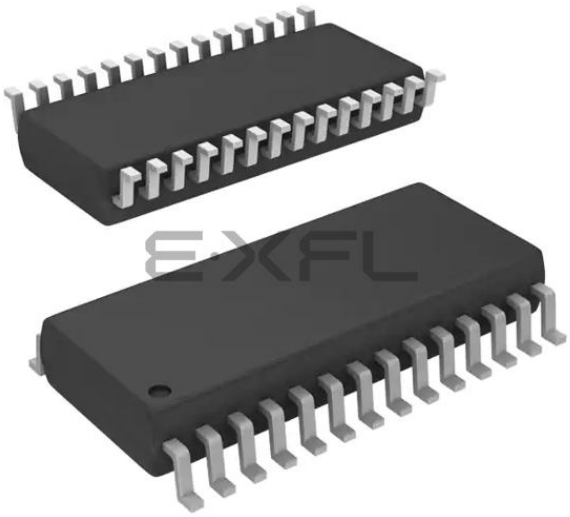


Welcome to E-XFL.COM

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-sxct

Product Summary Tables

Pin Assignments

Table 1. Pin Assignments

Name	I/O	28-pin SOIC	28-pin PDIP	48-pin SSOP	Description
D+[0], D–[0]	I/O	6, 7	7, 8	7, 8	Upstream port, USB differential data.
P0	I/O	P0[7:0] 10, 14, 11, 15, 12, 16, 13, 17	P0[7:0] 11, 15, 12, 16, 13, 17, 14, 18	P0[7:0] 20, 26, 21, 27, 22, 28, 23, 29	GPIO Port 0 capable of sinking 7 mA (typical).
P1	I/O	P1[2:0] 25, 27, 26	P1[2:0] 26, 4, 27	P1[7:0] 6, 43, 5, 44, 4, 45, 47, 46	GPIO Port 1 capable of sinking 7 mA (typical).
P2	I/O	P2[6:2] 19, 9, 20, 8, 21	P2[6:2] 20, 10, 21, 9, 23	P2[7:0] 18, 32, 17, 33, 15, 35, 14, 36	GPIO Port 2 capable of sinking 7 mA (typical). HAPI is also supported through P2[6:2].
P3	I/O	P3[2:0] 23, 5, 24	P3[2:0] 24, 6, 25	P3[7:0] 13, 37, 12, 39, 10, 41, 7, 42	GPIO Port 3, capable of sinking 12 mA (typical).
DAC	I/O			DAC[7,2:0] 19, 25, 24, 31	DAC Port with programmable current sink outputs. DAC[1:0] offer a programmable range of 3.2 to 16 mA typical. DAC[7,2] have a programmable sink current range of 0.2 to 1.0 mA typical.
$XTAL_{IN}$	IN	2	2	2	6-MHz crystal or external clock input.
$XTAL_{OUT}$	OUT	1	1	1	6-MHz crystal out.
V_{PP}	IN	18	19	30	Programming voltage supply, tie to ground during normal operation.
V_{CC}	IN	28	28	48	Voltage supply.
GND	IN	4, 22	5, 22	11, 16, 34, 40	Ground.
V_{REF}	IN	3	3	3	External 3.3 V supply voltage for the differential data output buffers and the D+ pull-up.
NC				38	No Connect.

I/O Register Summary

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

All undefined registers are reserved. It is important not to write to reserved registers as this may cause an undefined operation or increased current consumption during operation. When writing to registers with reserved bits, the reserved bits must be written with ‘0.’

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

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.

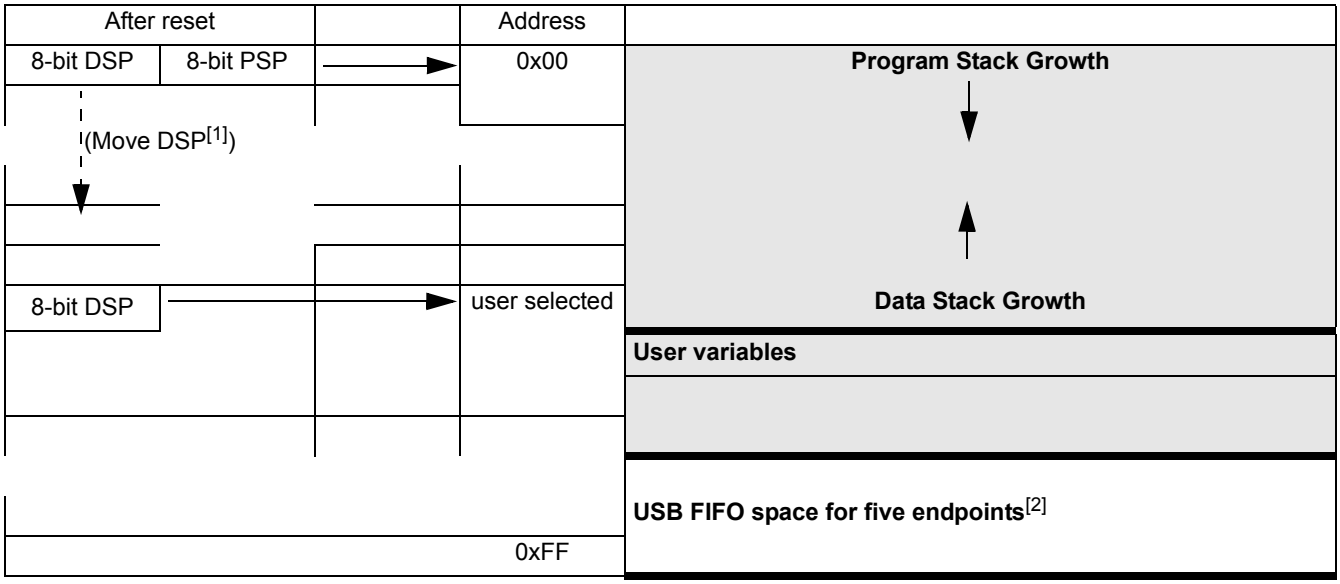
The Return from Interrupt (RETI) instruction decrements the PSP, then restores the second byte from memory addressed by the PSP. The PSP is decremented again and the first byte is restored from memory addressed by the PSP. After the program counter and flags have been restored from stack, the interrupts 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.

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 pre-decrements the DSP, then writes data to the memory location addressed by the DSP. A POP instruction reads data from the memory location addressed by the DSP, then post-increments the DSP.

During a reset, the DSP is reset to 0x00. A PUSH instruction when DSP equals 0x00 writes data at the top of the data RAM (address 0xFF). This writes data to the memory area reserved for USB endpoint FIFOs. Therefore, the DSP should be indexed at an appropriate memory location that does not compromise the Program Stack, user-defined memory (variables), or the USB endpoint FIFOs.

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 described in USB Device Endpoints on page 32. Example assembly instructions to do this with two device addresses (FIFOs begin at 0xD8) are shown below:

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

Address Modes

The CY7C64013C and CY7C64113C microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

Data (Immediate)

“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 0xD8:

■ MOV A,0D8h

This instruction requires 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 is the constant “0xD8.” A constant may be referred to by name if a prior “EQU” statement assigns the constant value to the name. For example, the following code is equivalent to the example shown above:

■ DSPINIT: EQU 0D8h

■ MOV A,DSPINIT

Direct

“Direct” address mode is used when the data operand is a variable stored in SRAM. In that case, the one byte address of the variable is encoded in the instruction. As an example, consider an instruction that loads A with the contents of memory address location 0x10:

■ MOV A,[10h]

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

■ buttons: EQU 10h

■ MOV A,[buttons]

Indexed

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

■ array: EQU 10h

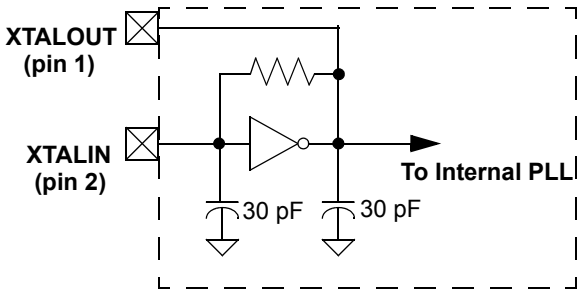
■ MOV X,3

■ MOV A,[X+array]

This would have the effect of loading A with the fourth element of the SRAM “array” that begins at address 0x10. The fourth element would be at address 0x13.

Clocking

Figure 1. Clock Oscillator On-Chip Circuit



The XTALIN and XTALOUT are the clock pins to the microcontroller. The user can connect an external oscillator or a crystal to these pins. When using an external crystal, keep PCB traces between the chip leads and crystal as short as possible (less than 2 cm). A 6-MHz fundamental frequency parallel resonant crystal can be connected to these pins to provide a reference frequency for the internal PLL. The two internal 30-pF load caps appear in series to the external crystal and would be equivalent to a 15-pF load. Therefore, the crystal must have a required load capacitance of about 15–18 pF. A ceramic

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.

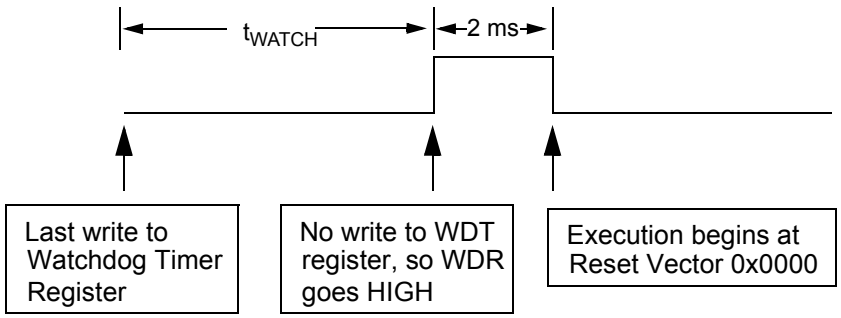
If a USB Bus Reset occurs on the upstream port during the 95-ms semi-suspend time, the semi-suspend state is aborted and program execution begins immediately from address 0x0000. In this case, the Bus Reset interrupt is pending but not serviced until firmware sets the USB Bus Reset Interrupt Enable bit (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 5. Port 1 Data

Port 1 Data								ADDRESS 0x01
Bit #	7	6	5	4	3	2	1	0
Bit Name	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Table 6. Port 2 Data

Port 2 Data								ADDRESS 0x02
Bit #	7	6	5	4	3	2	1	0
Bit Name	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Table 7. Port 3 Data

Port 3 Data								ADDRESS 0x03
Bit #	7	6	5	4	3	2	1	0
Bit Name	P3.7	P3.6	P3.5	P3.4	P3.3	P32	P3.1	P3.0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Special care should be taken with any unused GPIO data bits. An unused GPIO data bit, either a pin on the chip or a port bit that is not bonded on a particular package, must not be left floating when the device enters the suspend state. If a GPIO data bit is left floating, the leakage current caused by the floating bit may violate the suspend current limitation specified by the USB Specifications. If a ‘1’ is written to the unused data bit and the port is configured with open drain outputs, the unused data bit remains in an indeterminate state. Therefore, if an unused port bit is programmed in open-drain mode, it must be written with a ‘0.’ Notice that the CY7C64013C part always requires that the data bits P1[7:3], P2[7,1,0], and P3[7:3] be written with a ‘0.’

In normal non-HAPI mode, reads from a GPIO port always return the present state of the voltage at the pin, independent of the settings in the Port Data Registers. If HAPI mode is activated for a port, reads of that port return latched data as controlled by the HAPI signals (see Hardware Assisted Parallel Interface (HAPI) on page 24). During reset, all of the GPIO pins are set to a high-impedance input state (‘1’ in open drain mode). Writing a ‘0’ to a GPIO pin drives the pin LOW. In this state, a ‘0’ is always read on that GPIO pin unless an external source overdrives the internal pull-down device.

GPIO Configuration Port

Every GPIO port can be programmed as inputs with internal pull-ups, outputs LOW or HIGH, or Hi-Z (floating, the pin is not driven internally). In addition, the interrupt polarity for each port can be programmed. The Port Configuration bits (Table 8) and the Interrupt Enable bit (Table 10 on page 17 through Table 13 on page 18) determine the interrupt polarity of the port pins.

Table 8. GPIO Configuration Register

GPIO Configuration								ADDRESS 0x08
Bit #	7	6	5	4	3	2	1	0
Bit Name	Port 3 Config Bit 1	Port 3 Config Bit 0	Port 2 Config Bit 1	Port 2 Config Bit 0	Port 1 Config Bit 1	Port 1 Config Bit 0	Port 0 Config Bit 1	Port 0 Config Bit 0
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

As shown in Table 9 on page 17 below, a positive polarity on an input pin represents a rising edge interrupt (LOW to HIGH), and a negative polarity on an input pin represents a falling edge interrupt (HIGH to LOW).

The GPIO interrupt is generated when all of the following conditions are met: the Interrupt Enable bit of the associated Port Interrupt Enable Register is enabled, the GPIO Interrupt Enable bit of the Global Interrupt Enable Register (Table 28 on page 26) is enabled, the Interrupt Enable Sense (bit 2, Table 27 on page 25) is set, and the GPIO pin of the port sees an event matching the interrupt polarity.

Table 12. Port 2 Interrupt Enable

Port 2 Interrupt Enable								ADDRESS 0x06
Bit #	7	6	5	4	3	2	1	0
Bit Name	P2.7 Intr Enable	P2.6 Intr Enable	P2.5 Intr Enable	P2.4 Intr Enable	P2.3 Intr Enable	P2.2 Intr Enable	P2.1 Intr Enable	P2.0 Intr Enable
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

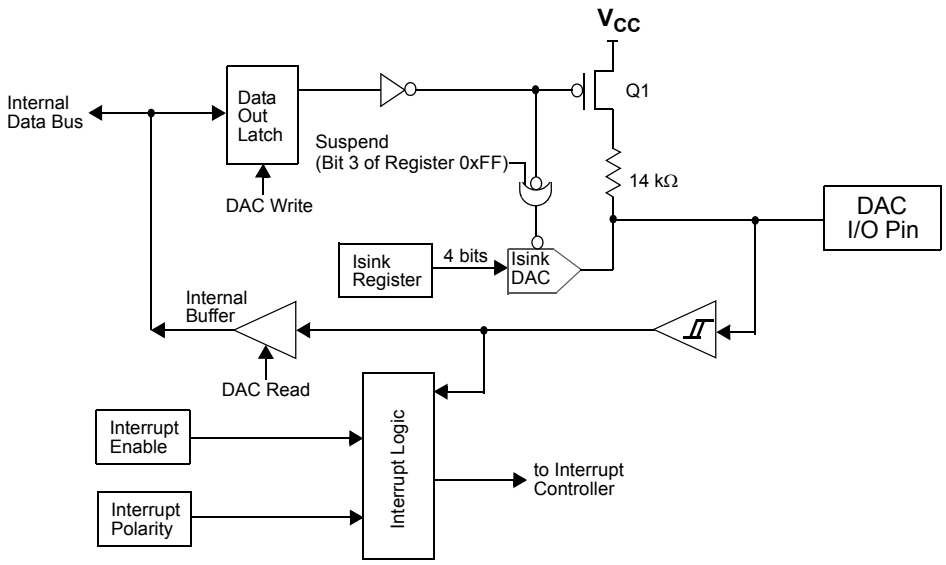
Table 13. Port 3 Interrupt Enable

Port 3 Interrupt Enable								ADDRESS 0x07
Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved (Set to 0)	P3.6 Intr Enable	P3.5 Intr Enable	P3.4 Intr Enable	P3.3 Intr Enable	P3.2 Intr Enable	P3.1 Intr Enable	P3.0 Intr Enable
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

DAC Port

The CY7C64113C features a programmable current sink 4 bit port which is also known as a DAC port. Each of these port I/O pins have a programmable current sink. Writing a ‘1’ to a DAC I/O pin disables the output current sink (Isink DAC) and drives the I/O pin HIGH through an integrated 14-kΩ resistor. When a ‘0’ is written to a DAC I/O pin, the Isink DAC is enabled and the pull-up resistor is disabled. This causes the I_{sink} DAC to sink current to drive the output LOW. Figure 4 shows a block diagram of the DAC port pin.

Figure 4. Block Diagram of a DAC Pin



The amount of sink current for the DAC I/O pin is programmable over 16 values based on the contents of the DAC Isink Register for that output pin. DAC[1:0] are high-current outputs that are programmable from 3.2 mA to 16 mA (typical). DAC[7:2] are low-current outputs, programmable from 0.2 mA to 1.0 mA (typical).

When the suspend bit in Processor Status and Control Register (see Table 27 on page 25) is set, the Isink DAC block of the DAC circuitry is disabled. Special care should be taken when the CY7C64x13C device is placed in the suspend mode. The DAC Port Data Register (see Table 14) should normally be loaded with all '1's (0xFF) before setting the suspend bit. If any of the DAC bits are set to '0' when the device is suspended, that DAC input will float. The floating pin could result in excessive current consumption by the device, unless an external load places the pin in a deterministic state.

Table 14. DAC Port Data

DAC Port Data								ADDRESS 0x30
Bit #	7	6	5	4	3	2	1	0
Bit Name	DAC[7]	Reserved	Reserved	Reserved	Reserved	DAC[2]	DAC[1]	DAC[0]
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bit [1..0]: High Current Output 3.2 mA to 16 mA typical

1= I/O pin is an output pulled HGH through the 14-kΩ resistor.

0 = I/O pin is an input with an internal 14-kΩ pull-up resistor

Bit [3..2]: Low Current Output 0.2 mA to 1 mA typical

1= I/O pin is an output pulled HGH through the 14-kΩ resistor.

0 = I/O pin is an input with an internal 14-kΩ pull-up resistor

DAC Isink Registers

Each DAC I/O pin has an associated DAC Isink register to program the output sink current when the output is driven LOW. The first Isink register (0x38) controls the current for DAC[0], the second (0x39) for DAC[1], and so on until the Isink register at 0x3F controls the current to DAC[7].

Table 15. DAC Sink Register

DAC Sink Register								ADDRESS 0x38 -0x3F
Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Isink[3]	Isink[2]	Isink[1]	Isink[0]
Read/Write					W	W	W	W
Reset	-	-	-	-	0	0	0	0

Bit [4..0]: Isink [x] (x= 0..4)

Writing all ‘0’s to the Isink register causes 1/5 of the max current to flow through the DAC I/O pin. Writing all ‘1’s to the Isink register provides the maximum current flow through the pin. The other 14 states of the DAC sink current are evenly spaced between these two values.

Bit [7..5]: Reserved

DAC Port Interrupts

A DAC port interrupt can be enabled/disabled for each pin individually. The DAC Port Interrupt Enable register provides this feature with an interrupt enable bit for each DAC I/O pin.All of the DAC Port Interrupt Enable register bits are cleared to ‘0’ during a reset. All DAC pins share a common interrupt, as explained in DAC Interrupt on page 29.

Table 16. DAC Port Interrupt Enable

DAC Port Interrupt							ADDRESS 0x31	
Bit #	7	6	5	4	3	2	1	0
Bit Name	Enable Bit 7	Reserved	Reserved	Reserved	Reserved	Enable Bit 2	Enable Bit 1	Enable Bit 0
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit [7..0]: Enable bit x (x= 0..2, 7)

1= Enables interrupts from the corresponding bit position;

0= Disables interrupts from the corresponding bit position

As an additional benefit, the interrupt polarity for each DAC pin is programmable with the DAC Port Interrupt Polarity register. Writing a ‘0’ to a bit selects negative polarity (falling edge) that causes an interrupt (if enabled) if a falling edge transition occurs on the corresponding input pin. Writing a ‘1’ to a bit in this register selects positive polarity (rising edge) that causes an interrupt (if enabled) if a rising edge transition occurs on the corresponding input pin. All of the DAC Port Interrupt Polarity register bits are cleared during a reset.

Table 17. DAC Port Interrupt Polarity

DAC Port Interrupt Polarity							ADDRESS 0x32	
Bit #	7	6	5	4	3	2	1	0
Bit Name	Enable Bit 7	Reserved	Reserved	Reserved	Reserved	Enable Bit 2	Enable Bit 1	Enable Bit 0
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Bit [7..0]: Enable bit x (x= 0..2, 7)

1= Selects positive polarity (rising edge) that causes an interrupt (if enabled);

0= Selects negative polarity (falling edge) that causes an interrupt (if enabled)

12-Bit Free-Running Timer

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

Table 18. Timer LSB Register

Timer LSB							ADDRESS 0x24
Bit #	7	6	5	4	3	2	1	0
Bit Name	Timer Bit 7	Timer Bit 6	Timer Bit 5	Timer Bit 4	Timer Bit 3	Timer Bit 2	Timer Bit 1	Timer Bit 0
Read/Write	R	R	R	R	R	R	R	R
Reset	0	0	0	0	0	0	0	0

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.

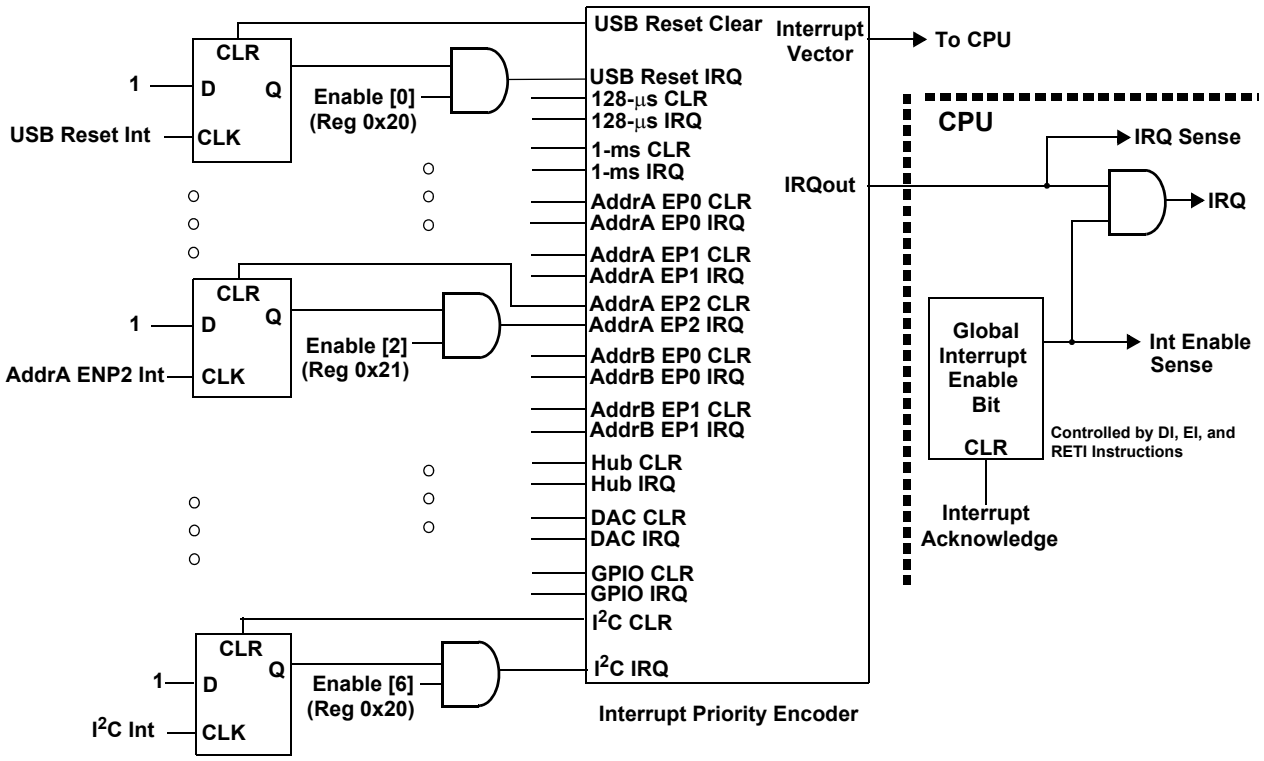
The Program Counter value as well as the Carry and Zero flags (CF, ZF) are stored onto the Program Stack by the automatic CALL instruction generated as part of the interrupt acknowledge process. The user firmware is responsible for ensuring that the processor state is preserved and restored during an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

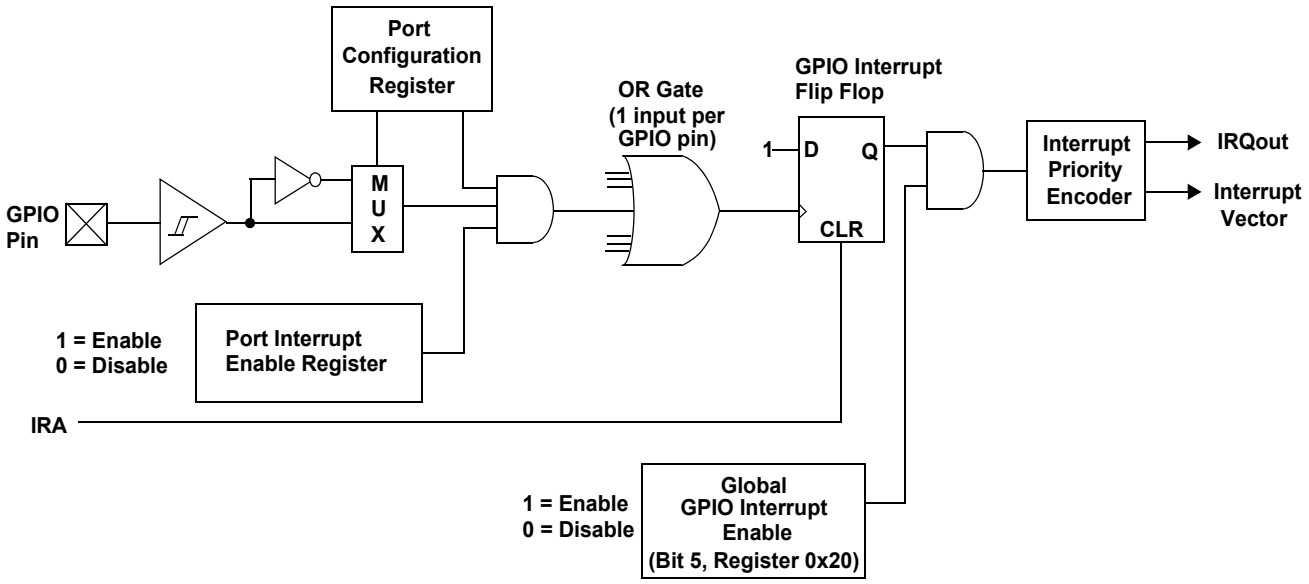
The Interrupt Vectors supported by the USB Controller are listed in Table 30 on page 29. The lowest-numbered interrupt (USB Bus Reset interrupt) has the highest priority, and the highest-numbered interrupt (I^2C interrupt) has the lowest priority.

Figure 7. Interrupt Controller Function Diagram



Although Reset is not an interrupt, the first instruction executed after a reset is at PROM address 0x0000h—which corresponds to the first entry in the Interrupt Vector Table. Because the JMP instruction is two bytes long, the interrupt vectors occupy two bytes.

Figure 8. GPIO Interrupt Structure



When HAPI is enabled, the HAPI logic takes over the interrupt vector and blocks any interrupt from the GPIO bits, including ports/bits not being used by HAPI. Operation of the HAPI interrupt is independent of the GPIO specific bit interrupt enables, and is enabled or disabled only by bit 5 of the Global Interrupt Enable Register (Table 28 on page 26) when HAPI is enabled. The settings of the GPIO bit interrupt enables on ports/bits not used by HAPI still effect the CMOS mode operation of those ports/bits. The effect of modifying the interrupt bits while the Port Config bits are set to “10” is shown in Table 9. The events that generate HAPI interrupts are described in Hardware Assisted Parallel Interface (HAPI) on page 24.

I²C Interrupt

The I²C interrupt occurs after various events on the I²C-compatible bus to signal the need for firmware interaction. This generally involves reading the I²C Status and Control Register (Table 24 on page 22) to determine the cause of the interrupt, loading/reading the I²C Data Register as appropriate, and finally writing the Status and Control Register to initiate the subsequent transaction. The interrupt indicates that status bits are stable and it is safe to read and write the I²C registers. Refer to I2C-compatible Controller on page 22 for details on the I²C registers.

When enabled, the I²C-compatible state machines generate interrupts on completion of the following conditions. The referenced bits are in the I²C Status and Control Register.

1. In **slave receive** mode, after the slave receives a byte of data: The *Addr* bit is set, if this is the first byte since a start or restart signal was sent by the external master. Firmware must read or write the data register as necessary, then set the *ACK, Xmit MODE,* and *Continue/Busy* bits appropriately for the next byte.
2. In **slave receive** mode, after a stop bit is detected: The *Received Stop* bit is set, if the stop bit follows a slave receive transaction where the *ACK* bit was cleared to 0, no stop bit detection occurs.
3. In **slave transmit** mode, after the slave transmits a byte of data: The *ACK* bit indicates if the master that requested the byte acknowledged the byte. If more bytes are to be sent, firmware writes the next byte into the Data Register and then sets the *Xmit MODE* and *Continue/Busy* bits as required.
4. In **master transmit** mode, after the master sends a byte of data. Firmware should load the Data Register if necessary, and set the *Xmit MODE, MSTR MODE*, and *Continue/Busy* bits appropriately. Clearing the *MSTR MODE* bit issues a stop signal to the I²C-compatible bus and return to the idle state.
5. In **master receive** mode, after the master receives a byte of data: Firmware should read the data and set the *ACK* and *Continue/Busy* bits appropriately for the next byte. Clearing the *MSTR MODE* bit at the same time causes the master state machine to issue a stop signal to the I²C-compatible bus and leave the I²C-compatible hardware in the idle state.
6. When the master loses arbitration: This condition clears the *MSTR MODE* bit and sets the *ARB Lost/Restart* bit immediately and then waits for a stop signal on the I²C-compatible bus to generate the interrupt.

The *Continue/Busy* bit is cleared by hardware prior to interrupt conditions 1 to 4. Once the Data Register has been read or written, firmware should configure the other control bits and set the *Continue/Busy* bit for subsequent transactions. Following an interrupt from master mode, firmware should perform only one write to the Status and Control Register that sets the *Continue/Busy* bit, without checking the value of the *Continue/Busy* bit. The Busy bit may otherwise be active and I²C register contents may be changed by the hardware during the transaction, until the I²C interrupt occurs.

USB Overview

The USB hardware consists of the logic for a full-speed USB Port. The full-speed serial interface engine (SIE) interfaces the microcontroller to the USB bus. An external series resistor (R_{ext}) must be placed in series with the D+ and D– lines, as close to

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) *(continued)*

Status In/extra Out																			
0	1	1	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	0	0	1	1	Stall	yes
0	1	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	1	NoChange				TX 0	yes
CONTROL READ																				
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 In/premature status Out																				
1	1	1	1	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
1	1	1	1	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	1	1	1	0	ACK (back)	yes
Nak In/premature status Out																				
1	1	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
1	1	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	0	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				NAK	yes
Status Out/extra In																				
0	0	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
0	0	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
0	0	1	0	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
0	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	1	UC	UC	NoChange				ignore	no
0	0	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	0	0	1	1	Stall	yes

OUT ENDPOINT																				
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**

Package Diagrams

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

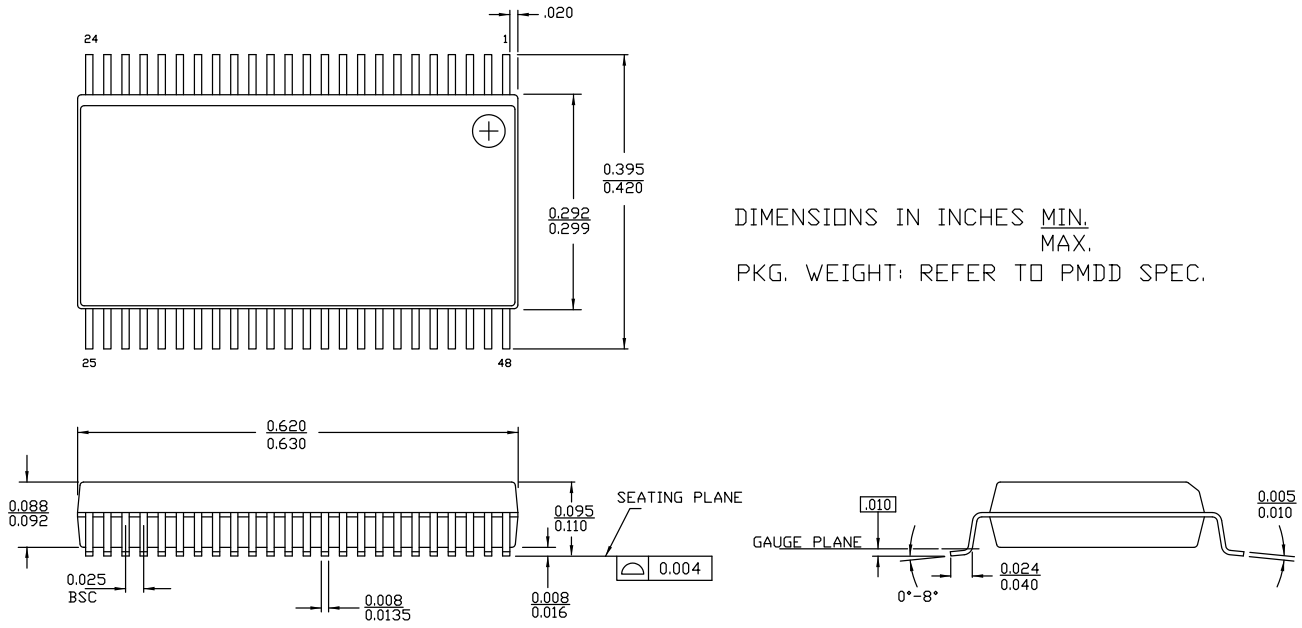
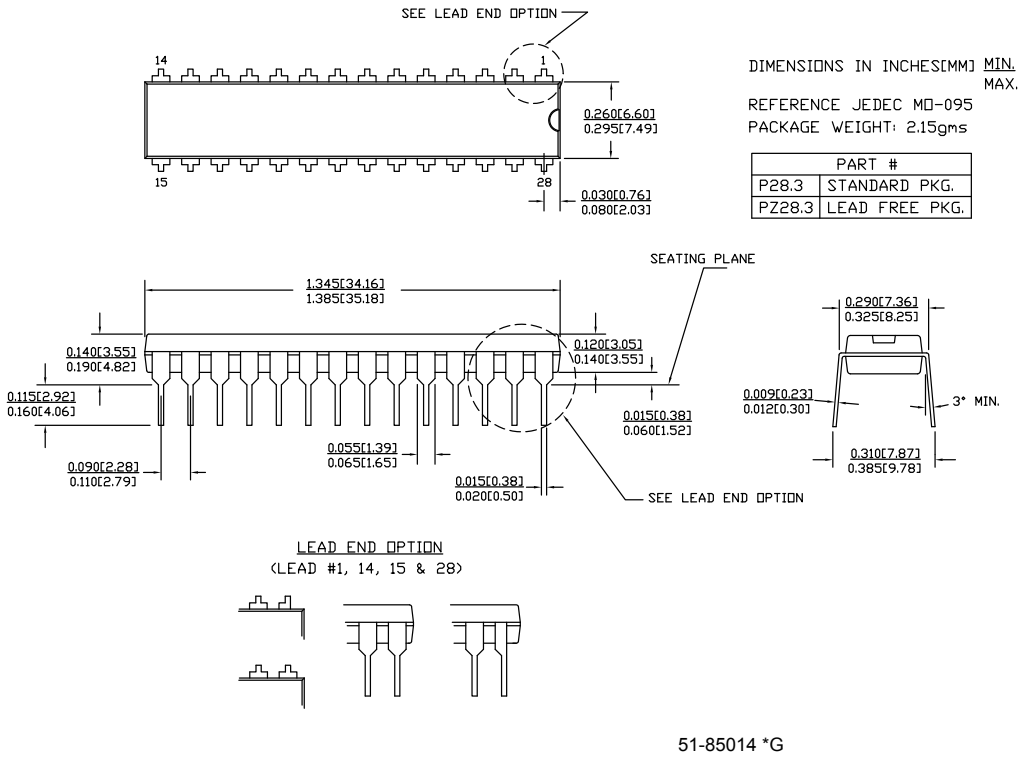


Figure 15. 28-pin PDIP (300 Mils) Package Outline, 51-85014



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.