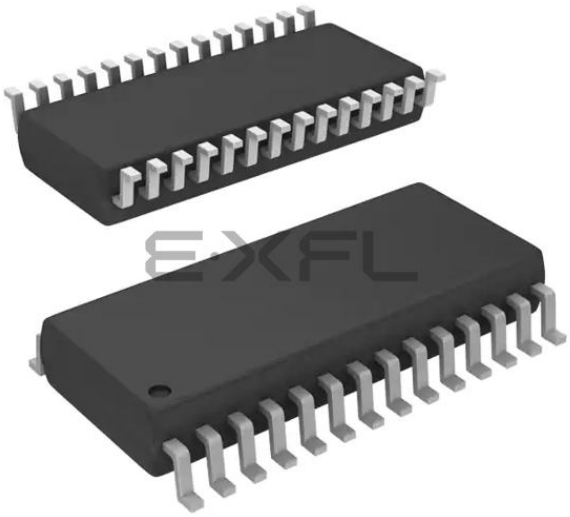


Welcome to E-XFL.COM

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

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

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Microcontroller
Core Processor	M8C
Program Memory Type	OTP (8kB)
Controller Series	CY7C640xx
RAM Size	256 x 8
Interface	I²C, USB, HAPI
Number of I/O	19
Voltage - Supply	4V ~ 5.25V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c64013c-sxc

Memory

The CY7C64013C and CY7C64113C have 8 KB of PROM.

Power on Reset, Watchdog and Free running Time

These parts include power-on reset logic, a Watchdog timer, and a 12-bit free-running timer. The power-on reset (POR) logic detects when power is applied to the device, resets the logic to a known state, and begins executing instructions at PROM address 0x0000. The Watchdog timer is used to ensure the microcontroller recovers after a period of inactivity. The firmware may become inactive for a variety of reasons, including errors in the code or a hardware failure such as waiting for an interrupt that never occurs.

I^2C and HAPI Interface

The microcontroller can communicate with external electronics through the GPIO pins. An I^2C-compatible interface accommodates a 100-kHz serial link with an external device. There is also a Hardware Assisted Parallel Interface (HAPI) which can be used to transfer data to an external device.

Timer

The free-running 12-bit timer clocked at 1 MHz provides two interrupt sources, 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 of the event and after the event is complete. The difference between the two readings indicates the duration of the event in microseconds. The upper four bits of the timer are latched into an internal register when the firmware reads the lower eight bits. A read from the upper four bits actually reads data from the internal register, instead of the timer. This feature eliminates the need for firmware to try to compensate if the upper four bits increment immediately after the lower eight bits are read.

Interrupts

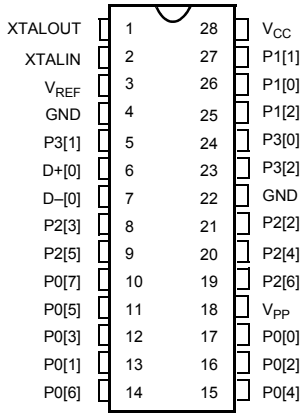
The microcontroller supports 11 maskable interrupts in the vectored interrupt controller. Interrupt sources include the USB Bus Reset interrupt, the 128-µs (bit 6) and 1.024-ms (bit 9) outputs from the free-running timer, five USB endpoints, the DAC port, the GPIO ports, and the I^2C-compatible master mode interface. The timer bits cause an interrupt (if enabled) when the bit toggles from LOW ‘0’ to HIGH ‘1.’ The USB endpoints interrupt after the USB host has written data to the endpoint FIFO or after the USB controller sends a packet to the USB host. The DAC ports have an additional level of masking that allows the user to select which DAC inputs can cause a DAC interrupt. The GPIO ports also 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 pin of the DAC port. Input transition polarity can be programmed for each GPIO port as part of the port configuration. The interrupt polarity can be rising edge (‘0’ to ‘1’) or falling edge (‘1’ to ‘0’).

Contents

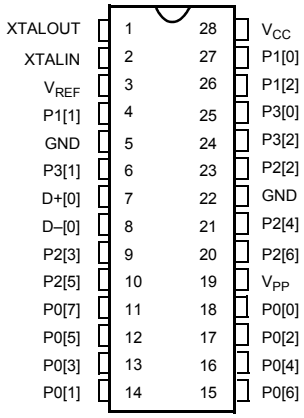
Pin Configurations

TOP VIEW

CY7C64013C
28-pin SOIC



CY7C64013C
28-pin PDIP



CY7C64113C
48-pin SSOP

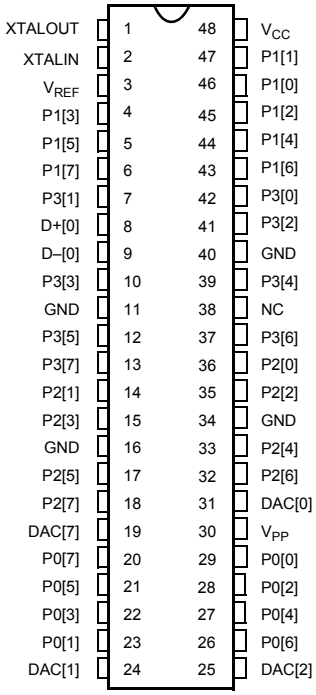


Table 2. I/O Register Summary

Register Name	I/O Address	Read/Write	Function	Page
Port 0 Data	0x00	R/W	GPIO Port 0 Data	15
Port 1 Data	0x01	R/W	GPIO Port 1 Data	16
Port 2 Data	0x02	R/W	GPIO Port 2 Data	16
Port 3 Data	0x03	R/W	GPIO Port 3 Data	16
Port 0 Interrupt Enable	0x04	W	Interrupt Enable for Pins in Port 0	17
Port 1 Interrupt Enable	0x05	W	Interrupt Enable for Pins in Port 1	17
Port 2 Interrupt Enable	0x06	W	Interrupt Enable for Pins in Port 2	18
Port 3 Interrupt Enable	0x07	W	Interrupt Enable for Pins in Port 3	18
GPIO Configuration	0x08	R/W	GPIO Port Configurations	16
HAPI and I^2C Configuration	0x09	R/W	HAPI Width and I^2C Position Configuration	21
USB Device Address A	0x10	R/W	USB Device Address A	32
EP A0 Counter Register	0x11	R/W	USB Address A, Endpoint 0 Counter	32
EP A0 Mode Register	0x12	R/W	USB Address A, Endpoint 0 Configuration	32
EP A1 Counter Register	0x13	R/W	USB Address A, Endpoint 1 Counter	32
EP A1 Mode Register	0x14	R/W	USB Address A, Endpoint 1 Configuration	32
EP A2 Counter Register	0x15	R/W	USB Address A, Endpoint 2 Counter	32
EP A2 Mode Register	0x16	R/W	USB Address A, Endpoint 2 Configuration	32
USB Status & Control	0x1F	R/W	USB Upstream Port Traffic Status and Control	31
Global Interrupt Enable	0x20	R/W	Global Interrupt Enable	26
Endpoint Interrupt Enable	0x21	R/W	USB Endpoint Interrupt Enables	27
Timer (LSB)	0x24	R	Lower 8 Bits of Free-running Timer (1 MHz)	20
Timer (MSB)	0x25	R	Upper 4 Bits of Free-running Timer	21
WDT Clear	0x26	W	Watchdog Timer Clear	14
I^2C Control & Status	0x28	R/W	I^2C Status and Control	22
I^2C Data	0x29	R/W	I^2C Data	22
DAC Data	0x30	R/W	DAC Data	19
DAC Interrupt Enable	0x31	W	Interrupt Enable for each DAC Pin	20
DAC Interrupt Polarity	0x32	W	Interrupt Polarity for each DAC Pin	20
DAC Isink	0x38-0x3F	W	Input Sink Current Control for each DAC Pin	19
Reserved	0x40		Reserved	
EP A3 Counter Register	0x41	R/W	USB Address A, Endpoint 3 Counter	32
EP A3 Mode Register	0x42	R/W	USB Address A, Endpoint 3 Configuration	32
EP A4 Counter Register	0x43	R/W	USB Address A, Endpoint 4 Counter	32
EP A4 Mode Register	0x44	R/W	USB Address A, Endpoint 4 Configuration	32
Reserved	0x48		Reserved	
Reserved	0x49		Reserved	
Reserved	0x4A		Reserved	
Reserved	0x4B		Reserved	
Reserved	0x4C		Reserved	

Instruction Set Summary

Refer to the *CYASM Assembler User's Guide* for more details.

Table 3. Instruction Set Summary

MNEMONIC	operand	opcode	cycles
HALT		00	7
ADD A,expr	data	01	4
ADD A,[expr]	direct	02	6
ADD A,[X+expr]	index	03	7
ADC A,expr	data	04	4
ADC A,[expr]	direct	05	6
ADC A,[X+expr]	index	06	7
SUB A,expr	data	07	4
SUB A,[expr]	direct	08	6
SUB A,[X+expr]	index	09	7
SBB A,expr	data	0A	4
SBB A,[expr]	direct	0B	6
SBB A,[X+expr]	index	0C	7
OR A,expr	data	0D	4
OR A,[expr]	direct	0E	6
OR A,[X+expr]	index	0F	7
AND A,expr	data	10	4
AND A,[expr]	direct	11	6
AND A,[X+expr]	index	12	7
XOR A,expr	data	13	4
XOR A,[expr]	direct	14	6
XOR A,[X+expr]	index	15	7
CMP A,expr	data	16	5
CMP A,[expr]	direct	17	7
CMP A,[X+expr]	index	18	8
MOV A,expr	data	19	4
MOV A,[expr]	direct	1A	5
MOV A,[X+expr]	index	1B	6
MOV X,expr	data	1C	4
MOV X,[expr]	direct	1D	5
reserved		1E	
XPAGE		1F	4
MOV A,X		40	4
MOV X,A		41	4
MOV PSP,A		60	4
CALL	addr	50 - 5F	10
JMP	addr	80-8F	5
CALL	addr	90-9F	10
JZ	addr	A0-AF	5
JNZ	addr	B0-BF	5

MNEMONIC	operand	opcode	cycles
NOP		20	4
INC A	acc	21	4
INC X	x	22	4
INC [expr]	direct	23	7
INC [X+expr]	index	24	8
DEC A	acc	25	4
DEC X	x	26	4
DEC [expr]	direct	27	7
DEC [X+expr]	index	28	8
IORD expr	address	29	5
IOWR expr	address	2A	5
POP A		2B	4
POP X		2C	4
PUSH A		2D	5
PUSH X		2E	5
SWAP A,X		2F	5
SWAP A,DSP		30	5
MOV [expr],A	direct	31	5
MOV [X+expr],A	index	32	6
OR [expr],A	direct	33	7
OR [X+expr],A	index	34	8
AND [expr],A	direct	35	7
AND [X+expr],A	index	36	8
XOR [expr],A	direct	37	7
XOR [X+expr],A	index	38	8
IOWX [X+expr]	index	39	6
CPL		3A	4
ASL		3B	4
ASR		3C	4
RLC		3D	4
RRC		3E	4
RET		3F	8
DI		70	4
EI		72	4
RETI		73	8
JC	addr	C0-CF	5
JNC	addr	D0-DF	5
JACC	addr	E0-EF	7
INDEX	addr	F0-FF	14

Program Memory Organization

after reset / 14-bit PC	Address	
→	0x0000	Program execution begins here after a reset
	0x0002	USB Bus Reset interrupt vector
	0x0004	128-µs timer interrupt vector
	0x0006	1.024-ms timer interrupt vector
	0x0008	USB address A endpoint 0 interrupt vector
	0x000A	USB address A endpoint 1 interrupt vector
	0x000C	USB address A endpoint 2 interrupt vector
	0x000E	USB address A endpoint 3 interrupt vector
	0x0010	USB address A endpoint 4 interrupt vector
	0x0012	Reserved
	0x0014	DAC interrupt vector
	0x0016	GPIO interrupt vector
	0x0018	I^2C interrupt vector
	0x001A	**Program Memory begins here**
	0x1FDF	**8 KB (-32) PROM ends here (CY7C64013C, CY7C64113C)**

8-Bit Accumulator (A)

The accumulator is the general-purpose register for the microcontroller.

8-Bit Temporary Register (X)

The “X” register is available to the firmware for temporary storage of intermediate results. The microcontroller can perform indexed operations based on the value in X. Refer to Indexed on page 13 for additional information.

8-Bit Program Stack Pointer (PSP)

During a reset, the program stack pointer (PSP) is set to 0x00 and “grows” upward from this address. The PSP may be set by firmware, using the MOV PSP,A instruction. The PSP supports interrupt service under hardware control and CALL, RET, and RETI instructions under firmware control. The PSP is not readable by the firmware.

During an interrupt acknowledge, interrupts are disabled and the 14-bit 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 PSP, then the PSP is incremented. The second byte is stored in memory addressed by the PSP, and the PSP is incremented again. The overall effect is to store the program counter and flags on the program “stack” and increment the PSP by two.

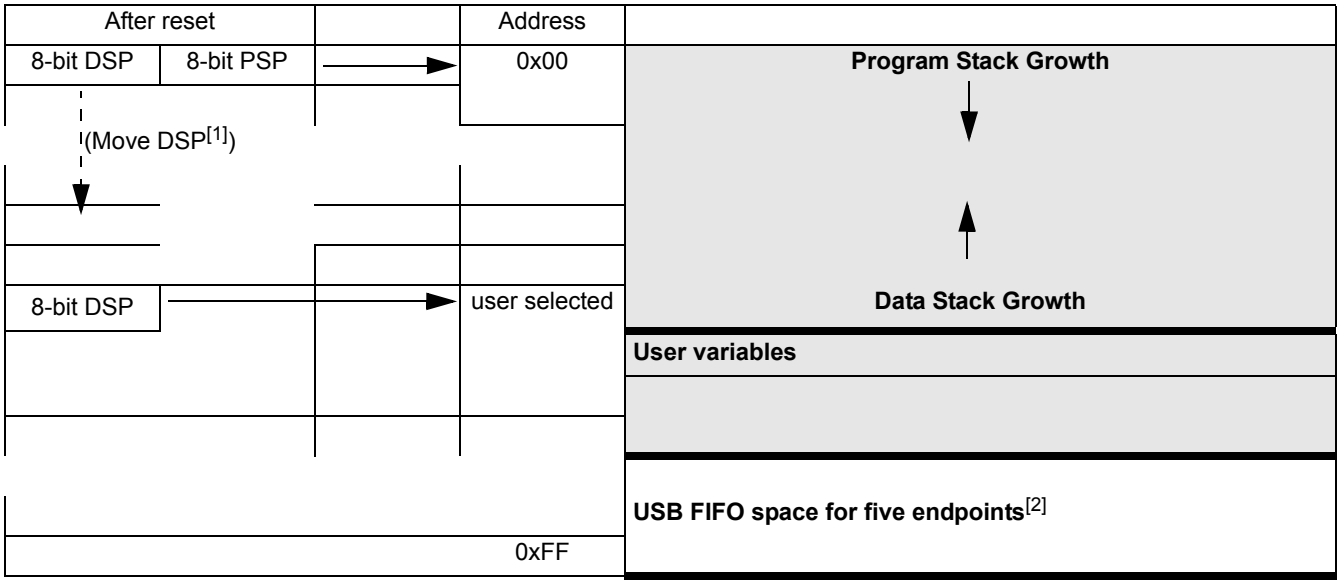
The Return from Interrupt (RETI) instruction decrements the PSP, then restores the second byte from memory addressed by the PSP. The PSP is decremented again and the first byte is restored from memory addressed by the PSP. After the program counter and flags have been restored from stack, the interrupts are enabled. The overall effect is to restore the program counter and flags from the program stack, decrement the PSP by two, and reenable 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 the program stack and decrements the PSP by two.

Data Memory Organization

The CY7C64x13C microcontrollers provide 256 bytes of data RAM. Normally, the SRAM is partitioned into four areas: program stack, user variables, data stack, and USB endpoint FIFOs. The following is one example of where the program stack, data stack, and user variables areas could be located.



Notes
1. Refer to 8-Bit Data Stack Pointer (DSP) on page 13 for a description of DSP.
2. Endpoint sizes are fixed by the Endpoint Size Bit (I/O register 0x1F, Bit 7), see Table 34 on page 32.

resonator does not allow the microcontroller to meet the timing specifications of full speed USB and therefore a ceramic resonator is not recommended with these parts.

An external 6-MHz clock can be applied to the XTALIN pin if the XTALOUT pin is left open. Grounding the XTALOUT pin when driving XTALIN with an oscillator does not work because the internal clock is effectively shorted to ground.

Reset

The CY7C64x13C supports two resets: Power-On Reset (POR) and a Watchdog Reset (WDR). Each of these resets causes:

- all registers to be restored to their default states,
- the USB Device Address to be set to 0,
- all interrupts to be disabled,
- the PSP and Data Stack Pointer (DSP) to be set to memory address 0x00.

The occurrence of a reset is recorded in the Processor Status and Control Register, as described in Processor Status and Control Register on page 25. Bits 4 and 6 are used to record the occurrence of POR and WDR, respectively. Firmware can interrogate these bits to determine the cause of a reset.

Program execution starts at ROM address 0x0000 after a 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. The firmware reset handler should configure the hardware before the “main” loop of code. Attempting to execute a RET or RETI in the firmware reset handler causes unpredictable execution results.

Power-On Reset (POR)

When V_{CC} is first applied to the chip, the Power-On Reset (POR) signal is asserted and the CY7C64x13C enters a “semi-suspend” state. During the semi-suspend state, which is different from the suspend state defined in the USB specification, the oscillator and all other blocks of the part are functional, except for the CPU. This semi-suspend time ensures that both a valid V_{CC} level is reached and that the internal PLL has time to stabilize before full operation begins. When the V_{CC} has risen above approximately 2.5 V, and the oscillator is stable, the POR is deasserted and the on-chip timer starts counting. The first 1 ms of suspend time is not interruptible, and the semi-suspend state continues for an additional 95 ms unless the count is bypassed by a USB Bus Reset on the upstream port. The 95 ms provides time for V_{CC} to stabilize at a valid operating voltage before the chip executes code.

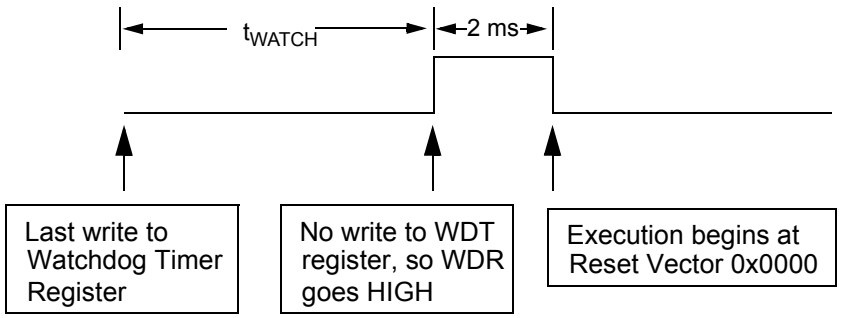
If a USB Bus Reset occurs on the upstream port during the 95-ms semi-suspend time, the semi-suspend state is aborted and program execution begins immediately from address 0x0000. In this case, the Bus Reset interrupt is pending but not serviced until firmware sets the USB Bus Reset Interrupt Enable bit (bit 0 of register 0x20) and enables interrupts with the EI command.

The POR signal is asserted whenever V_{CC} drops below approximately 2.5 V, and remains asserted until V_{CC} rises above this level again. Behavior is the same as described above.

Watchdog Reset (WDR)

The Watchdog Timer Reset (WDR) occurs when the internal Watchdog timer rolls over. Writing any value to the write-only Watchdog Restart Register at address 0x26 clears the timer. The timer rolls over and WDR occurs if it is not cleared within t_{WATCH} (8 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 Watchdog Timer Reset lasts for 2 ms, after which the microcontroller begins execution at ROM address 0x0000.

Figure 2. Watchdog Reset (WDR)



The USB transmitter is disabled by a Watchdog Reset because the USB Device Address Register is cleared. Otherwise, the USB Controller would respond to all address 0 transactions.

It is possible for the WDR bit of the Processor Status and Control Register (0xFF) to be set following a POR event. The WDR bit should be ignored If the firmware interrogates the Processor Status and Control Register for a Set condition on the WDR bit and if the POR (bit 3 of register 0xFF) bit is set.

Suspend Mode

The CY7C64x13C can be placed into a low-power state by setting the Suspend bit of the Processor Status and Control register. All logic blocks in the device are turned off except the GPIO interrupt logic and the USB receiver. The clock oscillator and PLL, as well as the free-running and Watchdog timers, are shut down. Only the occurrence of an enabled GPIO interrupt or non-idle bus activity at a USB upstream or downstream port

Table 25. I²C Status and Control Register Bit Definitions

Bit	Name	Description
0	I²C Enable	When set to '1', the I²C-compatible function is enabled. When cleared, I²C GPIO pins operate normally.
1	Received Stop	Reads 1 only in slave receive mode, when I²C Stop bit detected (unless firmware did not ACK the last transaction).
2	ARB Lost/Restart	Reads 1 to indicate master has lost arbitration. Reads 0 otherwise. Write to 1 in master mode to perform a restart sequence (also set Continue bit).
3	Addr	Reads 1 during first byte after start/restart in slave mode, or if master loses arbitration. Reads 0 otherwise. This bit should always be written as 0.
4	ACK	In receive mode, write 1 to generate ACK, 0 for no ACK. In transmit mode, reads 1 if ACK was received, 0 if no ACK received.
5	Xmit Mode	Write to 1 for transmit mode, 0 for receive mode.
6	Continue/Busy	Write 1 to indicate ready for next transaction. Reads 1 when I²C-compatible block is busy with a transaction, 0 when transaction is complete.
7	MSTR Mode	Write to 1 for master mode, 0 for slave mode. This bit is cleared if master loses arbitration. Clearing from 1 to 0 generates Stop bit.

Bit 7 : MSTR Mode

Setting this bit to 1 causes the I²C-compatible block to initiate a master mode transaction by sending a start bit and transmitting the first data byte from the data register (this typically holds the target address and R/W bit). Subsequent bytes are initiated by setting the Continue bit, as described below.

Clearing this bit (set to 0) causes the GPIO pins to operate normally

In master mode, the I²C-compatible block generates the clock (SCK), and drives the data line as required depending on transmit or receive state. The I²C-compatible block performs any required arbitration and clock synchronization. IN the event of a loss of arbitration, this MSTR bit is cleared, the ARB Lost bit is set, and an interrupt is generated by the microcontroller. If the chip is the target of an external master that wins arbitration, then the interrupt is held off until the transaction from the external master is completed.

When MSTR Mode is cleared from 1 to 0 by a firmware write, an I²C Stop bit is generated.

Bit 6 : Continue / Busy

This bit is written by the firmware to indicate that the firmware is ready for the next byte transaction to begin. In other words, the bit has responded to an interrupt request and has completed the required update or read of the data register. During a read this bit indicates if the hardware is busy and is locking out additional writes to the I²C Status and Control register. This locking allows the hardware to complete certain operations that may require an extended period of time. Following an I²C interrupt, the I²C-compatible block does not return to the Busy state until firmware sets the Continue bit. This allows the firmware to make one control register write without the need to check the Busy bit.

Bit 5 : Xmit Mode

This bit is set by firmware to enter transmit mode and perform a data transmit in master or slave mode. Clearing this bit sets the part in receive mode. Firmware generally determines the value of this bit from the R/W bit associated with the I²C address packet. The Xmit Mode bit state is ignored when initially writing the MSTR Mode or the Restart bits, as these cases always cause transmit mode for the first byte.

Bit 4 : ACK

This bit is set or cleared by firmware during receive operation to indicate if the hardware should generate an ACK signal on the I²C-compatible bus. Writing a 1 to this bit generates an ACK (SDA LOW) on the I²C-compatible bus at the ACK bit time. During transmits (Xmit Mode = 1), this bit should be cleared.

Bit 3 : Addr

This bit is set by the I²C-compatible block during the first byte of a slave receive transaction, after an I²C start or restart. The Addr bit is cleared when the firmware sets the Continue bit. This bit allows the firmware to recognize when the master has lost arbitration, and in slave mode it allows the firmware to recognize that a start or restart has occurred.

Bit 2 : ARB Lost/Restart

This bit is valid as a status bit (ARB Lost) after master mode transactions. In master mode, set this bit (along with the Continue and MSTR Mode bits) to perform an I²C restart sequence. The I²C target address for the restart must be written to the data register before setting the Continue bit. To prevent false ARB Lost signals, the Restart bit is cleared by hardware during the restart sequence.

Processor Status and Control Register

Table 27. Processor Status and Control Register

Processor Status and Control **ADDRESS 0xFF**

Bit #	7	6	5	4	3	2	1	0
Bit Name	IRQ Pending	Watchdog Reset	USB Bus Reset Interrupt	Power-On Reset	Suspend	Interrupt Enable Sense	Reserved	Run
Read/Write	R	R/W	R/W	R/W	R/W	R	R/W	R/W
Reset	0	0	0	1	0	0	0	1

Bit 0: Run

This bit is manipulated by the HALT instruction. When Halt is executed, all the bits of the Processor Status and Control Register are cleared to 0. Since the run bit is cleared, the processor stops at the end of the current instruction. The processor remains halted until an appropriate reset occurs (power-on or Watchdog). This bit should normally be written as a '1.'

Bit 1: Reserved

Bit 1 is reserved and must be written as a zero.

Bit 2: Interrupt Enable Sense

This bit indicates whether interrupts are enabled or disabled. Firmware has no direct control over this bit as writing a zero or one to this bit position has 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 (Table 28 on page 26) and USB End Point Interrupt Enable Register (Table 29 on page 27). Instructions DI, EI, and RETI manipulate the state of this bit.

Bit 3: Suspend

Writing a '1' to the Suspend bit halts the processor and cause the microcontroller to enter the suspend mode that significantly reduces power consumption. A pending, enabled interrupt or USB bus activity causes the device to come out of suspend. After coming out of suspend, the device resumes firmware execution at the instruction following the IOWR which put the part into suspend. An IOWR attempting to put the part into suspend is ignored if USB bus activity is present. See Suspend Mode on page 14 for more details on suspend mode operation.

Bit 4: Power-On Reset

The Power-On Reset is set to '1' during a power-on reset. The firmware can check bits 4 and 6 in the reset handler to determine whether a reset was caused by a power-on condition or a Watchdog timeout. A POR event may be followed by a Watchdog reset before firmware begins executing, as explained below.

Bit 5: USB Bus Reset Interrupt

The USB Bus Reset Interrupt bit is set when the USB Bus Reset is detected on receiving a USB Bus Reset signal on the upstream port. The USB Bus Reset signal is a single-ended zero (SE0) that lasts from 12 to 16 µs. An SE0 is defined as the condition in which both the D+ line and the D– line are LOW at the same time.

Bit 6: Watchdog Reset

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

Bit 7: IRQ Pending

The IRQ pending, when set, indicates that one or more of the interrupts has been recognized as active. An interrupt remains pending until its interrupt enable bit is set (Table 28 on page 26, Table 29 on page 27) and interrupts are globally enabled. At that point, the internal interrupt handling sequence clears this bit until another interrupt is detected as pending.

During power-up, the Processor Status and Control Register is set to 00010001, which indicates a POR (bit 4 set) has occurred and no interrupts are pending (bit 7 clear). During the 96 ms suspend at start-up (explained in Power-On Reset (POR) on page 14), a Watchdog Reset also occurs unless this suspend is aborted by an upstream SE0 before 8 ms. If a WDR occurs during the power-up suspend interval, firmware reads 01010001 from the Status and Control Register after power-up. Normally, the POR bit should be cleared so a subsequent WDR can be clearly identified. If an upstream bus reset is received before firmware examines this register, the Bus Reset bit may also be set.

During a Watchdog Reset, the Processor Status and Control Register(Table 27 on page 25) is set to 01XX0001b, which indicates a Watchdog Reset (bit 6 set) has occurred and no interrupts are pending (bit 7 clear). The Watchdog Reset does not effect the state of the POR and the Bus Reset Interrupt bits.

Table 29. USB Endpoint Interrupt Enable Register

USB Endpoint Interrupt Enable **ADDRESS 0X21**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	EPB1 Interrupt Enable	EPB0 Interrupt Enable	EPA2 Interrupt Enable	EPA1 Interrupt Enable	EPA0 Interrupt Enable
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	0	0	0	0	0

Figure 6. USB Endpoint Interrupt Enable Register

Bit 0: **EPA0 Interrupt Enable**

1= Enable Interrupt on data activity through endpoint A0;
0= Disable Interrupt on data activity through endpoint A0

Bit 1: **EPA1 Interrupt Enable**

1= Enable Interrupt on data activity through endpoint A1;
0= Disable Interrupt on data activity through endpoint A1

Bit 2: **EPA2 Interrupt Enable**

1= Enable Interrupt on data activity through endpoint A2;
0= Disable Interrupt on data activity through endpoint A2.

Bit 3: **EPB0 Interrupt Enable**

1= Enable Interrupt on data activity through endpoint B0;
0= Disable Interrupt on data activity through endpoint B0

Bit 4: **EPB1 Interrupt Enable**

1= Enable Interrupt on data activity through endpoint B1;
0= Disable Interrupt on data activity through endpoint B1

Bit [7..5] : Reserved

During a reset, the contents the Global Interrupt Enable Register and USB End Point Interrupt Enable Register are cleared, effectively, disabling all interrupts

The interrupt controller contains a separate flip-flop for each interrupt. See Figure 7 on page 28 for the logic block diagram of the interrupt controller. When an interrupt is generated, it is first registered as a pending interrupt. It stays pending until it is serviced or a reset occurs. A pending interrupt only generates an interrupt request if it is enabled by the corresponding bit in the interrupt enable registers. The highest priority interrupt request is serviced following the completion of the currently executing instruction.

When servicing an interrupt, the hardware does the following

1. Disables 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, Table 27 on page 25).
2. Clears the flip-flop of the current interrupt.
3. Generates an automatic CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector, see Interrupt Vectors).

The instruction in the interrupt table is typically a JMP instruction to the address of the Interrupt Service Routine (ISR). The user can re-enable interrupts in the interrupt service routine by executing an EI instruction. Interrupts can be nested to a level limited only by the available stack space.

The Program Counter value as well as the Carry and Zero flags (CF, ZF) are stored onto the Program Stack by the automatic CALL instruction generated as part of the interrupt acknowledge process. The user firmware is responsible for ensuring that the processor state is preserved and restored during an interrupt. The PUSH A instruction should typically be used as the first command in the ISR to save the accumulator value and the POP A instruction should be used to restore the accumulator value just before the RETI instruction. 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 exists the ISR. While the global interrupt enable bit is cleared, the presence of a pending interrupt can be detected by examining the IRQ Sense bit (Bit 7 in the Processor Status and Control Register).

Interrupt Vectors

The Interrupt Vectors supported by the USB Controller are listed in Table 30 on page 29. The lowest-numbered interrupt (USB Bus Reset interrupt) has the highest priority, and the highest-numbered interrupt (I^2C interrupt) has the lowest priority.

Table 30. Interrupt Vector Assignments

Interrupt Vector Number	ROM Address	Function
Not Applicable	0x0000	Execution after Reset begins here
1	0x0002	USB Bus Reset interrupt
2	0x0004	128- µs timer interrupt
3	0x0006	1.024-ms timer interrupt
4	0x0008	USB Address A Endpoint 0 interrupt
5	0x000A	USB Address A Endpoint 1 interrupt
6	0x000C	USB Address A Endpoint 2 interrupt
7	0x000E	USB Address A Endpoint 3 interrupt
8	0x0010	USB Address A Endpoint 4 interrupt
9	0x0012	Reserved
10	0x0014	DAC interrupt
11	0x0016	GPIO interrupt
12	0x0018	I^2C interrupt

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 executes a minimum of 16 clocks (1+10+5) or a maximum of 20 clocks (5+10+5) after the interrupt is issued. For a 12-MHz internal clock (6-MHz crystal), 20 clock periods is 20 / 12 MHz = 1.667 µs.

USB Bus Reset Interrupt

The USB Controller recognizes a USB Reset when a Single Ended Zero (SE0) condition persists on the upstream USB port for 12–16 µs (the Reset may be recognized for an SE0 as short as 12 µs, but is always recognized for an SE0 longer than 16 µs). SE0 is defined as the condition in which both the D+ line and the D– line are LOW. Bit 5 of the Status and Control Register is set to record this event. The interrupt is asserted at the end of the Bus Reset. If the USB reset occurs during the start-up delay following a POR, the delay is aborted as described in Power-On Reset (POR) on page 14. The USB Bus Reset Interrupt is generated when the SE0 state is deasserted.

A USB Bus Reset clears the following registers:

SIE Section:USB Device Address Registers (0x10, 0x40)

Timer Interrupt

There are two periodic timer interrupts: the 128- µs interrupt and the 1.024-ms interrupt. 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.

USB Endpoint Interrupts

There are five 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 is generated.

DAC Interrupt

Each DAC I/O pin can generate an interrupt, if enabled. The interrupt polarity for each DAC I/O pin is programmable. A positive polarity is a rising edge input while a negative polarity is a falling edge input. All of the DAC pins share a single interrupt vector, which means the firmware needs to read the DAC port to determine which pin or pins caused an interrupt.

If one DAC pin has triggered an interrupt, no other DAC pins can cause a DAC interrupt until that pin has returned to its inactive (non-trigger) state or the corresponding interrupt enable bit is cleared. The USB Controller does not assign interrupt priority to different DAC pins and the DAC Interrupt Enable Register is not cleared during the interrupt acknowledge process.

GPIO/HAPI Interrupt

Each of the GPIO pins can generate an interrupt, if enabled. The interrupt polarity can be programmed for each GPIO port as part of the GPIO configuration. All of the GPIO pins share a single interrupt vector, which means the firmware needs to read the GPIO ports with enabled interrupts to determine which pin or pins caused an interrupt. A block diagram of the GPIO interrupt logic is shown in Figure 8 on page 30. Refer to GPIO Configuration Port on page 16 and GPIO Interrupt Enable Ports on page 17 for more information of setting GPIO interrupt polarity and enabling individual GPIO interrupts.

If one port pin has 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 USB Controller does not assign interrupt priority to different port pins and the Port Interrupt Enable Registers are not cleared during the interrupt acknowledge process.

the corresponding pins as possible, to meet the USB driver requirements of the USB specifications.

USB Serial Interface Engine (SIE)

The SIE allows the CY7C64x13C 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/STALL
- Token type identification
- Address checking

Firmware is required to handle the following USB interface tasks:

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

USB Enumeration

The USB device is enumerated under firmware control. The following is a brief summary of the typical enumeration process of the CY7C64x13C by the USB host. For a detailed description of the enumeration process, refer to the USB specification.

In this description, ‘Firmware’ refers to embedded firmware in the CY7C64x13C 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 FIFOs.
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.

USB Upstream Port Status and Control

USB status and control is regulated by the USB Status and Control Register, as shown in Table 31. All bits in the register are cleared during reset.

Table 31. USB Status and Control Register

USB Status and Control — **ADDRESS 0x1F**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Endpoint Size	Endpoint Mode	D+ Upstream	D– Upstream	Bus Activity	Control Action Bit 2	Control Action Bit 1	Control Action Bit 0
Read/Write	R/W	R/W	R	R	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits[2..0] : Control Action

Set to control action as per Table 32.The three control bits allow the upstream port to be driven manually by firmware. For normal USB operation, all of these bits must be cleared. Table 32 shows how the control bits affect the upstream port.

Table 32. Control Bit Definition for Upstream Port

Control Bits	Control Action
000	Not Forcing (SIE Controls Driver)
001	Force D+[0] HIGH, D–[0] LOW
010	Force D+[0] LOW, D–[0] HIGH
011	Force SE0; D+[0] LOW, D–[0] LOW
100	Force D+[0] LOW, D–[0] LOW
101	Force D+[0] HiZ, D–[0] LOW
110	Force D+[0] LOW, D–[0] HiZ
111	Force D+[0] HiZ, D–[0] HiZ

USB Mode Tables

Table 38. USB Register Mode Encoding

Mode	Mode Bits	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
Isochronous Out	0101	ignore	ignore	always	For Isochronous endpoints
Status In Only	0110	accept	TX 0 BYte	stall	For Control Endpoints
Isochronous In	0111	ignore	TX Count	ignore	For Isochronous endpoints
Nak Out	1000	ignore	ignore	NAK	Is set by SIE on an ACK from mode 1001 (Ack Out)
Ack Out(STALL[3]=0) Ack Out(STALL[3]=1)	1001 1001	ignore ignore	ignore ignore	ACK stall	On issuance of an ACK this mode is changed by SIE to 1000 (NAK Out)
Nak Out - Status In	1010	accept	TX 0 BYte	NAK	Is set by SIE on an ACK from mode 1011 (Ack Out- Status In)
Ack Out - Status In	1011	accept	TX 0 BYte	ACK	On issuance of an ACK this mode is changed by SIE to 1010 (NAK Out - Status In)
Nak In	1100	ignore	NAK	ignore	Is set by SIE on an ACK from mode 1101 (Ack In)
Ack IN(STALL[3]=0) Ack IN(STALL[3]=1)	1101 1101	ignore ignore	TX Count stall	ignore ignore	On issuance of an ACK this mode is changed by SIE to 1100 (NAK In)
Nak In - Status Out	1110	accept	NAK	check	Is set by SIE on an ACK from mode 1111 (Ack In - Status Out)
Ack In - Status Out	1111	accept	TX Count	check	On issuance of an ACK this mode is changed by SIE to 1110 (NAK In - Status Out)

Mode

This lists the mnemonic given to the different modes that can be set in the Endpoint Mode Register by writing to the lower nibble (bits 0..3). The bit settings for different modes are covered in the column marked “Mode Bits”. The Status IN and Status OUT represent the Status stage in the IN or OUT transfer involving the control endpoint.

Mode Bits

These column lists the encoding for different modes by setting Bits[3..0] of the Endpoint Mode register. This modes represents how the SIE responds to different tokens sent by the host to an endpoint. For instance, if the mode bits are set to “0001” (NAK IN/OUT), the SIE will respond with an

- ACK on receiving a SETUP token from the host
- NAK on receiving an OUT token from the host
- NAK on receiving an IN token from the host

Refer to I2C-compatible Controller on page 22 for more information on the SIE functioning

SETUP, IN and OUT

These columns shows the SIE’s response to the host on receiving a SETUP, IN and OUT token depending on the mode set in the Endpoint Mode Register.

A “Check” on the OUT token column, implies that on receiving an OUT token the SIE checks to see whether the OUT packet is of zero length and has a Data Toggle (DTOG) set to ‘1.’ If the DTOG bit is set and the received OUT Packet has zero length, the OUT is ACKed to complete the transaction. If either of this condition is not met the SIE will respond with a STALLL or just ignore the transaction.

A “TX Count” entry in the IN column implies that the SIE transmit the number of bytes specified in the Byte Count (bits 3..0 of the Endpoint Count Register) to the host in response to the IN token received.

A “TX0 Byte” entry in the IN column implies that the SIE transmit a zero length byte packet in response to the IN token received from the host.

An “Ignore” in any of the columns means that the device will not send any handshake tokens (no ACK) to the host.

An “Accept” in any of the columns means that the device will respond with an ACK to a valid SETUP transaction tot he host.

Comments

Some Mode Bits are automatically changed by the SIE in response to certain USB transactions. For example, if the Mode Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode, the SIE will change the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACK’ing a valid status stage OUT token.

Table 39. Details of Modes for Differing Traffic Conditions (see Table 38 for the decode legend)

SETUP (if accepting SETUPs)																	
Properties of Incoming Packet							**Changes made by SIE to Internal Registers and Mode Bits**										
Mode Bits				**token**	**count**	**buffer**	**dval**	**DTOG**	**DVAL**	**COUNT**	**Setup**	**In**	**Out**	**ACK**	**Mode Bits**	**Response**	**Intr**
See Table 38				Setup	<= 10	data	valid	updates	1	updates	1	UC	UC	1	0 0 0 1	ACK	yes
See Table 38				Setup	> 10	junk	x	updates	updates	updates	1	UC	UC	UC	NoChange	ignore	yes
See Table 38				Setup	x	junk	invalid	updates	0	updates	1	UC	UC	UC	NoChange	ignore	yes
Properties of Incoming Packet								**Changes made by SIE to Internal Registers and Mode Bits**									
Mode Bits				**token**	**count**	**buffer**	**dval**	**DTOG**	**DVAL**	**COUNT**	**Setup**	**In**	**Out**	**ACK**	**Mode Bits**	**Response**	**Intr**
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
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
0	0	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange	Stall	yes
CONTROL WRITE																	
Properties of Incoming Packet								**Changes made by SIE to Internal Registers and Mode Bits**									
Mode Bits				**token**	**count**	**buffer**	**dval**	**DTOG**	**DVAL**	**COUNT**	**Setup**	**In**	**Out**	**ACK**	**Mode Bits**	**Response**	**Intr**
Normal Out/premature status In																	
1	0	1	1	Out	<= 10	data	valid	updates	1	updates	UC	UC	1	1	1 0 1 0	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	1	NoChange	TX 0	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

Figure 10. Clock Timing

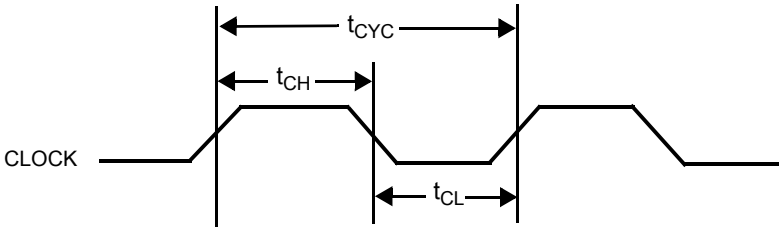


Figure 11. USB Data Signal Timing

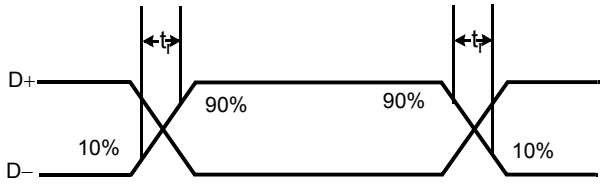
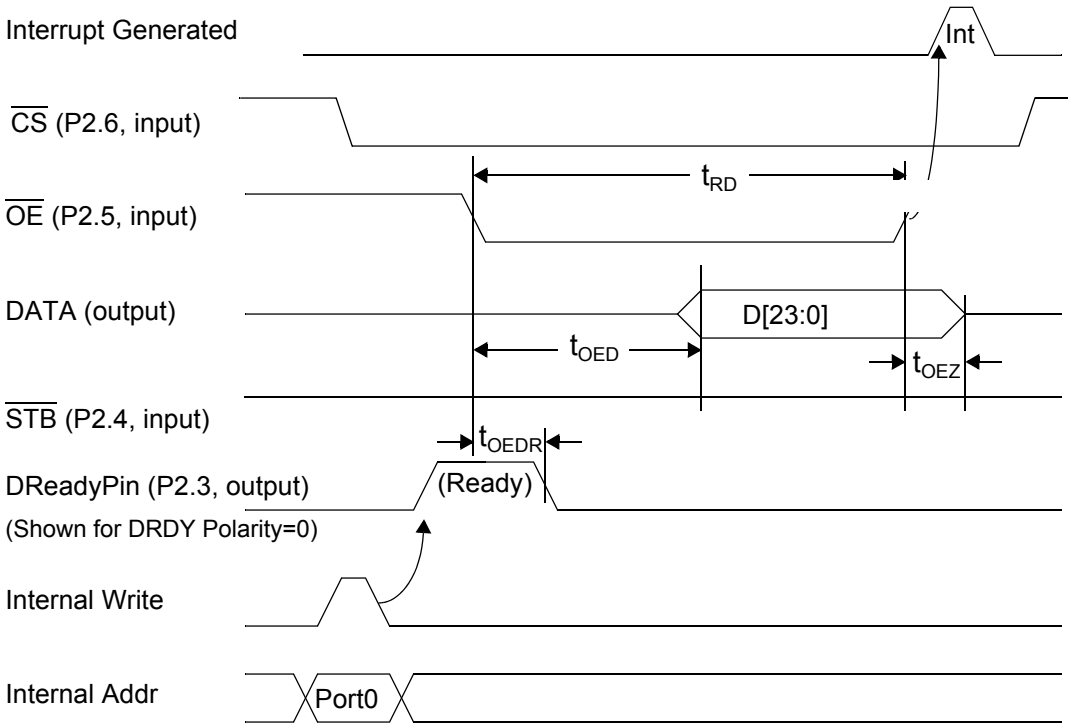


Figure 12. HAPI Read by External Interface from USB Microcontroller



Package Diagrams

Figure 14. 48-pin SSOP (300 Mils) Package Outline, 51-85061

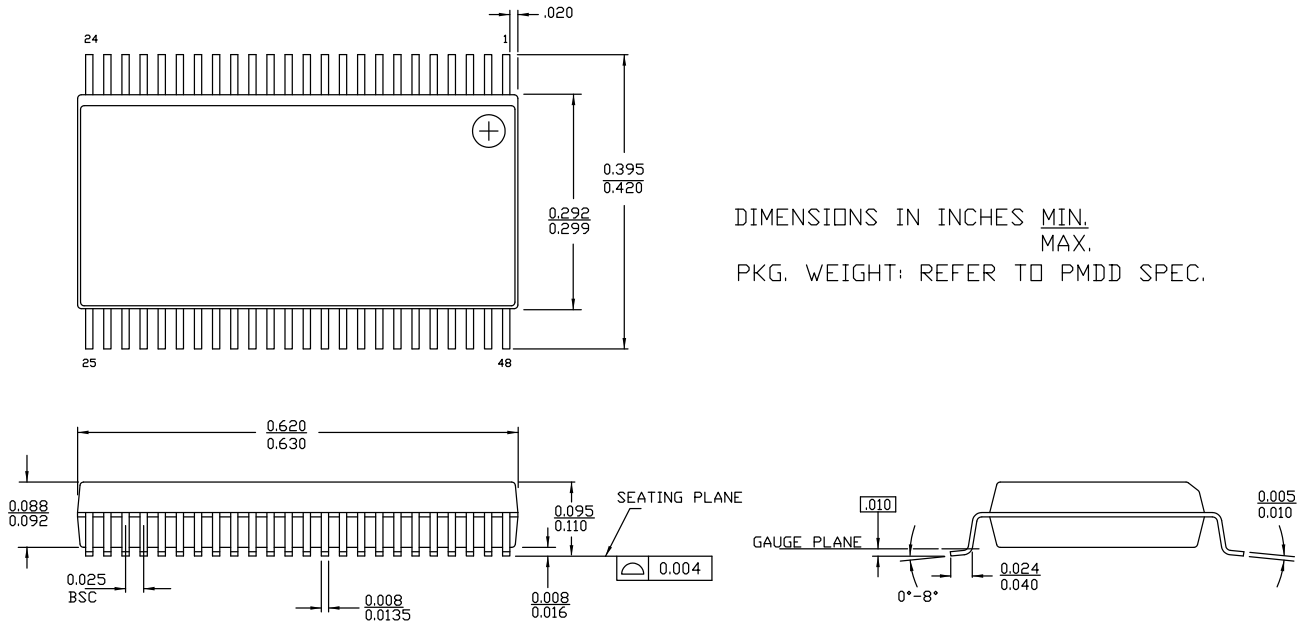
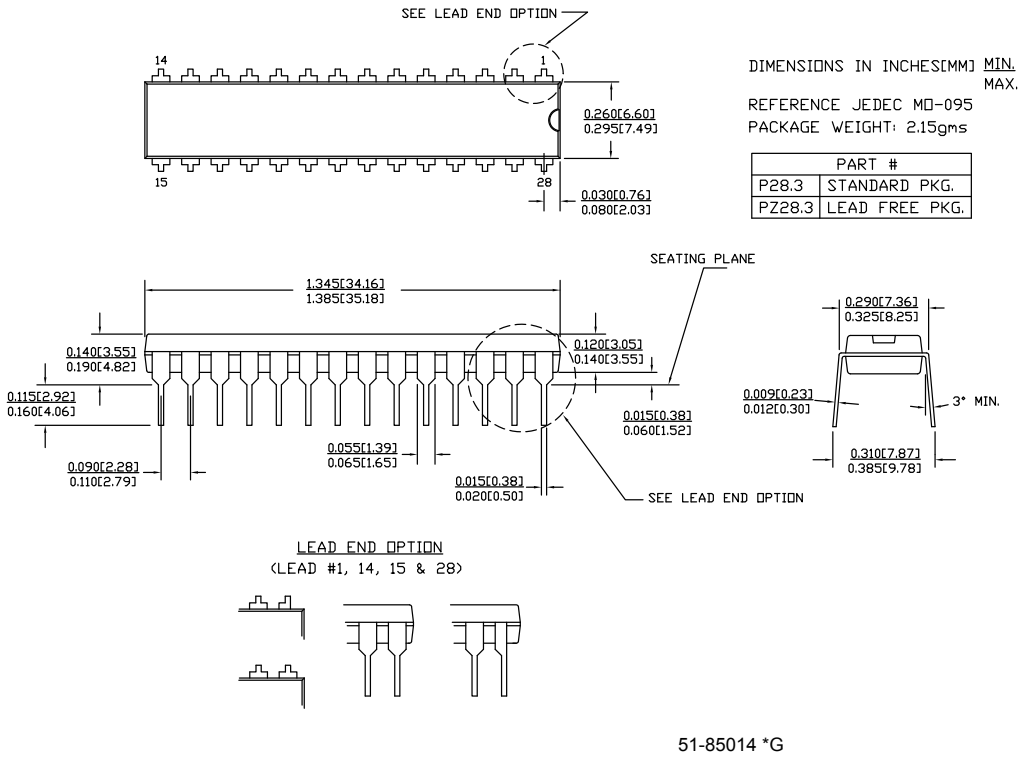


Figure 15. 28-pin PDIP (300 Mils) Package Outline, 51-85014



Acronyms

Acronym	Description
ADC	Analog-to-Digital Converter
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processing Unit
DAC	Digital-to-Analog Converter
EMI	Electromagnetic Interference
FBGA	Fine-Pitch Ball Grid Array
GPIO	General-Purpose Input/Output
HAPI	Hardware Assisted Parallel Interface
LED	Light-Emitting Diode
LSB	Least Significant Bit
MSB	Most Significant Bit
I/O	Input/Output
$\overline{OE}$	Output Enable
PCB	Printed Circuit Board
PDIP	Plastic Dual In-line Package
PLL	Phase-Locked Loop
POR	Power-On Reset
PROM	Programmable Read-Only Memory
RAM	Random Access Memory
SIE	Serial Interface Engine
SOIC	Small-Outline Integrated Circuit
SSOP	Shrink Small-Outline Package
USB	Universal Serial Bus
WDT	Watchdog Timer

Document Conventions

Units of Measure

Symbol	Unit of Measure
cm	centimeter
°C	degree Celsius
kHz	kilohertz
kΩ	kilohm
MHz	megahertz
µA	microampere
µs	microsecond
mA	milliampere
mm	millimeter
ms	millisecond
mW	milliwatt
ns	nanosecond
Ω	ohm
%	percent
pF	picofarad
V	volt
W	watt

Document History Page

Document Title: CY7C64013C/CY7C64113C, Full-Speed USB (12-Mbps) Function Document Number: 38-08001				
Rev.	**ECN No.**	**Issue Date**	**Orig. of Change**	**Description of Change**
**	109962	12/16/01	SZV	Change from Spec number: 38-00626 to 38-08001
*A	129715	02/05/04	MON	Added register bit definitions Added default bit state of each register Corrected the Schematic (location of the Pull up on D+) Added register summary Modified tables 19-1 and 19-2 Provided more explanation regarding locking/unlocking mechanism of the mode register.
*B	429099	See ECN	TYJ	Changed part numbers to the 'C' types. Included 'Cypress Perform' logo. Updated part numbers in the Ordering section.
*C	2897159	03/22/10	XUT	Removed inactive parts CY7C64013C-PXC and CY7C64113C-PVXC from the Ordering information table. Updated package diagrams.
*D	3190495	03/08/2011	NXZ	Added Ordering Code Definitions. Updated Package Diagrams. Added Acronyms and Units of Measure. Updated in new template.
*E	4349221	04/16/2014	DEJO	Updated Package Diagrams: spec 51-85061 – Changed revision from *D to *F. spec 51-85014 – Changed revision from *E to *G. spec 51-85026 – Changed revision from *F to *H. Updated in new template. Completing Sunset Review.

Sales, Solutions, and Legal Information

Worldwide Sales and Design Support

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

Products

Automotive	cypress.com/go/automotive
Clocks & Buffers	cypress.com/go/clocks
Interface	cypress.com/go/interface
Lighting & Power Control	cypress.com/go/powerpsoc cypress.com/go/plc
Memory	cypress.com/go/memory
PSoC	cypress.com/go/psoc
Touch Sensing	cypress.com/go/touch
USB Controllers	cypress.com/go/USB
Wireless/RF	cypress.com/go/wireless

PSoC® Solutions

psoc.cypress.com/solutions

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

Cypress Developer Community

Community | Forums | Blogs | Video | Training

Technical Support

cypress.com/go/support