Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!


## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

CY7C66013C, CY7C66113C

# Full Speed USB (12 Mbps) Peripheral Controller with Integrated Hub

## Features

- Full speed USB peripheral microcontroller with an integrated USB hub
  - Well suited for USB compound devices such as a keyboard hub function
- 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
  - 8 KB of PROM
- Integrated Master and Slave $I^2C$ compatible controller (100 kHz) enabled through General Purpose I/O (GPIO) pins
- Hardware Assisted Parallel Interface (HAPI) for data transfer to external devices
- I/O ports
  - Three GPIO ports (Port 0 to 2) capable of sinking 8 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 is 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 CY7C66113C device
  - 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 1.1
  - Conforms to USB HID Specification, Version 1.1
  - Supports one or two device addresses with up to five 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 four downstream USB ports
  - GPIO pins provide individual power control outputs for each downstream USB port
  - GPIO pins provide individual port over current inputs for each downstream USB port
- Improved output drivers to reduce electromagnetic interference (EMI)
- Operating voltage from 4.0V–5.5V DC
- Operating temperature from 0°C–70°C
- CY7C66013C available in 48-pin SSOP (-PVXC) packages
- CY7C66113C available in 56-pin QFN or 56-pin SSOP (-PVXC) packages
- Industry standard programmer support

## Functional Overview

The CY7C66013C and CY7C66113C are compound devices with a full speed USB microcontroller in combination with a USB hub. Each device is suited for combination peripheral functions with hubs such as a keyboard hub function. The 8-bit one time programmable microcontroller with a 12 Mbps USB Hub supports as many as four downstream ports.

### GPIO

The CY7C66013C features 29 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 is configured as inputs with internal pull ups, open drain outputs, or traditional CMOS outputs. Ports 0 to 2 are rated at 8 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 are connected together to drive a single output for more drive current capacity. Additionally, each I/O pin is used to generate a GPIO interrupt to the microcontroller. All of the GPIO interrupts all share the same “GPIO” interrupt vector.

The CY7C66113C has 31 GPIO pins (P0[7:0], P1[7:0], P2[7:0], P3[6:0]).

### DAC

The CY7C66113C has an additional port P4[7:0] that features an additional eight programmable sink current I/O pins (DAC). 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 is 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 is individually programmed to one of sixteen values using dedicated Isink registers. DAC bits DAC[1:0] is used as high current outputs with a programmable sink current range of 3.2 to 16 mA (typical). DAC bits DAC[7:2] have a programmable current sink range of 0.2 to 1.0 mA (typical). Multiple DAC pins are connected together to drive a single output that requires more sink current capacity. Each I/O pin is used to generate a DAC interrupt to the microcontroller. Also, the interrupt polarity for each DAC I/O pin is individually programmable.

### Clock

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.

### Memory

The CY7C66013C and CY7C66113C have 8 KB of PROM.

### Power on Reset, Watchdog, and Free Running Timer

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

### $I^2C$ and HAPI Interface

The microcontroller communicates with external electronics through the GPIO pins. An $I^2C$ compatible interface accommodates a 100 kHz serial link with an external device. There is also a HAPI to transfer data to an external device.

### Timer

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

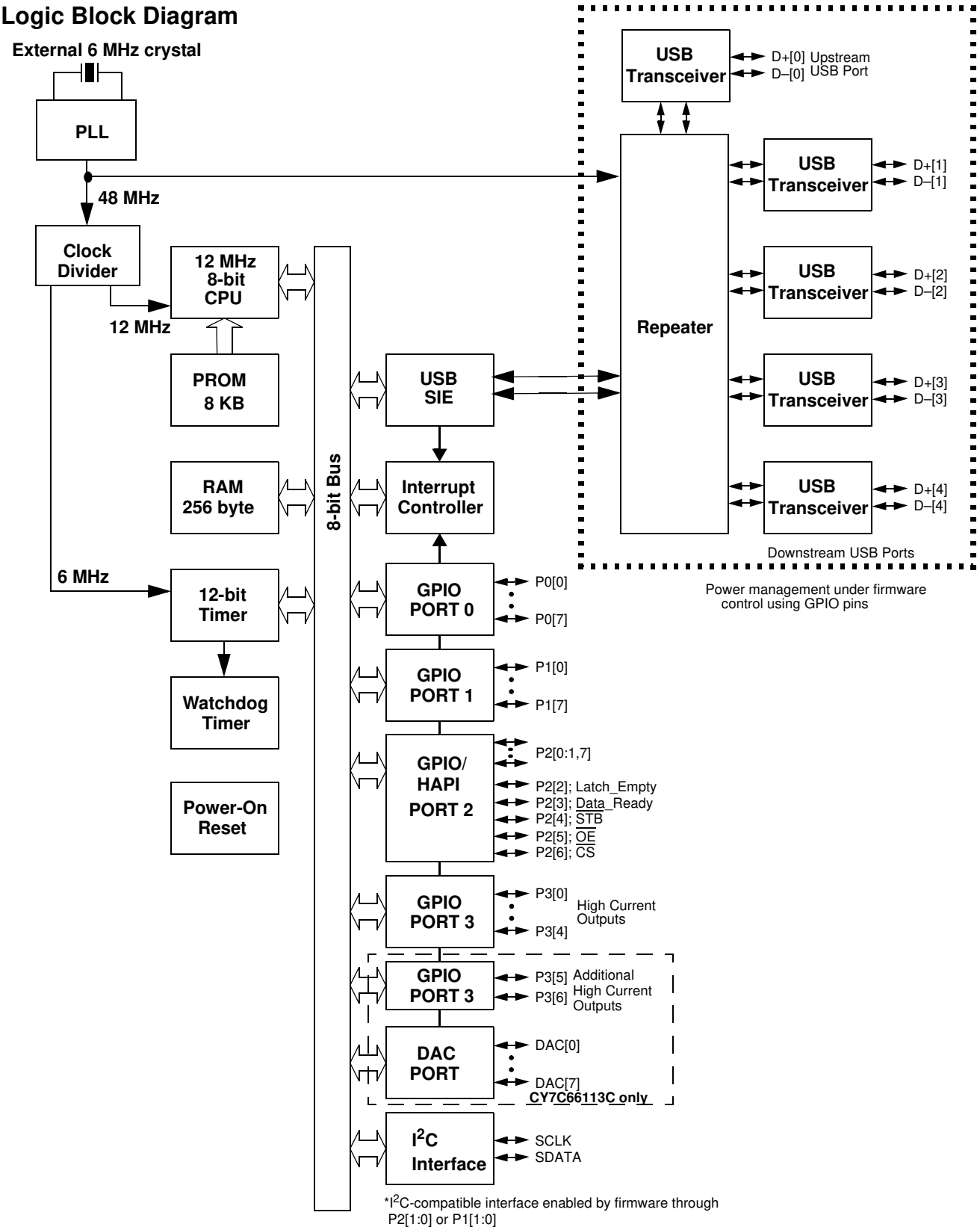
### Interrupts

The microcontroller supports eleven 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 $I^2C$ compatible master mode interface. The timer bits cause an interrupt (if enabled) when the bit toggles from LOW ‘0’ to HIGH ‘1.’ The USB endpoints interrupt after the USB host has written data to the endpoint FIFO or after the USB controller sends a packet to the USB host. The DAC ports have an additional level of masking that allows the user to select which DAC inputs causes a DAC interrupt. The GPIO ports also have a level of masking to select which GPIO inputs causes 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 is programmed for each GPIO port as part of the port configuration. The interrupt polarity can be rising edge (‘0’ to ‘1’) or falling edge (‘1’ to ‘0’).

### USB

The CY7C66013C and CY7C66113C 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 allows power management control of the downstream ports by 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

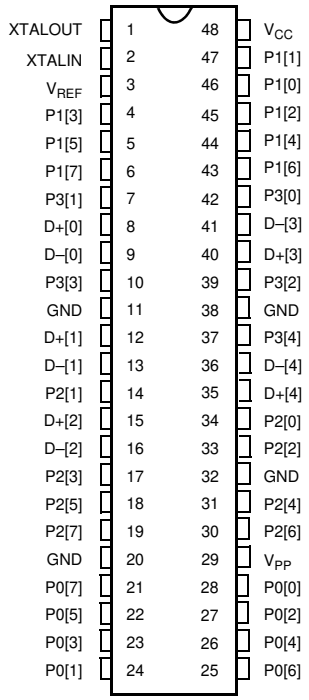


## Pin Configurations

**Figure 1. CY7C66013C 48-Pin SSOP and CY7C66113C 56-Pin SSOP**

**TOP VIEW**

**CY7C66013C**




**CY7C66113C**

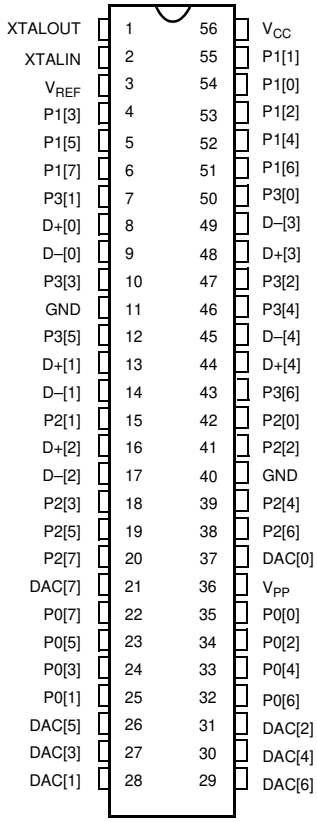


**Figure 2. CY7C66113C 56-Pin QFN**

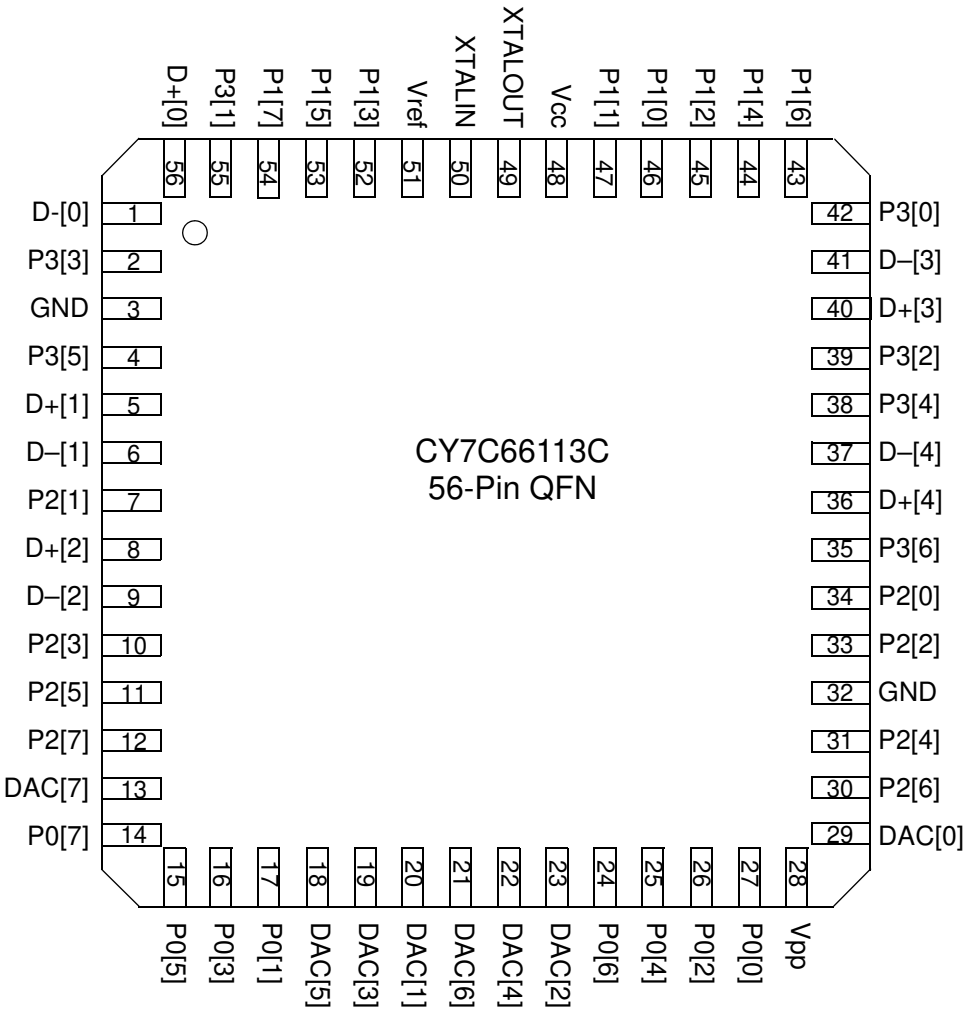
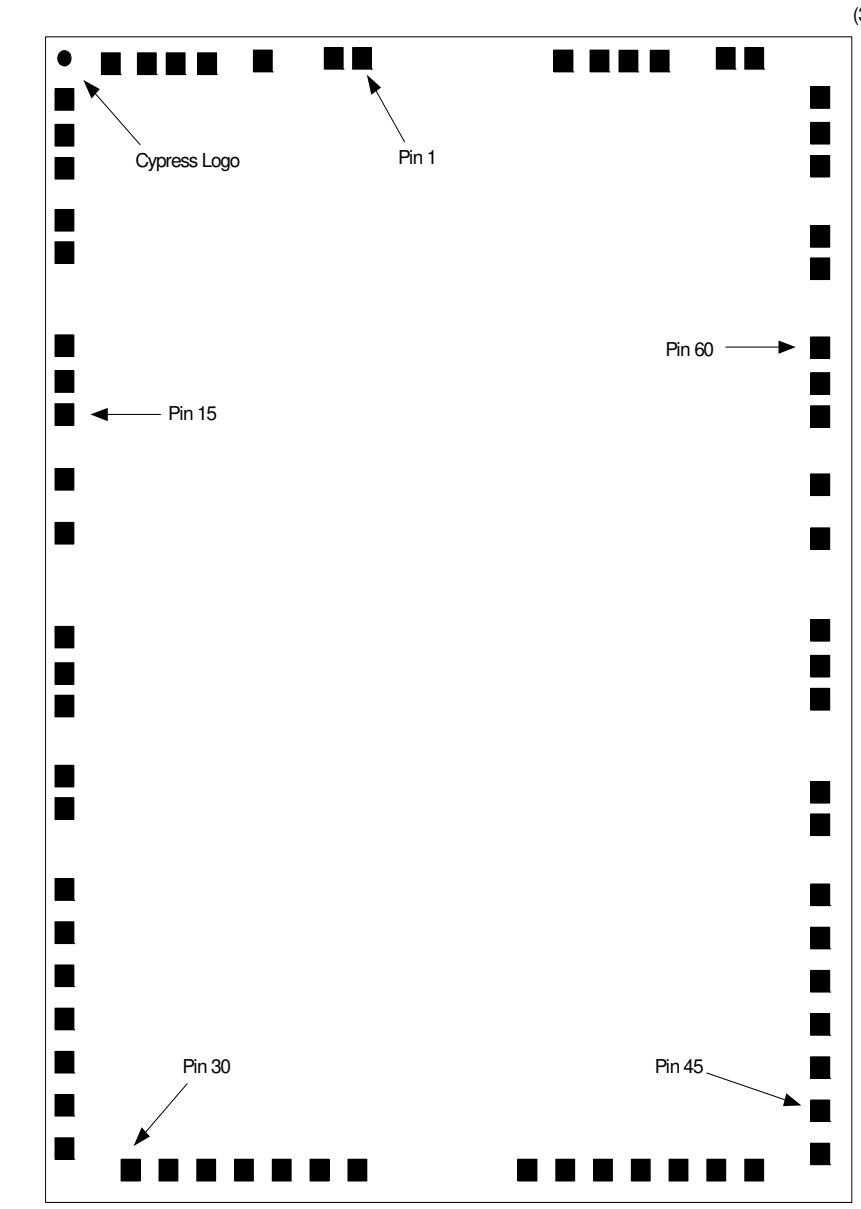


**Figure 3. CY7C66113C DIE**




DIE STEP: 3398 x 4194 microns
Die Size: 3322 x 4129 microns
Die Thickness: 14 mils = 355.6 microns
Pad Size: 80 x 80 microns

**Table 1. Pad Coordinates in Microns (0,0) to Bond Pad Centers**

| Pad No. | Pin Name | X | Y | Pad # | Pin Name | X | Y |
|---|---|---|---|---|---|---|---|
| 1 | XtalOut | 1274.2 | 3588.8 | 37 | DAC6 | 2000.6 | 210.6 |
| 2 | XtalIn | 1132.8 | 3588.8 | 38 | DAC4 | 2103.6 | 210.6 |
| 3 | Vref | 889.85 | 3588.8 | 39 | DAC2 | 2206.6 | 210.6 |
| 4 | Port11b | 684.65 | 3588.8 | 40 | Port06 | 2308.4 | 210.6 |
| 5 | Port13 | 581.65 | 3588.8 | 41 | Port04 | 2411.4 | 210.6 |
| 6 | Port15 | 478.65 | 3588.8 | 42 | Port02 | 2514.4 | 210.6 |
| 7 | Vss | 375.65 | 3588.8 | 43 | Port00 | 2617.4 | 210.6 |
| 8 | Port17 | 0 | 3408.35 | 44 | Vpp | 2992.4 | 25.4 |
| 9 | Port31 | 0 | 3162.05 | 45 | DAC0 | 2992.4 | 151.75 |
| 10 | Du+ | 0 | 3060.55 | 46 | Port26 | 2992.4 | 306.15 |
| 11 | Du– | 0 | 2752.4 | 47 | DD+6 | 2992.4 | 407.65 |
| 12 | Port33 | 0 | 2650.95 | 48 | DD–6 | 2992.4 | 715.75 |
| 13 | Vss | 0 | 2474.6 | 49 | Port24 | 2992.4 | 817.25 |
| 14 | Port35 | 0 | 2368.45 | 50 | Vss | 2992.4 | 923.4 |
| 15 | DD+1 | 0 | 2266.95 | 51 | Port22 | 2992.4 | 1086.75 |
| 16 | DD–1 | 0 | 1958.85 | 52 | DD+5 | 2992.4 | 1188.25 |
| 17 | Port37 | 0 | 1857.35 | 53 | DD–5 | 2992.4 | 1496.35 |
| 18 | Vref | 0 | 1680.4 | 54 | Port20 | 2992.4 | 1597.85 |
| 19 | Port21 | 0 | 1567.4 | 55 | Vref | 2992.4 | 1710.8 |
| 20 | DD+2 | 0 | 1465.95 | 56 | Port36 | 2992.4 | 1874.75 |
| 21 | DD–2 | 0 | 1157.85 | 57 | DD+4 | 2992.4 | 1976.25 |
| 22 | Port23 | 0 | 1056.35 | 58 | DD–4 | 2992.4 | 2284.35 |
| 23 | Vss | 0 | 880 | 59 | Port34 | 2992.4 | 2385.85 |
| 24 | Port25 | 0 | 773.85 | 60 | Vss | 2992.4 | 2492 |
| 25 | DD+7 | 0 | 672.35 | 61 | Port32 | 2992.4 | 2655.35 |
| 26 | DD–7 | 0 | 364.25 | 62 | DD+3 | 2992.4 | 2756.85 |
| 27 | Port27 | 0 | 262.75 | 63 | DD–3 | 2992.4 | 3064.95 |
| 28 | DAC7 | 0 | 100.75 | 64 | Port30 | 2992.4 | 3166.45 |
| 29 | Vss | 0 | 0 | 65 | Port16 | 2992.4 | 3412.25 |
| 30 | Port07 | 375.2 | 210.6 | 66 | Port14 | 2634.2 | 3588.8 |
| 31 | Port05 | 478.2 | 210.6 | 67 | Port12 | 2531.2 | 3588.8 |
| 32 | Port03 | 581.2 | 210.6 | 68 | Port10 | 2428.2 | 3588.8 |
| 33 | Port01 | 684.2 | 210.6 | 69 | Port11 | 2325.2 | 3588.8 |
| 34 | DAC5 | 788.4 | 210.6 | 70 | VCC | 2221.75 | 3588.8 |
| 35 | DAC3 | 891.4 | 210.6 | 71 | PadOpt | 2121.75 | 3588.8 |
| 36 | DAC1 | 994.4 | 210.6 | | | | |

# Product Summary Tables

## Pin Assignments

**Table 2. Pin Assignments**

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

## I/O Register Summary

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

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

**Table 3. I/O Register Summary**

| Register Name | I/O Address | Read/Write | Function | Page |
|---|---|---|---|---|
| Port 0 Data | 0x00 | R/W | GPIO Port 0 Data | 16 |
| Port 1 Data | 0x01 | R/W | GPIO Port 1 Data | 16 |
| Port 2 Data | 0x02 | R/W | GPIO Port 2 Data | 16 |
| Port 3 Data | 0x03 | R/W | GPIO Port 3 Data | 16 |
| Port 0 Interrupt Enable | 0x04 | W | Interrupt Enable for Pins in Port 0 | 18 |
| Port 1 Interrupt Enable | 0x05 | W | Interrupt Enable for Pins in Port 1 | 18 |
| Port 2 Interrupt Enable | 0x06 | W | Interrupt Enable for Pins in Port 2 | 18 |
| Port 3 Interrupt Enable | 0x07 | W | Interrupt Enable for Pins in Port 3 | 18 |
| GPIO Configuration | 0x08 | R/W | GPIO Port Configurations | 17 |

**Table 3. I/O Register Summary** (continued)

| Register Name | I/O Address | Read/Write | Function | Page |
|---|---|---|---|---|
| HAPI and $I^2C$ Configuration | 0x09 | R/W | HAPI Width and $I^2C$ Position Configuration | 22 |
| USB Device Address A | 0x10 | R/W | USB Device Address A | 38 |
| EP A0 Counter Register | 0x11 | R/W | USB Address A, Endpoint 0 Counter | 40 |
| EP A0 Mode Register | 0x12 | R/W | USB Address A, Endpoint 0 Configuration | 39 |
| EP A1 Counter Register | 0x13 | R/W | USB Address A, Endpoint 1 Counter | 40 |
| EP A1 Mode Register | 0x14 | R/W | USB Address A, Endpoint 1 Configuration | 40 |
| EP A2 Counter Register | 0x15 | R/W | USB Address A, Endpoint 2 Counter | 40 |
| EP A2 Mode Register | 0x16 | R/W | USB Address A, Endpoint 2 Configuration | 40 |
| USB Status & Control | 0x1F | R/W | USB Upstream Port Traffic Status and Control | 37 |
| Global Interrupt Enable | 0x20 | R/W | Global Interrupt Enable | 27 |
| Endpoint Interrupt Enable | 0x21 | R/W | USB Endpoint Interrupt Enables | 27 |
| Interrupt Vector | 0x23 | R | Pending Interrupt Vector Read/Clear | 29 |
| Timer (LSB) | 0x24 | R | Lower 8 Bits of Free-running Timer (1 MHz) | 21 |
| Timer (MSB) | 0x25 | R | Upper 4 Bits of Free-running Timer | 21 |
| WDT Clear | 0x26 | W | Watchdog Timer Clear | 14 |
| $I^2C$ Control & Status | 0x28 | R/W | $I^2C$ Status and Control | 23 |
| $I^2C$ Data | 0x29 | R/W | $I^2C$ Data | 23 |
| DAC Data | 0x30 | R/W | DAC Data | 19 |
| DAC Interrupt Enable | 0x31 | W | Interrupt Enable for each DAC Pin | 20 |
| DAC Interrupt Polarity | 0x32 | W | Interrupt Polarity for each DAC Pin | 20 |
| DAC Isink | 0x38-0x3F | W | Input Sink Current Control for each DAC Pin | 20 |
| USB Device Address B | 0x40 | R/W | USB Device Address B (not used in 5-endpoint mode) | 38 |
| EP B0 Counter Register | 0x41 | R/W | USB Address B, Endpoint 0 Counter | 40 |
| EP B0 Mode Register | 0x42 | R/W | USB Address B, Endpoint 0 Configuration, or<br>USB Address A, Endpoint 3 in 5-endpoint Mode | 39 |
| EP B1 Counter Register | 0x43 | R/W | USB Address B, Endpoint 1 Counter | 40 |
| EP B1 Mode Register | 0x44 | R/W | USB Address B, Endpoint 1 Configuration, or<br>USB Address A, Endpoint 4 in 5-endpoint Mode | 40 |
| Hub Port Connect Status | 0x48 | R/W | Hub Downstream Port Connect Status | 32 |
| Hub Port Enable | 0x49 | R/W | Hub Downstream Ports Enable | 33 |
| Hub Port Speed | 0x4A | R/W | Hub Downstream Ports Speed | 33 |
| Hub Port Control (Ports [4:1]) | 0x4B | R/W | Hub Downstream Ports Control | 34 |
| Hub Port Suspend | 0x4D | R/W | Hub Downstream Port Suspend Control | 35 |
| Hub Port Resume Status | 0x4E | R | Hub Downstream Ports Resume Status | 36 |
| Hub Ports SE0 Status | 0x4F | R | Hub Downstream Ports SE0 Status | 35 |
| Hub Ports Data | 0x50 | R | Hub Downstream Ports Differential Data | 35 |
| Hub Downstream Force Low | 0x51 | R/W | Hub Downstream Ports Force LOW | 34 |
| Processor Status & Control | 0xFF | R/W | Microprocessor Status and Control Register | 26 |

## Instruction Set Summary

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

**Table 4. Instruction Set Summary**

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

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

## Programming Model

### 14-Bit Program Counter (PC)

The 14-bit PC allows access to up to 8 KB of PROM available with the CY7C66x13C 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. Typically, this is a jump instruction to a reset handler that initializes the application (see Interrupt Vectors on page 28).

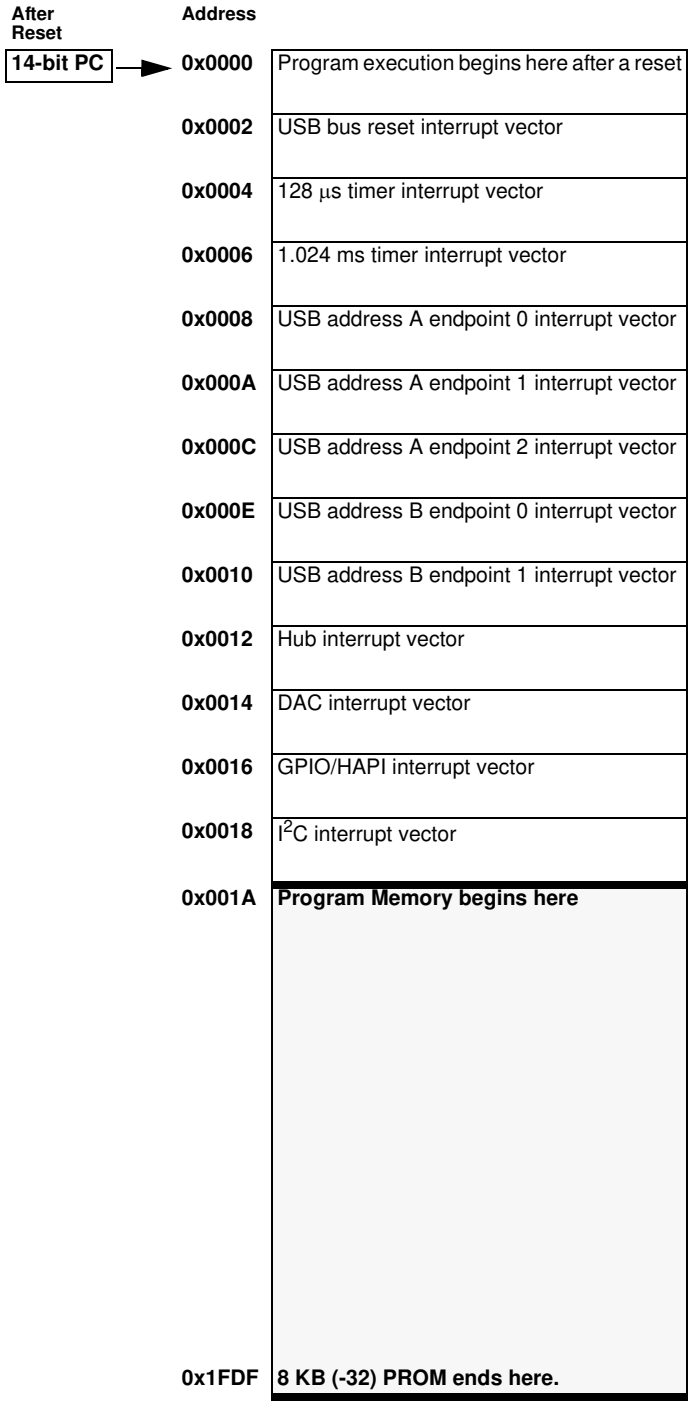
The lower eight bits of the program counter are incremented as instructions are loaded and executed. The upper six bits of the program counter are incremented by executing an XPAGE instruction. The last instruction executed within a 256-byte “page” of sequential code should be an XPAGE instruction. The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

The address of the next instruction to be executed, the carry flag, and the 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.

The program counter is not accessed directly by the firmware. The program stack is examined by reading SRAM from location 0x00 and up.

*Program Memory Organization*

**Figure 4. Program Memory Space with Interrupt Vector Table**



### 8-Bit Accumulator (A)

The accumulator is the general purpose register for the microcontroller.

### 8-Bit Temporary Register (X)

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

### 8-Bit Program Stack Pointer (PSP)

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

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

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

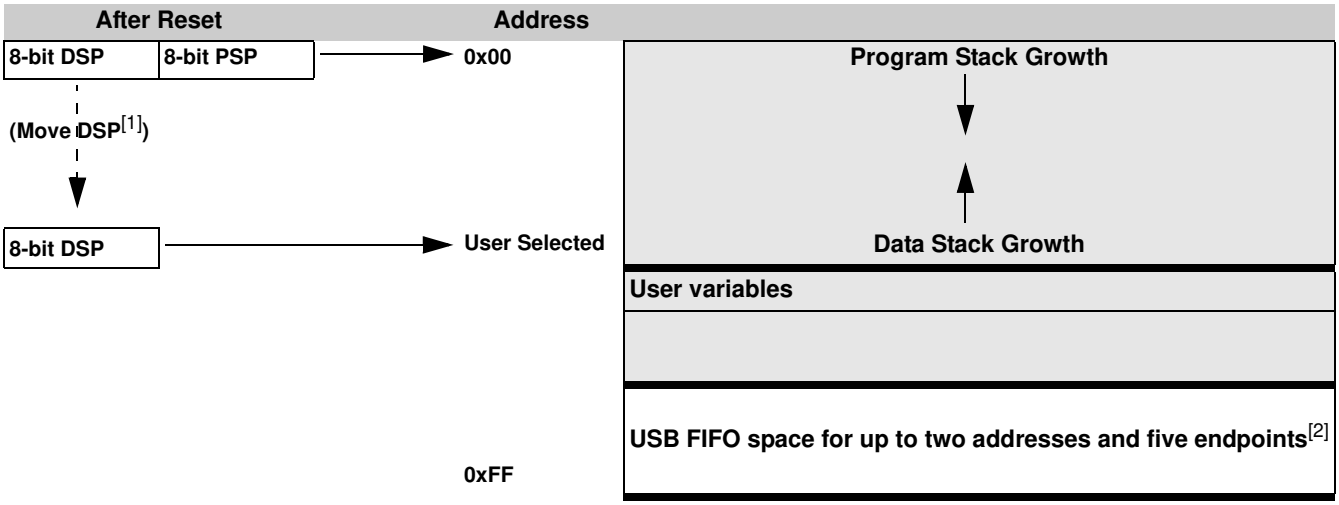
The Call Subroutine (CALL) instruction stores the program counter and flags on the program stack and increments the PSP by two.

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

*Data Memory Organization*

The CY7C66x13C microcontrollers provide 256 bytes of data RAM. Normally, the SRAM is partitioned into four areas: program stack, user variables, data stack, and USB endpoint FIFOs. The following is one example of where the program stack, data stack, and user variables areas are located.

**Table 5. SRAM Areas**




### 8-Bit Data Stack Pointer (DSP)

The Data Stack Pointer (DSP) supports PUSH and POP instructions that use the data stack for temporary storage. A PUSH instruction pre-decrements the DSP, then writes data to the memory location addressed by the DSP. A POP instruction reads data from the memory location addressed by the DSP, then post-increments the DSP.

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

For USB applications, the firmware should set the DSP to an appropriate location to avoid a memory conflict with RAM dedicated to USB FIFOs. The memory requirements for the USB endpoints are described in USB Device Endpoints on page 38. Example assembly instructions to do this with two device addresses (FIFOs begin at 0xD8) are shown:

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

**Notes**
1. Refer to 8-Bit Data Stack Pointer (DSP) for a description of DSP.
2. Endpoint sizes are fixed by the Endpoint Size Bit (I/O register 0x1F, Bit 7), see Table 14.

### Address Modes

The CY7C66013C and CY7C66113C microcontrollers support three addressing modes for instructions that require data operands: data, direct, and indexed.

*Data (Immediate)*

"Data" address mode refers to a data operand that is actually a constant encoded in the instruction. As an example, consider the instruction that loads A with the constant 0xD8:

```
MOV A, 0D8h.
```

This instruction requires two bytes of code where the first byte identifies the "MOV A" instruction with a data operand as the second byte. The second byte of the instruction is the constant "0xD8". A constant may be referred to by name if a prior "EQU" statement assigns the constant value to the name. For example, the following code is equivalent to the example described earlier:

```
DSPINIT: EQU 0D8h
MOV A, DSPINIT.
```

*Direct*

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

```
MOV A, [10h].
```

Normally, variable names are assigned to variable addresses using "EQU" statements to improve the readability of the assembler source code. As an example, the following code is equivalent to the example described earlier:

```
buttons: EQU 10h
MOV A, [buttons].
```

*Indexed*

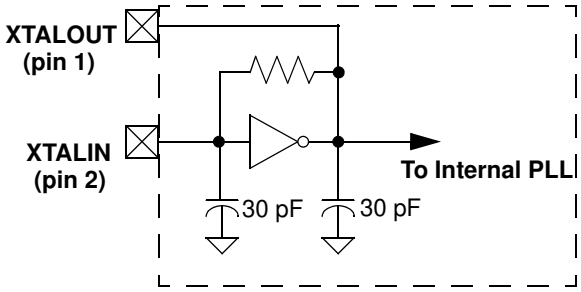
"Indexed" address mode allows the firmware to manipulate arrays of data stored in SRAM. The address of the data operand is the sum of a constant encoded in the instruction and the contents of the "X" register. Normally, the constant is the "base" address of an array of data and the X register contains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

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

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

## Clocking

**Figure 5. Clock Oscillator On-Chip Circuit**




The XTALIN and XTALOUT are the clock pins to the microcontroller. The user connects an external oscillator or a crystal to these pins. When using an external crystal, keep PCB traces between the chip leads and crystal as short as possible (less than 2 cm). A 6 MHz fundamental frequency parallel resonant crystal is connected to these pins to provide a reference frequency for the internal PLL. The two internal 30 pF load caps appear in series to the external crystal and would be equivalent to a 15 pF load. Therefore, the crystal must have a required load capacitance of about 15–18 pF. A ceramic resonator does not allow the microcontroller to meet the timing specifications of full speed USB and so a ceramic resonator is not recommended with these parts.

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

## Reset

The CY7C66x13C supports two resets: POR and a Watchdog Reset (WDR). Each of these resets causes:

- All registers to be restored to their default states.
- The USB device addresses to be set to 0.
- All interrupts to be disabled.
- The PSP and DSP to be set to memory address 0x00.

The occurrence of a reset is recorded in the Processor Status and Control Register, as described in Processor Status and Control Register on page 26. Bits 4 and 6 are used to record the occurrence of POR and WDR, respectively. Firmware interrogates these bits to determine the cause of a reset.

Program execution starts at ROM address 0x0000 after a reset. Although this looks similar to interrupt vector 0, there is an important difference. Reset processing does NOT push the program counter, carry flag, and zero flag onto program stack. The firmware reset handler should configure the hardware before the “main” loop of code. Attempting to execute a RET or RETI in the firmware reset handler causes unpredictable execution results.

### Power on Reset

When $V_{CC}$ is first applied to the chip, the POR signal is asserted and the CY7C66x13C enters a “semi-suspend” state. During the semi-suspend state, which is different from the suspend state defined in the USB specification, the oscillator and all other blocks of the part are functional, except for the CPU. This semi-suspend time ensures that both a valid $V_{CC}$ level is reached and that the internal PLL has time to stabilize before full operation begins. When the $V_{CC}$ rises above approximately 2.5V, and the oscillator is stable, the POR is deasserted and the on-chip timer starts counting. The first 1 ms of suspend time is not interruptible, and the semi-suspend state continues for an additional 95 ms unless the count is bypassed by a USB Bus Reset on the upstream port. The 95 ms provides time for $V_{CC}$ to stabilize at a valid operating voltage before the chip executes code.

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

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

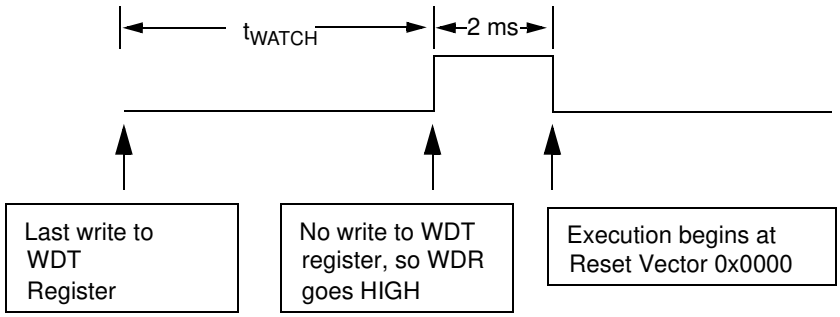
### Watchdog Reset

The WDR occurs when the internal WDT rolls over. Writing any value to the write only Watchdog Restart Register at address 0x26 clears the timer. The timer rolls over and WDR occurs if it is not cleared within $t_{WATCH}$ (8 ms minimum) of the last clear. Bit 6 of the Processor Status and Control Register is set to record this event (the register contents are set to 010X0001 by the WDR). A WDT Reset lasts for 2 ms, after which the microcontroller begins execution at ROM address 0x0000.

The USB transmitter is disabled by a WDR because the USB Device Address Registers are cleared (see USB Device Addresses). Otherwise, the USB Controller responds to all address 0 transactions.

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

**Figure 6. Watchdog Reset**



## Suspend Mode

The CY7C66x13C is placed into a low power state by setting the Suspend bit of the Processor Status and Control register. All logic blocks in the device are turned off except the GPIO interrupt logic and the USB receiver. The clock oscillator, PLL, and the free-running and WDTs are shut down. Only the occurrence of an enabled GPIO interrupt or non idle bus activity at a USB upstream or downstream port wakes the part from suspend. The Run bit in the Processor Status and Control Register must be set to resume a part out of suspend.

The clock oscillator restarts immediately after exiting suspend mode. The microcontroller returns to a fully functional state 1 ms after the oscillator is stable. The microcontroller executes the instruction following the I/O write that placed the device into suspend mode before servicing any interrupt requests.

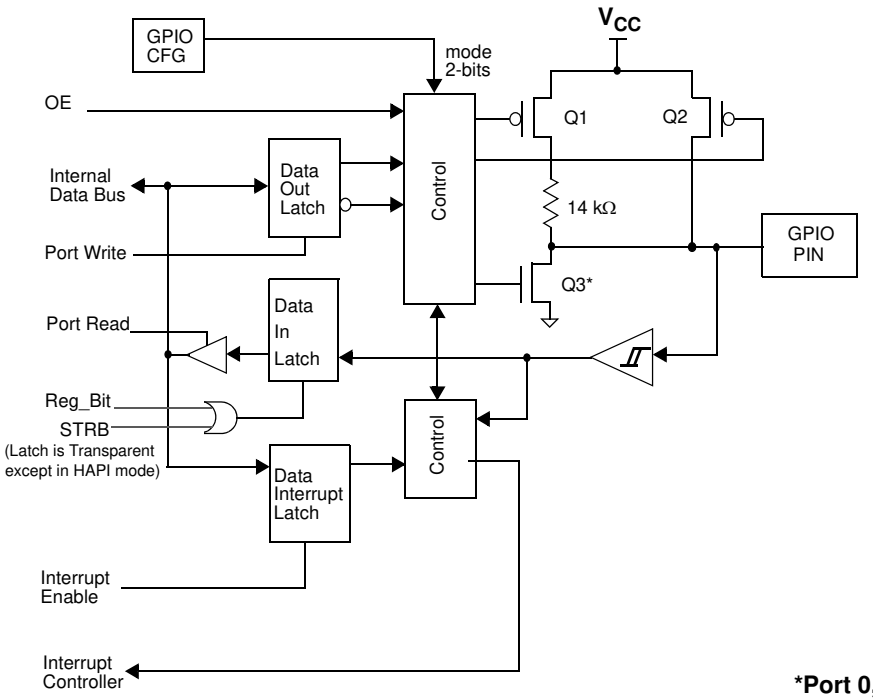
The GPIO interrupt allows the controller to wake up periodically and poll system components while maintaining a very low average power consumption. To achieve the lowest possible current during suspend mode, all I/O should be held at $V_{CC}$ or Gnd. This also applies to internal port pins that may not be bonded in a particular package.

Typical code for entering suspend is given here:

```
     ...  ; All GPIO set to low power state (no
floating pins)
     ...  ; Enable GPIO interrupts if desired
for wakeup
     mov a, 09h; Set suspend and run bits
     iowr FFh; Write to Status and Control
Register - Enter suspend, wait for USB activity
(or GPIO Interrupt)
     nop  ; This executes before any ISR
```

## General Purpose I/O (GPIO) Ports

**Figure 7. Block Diagram of a GPIO Pin**




***Port 0,1,2: Low $I_{sink}$**
**Port 3: High $I_{sink}$**

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 is configured as inputs with internal pull ups, open drain outputs, or traditional CMOS outputs. Port 3 offers a higher current drive, with typical current sink capability of 12 mA. The data for each GPIO port is accessible through the data registers. Port data registers are shown in Figure 8 through Figure 11, and are set to 1 on reset.

**Figure 8. Port 0 Data**

**Port 0 Data** **ADDRESS 0x00**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | P0.7 | P0.6 | P0.5 | P0.4 | P0.3 | P0.2 | P0.1 | P0.0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 9. Port1 Data**

**Port 1Data** **ADDRESS 0x01**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | P1.7 | P1.6 | P1.5 | P1.4 | P1.3 | P1.2 | P1.1 | P1.0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 10. Port 2 Data**

**Port 2 Data** **ADDRESS 0x02**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | P2.7 | P2.6 | P2.5 | P2.4 | P2.3 | P2.2 | P2.1 | P2.0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Figure 11. Port 3 Data**

**Port 3 Data** **ADDRESS 0x03**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | Reserved | P3.6 CY7C66113C only | P3.5 CY7C66113C only | P3.4 | P3.3 | P3.2 | P3.1 | P3.0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Special care should be taken with any unused GPIO data bits. An unused GPIO data bit, either a pin on the chip or a port bit that is not bonded on a particular package, must not be left floating when the device enters the suspend state. If a GPIO data bit is left floating, the leakage current caused by the floating bit may violate the suspend current limitation specified by the USB specifications. If a '1' is written to the unused data bit and the port is configured with open drain outputs, the unused data bit remains in an indeterminate state. Therefore, if an unused port bit is programmed in open-drain mode, it must be written with a '0.' Notice that the CY7C66013C always requires that P3[7:5] be written with a '0.' When the CY7C66113C is used the P3[7] should be written with a '0.'

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

## GPIO Configuration Port

Every GPIO port is programmed as inputs with internal pull ups, outputs LOW or HIGH, or Hi-Z (floating, the pin is not driven internally). In addition, the interrupt polarity for each port is programmed. The Port Configuration bits (Figure 12) and the Interrupt Enable bit (Figure 10 through Figure 16) determine the interrupt polarity of the port pins.

**Figure 12. GPIO Configuration Register**

**GPIO Configuration** **ADDRESS 0x08**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | Port 3 Config Bit 1 | Port 3 Config Bit 0 | Port 2 Config Bit 1 | Port 2 Config Bit 0 | Port 1 Config Bit 1 | Port 1 Config Bit 0 | Port 0 Config Bit 1 | Port 0 Config Bit 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

As shown in Table 6, a positive polarity on an input pin represents a rising edge interrupt (LOW to HIGH), and a negative polarity on an input pin represents a falling edge interrupt (HIGH to LOW).

The GPIO interrupt is generated when all of the following conditions are met: the Interrupt Enable bit of the associated Port Interrupt Enable Register is enabled, the GPIO Interrupt Enable bit of the Global Interrupt Enable Register (Figure 29) is enabled, the Interrupt Enable Sense (bit 2, Figure 28) is set, and the GPIO pin of the port sees an event matching the interrupt polarity.

The driving state of each GPIO pin is determined by the value written to the pin's Data Register (Figure 8 through Figure 11) and by its associated Port Configuration bits as shown in the GPIO Configuration Register (Figure 10). These ports are configured on a per port basis, so all pins in a given port are configured together. The possible port configurations are detailed in Table 6. As shown in this table, when a GPIO port is configured with CMOS outputs, interrupts from that port are disabled.

During reset, all the bits in the GPIO Configuration Register are written with '0' to select Hi-Z mode for all GPIO ports as the default configuration.

**Table 6. GPIO Port Output Control Truth Table and Interrupt Polarity**

| Port Config Bit 1 | Port Config Bit 0 | Data Register | Output Drive Strength | Interrupt Enable Bit | Interrupt Polarity |
|---|---|---|---|---|---|
| 1 | 1 | 0 | Output LOW | 0 | Disabled |
| | | 1 | Resistive | 1 | – (Falling Edge) |
| 1 | 0 | 0 | Output LOW | 0 | Disabled |
| | | 1 | Output HIGH | 1 | Disabled |
| 0 | 1 | 0 | Output LOW | 0 | Disabled |
| | | 1 | Hi-Z | 1 | – (Falling Edge) |
| 0 | 0 | 0 | Output LOW | 0 | Disabled |
| | | 1 | Hi-Z | 1 | + (Rising Edge) |

Q1, Q2, and Q3 discussed here are the transistors referenced in Figure 7. The available GPIO drive strength are:

- Output LOW Mode: The pin's Data Register is set to '0'

  Writing '0' to the pin's Data Register puts the pin in output LOW mode, regardless of the contents of the Port Configuration Bits[1:0]. In this mode, Q1 and Q2 are OFF. Q3 is ON. The GPIO pin is driven LOW through Q3.

- Output HIGH Mode: The pin's Data Register is set to 1 and the Port Configuration Bits[1:0] is set to '10'

  In this mode, Q1 and Q3 are OFF. Q2 is ON. The GPIO is pulled up through Q2. The GPIO pin is capable of sourcing current.

- Resistive Mode: The pin's Data Register is set to 1 and the Port Configuration Bits[1:0] is set to '11'

  Q2 and Q3 are OFF. Q1 is ON. The GPIO pin is pulled up with an internal 14 kΩ resistor. In resistive mode, the pin may serve as an input. Reading the pin's Data Register returns a logic HIGH if the pin is not driven LOW by an external source.

- Hi-Z Mode: The pin's Data Register is set to1 and Port Configuration Bits[1:0] is set either '00' or '01'

  Q1, Q2, and Q3 are all OFF. The GPIO pin is not driven internally. In this mode, the pin may serve as an input. Reading the Port Data Register returns the actual logic value on the port pins.

## GPIO Interrupt Enable Ports

Each GPIO pin is individually enabled or disabled as an interrupt source. The Port 0–3 Interrupt Enable registers provide this feature with an interrupt enable bit for each GPIO pin. When HAPI mode is enabled the GPIO interrupts are blocked, including ports not used by HAPI, so GPIO pins are not used as interrupt sources.

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. All GPIO pins share a common interrupt, as discussed in GPIO and HAPI Interrupt.

**Figure 13. Port 0 Interrupt Enable**

**Port 0 Interrupt Enable** **ADDRESS 0x04**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | P0.7 Intr Enable | P0.6 Intr Enable | P0.5 Intr Enable | P0.4 Intr Enable | P0.3 Intr Enable | P0.2 Intr Enable | P0.1 Intr Enable | P0.0 Intr Enable |
| Read/Write | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 14. Port 1 Interrupt Enable**

**Port 1 Interrupt Enable** **ADDRESS 0x05**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | P1.7 Intr Enable | P1.6 Intr Enable | P1.5 Intr Enable | P1.4 Intr Enable | P1.3 Intr Enable | P1.2 Intr Enable | P1.1 Intr Enable | P1.0 Intr Enable |
| Read/Write | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 15. Port 2 Interrupt Enable**

**Port 2 Interrupt Enable** **ADDRESS 0x06**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | P2.7 Intr Enable | P2.6 Intr Enable | P2.5 Intr Enable | P2.4 Intr Enable | P2.3 Intr Enable | P2.2 Intr Enable | P2.1 Intr Enable | P2.0 Intr Enable |
| Read/Write | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 16. Port 3 Interrupt Enable**

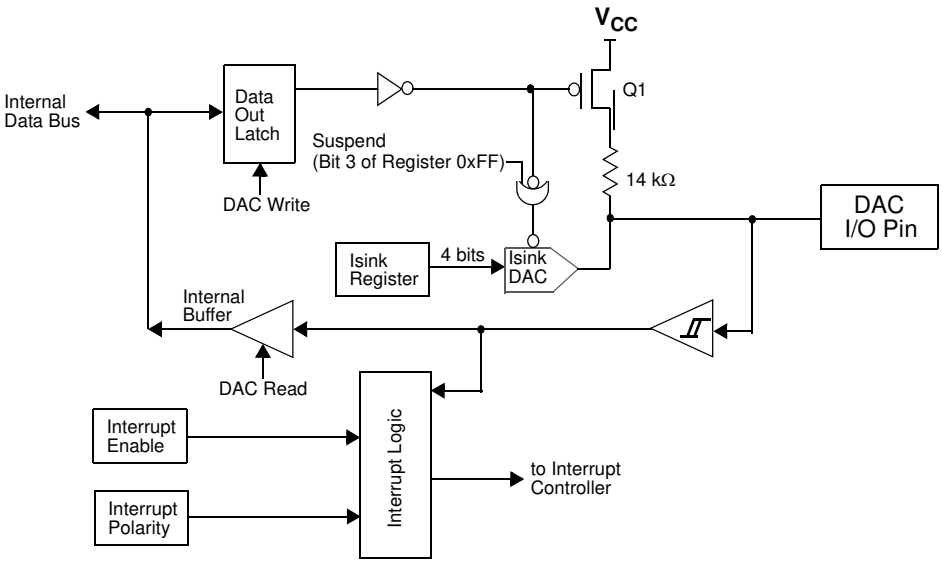
**Port 3 Interrupt Enable** **ADDRESS 0x07**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | Reserved | P3.6 Intr Enable CY7C66113C only | P3.5 Intr Enable CY7C66113C only | P3.4 Intr Enable | P3.3 Intr Enable | P3.2 Intr Enable | P3.1 Intr Enable | P3.0 Intr Enable |
| Read/Write | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

## DAC Port

The CY7C66113CC features a programmable sink current 8-bit port, which is also known as DAC port. Each of these port I/O pins have a programmable current sink. Writing a '1' to a DAC I/O pin disables the output current sink ($I_{sink}$ 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 $I_{sink}$ DAC is enabled and the pull up resistor is disabled. This causes the $I_{sink}$ DAC to sink current to drive the output LOW. Figure 17 shows a block diagram of the DAC port pin.

**Figure 17. Block Diagram of a DAC Pin**




The amount of sink current for the DAC I/O pin is programmable over 16 values based on the contents of the DAC Isink Register (Figure 19) for that output pin. DAC[1:0] are high current outputs that are programmable from 3.2 mA to 16 mA (typical). DAC[7:2] are low current outputs, programmable from 0.2 mA to 1.0 mA (typical).

When the suspend bit in Processor Status and Control Register (Figure 28) is set, the Isink DAC block of the DAC circuitry is disabled. Special care should be taken when the CY7C66113C device is placed in the suspend. The DAC Port Data Register (Figure 18) should normally be loaded with all '1's (Figure 28) before setting the suspend bit. If any of the DAC bits are set to '0' when the device is suspended, that DAC input floats. The floating pin could result in excessive current consumption by the device, unless an external load places the pin in a deterministic state.

**Figure 18. DAC Port Data**

**DAC Port Data** **ADDRESS 0x30**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | DAC[7] | DAC[6] | DAC[5] | DAC[4] | DAC[3] | DAC[2] | DAC[1] | DAC[0] |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

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

1= I/O pin is an output pulled HGH through the 14 kΩ resistor.
0 = I/O pin is an input with an internal 14 kΩ pull up resistor.

**Bit [7..2]: Low Current Output 0.2 mA to 1 mA typical**

1= I/O pin is an output pulled HGH through the 14 kΩ resistor.
0 = I/O pin is an input with an internal 14 kΩ pull up resistor.

## DAC Isink Registers

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

**Figure 19. DAC Sink Register**

**DAC Sink Register** **ADDRESS 0x38 –0x3F**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | Reserved | Reserved | Reserved | Reserved | Isink[3] | Isink[2] | Isink[1] | Isink[0] |
| Read/Write | | | | | W | W | W | W |
| Reset | - | - | - | - | 0 | 0 | 0 | 0 |

**Bit [3..0]: Isink [x] (x= 0..3)**

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

**Bit [7..4]: Reserved**

## DAC Port Interrupts

A DAC port interrupt is enabled or disabled for each pin individually. The DAC Port Interrupt Enable register provides this feature with an interrupt enable bit for each DAC I/O pin. All of the DAC Port Interrupt Enable register bits are cleared to ‘0’ during a reset. All DAC pins share a common interrupt, as explained in DAC Interrupt on page 30.

**Figure 20. DAC Port Interrupt Enable**

**DAC Port Interrupt** **ADDRESS 0x31**

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

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

1 = Enables interrupts from the corresponding bit position; 0= Disables interrupts from the corresponding bit position

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

**Figure 21. DAC Port Interrupt Polarity**

**DAC I/O Interrupt Polarity** **ADDRESS 0x32**

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

**Bit [7..0]: Polarity bit x (x= 0..7)**

1= Selects positive polarity (rising edge) that causes an interrupt (if enabled); 0 = Selects negative polarity (falling edge) that causes an interrupt (if enabled).

## 12-Bit Free-Running Timer

The 12-bit timer operates with a 1 µs tick, 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 eight bits of the timer is read directly by the firmware. Reading the lower 8 bits latches the upper four bits into a temporary register. When the firmware reads the upper four bits of the timer, it is actually reading the count stored in the temporary register. The effect of this is to ensure a stable 12-bit timer value is read, even when the two reads are separated in time.

**Figure 22. Timer LSB Register**

**Timer LSB** **ADDRESS 0x24**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | Timer Bit 7 | Timer Bit 6 | Timer Bit 5 | Timer Bit 4 | Timer Bit 3 | Timer Bit 2 | Timer Bit 1 | Timer Bit 0 |
| Read/Write | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit [7:0]: Timer lower eight bits**

**Figure 23. Timer MSB Register**

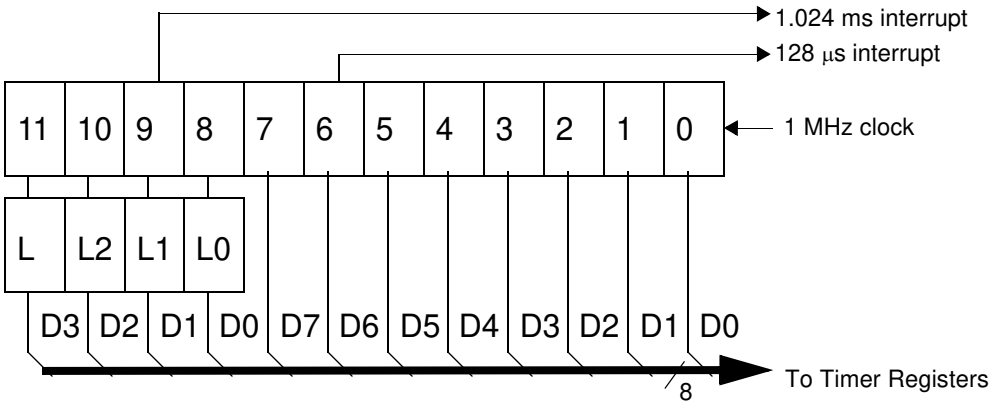
**Timer MSB** **ADDRESS 0x25**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | Reserved | Reserved | Reserved | Reserved | Timer Bit 11 | Timer Bit 10 | Timer Bit 9 | Timer Bit 8 |
| Read/Write | - | - | - | - | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit [3:0]: Timer higher nibble**

**Bit [7:4]: Reserved**

**Figure 24. Timer Block Diagram**



## I²C and HAPI Configuration Register

Internal hardware supports communication with external devices through two interfaces: a two wire I²C compatible, and a HAPI for 1, 2, or 3 byte transfers. The I²C compatible and HAPI functions, share a common configuration register (see Figure 25)[3]. All bits of this register are cleared on reset.

**Figure 25. HAPI/I²C Configuration Register**

**I²C Configuration** **ADDRESS 0x09**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | I²C Position | Reserved | LEMPTY Polarity | DRDY Polarity | Latch Empty | Data Ready | HAPI Port Width Bit 1 | HAPI Port Width Bit 0 |
| Read/Write | R/W | - | R/W | R/W | R | R | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bits [7,1:0] of the HAPI and I²C Configuration Register control the pin out configuration of the HAPI and I²C compatible interfaces. Bits [5:2] are used in HAPI mode only, and are described in Hardware Assisted Parallel Interface (HAPI). Table 7 shows the HAPI port configurations, and Table 8 shows I²C pin location configuration options. These I²C compatible options exist due to pin limitations in certain packages, and to allow simultaneous HAPI and I²C compatible operation.

HAPI operation is enabled whenever either HAPI Port Width Bit (Bit 1 or 0) is non zero. This affects GPIO operation as described in Hardware Assisted Parallel Interface (HAPI). The I²C compatible interface must be separately enabled.

**Table 7. HAPI Port Configuration**

| Port Width (Bit 0 and 1, Figure 25) | 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 8. I²C Port Configuration**

| I²C Position (Bit 7, Figure 25) | I²C Port Width (Bit 1, Figure 25) | I²C Position |
|---|---|---|
| Don't Care | 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 |

## I²C Compatible Controller

The I²C compatible block provides a versatile two wire communication with external devices, supporting master, slave, and multi-master modes of operation. The I²C compatible block functions by handling the low level signaling in hardware, and issuing interrupts as needed to allow firmware to take appropriate action during transactions. While waiting for firmware response, the hardware keeps the I²C compatible bus idle if necessary.

The I²C compatible interface generates an interrupt to the microcontroller at the end of each received or transmitted byte, when a stop bit is detected by the slave when in receive mode, or when arbitration is lost. Details of the interrupt responses are given in Hardware Assisted Parallel Interface (HAPI).

The I²C compatible interface consists of two registers, an I²C Data Register (Figure 14) and an I²C Status and Control Register (Figure 27). The Data Register is implemented as separate read and write registers. Generally, the I²C Status and Control Register are only monitored after the I²C interrupt, as all bits are valid at that time. Polling this register at other times could read misleading bit status if a transaction is underway.

The I²C SCL clock is connected to bit 0 of GPIO port 1 or GPIO port 2, and the I²C SDA data is connected to bit 1 of GPIO port 1 or GPIO port 2. Refer to I²C and HAPI Configuration Register for the bit definitions and functionality of the HAPI and ²C Configuration Register, which is used to set the locations of the configurable I²C pins. When the I²C compatible functionality is enabled by setting bit 0 of the I²C Status & Control Register, the two LSB ([1:0]) of the corresponding GPIO port is placed in Open Drain mode, regardless of the settings of the GPIO Configuration Register. The electrical characteristics of the I²C compatible interface is the same as that of GPIO ports 1 and 2. Note that the $I_{OL}$ (max) is 2 mA at $V_{OL}$ = 2.0V for ports 1 and 2.

All control of the I²C clock and data lines is performed by the I²C compatible block.

**Note**
3. I²C compatible function must be separately enabled.

**Figure 26. I²C Data Register**

**I²C Data** **ADDRESS 0x29**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | I²C Data 7 | I²C Data 6 | I²C Data 5 | I²C Data 4 | I²C Data 3 | I²C Data 2 | I²C Data 1 | I²C Data 0 |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | X | X | X | X | X | X | X | X |

**Bits [7..0]: I²C Data**

Contains 8-bit data on the I²C Bus.

**Figure 27. I²C Status and Control Register**

**I²C Status and Control** **ADDRESS 0x28**

| Bit # | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Bit Name | MSTR Mode | Continue/Busy | Xmit Mode | ACK | Addr | ARB Lost/Restart | Received Stop | I²C Enable |
| Read/Write | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

The I²C Status and Control register bits are defined in Table 9, with a more detailed description following.

**Table 9. I²C Status and Control Register Bit Definitions**

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

**Bit 7: MSTR Mode**

Setting this bit to 1 causes the $I^2C$ compatible block to initiate a master mode transaction by sending a start bit and transmitting the first data byte from the data register (this typically holds the target address and R/W bit). Subsequent bytes are initiated by setting the Continue bit, as described later in this section.

Clearing this bit (set to 0) causes the GPIO pins to operate normally. In master mode, the $I^2C$ compatible block generates the clock (SCK), and drives the data line as required depending on transmit or receive state. The $I^2C$ compatible block performs any required arbitration and clock synchronization. IN the event of a loss of arbitration, this MSTR bit is cleared, the ARB Lost bit is set, and an interrupt is generated by the microcontroller. If the chip is the target of an external master that wins arbitration, then the interrupt is held off until the transaction from the external master is completed.

When MSTR Mode is cleared from 1 to 0 by a firmware write, an $I^2C$ Stop bit is generated.

**Bit 6: Continue/Busy**

This bit is written by the firmware to indicate that the firmware is ready for the next byte transaction to begin. In other words, the bit has responded to an interrupt request and has completed the required update or read of the data register. During a read this bit indicates if the hardware is busy and is locking out additional writes to the $I^2C$ Status and Control register. This locking allows the hardware to complete certain operations that may require an extended period of time. Following an $I^2C$ interrupt, the $I^2C$ compatible block does not return to the Busy state until firmware sets the Continue bit. This allows the firmware to make one control register write without the need to check the Busy bit.

**Bit 5: Xmit Mode**

This bit is set by firmware to enter transmit mode and perform a data transmit in master or slave mode. Clearing this bit sets the part in receive mode. Firmware generally determines the value of this bit from the R/W bit associated with the $I^2C$ address packet. The Xmit Mode bit state is ignored when initially writing the MSTR Mode or the Restart bits, as these cases always cause transmit mode for the first byte.

**Bit 4: ACK**

This bit is set or cleared by firmware during receive operation to indicate if the hardware should generate an ACK signal on the $I^2C$ compatible bus. Writing a 1 to this bit generates an ACK (SDA LOW) on the $I^2C$ compatible bus at the ACK bit time. During transmits (Xmit Mode = 1), this bit should be cleared.

**Bit 3: Addr**

This bit is set by the $I^2C$ compatible block during the first byte of a slave receive transaction, after an $I^2C$ start or restart. The Addr bit is cleared when the firmware sets the Continue bit. This bit allows the firmware to recognize when the master has lost arbitration, and in slave mode it allows the firmware to recognize that a start or restart has occurred.

**Bit 2: ARB Lost/Restart**

This bit is valid as a status bit (ARB Lost) after master mode transactions. In master mode, set this bit (along with the Continue and MSTR Mode bits) to perform an $I^2C$ restart sequence. The $I^2C$ target address for the restart must be written to the data register before setting the Continue bit. To prevent false ARB Lost signals, the Restart bit is cleared by hardware during the restart sequence.

**Bit 1: Receive Stop**

This bit is set when the slave is in receive mode and detects a stop bit on the bus. The Receive Stop bit is not set if the firmware terminates the $I^2C$ transaction by not acknowledging the previous byte transmitted on the $I^2C$ compatible bus. For example, in receive mode if firmware sets the Continue bit and clears the ACK bit.

**Bit 0: $I^2C$ Enable**

Set this bit to override GPIO definition with $I^2C$ compatible function on the two $I^2C$ compatible pins. When this bit is cleared, these pins are free to function as GPIOs. In $I^2C$ compatible mode, the two pins operate in open drain mode, independent of the GPIO configuration setting.

## Hardware Assisted Parallel Interface (HAPI)

The CY7C66x13C processor provides a hardware assisted parallel interface for bus widths of 8, 16, or 24 bits, to accommodate data transfer with an external microcontroller or similar device. Control bits for selecting the byte width are in the HAPI and $I^2C$ Configuration Register (Figure 25), bits 1 and 0.

Signals are provided on Port 2 to control the HAPI interface. Table 10 describes these signals and the HAPI control bits in the HAPI and $I^2C$ Configuration Register. Enabling HAPI causes the GPIO setting in the GPIO Configuration Register (Figure 10) to be overridden. The Port 2 output pins are in CMOS output mode and Port 2 input pins are in input mode (open drain mode with Q3 OFF in Figure 7).

**Table 10. Port 2 Pin and HAPI Configuration Bit Definitions**

| Pin | Name | Direction | Description (Port 2 Pin) |
|---|---|---|---|
| P2[2] | LatEmptyPin | Out | Ready for more input data from external interface. |
| P2[3] | DReadyPin | Out | Output data ready for external interface. |
| P2[4] | STB | In | Strobe signal for latching incoming data. |
| P2[5] | OE | In | Output Enable, causes chip to output data. |
| P2[6] | CS | In | Chip Select (Gates $\overline{STB}$ and $\overline{OE}$). |
| **Bit** | **Name** | **R/W** | **Description (HAPI and $I^2C$ Configuration Register)** |
| 2 | Data Ready | R | Asserted after firmware writes data to Port 0, until $\overline{OE}$ driven LOW. |
| 3 | Latch Empty | R | Asserted after firmware reads data from Port 0, until $\overline{STB}$ driven LOW. |
| 4 | DRDY Polarity | R/W | Determines polarity of Data Ready bit and DReadyPin:<br>If 0, Data Ready is active LOW, DReadyPin is active HIGH.<br>If 1, Data Ready is active HIGH, DReadyPin is active LOW. |
| 5 | LEMPTY Polarity | R/W | Determines polarity of Latch Empty bit and LatEmptyPin:<br>If 0, Latch Empty is active LOW, LatEmptyPin is active HIGH.<br>If 1, Latch Empty is active HIGH, LatEmptyPin is active LOW. |

### HAPI Read by External Device from CY7C66x13C

In this case (see Figure 50), firmware writes data to the GPIO ports. If 16-bit or 24-bit transfers are being made, Port 0 is written last, because writes to Port 0 asserts the Data Ready bit and the DReadyPin to signal the external device that data is available.

The external device then drives the $\overline{OE}$ and $\overline{CS}$ pins active (LOW), which causes the HAPI data to be output on the port pins. When $\overline{OE}$ is returned HIGH (inactive), the HAPI/GPIO interrupt is generated. At that point, firmware is reload the HAPI latches for the next output, again writing Port 0 last.

The Data Ready bit reads the opposite state from the external DReadyPin on pin P2[3]. If the DRDY Polarity bit is 0, DReadyPin is active HIGH, and the Data Ready bit is active LOW.

### HAPI Write by External Device to CY7C66x13C

In this case (see Figure 52), the external device drives the $\overline{STB}$ and $\overline{CS}$ pins active (LOW) when it drives new data onto the port pins. When this happens, the internal latches become full, which causes the Latch Empty bit to be deasserted. When $\overline{STB}$ is returned HIGH (inactive), the HAPI and GPIO interrupt is generated. Firmware then reads the parallel ports to empty the HAPI latches. If 16-bit or 24-bit transfers are being made, Port 0 should be read last because reads from Port 0 assert the Latch Empty bit and the LatEmptyPin to signal the external device for more data.

The Latch Empty bit reads the opposite state from the external LatEmptyPin on pin P2[2]. If the LEMPTY Polarity bit is 0, LatEmptyPin is active HIGH, and the Latch Empty bit is active LOW.