Welcome to E-XFL.COM

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

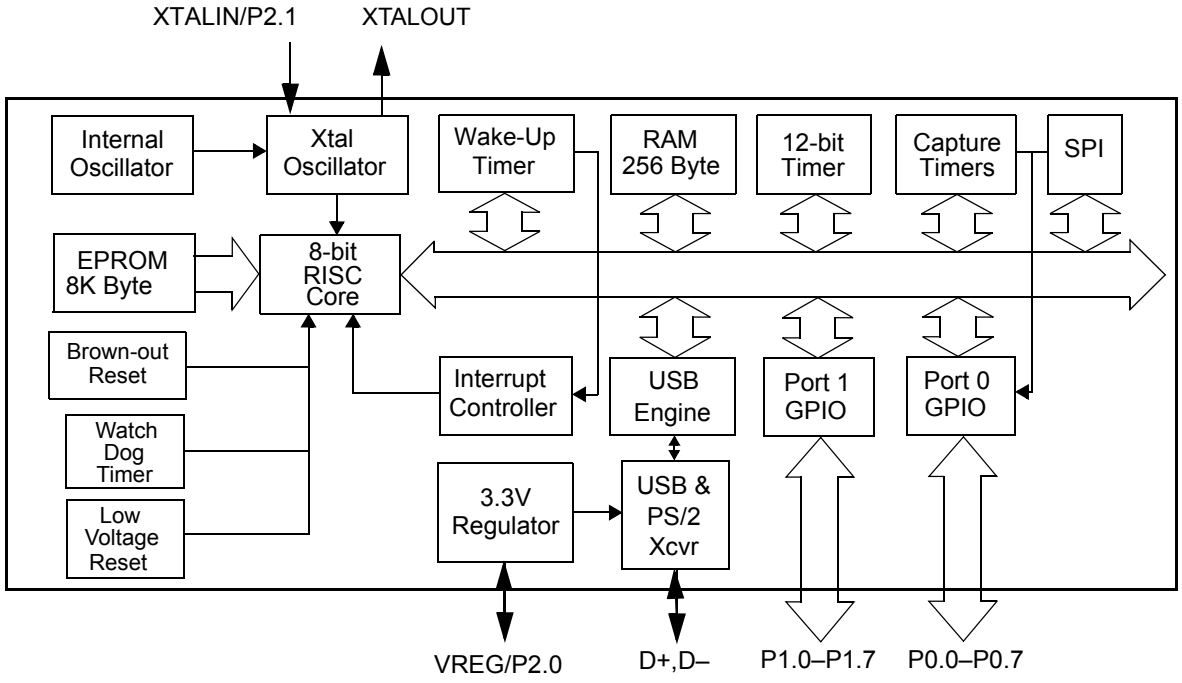
Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering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

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	-
Core Processor	-
Program Memory Type	-
Controller Series	-
RAM Size	-
Interface	-
Number of I/O	-
Voltage - Supply	-
Operating Temperature	-
Mounting Type	-
Package / Case	-
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c63722c-xc

Logic Block Diagram



Functional Overview

enCoRe USB—The New USB Standard

Cypress has reinvented its leadership position in the low-speed USB market with a new family of innovative microcontrollers. Introducing...enCoRe USB—“enhanced Component Reduction.” Cypress has leveraged its design expertise in USB solutions to create a new family of low-speed USB microcontrollers that enables peripheral developers to design new products with a minimum number of components. At the heart of the enCoRe USB technology is the breakthrough design of a crystalless oscillator. By integrating the oscillator into our chip, an external crystal or resonator is no longer needed. We have also integrated other external components commonly found in low-speed USB applications such as pull-up resistors, wake-up circuitry, and a 3.3 V regulator. All of this adds up to a lower system cost.

The CY7C637xxC is an 8-bit RISC one-time-programmable (OTP) microcontroller. The instruction set has been optimized specifically for USB and PS/2 operations, although the microcontrollers can be used for a variety of other embedded applications.

The CY7C637xxC features up to 16 GPIO pins to support USB, PS/2 and other applications. The I/O pins are grouped into two ports (Port 0 to 1) where each pin can be individually configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs with programmable drive strength of up to 50 mA output drive. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. Note the GPIO interrupts all share the same “GPIO” interrupt vector.

The CY7C637xxC microcontrollers feature an internal oscillator. With the presence of USB traffic, the internal oscillator can be set to precisely tune to USB timing requirements (6 MHz ±1.5%). Optionally, an external 6-MHz ceramic resonator can be used to provide a higher precision reference for USB operation. This clock generator reduces the clock-related noise emissions (EMI). The clock generator provides the 6- and 12-MHz clocks that remain internal to the microcontroller.

The CY7C637xxC has 8 Kbytes of EPROM and 256 bytes of data RAM for stack space, user variables, and USB FIFOs.

These parts include low-voltage reset logic, a Watchdog timer, a vectored interrupt controller, a 12-bit free-running timer, and capture timers. The low-voltage reset (LVR) logic detects when power is applied to the device, resets the logic to a known state, and begins executing instructions at EPROM address 0x0000. LVR will also reset the part when V_{CC} drops below the operating voltage range. The Watchdog timer can be used to ensure the firmware never gets stalled for more than approximately 8 ms.

The microcontroller supports 10 maskable interrupts in the vectored interrupt controller. Interrupt sources include the USB Bus-Reset, the 128-µs and 1.024-ms outputs from the free-running timer, three USB endpoints, two capture timers, an internal wake-up timer and the GPIO ports. The timers bits cause periodic interrupts when enabled. The USB endpoints interrupt after USB transactions complete on the bus. The capture timers interrupt whenever a new timer value is saved due to a selected GPIO edge event. The GPIO ports have a level of masking to select which GPIO inputs can cause a GPIO interrupt. For additional flexibility, the input transition polarity that causes an interrupt is programmable for each GPIO pin. The interrupt polarity can be either rising or falling edge [1].

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

The CY7C637xxC includes an integrated USB serial interface engine (SIE) that supports the integrated peripherals. The hardware supports one USB device address with three endpoints. The SIE allows the USB host to communicate with the function integrated into the microcontroller. A 3.3V regulated output pin provides a pull-up source for the external USB resistor on the D– pin.

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

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

Programming Model

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

Program Counter (PC)

The 14-bit program counter (PC) allows access for up to 8 Kbytes of EPROM using the CY7C637xxC architecture. The program counter is cleared during reset, such that the first instruction executed after a reset is at address 0x0000. This instruction is typically a jump instruction to a reset handler that initializes the application.

The lower 8 bits of the program counter are incremented as instructions are loaded and executed. The upper six bits of the program counter are incremented by executing an XPAGE instruction. As a result, the last instruction executed within a 256-byte "page" of sequential code should be an XPAGE instruction. The assembler directive "XPAGEON" will cause the assembler to insert XPAGE instructions automatically. As instructions can be either one or two bytes long, the assembler may occasionally need to insert a NOP followed by an XPAGE for correct execution.

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

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

8-bit Accumulator (A)

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

8-bit Index Register (X)

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

8-bit Program Stack Pointer (PSP)

During a reset, the program stack pointer (PSP) is set to zero. This means the program "stack" starts at RAM address 0x00 and "grows" upward from there. Note that the program stack pointer is directly addressable under firmware control, using the MOV PSP,A instruction. The PSP supports interrupt service under hardware control and CALL, RET, and RETI instructions under firmware control.

During an interrupt acknowledge, interrupts are disabled and the program counter, carry flag, and zero flag are written as two bytes of data memory. The first byte is stored in the memory addressed by the program stack pointer, then the PSP is incremented. The second byte is stored in memory addressed by the program stack pointer and the PSP is incremented again. The net effect is to store the program counter and flags on the program "stack" and increment the program stack pointer by two.

The return from interrupt (RETI) instruction decrements the program stack pointer, then restores the second byte from memory addressed by the PSP. The program stack pointer is decremented again and the first byte is restored from memory addressed by the PSP. After the program counter and flags have been restored from stack, the interrupts are enabled. The effect is to restore the program counter and flags from the program stack, decrement the program stack pointer by two, and re-enable interrupts.

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

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

Note that there are restrictions in using the JMP, CALL, and INDEX instructions across the 4-KByte boundary of the program memory. Refer to the *CYASM Assembler User's Guide* for a detailed description.

8-bit Data Stack Pointer (DSP)

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

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

For USB applications, the firmware should set the DSP to an appropriate location to avoid a memory conflict with RAM dedicated to USB FIFOs. The memory requirements for the USB endpoints are shown in Data Memory Organization on page 10 . For example, assembly instructions to set the DSP to 20h (giving 32 bytes for program and data stack combined) are shown below.

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

Address Modes

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

Data

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

- MOV A, 30h

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

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

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 3. No additional capacitance is included on chip at the XTALIN/OUT pins. Operation is controlled by the Clock Configuration Register, Figure 4.

Figure 3. Clock Oscillator On-chip Circuit

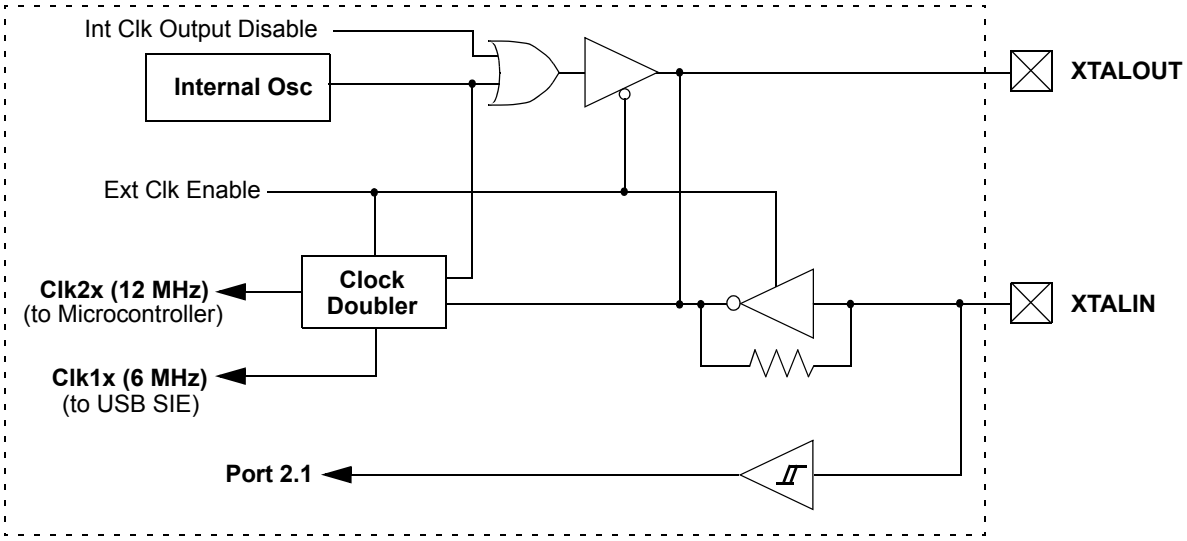


Figure 4. Clock Configuration Register (Address 0xF8)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Ext. Clock Resume Delay	Wake-up Timer Adjust Bit [2:0]			Low-voltage Reset Disable	Precision USB Clocking Enable	Internal Clock Output Disable	External Oscillator Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7: Ext. Clock Resume Delay

External Clock Resume Delay bit selects the delay time when switching to the external oscillator from the internal oscillator mode, or when waking from suspend mode with the external oscillator enabled.

1 = 4 ms delay.

0 = 128 μs delay.

The delay gives the oscillator time to start up. 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). Note that this bit only selects the delay time for the external clock mode. When waking from suspend mode with the internal oscillator (Bit 0 is LOW), the delay time is only 8 μs in addition to a delay of approximately 1 μs for the oscillator to start.

Bit [6:4]: Wake-up Timer Adjust Bit [2:0]

The Wake-up Timer Adjust Bits are used to adjust the Wake-up timer period.

If the Wake-up interrupt is enabled in the Global Interrupt Enable Register, the microcontroller will generate wake-up interrupts periodically. The frequency of these periodical wake-up interrupts is adjusted by setting the Wake-up Timer Adjust Bit [2:0], as described in Wake-up Timer on page 16. One common use of the wake-up interrupts is to generate periodical wake-up events during suspend mode to check for changes, such as looking for movement in a mouse, while maintaining a low average power.

Bit 3: Low-voltage Reset Disable

When V_{CC} drops below V_{LVR} (see DC Characteristics on page 42 for the value of V_{LVR}) and the Low-voltage Reset circuit is enabled, the microcontroller enters a partial suspend state for a period of t_{START} (see Switching Characteristics on

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 USB Port Status and Control on page 19.
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 4 on page 12) 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 8-bit Data Stack Pointer (DSP) on page 6.
8. Program execution begins at address 0x0000 after the appropriate time-out period.

Low-voltage Reset (LVR)

When V_{CC} is first applied to the chip, the internal oscillator is started and the Low-voltage Reset is initially enabled by default. At the point where V_{CC} has risen above V_{LVR} (see DC Characteristics on page 42 for the value of V_{LVR}), an internal counter starts counting for a period of t_{START} (see Switching Characteristics on page 44 for the value of t_{START}). During this t_{START} time, the microcontroller enters a partial suspend state to wait for V_{CC} to stabilize before it begins executing code from address 0x0000.

As long as the LVR circuit 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 (Figure 4 on page 12). In addition, the LVR is automatically disabled in suspend mode to save power. If the LVR was enabled before entering suspend mode, it becomes active again once the suspend mode ends.

When LVR is disabled during normal operation (i.e., by writing ‘0’ to the Low-voltage Reset Disable bit in the Clock Configuration Register), the chip may enter an unknown state if V_{CC} drops below V_{LVR}. Therefore, LVR should be enabled at all times during normal operation. If LVR is disabled (i.e., by firmware or during suspend mode), a secondary low-voltage monitor, BOR, becomes active, as described in the next section. The LVR/BOR Reset bit of the Processor Status and Control Register (Figure 34 on page 30), is set to ‘1’ if either a LVR or BOR has occurred.

Brown Out Reset (BOR)

The Brown Out Reset (BOR) circuit is always active and behaves like the POR. BOR is asserted whenever the V_{CC} voltage to the device is below an internally defined trip voltage of approximately 2.5 V. The BOR re-enables 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 resumes, and the microcontroller starts executing code from address 0x00 after the t_{START} delay.

In suspend mode, only the BOR detection is active, giving a reset if V_{CC} drops below approximately 2.5 V. Since the device is suspended and code is not executing, this lower reset voltage is safe for retaining the state of all registers and memory. Note that in suspend mode, LVR is disabled as discussed in Low-voltage Reset (LVR) .

Watchdog Reset (WDR)

The Watchdog Timer Reset (WDR) occurs when the internal Watchdog timer rolls over. Writing any value to the write-only Watchdog Reset 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} (see Figure 5) of the last clear. Bit 6 (Watchdog Reset bit) of the Processor Status and Control Register is set to record this event (see Processor Status and Control Register on page 30 for more details). A Watchdog Timer Reset typically lasts for 2–4 ms, after which the microcontroller begins execution at ROM address 0x0000.

Figure 15. USB Device Address Register (Address 0x10)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Device Address Enable	Device Address						
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

In either USB or PS/2 mode, this register is cleared by both hardware resets and the USB bus reset. See Interrupt Sources on page 32 for more information on the USB Bus Reset – PS/2 interrupt.

Bit 7: Device Address Enable

This bit must be enabled by firmware before the serial interface engine (SIE) will respond to USB traffic at the address specified in Bit [6:0].

1 = Enable USB device address.

0 = Disable USB device address.

Bit [6:0]: Device Address Bit [6:0]

These bits must be set by firmware during the USB enumeration process (i.e., SetAddress) to the non-zero address assigned by the USB host.

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 Data Memory Organization on page 10.

The EP0 endpoint mode register uses the format shown in Figure .

Figure 16. Endpoint 0 Mode Register (Address 0x12)

Bit #	7	6	5	4	3:0
Bit Name	SETUP Received	IN Received	OUT Received	ACKed Transaction	Mode Bit
Read/ Write	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0 0 0 0

The SIE provides a locking feature to prevent firmware from overwriting bits in the USB Endpoint 0 Mode Register. Writes to the register have no effect from the point that Bit[6:0] of the register are updated (by the SIE) until the firmware reads this register. The CPU can unlock this register by reading it.

Because of these hardware-locking features, firmware should perform an read after a write to the USB Endpoint 0 Mode Register and USB Endpoint 0 Count Register (Figure 18) to verify that the contents have changed as desired, and that the SIE has not updated these values.

Bit [7:4] of this register are cleared by any non-locked write to this register, regardless of the value written.

Bit 7: SETUP Received

1 = A valid SETUP packet has been received. This bit 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. While this bit is set to ‘1’, 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.

0 = No SETUP received. This bit is cleared by any non-locked writes to the register.

Bit 6: IN Received

1 = A valid IN packet has been received. This bit is updated to ‘1’ after the last received packet in an IN transaction. This bit is cleared by any non-locked writes to the register.

0 = No IN received. This bit is cleared by any non-locked writes to the register.

Bit 5: OUT Received

1 = A valid OUT packet has been received. This bit is updated to ‘1’ after the last received packet in an OUT transaction. This bit is cleared by any non-locked writes to the register.

0 = No OUT received. This bit is cleared by any non-locked writes to the register.

Bit 4: ACKed Transaction

The ACKed Transaction bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet.

1 = The transaction completes with an ACK.

0 = The transaction does not complete with an ACK.

Bit [3:0]: Mode Bit[3:0]

The endpoint modes determine how the SIE responds to USB traffic that the host sends to the endpoint. For example, if the endpoint Mode Bits [3:0] are set to 0001 which is NAK IN/OUT mode as shown in *Table 8*, the SIE will send NAK handshakes in response to any IN or OUT token sent to this endpoint. In this NAK IN/OUT mode, the SIE will send an ACK handshake when the host sends a SETUP token to this endpoint. The mode encoding is shown in *Table 8*. Additional information on the mode bits can be found in *Table 9* and *Table 10*. These modes give the firmware total control on how to respond to different tokens sent to the endpoints from the host.

In addition, the Mode Bits are automatically changed by the SIE in response to many USB transactions. For example, if the Mode Bit [3:0] are set to 1011 which is ACK OUT-NAK IN mode as shown in *Table 8*, the SIE will change the endpoint Mode Bit [3:0] to NAK IN/OUT (0001) mode after issuing an ACK handshake in response to an OUT token. Firmware needs to update the mode for the SIE to respond appropriately.

USB Non-control Endpoints

The CY7C637xxC feature two non-control endpoints, endpoint 1 (EP1) and endpoint 2 (EP2). The EP1 and EP2 Mode Registers do not have the locking mechanism of the EP0 Mode Register. The EP1 and EP2 Mode Registers use the format shown in Figure 17. EP1 uses an 8-byte FIFO at SRAM locations 0xF0–0xF7, EP2 uses an 8-byte FIFO at SRAM locations 0xE8–0xEF as shown in Data Memory Organization on page 10.

Figure 17. USB Endpoint EP1, EP2 Mode Registers (Addresses 0x14 and 0x16)

Bit #	7	6	5	4	3	2	1	0
Bit Name	STALL	Reserved		ACKed Transaction	Mode Bit			
Read/ Write	R/W	-	-	R/C	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7: STALL

1 = The SIE will stall an OUT packet if the Mode Bits are set to ACK-OUT, and the SIE will stall an IN packet if the mode bits are set to ACK-IN. See USB Mode Tables on page 36 for the available modes.

0 = This bit must be set to LOW for all other modes.

Bit [6:5]: Reserved. Must be written to zero during register writes.

Bit 4: ACKed Transaction

The ACKed transaction bit is set whenever the SIE engages in a transaction to the register's endpoint that completes with an ACK packet.

1 = The transaction completes with an ACK.

0 = The transaction does not complete with an ACK.

Bit [3:0]: Mode Bit [3:0]

The EP1 and EP2 Mode Bits operate in the same manner as the EP0 Mode Bits (see USB Mode Tables on page 36).

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 18.

Figure 18. Endpoint 0,1,2 Counter Registers (Addresses 0x11, 0x13 and 0x15)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Data Toggle	Data Valid	Reserved		Byte Count			
Read/Write	R/W	R/W	-	-	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7: Data Toggle

This bit selects the DATA packet's toggle state. For IN transactions, firmware must set this bit to the select the transmitted Data Toggle. For OUT or SETUP transactions, the hardware sets this bit to the state of the received Data Toggle bit.

1 = DATA1

0 = DATA0

Bit 6: Data Valid

This bit is used for OUT and SETUP tokens only. This bit is cleared to '0' if CRC, bitstuff, or PID errors have occurred. This bit does not update for some endpoint mode settings. Refer to *Table 10* for more details.

1 = Data is valid.

0 = Data is invalid. If enabled, the endpoint interrupt will occur even if invalid data is received.

Bit [5:4]: Reserved

Bit [3:0]: Byte Count Bit [3:0]

Byte Count Bits 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.

For Endpoint 0 Count Register, 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.

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 V_{REG} voltage, 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 DC Characteristics on page 42).

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 14*). 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, the V_{REG} pin can be used as an input and its state can be read at port P2.0. Refer to Figure 13 on page 19 for the Port 2 data register. This input has a TTL threshold.

In suspend mode, the regulator is automatically disabled. If VREG Enable bit is set (*Figure 14*), the VREG pin is pulled up to V_{CC} with an internal 6.2-kΩ resistor. This holds the proper V_{OH} state in suspend mode

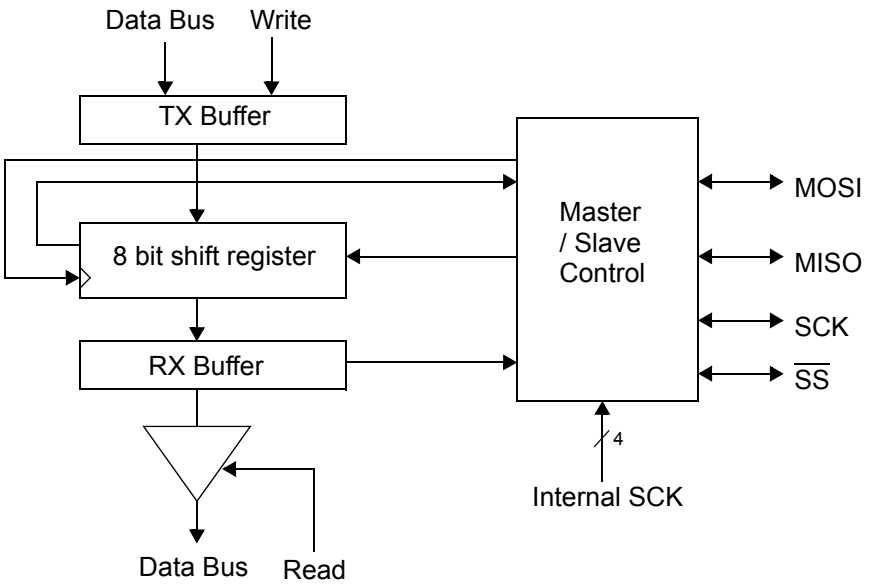
Note that enabling the device for USB (by setting the Device Address Enable bit, Figure 15) activates the internal regulator, even if the VREG Enable bit is cleared to 0. This insures proper

Serial Peripheral Interface (SPI)

SPI is a four-wire, full-duplex serial communication interface between a master device and one or more slave devices. The CY7C637xxC SPI circuit supports byte serial transfers in either Master or Slave modes. The block diagram of the SPI circuit is shown in Figure 20. The block contains buffers for both transmit and receive data for maximum flexibility and throughput. The CY7C637xxC can be configured as either an SPI Master or Slave. The external interface consists of Master-Out/Slave-In (MOSI), Master-In/Slave-Out (MISO), Serial Clock (SCK), and Slave Select ($\overline{SS}$).

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

Figure 20. SPI Block Diagram



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

Figure 21. SPI Data Register (Address 0x60)

Bit #	7	6	5	4	3	2	1	0
Bit Name	Data I/O							
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit [7:0]: Data I/O[7:0]

Writes to the SPI Data Register load the transmit buffer, while reads from this register read the receive buffer contents.

1 = Logic HIGH

0 = Logic LOW

Operation as an SPI Master

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

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

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

The byte shifting and SCK generation are handled by the hardware (based on firmware selection of the clock source). Data is shifted out on the MOSI pin (P0.5) and the serial clock is output on the SCK pin (P0.7). Data is received from the slave on the MISO pin (P0.6). The output pins must be set to the desired drive strength, and the GPIO data register must be set to 1 to enable a bypass mode for these pins. The MISO pin must be configured in the desired GPIO input mode. See General Purpose I/O Ports on page 17 for GPIO configuration details.

Figure 37. Interrupt Controller Logic Block Diagram

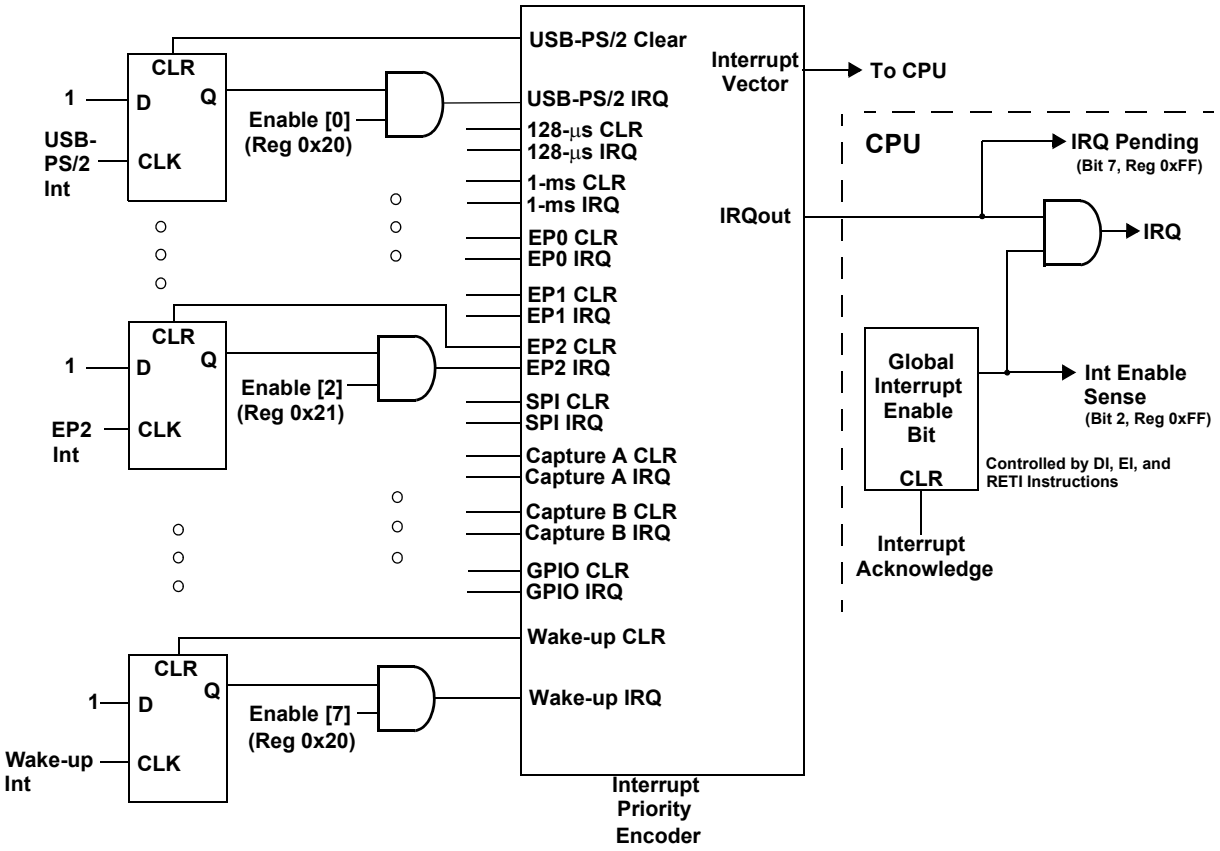


Figure 38. Port 0 Interrupt Enable Register (Address 0x04)

Bit #	7	6	5	4	3	2	1	0
Bit Name	P0 Interrupt Enable							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit [7:0]: P0 [7:0] Interrupt Enable

1 = Enables GPIO interrupts from the corresponding input pin.

0 = Disables GPIO interrupts from the corresponding input pin.

Figure 39. Port 1 Interrupt Enable Register (Address 0x05)

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

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

1 = Enables GPIO interrupts from the corresponding input pin.

0 = Disables GPIO interrupts from the corresponding input pin.

The polarity that triggers an interrupt is controlled independently for each GPIO pin by the GPIO Interrupt Polarity Registers. Figure and Figure 40 control the interrupt polarity of each GPIO pin.

Port 0 Interrupt Polarity Register (Address 0x06)

Bit #	7	6	5	4	3	2	1	0
Bit Name	P0 Interrupt Polarity							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit [7:0]: P0[7:0] Interrupt Polarity

1 = Rising GPIO edge

0 = Falling GPIO edge

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

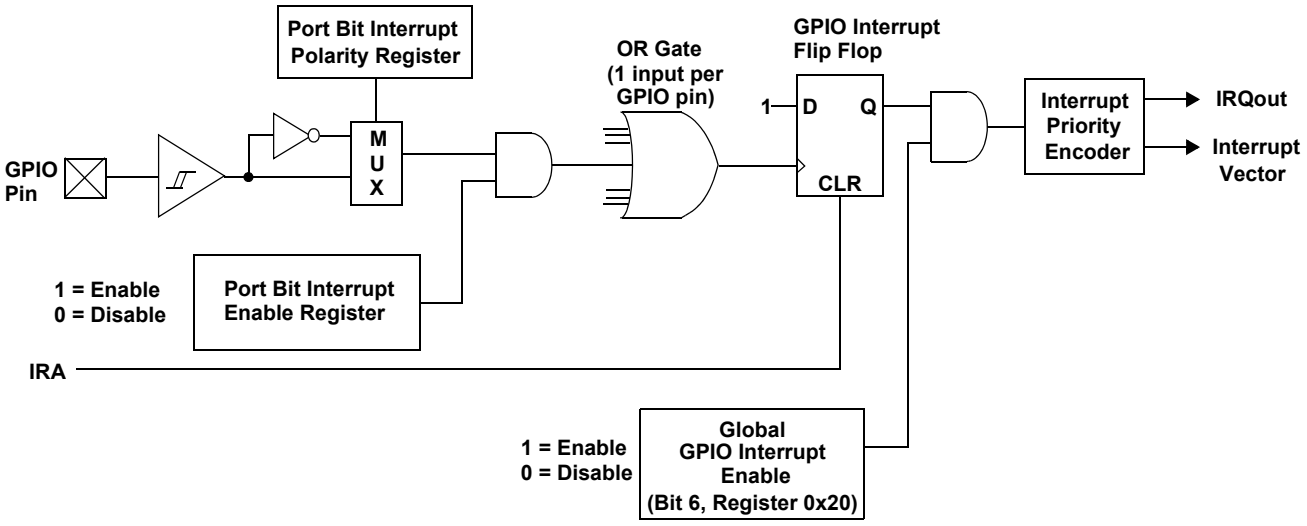
Bit #	7	6	5	4	3	2	1	0
Bit Name	P1 Interrupt Polarity							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

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

1 = Rising GPIO edge

0 = Falling GPIO edge

Figure 41. GPIO Interrupt Diagram



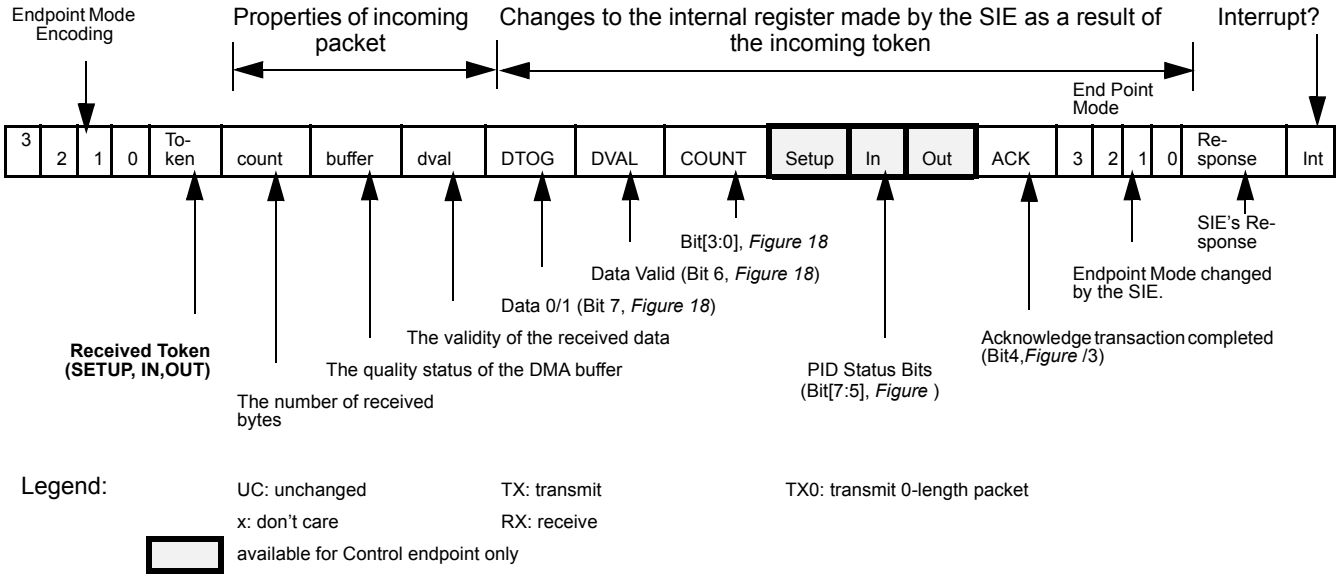
Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode as shown in Table 8, the SIE will change the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACKing a valid status stage OUT token. The firmware needs to update the mode for the SIE to respond appropriately. See Table 8 for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

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

A disabled endpoint will remain disabled until changed by firmware, and all endpoints reset to the Disabled mode (0000). Firmware normally enables the endpoint mode after a SetConfiguration request.

The control endpoint has three status bits for identifying the token type received (SETUP, IN, or OUT), but the endpoint must be placed in the correct mode to function as such. Non-control endpoints should not be placed into modes that accept SETUPs.

Table 9. Decode table for Table 10: "Details of Modes for Differing Traffic Conditions"



The response of the SIE can be summarized as follows:

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

Table 10. Details of Modes for Differing Traffic Conditions

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

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

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

Switching Characteristics (continued)

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

Figure 42. Clock Timing

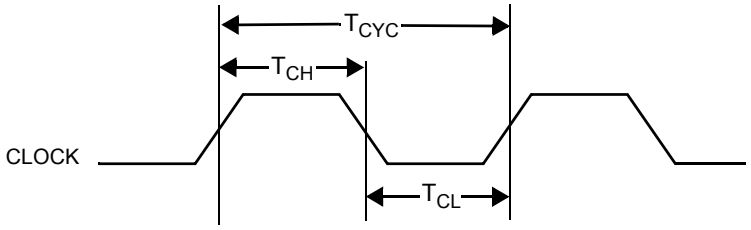


Figure 43. USB Data Signal Timing

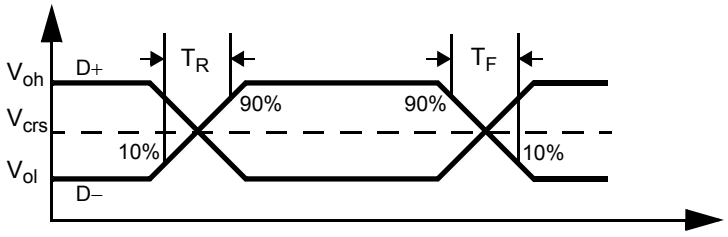


Figure 44. Receiver Jitter Tolerance

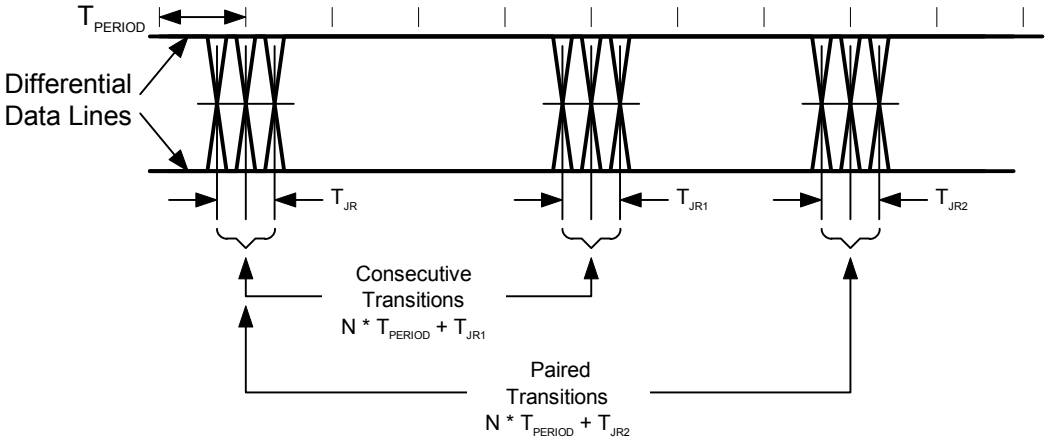


Figure 45. Differential to EOP Transition Skew and EOP Width

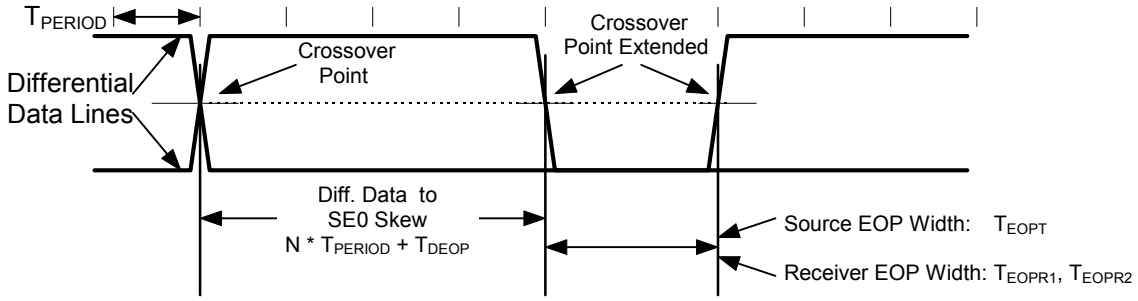


Figure 46. Differential Data Jitter

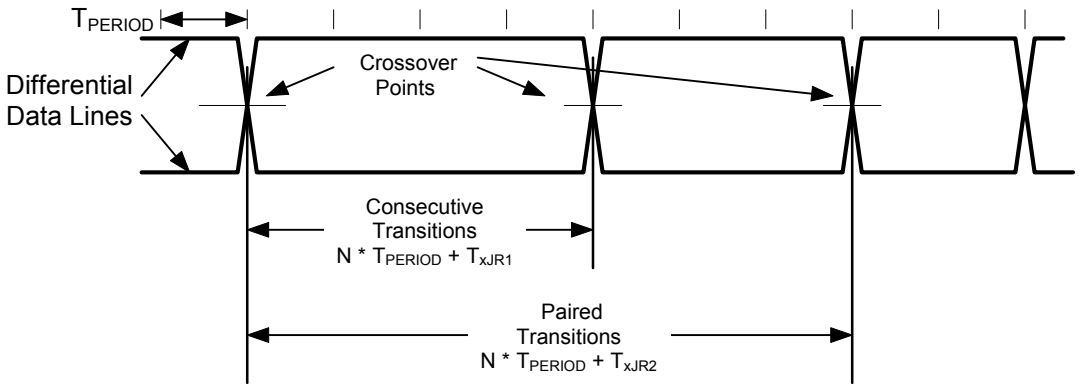


Figure 49. SPI Master Timing, CPHA = 1

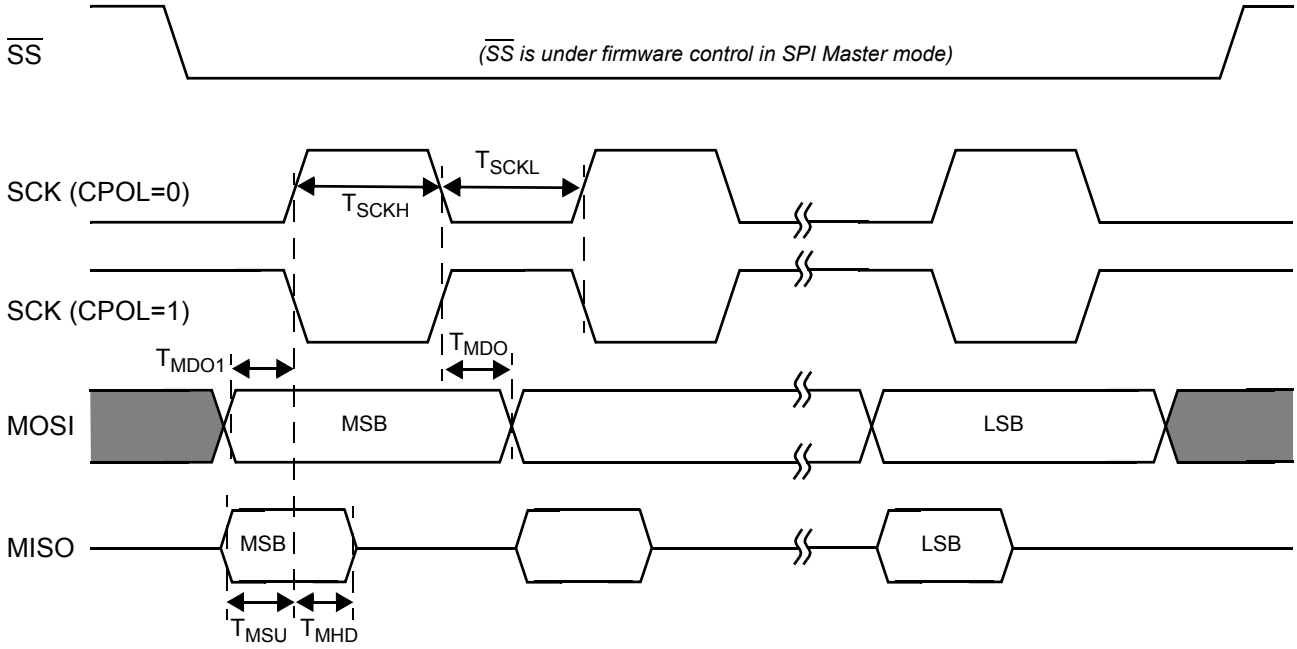


Figure 50. SPI Slave Timing, CPHA = 1

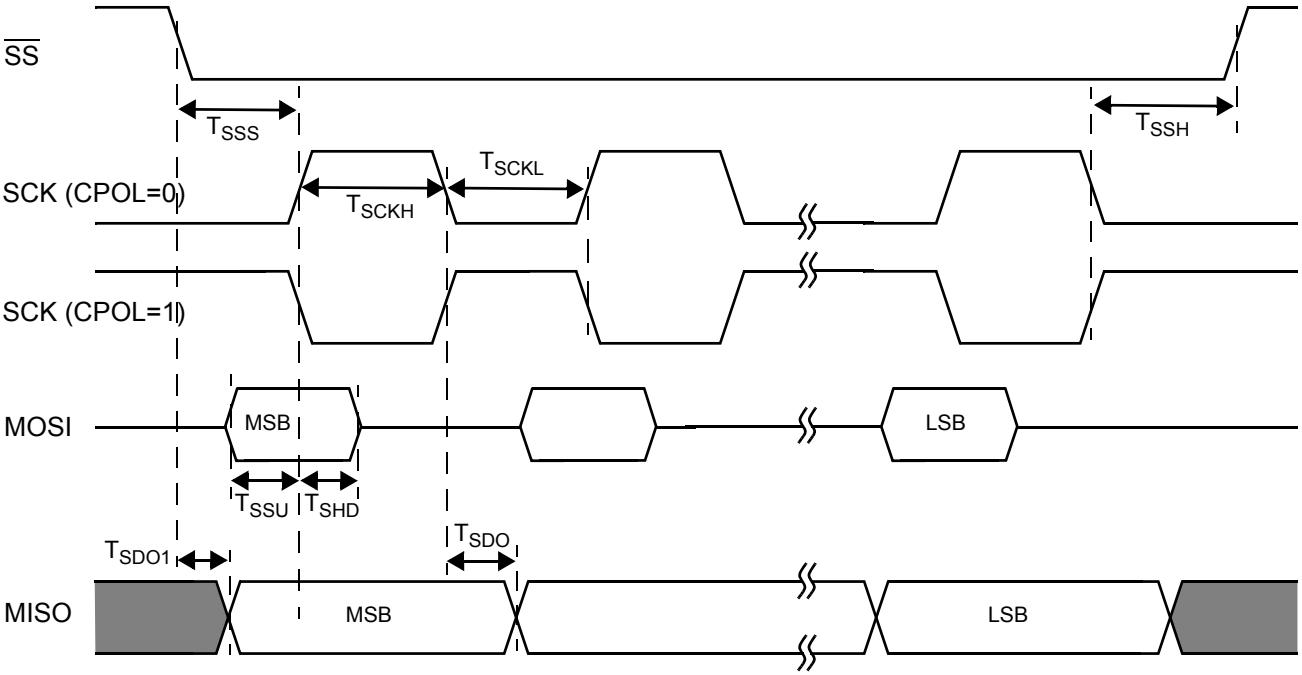


Figure 52. 18-pin SOIC (0.463 × 0.300 × 0.0932 Inches) Package Outline, 51-85023

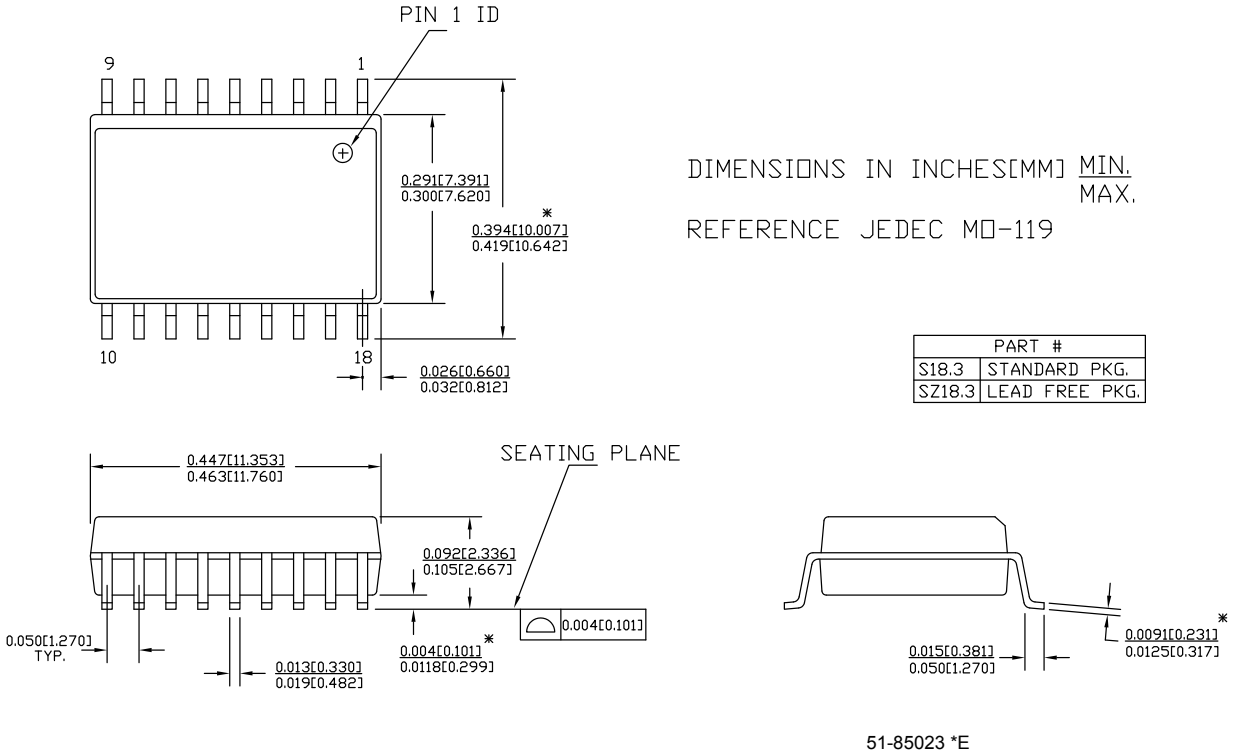


Figure 53. 24-pin SOIC (0.615 × 0.300 × 0.0932 Inches)

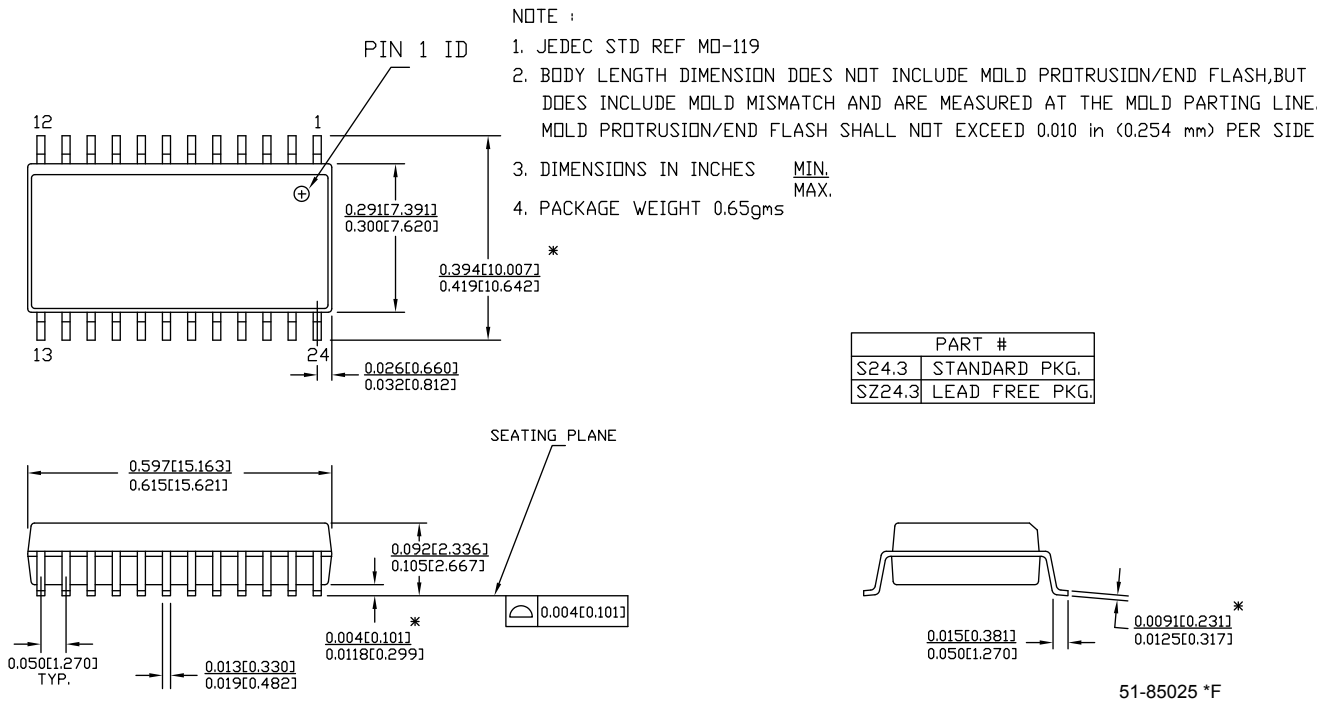


Figure 54. 24-pin PDIP (1.260 × 0.270 × 0.140 Inches) P24.3 Package Outline, 51-85013

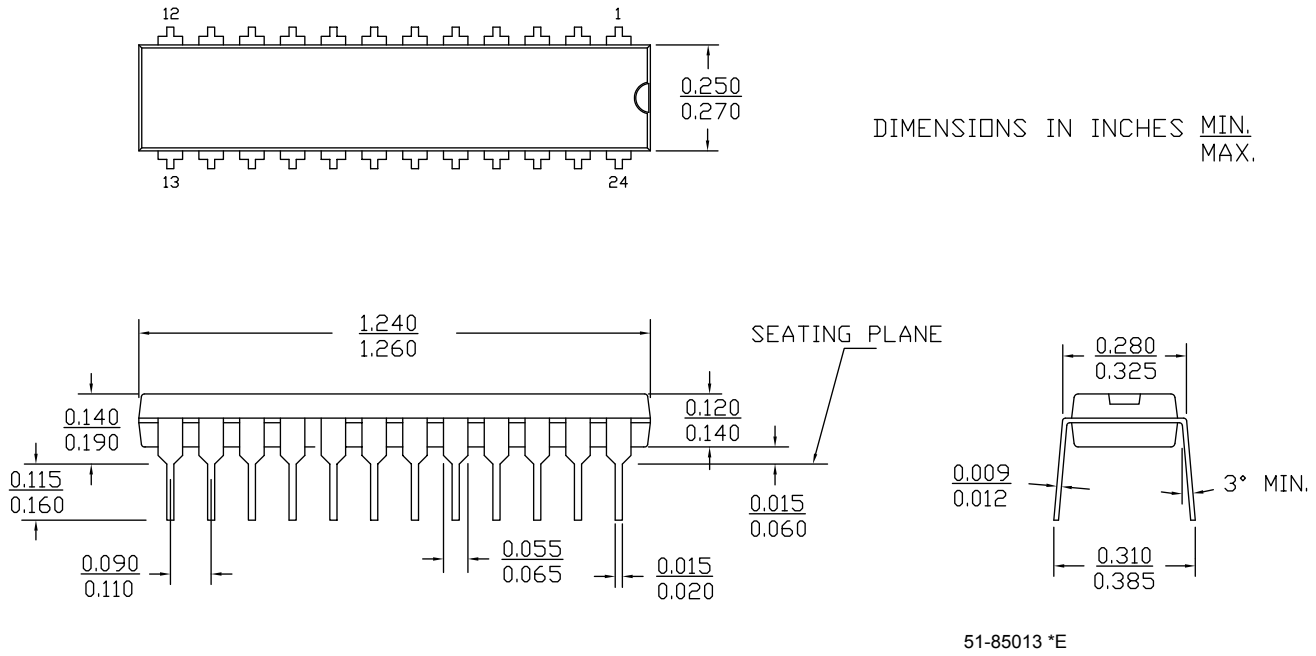


Figure 55. 24-pin QSOP (8.65 × 3.9 × 1.44 mm) O241 Package Outline, 51-85055

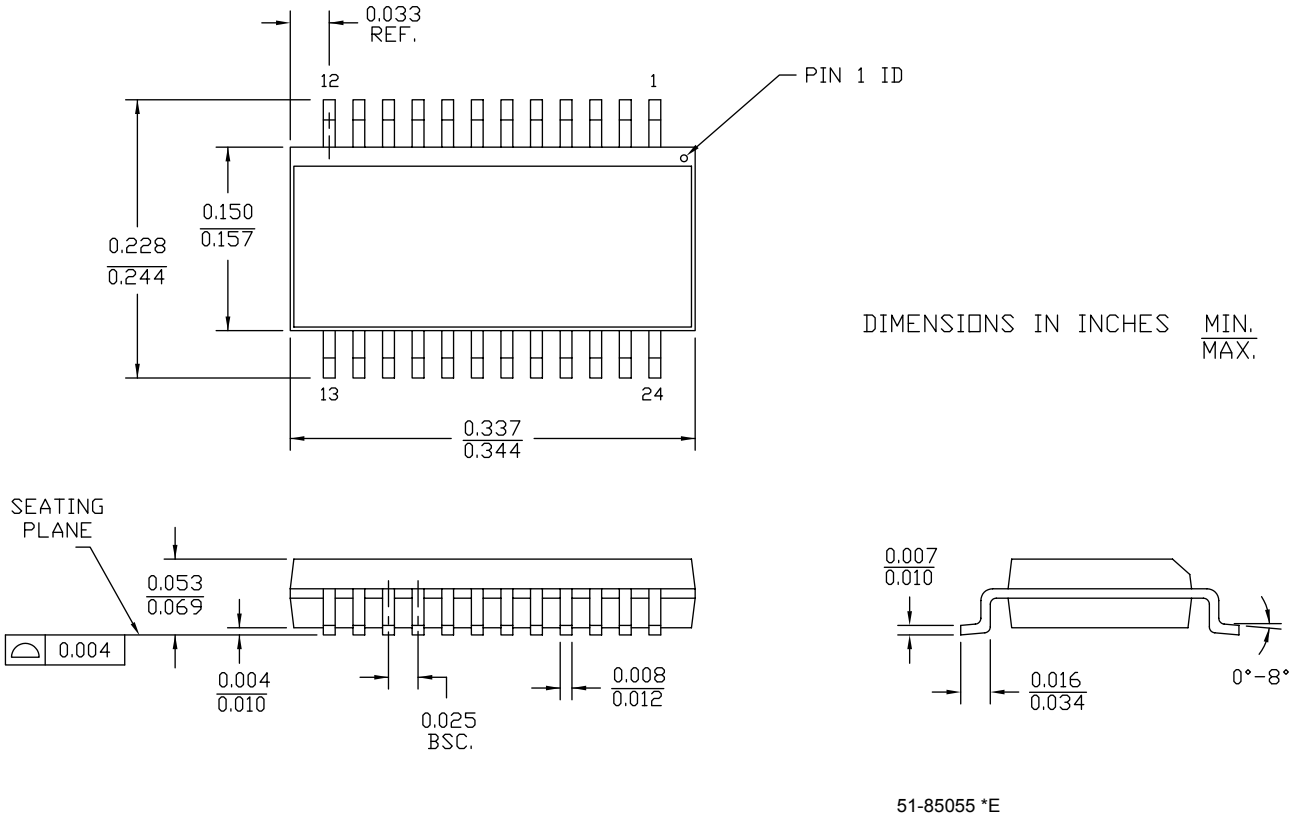


Figure 56. Die Form

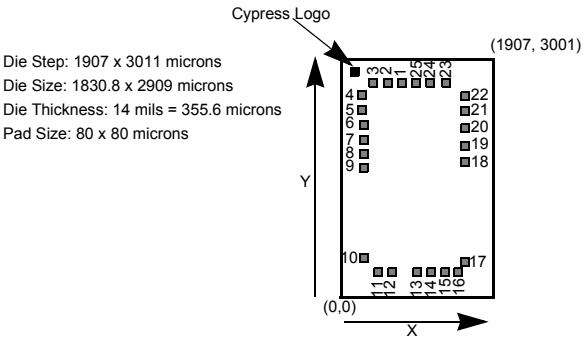


Table 11 below shows the die pad coordinates for the CY7C63722C-XC. The center location of each bond pad is relative to the bottom left corner of the die which has coordinate (0,0).

Table 11. CY7C63722C-XC Probe Pad Coordinates in microns ((0,0) to bond pad centers)

Pad Number	Pin Name	X (microns)	Y (microns)
1	P0.0	788.95	2843.15
2	P0.1	597.45	2843.15
3	P0.2	406.00	2843.15
4	P0.3	154.95	2687.95
5	P1.0	154.95	2496.45
6	P1.2	154.95	2305.05
7	P1.4	154.95	2113.60
8	P1.6	154.95	1922.05
9	Vss	154.95	1730.90
10	Vss	154.95	312.50
11	Vpp	363.90	184.85
12	VREG	531.70	184.85
13	XTALIN	1066.55	184.85
14	XTALOUT	1210.75	184.85
15	Vcc	1449.75	184.85
16	D–	1662.35	184.85
17	D+	1735.35	289.85
18	P1.7	1752.05	1832.75
19	P1.5	1752.05	2024.30
20	P1.3	1752.05	2215.75
21	P1.1	1752.05	2407.15
22	P0.7	1752.05	2598.65
23	P0.6	1393.25	2843.15
24	P0.5	1171.80	2843.15
25	P0.4	980.35	2843.15