CY7C63722/23
CY7C63742/43
enCoRe™ USB
Combination Low-Speed USB & PS/2 Peripheral Controller

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF TABLES

1.0 Features

- enCoRe™ USB - enhanced Component Reduction
  - Internal oscillator eliminates the need for an external crystal or resonator
  - Interface can auto-configure to operate as PS/2 or USB without the need for external components to switch between modes (no GPIO pins needed to manage dual mode capability)
  - Internal 3.3V regulator for USB pull-up resistor
  - Configurable GPIO for real-world interface without external components
- Flexible, cost-effective solution for applications that combine PS/2 and low-speed USB, such as mice, gamepads, joysticks, and many others.
- USB Specification Compliance
  - Conforms to USB Specification, Version 1.1
  - Conforms to USB HID Specification, Version 1.1
  - Supports 1 Low-Speed USB device address and 3 data endpoints
  - Integrated USB transceiver
  - 3.3V regulated output for USB pull-up resistor
- 8-bit RISC microcontroller
  - Harvard architecture
  - 6-MHz external ceramic resonator or internal clock mode
  - 12-MHz internal CPU clock
  - Internal memory
  - 256 bytes of RAM
  - 6 Kbytes of EPROM (CY7C63722, CY7C63742)
  - 8 Kbytes of EPROM (CY7C63723, CY7C63743)
  - Interface can auto-configure to operate as PS/2 or USB
  - No external components for switching between PS/2 and USB modes
  - No GPIO pins needed to manage dual mode capability
- I/O ports
  - Up to 16 versatile General Purpose I/O (GPIO) pins, individually configurable
  - High current drive on any GPIO pin: 50 mA/pin current sink
  - Each GPIO pin supports high-impedance inputs, internal pull-ups, open drain outputs or traditional CMOS outputs
  - Maskable interrupts on all I/O pins
- SPI serial communication block
  - Master or slave operation
  - 2 Mbit/s transfers
- Four 8-bit Input Capture registers
  - Two registers each for two input pins
  - Capture timer setting with 5 pre-scaler settings
  - Separate registers for rising and falling edge capture
  - Simplifies interface to RF inputs for wireless applications
- Internal low-power wake-up timer during suspend mode
  - Periodic wake-up with no external components
- Optional 6-MHz internal oscillator mode
  - Allows fast start-up from suspend mode
- Watch dog timer (WDT)
- Low Voltage Reset at 3.75V
- Internal brown-out reset for suspend mode
- Improved output drivers to reduce EMI
- Operating voltage from 4.0V to 5.5VDC

- **Operating temperature from 0 to 70 degrees Celsius**
- **CY7C63722/23 available in 18-pin PDIP**
- **CY7C63742/43 available in 24-pin SOIC, 24-pin PDIP**
- **Industry standard programmer support**

2.0 Functional Overview

2.1 *enCoRe* USB - The New USB Standard

Cypress has re-invented 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 will enable peripheral developers to design new products with a minimum number of components. At the heart of our *enCoRe*™ USB technology is the breakthrough design of a crystal-less oscillator. By integrating the oscillator into our chip, an external crystal or resonator is no longer needed. We have also integrated other external components commonly found in low-speed USB applications such as pull-up resistors, wake-up circuitry, and a 3.3V regulator. All of this adds up to a lower system cost.

The CY7C63722/23 and CY7C63742/43 are 8-bit RISC One Time Programmable (OTP) microcontrollers. The instruction set has been optimized specifically for USB and PS/2 operations, although the microcontrollers can be used for a variety of other embedded applications.

The CY7C637xx features up to 16 general purpose I/O (GPIO) pins to support USB, PS/2 and other applications. The I/O pins are grouped into two ports (Port 0 to 1) where each pin can be individually configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs with programmable drive strength of up to 50 mA output drive. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. Note the GPIO interrupts all share the same "GPIO" interrupt vector.

The CY7C637xx microcontrollers feature an internal 5% accurate 6-MHz clock source. Optionally, an external 6-MHz ceramic resonator can be used to provide a higher precision reference for USB operation. This clock generator reduces the clock-related noise emissions (EMI). The clock generator provides the 6- and 12-MHz clocks that remain internal to the microcontroller.

The CY7C637xx is offered with two EPROM options to maximize flexibility and minimize cost. The CY7C637x2 has 6 Kbytes of EPROM. The CY7C637x3 has 8 Kbytes of EPROM. All versions have 256 bytes of data RAM for stack space, user variables, and USB FIFOs.

These parts include low-voltage reset logic, a watch dog 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 watch dog timer can be used to ensure the firmware never gets stalled for more than approximately 8 ms.

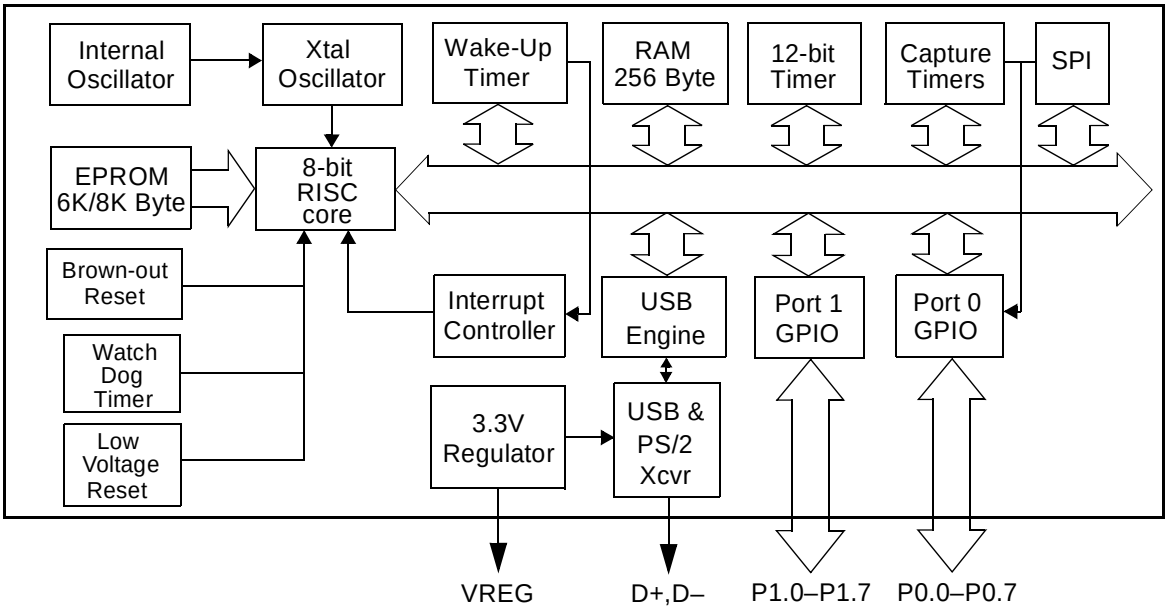
The microcontroller supports 10 maskable interrupts in the vectored interrupt controller. Interrupt sources include the USB Bus-Reset, the 128-µs and 1.024-ms outputs from the free-running timer, three USB endpoints, two capture timers, an internal wake-up timer and the GPIO ports. The timers bits cause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

The free-running 12-bit timer clocked at 1 MHz provides two interrupt sources as noted above (128 µs and 1.024 ms). The timer can be used to measure the duration of an event under firmware control by reading the timer at the start and end of an event, and subtracting the two values. The four capture timers save a programmable 8 bit range of the free-running timer when a GPIO edge occurs on the two capture pins (P0.0, P0.1).

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

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

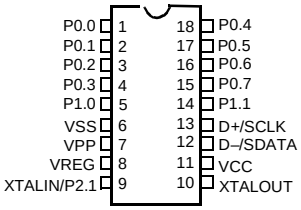
3.0 Logic Block Diagram



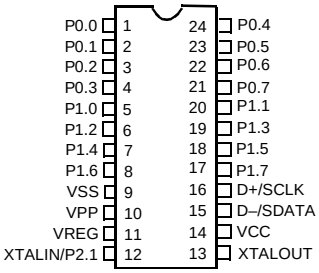
4.0 Pin Configurations

Top View

CY7C63722/23
18-pin PDIP



CY7C63742/43
24-pin SOIC/PDIP



5.0 Pin Assignments

Name	I/O	CY7C63722/23 18-Pin	CY7C63742/43 24-Pin	Description
D–/SDATA, D+/SCLK	I/O	12 13	15 16	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	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	IO 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.
XTALIN/P2.1	IN	9	12	6-MHz ceramic resonator or external clock input, or P2.1 input
XTALOUT	OUT	10	13	6-MHz ceramic resonator return pin or internal oscillator output
V_{PP}		7	10	Programming voltage supply, ground for normal operation
V_{CC}		11	14	Voltage supply
VREG/P2.0		8	11	Voltage supply for 1.5-kΩ USB pull-up resistor (3.3V nominal). Also serves as P2.0 input.
V_{SS}		6	9	Ground

6.0 Programming Model

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

6.1 Program Counter (PC)

The 13-bit program counter (PC) allows access for up to 8 Kbytes of EPROM using the CY7C637xx architecture. The program counter is cleared during reset, such that the first instruction executed after a reset is at address 0x0000. This 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 5 bits of the program counter are incremented by executing an XPAGE instruction. As a result, the last instruction executed within a 256-byte "page" of sequential code should be an XPAGE instruction. The assembler directive "XPAGEON" will cause the assembler to insert XPAGE instructions automatically. As instructions can be either one or two bytes long, the assembler may occasionally need to insert a NOP followed by an XPAGE for correct execution.

The program counter of the next instruction to be executed, carry flag, and zero flag are saved as two bytes on the program stack during an interrupt acknowledge or a CALL instruction. The program counter, carry flag, and zero flag are restored from the program stack only during a RETI instruction.

Please note the program counter cannot be accessed directly by the firmware. The program stack can be examined by reading SRAM from location 0x00 and up.

6.2 8-bit Accumulator (A)

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

6.3 8-bit Index Register (X)

The index register "X" is available to the firmware as an auxiliary accumulator. The X register also allows the processor to perform indexed operations by loading an index value into X.

6.4 8-bit Program Stack Pointer (PSP)

During a reset, the program stack pointer (PSP) is set to zero. This means the program "stack" starts at RAM address 0x00 and "grows" upward from there. Note 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 some jump, call, and index instructions across the 4KB boundary of the program memory. Refer to the *CYASM Assembler User's Guide* for a detailed description.

6.5 8-bit Data Stack Pointer (DSP)

The data stack pointer (DSP) supports PUSH and POP instructions that use the data stack for temporary storage. A PUSH instruction will pre-decrement the DSP, then write data to the memory location addressed by the DSP. A POP instruction will read data from the memory location addressed by the DSP, then post-increment the DSP.

During a reset, the Data Stack Pointer will be set to zero. A PUSH instruction when DSP equal zero will write data at the top of the data RAM (address 0xFF). This would write data to the memory area reserved for a FIFO for USB endpoint 0. In non-USB applications, this works fine and is not a problem.

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

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

6.6 Address Modes

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

6.6.1 Data

The “Data” address mode refers to a data operand that is actually a constant encoded in the instruction. As an example, consider the instruction that loads A with the constant 0x30:

- MOV A, 30h

This instruction will require two bytes of code where the first byte identifies the “MOV A” instruction with a data operand as the second byte. The second byte of the instruction will be the constant “0xE8h”. A constant may be referred to by name if a prior “EQU” statement assigns the constant value to the name. For example, the following code is equivalent to the example shown above:

- DSPINIT: EQU 30h
- MOV A,DSPINIT

6.6.2 Direct

“Direct” address mode is used when the data operand is a variable stored in SRAM. In that case, the one byte address of the variable is encoded in the instruction. As an example, consider an instruction that loads A with the contents of memory address location 0x10h:

- MOV A, [10h]

In normal usage, variable names are assigned to variable addresses using “EQU” statements to improve the readability of the assembler source code. As an example, the following code is equivalent to the example shown above:

- buttons: EQU 10h
- MOV A,[buttons]

6.6.3 Indexed

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

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

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

7.0 Instruction Set Summary

Refer to the *CYASM Assembler User's Guide* for detailed information on these instructions.

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	4
AND A,expr	data	10	4		SWAP A,DSP		30	4
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
MOV A,[expr]	direct	1A	5		CPL		3A	4
MOV A,[X+expr]	index	1B	6		ASL		3B	4
MOV X,expr	data	1C	4		ASR		3C	4
MOV X,[expr]	direct	1D	5		RLC		3D	4
reserved		1E			RRC		3E	4
XPAGE		1F	4		RET		3F	8
MOV A,X		40	4		DI		70	4
MOV X,A		41	4		EI		72	4
MOV PSP,A		60	4		RETI		73	8
CALL	addr	50 - 5F	10					
JMP	addr	80-8F	5		JC	addr	C0-CF	5
CALL	addr	90-9F	10		JNC	addr	D0-DF	5
JZ	addr	A0-AF	5		JACC	addr	E0-EF	7
JNZ	addr	B0-BF	5		INDEX	addr	F0-FF	14

8.0 Memory Organization

8.1 Program Memory Organization

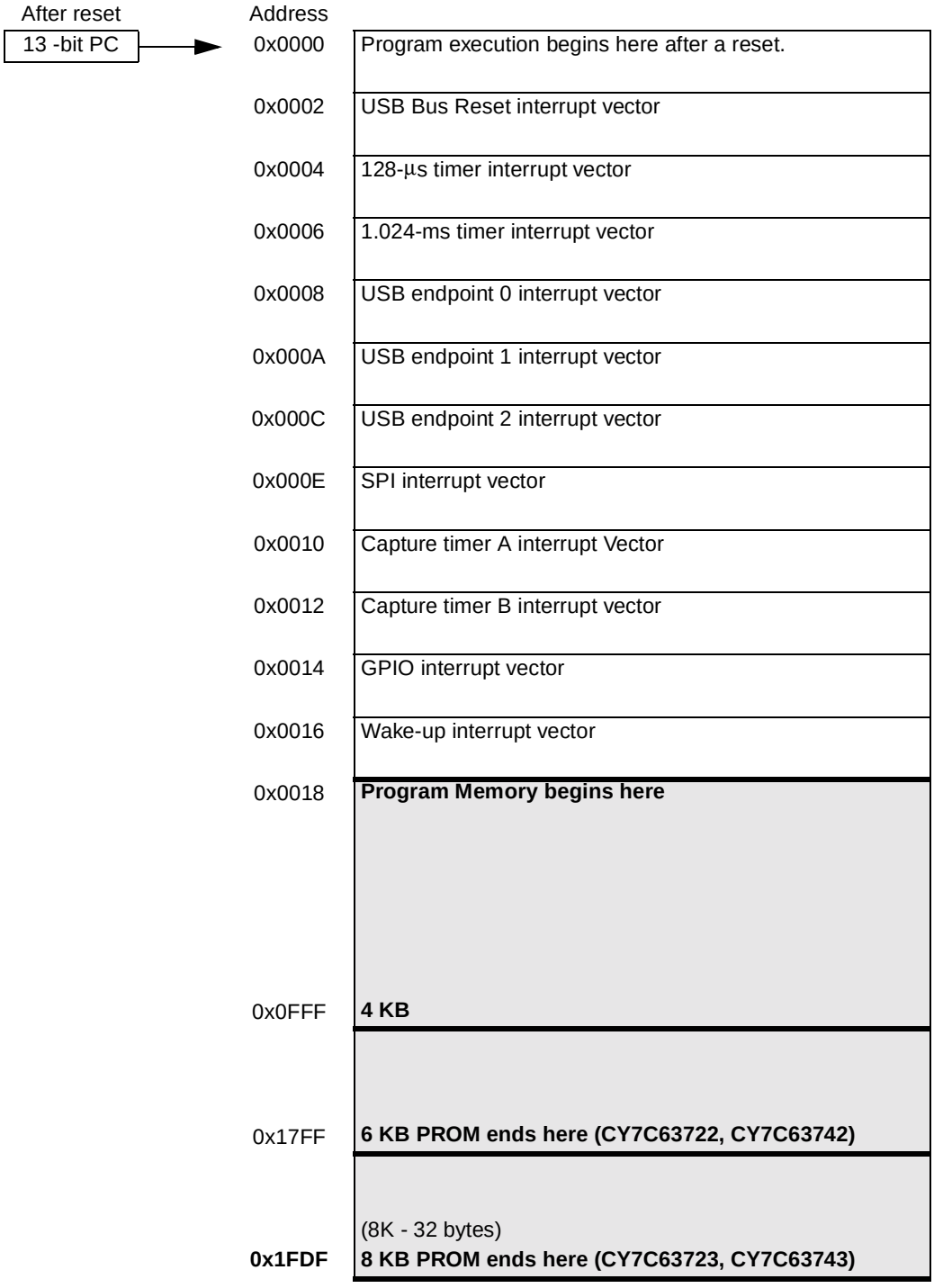
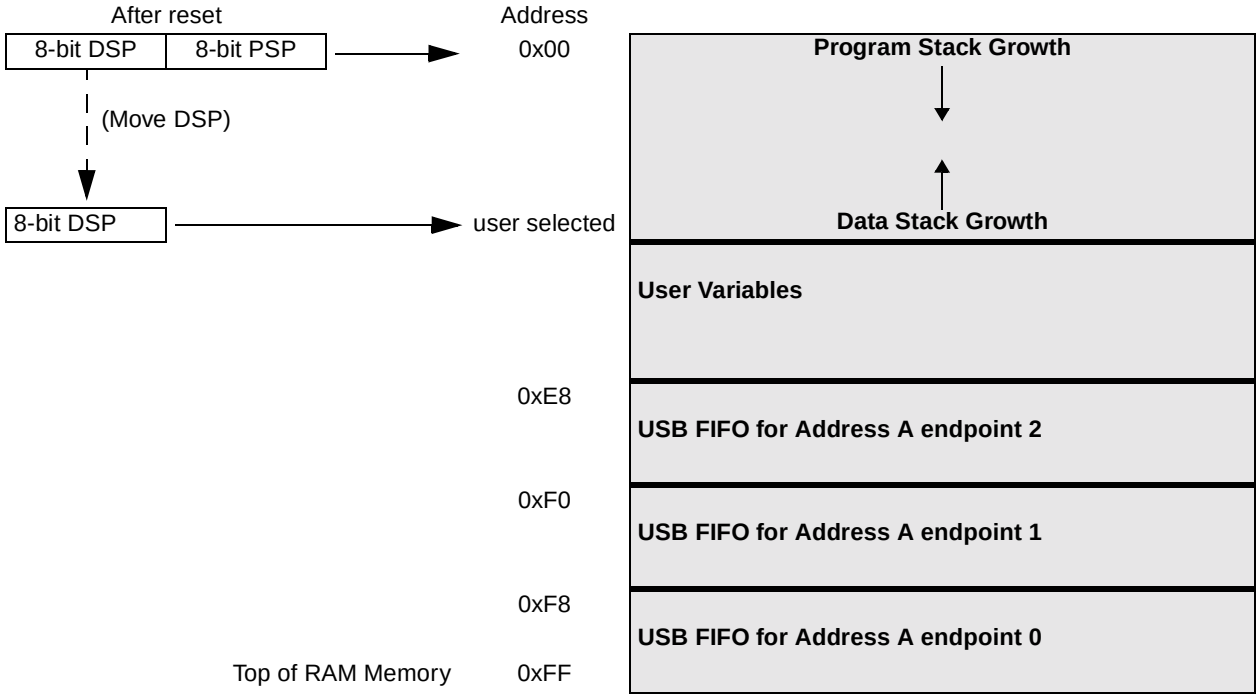


Figure 8-1. Program Memory Space with Interrupt Vector Table

8.2 Data Memory Organization

The CY7C637xx microcontrollers provide 256 bytes of data RAM. In normal usage, the SRAM is partitioned into four areas: program stack, data stack, user variables and USB endpoint FIFOs as shown below:



8.3 I/O Register Summary

I/O registers are accessed via the I/O Read (IORD) and I/O Write (IOWR, IOWX) instructions. IORD reads the selected port into the accumulator. IOWR writes data from the accumulator to the selected port. Indexed I/O Write (IOWX) adds the contents of X to the address in the instruction to form the port address and writes data from the accumulator to the specified port. Note that specifying address 0 with IOWX (e.g., IOWX 0h) means the I/O port is selected solely by the contents of X.

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

Table 8-1. I/O Register Summary

Register Name	I/O Address	Read/Write	Function	Fig.
Port 0 Data	0x00	R/W	GPIO Port 0	12-2
Port 1 Data	0x01	R/W	GPIO Port 1	12-3
Port 2 Data	0x02	R	Auxiliary input register for D+, D–, VREG, XTALIN	12-8
Port 0 Interrupt Enable	0x04	W	Interrupt enable for pins in Port 0	21-4
Port 1 Interrupt Enable	0x05	W	Interrupt enable for pins in Port 1	21-5
Port 0 Interrupt Polarity	0x06	W	Interrupt polarity for pins in Port 0	21-6
Port 1 Interrupt Polarity	0x07	W	Interrupt polarity for pins in Port 1	21-7
Port 0 Mode0	0x0A	W	Controls output configuration for Port 0	12-4
Port 0 Mode1	0x0B	W		12-5
Port 1 Mode0	0x0C	W	Controls output configuration for Port 1	12-6
Port 1 Mode1	0x0D	W		12-7
USB Device Address	0x10	R/W	USB Device Address register	14-1
EP0 Counter Register	0x11	R/W	USB Endpoint 0 counter register	14-4
EP0 Mode Register	0x12	R/W	USB Endpoint 0 configuration register	14-2
EP1 Counter Register	0x13	R/W	USB Endpoint 1 counter register	14-4
EP1 Mode Register	0x14	R/W	USB Endpoint 1 configuration register	14-3
EP2 Counter Register	0x15	R/W	USB Endpoint 2 counter register	14-4
EP2 Mode Register	0x16	R/W	USB Endpoint 2 configuration register	14-3
USB Status & Control	0x1F	R/W	USB status and control register	13-1
Global Interrupt Enable	0x20	R/W	Global interrupt enable register	21-1
Endpoint Interrupt Enable	0x21	R/W	USB endpoint interrupt enables	21-2
Timer (LSB)	0x24	R	Lower 8 bits of free-running timer (1 MHz)	18-1
Timer (MSB)	0x25	R	Upper 4 bits of free-running timer	18-2
WDR Clear	0x26	W	Watch Dog Reset clear	-
Capture Timer A Rising	0x40	R	Rising edge Capture Timer A data register	19-2
Capture Timer A Falling	0x41	R	Falling edge Capture Timer A data register	19-3
Capture Timer B Rising	0x42	R	Rising edge Capture Timer B data register	19-4
Capture Timer B Falling	0x43	R	Falling edge Capture Timer B data register	19-5
Capture TImer Configuration	0x44	R/W	Capture Timer configuration register	19-6
Capture Timer Status	0x45	R	Capture Timer status register	19-7
SPI Data	0x60	R/W	SPI read and write data register	17-2
SPI Control	0x61	R/W	SPI status and control register	17-3
Clock Configuration	0xF8	R/W	Internal / External Clock configuration register	9-2
Processor Status & Control	0xFF	R/W	Processor status and control	20-1

9.0 Clocking

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

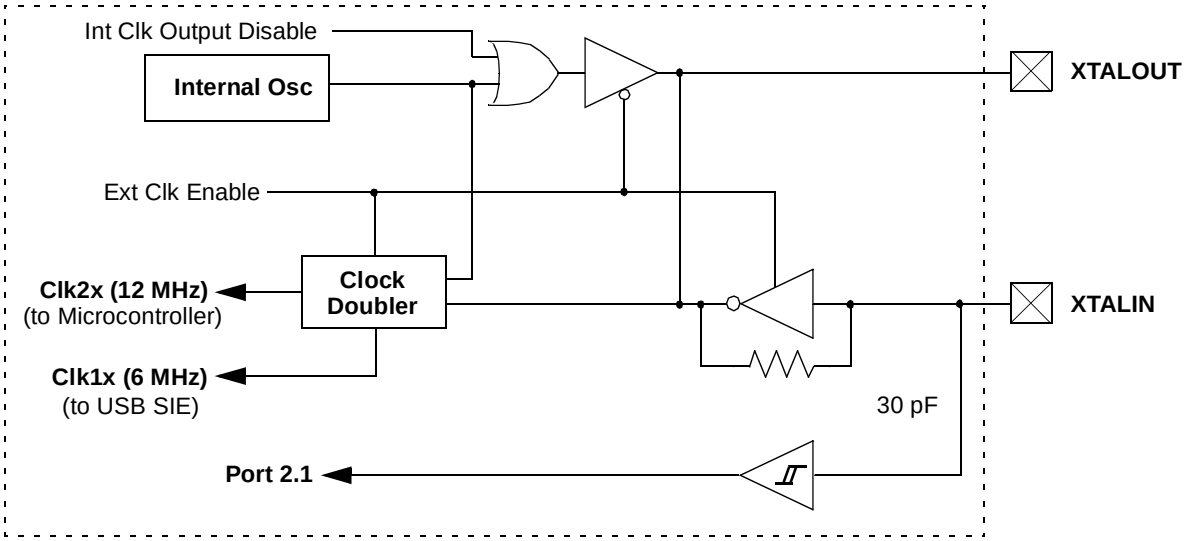


Figure 9-1. Clock Oscillator On-chip Circuit

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
Ext. Clock Resume Delay	Wake-up Timer Adjust Bit 2	Wake-up Timer Adjust Bit 1	Wake-up Timer Adjust Bit 0	Low Voltage Reset Disable	Precision USB Clocking Enable	Internal Clock Output Disable	External Oscillator Enable

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

All bits of the Clock Configuration Register reset to 0. Reserved bits must always be written as a 0.

Setting External Oscillator Enable (bit 0) high causes the part to switch to external clock mode, as described in Section 9.1. (If the bit is already set, writing a '1' again has no effect.) 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. When this bit is cleared XTALIN will be configured as an input with a weak pull down and can be used as a GPIO input (P2.1).

The Internal Clock Output Disable (bit 1) can be set to 1 to keep the internal clock from driving out to XTALOUT. If set, XTALOUT will drive high. This bit has no effect when the external oscillator is enabled.

The Precision USB Clocking Enable (bit 2) 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.

The Low Voltage Reset Disable (bit 3) disables the LVR circuit when set to 1. See Section 10.1.

The Wake-up Timer Adjust Bits (bits 6:4) are used to adjust the Wake-up timer period, as described in Section 11.2.

The Resume Delay (bit 7) selects the delay time when switching to the external oscillator, or when waking from suspend mode with the external oscillator enabled. The delay is 128 µs when this bit is 0, and 4 ms when this bit is 1. The shorter time is adequate for operation with ceramic resonators, while the longer time is preferred for start-up with a crystal. (These times **do not include** an initial oscillator start-up time which depends on the resonating element. This time is typically 50–100 µs for ceramic resonators and 1–10 ms for crystals). When waking from suspend mode with the internal oscillator, the delay time is only 8 µs in addition to a delay of approximately 1 µs for the oscillator to start.

9.1 Internal / External Oscillator Operation

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

Setting bit 0 of the Clock Configuration Register 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 (External Oscillator Enable); bit 7 cleared gives faster start-up
iowr F8h        ; Write to Clock Configuration Register
```

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

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

9.2 External Oscillator

The user can connect a low-cost ceramic resonator or an external oscillator to the XTALIN / XTALOUT pins to provide a precise reference frequency for the chip clock, as shown in *Figure 9-1*. The external components required are a ceramic resonator or crystal with external capacitors. To run from the external resonator, Bit 0 of the Clock Configuration Register must be set to 1, as explained in the previous section.

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

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

10.0 Reset

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

1. Low Voltage Reset (LVR)
2. Brown Out Reset (BOR)
3. Watch Dog Reset (WDR)

The occurrence of a reset is recorded in the Processor Status and Control Register (see *Figure 20-1*). Bits 4 and 6 are used to record the occurrence of LVR/BOR and WDR respectively. The firmware can interrogate these bits to determine the cause of a reset.

The microcontroller begins execution from ROM address 0x0000 after a LVR, BOR or WDR reset. Although this looks like interrupt vector 0, there is an important difference. Reset processing does NOT push the program counter, carry flag, and zero flag onto program stack. Attempting to execute either a RET or RETI in the reset handler will cause unpredictable execution results.

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

1. All registers are reset to their default states (all bits cleared, except in Processor Status and Control Register).
2. GPIO and USB pins are set to high-impedance state.
3. The VREG pin is set to high-impedance state.
4. Interrupts are disabled.
5. USB operation is disabled and must be enabled by firmware if desired, as explained in Section 14.1.
6. For a BOR or LVR, the external oscillator is disabled and Internal Clock mode is activated, followed by a time-out period t_{START} for V_{CC} to stabilize. A WDR does not change the clock mode, and there is no delay for V_{CC} stabilization on a WDR. Note that

the External Oscillator Enable (Bit 0, *Figure 9-2*) will be cleared by a WDR, but it does not take effect until suspend mode is entered.

7. The Program Stack Pointer (PSP) and Data Stack Pointer (DSP) reset to address 0x00. (Firmware should move the DSP for USB applications, as explained in Section 6.5.)
8. Program execution begins at address 0x0000 (after the appropriate time-out period).

10.1 Low Voltage Reset (LVR)

The CY7C637xx enters a partial suspend state when V_{CC} is first applied to the chip. The internal oscillator is started and the Low Voltage Reset (LVR) signal is initially asserted at power-up until V_{CC} has risen above V_{LVR}. At that point, the LVR is deasserted and an internal counter starts counting. After t_{START} the partial suspend state ends and program execution begins from address 0x0000. This provides time for V_{CC} to stabilize before the part executes code.

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

Whenever LVR is disabled (i.e. by firmware or during suspend mode), a secondary low-voltage monitor (BOR) is active, as described in the next section. The LVR/BOR bit, bit 4 of the Processor Status and Control Register (20-1), is set to “1” to indicate that one of these resets has occurred.

10.2 Brown Out Reset (BOR)

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

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

10.3 Watch Dog Reset (WDR)

The Watch Dog Timer Reset (WDR) occurs when the internal Watch Dog timer rolls over. Writing any value to the write-only Watch Dog Restart Register at address 0x26 will clear the timer. The timer will roll over and WDR will occur if it is not cleared within t_{WATCH} (10 ms minimum) of the last clear. Bit 6 of the Processor Status and Control Register is set to record this event (the register contents are set to 010X0001 by the WDR). A Watch Dog Timer Reset lasts for 2–4 ms after which the microcontroller begins execution at ROM address 0x0000. The clock mode (internal or external) is not changed by a WDR.

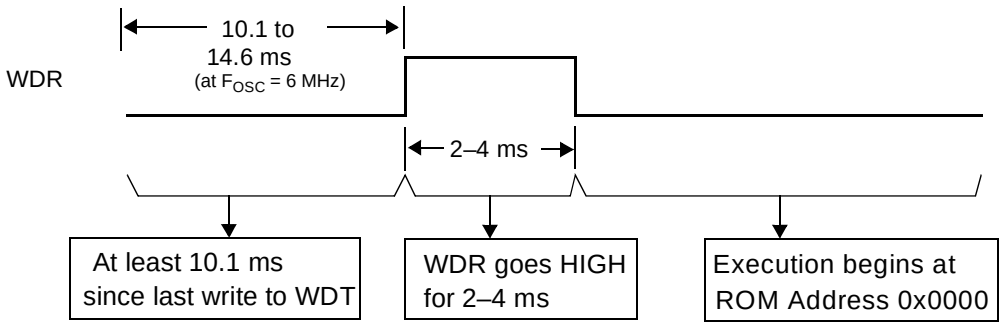


Figure 10-1. Watch Dog Reset (WDR)

11.0 Suspend Mode

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

The CY7C637xx is placed into a low-power state by setting the Suspend bit of the Processor Status and Control Register (*Figure 20-1*). All logic blocks in the device are turned off except the GPIO interrupt logic, the D–/SDATA pin input receiver, and (optionally) the wake-up timer. The clock oscillators, as well as the free-running and watch dog timers are shut down. Only the occurrence of

an enabled GPIO interrupt, wake-up interrupt, SPI slave interrupt, or a LOW state on the D–/SDATA pin will wake the part from suspend (D– LOW indicates non-idle USB activity). Once one of these resuming conditions occurs, clocks will be restarted and the device returns to full operation after the oscillator is stable and the selected delay period expires. This delay period is determined by selection of internal vs. external clock, and by the state of the Ext. Clock Resume Delay as explained in Section 9.0.

Note that executing the DI instruction to turn off all interrupts before suspending can cause an unintended wake-up from a **pending** interrupt. To avoid this, any interrupts not intended for waking from suspend should be disabled through the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register (Section 21.0). In that case executing the DI instruction is not necessary.

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 a 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. *Note that this also applies to internal port pins that may not be bonded in a particular package. Any unused bits of Ports 0 and 1 should typically be set to pull-up mode, even if the pins are not present on the package.* In addition, the GPIO *bit* interrupts (*Figure 21-4* and *Figure 21-5*) should be disabled for any pins that are not being used for a wake-up interrupt. This should be done even if the main GPIO Interrupt Enable (*Figure 21-1*) is off.

Typical code for entering suspend is shown below:

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

11.1 Clocking Mode on Wake-up from Suspend

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

The sequence of events for these modes is as follows:

Wake in Internal Clock Mode:

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

Wake in External Clock Mode:

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

11.2 Wake-up Timer

The wake-up timer runs whenever the wake-up interrupt is enabled, and is turned off whenever that interrupt is disabled. Operation is independent of whether the device is in suspend mode or if the global interrupt bit is enabled. Only the wake-up interrupt enable controls the wake-up timer.

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

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

Table 11-1. Wake-up Timer Adjust Settings

Adjust Bits [2:0] (Bits [6:4] of Register 0xF8)	Wake-up Time
000 (reset state)	1 * t_{WAKE}
001	2 * t_{WAKE}
010	4 * t_{WAKE}
011	8 * t_{WAKE}
100	16 * t_{WAKE}
101	32 * t_{WAKE}
110	64 * t_{WAKE}
111	128 * t_{WAKE}

12.0 General Purpose I/O Ports

Ports 0 and 1 provide up to 16 versatile GPIO pins that can be read or written (the number of pins depends on package type). Each pin can be configured as high-impedance inputs, inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs with selectable drive strengths. *Figure 12-1* shows a diagram of a GPIO port pin. Each I/O pin can be configured independently of any other pin. Port 0 is an 8-bit port; Port 1 contains either 2 bits, P1.1–P1.0 in the CY7C63722/23, or all 8-bits, P1.7 - P1.0 in the CY7C63742/43 parts. Each bit can also be selected as an interrupt source for the microcontroller, as explained in Section 21.3.6.

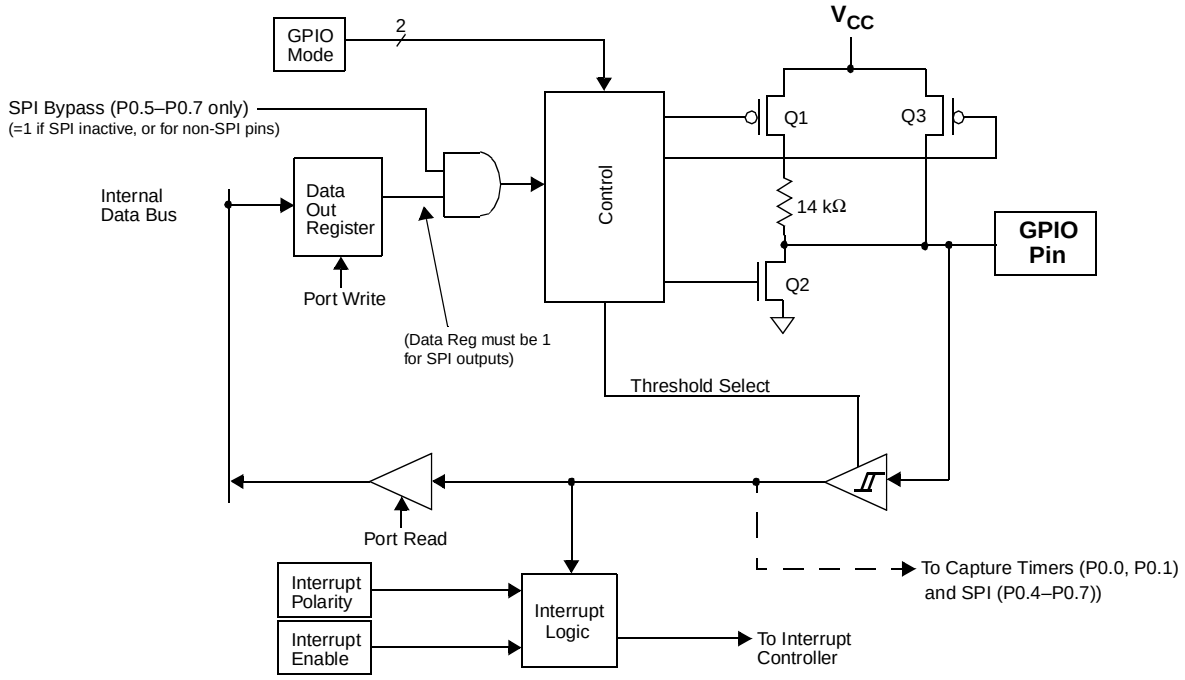


Figure 12-1. Block Diagram of GPIO Port (one pin shown)

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

Mode 00 High-impedance mode.
Mode 01 Medium Drive CMOS Mode: 8 mA sink current / 2 mA source current.
Mode 10 Low Drive / Resistive Mode: 2 mA sink current / 14 kΩ pull-up resistor.
Mode 11 High Drive CMOS Mode. 50 mA sink current / 2 mA source current.

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

Input thresholds are CMOS (centered around $V_{CC}/2$), or TTL as shown in the table. Both input modes include hysteresis to minimize noise sensitivity. In suspend mode, if a pin is used for a wake-up interrupt using an external R-C circuit, CMOS mode is preferred for lowest power.

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

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

Note that reading the GPIO port returns a byte based on the actual voltage of each pin, and does not affect the port's Data or Mode Registers.

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

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0

Figure 12-2. Port 0 Data (Address 0x00)[1]

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
Pins 7:2 only in CY7C63742/43						Pins 1:0 in all parts	

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

Note:
1. When performing a read of the Port 0 or Port 1 Data registers, above, only the status of the GPIO pins will be read. The registers content will NOT be read.

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P0.7 Mode0	P0.6 Mode0	P0.5 Mode0	P0.4 Mode0	P0.3 Mode0	P0.2 Mode0	P0.1 Mode0	P0.0 Mode0

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

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P0.7 Mode1	P0.6 Mode1	P0.5 Mode1	P0.4 Mode1	P0.3 Mode1	P0.2 Mode1	P0.1 Mode1	P0.0 Mode1

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

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P1.7 Mode0	P1.6 Mode0	P1.5 Mode0	P1.4 Mode0	P1.3 Mode0	P1.2 Mode0	P1.1 Mode0	P1.0 Mode0

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

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P1.7 Mode1	P1.6 Mode1	P1.5 Mode1	P1.4 Mode1	P1.3 Mode1	P1.2 Mode1	P1.1 Mode1	P1.0 Mode1

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

Refer to Section 21.3.6 for details on using the GPIO as interrupt sources.

12.1 Auxiliary Input Port

Port 2 serves as an auxiliary input port. The state of the D+ and D– pins can be read from this port, as shown in *Figure 12-8*. In addition, the VREG and XTALIN pins can serve as general purpose inputs in certain modes. For the VREG pin, refer to Section 15.0. For the XTALIN pin, refer to Section 9.1. In these modes, the pin states can be read from Port 2.

The Port 2 inputs all have TTL input thresholds.

7	6	5	4	3	2	1	0
-	-	R	R	-	-	R	R
Reserved	Reserved	D+ (SCLK) State	D– (SDATA) State	Reserved	Reserved	P2.1 (Internal Clock Mode only)	P2.0 VREG Pin State

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

13.0 USB Serial Interface Engine (SIE)

The SIE allows the microcontroller to communicate with the USB host. The SIE simplifies the interface between the microcontroller and USB by incorporating hardware that handles the following USB bus activity independently of the microcontroller:

- Bit stuffing/unstuffing
- Checksum generation/checking
- ACK/NAK
- Token type identification
- Address checking

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

- Coordinate enumeration by responding to set-up packets
- Fill and empty the FIFOs
- Suspend/Resume coordination
- Verify and select Data toggle values

13.1 USB Enumeration

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

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

13.2 USB Port Status and Control

USB status and control is regulated by the USB Status and Control Register as shown in *Figure 13-1*. All reserved bits must be written to zero. All bits in the register are cleared during reset.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
PS/2 Pull-up Enable	VREG Enable	USB Reset-PS/2 Activity Interrupt Mode	Reserved	USB Bus Activity	Control Bit 2	Control Bit 1	Control Bit 0

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

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

The Bus Activity bit (bit 3) is a “sticky” bit that indicates if any non-idle USB event has occurred on the USB bus. The user firmware should check and clear this bit periodically to detect any loss of bus activity. Writing a “0” to the Bus Activity bit clears it while writing a “1” preserves the current value. In other words, the firmware can clear the Bus Activity bit, but only the SIE can set it. The 1.024-ms timer interrupt service routine is normally used to check and clear the Bus Activity bit.

Bits 4 is reserved and must be written as a 0.

The USB-PS/2 Interrupt Mode (bit 5) selects the definition of the USB Reset / PS/2 Activity Interrupt. The default cleared state puts the interrupt into USB mode. Setting this bit HIGH switches the interrupt definition to PS/2 mode. Details of the mode definitions are given in Section 21.3.1.

VREG Enable (bit 6) enables the 3.3V output voltage on the VREG pin when set to 1. This output is provided to source current for a 1.5-kΩ pull-up resistor connected to the D– pin. On reset, this bit is cleared, so the VREG pin is in high-impedance state.

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

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

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

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

14.0 USB Device

The CY7C637xx supports one USB Device Address with three endpoints: EP0, EP1, and EP2. Control Endpoint 0 (EP0) allows the USB host to recognize, set-up, and control the device. In particular, EP0 is used to receive and transmit control (including set-up) packets.

14.1 USB Address Register

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

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Device Address Enable	Device Address Bit 6	Device Address Bit 5	Device Address Bit 4	Device Address Bit 3	Device Address Bit 2	Device Address Bit 1	Device Address Bit 0

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

Bit 7 (Device Address Enable) in the USB Device Address Register must be set by firmware before the serial interface engine (SIE) will respond to USB traffic at this address. The Device Address in bits [6:0] must be set by firmware during the USB enumeration process to the non-zero address assigned by the USB host. This register is cleared by both hardware resets and the USB bus reset.

14.2 USB Control Endpoint

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

The endpoint mode registers are cleared during reset. The EP0 endpoint mode register uses the format shown in *Figure 14-2*.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Endpoint 0 SETUP Received	Endpoint 0 IN Received	Endpoint 0 OUT Received	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0

Figure 14-2. USB EP0 Mode Register (Address 0x12)

Bits[7:5] in the endpoint 0 mode registers are “sticky” status bits that are set by the SIE to report the type of token that was most recently received by the corresponding device address. The sticky bits must be cleared by firmware as part of the USB processing.

The ACK bit (bit 4) is set whenever the SIE engages in a transaction to the register’s endpoint that completes with an ACK packet.

The SETUP PID status (bit 7) is forced HIGH from the start of the data packet phase of the SETUP transaction, until the start of the ACK packet returned by the SIE. The CPU is prevented from clearing this bit during this interval, and subsequently until the CPU first does a IORD to this endpoint 0 mode register.

Bits[6:0] of the endpoint 0 mode register are locked from CPU write operations whenever the SIE has updated one of these bits, which the SIE does only at the end of a packet transaction (SETUP... Data... ACK, or OUT... Data... ACK, or IN... Data... ACK). The CPU can unlock these bits by doing a subsequent read of this register.

Because of these hardware locking features, firmware must perform an IORD after an IOWR to an endpoint 0 register to verify that the contents have changed as desired, and that the SIE has not updated these values.

While the SETUP bit is set, the CPU cannot write to the EP0 FIFO. This prevents firmware from overwriting an incoming SETUP transaction before firmware has a chance to read the SETUP data.

The Mode bits (bits [3:0]) control how the endpoint responds to USB bus traffic. The mode bit encoding is shown in *Table 22-1*. Additional information on the mode bits can be found in *Table 22-2* and *Table 22-3*.

14.3 USB Non-Control Endpoints (2)

The format of the non-control endpoint mode registers is shown in *Figure 14-3*. EP1 uses an 8-byte FIFO at SRAM locations 0xF0–0xF7, while EP2 uses an 8-byte FIFO at SRAM locations 0xE8–0xEF, as shown in Section 8.2.

7	6	5	4	3	2	1	0
			R/W	R/W	R/W	R/W	R/W
STALL	Reserved	Reserved	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0

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

The Mode bits (bits [3:0]) of the Endpoint Mode Registers control how the endpoint responds to USB bus traffic. The mode bit encoding is shown in *Table 22-1*.

The ACK bit (bit 4) is set whenever the SIE engages in a transaction to the register’s endpoint that completes with an ACK packet.

If STALL (bit 7) is set, the SIE will stall an OUT packet if the mode bits are set to ACK-IN, and the SIE will stall an IN packet if the mode bits are set to ACK-OUT. For all other modes the STALL bit must be a LOW.

Bits 5 and 6 are reserved and must be written to zero during register writes.

14.4 USB Endpoint Counter Registers

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

7	6	5	4	3	2	1	0
R/W	R/W			R/W	R/W	R/W	R/W
Data 0/1 Toggle	Data Valid	Reserved	Reserved	Byte Count Bit 3	Byte Count Bit 2	Byte Count Bit 1	Byte Count Bit 0

Figure 14-4. USB Device Counter Registers (Addresses 0x11h, 0x13h, 0x15)

The counter bits (bits [3:0]) indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint FIFO. Valid values are 0 to 8 inclusive. For OUT or SETUP transactions, the count is updated by hardware to the number of data bytes received, plus 2 for the CRC bytes. Valid values are 2 to 10 inclusive.

Data Valid bit 6 is used for OUT and SETUP tokens only. Data is loaded into the FIFOs during the transaction, and then the Data Valid bit will be set if a proper CRC is received. If the CRC is not correct, the endpoint interrupt will occur, but Data Valid will be cleared to a zero.

Data 0/1 Toggle bit 7 selects the DATA packet’s toggle state: 0 for DATA0, 1 for DATA1. For IN transactions, firmware must set this bit to the desired state. For OUT or SETUP transactions, the hardware sets this bit to the state of the received Data Toggle bit.

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

15.0 USB Regulator Output

The VREG pin provides a regulated output for connecting the pull-up resistor required for USB operation. For USB, a 1.5 kΩ resistor is connected between the D– pin and the VREG pin, to indicate low-speed USB operation. Since the VREG output has an internal series resistance of approximately 200Ω, the external pull-up resistor required is R_{PU} (see Section 24.0).

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

Note that enabling the device for USB (by setting the Device Address Enable bit, *Figure 14-1*) activates the internal regulator, even if the VREG Enable bit is cleared to 0. This insures proper USB signaling in the case where the VREG pin is used as an input, and an external regulator is provided for the USB pull-up resistor. This also limits the swing on the DM and DP 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 designed to provide current for the USB pull-up resistor, and can only source current up to I_{REG}. In addition, the output voltage at the VREG pin is effectively disconnected when the CY7C637xx device transmits USB from the internal SIE. This means that the VREG pin does not provide a stable voltage during transmits, although this does not affect USB signaling.

16.0 PS/2 Operation

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

1. USB D+ and D– lines can also be used for PS/2 SCLK and SDATA pins, respectively. With USB disabled, these lines can be placed in a high impedance state that will pull up to V_{CC}. (Disable USB by clearing the Address Enable bit of the USB Device Address Register, *Figure 14-1*).
2. An interrupt is provided to indicate a long LOW state on the SDATA pin. This eliminates the need to poll this pin to check for PS/2 activity. Refer to Section 21.3.1.
3. Internal PS/2 pull-up resistors can be enabled on the SCLK and SDATA lines, so no GPIO pins are required for this task (bit 7, USB Status and Control Register, *Figure 13-1*).
4. The controlled slew rate outputs from these pins apply to both USB and PS/2 modes to minimize EMI.
5. The state of the SCLK and SDATA pins can be read, and can be individually driven low in an open drain mode. The pins are read at bits [5:4] of Port 2, and are driven with the Control Bits [2:0] of the USB Status and Control Register.
6. The VREG pin can be placed into a high-impedance state, so that a USB pull-up resistor on the D–/SDATA pin will not interfere with PS/2 operation (bit 6, USB Status and Control Register).

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

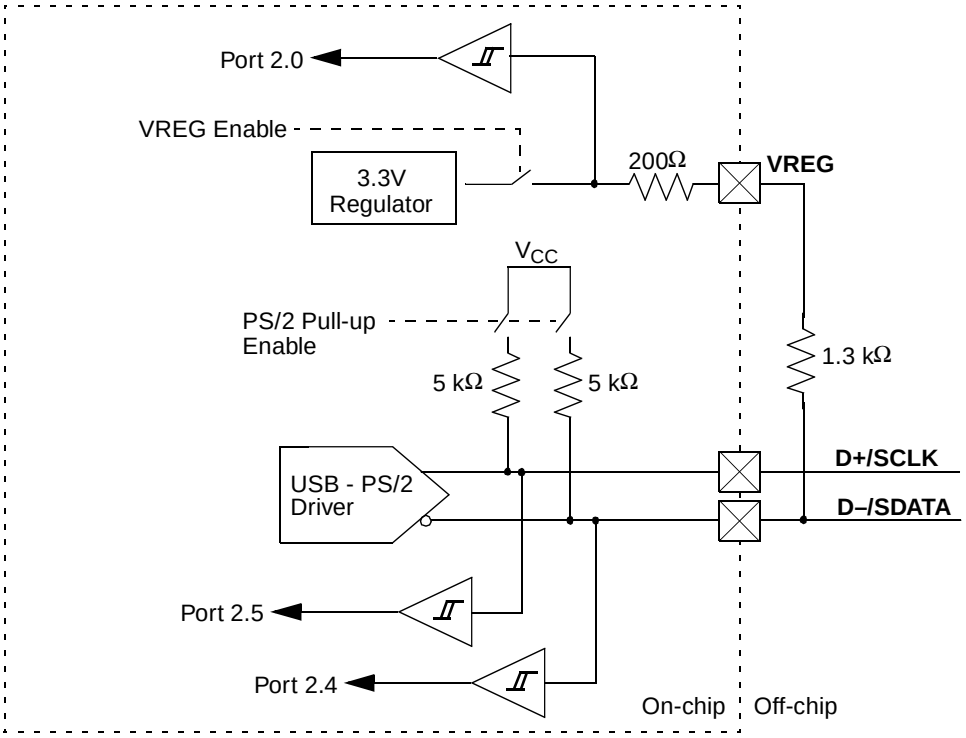


Figure 16-1. Diagram of USB - PS/2 System Connections

17.0 Serial Peripheral Interface (SPI)

SPI is a 4-wire, full-duplex serial communication interface between a master device and one or more slave devices. The CY7C637xx SPI circuit supports byte serial transfers in either Master or Slave modes. The block diagram of the SPI circuit is shown in *Figure 17-1*. The block contains buffers for both transmit and receive data for maximum flexibility and throughput. The CY7C637xx can be configured as either an SPI Master or Slave. The external interface consists of Master-Out/Slave-In (MOSI), Master-In/Slave-Out (MISO), Serial Clock (SCK), and Slave Select ($\overline{SS}$). Writes to the SPI Data Register (see *Figure 17-2*) load the transmit buffer, while reads from this register read the receive buffer contents.

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

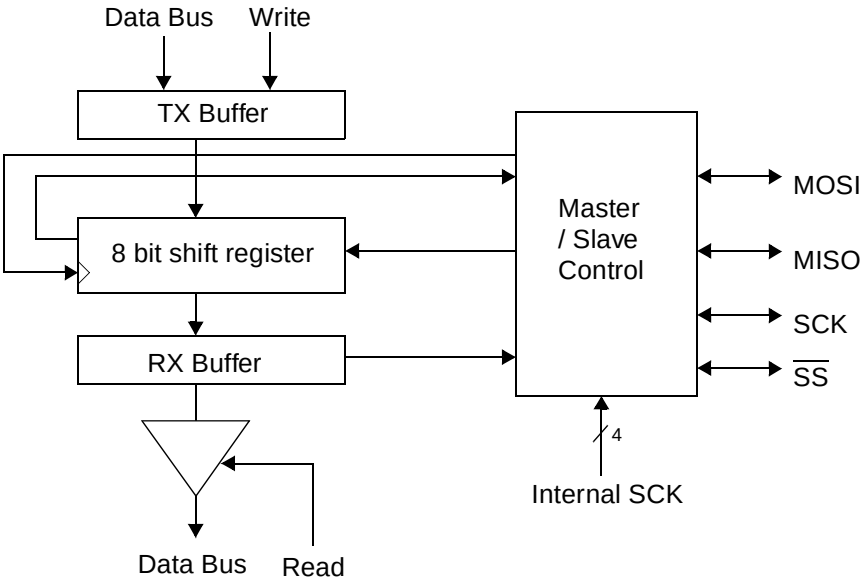


Figure 17-1. SPI Block Diagram

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Data I/O [7]	Data I/O [6]	Data I/O [5]	Data I/O [4]	Data I/O [3]	Data I/O [2]	Data I/O [1]	Data I/O [0]

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

17.1 Operation as an SPI Master

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

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

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

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

17.2 Master SCK Selection

The Master's SCK is programmable to one of four clock settings, as shown in *Figure 17-1*. The frequency is selected with the Clock Select Bits of the SPI control register. The hardware provides 8 output clocks on the SCK pin (P0.7) for each byte transfer. Clock phase and polarity are selected by the CPHA and CPOL control bits (see *Figures 17-1* and *17-4*).

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

17.3 Operation as an SPI Slave

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

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

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

Clock phase and polarity must be selected to match the SPI master, using the CPHA and CPOL control bits (see *Table 17-1* and *Figure 17-4*).

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

17.4 SPI Status and Control

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

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
TCMP	TBF	Mode[1]	Mode[0]	CPOL	CPHA	SCK Select [1]	SCK Select [0]

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

Table 17-1. SPI Control Register Definitions

Bit(s)	Definition	Function	
1:0	SCK Select	Master mode SCK frequency selection (no effect in Slave Mode):	
		00	2 Mbit/s
		01	1 Mbit/s
		10	0.5 Mbit/s
		11	0.0625 Mbit/sec
2	CPHA	SPI Clock Phase (see *Figure 17-4*)	
3	CPOL	SPI Clock Polarity (see *Figure 17-4*): 0 SCK idles LOW, 1 SCK idles HIGH	
5:4	Comm Modes	00	All Communications functions disabled (default)
		01	SPI Master Mode
		10	SPI Slave Mode
		11	reserved
6	TBF	Transmit Buffer Full. TBF=1 indicates data in the transmit buffer has not transferred to the shift register.	
7	TCMP	Transfer Complete. TCMP is set to 1 by the hardware when 8 bit transfer is complete. This bit is only cleared by firmware. The SPI interrupt is asserted at the same time TCMP is set to 1.	

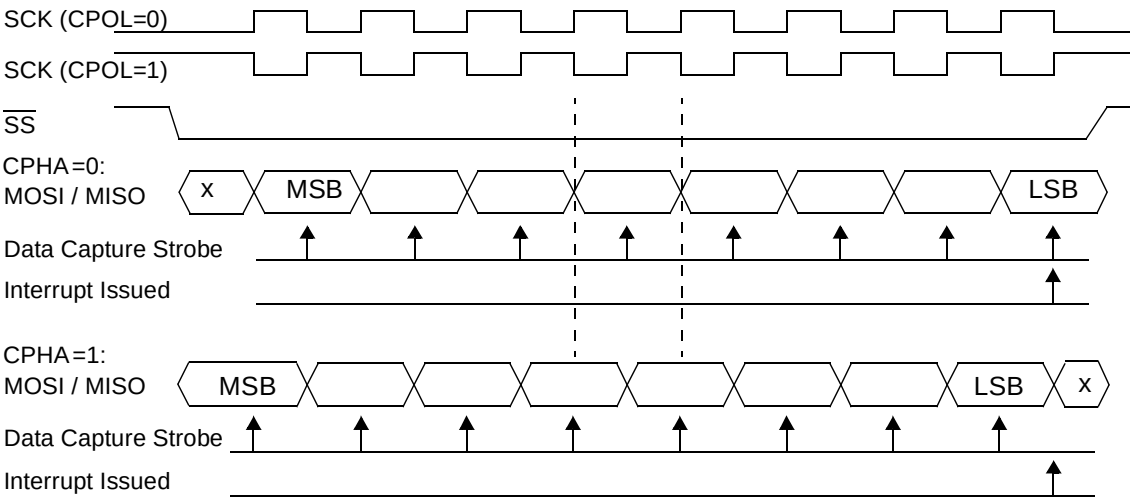


Figure 17-4. SPI Data Timing

17.5 SPI Interrupt

For SPI, an interrupt request is generated after a byte is received or transmitted. See Section 21.3.4 for details on the SPI interrupt.

17.6 SPI modes for GPIO pins

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

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

Table 17-2. SPI Pin Assignments

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

18.0 12-bit Free-running Timer

The 12-bit timer operates with a 1-µs tick, provides two interrupts (128 µs and 1.024 ms) and allows the firmware to directly time events that are up to 4 ms in duration. The lower 8 bits of the timer can be read directly by the firmware. Reading the lower 8 bits latches the upper 4 bits into a temporary register. When the firmware reads the upper 4 bits of the timer, it is actually reading the count stored in the temporary register. The effect of this is to ensure a stable 12-bit timer value can be read, even when the two reads are separated in time.

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
Timer Bit 7	Timer Bit 6	Timer Bit 5	Timer Bit 4	Timer Bit 3	Timer Bit 2	Timer Bit 1	Timer Bit 0

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

7	6	5	4	3	2	1	0
				R	R	R	R
Reserved	Reserved	Reserved	Reserved	Timer Bit 11	Timer Bit 10	Timer Bit 9	Timer Bit 8

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

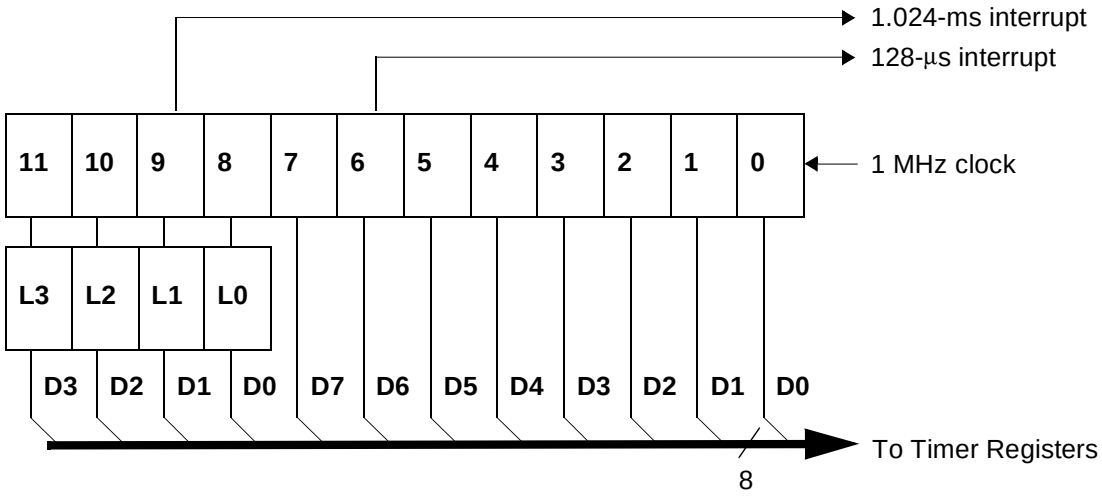


Figure 18-3. Timer Block Diagram

19.0 Timer Capture Registers

Four 8-bit timer capture 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 GPIO events occur. Each timer will capture 8 bits of the free-running timer into a data register. A prescaler allows selection of the capture timer tick size. Interrupts can be individually enabled for the four capture registers. A block diagram is shown in *Figure 19-1*.

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

The four capture data registers are read-only, and are shown in *Figure 19-2* through *Figure 19-5*.

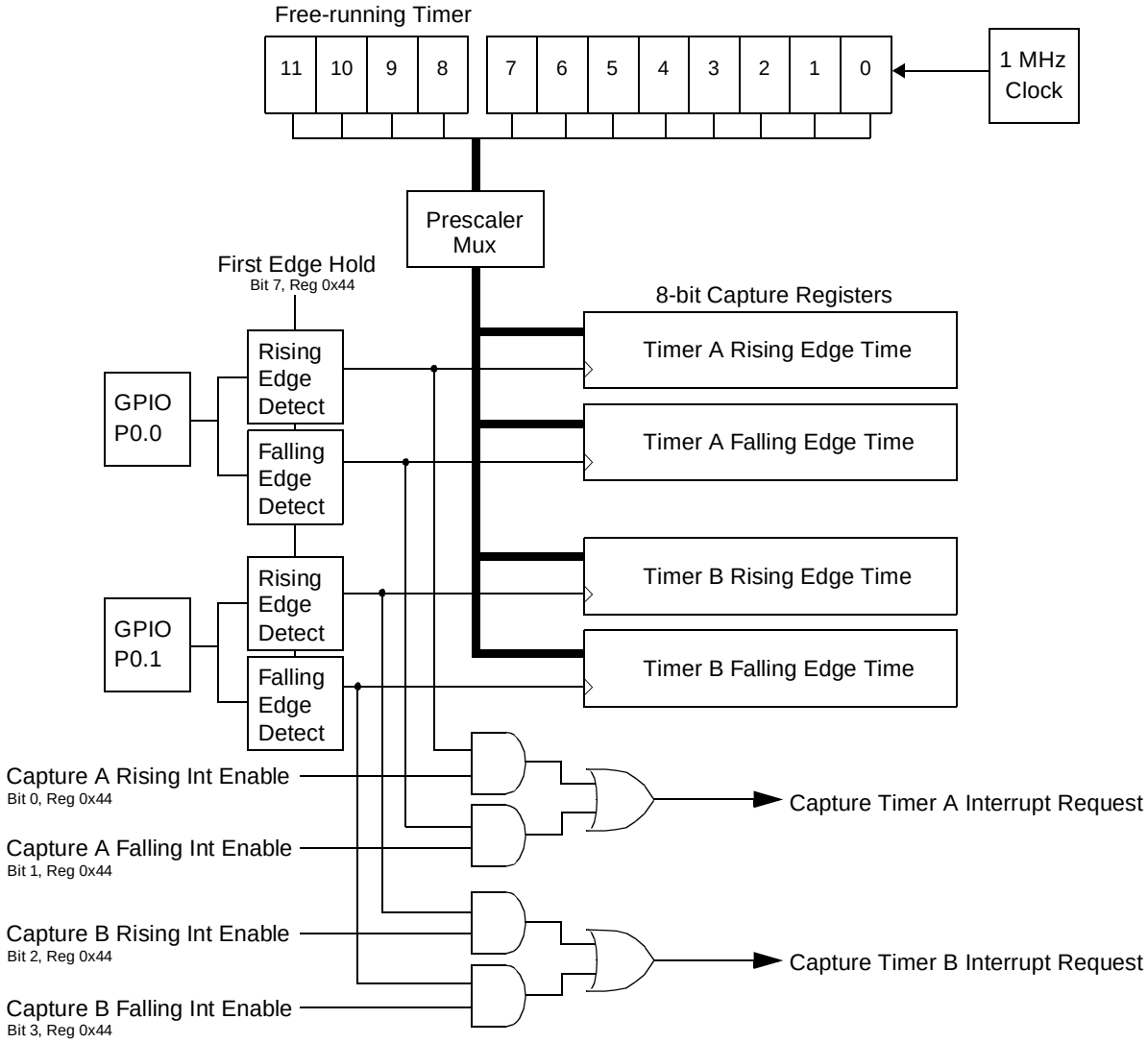


Figure 19-1. Capture Timers Block Diagram

Three prescaler bits allow the capture timer clock rate to be selected among 5 choices, as shown in *Table 19-1* below. The Capture Status Register (*Figure 19-6*) contains the prescale settings and the interrupt enables for the 4 possible events. Setting an enable bit allows for an interrupt from the respective timer event. Note that 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 in the Global Interrupt Enable register (Section 21.0) must be set to activate a capture interrupt.

The Capture Status Register (*Figure 19-7*) records 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.

By default, a capture timer register holds the time of the most recent edge for that register (i.e. if multiple edges have occurred before reading the capture timer, the time for the last one will be read). Setting the global First Edge Hold (bit 7, *Figure 19-6*) modifies this so that the first occurrence of an edge is held in the capture register until the data is read. In this case, subsequent edges are ignored until the capture register is read. The First Edge Hold function applies globally to all four capture timers.

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
Capture A Rising Bit 7	Capture A Rising Bit 6	Capture A Rising Bit 5	Capture A Rising Bit 4	Capture A Rising Bit 3	Capture A Rising Bit 2	Capture A Rising Bit 1	Capture A Rising Bit 0

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

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
Capture A Falling Bit 7	Capture A Falling Bit 6	Capture A Falling Bit 5	Capture A Falling Bit 4	Capture A Falling Bit 3	Capture A Falling Bit 2	Capture A Falling Bit 1	Capture A Falling Bit 0

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

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
Capture B Rising Bit 7	Capture B Rising Bit 6	Capture B Rising Bit 5	Capture B Rising Bit 4	Capture B Rising Bit 3	Capture B Rising Bit 2	Capture B Rising Bit 1	Capture B Rising Bit 0

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

7	6	5	4	3	2	1	0
R	R	R	R	R	R	R	R
Capture B Falling Bit 7	Capture B Falling Bit 6	Capture B Falling Bit 5	Capture B Falling Bit 4	Capture B Falling Bit 3	Capture B Falling Bit 2	Capture B Falling Bit 1	Capture B Falling Bit 0

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

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
First Edge Hold	Prescale Bit 2	Prescale Bit 1	Prescale Bit 0	Capture B Falling Int Enable	Capture B Rising Int Enable	Capture A Falling Int Enable	Capture A Rising Int Enable

Figure 19-6. Capture Timers Configuration Register (Address 0x44)

7	6	5	4	3	2	1	0
-	-	-	-	R	R	R	R
Reserved	Reserved	Reserved	Reserved	Capture B Falling Event	Capture B Rising Event	Capture A Falling Event	Capture A Rising Event

Figure 19-7. Capture Timers Status Register (Address 0x45)

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

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

20.0 Processor Status and Control Register

7	6	5	4	3	2	1	0
R	R/W	R/W	R/W	R/W	R	R/W	R/W
IRQ Pending	Watch Dog Reset	Bus Interrupt Event	Low Voltage or Brown-Out Reset	Suspend	Interrupt Enable Sense	Reserved	Run

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

The Run bit (bit 0) 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 watch dog). This bit should normally be written as a 1.

Bit 1 is a reserved bit that must be written as a 0.

The Interrupt Enable Sense (bit 2) 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. A ‘0’ indicates that interrupts are masked off and a ‘1’ indicates that the interrupts are enabled. This bit is further gated with the bit settings of the Global Interrupt Enable Register (0x20) and USB Endpoint Interrupt Enable Register (0x21). Instructions DI, EI, and RETI manipulate the state of this bit.

Writing a 1 to the Suspend bit (bit 3) will halt the processor and cause the microcontroller to enter the suspend mode that significantly reduces power consumption. A pending, enabled interrupt or USB bus activity will cause the device to come out of suspend. After coming out of suspend, the device will resume firmware execution at the instruction following the IOWR which put the part into suspend. When writing the suspend bit with a resume condition present (such as non-idle USB activity), the suspend state will still be entered, followed immediately by the wake-up process (with appropriate delays for the clock start-up). See Section 11.0 for more details on suspend mode operation.

The Low-Voltage or Brown-Out Reset (bit 4) 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 watch dog timeout. (Note that a LVR/BOR event may be followed by a watch dog reset before firmware begins executing, as explained below.)

The Bus Interrupt Event (bit 5) is set whenever the event for the USB Bus Reset / 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 (see *Figure 13-1*). The details on the event conditions that set this bit are given in Section 21.3.1. In either mode, this bit is set as soon as the event has lasted for the specified time (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.

The Watch Dog Reset (bit 6) is set during a reset initiated by the Watch Dog Timer. This indicates the Watch Dog Timer went for more than t_{WATCH} (8 ms minimum) between Watch Dog clears. This can occur with a POR/LVR event, as noted below.

IRQ pending (bit 7), when set, indicates one or more of the interrupts has been recognized as active. This bit is only valid if the Global Interrupt Enable bit is disabled. An interrupt will remain pending until its interrupt enable bit is set (registers 0x20 or 0x21) and interrupts are globally enabled. At that point the internal interrupt handling sequence will clear this bit until another interrupt is detected as pending.

During power-up, or during a low-voltage reset, the Processor Status and Control Register is set to 00010001, which indicates a LVR/BOR (bit 4 set) has occurred and no interrupts are pending (bit 7 clear). Note that during the t_{START} ms partial suspend at start-up (explained in Section 10.1), a Watch Dog 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 Watch Dog Reset, the Processor Status and Control Register is set to 01XX0001, which indicates a Watch Dog Reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear).

21.0 Interrupts

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

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
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 Reset / PS/2 Activity Intr. Enable

Figure 21-1. Global Interrupt Enable Register 0x20h (read/write)

7	6	5	4	3	2	1	0
					R/W	R/W	R/W
Reserved	Reserved	Reserved	Reserved	Reserved	EP2 Interrupt Enable	EP1 Interrupt Enable	EP0 Interrupt Enable

Figure 21-2. USB End Point Interrupt Enable Register (Address 0x21)

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

When servicing an interrupt, the hardware will first disable all interrupts by clearing the Global Interrupt Enable bit in the CPU (the state of this bit can be read at Bit 2 of the Processor Status and Control Register). Next, the flip-flop of the current interrupt is cleared. This is followed by an automatic CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector, see Section 21.1). The instruction in the interrupt table is typically a JMP instruction to the address of the Interrupt Service Routine (ISR). The user can re-enable interrupts in the interrupt service routine by executing an EI instruction. Interrupts can be nested to a level limited only by the available stack space.

The Program Counter value as well as the Carry and Zero flags (CF, ZF) are stored onto the Program Stack by the automatic CALL instruction generated as part of the interrupt acknowledge process. The user firmware is responsible for insuring 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).

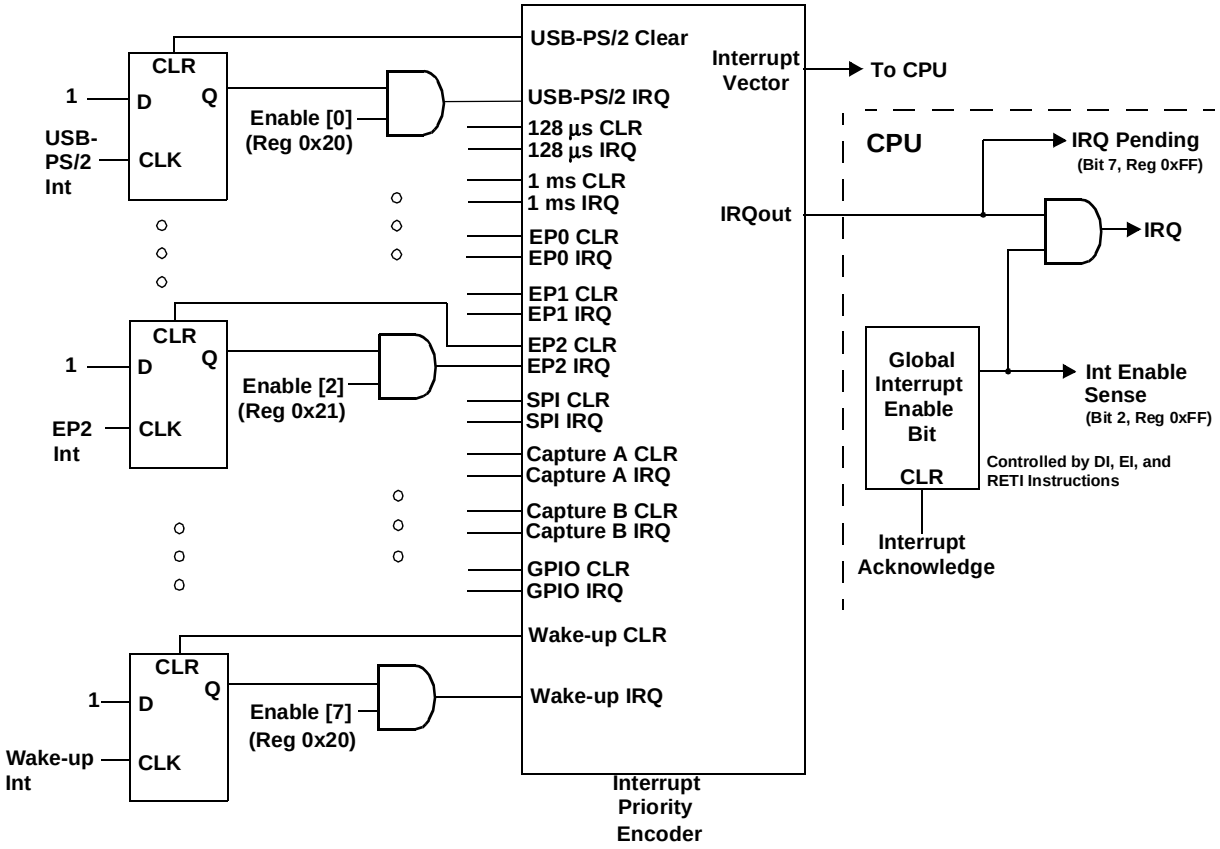


Figure 21-3. Interrupt Controller Logic Block Diagram

21.1 Interrupt Vectors

The Interrupt Vectors supported by the device are listed in *Table 21-1*. The highest priority interrupt is #1 (USB Bus Reset / PS/2 activity), and the lowest priority interrupt is #11 (Wake-up Timer). Although Reset is not an interrupt, the first instruction executed after a reset is at ROM address 0x0000h, which corresponds to the first entry in the Interrupt Vector Table. Interrupt vectors occupy 2 bytes to allow for a 2 byte JMP instruction to the appropriate Interrupt Service Routine (ISR).

Table 21-1. Interrupt Vector Assignments

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

21.2 Interrupt Latency

Interrupt latency can be calculated from the following equation:

Interrupt Latency = (Number of clock cycles remaining in the current instruction) + (10 clock cycles for the CALL instruction) + (5 clock cycles for the JMP instruction)

For example, if a 5 clock cycle instruction such as JC is being executed when an interrupt occurs, the first instruction of the Interrupt Service Routine will execute a minimum of 16 clocks (1+10+5) or a maximum of 20 clocks (5+10+5) after the interrupt is issued. With a 6 MHz external resonator, internal CPU clock speed is 12 MHz, so 20 clocks take 20 / 12 MHz = 1.67 µs.

21.3 Interrupt Sources

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

21.3.1 USB Bus Reset or PS/2 Activity

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

A USB bus reset is indicated by a single ended zero (SE0) on the USB D+ and D– pins. The USB 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.

21.3.2 Free Running Timer Interrupts

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

21.3.3 USB Endpoint Interrupts

There are three USB endpoint interrupts, one per endpoint. A USB endpoint interrupt is generated after the USB host writes to a USB endpoint FIFO or after the USB controller sends a packet to the USB host. The interrupt is generated on the last packet of the transaction (e.g., on the host's ACK during an IN, or on the device ACK during on OUT). If no ACK is received during an IN transaction, no interrupt will be generated.

21.3.4 SPI Interrupt

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

21.3.5 Capture Timer Interrupts

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

21.3.6 GPIO Interrupt

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

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

All of the GPIO pins share a single interrupt vector, which means the firmware will need to read the GPIO ports with enabled interrupts to determine which pin or pins caused an interrupt.The GPIO interrupt structure is illustrated in *Figure 21-8*.

Note that if one port pin triggered an interrupt, no other port pins can cause a GPIO interrupt until that port pin has returned to its inactive (non-trigger) state or its corresponding port interrupt enable bit is cleared. The CY7C637xx does not assign interrupt priority to different port pins and the Port Interrupt Enable Registers are not affected by the interrupt acknowledge process.

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P0.7 Intr Enable	P0.6 Intr Enable	P0.5 Intr Enable	P0.4 Intr Enable	P0.3 Intr Enable	P0.2 Intr Enable	P0.1 Intr Enable	P0.0 Intr Enable

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

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P1.7 Intr Enable	P1.6 Intr Enable	P1.5 Intr Enable	P1.4 Intr Enable	P1.3 Intr Enable	P1.2 Intr Enable	P1.1 Intr Enable	P1.0 Intr Enable

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

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P0.7 Intr Polarity	P0.6 Intr Polarity	P0.5 Intr Polarity	P0.4 Intr Polarity	P0.3 Intr Polarity	P0.2 Intr Polarity	P0.1 Intr Polarity	P0.0 Intr Polarity

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

7	6	5	4	3	2	1	0
W	W	W	W	W	W	W	W
P1.7 Intr Polarity	P1.6 Intr Polarity	P1.5 Intr Polarity	P1.4 Intr Polarity	P1.3 Intr Polarity	P1.2 Intr Polarity	P1.1 Intr Polarity	P1.0 Intr Polarity

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

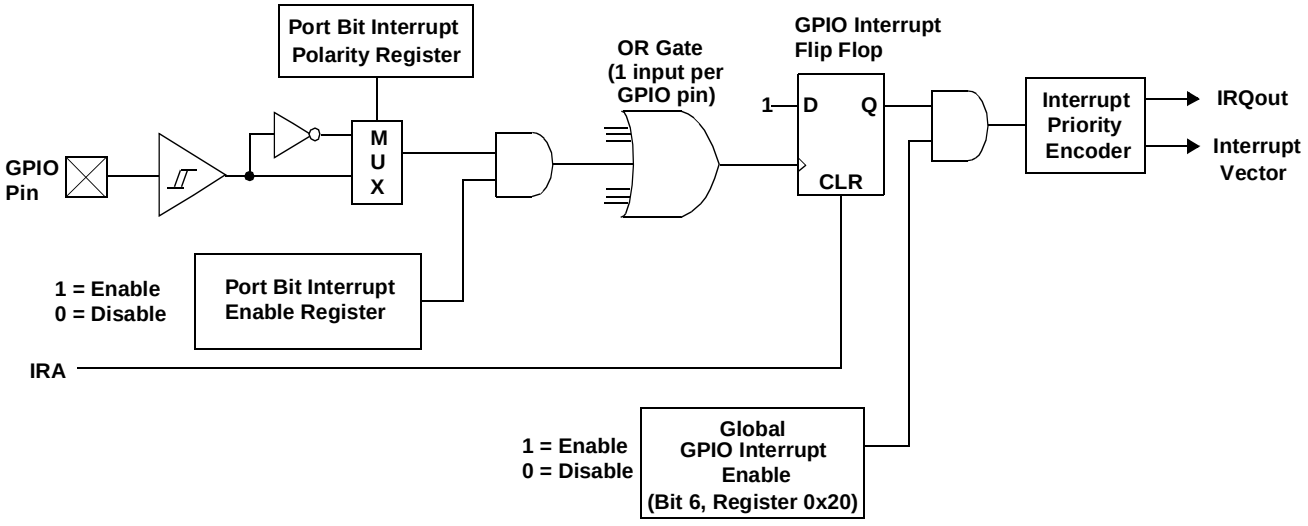


Figure 21-8. GPIO Interrupt Diagram

21.3.7 Wake-up Interrupt

The internal wake-up timer can be used to wake the part from suspend mode (although it can also provide an interrupt when the part is awake). The wake-up timer is cleared whenever the wake-up interrupt enable bit (Register 0x20) is written to a 0, and runs whenever that bit is written to a 1. When the interrupt is enabled, the wake-up timer provides periodic interrupts with period t_{WAKE}. The wake-up time can be adjusted by firmware as explained in Section 11.2.

22.0 USB Mode Tables

The following tables give details on mode setting for the USB Serial Interface Engine.

Table 22-1. USB Register Mode Encoding

Mode	Encoding	Setup	In	Out	Comments
Disable	**0000**	ignore	ignore	ignore	Ignore all USB traffic to this endpoint
Nak In/Out	**0001**	accept	NAK	NAK	Forced from Setup on Control endpoint, from modes other than 0000
Status Out Only	**0010**	accept	stall	check	For Control endpoints
Stall In/Out	**0011**	accept	stall	stall	For Control endpoints
Ignore In/Out	**0100**	accept	ignore	ignore	For Control endpoints
Reserved	**0101**	ignore	ignore	always	Not Complaint or Low-speed device
Status In Only	**0110**	accept	TX 0	stall	For Control Endpoints
Reserved	**0111**	ignore	TX cnt	ignore	Not Complaint or Low-speed device
Nak Out	**1000**	ignore	ignore	NAK	An ACK from mode 1001 --> 1000
Ack [3]=0) Ack Out(STALL[3]=1)	**1001** **1001**	ignore ignore	ignore ignore	ACK stall	This mode is changed by SIE on issuance of ACK --> 1000
Nak Out - Status In	**1010**	accept	TX 0	NAK	An ACK from mode 1011 --> 1010
Ack Out - NAK In	**1011**	accept	NAK	ACK	This mode is changed by SIE on issuance of ACK --> 0001
Nak In	**1100**	ignore	NAK	ignore	An ACK from mode 1101 --> 1100
Ack IN(STALL[3]=0) Ack IN(STALL[3]=1)	**1101** **1101**	ignore ignore	TX cnt stall	ignore ignore	This mode is changed by SIE on issuance of ACK --> 1100
Nak In - Status Out	**1110**	accept	NAK	check	An ACK from mode 1111 --> 111 Ack In - Status Out
Ack In - Status Out	**1111**	accept	TX cnt	check	This mode is changed by SIE on issuance of ACK -->1110

Note:

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

The 'In' column represents the SIE's response to the token type.

A disabled endpoint will remain disabled until changed by firmware, and all endpoints reset to the disabled state.

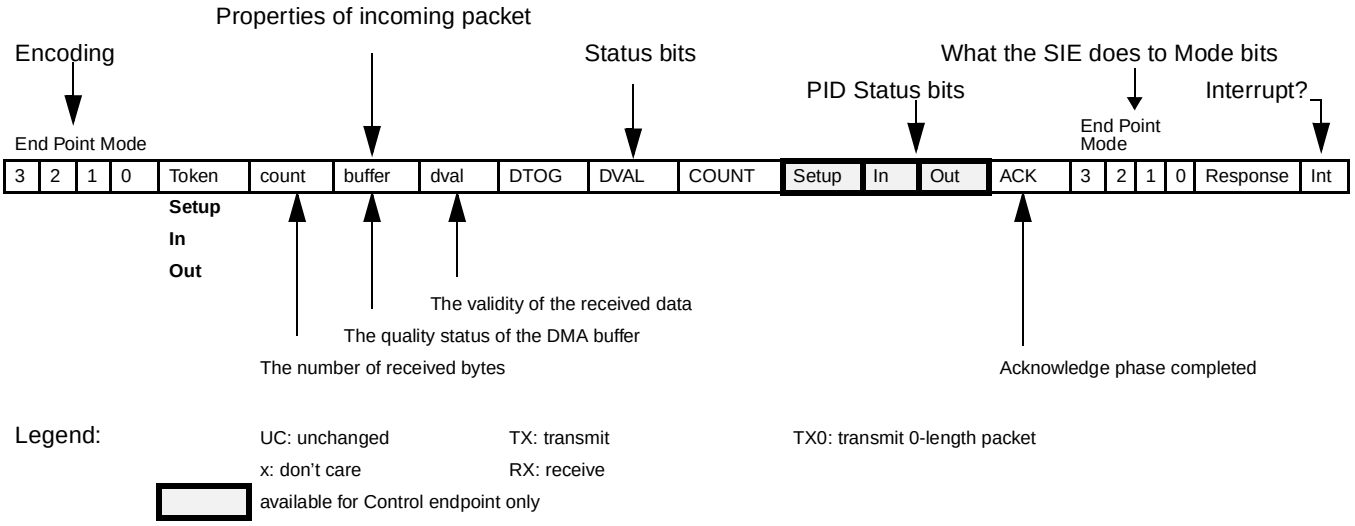
Any SETUP packet to an enabled endpoint with mode set to accept SETUPs will be changed by the SIE to 0001 (NAKing). Any mode set to accept a SETUP will ACK a valid SETUP transaction.

Most modes that control transactions involving an ending ACK will be changed by the SIE to a corresponding mode which NAKs subsequent packets following the ACK. Exceptions are modes 1010 and 1110.

A Control endpoint has three extra status bits for PID (Setup, In and Out), but must be placed in the correct mode to function as such. Non-Control endpoints should not be placed into modes that accept SETUPs.

A 'check' on an Out token during a Status transaction checks to see that the Out is of zero length and has a Data Toggle (DTOG) of 1.

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



The response of the SIE can be summarized as follows:

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

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

End Point Mode											PID				Set End Point Mode					
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	Setup	In	Out	ACK	3	2	1	0	response	int
Setup Packet (if accepting)																				
SeeTable 22-1.				Setup	<= 10	data	valid	updates	1	updates	1	UC	UC	1	0	0	0	1	ACK	yes
SeeTable 22-1.				Setup	> 10	junk	x	updates	updates	updates	1	UC	UC	UC	NoChange				ignore	yes
See Table 22-1.				Setup	x	junk	invalid	updates	0	updates	1	UC	UC	UC	NoChange				ignore	yes
Disabled																				
0	0	0	0	x	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
Nak In/Out																				
0	0	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes
1	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	0	0	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				NAK	yes
Ignore In/Out																				
0	1	0	0	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	0	0	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
Stall In/Out																				
0	0	1	1	Out	x	UC	x	UC	UC	UC	UC	UC	1	UC	NoChange				Stall	yes
1	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	0	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				Stall	yes
Control Write																				
Normal Out/**NAK In**																				
1	0	1	1	Out	<= 10	data	valid	updates	1	updates	UC	UC	1	1	**0**	**0**	**0**	**1**	ACK	yes
1	0	1	1	Out	> 10	junk	x	updates	updates	updates	UC	UC	1	UC	NoChange				ignore	yes
1	0	1	1	Out	x	junk	invalid	updates	0	updates	UC	UC	1	UC	NoChange				ignore	yes
1	0	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	**UC**	NoChange				**NAK**	yes
NAK Out/premature status In																				
1	0	1	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes
1	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	1	NoChange				TX 0	yes
Status In/extra Out																				
0	1	1	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	0	0	1	1	Stall	yes
0	1	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	1	NoChange				TX 0	yes
Control Read																				
Normal In/premature status Out																				
1	1	1	1	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
1	1	1	1	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	1	1	1	0	ACK (back)	yes
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	Setup	In	Out	ACK	3	2	1	0	response	int
Nak In/premature status Out																				
1	1	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
1	1	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	0	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no

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

1	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
1	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange	NAK	yes
Status Out/extra In																	
0	0	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange	ACK	yes
0	0	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0 0 1 1	Stall	yes
0	0	1	0	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0 0 1 1	Stall	yes
0	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
0	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
0	0	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	0 0 1 1	Stall	yes
Out endpoint																	
Normal Out/erroneous In																	
1	0	0	1	Out	<= 10	data	valid	updates	1	updates	UC	UC	1	1	1 0 0 0	ACK	yes
1	0	0	1	Out	> 10	junk	x	updates	updates	updates	UC	UC	1	UC	NoChange	ignore	yes
1	0	0	1	Out	x	junk	invalid	updates	0	updates	UC	UC	1	UC	NoChange	ignore	yes
1	0	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
NAK Out/erroneous In																	
1	0	0	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	NoChange	NAK	yes
1	0	0	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
1	0	0	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
1	0	0	0	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
Reserved																	
0	1	0	1	Out	x	updates	updates	updates	updates	updates	UC	UC	1	1	NoChange	RX	yes
0	1	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
In endpoint																	
Normal In/erroneous Out																	
1	1	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
1	1	0	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	1 1 0 0	ACK (back)	yes
NAK In/erroneous Out																	
1	1	0	0	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
1	1	0	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange	NAK	yes
Reserved																	
0	1	1	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange	ignore	no
0	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange	TX	yes

23.0 Absolute Maximum Ratings

Storage Temperature .....–65°C to +150°C
Ambient Temperature with Power Applied .....–0°C to +70°C
Supply Voltage on V_{CC} Relative to V_{SS} .....–0.5V to +7.0V
DC Input Voltage..... –0.5V to +VCC+0.5V
DC Voltage Applied to Outputs in High Z State ..... –0.5V to + VCC+0.5V
Maximum Total Sink Output Current into Port 0 and 1 and Pins..... 70 mA
Maximum Total Source Output Current into Port 0 and 1 and Pins ..... 30 mA
Maximum On-chip Power Dissipation on any GPIO Pin .....50 mW
Power Dissipation .....300 mW
Static Discharge Voltage ..... >2000V
Latch-up Current ..... >200 mA

24.0 DC Characteristics

F_{OSC} = 6 MHz; Operating Temperature = 0 to 70°C

	Parameter	Min	Max	Units	Conditions
	General				
V_{CC1}	Operating Voltage	4.0	5.5	V	Note 4
V_{CC2}	Operating Voltage	4.35	5.25	V	Note 4
I_{CC}	V_{CC} Operating Supply Current		25	mA	V_{CC} = 5.5V, no GPIO loading
I_{SB1}	Standby Current - No Wake-up Osc		25	µA	Oscillator off, D– > 2.8V
I_{SB2}	Standby Current - With Wake-up Osc		75	µA	Oscillator off, D– > 2.8V
V_{PP}	Programming Voltage (disabled)	–0.4	0.4	V	
T_{RSNTR}	Resonator Start-up Interval		256	µs	V_{CC} = 5.0V, ceramic resonator
I_{IL}	Input Leakage Current		1	µA	Any I/O pin
I_{SNK}	Max I_{SS} GPIO Sink Current		70	mA	Cumulative across all ports[5]
I_{SRC}	Max I_{CC} GPIO Source Current		30	mA	Cumulative across all ports[5]
	Low Voltage & Power-On Reset				
V_{LVR}	Low-Voltage Reset Trip Voltage	3.6	3.9	V	V_{CC} below V_{LVR} for >100 ns[6]
t_{VCCS}	V_{CC} Power-on Slew Time		100	ms	linear ramp: 0 to 4V[7]
	USB Interface				
V_{RG}	VREG Regulator Output Voltage	3.0	3.6	V	Load = R_{PU} +R_{PD}[8, 9]
C_{REG}	Capacitance on VREG Pin		300	pF	External cap not required
V_{OHU}	Static Output High, driven	2.8	3.6	V	R_{PD} to Gnd[4]
V_{OLU}	Static Output Low		0.3	V	With R_{PU} to VREG pin
V_{OHZ}	Static Output High, idle or suspend	2.7	3.6	V	R_{PD} connected D– to Gnd, R_{PU} connected D– to VREG pin[4]
V_{DI}	Differential Input Sensitivity	0.2		V	|(D+)–(D–)|
V_{CM}	Differential Input Common Mode Range	0.8	2.5	V	
V_{SE}	Single Ended Receiver Threshold	0.8	2.0	V	
C_{IN}	Transceiver Capacitance		20	pF	
I_{LO}	Hi-Z State Data Line Leakage	–10	10	µA	0 V < V_{in}<3.3 V (D+ or D– pins)
R_{PU}	External Bus Pull-up resistance (D–)	1.274	1.326	kΩ	1.3 kΩ ±2% to VREG[10]
R_{PD}	External Bus Pull-down resistance	14.25	15.75	kΩ	15 kΩ ±5% to Gnd
	PS/2 Interface				
V_{OLP}	Static Output Low		0.4	V	Isink = 5 mA, SDATA or SCLK pins
R_{PS2}	Internal PS/2 Pull-up Resistance	3	7	kΩ	SDATA, SCLK pins, PS/2 Enabled
	General Purpose I/O Interface				
R_{UP}	Pull-up Resistance	8	24	kΩ	
V_{ICR}	Input Threshold Voltage, CMOS mode	40%	60%	V_{CC}	Low to high edge, Port 0 or 1
V_{ICF}	Input Threshold Voltage, CMOS mode	35%	55%	V_{CC}	High to low edge, Port 0 or 1
V_{HC}	Input Hysteresis Voltage, CMOS mode	3%	10%	V_{CC}	High to low edge, Port 0 or 1
V_{ITTL}	Input Threshold Voltage, TTL mode	0.8	2.0	V	Ports 0, 1, and 2
V_{OL1A}	Output Low Voltage, high drive mode		0.8	V	I_{OL1} = 50 mA, Ports 0 or 1[4]
V_{OL1B}			0.4	V	I_{OL1} = 25 mA, Ports 0 or 1[4]
V_{OL2}	Output Low Voltage, medium drive mode		0.4	V	I_{OL2} = 8 mA, Ports 0 or 1[4]
V_{OL3}	Output Low Voltage, low drive mode		0.4	V	I_{OL3} = 2 mA, Ports 0 or 1[4]
V_{OH}	Output High Voltage, strong drive mode	V_{CC}–2		V	Port 0 or 1, I_{OH} = 2 mA[4]
R_{XIN}	Pull-down resistance, XTALIN pin	50		kΩ	Internal Clock Mode only

25.0 Switching Characteristics

Parameter	Description	Min.	Max.	Unit	Conditions
	Internal Clock Mode				
F_{ICLK}	Internal Clock Frequency	5.7	6.3	MHz	Internal Clock Mode enabled
F_{ICLK2}	Internal Clock Frequency, USB mode	5.91	6.09	MHz	Internal Clock Mode enabled, Bit 2 of register 0xF8h is set (Precision USB Clocking)[11]
	External Oscillator Mode				
T_{CYC}	Input Clock Cycle Time	164.2	169.2	ns	USB Operation, with External ±1.5% Ceramic Resonator or Crystal
T_{CH}	Clock HIGH Time	0.45 t_{CYC}		ns	
T_{CL}	Clock LOW Time	0.45 t_{CYC}		ns	
T_{START}	Time-out Delay after LVR/BOR	24	60	ms	
T_{WAKE}	Internal Wake-up Period	1	5	ms	Enabled Wake-up Interrupt[12]
T_{WATCH}	WatchDog Timer Period	10.1	14.6	ms	F_{OSC} = 6 MHz
	USB Driver Characteristics				
T_R	Transition Rise Time	75		ns	CLoad = 200 pF (10% to 90%[4])
T_R	Transition Rise Time		300	ns	CLoad = 600 pF (10% to 90%[4])
T_F	Transition Fall Time	75		ns	CLoad = 200 pF (10% to 90%[4])
T_F	Transition Fall Time		300	ns	CLoad = 600 pF (10% to 90%[4])
T_{RFM}	Rise/Fall Time Matching	80	125	%	t_r/t_f[4, 13]
V_{CRS}	Output Signal Crossover Voltage	1.3	2.0	V	CLoad = 200 to 600 pF[4]
	USB Data Timing				
T_{DRATE}	Low Speed Data Rate	1.4775	1.5225	Mb/s	Ave. Bit Rate (1.5 Mb/s ±1.5%)
T_{DJR1}	Receiver Data Jitter Tolerance	–75	75	ns	To Next Transition[14]
T_{DJR2}	Receiver Data Jitter Tolerance	–45	45	ns	For Paired Transitions[14]
T_{DEOP}	Differential to EOP transition Skew	–40	100	ns	Note 14
T_{EOPR2}	EOP Width at Receiver	670		ns	Accepts as EOP[14]
T_{EOPT}	Source EOP Width	1.25	1.50	µs	
T_{UDJ1}	Differential Driver Jitter	–95	95	ns	To next transition, *Figure 25-5*
T_{UDJ2}	Differential Driver Jitter	–150	150	ns	To paired transition, *Figure 25-5*
T_{LST}	Width of SE0 during Diff. Transition		210	ns	
	Non-USB Mode Driver Characteristics				Note 15
T_{FPS2}	SDATA / SCK Transition Fall Time	50	300	ns	CLoad = 150 pF to 600 pF
	SPI Timing				See *Figures 25-6 to 25-9*[16]
T_{SMCK}	SPI Master Clock Rate		2	MHz	F_{CLK}/3; see *Figure 17-1*
T_{SSCK}	SPI Slave Clock Rate		2.2	MHz	
T_{SCKH}	SPI Clock High Time	125		ns	High for CPOL=0, Low for CPOL=1
T_{SCKL}	SPI Clock Low Time	125		ns	Low for CPOL=0, High for CPOL=1
T_{MDO}	Master Data Output Time	–25	50	ns	SCK to data valid

Parameter	Description	Min.	Max.	Unit	Conditions
T_{MDO1}	Master Data Output Time, First bit with CPHA=1	100		ns	Time before leading SCK edge
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		100	ns	SCK to data valid
T_{SDO1}	Slave Data Output Time, First bit with CPHA=1		100	ns	Time after $\overline{SS}$ LOW to data valid
T_{SSS}	Slave Select Set-Up Time	150		ns	Before first SCK edge
T_{SSH}	Slave Select Hold Time	150		ns	After last SCK edge

Notes:

4. Full functionality is guaranteed in V_{CC1} range, except USB transmitter specifications and GPIO output currents are guaranteed for V_{CC2} range.
5. Total current cumulative across all Port pins, limited to minimize Power and Ground-Drop noise effects.
6. LVR is automatically disabled during suspend mode.
7. LVR will re-occur whenever V_{CC} drops below V_{LVR}. In suspend or with LVR disabled, BOR occurs whenever V_{CC} drops below approximately 2.5V.
8. V_{RG} specified for regulator enabled, idle conditions (i.e. no USB traffic), with load resistors listed. During USB transmits from the internal SIE, the VREG output is not regulated, and should not be used as a general source of regulated voltage in that case. During receive of USB data, the VREG output drops when D- is low due to internal series resistance of approximately 200Ω at the VREG pin.
9. In suspend mode, V_{RG} is only valid if R_{PU} is connected from D– to VREG pin, and R_{PD} is connected from D– to ground.
10. The 200Ω internal resistance at the VREG pin gives a standard USB pull-up using this value. Alternately, a 1.5 kΩ,5%pull-up from D- to an external 3.3V supply can be used.
11. Initially F_{ICLK2}=F_{ICLK} until a USB packet is received.
12. Wake-up time for Wake-up Adjust Bits cleared to 000b (minimum setting)
13. Tested at 200 pF.
14. Measured at cross-over point of differential data signals.
15. 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.
16. SPI timing specified for capacitive load of 50 pF, with GPIO output mode = 01 (medium low drive, strong high drive).

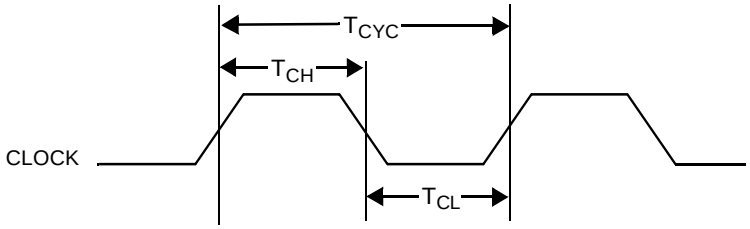


Figure 25-1. Clock Timing

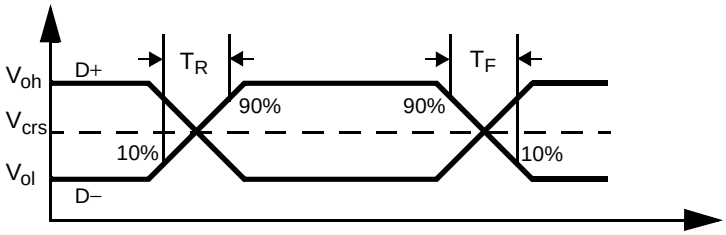


Figure 25-2. USB Data Signal Timing

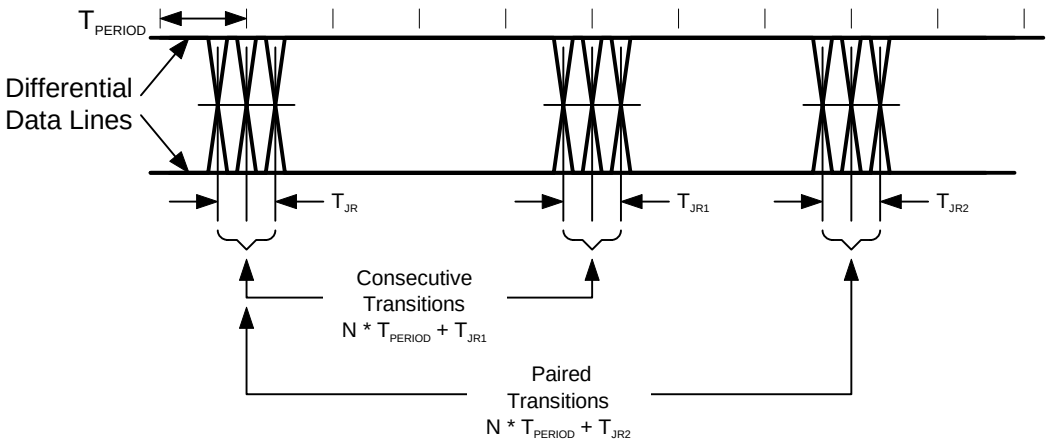


Figure 25-3. Receiver Jitter Tolerance

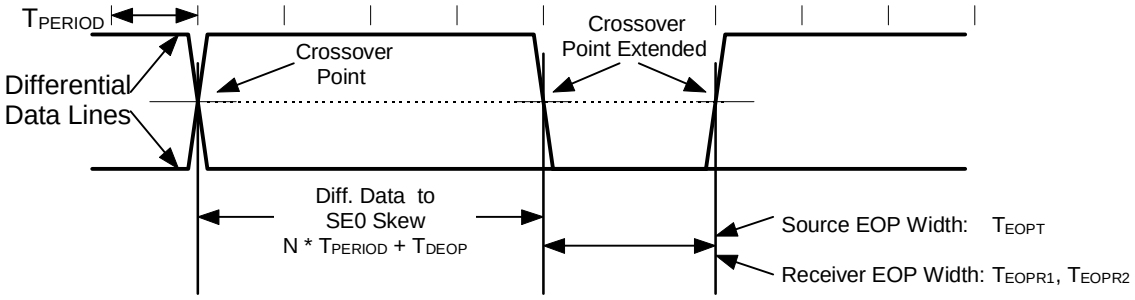


Figure 25-4. Differential to EOP Transition Skew and EOP Width

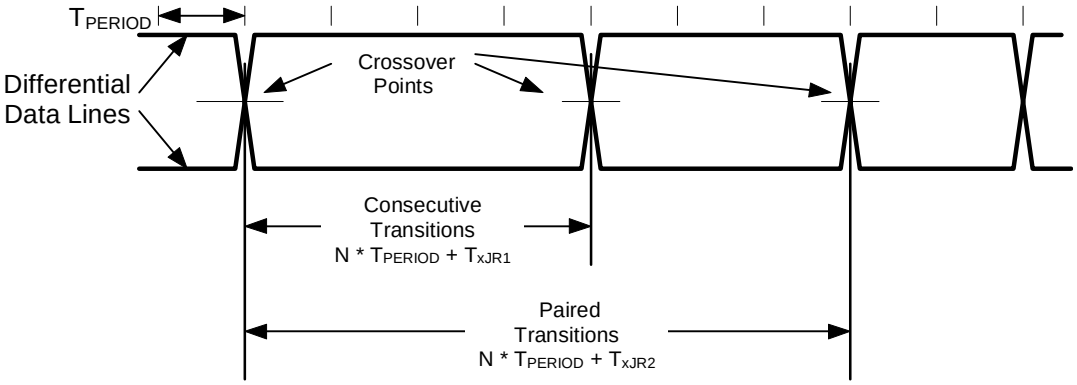


Figure 25-5. Differential Data Jitter

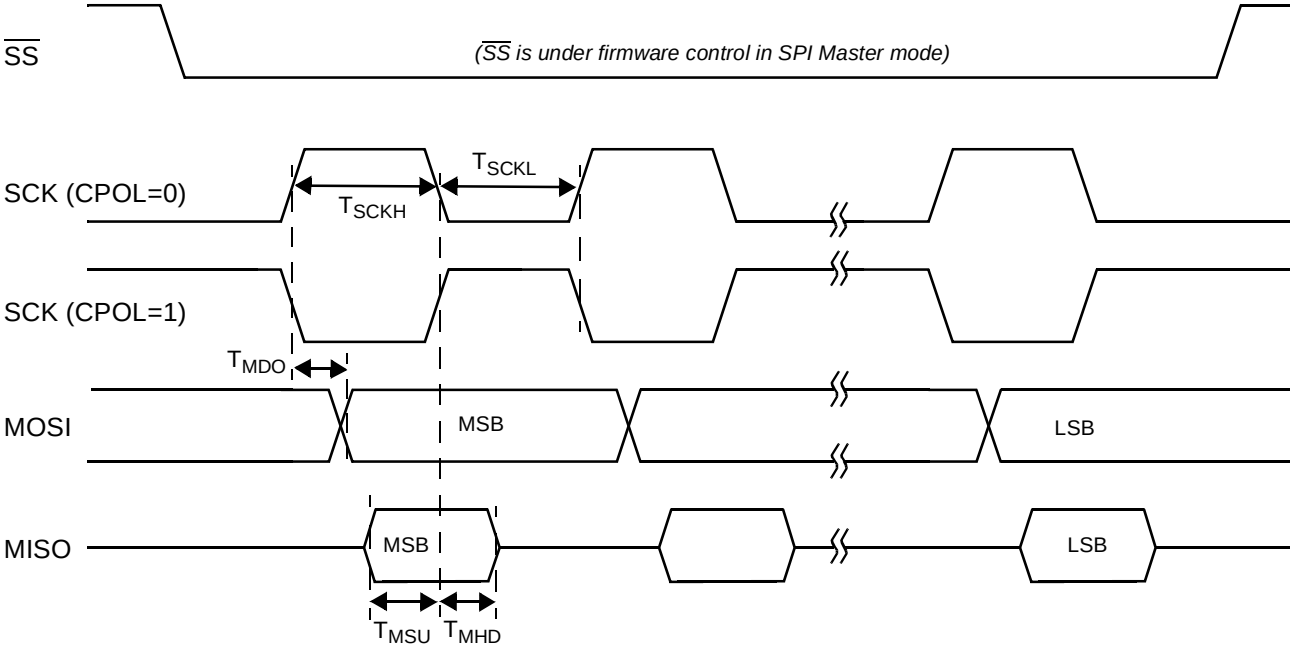


Figure 25-6. SPI Master Timing, CPHA=0

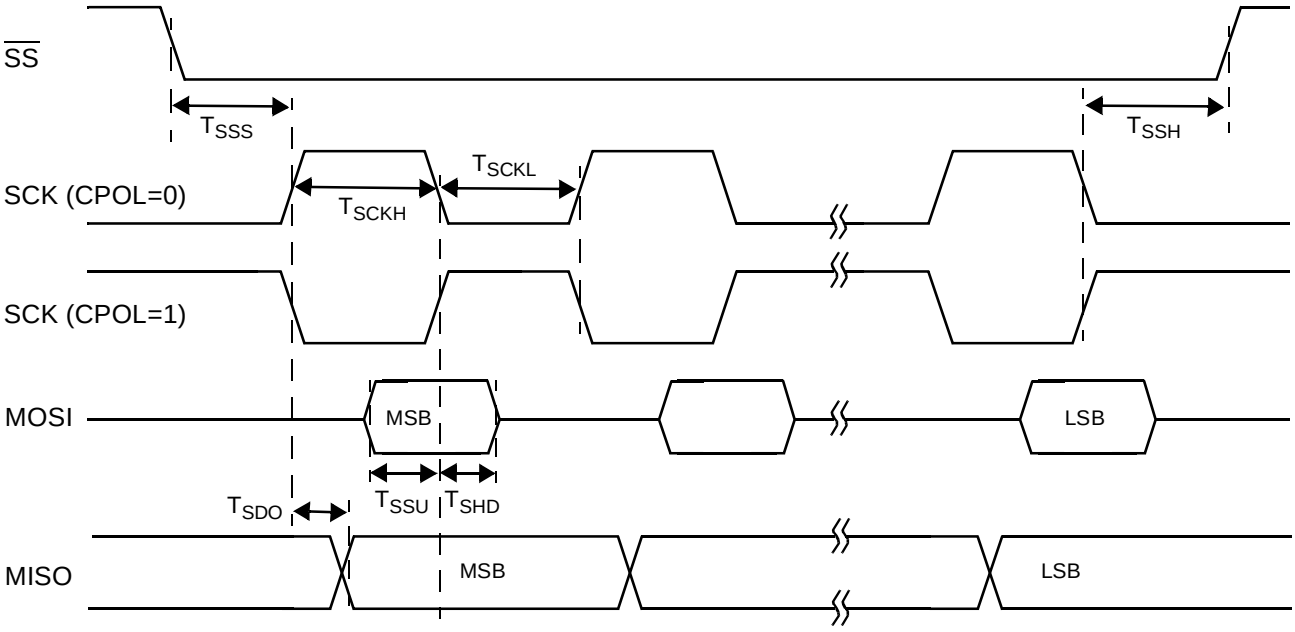


Figure 25-7. SPI Slave Timing, CPHA=0

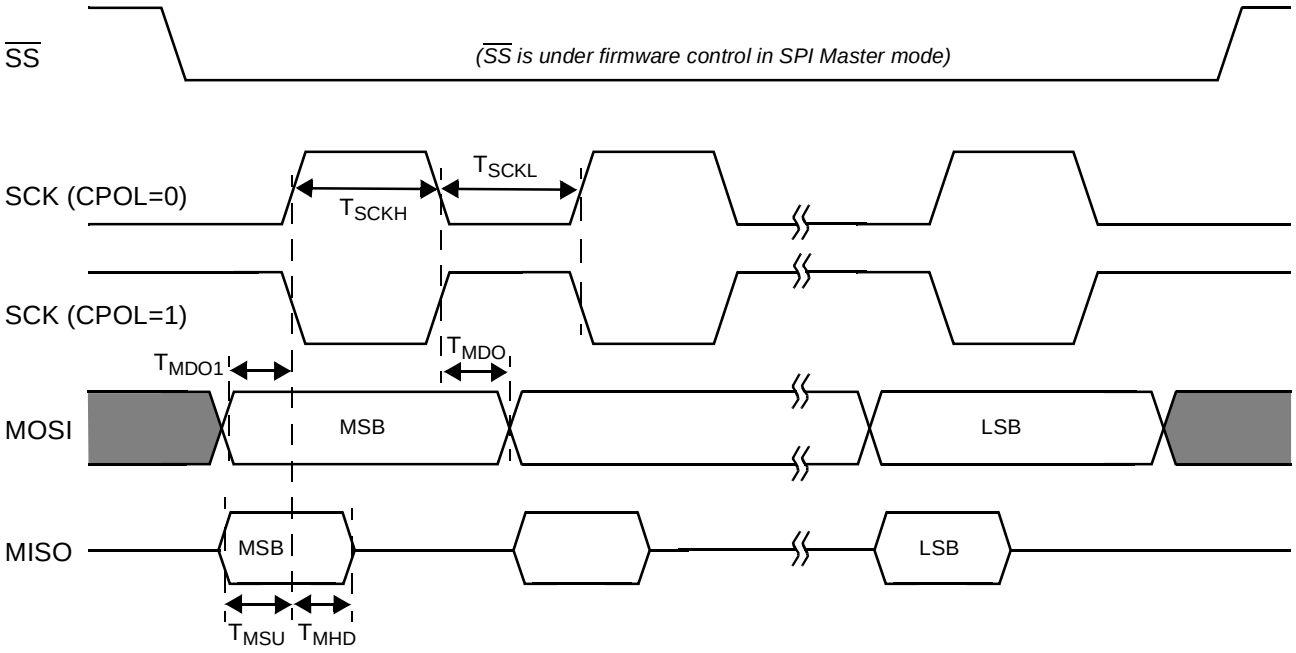


Figure 25-8. SPI Master Timing, CPHA=1

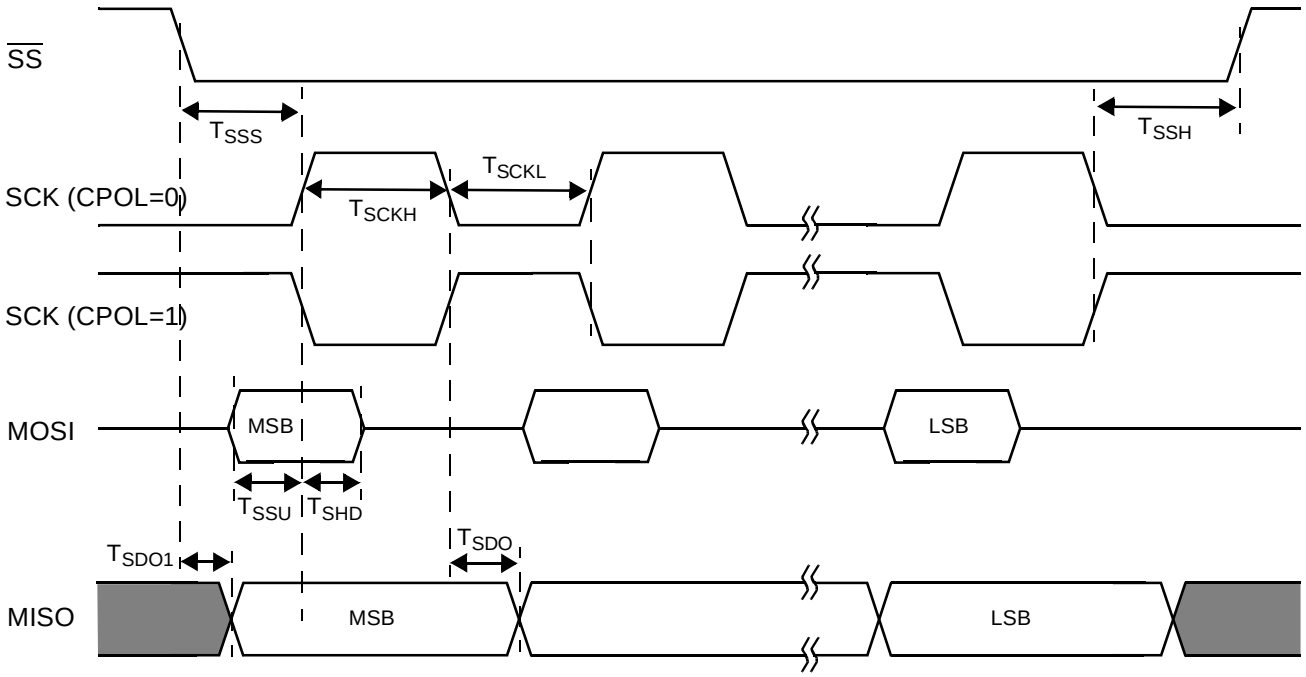


Figure 25-9. SPI Slave Timing, CPHA=1

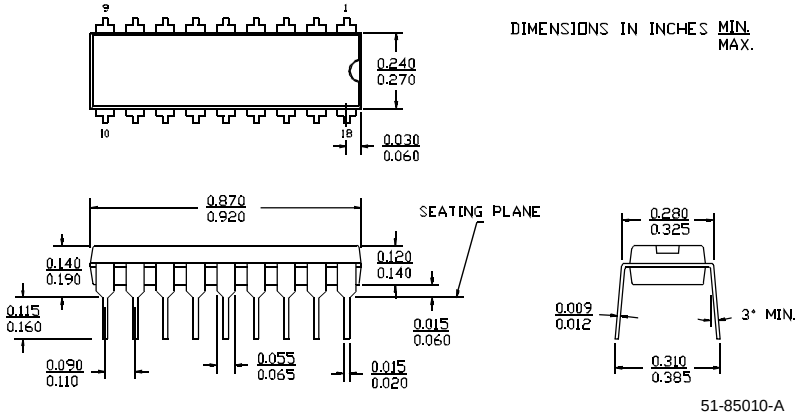
26.0 Ordering Information

Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C63722-PC	6 KB	P3	18-Pin (300-Mil) PDIP	Commercial
CY7C63723-PC	8 KB	P3	18-Pin (300-Mil) PDIP	Commercial
CY7C63742-PC	6 KB	P13	24-Pin (300-Mil) PDIP	Commercial
CY7C63743-PC	8 KB	P13	24-Pin (300-Mil) PDIP	Commercial
CY7C63742-SC	6 KB	S13	24-Pin Small Outline Package	Commercial
CY7C63743-SC	8 KB	S13	24-Pin Small Outline Package	Commercial

Document #: 38-00944

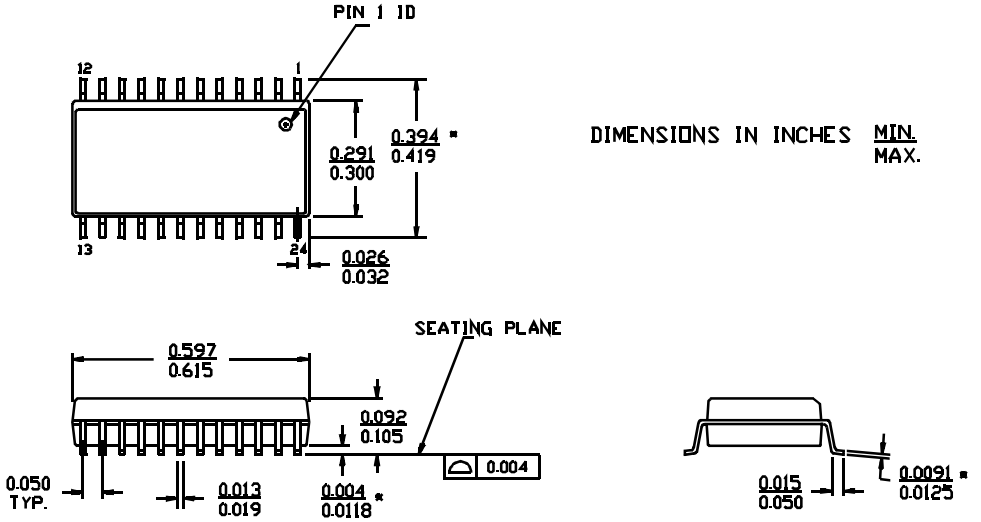
27.0 Package Diagrams

18-Lead (300-Mil) Molded DIP P3



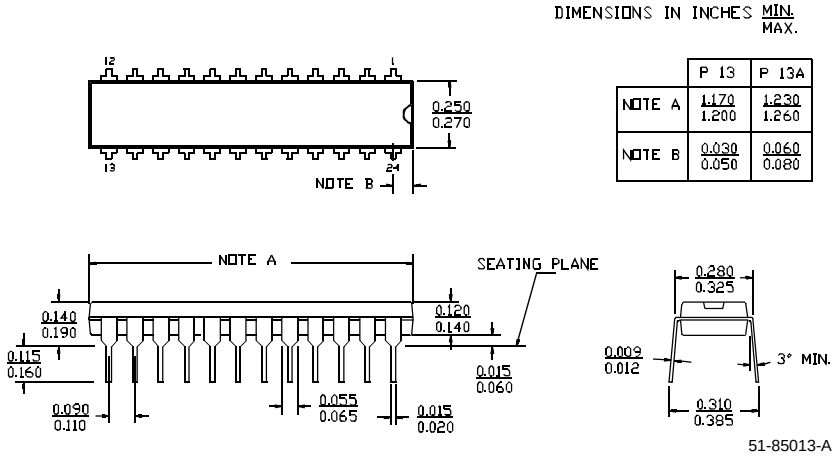
51-85010-A

24-Lead (300-Mil) Molded SOIC S13



51-85025-A

24-Lead (300-Mil) Molded DIP P13/P13A



	P 13	P 13A
NOTE A	1.170 1.200	1.230 1.260
NOTE B	0.030 0.050	0.060 0.080

51-85013-A