

January 29, 2008

Errata Document for CY7C67300, EZ-Host™ Programmable Embedded USB Host/Peripheral Controller

This document describes the errata for the CY7C67300. Details include errata trigger conditions, available workarounds, and silicon revision applicability. Compare this document to the device data sheet to fully understand device functionality.

Please contact your local Cypress Sales Representative if you have further questions.

Part Numbers Affected

Part Number	Device Characteristics
CY7C67300	All Packages

CY7C67300 Qualification Status

In Production

CY7C67300 Errata Summary

The following table defines the errata applicability to available CY7C67300 family devices. An 'X' indicates that the errata pertains to the selected device.

Note Errata titles are hyperlinked. Click on a table entry to jump to the description.

Items	CY7C67300	Rev. Letter	Fix Status
1. HPI write to SIE registers	X	A	No silicon revision planned, use workaround.
2. Hub and low-speed device attached to a root hub of the EZ-Host chip	X	A	No silicon revision planned, use workaround.
3. IDE register read when GPIO24 pin is low	X	A	No silicon revision planned, use workaround.
4. UART does not recognize framing errors	X	A	No silicon revision planned, use workaround.
5. UART does not override GPIO Control Register	X	A	No silicon revision planned, use workaround.
6. VBUS Interrupt (VBUS Valid) Requires De-bouncing	X	A	No silicon revision planned, use workaround.
7. Coupled SIE Interrupt Enable Bits	X	A	No silicon revision planned, use workaround.
8. Un-Initialized SIExmsg Registers	X	A	No silicon revision planned, use workaround.
9. BIOS USB Peripheral Mode: Descriptor Length	X	A	No silicon revision planned, use workaround.
10. Peripheral short packet issue	X	A	No silicon revision planned, use workaround.
11. Data toggle corruption issue	X	A	No silicon revision planned, use workaround.

Items	CY7C67300	Rev. Letter	Fix Status
12. Code fails to load from EEPROM	X	A	No silicon revision planned, use workaround.
13. ISOHCH Endpoint Descriptor error	X	A	No silicon revision planned, use workaround.
14. Missing Endpoint Interrupts in USB Peripheral Mode	X	A	No silicon revision planned, use workaround.

1. HPI write to SIE registers

• PROBLEM DEFINITION

Writing to the SIE2 Control register via HPI can corrupt the SIE1 Control register.

Writing to the SIE1 Control register via HPI can corrupt the SIE2 Control register.

• PARAMETERS AFFECTED

SIE Control registers.

• TRIGGER CONDITION(S)

When an external processor accesses the SIE1 or SIE2 register at the same time the internal CY16 CPU is also accessing the opposite SIE, the SIE accessed by the CY16 CPU will be corrupted. For example, the external processor writes a value of 0x80 to the SIE2 register 0xC0B0 while the internal CY16 is doing a read/write to the SIE1 register 0xC08C, the SIE1 register 0xC08C, will be corrupted with the value 0x80.

• SCOPE OF IMPACT

If the internal CPU and the external CPU access the SIEs at the same time, contention will occur resulting in incorrect data in one of the SIE registers.

• WORKAROUND

1. Use the LCP COMM_WRITE_CTRL_REG to handle the writing to SIE registers.
2. Use download code to handle SIE WRITE commands.
3. Avoid accessing SIE register from the external CPU. For example, route all the SIE interrupts to the software mailbox interrupt registers 0x144 and 0x148. This requires user to create download code.

• FIX STATUS

No silicon revision planned, use workaround. An implementation example is included in the Cypress Windows CE driver.

2. Hub and low-speed device attached to a root hub of the EZ-Host chip

• PROBLEM DEFINITION

When a hub and a low-speed device are connected to the same SIE, the hub does not pass the USB packets from the downstream to the upstream.

• PARAMETERS AFFECTED

SOF timing.

• TRIGGER CONDITIONS

Connecting a hub and a low-speed device to the same root hub (SIE) of the EZ-Host chip.

• SCOPE OF IMPACT

The EZ-Host does not generate an accurate 1 ms SOF time frame to the hub.

• WORKAROUND

1. Code can detect the condition and report a message to the user.
2. The low-speed device could be attached to one of the hub downstream ports.

• FIX STATUS

No silicon revision planned, use workaround.

3. IDE register read when GPIO24 pin is low**• PROBLEM DEFINITION**

The part does not service USB ISRs when the GPIO24 pin (also labeled as HPI_INT and IORDY) is low and any IDE register is read.

• PARAMETERS AFFECTED

USB ISRs do not get serviced.

• TRIGGER CONDITIONS

The IDE registers (0xC050 through 0xC06E) should not be read unless IDE is being used. Debuggers that read all memory locations while single stepping can cause this situation to manifest itself.

• SCOPE OF IMPACT

If you are debugging and using this pin, your application will appear to hang.

• WORKAROUND

When running in standalone mode, avoid using the GPIO24 pin if possible.

• FIX STATUS

No silicon revision planned, use workaround.

4. UART Does Not Recognize Framing Errors**• PROBLEM DEFINITION**

The UART is not designed to recognize framing errors.

• PARAMETERS AFFECTED

UART serial communications.

• TRIGGER CONDITIONS

Some platforms can cause EZ-Host to see a string of NULL characters and cause the UART to get out of sync.

• SCOPE OF IMPACT

This can cause the UART to lose connection with the host during serial communications. One example of this is if the UART is used as the debug port to the PC. This problem has occurred on, but is not limited to, Dell™ machines running Windows® XP or Windows® 2000.

• WORKAROUND

For general use, there is no workaround. If this problem is experienced while debugging, try running the debugger on a different host (PC/OS). Otherwise the USB port can be used for the debugging interface.

• FIX STATUS

No silicon revision planned, use workaround.

5. UART does not override GPIO Control Register**• PROBLEM DEFINITION**

When the UART is enabled, the GPIO Control Register still has control over GPIO 27 (UART RX pin). When enabled, the UART should override the GPIO Control Register, which defaults to setting the pin as an input.

• PARAMETERS AFFECTED

UART serial communications.

• TRIGGER CONDITIONS

Enabling UART.

• SCOPE OF IMPACT

GPIO 27 UART RX pin is controlled by GPIO Control Register and defaults to an input. The UART mode does not override the GPIO Control Register for this pin and can be inadvertently configured as an output.

• WORKAROUND

Ensure the GPIO Control Register is written appropriately to set GPIO27 as an input when the UART is enabled.

• FIX STATUS

No silicon revision planned, use workaround.

6. VBUS Interrupt (VBUS Valid) Requires Debouncing**• PROBLEM DEFINITION**

The VBUS interrupt in the Host/Device Status Registers [0xC090 and 0xC0B0] and OTG Control Register [0xC098] triggers multiple times whenever VBUS is turned on. It should only trigger once when VBUS rises above 4.4V and once when VBUS falls from above 4.4V to 0V.

• PARAMETERS AFFECTED

Electrical.

• TRIGGER CONDITIONS

VBUS turned on.

• SCOPE OF IMPACT

Host/Device Registers and OTG Control Register trigger multiple times.

• WORKAROUND

When reading the status of this interrupt, a software debounce should be implemented.

• FIX STATUS

No silicon revision planned, use workaround. Examples are provided in the Development Kit Software.

7. Coupled SIE Interrupt Enable Bits**• PROBLEM DEFINITION**

Host/Device 1 SIE events will still trigger an interrupt when only the Host/Device 2 SIE Interrupt Enable is set and vice versa.

• PARAMETERS AFFECTED

Host/Device SIE Interrupts.

• TRIGGER CONDITIONS

Setting only 1 Host/Device SIE Interrupt Enable.

• SCOPE OF IMPACT

The Host/Device global Interrupt Enable bits cannot be used to disable each Host/Device SIE independently. These bits are found in the Interrupt Enable Register (0xC00E).

• WORKAROUND

If an SIE Interrupt is desired, both Host/Device 1 and Host/Device 2 Interrupt Enable bits should be set in the Global Interrupt Enable Register (0xC00E). To properly mask an SIE Interrupt to a single SIE, the lower level Host/Device Interrupt Enable Registers (0xC08C and 0xC0AC) must be used. For example, setting the Host/Device 2 IE Register to 0x0000 will prevent any Host/Device 2 events from generating a Host/Device Interrupt. To disable all SIE interrupts, both Host/Device Interrupt Enable bits in the Interrupt Enable Register should be cleared.

• FIX STATUS

No silicon revision planned, use workaround.

8. Un-Initialized SIExmsg Registers**• PROBLEM DEFINITION**

The SIE1msg and SIE2msg Registers [0x0144 and 0x0148] are not initialized at power up.

• PARAMETERS AFFECTED

HPI interrupts.

• TRIGGER CONDITIONS

Power up initialization.

• SCOPE OF IMPACT

If you are using the HPI interface in co-processor mode, random data will be written to the SIE1msg and SIE2msg Registers [0x0144 and 0x0148] at power up. This will cause two improper HPI interrupts (HPI_INTR) to occur, one for each of the two SIExmsg Registers.

• WORKAROUND

The external processor should clear the SIExmsg Registers [0x0144 and 0x0148] shortly after nRESET is de-asserted and prior to the expected processing of proper HPI interrupts (generally 10 ms after nRESET is de-asserted).

• FIX STATUS

No silicon revision planned, use workaround.

9. BIOS USB Peripheral Mode: Descriptor Length

- **PROBLEM DEFINITION**

The BIOS will not properly return a descriptor or set of descriptors, if the length is a multiple of the control endpoint's maximum packet size.

- **PARAMETERS AFFECTED**

Control Endpoint maximum packet size.

- **TRIGGER CONDITIONS**

Get Descriptor requests.

- **SCOPE OF IMPACT**

If the descriptor length is a multiple of the maximum packet size, the BIOS will respond with a STALL instead of a zero-length data packet for the final IN request.

- **WORKAROUND**

If the requested descriptor length is a multiple of the maximum packet size, then either the maximum packet size or the descriptor length needs to change. A descriptor length can be increased by simply adding a padded byte to the end of a descriptor and increasing the descriptor Length byte by one. Section 9.5 (Descriptor) of the USB 2.0 specification allows a descriptor length to be larger than the value defined in the specification.

- **FIX STATUS**

No silicon revision planned, use workaround.

10. Peripheral short packet issue

- **PROBLEM DEFINITION**

When an SIE is configured as a peripheral, the SUSBx_RECEIVE function does not invoke the callback function when it receives a short packet.

- **PARAMETERS AFFECTED**

SIEx Endpoint x Interrupt (Interrupt 32-47).

- **TRIGGER CONDITIONS**

This issue is seen when an SIE is configured as a peripheral during an OUT data transfer when the host sends a zero length or short packet. If this occurs, the BIOS will behave as if a full packet was received and will continue to accept data until the Device n Endpoint n Count Register value is satisfied.

- **SCOPE OF IMPACT**

All peripheral functions are susceptible to this, as it is a normal occurrence with USB traffic.

- **WORKAROUND**

To fix this problem, the SIEx Endpoint x Interrupt must be replaced for any peripheral endpoint that is configured as an OUT endpoint.

1. Acquire the file called *susb1.s* from Cypress Support or download a newer version of the frameworks that has this fix applied and includes *susb1.s*.
2. Modify *fwxcfg.h* in your project to have the following flags and define/undef the fix for the endpoints you are using:

```
#define FIX_USB1_EP1
#define FIX_USB1_EP2
#undef  FIX_USB1_EP3
#undef  FIX_USB1_EP4
#undef  FIX_USB1_EP5
#undef  FIX_USB1_EP6
#undef  FIX_USB1_EP7

#undef  FIX_USB2_EP1
#undef  FIX_USB2_EP2
#undef  FIX_USB2_EP3
#undef  FIX_USB2_EP4
#undef  FIX_USB2_EP5
#undef  FIX_USB2_EP6
#undef  FIX_USB2_EP7
```

3. Add the new *susb1.s* to the included assembly source files in the make file.

For example : `ASM_SRC := startup.s isrs.s susb1.s`

4. Add `usb_init` somewhere in the startup code. This will likely be in *fwxmain.c* as demonstrated below:

```
void fwx_program_init(void)
{
void usb_init();/* define the prototype */
usb_init();
fwx_init();/* Initialize everything in the base framework. */
}
```

5. Build the project using the modified make file.

- **FIX STATUS**

No silicon revision planned, use workaround.

11. Data toggle corruption issue

- **PROBLEM DEFINITION**

When an SIE is configured as a peripheral, data toggle corruption as specified in the USB 2.0 specification, section 8.6.4, does not work as specified.

- **PARAMETERS AFFECTED**

SIEx Endpoint x Interrupt (Interrupt 32-47).

- **TRIGGER CONDITIONS**

This issue is seen when an SIE is configured as a peripheral and the host sends an incorrect data toggle. According to the USB specification, when an incorrect data toggle is seen from the host, the peripheral should throw away the data but increment the data toggle bit to re-synchronize the data toggle bits. In the current ROM BIOS, the SIEx Endpoint x Interrupt will ignore the data toggle error and accept the data.

- **SCOPE OF IMPACT**

All peripheral functions are susceptible to this as it is a normal occurrence with USB traffic.

- **WORKAROUND**

To fix this problem, the SIEx Endpoint x Interrupt must be replaced for any endpoint that is configured as an OUT endpoint.

1. Acquire the file called *susb1.s* from Cypress Support or download a newer version of the frameworks that has this included.
2. Modify *fwxcfg.h* in your project to have the following flags and define/undef the fix for the endpoints you are using:

```
#define FIX_USB1_EP1
#define FIX_USB1_EP2
#undef  FIX_USB1_EP3
#undef  FIX_USB1_EP4
#undef  FIX_USB1_EP5
#undef  FIX_USB1_EP6
#undef  FIX_USB1_EP7

#undef  FIX_USB2_EP1
#undef  FIX_USB2_EP2
#undef  FIX_USB2_EP3
#undef  FIX_USB2_EP4
#undef  FIX_USB2_EP5
#undef  FIX_USB2_EP6
#undef  FIX_USB2_EP7
```

3. Add the new *susb1.s* to the included assembly source files in the make file.
For example : `ASM_SRC := startup.s isrs.s susb1.s`

4. Add `usb_init` somewhere in the startup code. This will likely be in *fwxmain.c* as demonstrated below:

```
void fwx_program_init(void)
{
    void usb_init(); /* define the prototype */

    usb_init();

    fwx_init();      /* Initialize everything in the base framework. */
}
```

5. Build the project using the modified make file.

- **FIX STATUS**

No silicon revision planned, use workaround.

12. Code fails to load from EEPROM

- **PROBLEM DEFINITION**

If, while the BIOS is loading firmware, the part is reset and at that time the EEPROM is driving the SDA line low, the BIOS will configure the part for co-processor mode instead of standalone mode.

- **PARAMETERS AFFECTED**

Initialization.

- **TRIGGER CONDITIONS**

Reset the part while firmware is being loaded from the EEPROM.

- **SCOPE OF IMPACT**

The firmware download process will not finish, leaving the part configured in co-processor mode.

- **WORKAROUND**

There is no workaround. Cycle power to the EEPROM to download firmware.

- **FIX STATUS**

No silicon revision planned, use workaround.

13. ISOCH Endpoint Descriptor error

- **PROBLEM DEFINITION**

Setting any bit other than 1:0 in the `bmAttributes` field will cause the ISOCH Endpoint Descriptors to be reported incorrectly.

- **PARAMETERS AFFECTED**

Isochronous transfers.

- **TRIGGER CONDITIONS**

Setting any bit other than 1:0 in the `bmAttributes` field.

- **SCOPE OF IMPACT**

If the `bmAttributes` field in the Endpoint Descriptor is using bits 5...2, the BIOS will not correctly parse the endpoint and set up the part correctly for ISO transfers.

- **WORKAROUND**

There are two methods that can be used.

- A. Mask these bits before the BIOS parses the descriptors using the `SET_INTERFACE` handler. An example of this is given in the "Using Multiple Interfaces to Implement a USB Isochronous Composite Peripheral with EZ-Host™ and EZ-OTG™."

- B. Replace the BIOS delta config interrupt and modify the USB-parse code to mask off all but the lower two bits of the bmAttribute. A possible solution might look like this.

```

@@test b[r8+bEPAttribute], 0x01 ; check ISO
rz
test b[r8+bEPAttribute], 0x02
rnz
;if we get here, then the lower two bits
;of bEPAttribute = 01 meaning it is ISO or r2,EP_ISO
ret

```

- **FIX STATUS**

No silicon revision planned, use workaround.

14. Missing Endpoint Interrupts in USB Peripheral Mode

- **PROBLEM DEFINITION**

USB peripheral designs may miss endpoint interrupts when receiving Endpoint 0 (EP0) Control transfer requests mixed with other endpoint transfer type transactions.

- **PARAMETERS AFFECTED**

All USB peripheral mode endpoint communication may potentially be affected.

- **TRIGGER CONDITIONS**

An endpoint interrupt may be missed when a non-EP0 transaction completes (with ACK) during an EP0 Control transfer or within ~200 μ s before or after the EP0 Control transfer. Other processor activity and interrupts may influence the likelihood of this failure occurring.

- **SCOPE OF IMPACT**

This errata item applies to USB peripheral mode only. All USB peripheral designs that mix Standard, Vendor, or Class EP0 requests with other transfer types are potential candidates for this issue. If this problem occurs and an endpoint interrupt is missed, the endpoint will likely not be rearmed and therefore endpoint communication will be halted and the host system may reset the device.

- **WORKAROUND**

- The PC Application and/or driver must be developed to ensure at least ~200 μ s of idle time is given before and after any EP0 transfers. The driver must ensure that no other transfer type transactions occur during the EP0 transfer or during this idle time before and after the EP0 transfer. Therefore, the driver cannot submit asynchronous Control transfers. The driver must submit Control transfer requests synchronously with other transfer requests. In addition, the driver must be aware of any interrupt endpoint scheduling (which is under control of the host controller driver) when submitting Control transfers. This generally means that a vendor-specific driver is required.
- USB endpoint communication stress testing of any USB peripheral designs that mix EP0 Control transfers with other transfer types is recommended.

- **FIX STATUS**

No silicon revision planned, use workaround.

References

- Document # 38-08015, CY7C67300 EZ-Host™ Programmable Embedded USB Host/Peripheral Controller

Document History Page

Document Title: Errata Document for CY7C67300, EZ-Host™ Programmable Embedded USB Host/Peripheral Controller Document Number: 38-17022				
REV.	ECN NO.	Issue Date	Orig. of Change	Description of Change
**	227956	See ECN	kku	Initial release
*A	237769	See ECN	ari	Adding items 10 and 11
*B	522767	See ECN	bha	Added items 12 and 13
*C	2011786	See ECN	hmt	Added item 14