Welcome to E-XFL.COM

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

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

What Are Embedded - Microcontrollers - Application Specific?

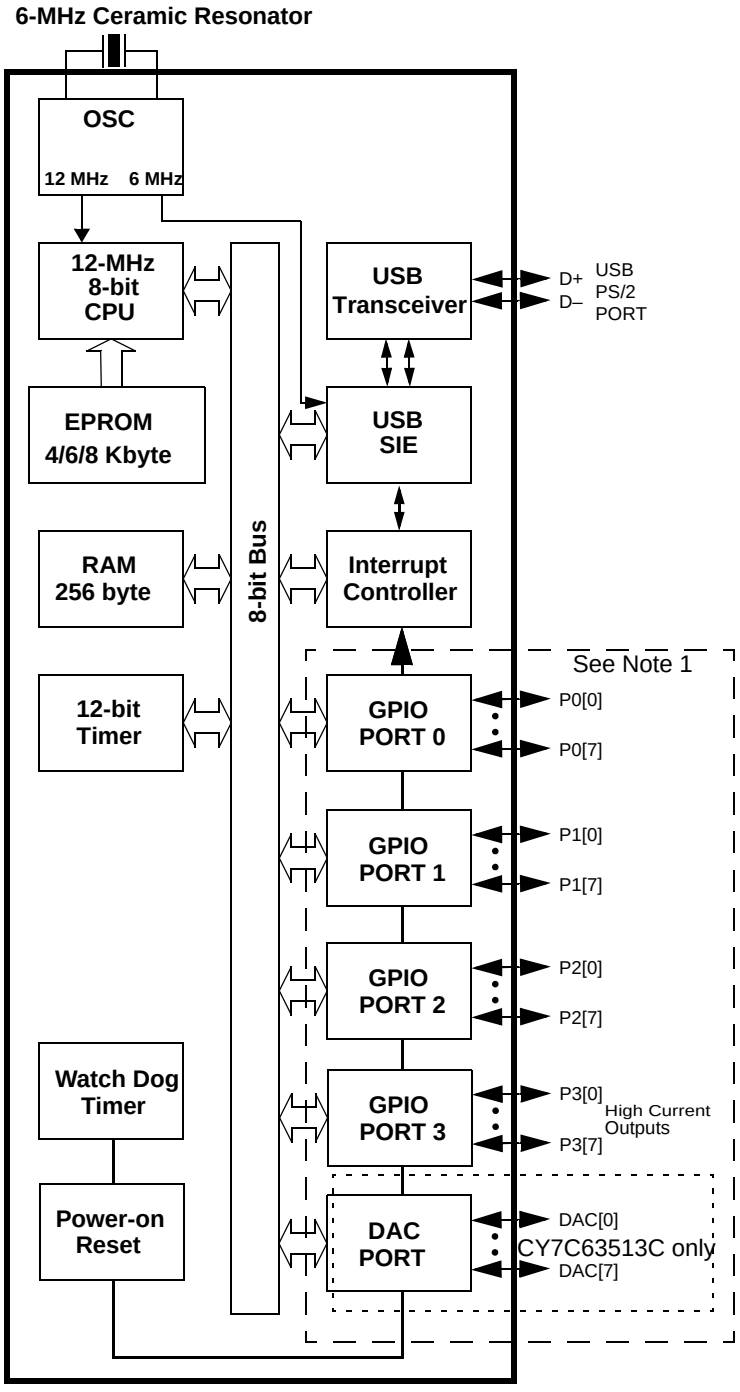
Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Microcontroller
Core Processor	M8B
Program Memory Type	OTP (8kB)
Controller Series	CY7C635xx
RAM Size	256 x 8
Interface	PS/2, USB
Number of I/O	40
Voltage - Supply	4V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	48-BSSOP (0.295", 7.50mm Width)
Supplier Device Package	48-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c63513c-pvxc

Contents

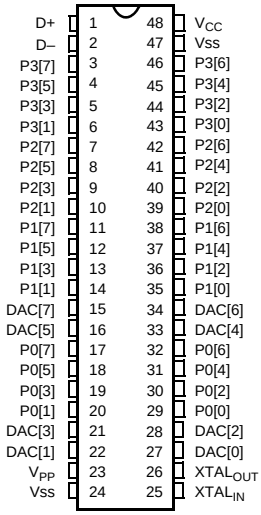
.

Logic Block Diagram

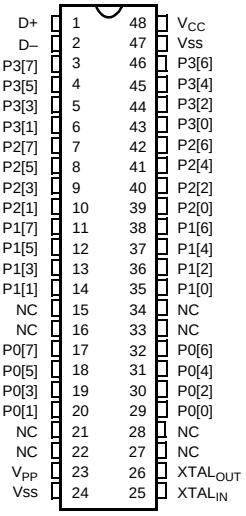


Pin Configuration

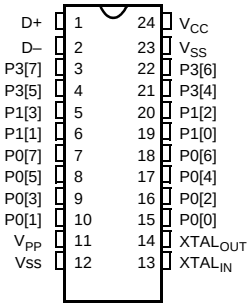
CY7C63513C 48-pin SSOP



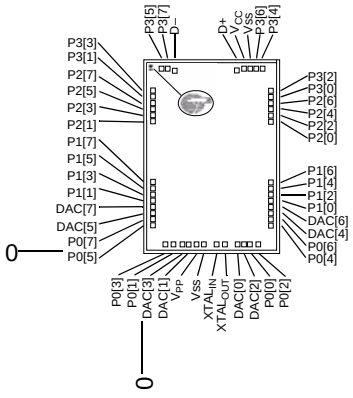
CY7C63413C 48-pin SSOP



CY7C63613C 24-pin SOIC



CY7C63413C 48-Pad Die



Note
1. CY7C63613C is not bonded out for all GPIO pins shown in Logic Block Diagram. Refer to pin configuration diagram for bonded out pins. See note on page 12 for firmware code needed for unused GPIO pins.

PSP,A instruction. The PSP supports interrupt service under hardware control and CALL, RET, and RETI instructions under firmware control.

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

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

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

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

8-bit Data Stack Pointer (DSP)

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

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

Address Modes

The CY7C63413C/513C/613C microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

Data

The “Data” address mode refers to a data operand that is actually a constant encoded in the instruction. As an example, consider the instruction that loads A with the constant 0xE8:

■ MOV A,0E8h

This instruction will require two bytes of code where the first byte identifies the “MOV A” instruction with a data operand as the second byte. The second byte of the instruction will be the constant “0xE8”. 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 0E8h

■ 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]

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

■ buttons: EQU 10h

■ MOV A,[buttons]

Indexed

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

■ array: EQU 10h

■ MOV X,3

■ MOV A,[x+array]

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

Data Memory Organization

The CY7C63413C/513C/613C microcontrollers provide 256 bytes of data RAM. In normal usage, the SRAM is partitioned into four areas: program stack, data stack, user variables and USB endpoint FIFOs as shown below:

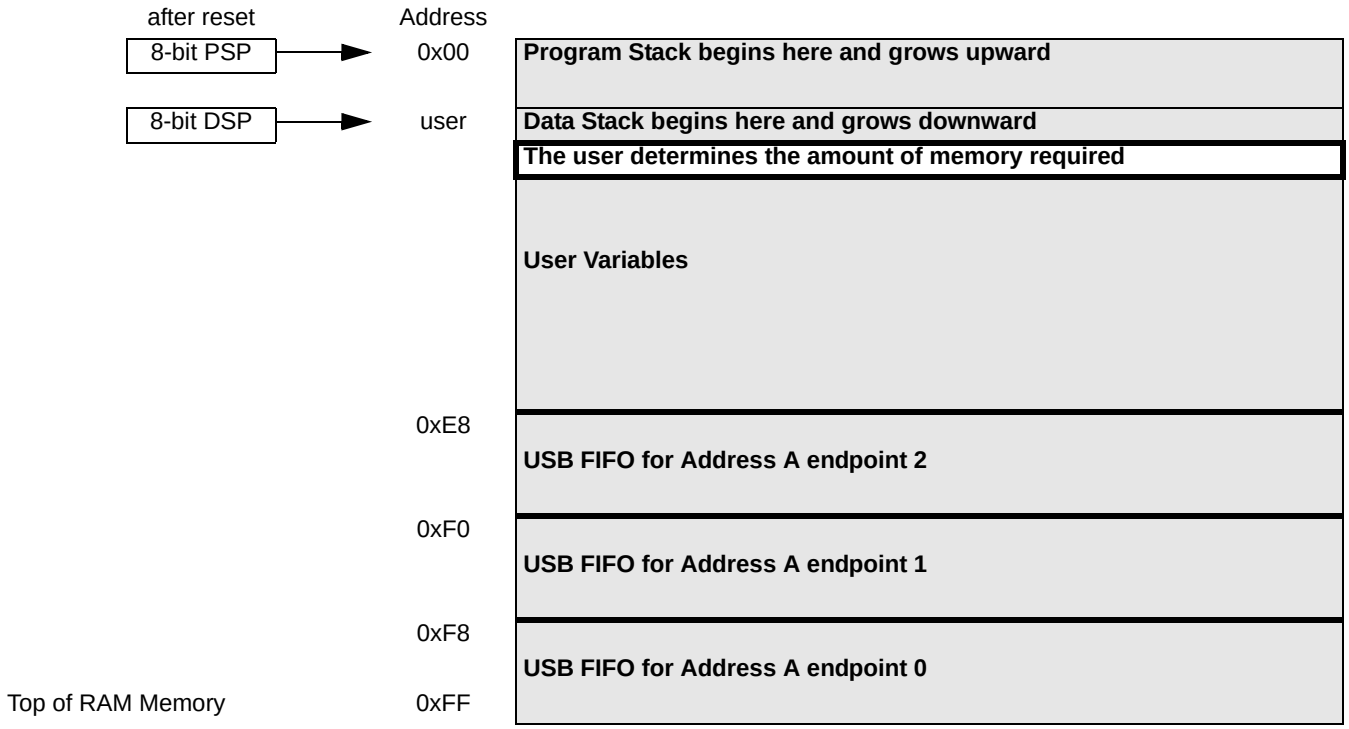
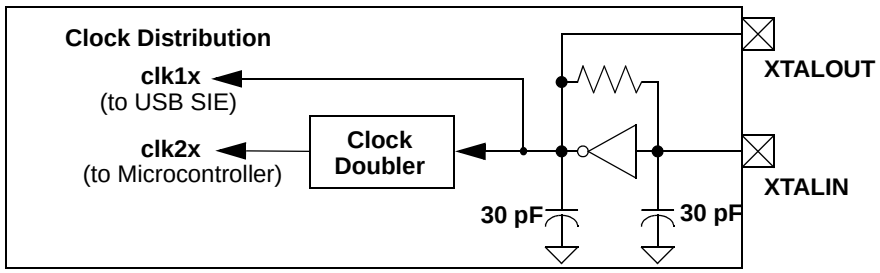


Figure 2. Clock Oscillator On-chip Circuit



Clocking

The $XTAL_{IN}$ and $XTAL_{OUT}$ are the clock pins to the microcontroller. The user can connect a low-cost ceramic resonator or an external oscillator can be connected to these pins to provide a reference frequency for the internal clock distribution and clock doubler.

An external 6-MHz clock can be applied to the $XTAL_{IN}$ pin if the $XTAL_{OUT}$ pin is left open. Please note that grounding the $XTAL_{OUT}$ pin is not permissible as the internal clock is effectively shorted to ground.

Reset

The USB Controller supports three types of resets. All registers are restored to their default states during a reset. The USB Device Addresses are set to 0 and all interrupts are disabled. In addition, the program stack pointer (PSP) and data stack pointer (DSP) are set to 0x00. For USB applications, the firmware should set the DSP below 0xE8 to avoid a memory conflict with RAM dedicated to USB FIFOs. The assembly instructions to do this are shown below:

```
Mov A, E8h     ; Move 0xE8 hex into Accumulator
Swap A,dsp     ; Swap accumulator value into dsp register
```

The three reset types are:

1. Power-On Reset (POR)
2. Watch Dog Reset (WDR)
3. USB Bus Reset (non hardware reset)

The occurrence of a reset is recorded in the Processor Status and Control Register located at I/O address 0xFF. Bits 4, 5, and 6 are used to record the occurrence of POR, USB Reset, and WDR respectively. The firmware can interrogate these bits to determine the cause of a reset.

The microcontroller begins execution from ROM address 0x0000 after a POR or WDR reset. Although this looks like interrupt vector 0, there is an important difference. Reset processing does NOT push the program counter, carry flag, and zero flag onto program stack. That means the reset handler in firmware should initialize the hardware and begin executing the “main” loop of code. Attempting to execute either a RET or RETI in the reset handler will cause unpredictable execution results.

Power-On Reset (POR)

Power-On Reset (POR) occurs every time the V_{CC} voltage to the device ramps from 0V to an internally defined trip voltage (Vrst) of approximately 1/2 full supply voltage. In addition to the normal reset initialization noted under “Reset,” bit 4 (PORS) of the Processor Status and Control Register is set to “1” to indicate to the firmware that a Power-On Reset occurred. The POR event forces the GPIO ports into input mode (high impedance), and the state of Port 3 bit 7 is used to control how the part will respond after the POR releases.

If Port 3 bit 7 is HIGH (pulled to V_{CC}) and the USB I/O are at the idle state (DM HIGH and DP LOW) the part will go into a semi-permanent power down/suspend mode, waiting for the USB I/O to go to one of Bus Reset, K (resume) or SE0. If Port 3 bit 7 is still HIGH when the part comes out of suspend, then a 128-µs timer starts, delaying CPU operation until the ceramic resonator has stabilized.

If Port 3 bit 7 was LOW (pulled to V_{SS}) the part will start a 96-ms timer, delaying CPU operation until V_{CC} has stabilized, then continuing to run as reset.

Firmware should clear the POR Status (PORS) bit in register 0xFF before going into suspend as this status bit selects the 128-µs or 96-ms start-up timer value as follows: IF Port 3 bit 7 is HIGH then 128-µs is always used; ELSE if PORS is HIGH then 96-ms is used; ELSE 128-µs is used.

Watch Dog Reset (WDR)

The Watch Dog Timer Reset (WDR) occurs when the Most Significant Bit (MSB) of the 2-bit Watch Dog Timer Register transitions from LOW to HIGH. In addition to the normal reset initialization noted under “Reset,” bit 6 of the Processor Status and Control Register is set to “1” to indicate to the firmware that a Watch Dog Reset occurred.

The Watch Dog Timer is a 2-bit timer clocked by a 4.096-ms clock (bit 11) from the free-running timer. Writing any value to the write-only Watch Dog Clear I/O port (0x26) will clear the Watch Dog Timer.

In some applications, the Watch Dog Timer may be cleared in the 1.024-ms timer interrupt service routine. If the 1.024-ms timer interrupt service routine does not get executed for 8.192 ms or more, a Watch Dog Timer Reset will occur. A Watch Dog Timer Reset lasts for 2.048 ms after which the microcontroller begins execution at ROM address 0x0000. The USB transmitter is disabled by a Watch Dog Reset because the USB Device Address Register is cleared. Otherwise, the USB Controller would respond to all address 0 transactions. The USB transmitter remains disabled until the MSB of the USB address register is set.

General Purpose I/O Ports

Ports 0 to 2 provide 24 GPI/O pins that can be read or written. Each port (8 bits) can be configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs. Please note an open drain output is also a high-impedance (no pull-up) input. All of the I/O pins within a given port have the same configuration. Ports 0 to 2 are considered low current drive with typical current sink capability of 7 mA.

The internal pull-up resistors are typically 7 kΩ. Two factors govern the enabling and disabling of the internal pull-up resistors: the port configuration selected in the GPI/O Configuration register and the state of the output data bit. If the GPI/O Configuration selected is “Resistive” and the output data bit is “1,” then the internal pull-up resistor is enabled for that GPI/O pin. Otherwise, Q1 is turned off and the 7-kΩ pull-up is disabled. Q2 is “ON” to sink current whenever the output data bit is written as a “0.” Q3 provides “HIGH” source current when the GPI/O port is configured for CMOS outputs and the output data bit is written as a “1”. Q2 and Q3 are sized to sink and source, respectively, roughly the same amount of current to support traditional CMOS outputs with symmetric drive.

Figure 3. Block Diagram of a GPIO Line

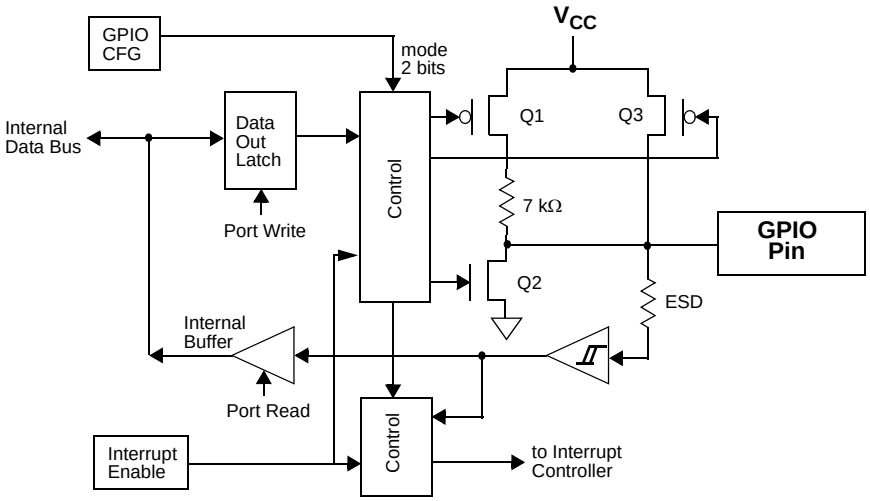


Table 2. Port 0 Data

Addr: 0x00		Port 0 Data					
P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 3. Port 1 Data

Addr: 0x01		Port 1 Data					
P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 7. Port 0 Interrupt Enable

Addr: 0x04		Port 0 Interrupt Enable					
P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]
W	W	W	W	W	W	W	W

Table 8. Port 1 Interrupt Enable

Addr: 0x05		Port 1 Interrupt Enable					
P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]
W	W	W	W	W	W	W	W

Table 9. Port 2 Interrupt Enable

Addr: 0x06		Port 2 Interrupt Enable					
P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]
W	W	W	W	W	W	W	W

Table 10. Port 3 Interrupt Enable

Addr: 0x07		Port 3 Interrupt Enable					
P3[7]	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]
W	W	W	W	W	W	W	W

Table 11. Possible Port Configurations

Port Configuration bits	Pin Interrupt Bit	Driver Mode	Interrupt Polarity
11	X	Resistive	–
10	0	CMOS Output	disabled
10	1	Open Drain	disabled
01	X	Open Drain	–
00	X	Open Drain	+ (default)

In “Resistive” mode, a 7-kΩ pull-up resistor is conditionally enabled for all pins of a GPIO port. The resistor is enabled for any pin that has been written as a “1.” The resistor is disabled on any pin that has been written as a “0.” An I/O pin will be driven high through a 7-kΩ pull-up resistor when a “1” has been written to the pin. Or the output pin will be driven LOW, with the pull-up disabled, when a “0” has been written to the pin. An I/O pin that has been written as a “1” can be used as an input pin with an integrated 7-kΩ pull-up resistor. Resistive mode selects a negative (falling edge) interrupt polarity on all pins that have the GPIO interrupt enabled.

In “CMOS” mode, all pins of the GPIO port are outputs that are actively driven. The current source and sink capacity are roughly the same (symmetric output drive). A CMOS port is not a possible source for interrupts.

A port configured in CMOS mode has interrupt generation disabled, yet the interrupt mask bits serve to control port direction. If a port’s associated Interrupt Mask bits are cleared, those port bits are strictly outputs. If the Interrupt Mask bits are set then those bits will be open drain inputs. As open drain inputs, if their data output values are ‘1’ those port pins will be CMOS inputs (HIGH Z output).

In “Open Drain” mode the internal pull-up resistor and CMOS driver (HIGH) are both disabled. An I/O pin that has been written as a “1” can be used as either a high-impedance input or a three-state output. An I/O pin that has been written as a “0” will drive the output LOW. The interrupt polarity for an open drain GPIO port can be selected as either positive (rising edge) or negative (falling edge).

During reset, all of the bits in the GPIO Configuration Register are written with “0.” This selects the default configuration: Open Drain output, positive interrupt polarity for all GPIO ports.

Table 12. GPIO Configuration Register

Addr: 0x08		GPIO Configuration Register					
7	6	5	4	3	2	1	0
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
W	W	W	W	W	W	W	W

Table 14. DAC Port Interrupt Enable

Addr: 0x31		DAC Port Interrupt Enable					
DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]
W	W	W	W	W	W	W	W

Table 15. DAC Port Interrupt Polarity

Addr: 0x32		DAC Port Interrupt Polarity					
DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]
W	W	W	W	W	W	W	W

Table 16. DAC Port Isink

Addr: 0x38-0x3F		DAC Port Interrupt Polarity					
Reserved				Isink Value			
				Isink[3]	Isink[2]	Isink[1]	Isink[0]
				W	W	W	W

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

Control Bits	Control Action
000	Not forcing (SIE controls driver)
001	Force K (D+ HIGH, D– LOW)
010	Force J (D+ LOW, D– HIGH)
011	Force SE0 (D+ LOW, D– LOW)
100	Force SE0 (D– LOW, D+ LOW)
101	Force D– LOW, D+ HiZ
110	Force D– HiZ, D+ LOW
111	Force D– HiZ, D+ HiZ

USB Device

USB Device Address A includes three endpoints: EPA0, EPA1, and EPA2. End Point 0 (EPA0) allows the USB host to recognize, set up, and control the device. In particular, EPA0 is used to receive and transmit control (including set-up) packets.

USB Ports

The USB Controller provides one USB device address with three endpoints. The USB Device Address Register contents are cleared during a reset, setting the USB device address to zero and marking this address as disabled. Figure 18 shows the format of the USB Address Register.

Bit 7 (Device Address Enable) in the USB Device Address Register must be set by firmware before the serial interface engine (SIE) will respond to USB traffic to this address. The Device Address in bits [6:0] must be set by firmware during the USB enumeration process to an address assigned by the USB host that does not equal zero. This register is cleared by a hardware reset or the USB bus reset.

Device Endpoints (3)

The USB controller communicates with the host using dedicated FIFOs, one per endpoint. Each endpoint FIFO is implemented as 8 bytes of dedicated SRAM. There are three endpoints defined for Device “A” that are labeled “EPA0,” “EPA1,” and EPA2.”

All USB devices are required to have an endpoint number 0 (EPA0) that is used to initialize and control the USB device. End Point 0 provides access to the device configuration information and allows generic USB status and control accesses. End Point 0 is bidirectional as the USB controller can both receive and transmit data.

The endpoint mode registers are cleared during reset. The EPA0 endpoint mode register uses the format shown in Table 19.

Bits[7:5] in the endpoint 0 mode registers (EPA0) are “sticky” status bits that are set by the SIE to report the type of token that was most recently received. The sticky bits must be cleared by firmware as part of the USB processing.

The endpoint mode registers for EPA1 and EPA2 do not use bits [7:5] as shown in Table 20.

Table 18. USB Device Address Register

Addr:0x10		USB Device Address Register					
Device Address Enable	Device Address Bit 6	Device Address Bit 5	Device Address Bit 4	Device Address Bit 3	Device Address Bit 2	Device Address Bit 1	Device Address Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 19. USB Device EPA0, Mode Register

Addr:0x12		USB Device EPA0, Mode Register					
Endpoint 0 Set-up Received	Endpoint 0 In Received	Endpoint 0 Out Received	Acknowledge	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 20. USB Device Endpoint Mode Register

Addr: 0x14, 0x16		USB Device Endpoint Mode Register					
Reserved	Reserved	Reserved	Acknowledge	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

The ‘Acknowledge’ bit is set whenever the SIE engages in a transaction that completes with an ‘ACK’ packet.

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

Bits[6:0] of the endpoint 0 mode register are locked from CPU IOWR operations only if the SIE has updated one of these bits, which the SIE does only at the end of a packet transaction (set-up... Data... ACK, or Out... Data... ACK, or In... Data... ACK). The CPU can unlock these bits by doing a subsequent I/O read of this register.

Firmware must do an IORD after an IOWR to an endpoint 0 register to verify that the contents have changed and that the SIE has not updated these values.

While the ‘set-up’ bit is set, the CPU cannot write to the DMA buffers at memory locations 0xE0 through 0xE7 and 0xF8 through 0xFF. This prevents an incoming set-up transaction from conflicting with a previous In data buffer filling operation by firmware.

The mode bits (bits [3:0]) in an Endpoint Mode Register control how the endpoint responds to USB bus traffic. The mode bit encoding is shown in Section .

The format of the endpoint Device counter registers is shown in Table 21.

Bits 0 to 3 indicate the number of data bytes to be transmitted during an IN packet, valid values are 0 to 8 inclusive. Data Valid bit 6 is used for OUT and set-up tokens only. Data 0/1 Toggle bit 7 selects the DATA packet’s toggle state: 0 for DATA0, 1 for DATA1.

Table 21. USB Device Counter Registers

Addr: 0x11, 0x13, 0x15		USB Device Counter Registers					
Data 0/1 Toggle	Data Valid	Reserved	Reserved	Byte count Bit 3	Byte count Bit 2	Byte count Bit 1	Byte count Bit 0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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 actually reading 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.

Timer (LSB)

Table 22. Timer Register

Addr: 0x24		Timer Register (LSB)					
Timer Bit 7	Timer Bit 6	Timer Bit 5	Timer Bit 4	Timer Bit 3	Timer Bit 2	Timer Bit 1	Timer Bit 0
R	R	R	R	R	R	R	R

Timer (MSB)

Table 23. Timer Register

Addr: 0x25		Timer Register (MSB)					
Reserved	Reserved	Reserved	Reserved	Timer Bit 11	Timer Bit 10	Timer Bit 9	Timer Bit 8
				R	R	R	R

Figure 6. Timer Block Diagram

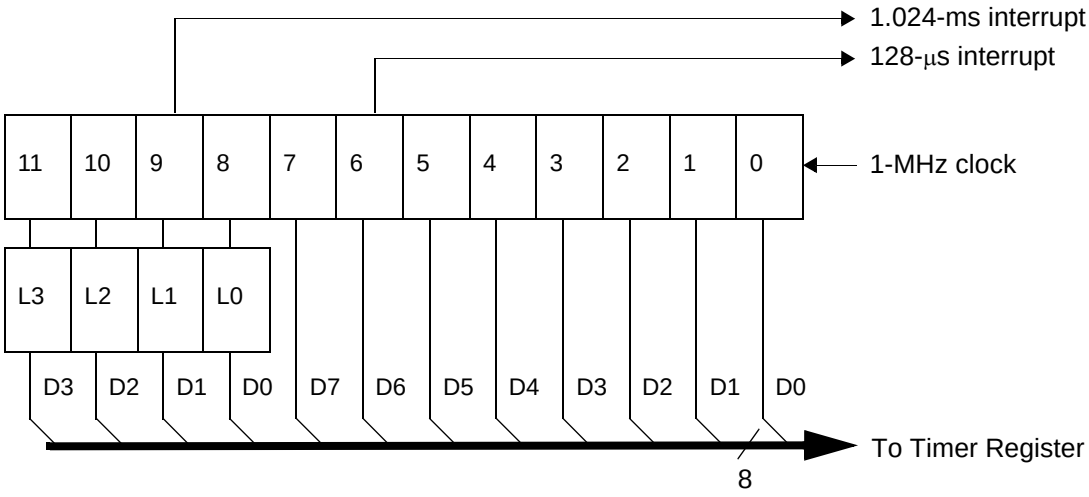


Table 25. USB End Point Interrupt Enable Register

Addr: 0x21		USB End Point Interrupt Enable Register					
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Reserved	EPA2 Interrupt Enable	EPA1 Interrupt Enable	EPA0 Interrupt Enable
					R/W	R/W	R/W

Interrupt Vectors

The Interrupt Vectors supported by the USB Controller are listed in Table 26. Although Reset is not an interrupt, per se, the first instruction executed after a reset is at PROM address 0x0000—which corresponds to the first entry in the Interrupt Vector Table. Because the JMP instruction is 2 bytes long, the interrupt vectors occupy 2 bytes.

Table 26. 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	Reserved
8	0x0010	Reserved
9	0x0012	Reserved
10	0x0014	DAC interrupt
11	0x0016	GPIO interrupt
12	0x0018	Reserved

Interrupt Latency

Interrupt latency can be calculated from the following equation:

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

For example, if a 5 clock cycle instruction such as JC is being executed when an interrupt occurs, the first instruction of the Interrupt Service Routine will execute a min. of 16 clocks (1+10+5) or a max. of 20 clocks (5+10+5) after the interrupt is issued. Remember that the interrupt latches are sampled at the rising edge of the last clock cycle in the current instruction.

USB Bus Reset Interrupt

The USB Bus Reset interrupt is asserted when a USB bus reset condition is detected. A USB bus reset is indicated by a single ended zero (SE0) on the upstream port for more than 8 microseconds.

Timer Interrupt

There are two 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 interrupts first or the suspend request first.

USB Endpoint Interrupts

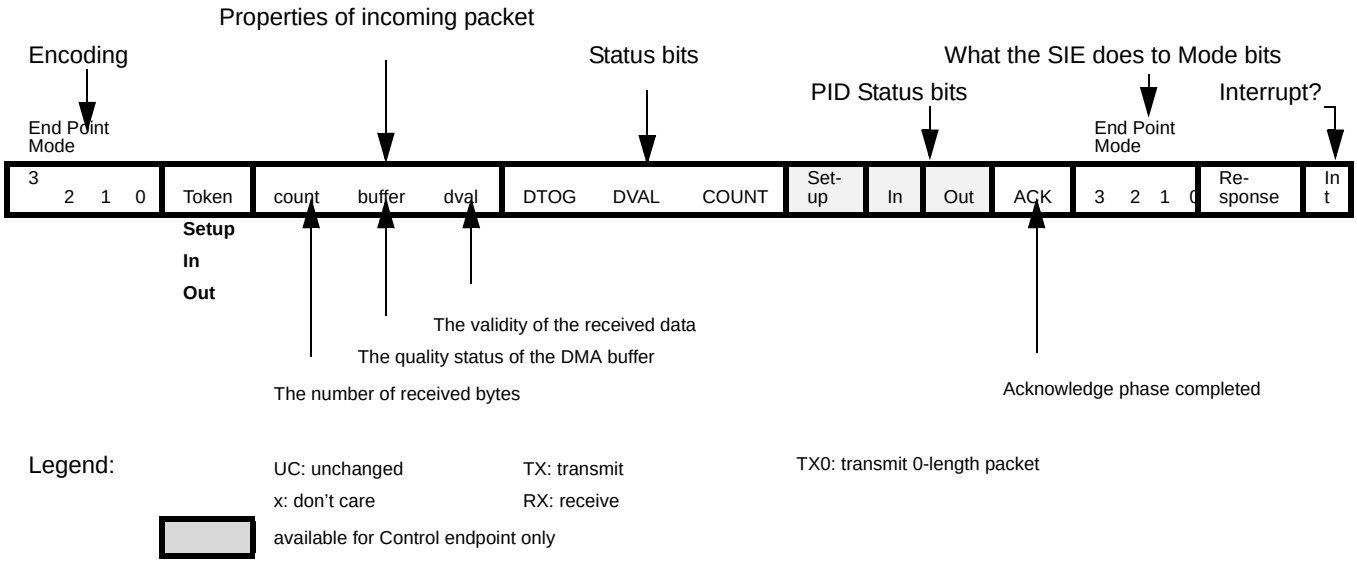
There are three USB endpoint interrupts, one per endpoint. The USB endpoints interrupt after the either the USB host or the USB controller sends a packet to the USB.

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 will need to read the DAC port to determine which pin or pins caused an interrupt.

Please note that if one DAC pin 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

Figure 7. Decode table for Table 28: "Details of Modes for Differing Traffic Conditions"



The response of the SIE can be summarized as follows:

1. the SIE will only respond to valid transactions, and will ignore non-valid ones;
2. the SIE will generate IRQ when a valid transaction is completed or when the DMA buffer is corrupted
3. an incoming Data packet is valid if the count is <= 10 (CRC inclusive) and passes all error checking;
4. a Setup will be ignored by all non-Control endpoints (in appropriate modes);
5. an In will be ignored by an Out configured endpoint and vice versa.

The In and Out PID status is updated at the end of a transaction.

The Setup PID status is updated at the beginning of the Data packet phase.

The entire EndPoint 0 mode and the Count register are locked to CPU writes at the end of any transaction in which an ACK is transferred. These registers are only unlocked upon a CPU read of these registers, and only if that read happens after the transaction completes. This represents about a 1-µs window to which to the CPU is locked from register writes to these USB registers. Normally the firmware does a register read at the beginning of the ISR to unlock and get the mode register information. The interlock on the Mode and Count registers ensures that the firmware recognizes the changes that the SIE might have made during the previous transaction.

Table 28. Details of Modes for Differing Traffic Conditions

End Point Mode											PID				Set End Point Mode					
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	Setup	In	Out	ACK	3	2	1	0	response	int
Setup Packet (if accepting)																				
See Table 27				Setup	<= 10	data	valid	updates	1	updates	1	UC	UC	1	0	0	0	1	ACK	yes
See Table 27				Setup	> 10	junk	x	updates	updates	updates	1	UC	UC	UC	NoChange				ignore	yes
See Table 27				Setup	x	junk	invalid	updates	0	updates	1	UC	UC	UC	NoChange				ignore	yes
Disabled																				
0	0	0	0	x	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
Nak In/Out																				
0	0	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes
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																				
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
Status In/extra Out																				
0	1	1	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	0	0	1	1	Stall	yes
0	1	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	1	NoChange				TX 0	yes
Control Read																				
Normal In/premature status Out																				
1	1	1	1	Out	2	UC	valid	1	1	updates	UC	UC	1	1	NoChange				ACK	yes
1	1	1	1	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
1	1	1	1	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	1	1	1	0	ACK (back)	yes
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

Table 28. Details of Modes for Differing Traffic Conditions (continued)

0	0	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes

End Point Mode											PID				Set End Point Mode					
3	**2**	**1**	**0**	**token**	**count**	**buffer**	**dval**	**DTOG**	**DVAL**	**COUNT**	**Setup**	**In**	**Out**	**ACK**	**3**	**2**	**1**	**0**	**response**	**int**
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	UC	UC	UC	UC	ignore	no
0	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	UC	UC	UC	UC	ignore	no
0	0	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	0	0	1	1	Stall	yes
Out endpoint																				
Normal Out/erroneous In																				
1	0	0	1	Out	<= 10	data	valid	updates	1	updates	UC	UC	1	1	1	0	0	0	ACK	yes
1	0	0	1	Out	> 10	junk	x	updates	updates	updates	UC	UC	1	UC	NoChange				ignore	yes
1	0	0	1	Out	x	junk	invalid	updates	0	updates	UC	UC	1	UC	NoChange				ignore	yes
1	0	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
NAK Out/erroneous In																				
1	0	0	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes
1	0	0	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	0	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	0	0	0	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
Isochronous endpoint (Out)																				
0	1	0	1	Out	x	updates	updates	updates	updates	updates	UC	UC	1	1	NoChange				RX	yes
0	1	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
In endpoint																				
Normal In/erroneous Out																				
1	1	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	0	1	In	x	UC	x	UC	UC	UC	UC	1	UC	1	1	1	0	0	ACK (back)	yes
NAK In/erroneous Out																				
1	1	0	0	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
1	1	0	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				NAK	yes
Isochronous endpoint (In)																				
0	1	1	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
0	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	NoChange				TX	yes

DC Characteristics Fosc = 6 MHz; operating temperature = 0 to 70 °C (continued)

	Parameter	Min	Max	Unit	Conditions
V_H	Input hysteresis voltage	6%	12%	V_{CC}	All ports, HIGH to LOW edge
I_{ol}	Sink current	7.2	16.5	mA	Port 3, Vout = 1.0 V[9]
I_{ol}	Sink current	3.5	10.6	mA	Port 0,1,2, Vout = 2.0 V [9]
I_{oh}	Source current	1.4	7.5	mA	Voh = 2.4 V (all ports 0,1,2,3)[9]
DAC Interface					
R_{up}	Pull-up resistance	8.0 K	20.0 K	Ohms	Note16
$I_{sink0(0)}$	DAC[7:2] sink current (0)[17]	0.1	0.3	mA	Vout = 2.0 VDC[10]
$I_{sink0(F)}$	DAC[7:2] sink current (F)[17]	0.5	1.5	mA	Vout = 2.0 DC[10]
$I_{sink1(0)}$	DAC[1:0] sink current (0)[17]	1.6	4.8	mA	Vout = 2.0 VDC[10]
$I_{sink1(F)}$	DAC[1:0] sink current (F)[17]	8	24	mA	Vout = 2.0 VDC[10]
I_{range}	Programmed Isink ratio: max/min	4	6	–	Vout = 2.0 VDC[10,14]
I_{lin}	Differential nonlinearity	–	0.5	lsb	Any pin[12]
t_{sink}	Current sink response time	–	0.8	µs	Full scale transition
T_{ratio}	Tracking ratio DAC[1:0] to DAC[7:2]	14	21	–	Vout = 2.0 V[13]

Switching Characteristics

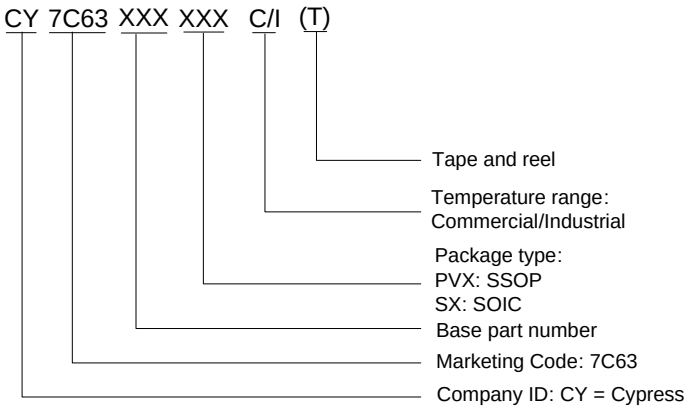
Parameter	Description	Min	Max	Unit	Conditions
Clock					
t_{CYC}	Input Clock Cycle Time	165.0	168.3	ns	–
t_{CH}	Clock HIGH Time	0.45 t_{CYC}	–	ns	–
t_{CL}	Clock LOW Time	0.45 t_{CYC}	–	ns	–
USB Driver Characteristics					
t_r	Transition Rise Time	75	–	ns	CLoad = 50 pF[10, 11]
t_r	Transition Rise Time	–	300	ns	CLoad = 600 pF [10, 11]
t_f	Transition Fall Time	75	–	ns	CLoad = 50 pF[10, 11]
t_f	Transition Fall Time	–	300	ns	CLoad = 600 pF[10, 11]
t_{rfm}	Rise/Fall Time Matching	80	125	%	t_r/t_f[10, 11]
V_{crs}	Output Signal Crossover Voltage	1.3	2.0	V	Notes 10 and 11
USB Data Timing					
t_{drate}	Low Speed Data Rate	1.4775	1.5225	Mbs	Ave. Bit Rate (1.5 Mb/s ± 1.5%)
t_{djr1}	Receiver Data Jitter Tolerance	–75	75	ns	To Next Transition[15]
t_{djr2}	Receiver Data Jitter Tolerance	–45	45	ns	For Paired Transitions[15]
t_{deop}	Differential to EOP Transition Skew	–40	100	ns	Note 18
t_{eopr1}	EOP Width at Receiver	330	–	ns	Rejects as EOP[15]
t_{eopr2}	EOP Width at Receiver	675	–	ns	Accepts as EOP[15]
t_{eopt}	Source EOP Width	1.25	1.50	µs	–
t_{udj1}	Differential Driver Jitter	–95	95	ns	To next transition, Figure 12 on page 30.
t_{udj2}	Differential Driver Jitter	–150	150	ns	To paired transition, Figure 12 on page 30.

Notes
9. Functionality is guaranteed of the $V_{CC\ (1)}$ range, except USB transmitter and DACs.
10. USB transmitter functionality is guaranteed over the $V_{CC\ (2)}$ range, as well as DAC outputs.
11. Per Table 7-7 of revision 1.1 of USB specification, for C_{LOAD} of 50–600 pF.
12. Measured as largest step size vs. nominal according to measured full scale and zero programmed values.
13. T_{ratio} = Isink1[1:0](n)/Isink0[7:2](n) for the same n, programmed.
14. Irange: Isinkn(15)/ Isinkn(0) for the same pin.
15. Measured at crossover point of differential data signals.
16. Limits total bus capacitance loading (C_{LOAD}) to 400 pF per section 7.1.5 of revision 1.1 of USB specification.
17. DAC I/O Port not bonded out on CY7C63613C. See note on page 12 for firmware code needed for unused pins.
18. Total current cumulative across all Port pins flowing to V_{SS} is limited to minimize Ground-Drop noise effects.

Ordering Information

Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C63413C-PVXC	8 KB	SP48	48-pin Shrunk Small Outline Package	Commercial
CY7C63413C-PVXCT	8 KB	SP48	48-pin SSOP Pb-free Tape-reel	Commercial
CY7C63513C-PVXC	8 KB	SP48	48-pin SSOP Pb-free	Commercial
CY7C63613C-SXC	8 KB	SZ24.3	24-pin (300 mil) SOIC Pb-free	Commercial
CY7C63613C-SXCT	8 KB	SZ24.3	24-pin (300 mil) SOIC Pb-free Tape-reel	Commercial
CY7C63513C-PVXCT	8 KB	SP48	48-pin SSOP Pb-free Tape-reel	Commercial

Ordering Code Definition



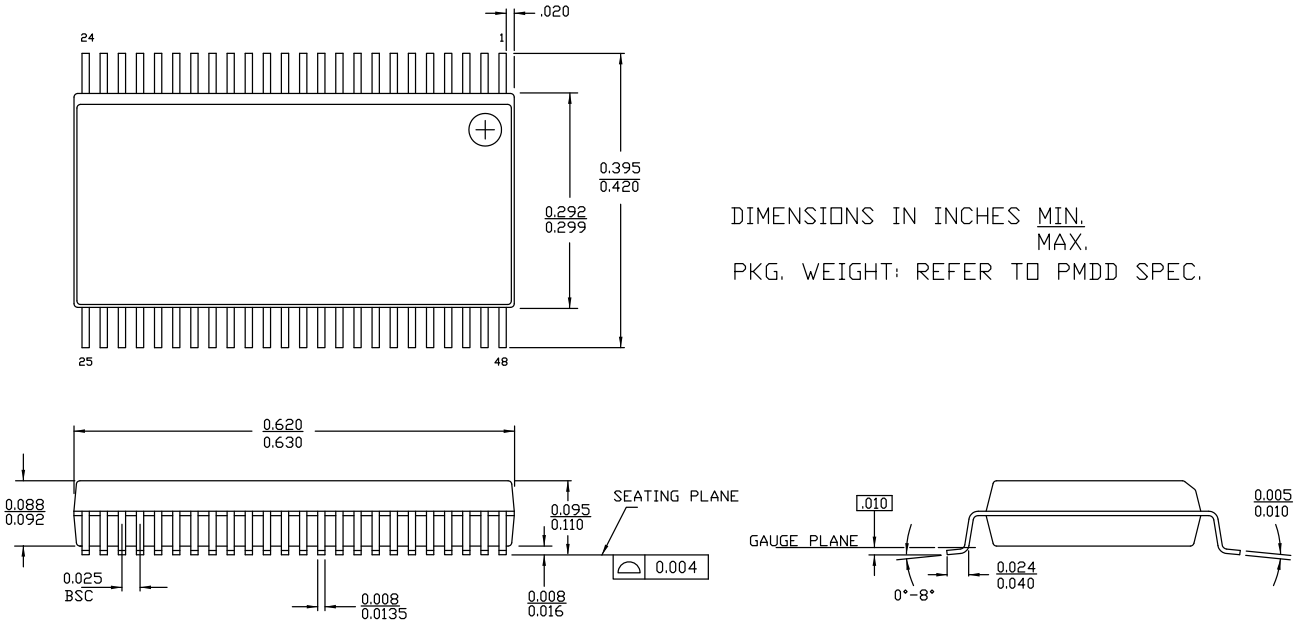
Die Pad Locations

Table 29. DIe Pad Locations (in microns)

Pad #	Pin Name	X	Y	Pad #	Pin Name	X	Y
1	D+	1496.95	2995.00	48	V_{CC}	1619.65	3023.60
2	D-	467.40	2995.00	47	V_{SS}	1719.65	3023.60
3	Port3[7]	345.15	3023.60	46	Port3[6]	1823.10	3023.60
4	Port3[5]	242.15	3023.60	45	Port3[4]	1926.10	3023.60
5	Port3[3]	98.00	2661.25	44	Port3[2]	2066.30	2657.35
6	Port3[1]	98.00	2558.25	43	Port3[0]	2066.30	2554.35
7	Port2[7]	98.00	2455.25	42	Port2[6]	2066.30	2451.35
8	Port2[5]	98.00	2352.25	41	Port2[4]	2066.30	2348.35
9	Port2[3]	98.00	2249.25	40	Port2[2]	2066.30	2245.35
10	Port2[1]	98.00	2146.25	39	Port2[0]	2066.30	2142.35
11	Por1[7]	98.00	1134.25	38	Port1[6]	2066.30	1130.35
12	Por1[5]	98.00	1031.25	37	Port1[4]	2066.30	1027.35
13	Por1[3]	98.00	928.25	36	Port1[2]	2066.30	924.35
14	Por1[1]	98.00	825.25	35	Port1[0]	2066.30	821.35
15	DAC7	98.00	721.05	34	DAC6	2066.30	719.55
16	DAC5	98.00	618.05	33	DAC4	2066.30	616.55
17	Port0[7]	98.00	516.25	32	Port0[6]	2066.30	512.35
18	Port0[5]	98.00	413.25	31	Port0[4]	2066.30	409.35
19	Port0[3]	306.30	98.00	30	Port0[2]	1858.00	98.00
20	Port0[1]	442.15	98.00	29	Port0[0]	1718.30	98.00
21	DAC3	593.40	98.00	28	DAC2	1618.50	98.00
22	DAC1	696.40	98.00	27	DAC0	1513.50	98.00
23	V_{PP}	824.25	98.00	26	XtalOut	1301.90	98.00
24	V_{SS}	949.65	98.00	25	XtalIn	1160.50	98.00

Package Diagrams

Figure 13. 48-pin SSOP (300 Mils) Package Outline, 51-85061



51-85061 *F

Figure 14. 24-pin SOIC (0.615 × 0.300 × 0.0932 Inches) Package Outline, 51-85025