Welcome to E-XFL.COM

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

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

What Are Embedded - Microcontrollers - Application Specific?

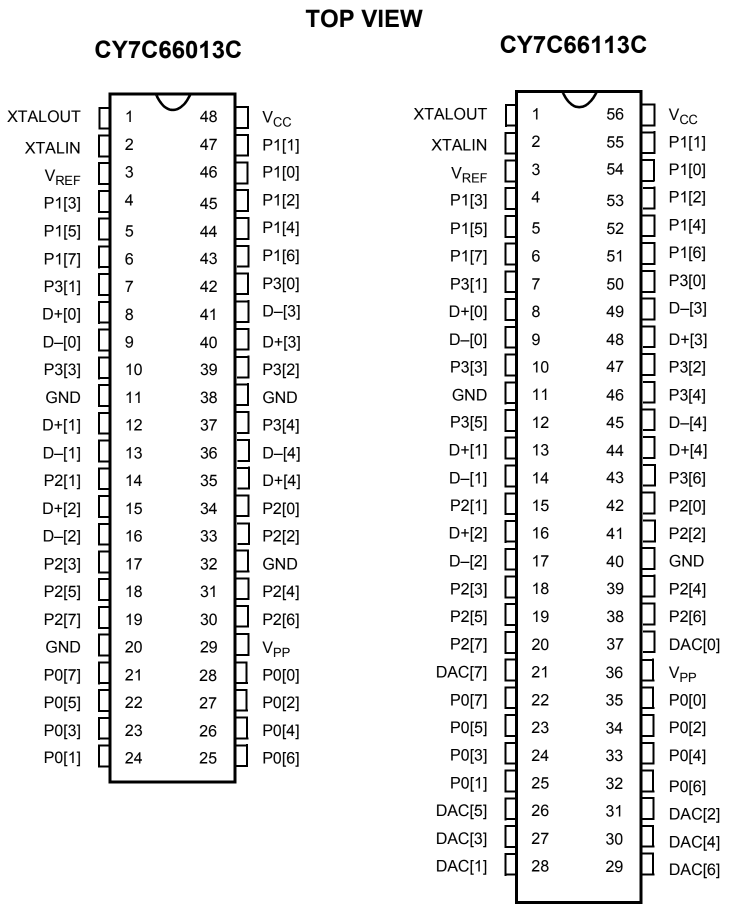
Application specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Hub/Microcontroller
Core Processor	M8
Program Memory Type	OTP (8kB)
Controller Series	USB Hub
RAM Size	256 x 8
Interface	I²C, USB, HAPI
Number of I/O	29
Voltage - Supply	4V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	48-BSSOP (0.295", 7.50mm Width)
Supplier Device Package	48-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c66013c-pvxc

Contents

Pin Configurations

Figure 1. CY7C66013C 48-pin SSOP and CY7C66113C 56-pin SSOP



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 3.3 V 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	17
Port 2 Data	0x02	R/W	GPIO Port 2 Data	17
Port 3 Data	0x03	R/W	GPIO Port 3 Data	17
Port 0 Interrupt Enable	0x04	W	Interrupt Enable for Pins in Port 0	19
Port 1 Interrupt Enable	0x05	W	Interrupt Enable for Pins in Port 1	19
Port 2 Interrupt Enable	0x06	W	Interrupt Enable for Pins in Port 2	19
Port 3 Interrupt Enable	0x07	W	Interrupt Enable for Pins in Port 3	19
GPIO Configuration	0x08	R/W	GPIO Port Configurations	18

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 29).

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

Table 5. Program Memory Space with Interrupt Vector Table

After Reset	Address	
14-bit PC	0x0000	Program execution begins here after a reset
	0x0002	USB bus reset interrupt vector
	0x0004	128 µs timer interrupt vector
	0x0006	1.024 ms timer interrupt vector
	0x0008	USB address A endpoint 0 interrupt vector
	0x000A	USB address A endpoint 1 interrupt vector
	0x000C	USB address A endpoint 2 interrupt vector
	0x000E	USB address B endpoint 0 interrupt vector
	0x0010	USB address B endpoint 1 interrupt vector
	0x0012	Hub interrupt vector
	0x0014	DAC interrupt vector
	0x0016	GPIO/HAPI interrupt vector
	0x0018	I^{2}C interrupt vector
	0x001A	**Program Memory begins here**
	0x1FDF	**8 kB (-32) PROM ends here.**

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 27. 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 2.5 V, and the oscillator is stable, the POR is deasserted and the on-chip timer starts counting. The first 1 ms of suspend time is not interruptible, and the semi-suspend state continues for an additional 95 ms unless the count is bypassed by a USB Bus Reset on the upstream port. The 95 ms provides time for V_{CC} to stabilize at a valid operating voltage before the chip executes code.

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

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

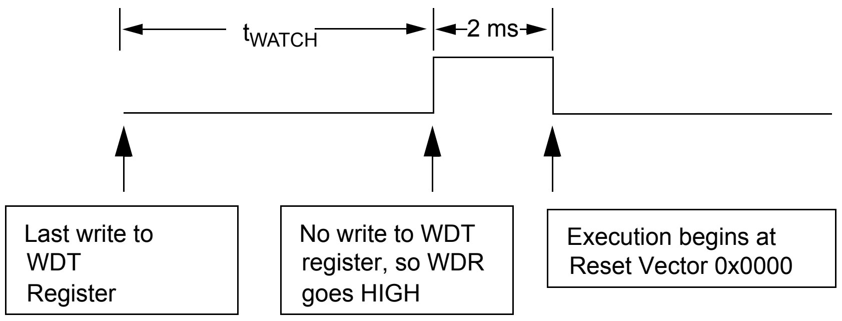
Watchdog Reset

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 on page 39). 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 5. Watchdog Reset


tWATCH
2 ms
Last write to WDT Register
No write to WDT register, so WDR goes HIGH
Execution begins at Reset Vector 0x0000

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.

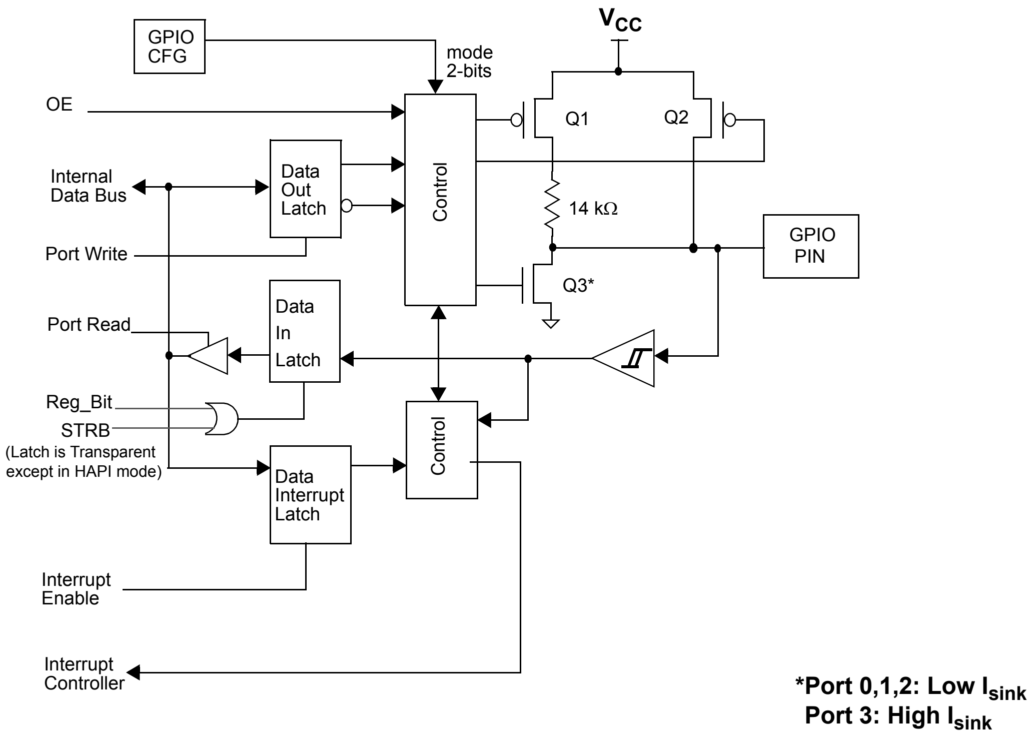
The GPIO interrupt allows the controller to wake up periodically and poll system components while maintaining a very low average power consumption. To achieve the lowest possible current during suspend mode, all I/O should be held at V_{CC} or Gnd. This also applies to internal port pins that may not 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 6. 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 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 Table 7 on page 17 through Table 10 on page 17, and are set to 1 on reset.

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 (Table 23 on page 23), bits 1 and 0.

Signals are provided on Port 2 to control the HAPI interface. Table 29 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 (Table 9 on page 17) 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 6 on page 16).

Table 29. 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 $\overline{\text{STB}}$ and $\overline{\text{OE}}$).
Bit	**Name**	**R/W**	**Description (HAPI and I^2C Configuration Register)**
2	Data Ready	R	Asserted after firmware writes data to Port 0, until $\overline{\text{OE}}$ driven LOW.
3	Latch Empty	R	Asserted after firmware reads data from Port 0, until $\overline{\text{STB}}$ driven LOW.
4	DRDY Polarity	R/W	Determines polarity of Data Ready bit and DReadyPin: If 0, Data Ready is active LOW, DReadyPin is active HIGH. If 1, Data Ready is active HIGH, DReadyPin is active LOW.
5	LEMPTY Polarity	R/W	Determines polarity of Latch Empty bit and LatEmptyPin: If 0, Latch Empty is active LOW, LatEmptyPin is active HIGH. If 1, Latch Empty is active HIGH, LatEmptyPin is active LOW.

HAPI Read by External Device from CY7C66x13C

In this case (see Figure 14 on page 54), 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 $\overline{\text{OE}}$ and $\overline{\text{CS}}$ pins active (LOW), which causes the HAPI data to be output on the port pins. When $\overline{\text{OE}}$ 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 16 on page 55), the external device drives the $\overline{\text{STB}}$ and $\overline{\text{CS}}$ 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 $\overline{\text{STB}}$ 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.

Processor Status and Control Register

Table 30. Processor Status and Control Register

Processor Status and Control **ADDRESS 0xFF**

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

Bit 0: Run

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

Bit 1: Reserved

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

Bit 2: Interrupt Enable Sense

This bit indicates whether interrupts are enabled or disabled. Firmware has no direct control over this bit as writing a zero or one to this bit position has no effect on interrupts. A ‘0’ indicates that interrupts are masked off and a ‘1’ indicates that the interrupts are enabled. This bit is further gated with the bit settings of the Global Interrupt Enable Register (Table 31 on page 28) and USB End Point Interrupt Enable Register (Table 32 on page 28). Instructions DI, EI, and RETI manipulate the state of this bit.

Bit 3: Suspend

Writing a ‘1’ to the Suspend bit halts the processor and cause the microcontroller to enter the suspend mode that significantly reduces power consumption. A pending, enabled interrupt or USB bus activity causes the device to come out of suspend. After coming out of suspend, the device resumes firmware execution at the instruction following the IOWR which put the part into suspend. An IOWR attempting to put the part into suspend is ignored if USB bus activity is present. See Suspend Mode on page 16 for more details on suspend mode operation.

Bit 4: Power on Reset

The POR is set to ‘1’ during a power on reset. The firmware checks bits 4 and 6 in the reset handler to determine whether a reset was caused by a power on condition or a Watchdog timeout. A POR event may be followed by a WDR before firmware begins executing, as explained here.

Bit 5: USB Bus Reset Interrupt

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

Bit 6: WDR

The WDR is set during a reset initiated by the WDT. This indicates the WDT went for more than t_{WATCH} (8 ms minimum) between Watchdog clears. This occurs with a POR event.

Bit 7: IRQ Pending

The IRQ pending, when set, indicates that one or more of the interrupts is recognized as active. An interrupt remains pending until its interrupt enable bit is set (Table 31 on page 28, Table 32 on page 28) and interrupts are globally enabled. At that point, the internal interrupt handling sequence clears this bit until another interrupt is detected as pending.

During power up, the Processor Status and Control Register is set to 00010001, which indicates a POR (bit 4 set) has occurred and no interrupts are pending (bit 7 clear). During the 96 ms suspend at start up (explained in Power on Reset on page 15), a WDR also occurs unless this suspend is aborted by an upstream SE0 before 8 ms. If a WDR occurs during the power up suspend interval, firmware reads 01010001 from the Status and Control Register after power up. Normally, the POR bit should be cleared so a subsequent WDR is clearly identified. If an upstream bus reset is received before firmware examines this register, the Bus Reset bit may also be set.

During a WDR, the Processor Status and Control Register is set to 01XX0001, which indicates a WDR (bit 6 set) has occurred and no interrupts are pending (bit 7 clear). The WDR does not effect the state of the POR and the Bus Reset Interrupt bits.

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

Table 33. 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 and HAPI interrupt
12	0x0018	I^2C interrupt

Interrupt Latency

Interrupt latency is 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 five clock cycle instruction such as JC is being executed when an interrupt occurs, the first instruction of the Interrupt Service Routine executes a minimum of 16 clocks (1+10+5) or a maximum of 20 clocks (5+10+5) after the interrupt is issued. For a 12 MHz internal clock (6 MHz crystal), 20 clock periods is 20/12 MHz = 1.667 μs.

USB Bus Reset Interrupt

The USB Controller recognizes a USB Reset when a Single Ended Zero (SE0) condition persists on the upstream USB port for 12–16 μs. SE0 is defined as the condition in which both the D+ line and the D– line are LOW. A USB Bus Reset may be recognized for an SE0 as short as 12 μs, but is always recognized for an SE0 longer than 16 μs. When a USB Bus Reset is detected, bit 5 of the Processor Status and Control Register (Table 30 on page 27) is set to record this event. In addition, the controller clears the following registers:

SIE Section:	USB Device Address Registers (0x10, 0x40)
Hub Section:	Hub Ports Connect Status (0x48)
	Hub Ports Enable (0x49)
	Hub Ports Speed (0x4A)
	Hub Ports Suspend (0x4D)
	Hub Ports Resume Status (0x4E)
	Hub Ports SE0 Status (0x4F)
	Hub Ports Data (0x50)
	Hub Downstream Force (0x51).

A USB Bus Reset Interrupt is generated at the end of the USB Bus Reset condition when the SE0 state is deasserted. If the USB reset occurs during the start up delay following a POR, the delay is aborted as described in Power on Reset on page 15.

Timer Interrupt

There are two periodic timer interrupts: the 128 μs interrupt and the 1.024 ms interrupt. The user should disable both timer interrupts before going into the suspend mode to avoid possible conflicts between servicing the timer interrupts first or the suspend request first.

USB Endpoint Interrupts

There are five USB endpoint interrupts, one per endpoint. A USB endpoint interrupt is generated after the USB host writes to a USB endpoint FIFO or after the USB controller sends a packet to the USB host. The interrupt is generated on the last packet of the transaction. For example, on the host's ACK during an IN, or on the device ACK during on OUT. If no ACK is received during an IN transaction, no interrupt is generated.

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 (Table 36 on page 34). The connect and disconnect event on a port does not generate an interrupt if the SIE does not drive the port (that is, the port is being forced).

DAC Interrupt

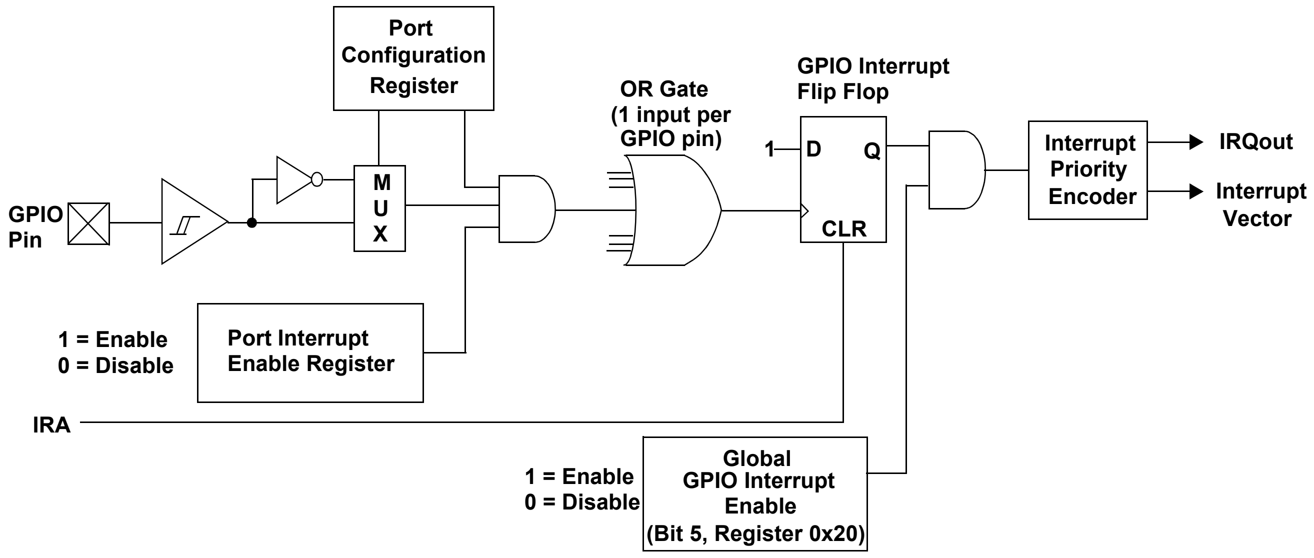
Each DAC I/O pin generates an interrupt, if enabled. The interrupt polarity for each DAC I/O pin is programmable. A positive polarity is a rising edge input while a negative polarity is a falling edge input. All of the DAC pins share a single interrupt vector, which means the firmware needs to read the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

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

GPIO and HAPI Interrupt

Each of the GPIO pins generates an interrupt, if enabled. The interrupt polarity is programmed for each GPIO port as part of the GPIO configuration. All of the GPIO pins share a single interrupt vector, which means the firmware needs to read the GPIO ports with enabled interrupts to determine which pin or pins caused an interrupt. A block diagram of the GPIO interrupt logic is shown in Figure 10.
Refer to GPIO Configuration Port on page 18 and GPIO Interrupt Enable Ports on page 19 for more information about setting GPIO interrupt polarity and enabling individual GPIO interrupts.

Figure 10. GPIO Interrupt Structure


Port Configuration Register
OR Gate (1 input per GPIO pin)
GPIO Interrupt Flip Flop
1
D
Q
CLR
M U X
GPIO Pin
Interrupt Priority Encoder
IRQout
Interrupt Vector
1 = Enable
0 = Disable
Port Interrupt Enable Register
IRA
1 = Enable
0 = Disable
Global GPIO Interrupt Enable (Bit 5, Register 0x20)

If one port pin has triggered an interrupt, no other port pins 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 and bits not 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 and bits not used by HAPI still effect the CMOS mode operation of those ports and bits. The effect of modifying the interrupt bits while the Port Config bits are set to ‘10’ is shown in Table 12 on page 18. The events that generate HAPI interrupts are described in Hardware Assisted Parallel Interface (HAPI) on page 26.

I^2C Interrupt

The I^2C interrupt occurs after various events on the I^2C compatible bus to signal the need for firmware interaction. This generally involves reading the I^2C Status and Control Register (Table 27 on page 24) to determine the cause of the interrupt, loading and reading the I^2C Data Register as appropriate, and finally writing the Processor Status and Control Register (Table 30 on page 27) to initiate the subsequent transaction. The interrupt indicates that status bits are stable and it is safe to read and write the I^2C registers.

When enabled, the I^2C compatible state machines generate interrupts on completion of the following conditions. The referenced bits are in the I^2C Status and Control Register.

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

The *Continue/Busy* bit is cleared by hardware prior to interrupt conditions 1 to 4. When the Data Register is read or written, firmware should configure the other control bits and set the *Continue/Busy* bit for subsequent transactions. Following an interrupt from master mode, firmware should perform only one write to the Status and Control Register that sets the *Continue/Busy* bit, without checking the value of the *Continue/Busy* bit. The Busy bit may otherwise be active and I^2C register contents may be changed by the hardware during the transaction, until the I^2C interrupt occurs.

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 full speed Serial Interface Engine. An external series resistor of R_{ext} must be placed in series with all upstream and downstream USB outputs to meet the USB driver requirements of the USB specification. The CY7C66x13C microcontroller provides the functionality of a compound device consisting of a USB hub and permanently attached functions.

USB Serial Interface Engine

The SIE allows the CY7C66x13C 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 and unstuffing
- Checksum generation and 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 and Resume coordination
- Verify and select DATA toggle values.

USB Enumeration

The internal hub and any compound device function are enumerated under firmware control. The hub is enumerated first, followed by any integrated compound function. After the hub is enumerated, the USB host reads hub connection status to determine which (if any) of the downstream ports need to be enumerated. The following is a brief summary of the typical enumeration process of the CY7C66x13C by the USB host. For a detailed description of the enumeration process, refer to the USB specification.

In this description, “Firmware” refers to embedded firmware in the CY7C66x13C controller.

1. The host computer sends a SETUP packet followed by a DATA packet to USB address 0 requesting the Device descriptor.
2. Firmware decodes the request and retrieves its Device descriptor from the program memory tables.
3. The host computer performs a control read sequence and Firmware responds by sending the Device descriptor over the USB bus, via the on-chip FIFOs.
4. After receiving the descriptor, the host sends a SETUP packet followed by a DATA packet to address 0 assigning a new USB address to the device.

5. Firmware stores the new address in its USB Device Address Register (for example, as Address B) after the no data control sequence completes.
6. The host sends a request for the Device descriptor using the new USB address.
7. Firmware decodes the request and retrieves the Device descriptor from program memory tables.
8. The host performs a control read sequence and Firmware responds by sending its Device descriptor over the USB bus.
9. The host generates control reads from the device to request the Configuration and Report descriptors.
10. When the device receives a Set Configuration request, its functions may now be used.
11. Following enumeration as a hub, Firmware optionally indicates to the host that a compound device exists (for example, the keyboard in a keyboard/hub device).
12. The host carries out the enumeration process with this additional function as though it were attached downstream from the hub.
13. When the host assigns an address to this device, it is stored as the other USB address (for example, Address A).

USB Hub

A USB hub is required to support:

- Connectivity behavior: service connect and disconnect detection
- Bus fault detection and recovery
- Full and 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 CY7C66013C and CY7C66113C 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–). Each downstream port provided by the hub requires external R_{UDN} resistors from each signal line to ground, so that when a downstream port has no device connected, the hub reads a LOW (zero) on both D+ and D–. This condition is used to identify the “no connect” state.

The hub must have a resistor R_{UUP} connected between its upstream D+ line and V_{REG} to indicate it is a full speed USB device.

The hub generates an EOP at EOF1, in accordance with the USB 1.1 Specification, Section 11.2.2.

Connecting and Disconnecting a USB Device

A low speed (1.5 Mbps) USB device has 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 sees a LOW on D+ and a HIGH on D–. This causes the hub repeater to set a connect bit in the Hub Ports Connect Status register for the downstream port. Then the hub repeater generates a Hub Interrupt to notify the microcontroller that there is a change in the Hub downstream status.

A full speed (12 Mbps) USB device has a pull up resistor from the D+ pin, so the hub sees a HIGH on D+ and a LOW on D–. In this case, the hub repeater sets a connect bit in the Hub Ports Connect Status register, clears a bit in the Hub Ports Speed register (for full speed), and generates a Hub Interrupt to notify the microcontroller of the change in Hub status. The firmware sets the speed of this port in the Hub Ports Speed Register (see Table 35 on page 34).

Connects are recorded by the time a non SE0 state lasts for more than 2.5 μs on a downstream port.

When a USB device is disconnected from the Hub, the downstream signal pair eventually floats to a single ended zero state. The hub repeater recognizes a disconnect when the SE0 state on a downstream port lasts from 2.0 to 2.5 μs. On a disconnect, the corresponding bit in the Hub Ports Connect Status register is cleared, and the Hub Interrupt is generated.

Table 34. Hub Ports Connect Status

Hub Ports Connect Status **ADDRESS 0x48**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Port 4 Connect Status	Port 3 Connect Status	Port 2 Connect Status	Port 1 Connect Status
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

Bit [0..3]: Port x Connect Status (where x = 1..4)

When set to 1, Port x is connected; When set to 0, Port x is disconnected.

Bit [7..4]: Reserved.

The Hub Ports Connect Status register is cleared to zero by reset or USB bus 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.

USB Upstream Port Status and Control

USB status and control is regulated by the USB Status and Control Register, as shown in Table 44. All bits in the register are cleared during reset.

Table 44. USB Status and Control Register

USB Status and Control **ADDRESS 0x1F**

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

Bits[2..0]: Control Action

Set to control action as per Table 45.The three control bits allow the upstream port to be driven manually by firmware. For normal USB operation, all of these bits must be cleared. Table 45 shows how the control bits affect the upstream port.

Table 45. 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

Bit 3: Bus Activity

This is a “sticky” bit that indicates if any non idle USB event has occurred on the upstream USB port. 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 clears the Bus Activity bit, but only the SIE can set it.

Bits 4 and 5: D– Upstream and D+ Upstream

These bits give the state of each upstream port pin individually: 1 = HIGH, 0 = LOW.

Bit 6: Endpoint Mode

This bit used to configure the number of USB endpoints. See USB Device Endpoints on page 39 for a detailed description.

Bit 7: Endpoint Size

This bit used to configure the number of USB endpoints. See USB Device Endpoints on page 39 for a detailed description.

The hub generates an EOP at EOF1 in accordance with the USB 1.1 Specification.

Endpoint Mode and Count Registers Update and Locking Mechanism

The contents of the endpoint mode and counter registers are updated, based on the packet flow diagram in Figure 11 on page 43. Two time points, UPDATE and SETUP, are shown in the same figure. The following activities occur at each time point:

SETUP:

The SETUP bit of the endpoint 0 mode register is forced HIGH at this time. This bit is forced HIGH by the SIE until the end of the data phase of a control write transfer. The SETUP bit can not be cleared by firmware during this time.

The affected mode and counter registers of endpoint 0 are locked from any CPU writes when they are updated. These registers are unlocked by a CPU read, only if the read operation occurs after the UPDATE. The firmware needs to perform a register read as a part of the endpoint ISR processing to unlock the effected registers. The locking mechanism on mode and counter registers ensures that the firmware recognizes the changes that the SIE might have made since the previous I/O read of that register.

UPDATE:

1. Endpoint Mode Register – All the bits are updated (except the SETUP bit of the endpoint 0 mode register).
2. Counter Registers – All bits are updated.
3. Interrupt – If an interrupt is to be generated as a result of the transaction, the interrupt flag for the corresponding endpoint is set at this time. For details on what conditions are required to generate an endpoint interrupt, refer to Table 52 on page 45.
4. The contents of the updated endpoint 0 mode and counter registers are locked, except the SETUP bit of the endpoint 0 mode register which was locked earlier.

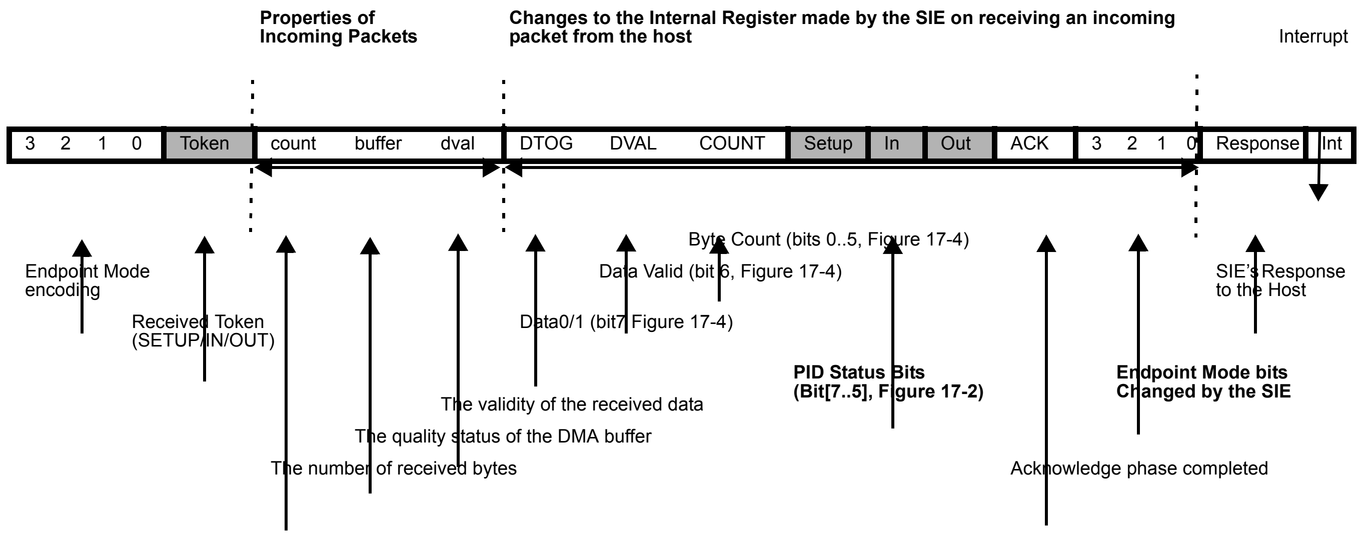
Comments

Some Mode Bits are automatically changed by the SIE in response to certain USB transactions. For example, if the Mode Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode as shown in Table 47 on page 39, the SIE changes the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACK'ing a valid status stage OUT token. The firmware needs to update the mode for the SIE to respond appropriately. See Table 38 on page 35 for more details on what modes are changed by the SIE. A disabled endpoint remains disabled until changed by firmware, and all endpoints reset to the disabled mode (0000). Firmware normally enables the endpoint mode after a SetConfiguration request.

Any SETUP packet to an enabled endpoint with mode set to accept SETUPs are changed by the SIE to 0001 (NAKing INs and OUTs). Any mode set to accept a SETUP sends an ACK handshake to a valid SETUP token.

The control endpoint has three status bits for identifying the token type received (SETUP, IN, or OUT), but the endpoint must be placed in the correct mode to function as such. Non control endpoints should not be placed into modes that accept SETUPs. Note that most modes that control transactions involving an ending ACK, are changed by the SIE to a corresponding mode which NAKs subsequent packets following the ACK. Exceptions are modes 1010 and 1110.

Table 52. Decode Table for Table 53


Properties of Incoming Packets
Changes to the Internal Register made by the SIE on receiving an incoming packet from the host
Interrupt
3 2 1 0
Token
count
buffer
dval
DTOG
DVAL
COUNT
Setup
In
Out
ACK
3 2 1 0
Response
Int
Endpoint Mode encoding
Received Token (SETUP/IN/OUT)
The number of received bytes
The quality status of the DMA buffer
The validity of the received data
Data0/1 (bit7 Figure 17-4)
Data Valid (bit 6, Figure 17-4)
Byte Count (bits 0..5, Figure 17-4)
PID Status Bits (Bit[7..5], Figure 17-2)
Acknowledge phase completed
Endpoint Mode bits Changed by the SIE
SIE's Response to the Host

Legend:

TX: transmit	UC : unchanged
RX: receive	TX0:Transmit 0 length packet
▨	available for Control endpoint only
x: don't care	

The response of the SIE are summarized as follows:

- The SIE only responds to valid transactions, and ignores invalid ones.
- The SIE generates an interrupt when a valid transaction is completed or when the FIFO is corrupted. FIFO corruption occurs during an OUT or SETUP transaction to a valid internal address, that ends with a invalid CRC.
- An incoming Data packet is valid if the count is $\leq$ Endpoint Size + 2 (includes CRC) and passes all error checking.
- An IN is 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 register and the Count register are locked to CPU writes at the end of any transaction to that endpoint in which an ACK is transferred. These registers are only unlocked by a CPU read of the register, which should be done by the firmware only after the transaction is complete. This represents about a 1 µs window in which the CPU is locked from register writes to these USB registers. Normally the firmware should perform a register read at the beginning of the Endpoint ISRs 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. Note that the setup bit of the mode register is NOT locked. This means that before writing to the mode register, firmware must first read the register to make sure that the setup bit is not set (which indicates a setup was received, while processing the current USB request). This read unlocks the register. So care must be taken not to overwrite the register elsewhere.

Table 53. Details of Modes for Differing Traffic Conditions (see Table 52 on page 45 for the decode legend) *(continued)*

Nak In/premature status Out																				
1	1	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	No Change				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	No Change				ignore	no
1	1	1	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
1	1	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	No Change				NAK	yes
Status Out/extra In																				
0	0	1	0	Out	2	UC	valid	1	1	updates	UC	UC	1	1	No Change				ACK	yes
0	0	1	0	Out	2	UC	valid	0	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
0	0	1	0	Out	!=2	UC	valid	updates	1	updates	UC	UC	1	UC	0	0	1	1	Stall	yes
0	0	1	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
0	0	1	0	Out	x	UC	invalid	UC	UC	UC	UC	1	UC	UC	No Change				ignore	no
0	0	1	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	0	0	1	1	Stall	yes
OUT ENDPOINT																				
Properties of Incoming Packet								**Changes made by SIE to Internal Registers and Mode Bits**												
Mode Bits				**token**	**count**	**buffer**	**dval**	**DTOG**	**DVAL**	**COUNT**	**Setup**	**In**	**Out**	**ACK**	**Mode Bits**				**Response**	**Intr**
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	No Change				ignore	yes
1	0	0	1	Out	x	junk	invalid	updates	0	updates	UC	UC	1	UC	No Change				ignore	yes
1	0	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
																			(STALL[4] = 0)	
1	0	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				Stall	no
																			(STALL[4] = 1)	
NAK Out/erroneous In																				
1	0	0	0	Out	<= 10	UC	valid	UC	UC	UC	UC	UC	1	UC	No Change				NAK	yes
1	0	0	0	Out	> 10	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
1	0	0	0	Out	x	UC	invalid	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
1	0	0	0	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
Isochronous endpoint (Out)																				
0	1	0	1	Out	x	updates	updates	updates	updates	updates	UC	UC	1	1	No Change				RX	yes
0	1	0	1	In	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
IN ENDPOINT																				
Properties of Incoming Packet								**Changes made by SIE to Internal Registers and Mode Bits**												
Mode Bits				**token**	**count**	**buffer**	**dval**	**DTOG**	**DVAL**	**COUNT**	**Setup**	**In**	**Out**	**ACK**	**Mode Bits**				**Response**	**Intr**
Normal In/erroneous Out																				
1	1	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				ignore	no
																			(STALL[4] = 0)	
1	1	0	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change				stall	no
																			(STALL[4] = 1)	
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	No Change				ignore	no

Table 53. Details of Modes for Differing Traffic Conditions (see Table 52 on page 45 for the decode legend) *(continued)*

1	1	0	0	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	No Change	NAK	yes
Isochronous endpoint (In)																	
0	1	1	1	Out	x	UC	x	UC	UC	UC	UC	UC	UC	UC	No Change	ignore	no
0	1	1	1	In	x	UC	x	UC	UC	UC	UC	1	UC	UC	No Change	TX	yes

Register Summary

	Addr	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Read/Write/ Both[5, 6, 7]	Default/ Reset [8]
GPIO CONFIGURATION PORTS 0, 1, 2 AND 3	0x00	Port 0 Data	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0	bbbbbbbb	11111111
	0x01	Port 1 Data	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	bbbbbbbb	11111111
	0x02	Port 2 Data	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	bbbbbbbb	11111111
	0x03	Port 3 Data	Reserved	P3.6 CY7C66113C only	P3.5 CY7C66113C only	P3.4	P3.3	P3.2	P3.1	P3.0	bbbbbbbb	-1111111
	0x04	Port 0 Interrupt Enable	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	wwwwwwww	00000000
	0x05	Port 1 Interrupt Enable	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	wwwwwwww	00000000
	0x06	Port 2 Interrupt Enable	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	wwwwwwww	00000000
	0x07	Port 3 Interrupt Enable	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	wwwwwwww	00000000
	0x08	GPIO Configuration	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	bbbbbbbb	00000000
HAPI I^{2}C	0x09	HAPI/I^{2}C Configuration	I^{2}C Position	Reserved	LEMPTY Polarity	DRDY Polarity	Latch Empty	Data Ready	Port Width bit 1	Port Width bit 0	b-bbrrbb	00000000
Endpoint A0, AI and A2 Configuration	0x10	USB Device Address A	Device Address A Enable	Device Address A Bit 6	Device Address A Bit 5	Device Address A Bit 4	Device Address A Bit 3	Device Address A Bit 2	Device Address A Bit 1	Device Address A Bit 0	bbbbbbbb	00000000
Endpoint A0, AI AND A2 Configuration	0x11	EP A0 Counter Register	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	bbbbbbbb	00000000
	0x12	EP A0 Mode Register	Endpoint0 SETUP Received	Endpoint0 IN Received	Endpoint0 OUT Received	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0	bbbbbbbb	00000000
	0x13	EP A1 Counter Register	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	bbbbbbbb	00000000
	0x14	EP A1 Mode Register	STALL	-	-	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0	bbbbbbbb	00000000
	0x15	EP A2 Counter Register	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	bbbbbbbb	00000000
	0x16	EP A2 Mode Register	STALL	-	-	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0	bbbbbbbb	00000000
USB-CS	0x1F	USB Status and Control	Endpoint Size	Endpoint Mode	D+ Upstream	D– Upstream	Bus Activity	Control Bit 2	Control Bit 1	Control Bit 0	bbrrbbbb	-0xx0000

Notes
5. B: Read and Write.
6. W: Write.
7. R: Read.
8. X: Unknown

Register Summary *(continued)*

	Addr	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Read/Write/ Both[5, 6, 7]	Default/ Reset [8]
INTERRUPT	0x20	Global Interrupt Enable	Reserved	I^{2}C Interrupt Enable	GPIO Interrupt Enable	DAC Interrupt Enable	USB Hub Interrupt Enable	1.024-ms Interrupt Enable	128 µs Interrupt Enable	USB Bus RESET Interrupt Enable	-bbbbbbb	-0000000
	0x21	Endpoint Interrupt Enable	Reserved	Reserved	Reserved	EPB1 Interrupt Enable	EPB0 Interrupt Enable	EPA2 Interrupt Enable	EPA1 Interrupt Enable	EPA0 Interrupt Enable	---bbbbb	---00000
TIMER	0x24	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	rrrrrrrr	00000000
	0x25	Timer (MSB)	Reserved	Reserved	Reserved	Reserved	Timer Bit 11	Timer Bit 10	Time Bit 9	Timer Bit 8	----rrrr	----0000
I^{2}C	0x28	I^{2}C Control and Status	MSTR Mode	Continue/ Busy	Xmit Mode	ACK	Addr	ARB Lost/ Restart	Received Stop	I^{2}C Enable	bbbbbbbb	00000000
	0x29	I^{2}C Data	I^{2}C Data 7	I^{2}C Data 6	I^{2}C Data 5	I^{2}C Data 4	I^{2}C Data 3	I^{2}C Data 2	I^{2}C Data 1	I^{2}C Data 0	bbbbbbbb	xxxxxxxx
ENDPOINT B0, B1 CONFIGURATION	0x40	USB Device Address B	Device Address B Enable	Device Address B Bit 6	Device Address B Bit 5	Device Address B Bit 4	Device Address B Bit 3	Device Address B Bit 2	Device Address B Bit 1	Device Address B Bit 0	bbbbbbbb	00000000
	0x41	EP B0 Counter Register	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	bbbbbbbb	00000000
	0x42	EP B0 Mode Register	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	bbbbbbbb	00000000
	0x43	EP B1 Counter Register	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	bbbbbbbb	00000000
	0x44	EP B1 Mode Register	STALL	-	-	ACK	Mode Bit 3	Mode Bit 2	Mode Bit 1	Mode Bit 0	b--bbbbb	00000000
HUB PORT CONTROL, STATUS, SUSPEND RESUME, SE0, FORCE LOW	0x48	Hub Port Connect Status	Reserved	Reserved	Reserved	Reserved	Port 4 Connect Status	Port 3 Connect Status	Port 2 Connect Status	Port 1 Connect Status	----bbbb	00000000
	0x49	Hub Port Enable	Reserved	Reserved	Reserved	Reserved	Port 4 Enable	Port 3 Enable	Port 2 Enable	Port 1 Enable	----bbbb	00000000
	0x4A	Hub Port Speed	Reserved	Reserved	Reserved	Reserved	Port 4 Speed	Port 3 Speed	Port 2 Speed	Port 1 Speed	----bbbb	00000000
	0x4B	Hub Port Control (Ports 4:1)	Port 4 Control Bit 1	Port 4 Control Bit 0	Port 3 Control Bit 1	Port 3 Control Bit 0	Port 2 Control Bit 1	Port 2 Control Bit 0	Port 1 Control Bit 1	Port 1 Control Bit 0	bbbbbbbb	00000000
	0x4D	Hub Port Suspend	Device Remote Wakeup	Reserved	Reserved	Reserved	Port 4 Selective Suspend	Port 3 Selective Suspend	Port 2 Selective Suspend	Port 1 Selective Suspend	b---bbbb	00000000
	0x4E	Hub Port Resume Status	Reserved	Reserved	Reserved	Reserved	Resume 4	Resume 3	Resume 2	Resume 1	----rrrr	00000000
	0x4F	Hub Port SE0 Status	Reserved	Reserved	Reserved	Reserved	Port 4 SE0 Status	Port 3 SE0 Status	Port 2 SE0 Status	Port 1 SE0 Status	----rrrr	00000000
	0x50	Hub Ports Data	Reserved	Reserved	Reserved	Reserved	Port 4 Diff. Data	Port 3 Diff. Data	Port 2 Diff. Data	Port 1 Diff. Data	----rrrr	00000000
	0x51	Hub Port Force Low (Ports 4:1)	Force Low D+[4]	Force Low D–[4]	Force Low D+[3]	Force Low D–[3]	Force Low D+[2]	Force Low D–[2]	Force Low D+[1]	Force Low D–[1]	bbbbbbbb	00000000
	0xFF	Process Status & Control	IRQ Pending	WDR	USB Bus Reset Interrupt	Power-on Reset	Suspend	Interrupt Enable Sense	Reserved	Run	rbbbbrbb	00010001

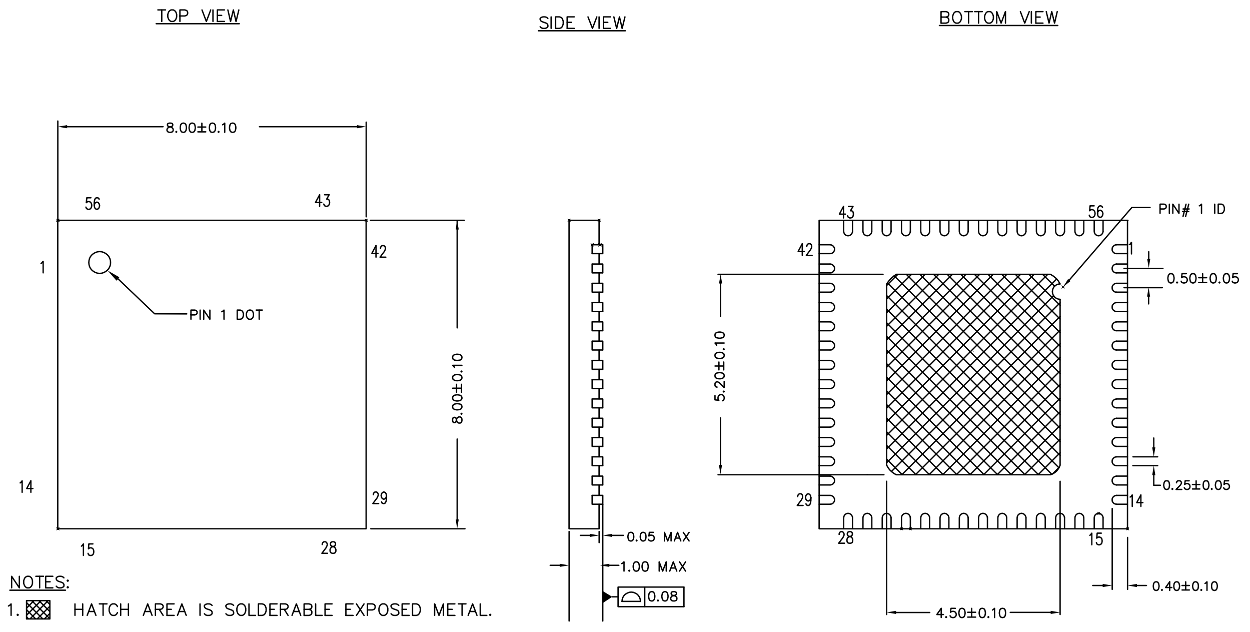
Switching Characteristics

(F_{OSC} = 6.0 MHz)

Parameter	Description	Min	Max	Unit
Clock Source				
f_{OSC}	Clock Rate	6 ± 0.25%	–	MHz
t_{cyc}	Clock Period	166.25	167.08	ns
t_{CH}	Clock HIGH time	0.45 t_{CYC}	–	ns
t_{CL}	Clock LOW time	0.45 t_{CYC}	–	ns
USB Full Speed Signaling[15]				
t_{rfs}	Transition Rise Time	4	20	ns
t_{ffs}	Transition Fall Time	4	20	ns
t_{rfmfs}	Rise/Fall Time Matching; (t_r/t_f)	90	111	%
$t_{dratefs}$	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 Pulse Width	15	–	ns
t_{OED}	OE LOW to Data Valid[16, 17]	–	40	ns
t_{OEZ}	OE HIGH to Data High Z[17]	–	20	ns
t_{OEDR}	OE LOW to Data_Ready Deasserted[16, 17]	0	60	ns
HAPI Write Cycle Timing				
t_{WR}	Write Strobe Width	15	–	ns
t_{DSTB}	Data Valid to STB HIGH (Data Setup Time)[17]	5	–	ns
t_{STBZ}	STB HIGH to Data High Z (Data Hold Time)[17]	15	–	ns
t_{STBLE}	STB LOW to Latch_Empty Deasserted[16, 17]	0	50	ns
Timer Signals				
t_{watch}	WDT Period	8.192	14.336	ms

Notes
14.
15. Per Table 7-6 of revision 1.1 of USB specification.
16. For 25 pF load.
17. Assumes chip select $\overline{CS}$ is asserted (LOW).

Figure 19. 56-pin QFN (8 × 8 × 1.0 mm) 4.5 × 5.2 mm E-Pad (Sawn) Package Outline



NOTES:
1. ▨ HATCH AREA IS SOLDERABLE EXPOSED METAL.
2. REFERENCE JEDEC#: MO-220
3. PACKAGE WEIGHT: 162 ± 16 mg
4. ALL DIMENSIONS ARE IN MILLIMETERS

001-53450 *D