Errata: USB: The Universal Serial Bus
1st Edition
Dated: 5 Aug 2015

Intro: Changed a few instants of “throughout” to “through out”.
Changed “hexadecimal number” to “hexadecimal numbers”.

Page i-2: Last paragraph: *Hear* should be *Here*.

Page i-3: Last paragraph: The “a” in *assembly* should be capitalized.

Page i-8: Must add the \USB sub-directory to the destination
C:\FYSOS\xcopy D:\FYSOS\MAIN\USB C:\FYSOS\MAIN /E /I /-Y /D<*enter*>
Should be:
C:\xcopy D:\FYSOS\MAIN\USB C:\FYSOS\MAIN\USB /E /I /-Y /D<*enter*>

Page i-9: Heading: Prerequisite to using this book.
The Specification files are not included on the disc.
Appendix A defines where to get these files.

Listing 4-2: Last line:
Change
~QUEUE_HEAD_PTR_MASK
to
QUEUE_HEAD_PTR_MASK
(remove the ‘~’ character)

Page 5-13:
“its power by setting bit 2 in its respective register”
should be
“its power by setting bit 8 in its respective register”

Page 5-14: Paragraph starting with:
“The Port Power Control Mask field is...”
As with the fix on Page 5-13 above, it is bit 8, not bit 2...

Page 6-8: Last sentence in the “Transfer Descriptors” heading.
“See Appendix B for information on ISO transfers.”
Should be Appendix D.

Chapter 8 Heading: Should be “EHCI Stack”, not “UHCI Stack”.

Page 8-14: Last Paragraph:
Ignore last paragraph of this page, it is a remnant of an old driver.
This paragraph, and the first one on the next page, when combined,
should read:
“As stated earlier, one of the dwords in the scratch area of an FYSOS Queue Head buffer, is used for inserting queues into a linked list. We use these 4 bytes after each queue to allow insertion and removal of an unlimited amount of queues in each main queue in the schedule. Each of

these 8 queues is called Queue Heads and uses a linked list to add or remove other queues. To insert a queue, simply find the last queue in the desired queue head and point its queue head pointer to the address of your queue, and point the previous pointer member of your new queue to the queue head pointer of the last queue found."

Page 8-10: Table 8-5 should have cell separators.

Page 9-6: Top of page. Context Size should be 64-byte or 32-byte blocks, not 64-bit or 32-bit blocks.

Page 9-7: Description under Table 9-10:
"The Doorbell Array Offset field..."
should be
"The Runtime Register Space Offset field..."

Page 9-21: Table 9-25:
The first column in table 9-25, the 0x0400 and 0x0440 should be 0x0420 and 0x0460 respectively. Then the 192 byte reserved size should now be 160 bytes in size.

Page 9-40:
"...Device Notification Control ... to have bit 2 set."
"xhci_write_oper_reg(base, xHC_OPS_USBDnctrl, (1<<2));"
should be:
"...Device Notification Control ... to have bit 1 set."
"xhci_write_oper_reg(base, xHC_OPS_USBDnctrl, (1<<1));"

Page 11-6:
Clear the toggle bit in the SETUP packet, then toggle it for each packet there after within this transfer, making sure the STATUS packet has it set. The description (two places) on this page state just the opposite, which is wrong. (I don't know how I let that make it to production... :-(

Figure 13-1: TD2 0x12340E0:
Third dword in that TD should be 0x80000C80.

Figure 13-4: TD1 0x12340A0:
Third dword in that TD should be 0x80000C80.

Page 14-11: Table 14-5:
"Physical Address of Set Address Command TRB"
should be
"Physical Address of No OP Command TRB"

Page 14-24: Table 14-18:
"Physical Address of Set Address Command TRB"
should be
"Physical Address of Configure EP Command TRB"

Page 14-28: Figure 14-19:
"num_entries = 4"
should be
"num_entries = 5"

Page 17-1: "?? Note:"
'256gig' should be '256 Meg'. Sorry. However, a few years after the publication, you can now find 128gig drives, but the most common ones, at the moment, are 8-, 16-, 32-, and 64-gig drives.

Page 17-1: Footnote: The link in the foot note is now broken.
Try this one: http://www.kingston.com/datasheets/kusbdti_us.pdf

Page 17-26: Should have a few spaces above "Request Sense".

Page 19-5: Note about the self-powered bit:
Remove: "This bit is set per the USB 2.0 specification."

Page 21-1: (twice)
"...just as if it were a high-speed USB 2.0 hub."
should be:
"...just as if it were a full-speed USB 2.0 hub."

Page 21-6:
"The first entry type is the USB 2.0 Extension entry..."
should be
"The first entry type is the Wireless USB entry..."

Appendix D:
"14 times a minute" should be "14 times a second".

Added Appendix Q

Page BIB-2: Mistakenly credited "James Harris" twice, even though I am sure he may deserve it. :-)

Changed 'vender' to 'vendor' in multiple chapters.

CDROM:
I have a small update to the gd_ehci.exe source files. Send me an email asking for the new files using the email address you originally used to request the CD-ROM image or if using a new email address, include the address you originally used within the body of the new email. This way, I will verify the address and no proof of purchase will be needed.

GD_OHCI: OHCI: Get Descriptor Utility:
I thought for sure I had checked this utility before I released the CDROM, but apparently not, since this utility doesn't work at all. I have now fixed this utility. If you have the CDROM and would like the fixed version, please let me know. Contact information is in Appendix X.

Please change the read_pci() and write_pci() functions in pci.h to the following:

```
// read from the pci config space
bit32u read_pci(const bit8u bus, const bit8u dev, const bit8u func,
                const bit8u port, const bit8u len) {
  bit32u ret = 0xFFFFFFFF;

  const bit32u val = 0x80000000 |
    (bus << 16) |
    (dev << 11) |
    (func << 8) |
    (port & 0xFC);
  outpd(PCI_ADDR, val);

  switch (len) {
    case 1:
      ret = inp(PCI_DATA + (port & 0x3));
      break;
    case 2:
      if ((port & 0x3) < 3)
        ret = inpw(PCI_DATA + (port & 0x3));
      break;
    case 4:
      if ((port & 0x3) == 0)
        ret = inpd(PCI_DATA);
      break;
  }

  return ret;
}
```

```
// write to the pci config space
void write_pci(const bit8u bus, const bit8u dev, const bit8u func,
               const bit8u port, const bit8u len, const bit32u value) {

  bit32u val = 0x80000000 |
    (bus << 16) |
    (dev << 11) |
    (func << 8) |
    (port & 0xFC);
  outpd(PCI_ADDR, val);

  switch (len) {
    case 1:
      outp(PCI_DATA + (port & 0x3), value & 0xFF);
      break;
    case 2:
      if ((port & 0x3) < 3)
        outpw(PCI_DATA + (port & 0x3), value & 0xFFFF);
      break;
    case 4:
      if ((port & 0x3) == 0)
        outpd(PCI_DATA, value);
      break;
  }
}
```