CY7C66011/12/13
CY7C66111/12/13
Full-Speed USB (12 Mbps) Peripheral Controller with Integrated Hub

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF FIGURES

LIST OF FIGURES (continued)

LIST OF TABLES

1.0 Features

- Full-speed USB Microcontroller with an integrated USB hub
- 8-bit USB Optimized Microcontroller
  - Harvard architecture
  - 6-MHz external clock source
  - 12-MHz internal CPU clock
  - 48-MHz internal Hub clock
- Internal memory
  - 256 bytes of RAM
  - 4 KB of EPROM (CY7C66011, CY7C66111)
  - 6 KB of EPROM (CY7C66012, CY7C66112)
  - 8 KB of EPROM (CY7C66013, CY7C66113)
- Integrated Master/Slave I^2C Controller (100 kHz) enabled through GPIO pins
- Hardware Assisted Parallel Interface (HAPI) for data transfer to external devices
- I/O ports
  - Three General-Purpose I/O (GPIO) ports (Port 0 to 2) capable of sinking 7 mA per pin (typical)
  - An additional GPIO port (Port 3) capable of sinking 12 mA per pin (typical) for high current requirements: LEDs
  - Higher current drive achievable by connecting multiple GPIO pins together to drive a common output
  - Each GPIO port can be configured as inputs with internal pull-ups or open drain outputs or traditional CMOS outputs
  - A Digital to Analog Conversion (DAC) port with programmable current sink outputs is available on the CY7C66111/12/13 devices
  - Maskable interrupts on all I/O pins
- 12-bit free-running timer with one microsecond clock ticks
- Watchdog timer (WDT)
- Internal power-on reset (POR)
- USB Specification Compliance
  - Conforms to USB Specification, Version 1.0
  - Conforms to USB HID Specification, Version 1.0
  - Supports one or two device addresses with up to 5 user configured endpoints
    - Up to two 8-byte control endpoints
    - Up to four 8-byte data endpoints
    - Up to two 32-byte data endpoints
  - Integrated USB transceivers
  - Supports 4 Downstream USB ports
  - GPIO pins can provide individual power control outputs for each Downstream USB port
  - GPIO pins can provide individual port over current inputs for each Downstream USB port
- Improved output drivers to reduce EMI
- Operating voltage from 4.0V to 5.5V DC
- Operating temperature from 0 to 70 degrees Celsius
- CY7C66011/12/13 available in 48-pin PDIP or 48-pin SSOP packages
- CY7C66111/12/13 available in 56-pin SSOP packages
- A windowed package is available to enable quick development cycles:
  - CY7C66013-WC: 48-pin Windowed Sidebraze
- Industry standard programmer support

2.0 Functional Overview

The CY7C66011/12/13 and CY7C66111/12/13 are 8-bit microcontrollers (One Time Programmable or Windowed) with a built-in 12-Mbps USB hub that supports up to four downstream ports. The instruction set has been optimized specifically for USB operations, although the microcontrollers can be used for a variety of non-USB embedded applications.

The CY7C66011/12/13 features 29 General-Purpose I/O (GPIO) pins to support USB and other applications. The I/O pins are grouped into four ports (P0[7:0], P1[7:0], P2[7:0], P3[4:0]) where each port can be configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs. Ports 0 to 2 are rated at 7 mA per pin (typical) sink current. Port 3 pins are rated at 12 mA per pin (typical) sink current, which allows these pins to drive LEDs. Multiple GPIO pins can be connected together to drive a single output for more drive current capacity. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller. All of the GPIO interrupts all share the same “GPIO” interrupt vector.

The CY7C66111/12/13 has 31 GPIO pins (P0[7:0], P1[7:0], P2[7:0], P3[6:0]). Additionally, the CY7C66111/12/13 features an additional 8 I/O pins in the Digital to Analog Conversion (DAC) port (P4[7:0]). Every DAC pin includes an integrated 14-KΩ pull-up resistor. When a ‘1’ is written to a DAC I/O pin, the output current sink is disabled and the output pin is driven HIGH by the internal pull-up resistor. When a ‘0’ is written to a DAC I/O pin, the internal pull-up is disabled and the output pin provides the programmed amount of sink current. A DAC I/O pin can be used as an input with an internal pull-up by writing a ‘1’ to the pin.

The sink current for each DAC I/O pin can be individually programmed to one of sixteen values using dedicated Isink registers. DAC bits P4[1:0] can be used as high current outputs with a programmable sink current range of 3.2 to 16 mA (typical). DAC bits P4[7:2] have a programmable current sink range of 0.2 to 1.0 mA (typical). Multiple DAC pins can be connected together to drive a single output that requires more sink current capacity. Each I/O pin can be used to generate a DAC interrupt to the microcontroller and the interrupt polarity for each DAC I/O pin is individually programmable.

The microcontroller uses an external 6 MHz crystal and an internal oscillator to provide a reference to an internal PLL-based clock generator. This technology allows the customer application to use an inexpensive 6-MHz fundamental crystal that reduces the clock-related noise emissions (EMI). A PLL clock generator provides the 6-, 12-, and 48-MHz clock signals for distribution within the microcontroller.

The CY7C66011/12/13 and CY7C66111/12/13 are offered with three EPROM options. The CY7C66011 and CY7C66111 have 4 KB of EPROM. The CY7C66012 and CY7C66112 have 6 KB of EPROM. The CY7C66013 and CY7C66113 have 8 KB of EPROM.

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

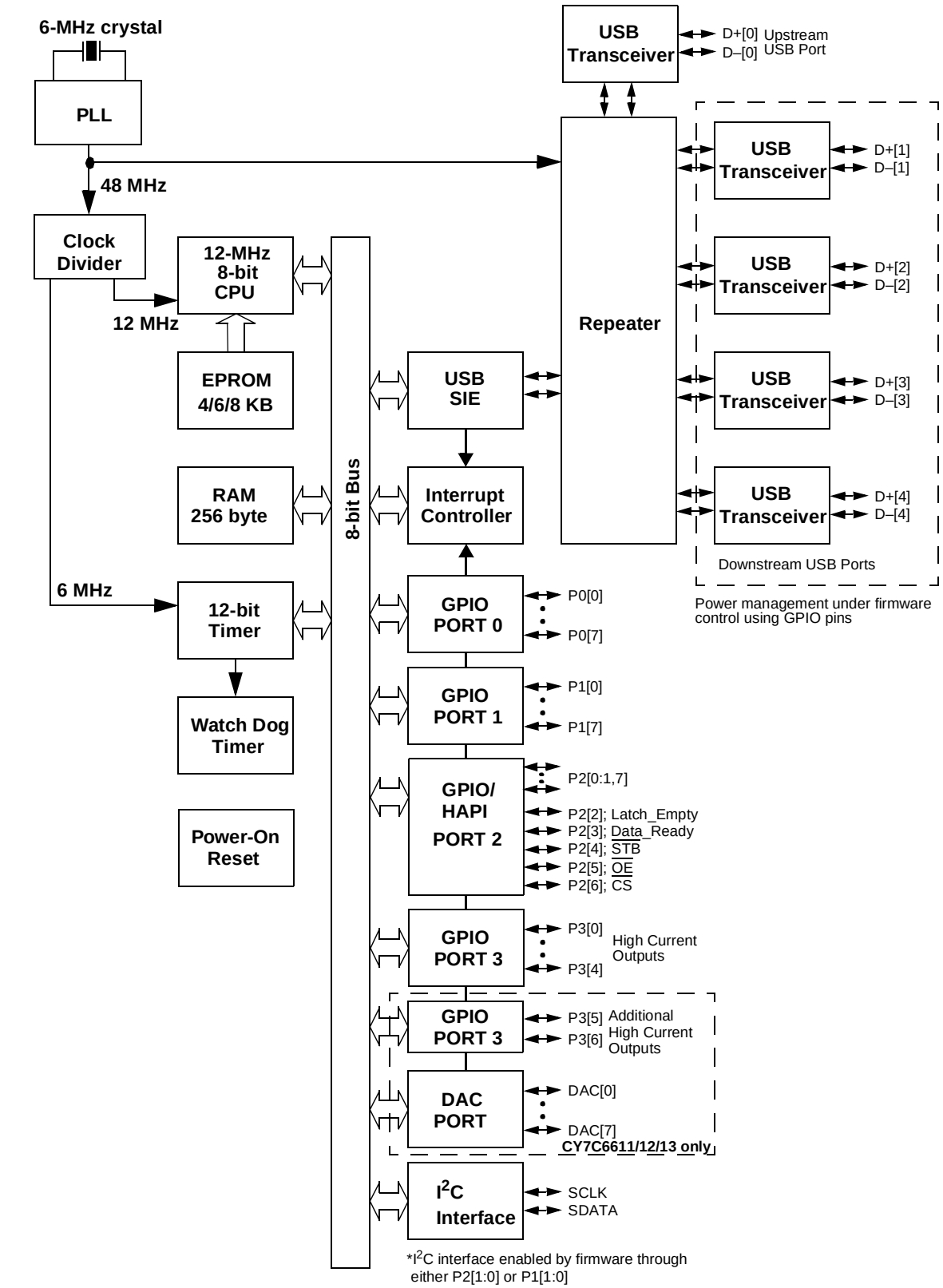
The microcontroller can communicate with external electronics through the GPIO pins. An I^2C interface accommodates a 100-kHz serial link with an external device. There is also a Hardware Assisted Parallel Interface which can be used to transfer data to an external device.

The free-running 12-bit timer clocked at 1 MHz provides two interrupt sources, 128-µs and 1.024-ms. The timer can be used to measure the duration of an event under firmware control by reading the timer twice: once at the start of the event, and once after the event is complete. The difference between the two readings indicates the duration of the event measured in microseconds. The upper four bits of the timer are latched into an internal register when the firmware reads the lower eight bits. A read from the upper four bits actually reads data from the internal register, instead of the timer. This feature eliminates the need for firmware to attempt to compensate if the upper four bits happened to increment right after the lower eight bits are read.

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

The CY7C66011/12/13 and CY7C66111/12/13 include an integrated USB serial interface engine (SIE) that supports the integrated peripherals and the hub controller function. The hardware supports up to two USB device addresses with one device address for the hub (two endpoints) and a device address for a compound device (three endpoints). The SIE allows the USB host to communicate with the hub and functions integrated into the microcontroller. The part includes a 1:4 hub repeater with one upstream port and four downstream ports. The USB hub includes power management control of the downstream ports using GPIO pins assigned by the user firmware. The user has the option of ganging the downstream ports together with a single pair of power management pins, or providing power management for each port with four pairs of power management pins.

Logic Block Diagram

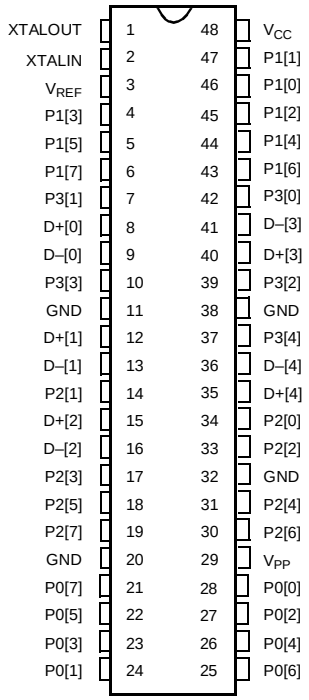


3.0 Pin Configurations

TOP VIEW

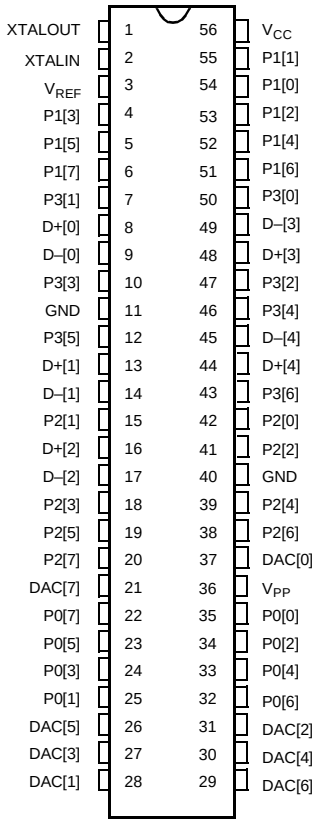
CY7C66011/12/13

48-pin PDIP/SSOP/WC



CY7C66111/12/13

56-pin SSOP



4.0 Product Summary Tables

4.1 Pin Assignments

Table 4-1. Pin Assignments

Name	I/O	48-Pin	56-Pin	Description
D+[0], D–[0]	I/O	8,9	8,9	Upstream port, USB differential data
D+[1], D–[1]	I/O	12,13	13,14	Downstream port 1, USB differential data
D+[2], D–[2]	I/O	15,16	16,17	Downstream port 2, USB differential data
D+[3], D–[3]	I/O	40,41	48,49	Downstream port 3, USB differential data
D+[4], D–[4]	I/O	35,36	44,45	Downstream port 4, USB differential data
P0[7:0]	I/O	21, 25, 22, 26, 23, 27, 24, 28	22, 32, 23, 33, 24, 34, 25, 35	General purpose I/O (GPIO) port 0 capable of sinking 7 mA (typical)
P1[7:0]	I/O	6, 43, 5, 44, 4, 45, 47, 46	6, 51, 5, 52, 4, 53, 55, 54	GPIO Port 1 capable of sinking 7 mA (typical)
P2[7:0]	I/O	19, 30, 18, 31, 17, 33, 14, 34	20, 38, 19, 39, 18, 41, 15, 42	GPIO Port 2 capable of sinking 7 mA (typical)
P3[6:0]	I/O	37, 10, 39, 7, 42	43, 12, 46, 10, 47, 7, 50	GPIO Port 3 capable of sinking 12 mA (typical)
DAC[7:0]	I/O	n/a	21, 29, 26, 30, 27, 31, 28, 37	Digital to Analog Converter (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	6-MHz crystal or external clock input
$XTAL_{OUT}$	OUT	1	1	6-MHz crystal out
V_{PP}		29	36	Programming voltage supply, tie to ground during normal operation
V_{CC}		48	56	Voltage supply
GND		11, 20, 32, 38	11, 40	Ground
V_{REF}	IN	3	3	External 3.3V supply voltage for the downstream differential data output buffers and the D+ pull up

4.2 I/O Register Summary

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

Table 4-2. I/O Register Summary

Register Name	I/O Address	Read/Write	Function
Port 0 Data	0x00	R/W	GPIO Port 0 Data
Port 1 Data	0x01	R/W	GPIO Port 1 Data
Port 2 Data	0x02	R/W	GPIO Port 2 Data
Port 3 Data	0x03	R/W	GPIO Port 3 Data
Port 0 Interrupt Enable	0x04	W	Interrupt enable for pins in Port 0
Port 1 Interrupt Enable	0x05	W	Interrupt enable for pins in Port 1
Port 2 Interrupt Enable	0x06	W	Interrupt enable for pins in Port 2
Port 3 Interrupt Enable	0x07	W	Interrupt enable for pins in Port 3
GPIO Configuration	0x08	R/W	GPIO Port Configurations

Table 4-2. I/O Register Summary (continued)

Register Name	I/O Address	Read/Write	Function
HAPI and I²C Configuration	0x09	R/W	HAPI Width and I²C Position Configuration
USB Device Address A	0x10	R/W	USB Device Address A
EP A0 Counter Register	0x11	R/W	USB Address A, Endpoint 0 counter
EP A0 Mode Register	0x12	R/W	USB Address A, Endpoint 0 configuration
EP A1 Counter Register	0x13	R/W	USB Address A, Endpoint 1 counter
EP A1 Mode Register	0x14	R/W	USB Address A, Endpoint 1 configuration
EP A2 Counter Register	0x15	R/W	USB Address A, Endpoint 2 counter
EP A2 Mode Register	0x16	R/W	USB Address A, Endpoint 2 configuration
USB Status & Control	0x1F	R/W	USB upstream port traffic status and control
Global Interrupt Enable	0x20	R/W	Global interrupt enable
Endpoint Interrupt Enable	0x21	R/W	USB endpoint interrupt enables
Reserved[1]	0x23		Reserved
Timer (LSB)	0x24	R	Lower 8 bits of free-running timer (1 MHz)
Timer (MSB)	0x25	R	Upper 4 bits of free-running timer
WDT Clear	0x26	W	Watch Dog Timer clear
I²C Control & Status	0x28	R/W	I²C status and control
I²C Data	0x29	R/W	I²C Data
DAC Data	0x30	R/W	DAC Data
DAC Interrupt Enable	0x31	W	Interrupt enable for each DAC pin
DAC Interrupt Polarity	0x32	W	Interrupt polarity for each DAC pin
DAC Isink	0x38-0x3F	W	Input sink current control for each DAC pin
USB Device Address B	0x40	R/W	USB Device Address B
EP B0 Counter Register	0x41	R/W	USB Address B, Endpoint 0 counter
EP B0 Mode Register	0x42	R/W	USB Address B, Endpoint 0 configuration
EP B1 Counter Register	0x43	R/W	USB Address B, Endpoint 1 counter
EP B1 Mode Register	0x44	R/W	USB Address B, Endpoint 1 configuration
Hub Port Connect Status	0x48	R	Hub downstream port connect status
Hub Port Enable	0x49	R/W	Hub downstream ports enable
Hub Port Speed	0x4A	R	Hub downstream ports speed
Hub Port Control (Ports [4:1])	0x4B	R/W	Hub downstream ports control (Ports [4:1])
Reserved[1]	0x4C		Reserved
Hub Port Suspend	0x4D	R/W	Hub downstream port suspend control
Hub Port Resume Status	0x4E	R	Hub downstream ports resume status
Hub Ports SE0 Status	0x4F	R	Hub downstream ports SE0 status
Hub Ports Data	0x50	R	Hub downstream ports differential data
Device ID Register	0xF3	R	USB ID for Device Revision (0x3Z)[2]
Manufacturer ID Register	0xF4	R	USB ID for Cypress Semiconductor (0x34)
Processor Status & Control	0xFF	R/W	Microprocessor Status and Control Register

Notes:
1. All undefined registers are reserved. Special care should be taken not to write to reserved registers as this action may result in increased current consumption during operation. When writing to registers with reserved bits, the reserved bits must be written with '0'.
2. Second nibble 'Z' indicates silicon version. Currently Z is '2'.

4.3 Instruction Set Summary

Table 4-3. Instruction Set Summary

MNEMONIC	operand	opcode	cycles		MNEMONIC	operand	opcode	cycles
HALT		00	7		NOP		20	4
ADD A,expr	data	01	4		INC A	acc	21	4
ADD A,[expr]	direct	02	6		INC X	x	22	4
ADD A,[X+expr]	index	03	7		INC [expr]	direct	23	7
ADC A,expr	data	04	4		INC [X+expr]	index	24	8
ADC A,[expr]	direct	05	6		DEC A	acc	25	4
ADC A,[X+expr]	index	06	7		DEC X	x	26	4
SUB A,expr	data	07	4		DEC [expr]	direct	27	7
SUB A,[expr]	direct	08	6		DEC [X+expr]	index	28	8
SUB A,[X+expr]	index	09	7		IORD expr	address	29	5
SBB A,expr	data	0A	4		IOWR expr	address	2A	5
SBB A,[expr]	direct	0B	6		POP A		2B	4
SBB A,[X+expr]	index	0C	7		POP X		2C	4
OR A,expr	data	0D	4		PUSH A		2D	5
OR A,[expr]	direct	0E	6		PUSH X		2E	5
OR A,[X+expr]	index	0F	7		SWAP A,X		2F	5
AND A,expr	data	10	4		SWAP A,DSP		30	5
AND A,[expr]	direct	11	6		MOV [expr],A	direct	31	5
AND A,[X+expr]	index	12	7		MOV [X+expr],A	index	32	6
XOR A,expr	data	13	4		OR [expr],A	direct	33	7
XOR A,[expr]	direct	14	6		OR [X+expr],A	index	34	8
XOR A,[X+expr]	index	15	7		AND [expr],A	direct	35	7
CMP A,expr	data	16	5		AND [X+expr],A	index	36	8
CMP A,[expr]	direct	17	7		XOR [expr],A	direct	37	7
CMP A,[X+expr]	index	18	8		XOR [X+expr],A	index	38	8
MOV A,expr	data	19	4		IOWX [X+expr]	index	39	6
MOV A,[expr]	direct	1A	5		CPL		3A	4
MOV A,[X+expr]	index	1B	6		ASL		3B	4
MOV X,expr	data	1C	4		ASR		3C	4
MOV X,[expr]	direct	1D	5		RLC		3D	4
reserved		1E			RRC		3E	4
XPAGE		1F	4		RET		3F	8
MOV A,X		40	4		DI		70	4
MOV X,A		41	4		EI		72	4
MOV PSP,A		60	4		RETI		73	8
CALL	addr	50 - 5F	10		JC	addr	C0-CF	5
JMP	addr	80-8F	5		JNC	addr	D0-DF	5
CALL	addr	90-9F	10		JACC	addr	E0-EF	7
JZ	addr	A0-AF	5		INDEX	addr	F0-FF	14
JNZ	addr	B0-BF	5					

5.0 Programming Model

5.1 14-Bit Program Counter (PC)

The 14-bit program counter (PC) allows access to up to 8 KB of EPROM available with the CY7C66xxx architecture. The top 32 bytes of the ROM in the 8K part are reserved for testing purposes. The program counter is cleared during reset, such that the first instruction executed after a reset is at address 0x0000h. This is typically a jump instruction to a reset handler that initializes the application, reference Interrupt Vectors on page 24.

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

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

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

5.1.1 Program Memory Organization

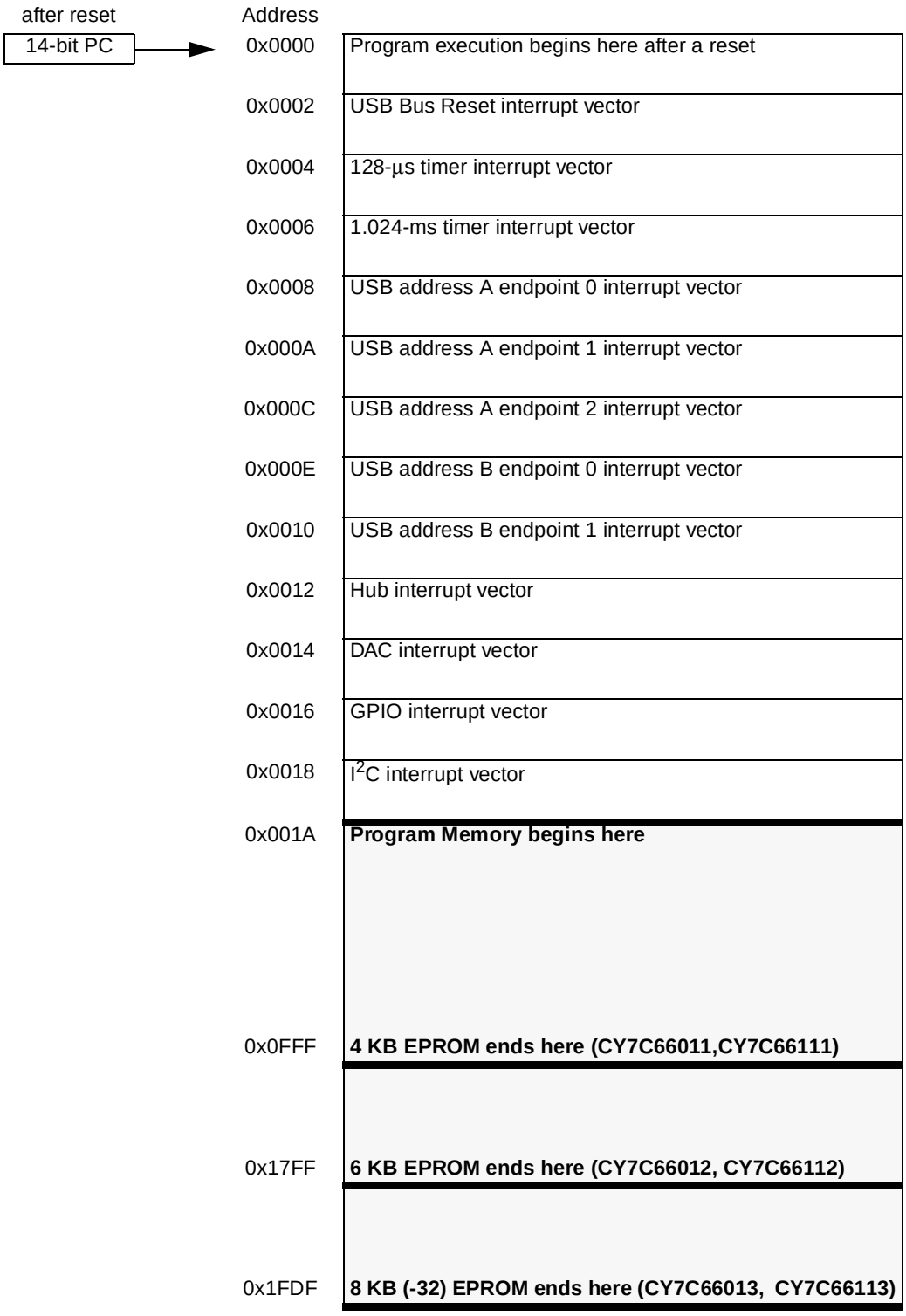


Figure 5-1. Program Memory Space with Interrupt Vector Table

5.2 8-Bit Accumulator (A)

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

5.3 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 Section 5.6.3 for additional information.

5.4 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 program stack pointer 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 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 overall effect is to store the program counter and flags on the program “stack” and increment the program stack pointer by two.

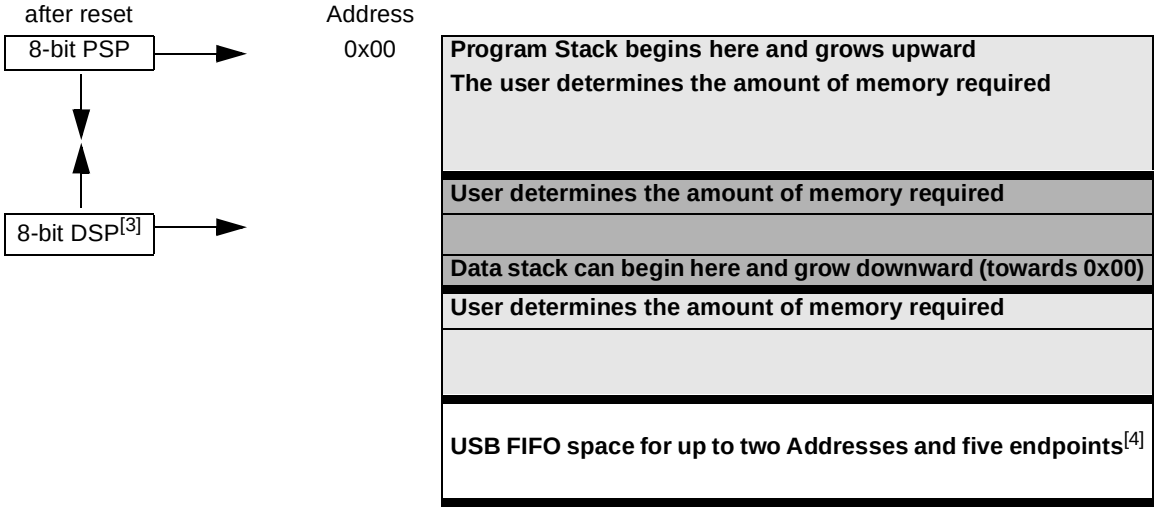
The return from interrupt (RETI) instruction decrements the program stack pointer, then restores the second byte from memory addressed by the PSP. The program stack pointer is decremented again and the first byte is restored from memory addressed by the PSP. After the program counter and flags have been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

5.4.1 Data Memory Organization

The CY7C66112 and CY7C66113 microcontrollers provide 256 bytes of data RAM. In normal usage, the SRAM is partitioned into four areas: program stack, user variables, data stack, and USB endpoint FIFOs, as shown below:



Notes:
3. Refer to Section 5.5 for a description of DSP.
4. Endpoint sizes are fixed by the Endpoint Size Bit (I/O register 0x1F, Bit 7), see *Table 18-1.*

5.5 8-Bit Data Stack Pointer (DSP)

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

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

For USB applications, the firmware should set the DSP to the appropriate location to avoid a memory conflict with RAM dedicated to USB FIFOs. The memory requirements for the USB endpoints are described in the section on USB Device Endpoints on page 30. The assembly instructions to do this with two device addresses (0xD8) are shown below:

```
MOV A,D8h    ; Move D8 hex into Accumulator
SWAP A,DSP   ; swap accumulator value into DSP register
```

5.6 Address Modes

The CY7C66011/12/13 and CY7C66111/12/13 microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

5.6.1 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 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 “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

5.6.2 Direct

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

5.6.3 Indexed

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

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

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

6.0 Clocking

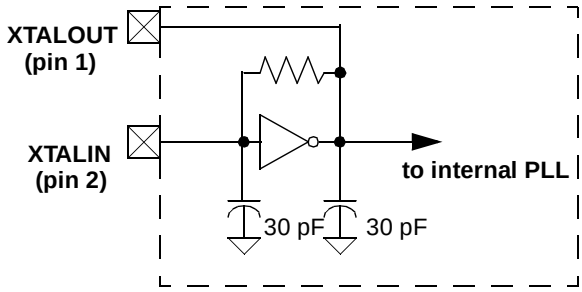


Figure 6-1. Clock Oscillator On-Chip Circuit

The XTALIN and XTALOUT are the clock pins to the microcontroller. The user can connect either an external oscillator or a crystal to these pins. A 6-MHz fundamental crystal can be connected to these pins to provide a reference frequency for the internal PLL. A ceramic resonator will not allow the microcontroller to meet the timing specifications of a high 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. Please note that grounding the XTALOUT pin when driving XTALIN with an oscillator will not work as the internal clock is effectively shorted to ground.

7.0 Reset

The CY7C66xxx supports two resets: Power-On Reset (POR) and a Watch Dog Reset (WDR). Each of these resets will cause:

- all Registers to be restored to their default states,
- the USB Device Addresses to be set to 0,
- all interrupts to be disabled,
- the Program Stack Pointer (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 located at I/O address 0xFF. 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 either a RET or RETI in the firmware reset handler will cause unpredictable execution results.

7.1 Power-On Reset (POR)

The CY7C66xxx enters a suspend state when the V_{CC} is first applied to the chip. The Power-On Reset (POR) signal is asserted during this suspend time and ensures that both a valid V_{CC} level (V_{rst}) is reached and that the internal PLL has time to stabilize. When the V_{CC} has risen above V_{rst} and the oscillator is stable the POR is deasserted and the on-chip timer will start counting. The first 1 ms of suspend time is not interruptible, but the suspend state will continue for an additional 127 ms unless the count is bypassed by a USB Bus Reset interrupt on the upstream port. Upon a USB Bus Reset, the interrupt will be pending and the IRQ is generated only after the firmware enables the Interrupt Mask (bit 2 of register 0xFF) and the USB Bus Reset interrupt (bit 0 of register 0x20). This 127 ms time period guarantees that when the V_{CC} ramp takes a very long time to reach full operating voltage the chip will not execute code. If the oscillator is stable and V_{CC} is above V_{rst}, a USB Bus Reset will cause the additional 127 ms of suspend to be bypassed. When 127 ms has passed or a USB Bus Reset is asserted, the chip will immediately begin execution of code at address zero (0x0). Notice that this response to a USB Bus Reset will only occur after a minimum of 1 ms to allow the PLL to stabilize.

The POR signal will also be asserted if V_{CC} drops below the predetermined value (V_{rst}). The POR will remain asserted until the V_{CC} value once again rises above V_{rst}. Behavior is the same as described above.

7.2 Watch Dog Reset (WDR)

The Watch Dog Timer Reset (WDR) occurs when the Most Significant Bit (MSB) of the 4-bit Watch Dog Timer Register (0x26) 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.

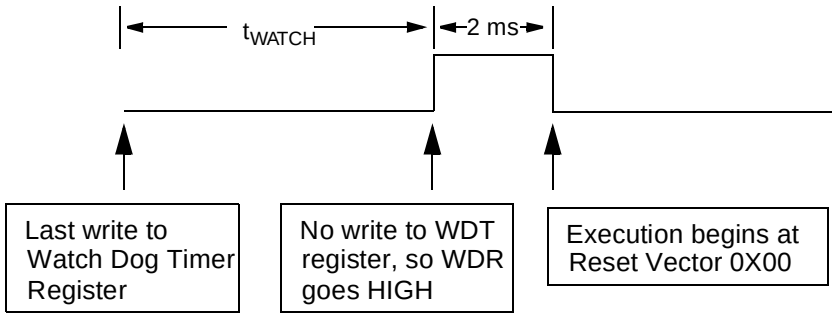


Figure 7-1. Watch Dog Reset (WDR)

The Watch Dog Timer is clocked by a 1.024-ms clock (bit 9) from the free-running timer. The 1.024-ms clock is the same timer output that can be used to interrupt the processor every 1.024 ms. Writing any value to the write-only Watch Dog Clear I/O register (0x26) will clear the 4-bit Watch Dog Timer.

If the 1.024-ms timer interrupt service routine does not get executed for t_{WATCH} (8 ms minimum) a Watch Dog Timer Reset will occur. A Watch Dog Timer Reset lasts for 2 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 Registers are cleared. Otherwise, the USB Controller would respond to all address 0 transactions.

8.0 General-Purpose I/O Ports

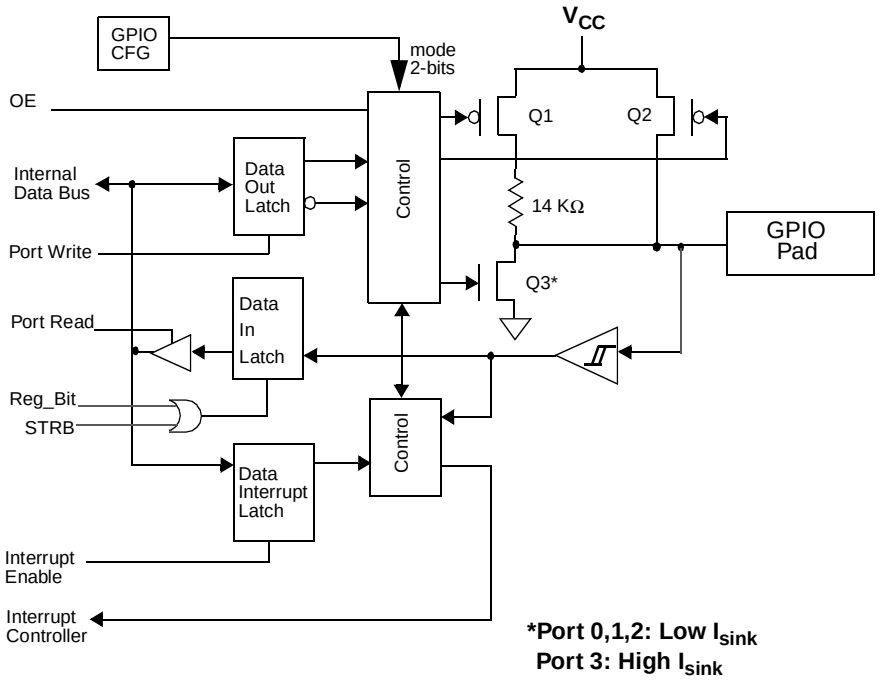


Figure 8-1. Block Diagram of a GPIO Pin

There are up to 31 GPIO pins (P0[7:0], P1[7:0], P2[7:0], and P3[6:0]) for the hardware interface. The number of GPIO pins changes based on the package type of the chip. Each port can be configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs. Ports 0 to 2 are considered low current drive with typical current sink capability of 7 mA. Port 3 offers higher current drive with a typical current sink capability of 12 mA. The data for each GPIO port is accessible through the data registers.

P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]

Figure 8-2. Port 0 Data 0x00 (read/write)

P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]

Figure 8-3. Port 1 Data 0x01 (read/write)

P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]

Figure 8-4. Port 2 Data 0x02 (read/write)

P3[7]	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]

Figure 8-5. Port 3 Data 0x03 (read/write)

Special care should be exercised 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 will

be 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 CY7C66011/12/13 will always require that P3[7:5] be written with a '0.' When the CY7C66111/12/13 is used the P3[7] should be written with a '0.'

During reset, all of the GPIO pins are set to input ('1' in open drain) state. Writing a '0' to a GPIO pin enables the output current sink to ground (LOW). In this state, a '0' will always be read on that GPIO pin unless an external source overdrives the output current sink.

8.1 GPIO Configuration Port

Every GPIO port can be programmed as inputs with internal pull-ups, open drain outputs, and traditional CMOS outputs. In addition, the interrupt polarity for each port can be programmed. With positive interrupt polarity, a rising edge ('0' to '1') on an input pin causes an interrupt. With negative polarity, a falling edge ('1' to '0') on an input pin causes an interrupt. As shown in the table below, when a GPIO port is configured with CMOS outputs, interrupts from that port are disabled. The GPIO Configuration Port register provides two bits per port to program these features. The possible port configurations are detailed in *Table 8-1*.

Table 8-1. Port Configurations

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

In "Resistive" mode, a 14-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 14-kΩ pull-up resistor when a '1' has been written to the pin. 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 14-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.

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 an input or an open drain 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' to select Open Drain output, positive interrupt polarity for all GPIO ports as the default configuration.

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

Figure 8-6. GPIO Configuration Register 0x08 (read/write)

8.2 GPIO Interrupt Enable Ports

When HAPI (HAPI is discussed in Section 13.0) is enabled the GPIO interrupts are blocked, including ports not used by HAPI, and cannot interrupt the microcontroller.

During a reset, GPIO interrupts are disabled by clearing all of the GPIO interrupt enable ports. Writing a '1' to a GPIO Interrupt Enable bit enables GPIO interrupts from the corresponding input pin.

P0[7]	P0[6]	P0[5]	P0[4]	P0[3]	P0[2]	P0[1]	P0[0]

Figure 8-7. Port 0 Interrupt Enable 0x04 (read/write)

P1[7]	P1[6]	P1[5]	P1[4]	P1[3]	P1[2]	P1[1]	P1[0]

Figure 8-8. Port 1 Interrupt Enable 0x05 (read/write)

P2[7]	P2[6]	P2[5]	P2[4]	P2[3]	P2[2]	P2[1]	P2[0]

Figure 8-9. Port 2 Interrupt Enable 0x06 (read/write)

not used	P3[6]	P3[5]	P3[4]	P3[3]	P3[2]	P3[1]	P3[0]

Figure 8-10. Port 3 Interrupt Enable 0x07 (read/write)

9.0 DAC Port

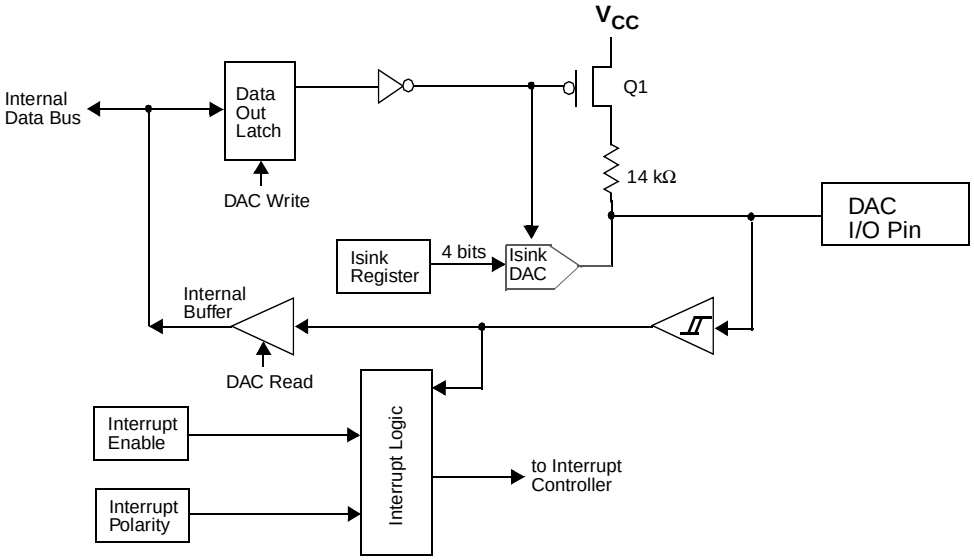


Figure 9-1. Block Diagram of a DAC Pin

The CY7C66111/12/13 features a Digital to Analog Conversion (DAC) port which has programmable current sink on each I/O pin. 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. A '0' output will cause the Isink DAC to sink current to drive the output LOW. 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 a minimum of 3.2 mA to a maximum of 16 mA (typical). DAC[7:2] are low current outputs that are programmable from a minimum of 0.2 mA to a maximum of 1.0 mA (typical).

When a DAC I/O bit is written as a '1', the I/O pin is either an output pulled HIGH through the 14-kΩ resistor or an input with an internal 14-kΩ pull-up resistor. All DAC port data bits are set to '1' during reset.

Low current outputs 0.2 mA to 1.0 mA typical						High current outputs 3.2 mA to 16 mA typical	
DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]

Figure 9-2. DAC Port Data 0x30 (read/write)

9.1 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 mask bit for each DAC I/O pin. Writing a '1' to a bit in this register enables interrupts from the corresponding bit position. Writing a '0' to a bit in the DAC Port Interrupt Enable register disables interrupts from the corresponding bit position. All of the DAC Port Interrupt Enable register bits are cleared to '0' during a reset.

DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]

Figure 9-3. DAC Port Interrupt Enable 0x31 (write only)

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 will cause 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 will cause 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.

DAC[7]	DAC[6]	DAC[5]	DAC[4]	DAC[3]	DAC[2]	DAC[1]	DAC[0]

Figure 9-4. DAC Port Interrupt Polarity 0x32 (write only)

9.2 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]. Writing all '0's to the Isink register will cause 1/5 of the max current to flow through the DAC I/O pin. While writing all '1's to the Isink register will provide the maximum current flow through the pin. The other 14 states of the DAC sink current are evenly spaced between these two values.

Reserved				Isink Value			
				Isink[3]	Isink[2]	Isink[1]	Isink[0]

Figure 9-5. DAC Port Isink 0x38 to 0x3F (write only)

10.0 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.

10.1 Timer (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

Figure 10-1. Timer Register 0x24 (read only)

10.2 Timer (MSB)

Reserved	Reserved	Reserved	Reserved	Timer Bit 11	Timer Bit 10	Timer Bit 9	Timer Bit 8

Figure 10-2. Timer Register 0x25 (read only)

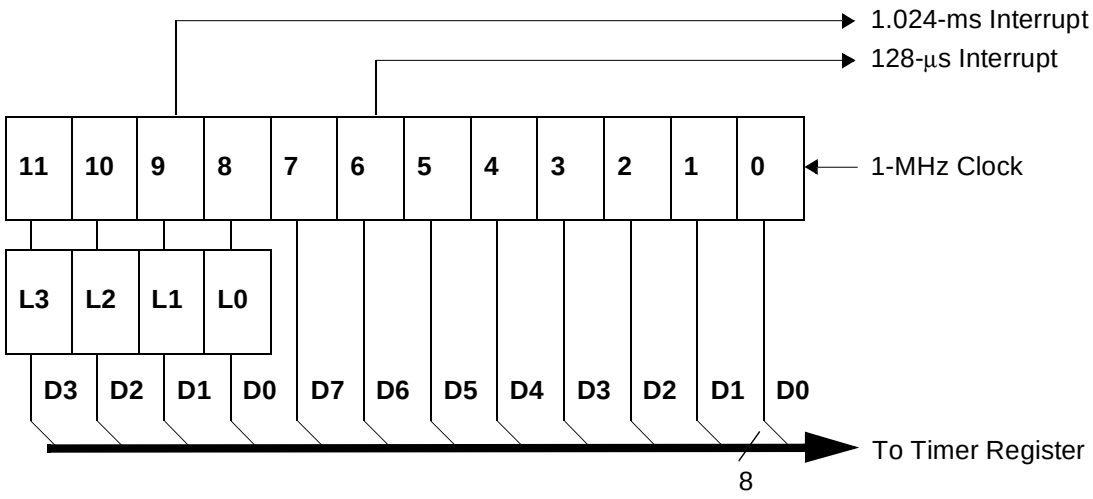


Figure 10-3. Timer Block Diagram

11.0 HAPI and I²C Configuration Register

HAPI (Hardware Assisted Parallel Interface) provides for a 1, 2, or 3 byte interface to an external device. I²C interface utilizes the standard interface for Inter-IC Communication. Register 0x09 configures the HAPI and I²C interfaces. The detailed operation of HAPI and I²C is discussed in Sections 12.0 and 13.0.

The operation of these interfaces is explained below.

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R	R	R/W	R/W
I²C Position	Reserved	LEMPTY Polarity	DRDY Polarity	Latch Empty	Data Ready	HAPI Port Width Bit 1	HAPI Port Width Bit 0

Figure 11-1. HAPI/I²C Configuration Register 0x09 (read/write)

Bits [7,1:0] of the HAPI/I²C Configuration Register control the pin out configuration of the HAPI and I²C interfaces.

Latch Empty and Data Ready bits: These bits are an inversion of the respective pins. They exist so that the firmware can determine what type of operations need to be performed on the HAPI port following an interrupt. Care must be taken to consider the state of LEMPTY and DRDY polarity signals when reading these bits. For example, following a HAPI interrupt due to an external device reading data from the HAPI port the DRDY signal will have transitioned from a 1 to a 0. The Data Ready bit will transition from a 0 to a 1 indicating that the ports need to be reloaded with data for the external device to read.

DRDY and LEMPTY Polarity bits: These bits control the polarity of the output pins. The default polarity is high true. Setting this bit to a 1 changes the polarity to low true.

Table 11-1. HAPI Port Configuration

Port Width Bits[1:0]	HAPI Port Width
11	24 Bits: P3[7:0], P1[7:0], P0[7:0]
10	16 Bits: P1[7:0], P0[7:0]
01	8 Bits: P0[7:0]
00	No HAPI Interface

Table 11-2. I²C Port Configuration

I²C Position Bit	Port Width Bit[1]	I²C Position
X	1	I²C on P2[1:0], 0:SCL, 1:SDA
0	0	I²C on P1[1:0], 0:SCL, 1:SDA
1	0	I²C on P2[1:0], 0:SCL, 1:SDA

12.0 I²C Master Mode Controller

The I²C interface consists of two registers, a control and status register, and a data register. The status and control register is located at I/O address 0x28. The data register is located at I/O address 0x29 and is implemented as separate read and write registers. The bit definition and functionality for the HAPI/I²C Configuration Register (0x09) are explained in Section 11.0.

Note that once the I²C functionality is enabled by setting the bit 0 of the I²C Status & Control Register to a HIGH, the LS 2 bits of the corresponding port will be placed in the Open Drain mode, regardless of the settings of the GPIO Configuration Register (0x08).

7	6	5	4	3	2	1	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
MSTR Mode	Continue/ Busy	Xmit Mode	ACK	Addr	ARB Lost/ Restart	Received Stop	I²C Enable

Figure 12-1. I²C Status & Control 0x28 (read/write)

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

Figure 12-2. I²C Data 0x29 (read/write)

The I²C SCLK is connected to bit 0 of either GPIO port 1 or GPIO port 2. The I²C SDATA is connected to bit 1 of either GPIO port 1 or GPIO port 2. The port selection is determined by settings in the register 0x09, HAPI/I²C Configuration register.

The control/status bits are defined as follows:

MSTR Mode: This bit causes the I²C to initiate a master mode transaction. The contents of the data register are transmitted on the I²C bus as the target address. The I²C block performs required arbitration and clock synchronization. The loss of arbitration will result in the clearing of this bit, the setting of the ARB Lost bit, and the generation of an interrupt to the microcontroller. If the chip is target of an external master then the interrupt will be held off until the transaction from the external master is completed.

Continue / Busy: This bit is written by the firmware to indicate that the firmware has responded to an interrupt request and has completed the required update/read of the data register. On a read this bit indicates if the hardware is busy and can not accept additional writes to the I²C control register. This bit is set following a write to the control register that sets the Continue bit to a one (1). Whenever this bit is set the hardware inhibits writes to the control register. This is to allow for the hardware to complete certain operations that may require an extended period of time. This bit will not be set following an interrupt until the Continue bit is set. This allows the firmware to make one write to the control register without the need to check the Busy bit.

Xmit Mode: When the I²C needs to perform a transmit of data to an external master or slave this bit is set in conjunction with the Continue bit. The firmware will determine the value of this bit by looking at the R/W bit of the address received as a slave or directly set it when sending as a master.

ACK: This bit is set or cleared by the firmware during receive operation to indicate if the hardware should generate an ACK signal on the I²C bus. This bit is written with the inverse value of the ACK bit from the I²C bus at the end of a transmit operation. The inversion is present since the ACK signal on the I²C bus is true LOW, this bit is true when HIGH (1).

Addr: This bit is set by the I²C during the first byte of a slave receive transaction. This bit is cleared when the firmware sets the Continue bit after reading the byte. The Xmit bit must be set according to the R/W bit of the address. The primary purpose of this bit is to allow the firmware to recognize when the master has lost arbitration. Since the chip may be the target of the external master the firmware needs to recognize the difference between an interrupt due to the chip receiving an address byte and the chip completing the send of an address.

ARB Lost/Restart: This bit is valid as a status bit (ARB Lost) only after a MSTR Mode transaction has been attempted. See the description of the MSTR Mode bit above. If during a master mode transaction the user wishes to perform an I²C restart sequence then this bit can be set, along with Continue and MSTR Mode, after the last data byte is sent. The I²C target address must be written to the data register prior to setting the Continue bit as the Continue bit will cause the hardware to immediately begin the restart sequence and transmission of the data register contents. To prevent false ARB Lost signals, the Restart bit is cleared by hardware during the restart sequence.

Receive Stop: This bit is set when the slave is in receive mode and detects a stop bit on the bus. This bit is the only method for the firmware to determine that a slave receive transaction has been terminated by the external master. The Receive Stop bit will not be set if the firmware terminates the I^2C transaction by not acknowledging the previous byte transmitted on the I^2C bus. This would happen when the firmware sets the Continue bit and clears the ACK bit in the I^2C Status & Control Register (0x28).

I^2C Enable: When this bit is cleared the GPIO pins are free to function as GPIOs. Setting this bit sets the pins in the proper mode and routes the appropriate signals to and from the pins so that they can be connected to the I^2C bus.

The I^2C will generate an interrupt to the microcontroller at the end of each byte received or transmitted, when a stop bit is detected by the slave when in receive mode, and when arbitration is lost.

All control of the I^2C data and clock lines is performed by the chip.

13.0 Hardware Assisted Parallel Interface (HAPI)

The CY7C66xxx processor will provide a hardware assisted parallel interface for bus widths of 8, 16, or 24 bits, to accommodate data transfer with an external microcontroller or a similar device. The bit definition and functionality for the HAPI/I^2C Configuration Register (0x09) is explained in Section 11.0.

The following signals are provided on P2 to control the HAPI interface.

Pin	Name	Direct	Polarity	Description
P2[2]	Latch_Empty	Out	Selectable	Ready for more In Data from Ext. Interface
P2[3]	Data_Ready	Out	Selectable	Out Data ready for Ext. Interface
P2[4]	$\overline{STB}$	In	Lo-True	Latches incoming data
P2[5]	$\overline{OE}$	In	Lo-True	Output Enable—Ready for more Out Data
P2[6]	$\overline{CS}$	In	Lo-True	Chip Select (Gates /STB and /OE)

The control bits for selecting 8-, 16-, or 24-bit width are Register 0x09, bits 1 and 0.

When writing a 24-bit or a 16-bit value to be transferred out to the External Interface, Byte 0 (to Port0) should be written last, as this will trigger the assertion of Data_Ready. Likewise, when reading a 24-bit or a 16-bit value from the External Interface, Byte 0 (from Port0) should be read last, as this will trigger the assertion of Latch_Empty.

Both $\overline{STB}$ and $\overline{OE}$ cause a GPIO Interrupt (Vector 11). Firmware should read bits [3:2] of the HAPI/I^2C Configuration register to see if the interrupt was caused by $\overline{STB}$ or $\overline{OE}$.

The status bits will update with the same timing or sooner than the external pins. The interrupt will be held off till the end of the external cycle.

14.0 Processor Status and Control Register

7	6	5	4	3	2	1	0
R	R/W	R/W	R/W	R/W	R	R/W	R/W
IRQ Pending	Watch Dog Reset	USB Bus Reset Interrupt	Power-On Reset	Suspend, Wait for Interrupt	Interrupt Mask	Single Step	Run

Figure 14-1. Processor Status and Control Register 0xFF

The “Run” (bit 0) is manipulated by the HALT instruction. When Halt is executed, the processor clears the run bit and halts at the end of the current instruction. The processor remains halted until an appropriate reset occurs (power-on or watchdog).

The “Single Step” (bit 1) is provided to support a hardware debugger. When single step is set, the processor will execute one instruction and halt (clear the run bit).

The “Interrupt Mask” (bit 2) shows whether interrupts are enabled or disabled. The firmware has no direct control over this bit as writing a zero or one to this bit position will have no effect on interrupts. A ‘0’ indicates that interrupts are masked and a ‘1’ indicates that the interrupts are enabled. This bit is further gated with the bit settings of the Global Interrupt Enable Register (0x20) and USB End Point Interrupt Enable Register (0x21). Instructions DI, EI, and RETI manipulate the internal hardware that controls the state of the interrupt mask bit in the Processor Status and Control Register.

Writing a ‘1’ to “Suspend, wait for interrupt” (bit 3) will halt the processor and cause the microcontroller to enter the “suspend” mode that significantly reduces power consumption. The program counter that is pushed onto the program stack by the interrupt service routine will be the instruction after the one that wrote ‘1’ to bit 3 of the Processor Status and Control Register.

The “Power-On Reset” (bit 4) is only 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.

The “USB Bus Reset Interrupt” (bit 5) will occur when a USB Bus Reset is received. The USB Bus Reset is a singled-ended zero (SE0) that lasts at least 12–16 µs. An SE0 is defined as the condition in which both the D+ line and the D– line are LOW at the same time. When the SIE detects this condition, the USB Bus Reset interrupt bit is set in the Processor Status and Control register and an USB Bus Reset interrupt is generated.

The “Watch Dog Reset” (bit 6) is set during a reset initiated by the Watch Dog Timer. This indicates the Watch Dog Timer went for more than t_{WATCH} (8 ms minimum) between Watch Dog clears.

The “IRQ pending” (bit 7) indicates one or more of the interrupts has been recognized as active. The interrupt acknowledge sequence will clear this bit until the next interrupt is detected.

During Power-On Reset, the Processor Status and Control Register is set to 00010001, which indicates a Power-On Reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear).

During a Watch Dog Reset, the Processor Status and Control Register is set to 01XX0001, which indicates a Watch Dog Reset (bit 4 set) has occurred and no interrupts are pending (bit 7 clear).

15.0 Interrupts

All interrupts are maskable by the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register. Writing a ‘1’ to a bit position enables the interrupt associated with that bit position. During a reset, the contents the Global Interrupt Enable Register and USB End Point Interrupt Enable Register are cleared, effectively disabling all interrupts.

7	6	5	4	3	2	1	0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reserved 0	I^2C Interrupt Enable	GPIO/HAPI Interrupt Enable	DAC Interrupt Enable	USB Hub Interrupt Enable	1.024-ms Interrupt Enable	128-µs Interrupt Enable	USB Bus RST Interrupt Enable

Figure 15-1. Global Interrupt Enable Register 0x20 (read/write)

7	6	5	4	3	2	1	0
			R/W	R/W	R/W	R/W	R/W
Reserved 0	Reserved 0	Reserved 0	EPB1 Interrupt Enable	EPB0 Interrupt Enable	EPA2 Interrupt Enable	EPA1 Interrupt Enable	EPA0 Interrupt Enable

Figure 15-2. USB Endpoint Interrupt Enable Register 0x21 (read/write)

Pending interrupt requests are recognized during the last clock cycle of the current instruction. When servicing an interrupt, the hardware will first disable all interrupts by clearing the Interrupt Mask bit in the Processor Status and Control Register. Next, the interrupt latch of the current interrupt is cleared. This is followed by a CALL instruction to the ROM address associated with the interrupt being serviced (i.e., the Interrupt Vector). 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 automatically stored onto the Program Stack by the CALL instruction as part of the interrupt acknowledge process. The user firmware is responsible for insuring that the processor state is preserved and restored during an interrupt. The PUSH A instruction should be used as the first command in the ISR to save the accumulator value and the POP A instruction should be used just before the RETI instruction to restore the accumulator value. The program counter, CF, and ZF are restored and interrupts are enabled when the RETI instruction is executed.

15.1 Interrupt Vectors

The Interrupt Vectors supported by the USB Controller are listed in *Table 15-1*. The lowest-numbered interrupt (USB Bus Reset interrupt) has the highest priority, and the highest-numbered interrupt (I^2C interrupt) has the lowest priority. 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 2 bytes long, the interrupt vectors occupy 2 bytes.

Table 15-1. Interrupt Vector Assignments

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

15.2 Interrupt Latency

Interrupt latency can be calculated from the following equation:

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

For example, if a 5 clock cycle instruction such as JC is being executed when an interrupt occurs, the first instruction of the Interrupt Service Routine will execute a 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.

15.3 USB Bus Reset Interrupt

The USB Controller recognizes a USB Reset when a Single Ended Zero (SE0) condition persists for 12–16 µs (the Reset may be recognized for an SE0 as short as 12 µs, but will always be recognized for an SE0 longer than 16 µs). SE0 is defined as the condition in which both the D+ line and the D– line are LOW. Bit 5 of the Status and Control Register will be set to record this event. If the USB reset happens while the device is suspended (such as after a POR), the suspend condition will be cleared and the clock oscillator will be restarted.

15.4 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 timer interrupts first or the suspend request first.

15.5 USB Endpoint Interrupts

There are five USB endpoint interrupts, one per endpoint. A USB endpoint interrupt is generated after the USB host writes to a USB endpoint FIFO or after the USB controller sends a packet to the USB host.

15.6 USB Hub Interrupt

A USB hub interrupt is generated by the hardware after a connect/disconnect change, babble, or a resume event is detected by the USB repeater hardware. The babble and resume events are additionally gated by the corresponding bits of the Hub Port Enable Register (0x49). The connect/disconnect event on a port will not generate an interrupt if the SIE does not drive the port, i.e., the port is being forced.

15.7 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.

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

15.8 GPIO/HAPI Interrupt

Each of the GPIO pins can generate an interrupt, if enabled. The interrupt polarity can be programmed for each GPIO port as part of the GPIO configuration. All of the GPIO pins share a single interrupt vector, which means the firmware will need to read the GPIO ports with enabled interrupts to determine which pin or pins caused an interrupt.

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

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 (0x20) when HAPI is enabled.The settings of the GPIO bit interrupt enables on ports/bits not used by HAPI will 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 8-1*.

15.9 I^2C Interrupt

The I^2C interrupt serves two purposes. First, it informs the system that the I^2C hardware requires attention. Second, it indicates that the contents of the I^2C Control & Status Register (0x28) are valid and that the control bits may be written to. The control bits may change during I^2C transactions prior to the generation of an interrupt. When this happens the firmware must wait for the interrupt to perform any writes to the Control Register as certain actions are taken when an interrupt is generated that may again change the value of the Control Register and overwrite the data written there by the firmware. When enabled, the I^2C state machines will generate interrupts on the following conditions:

1. When the slave receives a byte of data and the master is either inactive or has lost arbitration. The address bit will be set if this is the first byte since the start or restart signal was sent by the external master. The microcontroller must write data into the data register if necessary then set the Ack, Xmit, and Continue/Busy bits appropriately.
2. When the slave is in receive mode and detects a stop bit indicating the end of a transaction the Received Stop bit will be set and an interrupt will be generated.
3. When the slave transmits a byte of data. The Ack bit of the I^2C Control & Status register will indicate if the master that requested the byte acknowledged the byte. The microcontroller needs to write the next byte of data into the data register and then set the Xmit and Continue/Busy bits as required.
4. When the master sends a byte of data. The setting of the Master bit initiates the master address transmit so the microcontroller must place the data (address of target) in the data register prior to setting the Master bit. When the transmit is done the Ack bit will indicate if the external target acknowledged the address. The microcontroller must set the Xmit, Master, and Continue/Busy bits appropriately. Clearing the Master bit will cause the master state machine to issue a stop signal to the I^2C bus and return to the idle state.
5. When the master receives a byte of data. The microcontroller must set the Ack and Continue/Busy bits appropriately. Clearing the Master bit at the same time will cause the master state machine to issue a stop signal to the I^2C bus and return to the idle state.
6. When the master loses arbitration. This condition clears the Master bit and sets the Arbitration Lost bit immediately and then waits for a stop signal on the I^2C bus to generate the interrupt.

The Continue bit is cleared prior to interrupt conditions 1 to 4 and the state machines require that this bit be set by the microcontroller to acknowledge the interrupt condition and indicate that the data register and status/control bits have been properly set up for the state machine to continue.

Following an interrupt from the master mode controller the firmware may perform only one write to the control register 0x28 that sets the Continue bit (register 0x28, bit 6, write) without checking the value of the Busy bit (register 0x28, bit 6, read). Following the first write that sets the Continue bit, the hardware may engage a write inhibit function for a short period of time.

16.0 USB Overview

The USB hardware includes a USB Hub repeater with one upstream and four downstream ports. The USB Hub repeater interfaces to the microcontroller through a high-speed serial interface engine (SIE). An external series resistor of R_{ext}=20 Ω (±5%) must be placed in series with all the upstream and downstream USB outputs in order to meet the USB driver requirement, as defined by the USB specification. The CY7C66xxx microcontroller can provide the functionality of a compound device consisting of a USB hub and permanently attached functions.

16.1 USB Serial Interface Engine (SIE)

The SIE allows the CY7C66xxx microcontroller to communicate with the USB host through the USB repeater portion of the hub. 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

16.2 USB Enumeration

The enumeration sequence in a compound device begins with the hub and finishes with the device functions. The hub is enumerated first. Then the integrated compound function is enumerated. Then the Hub connection status is read to determine which, if any, of the downstream ports need to be enumerated. Following is a very brief summary of the enumeration process of a USB function by the host. For a detailed description of the enumeration process, refer to the latest USB specifications published by USBIF, or check their web site at www.usb.org.

1. The host computer sends a SETUP packet followed by a DATA packet to USB address 0 requesting the Device descriptor.
2. The USB Controller decodes the request and retrieves its Device descriptor from the program memory space.
3. The host computer performs a control read sequence and the USB Controller responds by sending the Device descriptor over the USB bus.
4. After receiving the descriptor, the host computer sends a SETUP packet followed by a DATA packet to address 0 assigning a new USB address to the device.
5. The USB Controller 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. The USB Controller decodes the request and retrieves the Device descriptor from the program memory.
8. The host performs a control read sequence and the USB Controller responds by sending its Device descriptor over the USB bus.
9. The host generates control reads to the USB Controller to request the Configuration and Report descriptors.
10. The USB Controller retrieves the descriptors from its program space and returns the data to the host over the USB.
11. Enumeration is complete after the host has received all the descriptors.

17.0 USB Hub

A USB hub is required to support:

- Connectivity behavior
- Bus fault detection and recovery
- Full-/Low-speed device support

These features are mapped onto a hub repeater and a hub controller. The hub controller is supported by the processor integrated into the CY7C66011/12/13 and CY7C66111/12/13 microcontrollers. The hardware in the hub repeater detects whether a USB device is connected to a downstream port and the interface speed of the downstream device. The connection to a downstream port is through a differential signal pair (D+ and D–). Every downstream port provided by the hub requires an external 15-kΩ resistors from each signal line to ground. The effect is when a device is not present, the hub will read a LOW (zero) on both D+ and D–. This condition, referred to as a single-ended-zero (SE0), can be used to identify the “no connect” state.

17.1 Connecting/Disconnecting a USB Device

A low-speed (1.5 Mbps) USB device will have a pull-up resistor on the D– pin. At connect time, the bias resistors set the signal levels on the D+ and D– lines. When a low-speed device is connected to a hub port, the hub will see a LOW on D+ and a HIGH on D–. This tells the hub repeater that a low-speed device is connected to the port. The hub repeater sets a connect bit in the Hub Port Connect Status register for the downstream port. The hub repeater also sets a bit in the Hub Port Speed register to

indicate this port is low speed. Then the hub repeater generates a “Hub Interrupt”, if the interrupt is enabled, to notify the microcontroller that there has been a change in the Hub downstream status.

A high-speed (12 Mbps) USB device will have a pull-up resistor from the D+ pin. That means the hub will see a HIGH on D+ and a LOW on D–. This tells the hub that a full-speed device is connected to the port. The hub repeater sets a connect bit in the Hub Port Connect Status register, clears a bit in the Hub Port Speed register (high-speed), and generates a Hub Interrupt, if enabled, to ask the microcontroller to read the Hub status.

When a USB device is disconnected from the Hub, the differential signal pair will eventually reach a single-ended-zero state when the hub is not driving data to the port. The hub repeater will recognize the SE0 as a disconnect and clear the corresponding bit in the Hub Port Connect Status register. Then generates a Hub interrupt, if enabled, to the microcontroller.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Port 4	Port 3	Port 2	Port 1

Figure 17-1. Hub Port Connect Status 0x48 (read only)

The Hub Port Connect Status register is cleared to zero by reset, then set to match the hardware configuration by the hub repeater hardware. The Reserved bits [7:4] should always read as ‘0’ to indicate no connection.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Port 4	Port 3	Port 2	Port 1

Figure 17-2. Hub Port Speed 0x4A (read only)

The Hub Port Speed register is cleared to zero by reset, then set to match the hardware configuration by the hub repeater hardware. A zero setting indicates a high speed device. The Reserved bits [7:4] should always read as ‘0.’

17.2 Enabling/Disabling a USB Device

After a USB device connection has been detected and reported to the microcontroller, the firmware needs to update status bits in the hub status data structure that is polled periodically by the USB host. The host responds by sending a packet that instructs the hub to enable the downstream port. The microcontroller responds to the packet by setting a bit in the Hub Port Enable register for the downstream port. The hub repeater hardware responds to an enable bit in the Hub Port Enable register by enabling the downstream port. Then the USB host enumerates the device that was just detected.

When a USB device disconnection has been detected and reported to the microcontroller, the firmware needs to update status bits in the hub status data structure that is polled periodically by the USB host. The host responds by sending a packet that instructs the hub to disable the downstream port. The microcontroller responds to the packet by clearing a bit in the Hub Port Enable register for the downstream port. The hub repeater hardware responds to the deassertion of the enable bit by disabling the downstream port.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Port 4	Port 3	Port 2	Port 1

Figure 17-3. Hub Port Enable Register 0x49 (read/write)

The Hub Port Enable register is cleared to zero by reset to disable all downstream ports as the default condition. A bit will also be cleared by the hub interface if babble is detected on that downstream port.

17.3 Hub Downstream Ports Status and Control

Data transfer on hub downstream ports is controlled according to the bit settings of the Hub Downstream Ports Control Register (0x4B). Each downstream port is controlled by two bits. the control bits are defined in the table below. The Hub Downstream Ports Control Register is cleared upon reset.

7	6	5	4	3	2	1	0
Port 4(1)	Port 4(0)	Port 3(1)	Port 3(0)	Port 2(1)	Port 2(0)	Port 1(1)	Port 1(0)

Figure 17-4. Hub Downstream Ports Control Register 0x4B (read/write)

Table 17-1. Control Bit Definition for Downstream Ports

Control Bits bit1 bit 0	Control Action
0 0	Not Forcing
0 1	Force Differential '1'
1 0	Force Differential '0'
1 1	Force SE0 state

The data received on downstream ports can be read through the HUB Ports SE0 Status Register (0x4F) and the Hub Ports Data Register (0x50). The data read through the Hub Ports Data Register is the differential data only and not dependent on the settings of the Hub Port Speed Register (0x4A). When the SE0 condition is sensed on a downstream port, the corresponding bits of the Hub Ports Data Register are undefined. Hub Ports SE0 Status Register and Hub Ports Data Register are cleared upon reset.

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Port 4	Port 3	Port 2	Port 1

Figure 17-5. Hub Ports SE0 Status Register 0x4F (read only)

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Port 4	Port 3	Port 2	Port 1

Figure 17-6. Hub Ports Data Register 0x50 (read only)

The Hub Ports Suspend Register (0x4D) and Hub Ports Resume Status Register (0x4E) indicate the suspend and resume conditions on downstream ports. If the enable Device Remote Wakeup bit (bit 7) of the Hub Ports Suspend Register (0x4D) is set, a downstream device will wake up the hub from suspend upon a connect or a disconnect event. A resume bit is set automatically by the hardware when the SIE section of the microcontroller detects a resume condition on a suspended downstream port. The resume condition is a differential '1' for a low-speed device and a differential '0' for a high-speed device.

7	6	5	4	3	2	1	0
Device Remote Wakeup	Reserved	Reserved	Reserved	Suspend 4	Suspend 3	Suspend 2	Suspend 1

Figure 17-7. Hub Ports Suspend Register 0x4D (read/write)

7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	Reserved	Resume 4	Resume 3	Resume 2	Resume 1

Figure 17-8. Hub Ports Resume Status Register 0x4E (read only)

17.4 USB Upstream Port Status and Control

USB status and control is regulated by the USB Status and Control Register located at I/O address 0x1F as shown in *Figure 17-9*. This is a read/write register. All bits in the register are cleared during reset.

7	6	5	4	3	2	1	0
R/W	R/W	R	R	R/W	R/W	R/W	R/W
Endpoint Size	Endpoint Mode	D+	D–	Bus Activity	Control Bit 2	Control Bit 1	Control Bit 0

Figure 17-9. USB Status and Control Register 0x1F (read/write)

The Bus Activity bit is a "sticky" bit that indicates if any non-idle USB event has occurred on the upstream USB port. 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.

Table 17-2. 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

18.0 USB Compound Device

The compound device (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.

18.1 USB Device Addresses

The USB Controller provides two USB Device Address Registers (A and B). Upon reset and under default conditions, Device A has three endpoints while Device B has two endpoints. The USB Device Address Register contents are cleared during a reset, setting the USB device addresses to zero and marking these addresses as disabled. *Figure 18-1* shows the format of the USB 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

Figure 18-1. USB Device Address Registers 0x10, 0x40 (read/write)

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 these addresses. The Device Address in bits [6:0] are set during the USB enumeration process to a non-zero address assigned by the USB host.

18.2 USB Device Endpoints

The CY7C66xxx controller supports up to two addresses and five endpoints for communication with the host. The configuration of these endpoints, and associated FIFOs, is controlled by bits [7,6] of the of the USB Status and Control Register (0x1F) Bit 7 controls the size of the endpoints and bit 6 controls the number of addresses. These configuration options are detailed in *Table 18-1*.

Table 18-1. Memory Allocation for Endpoints

I/O status [7,6]	[0,0]			[1,0]			[0,1]			[1,1]		
	Label	Address	Size	Label	Address	Size	Label	Address	Size	Label	Address	Size
	EPB1	0xD8	8	EPB0	0xA8	8	EPA4	0xD8	8	EPA4	0xA8	8
	EPB0	0xE0	8	EPB1	0xB0	8	EPA3	0xE0	8	EPA3	0xB0	8
	EPA2	0xE8	8	EPA0	0xB8	8	EPA2	0xE8	8	EPA0	0xB8	8
	EPA1	0xF0	8	EPA1	0xC0	32	EPA1	0xF0	8	EPA1	0xC0	32
	EPA0	0xF8	8	EPA2	0xE0	32	EPA0	0xF8	8	EPA2	0xE0	32

All USB devices are required to have a control endpoint 0 (EPA0 and EPB0) that is used to initialize and control each USB address. Endpoint 0 provides access to the device configuration information and allows generic USB status and control accesses. Endpoint 0 is bidirectional as the USB controller can both receive and transmit data.

The endpoint mode registers are cleared during reset. The EPA0 and EPB0 endpoint mode registers use the format shown in *Figure 18-2*.

Endpoint 0 SETUP Received	Endpoint 0 IN Received	Endpoint 0 OUT Received	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0

Figure 18-2. USB Device Endpoint Zero Mode Registers 0x12 and 0x42, (read/write)

Bits[7:5] in the endpoint 0 mode registers (EPA0 and EPB0) are “sticky” status bits that are set by the SIE to report the type of token that was most recently received by the corresponding device address. The sticky bits must be cleared by firmware as part of the USB processing.

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

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

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

Firmware must do a 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 ‘SETUP’ bit is set, the CPU cannot write to the endpoint zero DMA buffers. This prevents an incoming SETUP transaction from conflicting with a previous In data buffer filling operation by firmware. Reference *Table 18-1* for the appropriate endpoint zero memory locations.

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 *Table 19-1*. Additional information on the mode bits can be found in *Table 19-2* and *Table 19-3*.

STALL	Not Used	Not Used	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0

Figure 18-3. USB Non-Control Device Endpoint Mode Registers 0x14, 0x16, 0x44, (read/write)

The mode bits (bits [3:0]) of the Endpoint Mode Register control how the endpoint responds to USB bus traffic. The mode bit encoding is shown in *Table 19-1*. If STALL bit (bit 7), the SIE will stall an OUT packet if the mode bits are set to ACK-IN, and the SIE will stall an IN packet if the mode bits are set to ACK-OUT. For all other modes the STALL bit must be a LOW. For non-zero endpoints, bits [6:5] are reserved.

The format of the endpoint Device counter registers is shown below:

Data 0/1 Toggle	Data Valid	Byte Count Bit 5	Byte Count Bit 4	Byte Count Bit 3	Byte Count Bit 2	Byte Count Bit 1	Byte Count Bit 0

Figure 18-4. USB Device Counter Registers 0x11, 0x13, 0x15, 0x41, 0x43 (read/write)

Bits 0 to 5 indicate the number of data bytes to be transmitted during an IN packet, valid values are 0 to 32 inclusive. Data Valid bit 6 is used for OUT and SETUP tokens only. Data 0/1 Toggle bit 7 selects the DATA packet’s toggle state: 0 for DATA0, 1 for DATA1.

19.0 Truth Tables

Table 19-1. USB Register Mode Encoding

Mode	Encoding	Setup	In	Out	Comments
Disable	**0000**	ignore	ignore	ignore	Ignore all USB traffic to this endpoint
Nak In/Out	**0001**	accept	NAK	NAK	Forced from Setup on Control endpoint, from modes other than 0000
Status Out Only	**0010**	accept	stall	check	For Control endpoints
Stall In/Out	**0011**	accept	stall	stall	For Control endpoints
Ignore In/Out	**0100**	accept	ignore	ignore	For Control endpoints
Isochronous Out	**0101**	ignore	ignore	always	For Isochronous endpoints
Status In Only	**0110**	accept	TX 0	stall	For Control Endpoints
Isochronous In	**0111**	ignore	TX cnt	ignore	For Isochronous endpoints
Nak Out	**1000**	ignore	ignore	NAK	An ACK from mode 1001 --> 1000
Ack Out(STALL[5]=0) Ack Out(STALL[5]=1)	**1001** **1001**	ignore ignore	ignore stall	ACK ACK	This mode is changed by SIE on issuance of ACK --> 1000
Nak Out - Status In	**1010**	accept	TX 0	NAK	An ACK from mode 1011 --> 1010
Ack Out - Status In	**1011**	accept	TX 0	ACK	This mode is changed by SIE on issuance of ACK --> 1010
Nak In	**1100**	ignore	NAK	ignore	An ACK from mode 1101 --> 1100
Ack IN(STALL[5]=0) Ack IN(STALL[5]=1)	**1101** **1101**	ignore ignore	TX cnt TX cnt	ignore stall	This mode is changed by SIE on issuance of ACK --> 1100
Nak In - Status Out	**1110**	accept	NAK	check	An ACK from mode 1111 --> 111 Ack In - Status Out
Ack In - Status Out	**1111**	accept	TX cnt	Check	This mode is changed by SIE on issuance of ACK -->1110

Note:
5. STALL bit is the bit 7 of the USB Non-Control Device Endpoint Mode registers. Refer to Section 18.2 for more explanation.

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

A disabled endpoint will remain such until firmware changes it, and all endpoints reset to disabled.

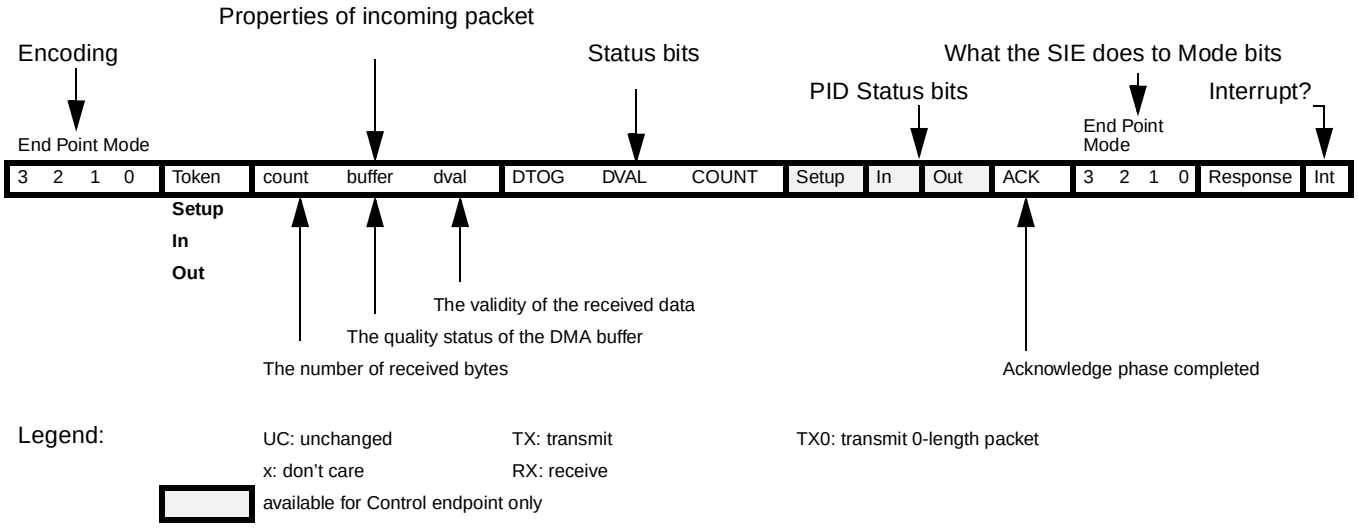
Any Setup packet to an enabled and accepting endpoint will be changed by the SIE to 0001 (NAKing). Any mode which indicates the acceptance of a Setup will acknowledge it.

Most modes that control transactions involving an ending ACK will be changed by the SIE to a corresponding mode which NAKs follow on packets.

A Control endpoint has three extra status bits for PID (Setup, In and Out), but must be placed in the correct mode to function as such. Also a non-Control endpoint can be made to act as a Control endpoint if it is placed in a non appropriate mode.

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

Table 19-2. Decode table for*Table 19-3*: "Details of Modes for Differing Traffic Conditions"



The response of the SIE can be summarized as follows:

1. The SIE will only respond to valid transactions, and will ignore non-valid ones;
2. The SIE will generate 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 visa 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 19-3. Details of Modes for Differing Traffic Conditions

End Point Mode											PID				Set End Point Mode					
3	2	1	0	token	count	buffer	dval	DTOG	DVAL	COUNT	Setup	In	Out	ACK	3	2	1	0	response	int
Setup Packet (if accepting)																				
See *Table 19-1*				Setup	<= 10	data	valid	updates	1	updates	1	UC	UC	1	0	0	0	1	ACK	yes
See *Table 19-1*				Setup	> 10	junk	x	updates	updates	updates	1	UC	UC	UC	NoChange				ignore	yes
See *Table 19-1*				Setup	x	junk	invalid	updates	0	updates	1	UC	UC	UC	NoChange				ignore	yes
Disabled																				
0	0	0	0	x	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				ignore	no
Nak In/Out																				
0	0	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	1	UC	NoChange				NAK	yes
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
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

Table 19-3. Details of Modes for Differing Traffic Conditions (continued)

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	> 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																				
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 (STALL[5] = 0)	no
1	0	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				Stall (STALL[5] = 1)	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 (STALL[5] = 0)	no
1	1	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	NoChange				stall (STALL[5] = 1)	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

20.0 Absolute Maximum Ratings

Storage Temperature ..... –65°C to +150°C
Ambient Temperature with Power Applied ..... –0°C to +70°C
Supply voltage on V_{CC} relative to V_{SS} .....–0.5V to +7.0V
DC Input Voltage..... –0.5V to +V_{CC}+0.5V
DC Voltage applied to Outputs in High Z State..... –0.5V to +V_{CC}+0.5V
Power Dissipation .....500 mW
Static Discharge Voltage ..... >2000V
Latch-up Current ..... >200 mA
Max Output Sink Current into Port 0, 1, 2, 3, and DAC[1:0] Pins (V_{out} = 2.0V) ..... 60 mA
Max Output Sink Current into DAC[7:2] Pins (V_{out} = 2.0V) ..... 10 mA
Max Output Source Current from Port 0, 1, 2, 3 (V_{out} = 2.0V) ..... 30 mA
Max I_{CC}..... 50 mA PLUS any current sourced out of GPIO/DAC ports
Max I_{ref}..... 5 mA PLUS 18 mA per USB cable (upstream or downstream) driven
(Note: 18 mA per cable is based on consumption of a transition every 2 high-speed bit times on average)

21.0 Electrical Characteristics

Fosc = 6 MHz; Operating Temperature = 0 to 70°C, V_{CC} = 4.0 to 5.25 Volts

	Parameter	Min	Max	Units	Conditions
	General				
V_{ref}	Reference Voltage	3.15	3.45	V	3.3V ±5%
V_{pp}	Programming Voltage (disabled)	–0.4	0.4	V	
I_{SB1}	Supply Current—Suspend Mode		50	µA	Oscillator off, D+> V_{oh} min
I_{il}	Input Leakage Current		1	µA	any pin
	USB Interface				
V_{di}	Differential Input Sensitivity	0.2		V	|(D+)–(D–)|
V_{cm}	Differential Input Common Mode Range	0.8	2.5	V	
V_{se}	Single Ended Receiver Threshold	0.8	2.0	V	
C_{in}	Transceiver Capacitance		20	pF	
I_{lo}	Hi-Z State Data Line Leakage	–10	10	µs	0 V < V_{in}<3.3 V
	Power On Reset				
V_{rst}	POR Voltage	2.0	3.4	V	Note 6
t_{vccs}	V_{CC} Ramp Rate	0	100	ms	linear ramp V_{CC}: 0 to Operating Voltage
	USB Upstream/Downstream Port				
V_{oh}	Static Output High	2.8	3.6	V	15 kΩ ± 5% to Gnd[7]
V_{ol}	Static Output Low		0.3	V	
Z_o	USB Driver Output Impedance	28	44	Ω	Including R_{ext}= 20Ω
	General Purpose I/O				
R_{up}	Pull-up Resistance (typical 14 kΩ)	8.0	24.0	kΩ	
V_{ith}	Input Threshold voltage	45%	65%	V_{CC}	All ports, low to high edge
V_H	Input Hysteresis voltage	6%	12%	V_{CC}	All ports, high to low edge
I_{sink0}	Port 0,1,2 Sink Current (typical 7 ma)	3.5	10.6	mA	V_{out} = 2.0V DC
I_{sink3}	Port 3 Sink Current (typical 12 ma)	8	24	mA	V_{out} = 2.0V DC
I_{oh}	Source Current	1.9	7.5	mA	V_{oh} = 2.4V (all ports 0,1,2,3)
	DAC Interface				
R_{up}	Pull-up Resistance (typical 14 kΩ)	8.0	20.0	kΩ	
$I_{sink0(0)}$	DAC[7:2] Sink current (0)	0.1	0.3	mA	V_{out} = 2.0 V DC
$I_{sink0(F)}$	DAC[7:2] Sink current (F)	0.5	1.5	mA	V_{out} = 2.0 V DC
$I_{sink1(0)}$	DAC[1:0] Sink current (0)	1.6	4.8	mA	V_{out} = 2.0 V DC
$I_{sink1(F)}$	DAC[1:0] Sink current (F)	8	24	mA	V_{out} = 2.0 V DC
I_{range}	Programmed Isink Ratio: max/min	4	6		V_{out} = 2.0 V DC[8]
T_{ratio}	Tracking Ratio DAC[1:0] to DAC[7:2]	14	20		V_{out} = 2.0V[9]
$I_{sinkDAC}$	DAC Sink Current	1.6	4.8	mA	V_{out} = 2.0V DC
I_{lin}	Differential Nonlinearity		0.5	lsb	Port 0 or Port 1

Notes:

6. Power on Reset will occur until the voltage on V increases above V_{rst}.
7. Rx: With external resistor, 1.5 kΩ, 5%, to 2.8V.
8. Irange: Isinkn(15)/ Isinkn(0) for the same pin.
9. T_{ratio} = Isink1[1:0](n)/Isink0[7:2](n) for the same n, programmed.

22.0 Switching Characteristics

Parameter	Description	Min.	Max.	Unit
	Clock Source			
f_{OSC}	Clock Rate	6 ± 0.25%		MHz
t_{cyc}	Clock Period	166.25	167.08	nsec
t_{CH}	Clock HIGH time	0.45 t_{CYC}		ns
t_{CL}	Clock LOW time	0.45 t_{CYC}		ns
	USB High Speed Signaling			
t_r	Transition Rise Time[10]	4	20	ns
t_f	Transition Fall Time[10]	4	20	ns
t_{rfm}	Rise / Fall Time Matching; (t_r/t_f)	90	110	%
tdrate	Full Speed Date Rate	12 ± 0.25%		Mb/s
	DAC Interface			
t_{sink}	Current Sink Response Time		0.8	µs
	HAPI Read Cycle Timing			
t_{RD}	Read Cycle Time	59		ns
$t_{\overline{OED}}$	OE LOW to Data Valid		40	ns
t_{OEZ}	OE HIGH to Data High-Z		19	ns
$t_{\overline{OEDR}}$	OE LOW to Data_Ready LOW	15	52	ns
	HAPI Write Cycle Timing			
t_{WR}	Write Cycle Time	20		ns
t_{DSTB}	Data Valid to STB HIGH (Data Set-up Time)	5		ns
t_{STBZ}	STB HIGH to Data High-Z (Data Hold Time)	15		ns
$t_{\overline{STBLE}}$	STB LOW to Latch_Empty LOW	15	52	ns
	Timer Signals			
t_{int1}	Internal Timer #1 Interrupt Period	128	128	µs
t_{int2}	Internal Timer #2 Interrupt Period	1.024	1.024	ms
t_{watch}	WatchDog Timer Period	8.192	14.336	ms

Note:
10. Per Table 7-5 of revision 1.0 of USB specification, for Cload of 50 pF.

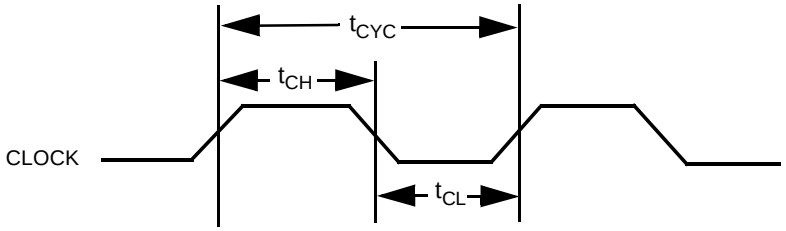


Figure 22-1. Clock Timing

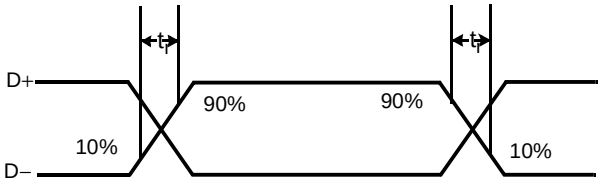


Figure 22-2. USB Data Signal Timing

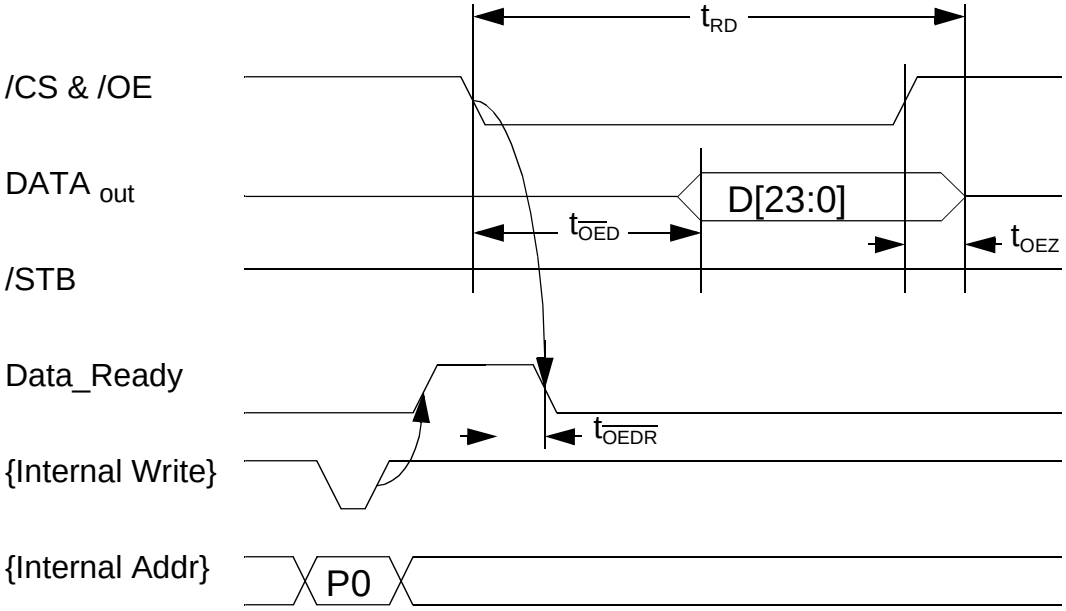


Figure 22-3. HAPI Read by External Interface from USB Microcontroller

Table 22-1. HAPI Read Cycle Timing

Parameters	Description	Minimum	Maximum
t_{RD}	Read Cycle Time	59 ns	
$t_{\overline{OED}}$	OE LOW **and** CS LOW to Data Valid		40 ns
t_{OEZ}	OE HIGH **or** CS HIGH to Data High-Z		19 ns
$t_{\overline{OEDR}}$	OE LOW **and** CS LOW to Data_Ready LOW	15 ns	52 ns

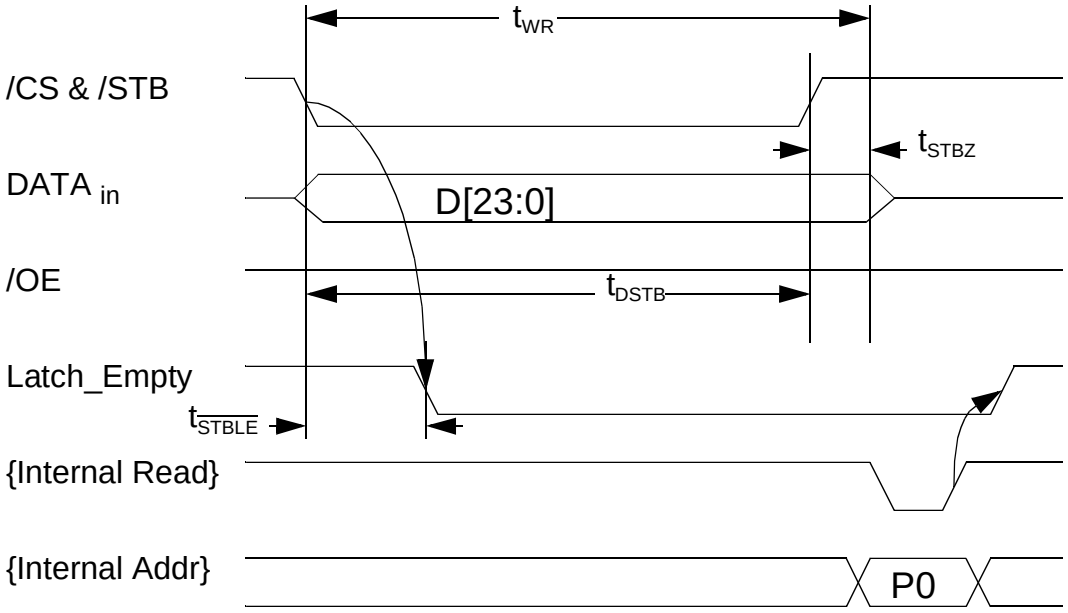


Figure 22-4. HAPI Write by External Device to USB Microcontroller

Table 22-2. HAPI Write Cycle Timing

Parameters	Description	Minimum	Maximum
t_{WR}	Write Cycle Time	20 ns	
t_{DSTB}	Data Valid **and** STB LOW **and** CS LOW to STB HIGH **or** CS HIGH (Data Set-Up Time)	5 ns	
t_{STBZ}	STB HIGH **or** CS HIGH to Data High-Z (Data Hold Time)	15 ns	
$t_{\overline{STBLE}}$	STB LOW **and** CS LOW to Latch_Empty LOW	15 ns	52 ns

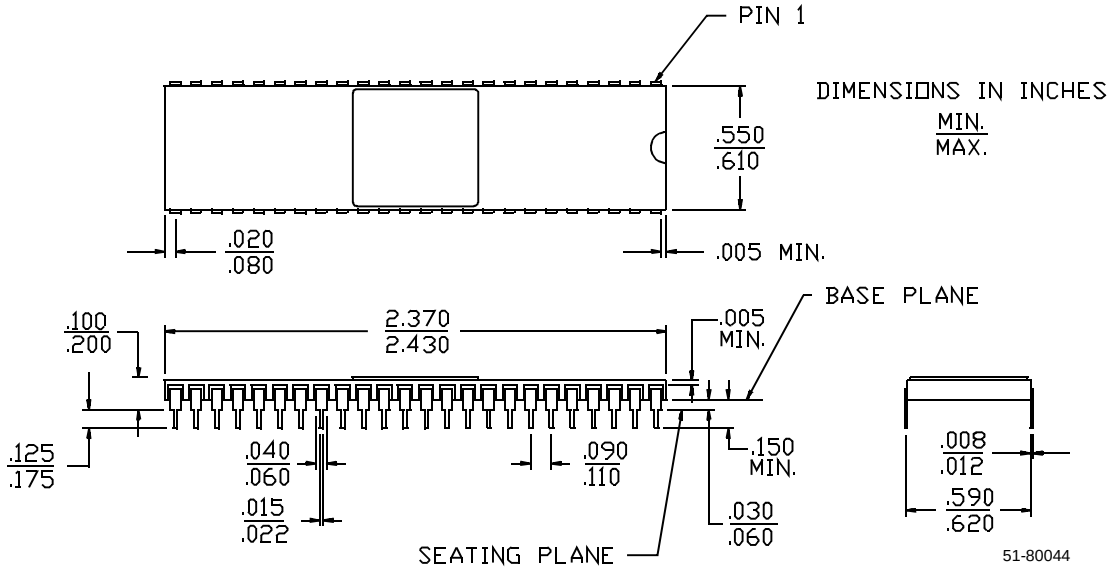
23.0 Ordering Information

Ordering Code	EPROM Size	Package Name	Package Type	Operating Range
CY7C66011-PVC	4 KB	O48	48-Pin (300-Mil) SSOP	Commercial
CY7C66011-PC	4 KB	P25	48-Pin (600-Mil) PDIP	Commercial
CY7C66012-PVC	6 KB	O48	48-Pin (300-Mil) SSOP	Commercial
CY7C66012-PC	6 KB	P25	48-Pin (600-Mil) PDIP	Commercial
CY7C66013-PVC	8 KB	O48	48-Pin (300-Mil) SSOP	Commercial
CY7C66013-PC	8 KB	P25	48-Pin (600-Mil) PDIP	Commercial
CY7C66013-WVC	8 KB	D26	48-Pin Windowed SideBraze	Commercial
CY7C66111-PVC	4 KB	O56	56-Pin (300-Mil) SSOP	Commercial
CY7C66112-PVC	6 KB	O56	56-Pin (300-Mil) SSOP	Commercial
CY7C66113-PVC	8 KB	O56	56-Pin (300-Mil) SSOP	Commercial

Document #:38-00591-C

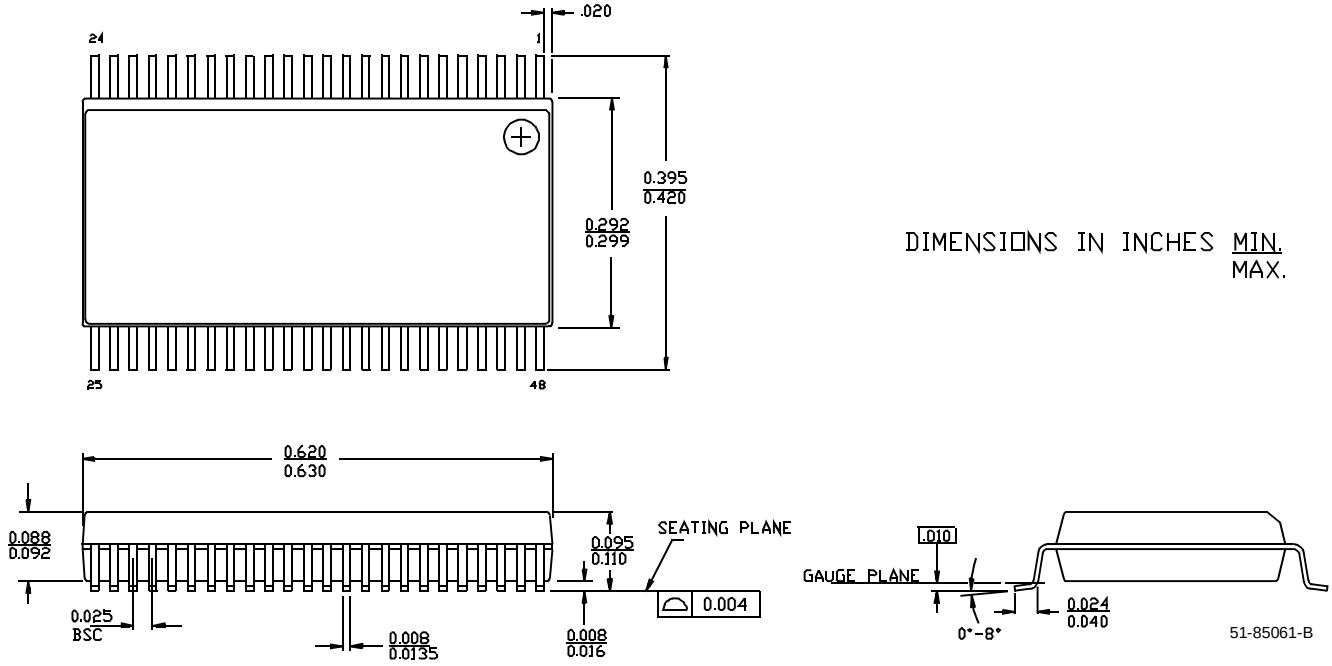
24.0 Package Diagrams

48-Lead (600-Mil) Sidebraze DIP D26
MIL-STD-1835 D-14 Config. C

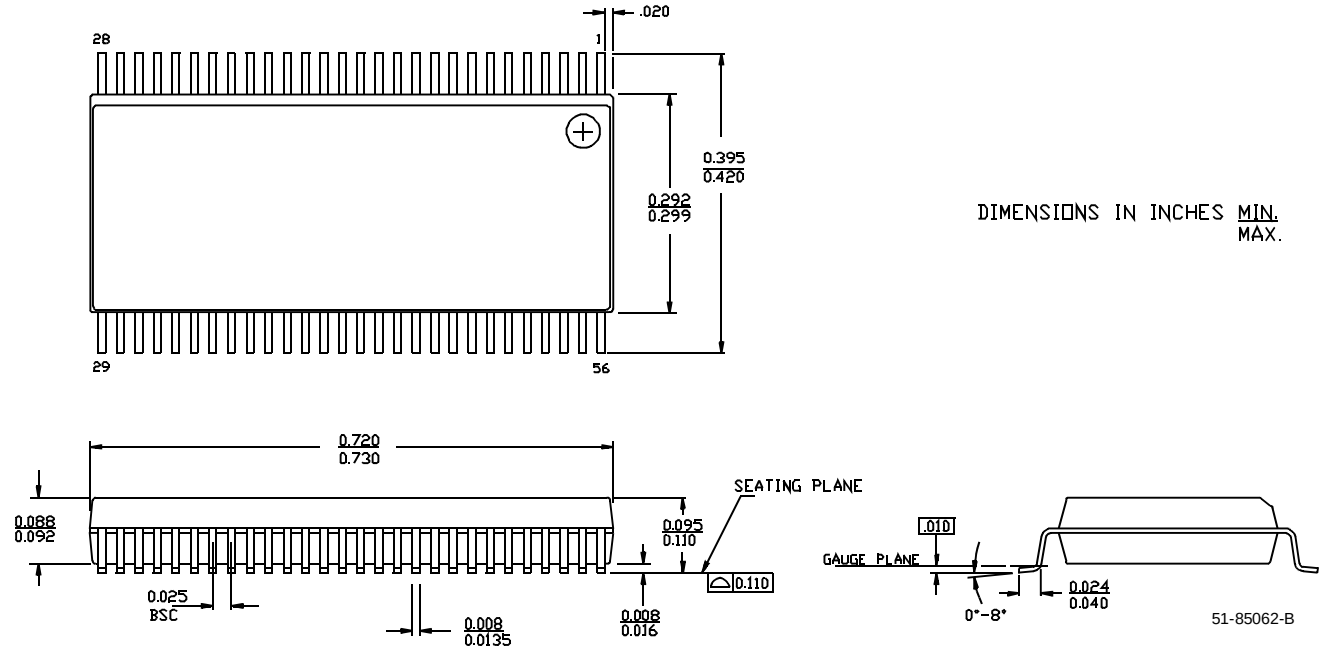


24.0 Package Diagrams (continued)

48-Lead Shrunk Small Outline Package O48

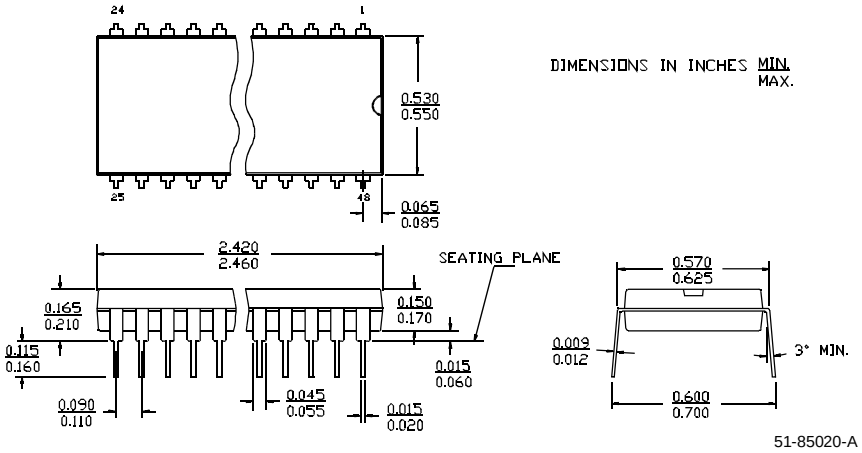


56-Lead Shrunk Small Outline Package O56



24.0 Package Diagrams (continued)

48-Lead (600-Mil) Molded DIP P25



51-85020-A