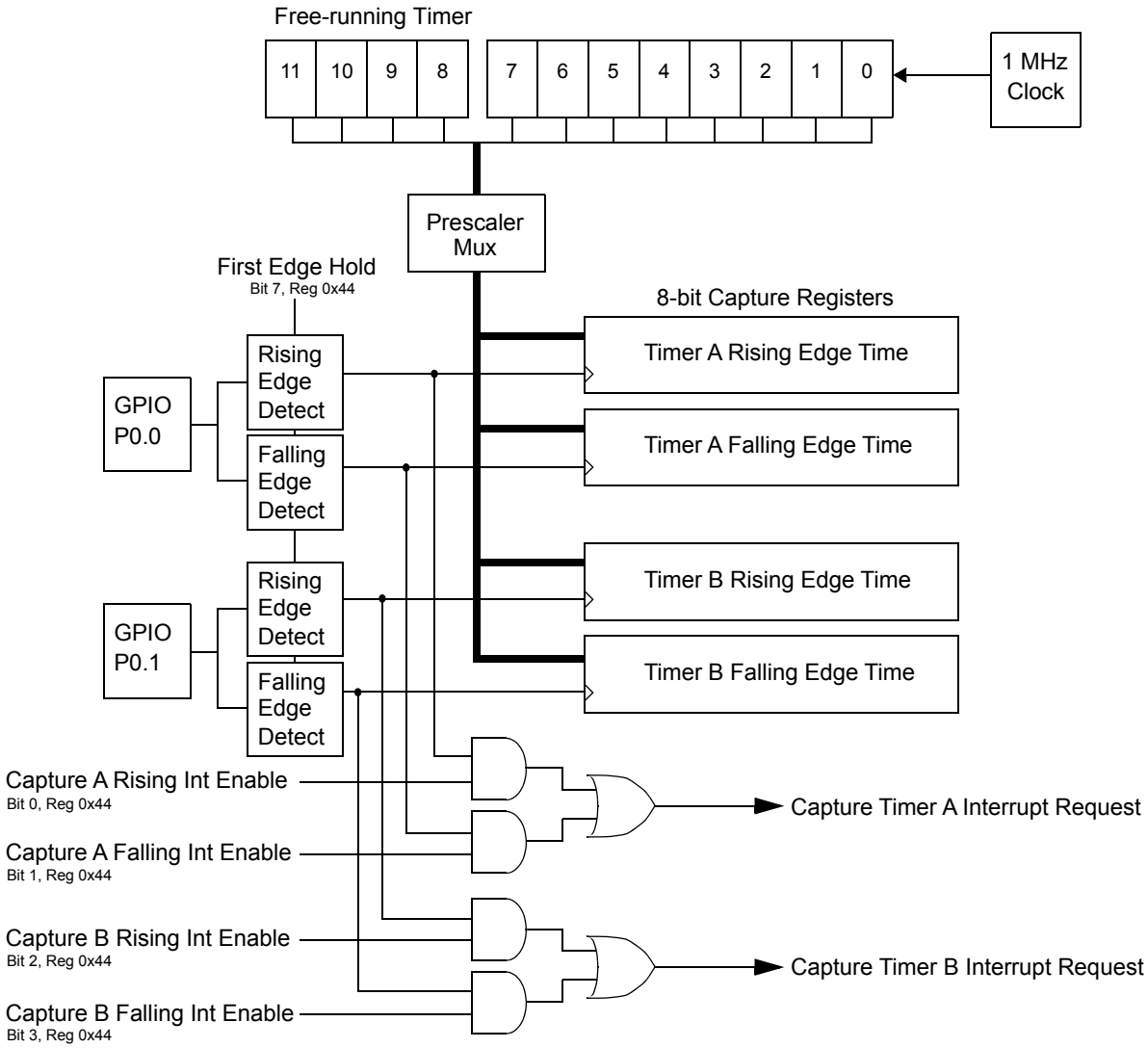
Table 5. SPI Pin Assignments

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



The four Capture Timer Data Registers are read-only, and are shown in Figure 28 through Figure 31.

Out of the 12-bit free running timer, the 8-bit captured in the Capture Timer Data Registers are determined by the Prescale Bit [2:0] in the Capture Timer Configuration Register (Figure 33).

Figure 28. Capture Timer A-Rising, Data Register (Address 0x40)

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

Figure 29. Capture Timer A-Falling, Data Register (Address 0x41)

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

Figure 30. Capture Timer B-Rising, Data Register (Address 0x42)

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

Figure 31. Capture Timer B-Falling, Data Register (Address 0x43)

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

Figure 32. Capture Timer Status Register (Address 0x45)

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

Bit [7:4]: Reserved.

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

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

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

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

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

Figure 33. Capture Timer Configuration Register (Address 0x44)

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

Bit 7: First Edge Hold

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

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

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

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

Three prescaler bits allow the capture timer clock rate to be selected among 5 choices, as shown in Table 6 below.

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

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

Both Capture A events share a common interrupt request, as do the two Capture B events. In addition to the event enables, the main Capture Interrupt Enables bit in the Global Interrupt Enable register (Interrupt Sources on page 32) must be set to activate a capture interrupt.

1 = Enable interrupt

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.

1 = Suspend the processor.

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

Bit 2: Interrupt Enable Sense

This bit shows whether interrupts are enabled or disabled. Firmware has no direct control over this bit as writing a zero or one to this bit position will have no effect on interrupts. This bit is further gated with the bit settings of the Global Interrupt Enable Register (Figure 35) and USB Endpoint Interrupt Enable Register (Figure 36). Instructions DI, EI, and RETI manipulate the state of this bit.

1 = Interrupts are enabled.

0 = Interrupts are masked off.

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

Bit 0: Run

This bit is manipulated by the HALT instruction. When Halt is executed, the processor clears the run bit and halts at the end of the current instruction. The processor remains halted until a reset occurs (low-voltage, brown-out, or Watchdog). This bit should normally be written as a '1'.

During power-up, or during a low-voltage reset, the Processor Status and Control Register is set to 00010001, which indicates a LVR/BOR (bit 4 set) has occurred and no interrupts are pending (bit 7 clear). Note that during the t_{START} ms partial suspend at start-up (explained in Reset on page 13), a Watchdog Reset will also occur. When a WDR occurs during the power-up suspend interval, firmware would read 01010001 from the Status and Control Register after power-up. Normally the LVR/BOR bit should be cleared so that a subsequent WDR can be clearly identified. *Note that if a USB bus reset (long SE0) is received before firmware examines this register, the Bus Interrupt Event bit would also be set.*

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

Interrupts

Interrupts can be generated by the GPIO lines, the internal free-running timer, the SPI block, the capture timers, on various USB events, PS/2 activity, or by the wake-up timer. All interrupts are maskable by the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register. Writing a '1' to a bit position enables the interrupt associated with that bit position. During a reset, the contents of the interrupt enable registers are cleared, along with the Global Interrupt enable bit of the CPU, effectively disabling all interrupts.

The interrupt controller contains a separate flip-flop for each interrupt. See Figure 37 for the logic block diagram of the interrupt controller. When an interrupt is generated it is first registered as a pending interrupt. It will stay pending until it is serviced or a reset occurs. A pending interrupt will only generate an interrupt request if it is enabled by the corresponding bit in the interrupt enable registers. The highest priority interrupt request will be serviced following the completion of the currently executing instruction.

When servicing an interrupt, the hardware will first disable all interrupts by clearing the Global Interrupt Enable bit in the CPU (the state of this bit can be read at Bit 2 of the Processor Status and Control Register). Next, the flip-flop of the current interrupt is cleared. This is followed by an automatic CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector, see Interrupt Vectors on page 31). The instruction in the interrupt table is typically a JMP instruction to the address of the Interrupt Service Routine (ISR). The user can re-enable interrupts in the interrupt service routine by executing an EI instruction. Interrupts can be nested to a level limited only by the available stack space.

The Program Counter value as well as the Carry and Zero flags (CF, ZF) are stored onto the Program Stack by the automatic CALL instruction generated as part of the interrupt acknowledge process. The user firmware is responsible for ensuring that the processor state is preserved and restored during an interrupt. The PUSH A instruction should typically be used as the first command in the ISR to save the accumulator value and the POP A instruction should be used just before the RETI instruction to restore the accumulator value. The program counter, CF and ZF are restored and interrupts are enabled when the RETI instruction is executed.

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

Interrupt Vectors

The Interrupt Vectors supported by the device are listed in Table 7. The highest priority interrupt is #1 (USB Bus Reset / PS/2 activity), and the lowest priority interrupt is #11 (Wake-up Timer). Although Reset is not an interrupt, the first instruction executed after a reset is at ROM address 0x0000, which corresponds to the first entry in the Interrupt Vector Table. Interrupt vectors occupy two bytes to allow for a two-byte JMP instruction to the appropriate Interrupt Service Routine (ISR).

Table 7. Interrupt Vector Assignments

Interrupt Vector No.	ROM Address	Function
not applicable	0x0000	Execution after Reset begins here
1	0x0002	USB Bus Reset or PS/2 Activity interrupt
2	0x0004	128-µs timer interrupt
3	0x0006	1.024-ms timer interrupt
4	0x0008	USB Endpoint 0 interrupt
5	0x000A	USB Endpoint 1 interrupt
6	0x000C	USB Endpoint 2 interrupt
7	0x000E	SPI Interrupt
8	0x0010	Capture Timer A interrupt

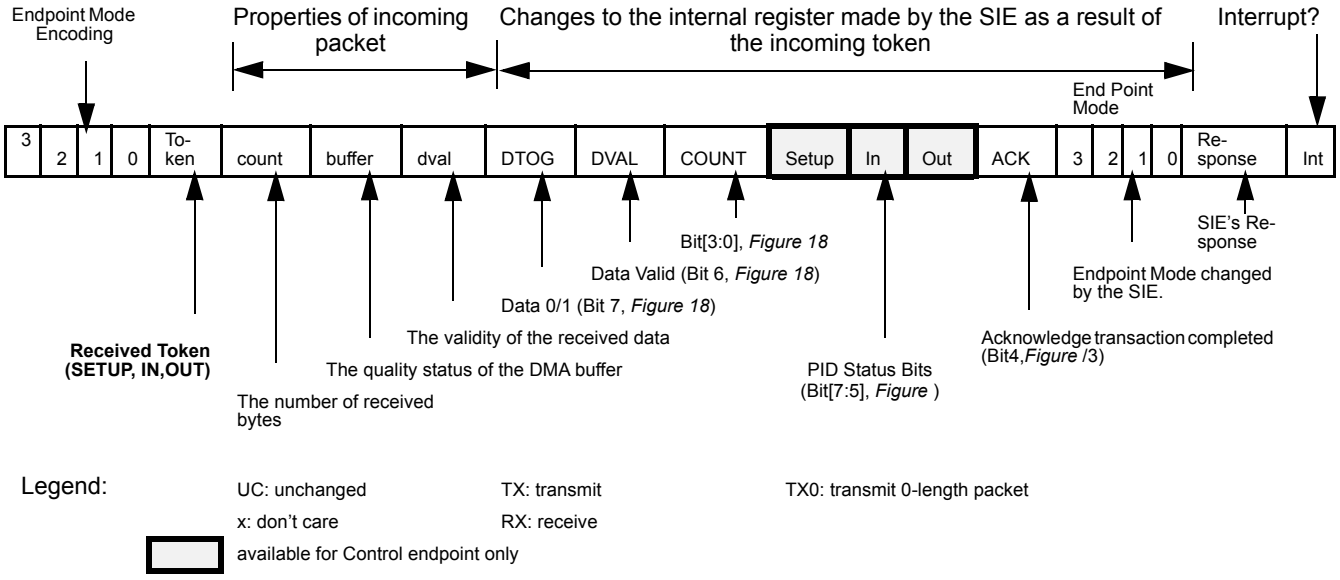
Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode as shown in Table 8, the SIE will change the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACKing a valid status stage OUT token. The firmware needs to update the mode for the SIE to respond appropriately. See Table 8 for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

Register Summary

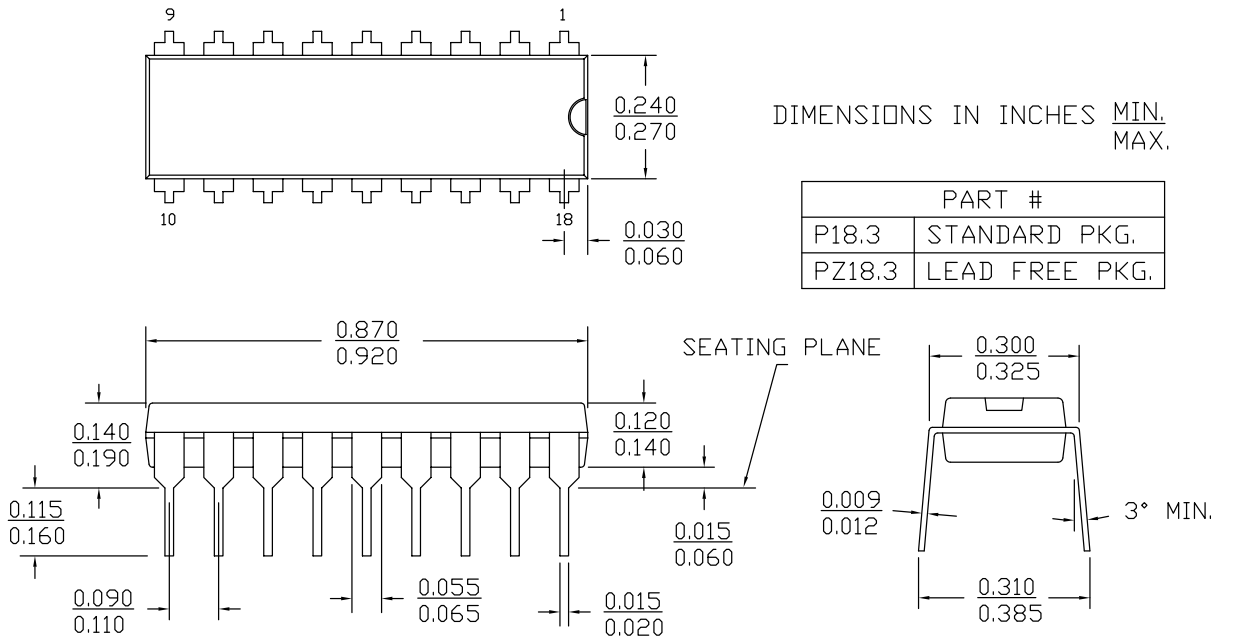
	Address	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Read/Write/ Both/	Default/ Reset
GPIO CONFIGURATION PORTS 0, 1 AND 2	0x00	Port 0 Data	P0								BBBBBBBB	00000000
	0x01	Port 1 Data	P1								BBBBBBBB	00000000
	0x02	Port 2 Data	Reserved		D+(SCLK) State	D- (SDATA) State	Reserved		P2.1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

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

Figure 54. 24-pin PDIP (1.260 × 0.270 × 0.140 Inches) P24.3 Package Outline, 51-85013

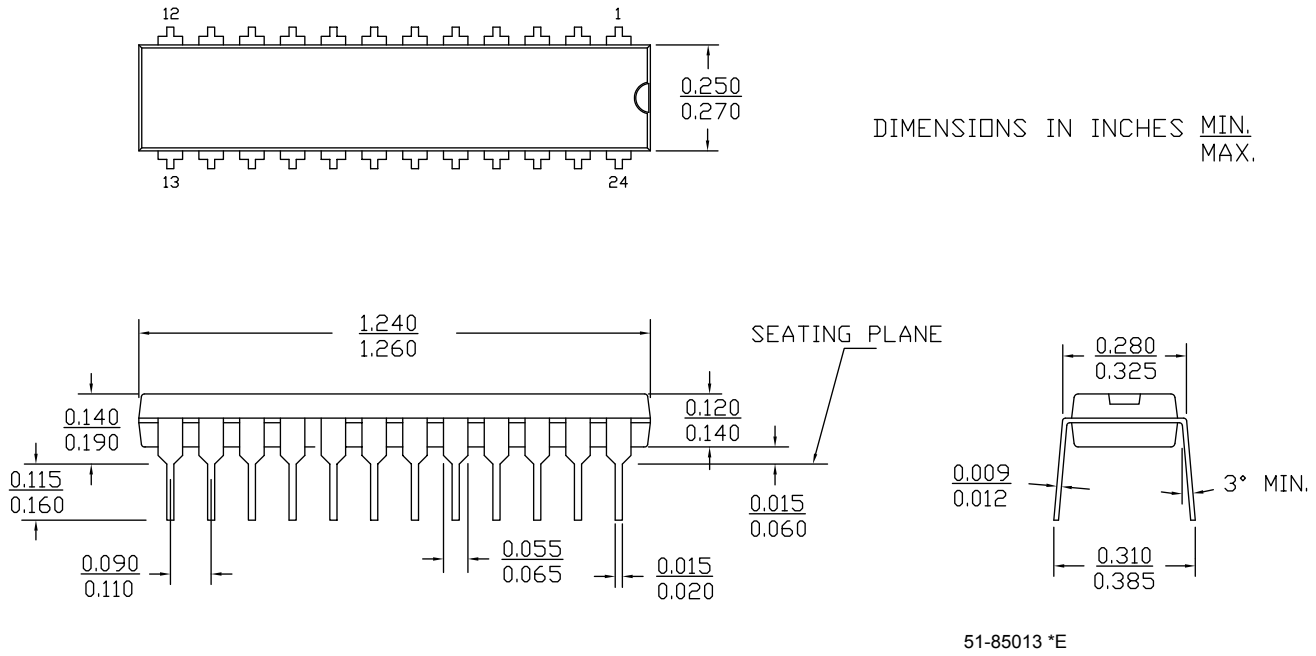
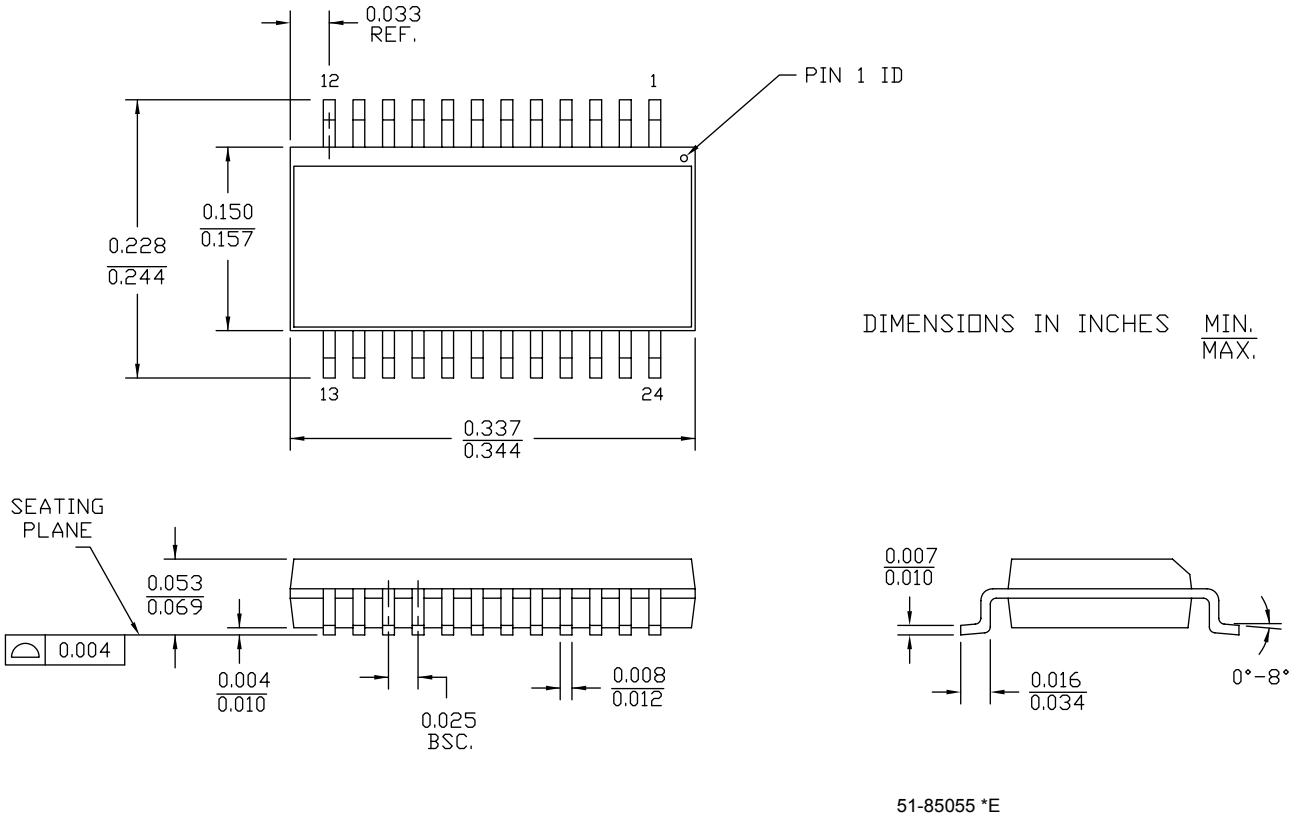
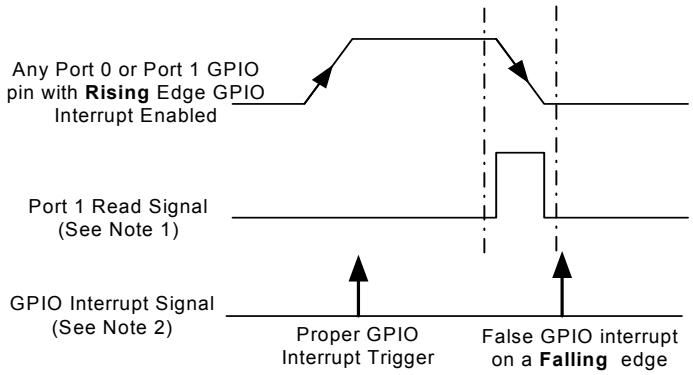


Figure 55. 24-pin QSOP (8.65 × 3.9 × 1.44 mm) O241 Package Outline, 51-85055





Note 1: Port 1 Read is an internal signal that is asserted when Port 1 is read with an "IORD 01h" instruction.

Note 2: The GPIO Interrupt signal is an internal signal. The arrow indicates that a GPIO interrupt is triggered.

■ **Parameters Affected**

Interrupts

■ **Trigger Condition(S)**

Reading the GPIO Port 1 when either rising or falling edge interrupts are enabled for a GPIO pin.

■ **Scope of Impact**

The chip enters the GPIO Interrupt Service Routine (ISR) in error.

■ **Workaround**

Workarounds will need to be tailored to individual applications based on the flexibility of changing the GPIO usage, the timing of the GPIO interrupt sources and firmware interrupt latencies.

■ **Fix Status**

No silicon fix is planned.

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