Errata: USB: The Universal Serial Bus
2nd Edition
Dated: 24 November 2017

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

Page 6-2: Bottom of page: should point to Figure 6-1, not 6-2.

Page 7-8: Middle of page: "effected" to "affected".

Page 7-16: The Port Enable bit can be disabled by writing a 0 to this bit. However, it cannot be enabled by writing a 1 to this bit.

Page 9-10: Legacy Support Ext. Caps List entry:
The text states that if this entry exists, it **must** be the first entry. I remember reading a specification somewhere that stated that it would be the first entry. Also, the xHCI specification states that it will be at **Extended Capabilities Address + 0x00**. However, other comments and documents tend to disagree, therefore, don't assume it is the first entry.

Page 9-34: Should be Figure 9-3 in both instances.

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 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 states just the opposite, which is wrong. (I don't know how I let that make it to production... :-(

Figure 11-1: TD0 0x1234010:
Third dword in that TD should be 0x00E0002D.
Figure 11-1: TD1 0x1234030:
Third dword in that TD should be 0x00E80069.
Figure 11-1: TD2 0x1234050:
Third dword in that TD should be 0xFFE800E1.
Figure 11-2: TD0 0x1234010:
Third dword in that TD should be 0x00E0002D.

Figure 11-2: TD1 0x1234030:
  Third dword in that TD should be 0xFFE80069.
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 12-16: Note Box:
  You need to clear the bufferRounding bit, not set it, for the
  controller to stop execution of the ED.

Starting with Page 22-8, the table listed as 22-9 should be 22-11,
  with each table and reference to that table there after, incremented
  by 2. I added a couple of tables and forgot to update the rest...
  However, I got the Appendix C correct... :-)

Appendix G: Table G-3:
  Two pointers to Table G-3 and G-4, should be Tables G-4 and G-5.

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;

  const bit32u val = 0x80000000 |
    (bus << 16) |
    (dev << 11) |
    (func << 8) |
    (port & 0xFC);
  outpd(PCI_ADDR, val);
  ret = (inpd(PCI_DATA) >>
                     ((port & 3) * 8)) & (0xFFFFFFFF >> ((4-len) * 8));
  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, bit32u value) {
  bit32u val = 0x80000000 |
    (bus << 16) |
    (dev << 11) |
    (func << 8) |
    (port & 0xFC);
```

```
  outpd(PCI_ADDR, val);

  // get current value
  val = inpd(PCI_DATA);
  // make sure value is of 'len' size
  value &= (0xFFFFFFFF >> ((4-len) * 8));

  // mask out new section
  if (len < 4) {
    val &= (0xFFFFFFFF << (len * 8));
    val |= value;
  } else
    val = value;

  outpd(PCI_DATA, val);
}
```

And Listing 2-1 on page 2-3 should be:

Listing 2-1: Read value from the PCI

```
bit32u read_pci(const bit8u bus,
                const bit8u dev,
                const bit8u func,
                const bit8u port,
                const bit8u len) {
  bit32u ret;

  const bit32u val = 0x80000000 |
    (bus << 16) |
    (dev << 11) |
    (func << 8) |
    (port & 0xFC);
  outpd(PCI_ADDR, val);
  ret = (inpd(PCI_DATA) >>
              ((port & 3) * 8)) & (0xFFFFFFFF >> ((4-len) * 8));
  return ret;
}
```

The USB_IF has moved some of their files. In Appendix A, the following URLs have been updated:

Rev 1.1 MSD Control/Block/Interrupt (usb_msc_cbi_1.1.pdf)
http://www.usb.org/developers/docs/devclass_docs/usb_msc_cbi_1.1.pdf

Rev 1.0 of MSD UFI (Floppy) (usbmass-ufi10.pdf)
http://www.usb.org/developers/docs/devclass_docs/usbmass-ufi10.pdf

Rev 1.0 of MSD Bulk transport (usbmassbulk_10.pdf)
http://www.usb.org/developers/docs/devclass_docs/usbmassbulk_10.pdf

Device Class Documentation
http://www.usb.org/developers/docs/devclass_docs/

Rev 1.1 of the USB Printing Devices (usbprint11.pdf)
http://www.usb.org/developers/docs/devclass_docs/usbprint11a021811.pdf
Rev 1.4b Mass Storage Class (usb_msc_overview_1.4b.pdf)
www.usb.org/developers/docs/devclass_docs/
Mass_Storage_Specification_Overview_v1.4_2-19-2010.pdf

Rev 1.12 of the HID Usage Tables (hut1_12.pdf)
http://www.usb.org/developers/hidpage/Hut1_12v2.pdf

Rev 1.11 of the HID specification (hid1_11.pdf)
http://www.usb.org/developers/hidpage/HID1_11.pdf

Intel Panther Point Controllers

The Panther Point chipset has a xHCI controller and at least one EHCI controller that can share the USB sockets. When the EHCI controller(s) have control of the sockets, all devices are treated as USB 2.0 devices, including Super Speed devices. The xHCI controller will not see a connection. Therefore, you must switch all available sockets to the xHCI controller, noting that not all sockets may be switchable.

To do this, you write 0xFFFFFFFF to the PCI's Config Space Registers USB3_PSSN (0xD8) and XUSB2PR (0xD0). Register USB3_PSSN being the xHCI control mask with a set bit indicating an xHCI socket, and register XUSB2PR being the EHCI control mask with a clear bit indicating an EHCI socket. Therefore setting all bits in both registers will switch all available xHCI sockets to the xHCI controller.

To see if the installed controller is a Panther Point, you must check the PCI VendorID and DeviceID values. The VendorID will be 0x8086 indicating Intel, while the DeviceID should be 0x1E31, with a revision register value of 04h.

UHCI and the Schedule

It seems that the way I was describing the UHCI's stack was quite confusing to some people, so I have reworded it and updated the outline at the end of chapter 4. See the next page for the modified outline.

Outline 4-1: The UHCI Process