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	31
Voltage - Supply	4V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Surface Mount
Package / Case	56-BSSOP (0.295", 7.50mm Width)
Supplier Device Package	56-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c66113c-pvxc

Contents

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

Table 7. 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

Table 8. 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

Table 9. Port 2 Data

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

Table 10. 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) on page 26). 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 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 on page 31.

Table 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

Table 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

Table 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

Table 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

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 Table 23)[3]. All bits of this register are cleared on reset.

Table 23. 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) on page 26. Table 24 shows the HAPI port configurations, and Table 25 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) on page 26. The I²C compatible interface must be separately enabled.

Table 24. HAPI Port Configuration

Port Width (Bit 0 and 1, Figure 23)	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 25. I²C Port Configuration

I²C Position (Bit 7, Table 23 on page 23)	I²C Port Width (Bit 1, Table 23 on page 23)	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) on page 26.

The I²C compatible interface consists of two registers, an I²C Data Register (Table 14 on page 19) and an I²C Status and Control Register (Table 27 on page 24). 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 on page 23 for the bit definitions and functionality of the HAPI and I²C 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} = 2.0 V 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.

Table 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.

Table 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 28, with a more detailed description following.

Table 28. I²C Status and Control Register Bit Definitions

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

Bit 7: MSTR Mode

Setting this bit to 1 causes the I²C compatible block to initiate a master mode transaction by sending a start bit and transmitting the first data byte from the data register (this typically holds the target address and R/W bit). Subsequent bytes are initiated by setting the Continue bit, as described later in this section.

Clearing this bit (set to 0) causes the GPIO pins to operate normally. In master mode, the I²C compatible block generates the clock (SCK), and drives the data line as required depending on transmit or receive state. The I²C compatible block performs any required arbitration and clock synchronization. IN the event of a loss of arbitration, this MSTR bit is cleared, the ARB Lost bit is set, and an interrupt is generated by the microcontroller. If the chip is the target of an external master that wins arbitration, then the interrupt is held off until the transaction from the external master is completed.

When MSTR Mode is cleared from 1 to 0 by a firmware write, an I²C Stop bit is generated.

Bit 6: Continue/Busy

This bit is written by the firmware to indicate that the firmware is ready for the next byte transaction to begin. In other words, the bit has responded to an interrupt request and has completed the required update or read of the data register. During a read this bit indicates if the hardware is busy and is locking out additional writes to the I²C Status and Control register. This locking allows the hardware to complete certain operations that may require an extended period of time. Following an I²C interrupt, the I²C compatible block does not return to the Busy state until firmware sets the Continue bit. This allows the firmware to make one control register write without the need to check the Busy bit.

Bit 5: Xmit Mode

This bit is set by firmware to enter transmit mode and perform a data transmit in master or slave mode. Clearing this bit sets the part in receive mode. Firmware generally determines the value of this bit from the R/W bit associated with the I²C address packet. The Xmit Mode bit state is ignored when initially writing the MSTR Mode or the Restart bits, as these cases always cause transmit mode for the first byte.

Bit 4: ACK

This bit is set or cleared by firmware during receive operation to indicate if the hardware should generate an ACK signal on the I²C compatible bus. Writing a 1 to this bit generates an ACK (SDA LOW) on the I²C compatible bus at the ACK bit time. During transmits (Xmit Mode = 1), this bit should be cleared.

Bit 3: Addr

This bit is set by the I²C compatible block during the first byte of a slave receive transaction, after an I²C start or restart. The Addr bit is cleared when the firmware sets the Continue bit. This bit allows the firmware to recognize when the master has lost arbitration, and in slave mode it allows the firmware to recognize that a start or restart has occurred.

Bit 2: ARB Lost/Restart

This bit is valid as a status bit (ARB Lost) after master mode transactions. In master mode, set this bit (along with the Continue and MSTR Mode bits) to perform an I²C restart sequence. The I²C target address for the restart must be written to the data register before setting the Continue bit. To prevent false ARB Lost signals, the Restart bit is cleared by hardware during the restart sequence.

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 I²C transaction by not acknowledging the previous byte transmitted on the I²C compatible bus. For example, in receive mode if firmware sets the Continue bit and clears the ACK bit.

Bit 0: I²C Enable

Set this bit to override GPIO definition with I²C compatible function on the two I²C compatible pins. When this bit is cleared, these pins are free to function as GPIOs. In I²C 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 (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 $\overline{STB}$ 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: 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 $\overline{OE}$ 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 16 on page 55), 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.

Interrupts

Interrupts are generated by the GPIO and DAC pins, the internal timers, I²C compatible or HAPI operation, the internal USB hub, or on various USB traffic conditions. All interrupts are maskable by the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register. Writing a '1' to a bit position enables the interrupt associated with that bit position.

Table 31. Global Interrupt Enable Register

Global Interrupt Enable Register **ADDRESS 0X20**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	I²C Interrupt Enable	GPIO Interrupt Enable	DAC Interrupt Enable	USB Hub Interrupt Enable	1.024 ms Interrupt Enable	128 µs Interrupt Enable	USB Bus RST Interrupt Enable
Read/Write	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	0	0	0	0	0	0	0

Bit 0: USB Bus RST Interrupt Enable

1 = Enable Interrupt on a USB Bus Reset;
0 = Disable interrupt on a USB Bus Reset
(Refer to USB Bus Reset Interrupt on page 30).

Bit 1: 128 µs Interrupt Enable

1 = Enable Timer interrupt every 128 µs;
0 = Disable Timer Interrupt for every 128 µs.

Bit 2: 1.024 ms Interrupt Enable

1= Enable Timer interrupt every 1.024 ms;
0 = Disable Timer Interrupt every 1.024 ms.

Bit 3: USB Hub Interrupt Enable

1 = Enable Interrupt on a Hub status change;
0 = Disable interrupt due to hub status change.
(Refer to USB Hub Interrupt on page 30.)

Bit 4: DAC Interrupt Enable

1 = Enable DAC Interrupt; 0 = Disable DAC interrupt.

Bit 5: GPIO Interrupt Enable

1 = Enable Interrupt on falling and rising edge on any GPIO; 0 = Disable Interrupt on falling and rising edge on any GPIO. (Refer to sections GPIO and HAPI Interrupt on page 31, GPIO Configuration Port on page 18, and GPIO Interrupt Enable Ports on page 19.)

Bit 6: I²C Interrupt Enable

1 = Enable Interrupt on I2C related activity;
0 = Disable I2C related activity interrupt.
(Refer to I²C Interrupt on page 32.)

Bit 7: Reserved.

Table 32. USB Endpoint Interrupt Enable Register

USB Endpoint Interrupt Enable **ADDRESS 0X21**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	EPB1 Interrupt Enable	EPB0 Interrupt Enable	EPA2 Interrupt Enable	EPA1 Interrupt Enable	EPA0 Interrupt Enable
Read/Write	-	-	-	R/W	R/W	R/W	R/W	R/W
Reset	-	-	-	0	0	0	0	0

Bit 0: EPA0 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint A0;
0 = Disable Interrupt on data activity through endpoint A0.

Bit 1: EPA1 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint A1;
0 = Disable Interrupt on data activity through endpoint A1.

Bit 2: EPA2 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint A2;
0 = Disable Interrupt on data activity through endpoint A2.

Bit 3: EPB0 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint B0;
0 = Disable Interrupt on data activity through endpoint B0.

Bit 4: EPB1 Interrupt Enable

1 = Enable Interrupt on data activity through endpoint B1;
0 = Disable Interrupt on data activity through endpoint B1.

Bit [7..5]: Reserved

During a reset, the contents the Global Interrupt Enable Register and USB End Point Interrupt Enable Register are cleared, effectively, disabling all interrupts.

The interrupt controller contains a separate flip flop for each interrupt. See Figure 9 on page 29 for the logic block diagram of the interrupt controller. When an interrupt is generated, it is first registered as a pending interrupt. It stays pending until it is serviced or a reset occurs. A pending interrupt only generates an interrupt request if it is enabled by the corresponding bit in the interrupt enable registers. The highest priority interrupt request

DAC Interrupt

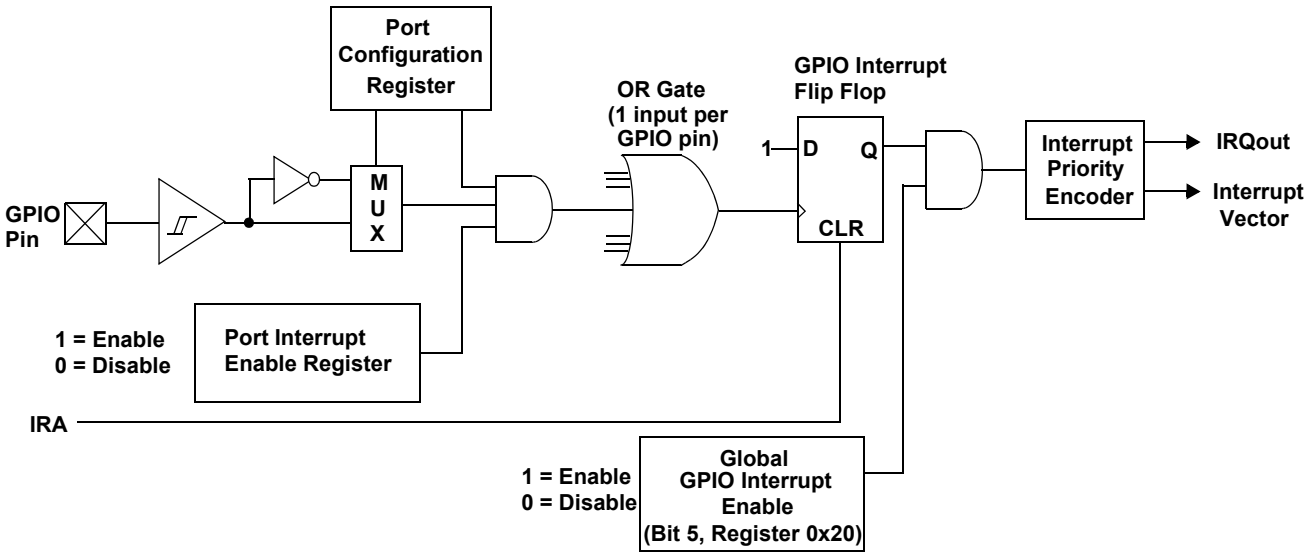
Each DAC I/O pin generates an interrupt, if enabled. The interrupt polarity for each DAC I/O pin is programmable. A positive polarity is a rising edge input while a negative polarity is a falling edge input. All of the DAC pins share a single interrupt vector, which means the firmware needs to read the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



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.

Hub Downstream Ports Status and Control

Data transfer on hub downstream ports is controlled according to the bit settings of the Hub Downstream Ports Control Register (Table 37). Each downstream port is controlled by two bits, as defined in Table 38. The Hub Downstream Ports Control Register is cleared upon reset or bus reset, and the reset state is the state for normal USB traffic. Any downstream port being forced must be marked as disabled (Table 36 on page 34) for proper operation of the hub repeater.

Firmware uses this register for driving bus reset and resume signaling to downstream ports. Controlling the port pins through this register uses standard USB edge rate control according to the speed of the port, set in the Hub Port Speed Register.

The downstream USB ports are designed for connection of USB devices, but also serves as output ports under firmware control. This allows unused USB ports to be used for functions such as driving LEDs or providing additional input signals. Pulling up these pins to voltages above V_{REF} may cause current flow into the pin.

This register is not reset by bus reset. These bits must be cleared before going into suspend.

Table 37. Hub Downstream Ports Control Register

Hub Downstream Ports Control Register **ADDRESS 0x4B**

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

Table 38. Control Bit Definition for Downstream Ports

Control Bits		Control Action
Bit1	Bit 0	
0	0	Not Forcing (Normal USB Function)
0	1	Force Differential ‘1’ (D+ HIGH, D– LOW)
1	0	Force Differential ‘0’ (D+ LOW, D– HIGH)
1	1	Force SE0 state

An alternate means of forcing the downstream ports is through the Hub Ports Force Low Register (Table 39). With these registers the pins of the downstream ports are individually forced LOW, or left unforced. Unlike the Hub Downstream Ports Control Register, above, the Force Low Register does not produce standard USB edge rate control on the forced pins. However, this register allows downstream port pins to be held LOW in suspend. This register is used to drive SE0 on all downstream ports when unconfigured, as required in the USB 1.1 specification.

Table 39. Hub Ports Force Low Register

Hub Ports Force Low **ADDRESS 0x51**

Bit #	7	6	5	4	3	2	1	0
Bit Name	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]
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 data state of downstream ports are read through the HUB Ports SE0 Status Register (Table 40 on page 36) and the Hub Ports Data Register (Table 41 on page 36). The data read from the Hub Ports Data Register is the differential data only and is independent of the settings of the Hub Ports Speed Register (Table 35 on page 34). When the SE0 condition is sensed on a downstream port, the corresponding bits of the Hub Ports Data Register hold the last differential data state before the SE0. Hub Ports SE0 Status Register and Hub Ports Data Register are cleared upon reset or bus reset.

Table 40. Hub Ports SE0 Status Register

Hub Ports SE0 Status **ADDRESS 0x4F**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Port 4 SE0 Status	Port 3 SE0 Status	Port 2 SE0 Status	Port 1 SE0 Status
Read/Write	-	-	-	-	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit [0..3]: Port x SE0 Status (where x = 1..4)

Set to 1 if a SE0 is output on the Port x bus; Set to 0 if a Non-SE0 is output on the Port x bus.

Bit [7..4]: Reserved.

Table 41. Hub Ports Data Register

Hub Ports Data **ADDRESS 0x50**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved	Reserved	Reserved	Reserved	Port 4 Diff. Data	Port 3 Diff. Data	Port 2 Diff. Data	Port 1 Diff. Data
Read/Write	-	-	-	-	R	R	R	R
Reset	0	0	0	0	0	0	0	0

Bit [0..3]: Port x Diff Data (where x = 1..4)

Set to 1 if D+ > D– (forced differential 1, if signal is differential, i.e. not a SE0 or SE1). Set to 0 if D– > D+ (forced differential 0, if signal is differential, i.e., not a SE0 or SE1);

Bit [7..4]: Reserved.

Downstream Port Suspend and Resume

The Hub Ports Suspend Register (Table 42) and Hub Ports Resume Status Register (Table 49 on page 41) indicate the suspend and resume conditions on downstream ports. The suspend register must be set by firmware for any ports that are selectively suspended. Also, this register is only valid for ports that are selectively suspended.

If a port is marked as selectively suspended, normal USB traffic is not sent to that port. Resume traffic is also prevented from going to that port, unless the Resume comes from the selectively suspended port. If a resume condition is detected on the port, hardware reflects a Resume back to the port, sets the Resume bit in the Hub Ports Resume Register, and generates a hub interrupt. If a disconnect occurs on a port marked as selectively suspended, the suspend bit is cleared.

The Device Remote Wakeup bit (bit 7) of the Hub Ports Suspend Register controls whether or not the resume signal is propagated by the hub after a connect or a disconnect event. If the Device Remote Wakeup bit is set, the hub automatically propagates the resume signal after a connect or a disconnect event. If the Device Remote Wakeup bit is cleared, the hub does not propagate the resume signal. The setting of the Device Remote Wakeup flag has no impact on the propagation of the resume signal after a downstream remote wakeup event. The hub automatically propagates the resume signal after a remote wakeup event, regardless of the state of the Device Remote wakeup bit. The state of this bit has no impact on the generation of the hub interrupt. These registers are cleared on reset or USB bus reset.

Table 42. Hub Ports Suspend Register

Hub Ports Suspend **ADDRESS 0x4D**

Bit #	7	6	5	4	3	2	1	0
Bit Name	Device Remote Wakeup	Reserved	Reserved	Reserved	Port 4 Selective Suspend	Port 3 Selective Suspend	Port 2 Selective Suspend	Port 1 Selective Suspend
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 Selective Suspend (where x = 1..4)

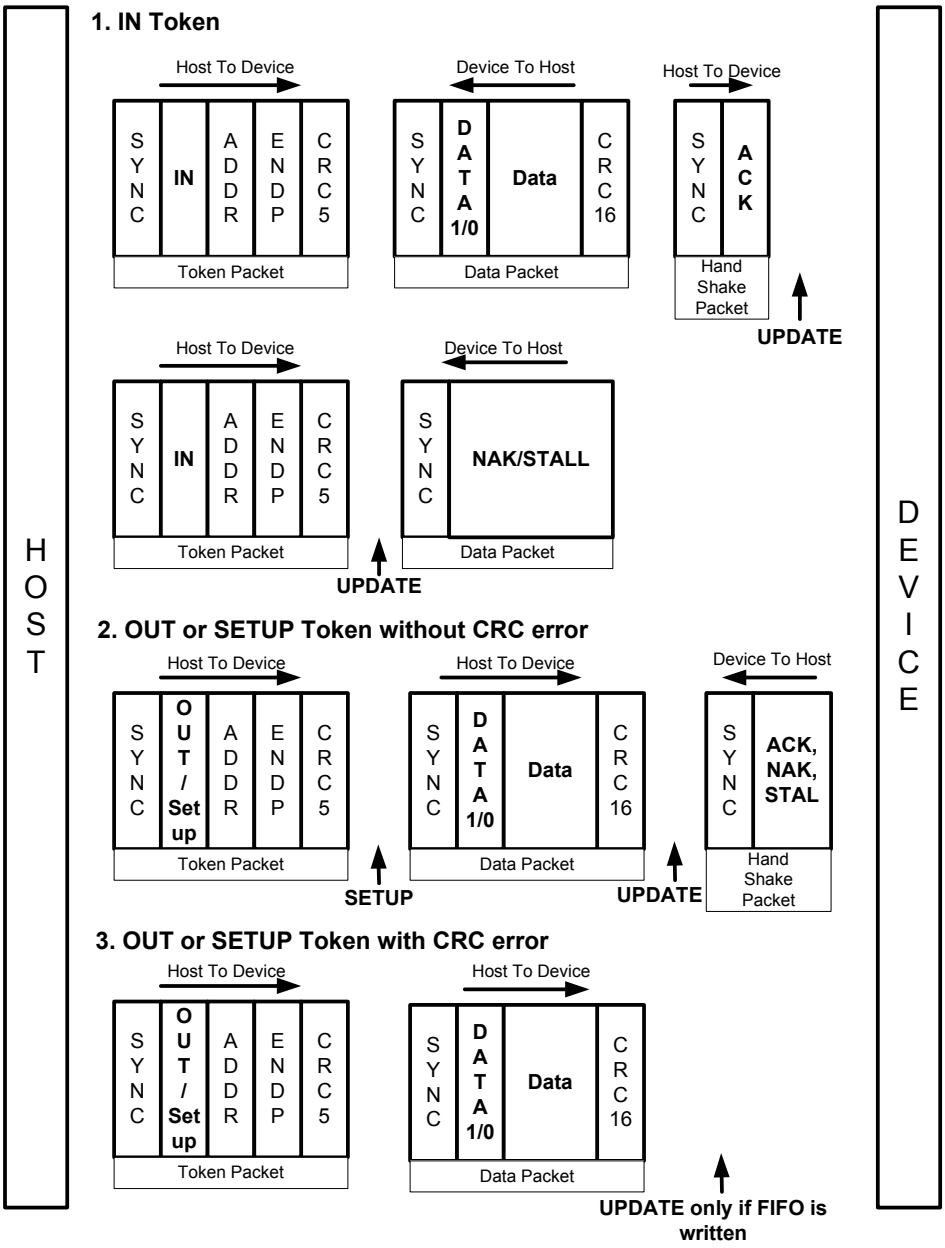
Set to 1 if Port x is Selectively Suspended; Set to 0 if Port x Do not suspend.

Bit 7: Device Remote Wakeup.

When set to 1, Enable hardware upstream resume signaling for connect and disconnect events during global resume.

When set to 0, Disable hardware upstream resume signaling for connect and disconnect events during global resume.

Figure 11. Token and Data Packet Flow Diagram



USB Mode Tables

Table 51. USB Register Mode Encoding

Mode	Mode Bits	SETUP	IN	OUT	Comments
Disable	0000	ignore	ignore	ignore	Ignore all USB traffic to this endpoint
Nak In/Out	0001	accept	NAK	NAK	Forced from Setup on Control endpoint, from modes other than 0000
Status Out Only	0010	accept	stall	check	For Control endpoints
Stall In/Out	0011	accept	stall	stall	For Control endpoints
Ignore In/Out	0100	accept	ignore	ignore	For Control endpoints
Isochronous Out	0101	ignore	ignore	always	For Isochronous endpoints
Status In Only	0110	accept	TX 0 Byte	stall	For Control Endpoints
Isochronous In	0111	ignore	TX count	ignore	For Isochronous endpoints
Nak Out	1000	ignore	ignore	NAK	Is set by SIE on an ACK from mode 1001 (Ack Out)
Ack Out(STALL[4]=0) Ack Out(STALL[4]=1)	1001 1001	ignore ignore	ignore ignore	ACK stall	On issuance of an ACK this mode is changed by SIE to 1000 (NAK Out)
Nak Out-Status In	1010	accept	TX 0 Byte	NAK	Is set by SIE on an ACK from mode 1011 (Ack Out-Status In)
Ack Out-Status In	1011	accept	TX 0 Byte	ACK	On issuance of an ACK this mode is changed by SIE to 1010 (NAK Out – Status In)
Nak In	1100	ignore	NAK	ignore	Is set by SIE on an ACK from mode 1101 (Ack In)
Ack IN(STALL[4]=0) Ack IN(STALL[4]=1)	1101 1101	ignore ignore	TX count stall	ignore ignore	On issuance of an ACK this mode is changed by SIE to 1100 (NAK In)
Nak In – Status Out	1110	accept	NAK	check	Is set by SIE on an ACK from mode 1111 (Ack In – Status Out)
Ack In – Status Out	1111	accept	TX Count	check	On issuance of an ACK this mode is changed by SIE to 1110 (NAK In – Status Out)

Mode

This lists the mnemonic given to the different modes that are set in the Endpoint Mode Register by writing to the lower nibble (bits 0..3). The bit settings for different modes are covered in the column marked “Mode Bits.” The Status IN and Status OUT represent the Status stage in the IN or OUT transfer involving the control endpoint.

Mode Bits

These column lists the encoding for different modes by setting Bits[3..0] of the Endpoint Mode register. This modes represents how the SIE responds to different tokens sent by the host to an endpoint. For instance, if the mode bits are set to “0001” (NAK IN/OUT), the SIE responds with an

- ACK on receiving a SETUP token from the host
- NAK on receiving an OUT token from the host
- NAK on receiving an IN token from the host

Refer to I2C Compatible Controller on page 23 for more information on SIE functioning.

SETUP, IN, and OUT

These columns shows the SIE’s response to the host on receiving a SETUP, IN, and OUT token depending on the mode set in the Endpoint Mode Register.

A “Check” on the OUT token column, implies that on receiving an OUT token the SIE checks to see whether the OUT packet is of zero length and has a Data Toggle (DTOG) set to ‘1.’ If the DTOG bit is set and the received OUT Packet has zero length, the OUT is ACKed to complete the transaction. If either of this condition is not met the SIE responds with a STALLL or just ignore the transaction.

A “TX Count” entry in the IN column implies that the SIE transmit the number of bytes specified in the Byte Count (bits 3..0 of the Endpoint Count Register) to the host in response to the IN token received.

A “TX0 Byte” entry in the IN column implies that the SIE transmit a zero length byte packet in response to the IN token received from the host.

An “Ignore” in any of the columns means that the device does not send any handshake tokens (no ACK) to the host.

An “Accept” in any of the columns means that the device responds with an ACK to a valid SETUP transaction to the host.

Note
4. STALL bit is bit 7 of the USB Non Control Device Endpoint Mode registers. For more information, refer to USB Non Control Endpoint Mode Registers on page 41.

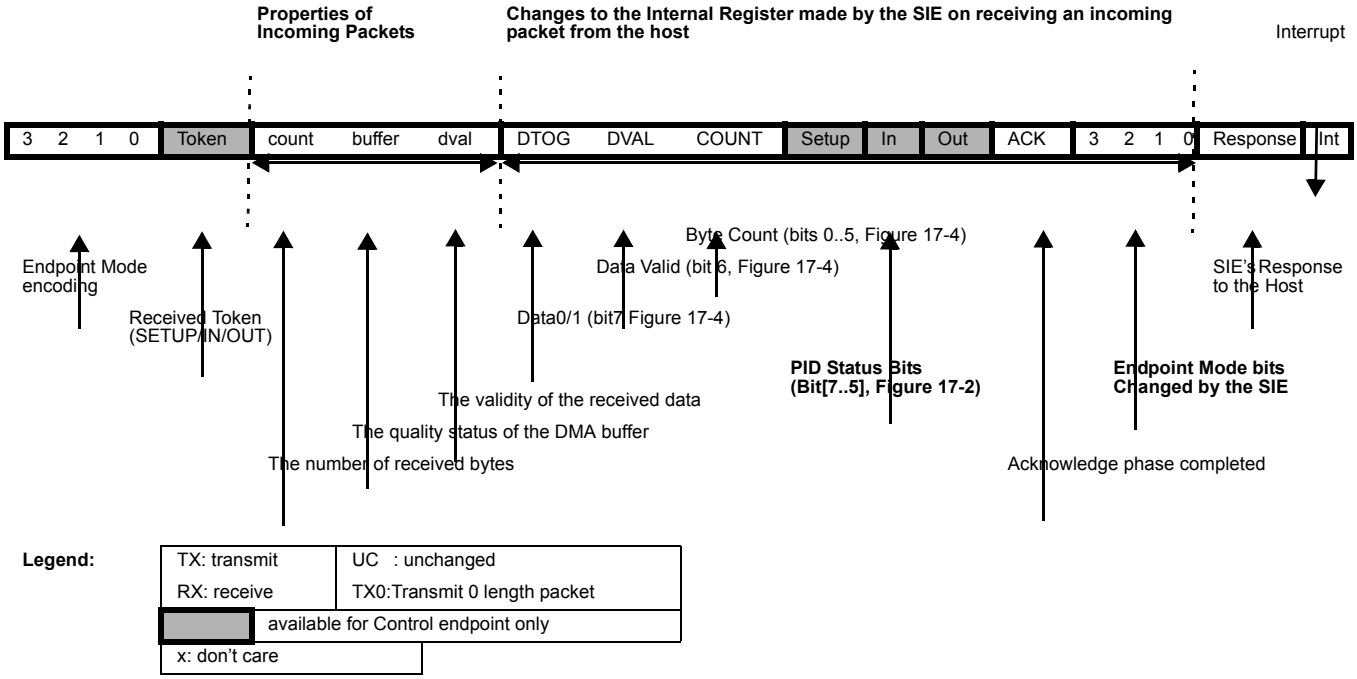
Comments

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



The response of the SIE are summarized as follows:

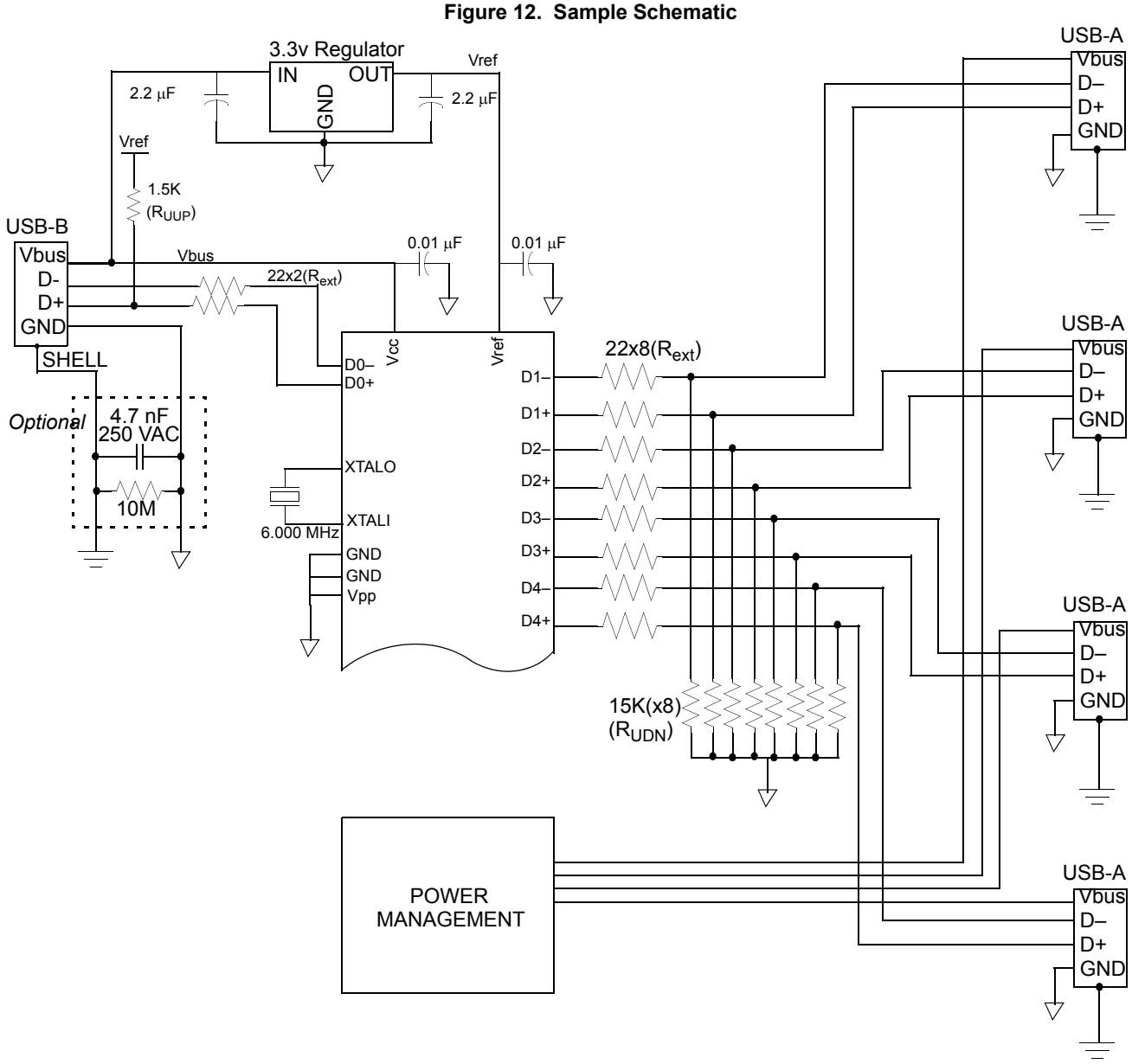
- The SIE only responds to valid transactions, and ignores invalid ones.
- The SIE generates an interrupt when a valid transaction is completed or when the FIFO is corrupted. FIFO corruption occurs during an OUT or SETUP transaction to a valid internal address, that ends with a invalid CRC.
- An incoming Data packet is valid if the count is ≤ 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			

Sample Schematic

Figure 12. Sample Schematic



Package Diagrams

Figure 17. 48-pin SSOP (300 Mils) Package Outline

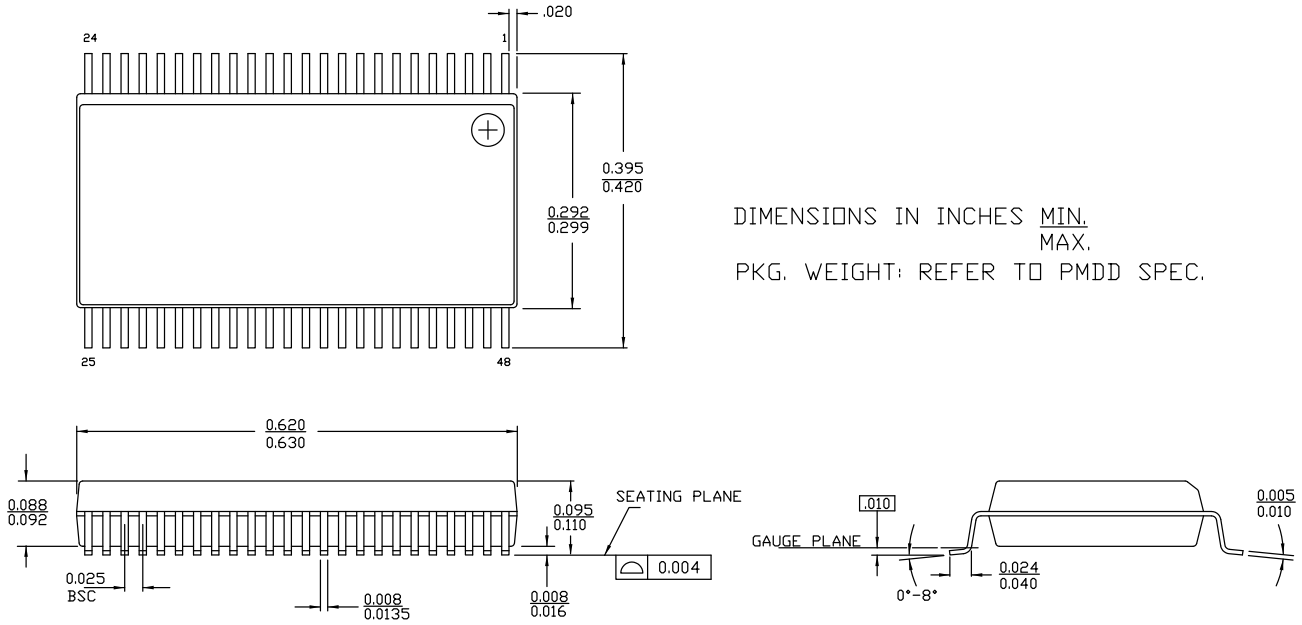
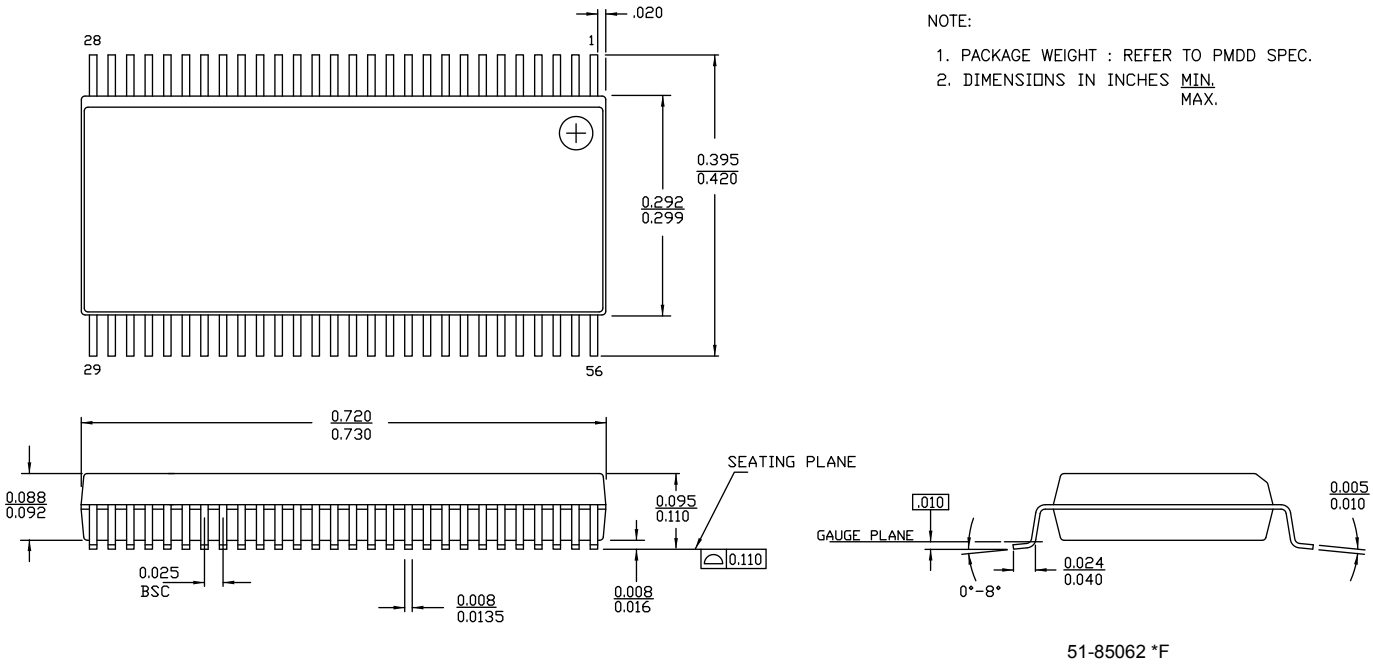


Figure 18. 56-pin SSOP (300 Mils) Package Outline



Document History Page

Document Title: CY7C66013C/CY7C66113C, Full-Speed USB (12 Mbps) Peripheral Controller with Integrated Hub Document Number: 38-08024				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**	114525	DSG	3/27/02	Change from Spec number: 38-00591 to 38-08024
*A	124768	MON	03/20/03	Added register bit definitions; Added default bit state of each register. Corrected the Schematic (location of the pull-up on D+). Added register summary. Removed information on the availability of the part in PDIP package. Modified Table 51 and provided more explanation regarding locking/unlocking mechanism of the mode register. Removed any information regarding the speed detect bit in Hub Port Speed register being set by hardware.
*B	417632	BHA	See ECN	Updated part number and ordering information. Added QFN Package Drawing and Design Notes. Corrected bit names in Figures 9-3, 9-4, 9-5, 9-8, 9-9, 9-10, 10-5, 16-1, 18-1, 18-2, 18-3, 18-6, 18-7, 18-9, 18-10. Removed Hub Ports Force Low register address 0x52. Added HAPI to Interrupt Vector Number 11 in Table 16-1. Corrected bit names in Section 21.0. Corrected Units in Table 24.0 for R_{UUP}, R_{UDN}, R_{EXT}, and Z_O. Added DIE diagram and related information. Added HAPI to GPIO interrupt vector in Table 5-1 and figure 16-3
*C	1825466	TLY / PYRS	See ECN	Changed Title from “CY7C66013, CY7C66113 Full Speed USB (12 Mbps) Peripheral Controller with Integrated Hub” to “CY7C66013C, CY7C66113C Full Speed USB (12 Mbps) Peripheral Controller with Integrated Hub”. Changed package description for CY7C66013C and CY7C66113C from -PVC to -PVXC
*D	2720540	DPT / AESA	06/18/09	Added 56 QFN 8x8x1 mm package diagram and ordering information
*E	2896318	AESA	03/18/10	Removed Part CY7C66113C-LFXC. Updated all package diagrams.
*F	3057657	AJHA	10/13/10	Added “Not recommended for new designs” watermark in the PDF. No technical or content updates.
*G	3177081	NXZ	02/18/2011	Added Ordering Code Definitions under Ordering Information. Added Acronyms and Units of Measure. Updated to new template.
*H	4313900	AKSL	03/21/2014	Removed “Not recommended for new designs” watermark. Updated Package Diagrams.
*I	5693560	HBM	04/12/2017	Updated to new template. Completing Sunset Review.

Sales, Solutions, and Legal Information

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support