Errata for “USB Revision 2.0 April 27, 2000” as of May 28, 2002:

Check the http://www.usb.org website for the latest errata.

Chapter 7

Clarify endpoint number used in Test_SE0_NAK:

Background: There is potential confusion over the endpoint number used in the IN packet sent to devices when in test mode Test_SE0_NAK. No known impact to host controllers or devices.

Change: p 169, section 7.1.20, 1st bullet, 3rd sentence: Change sentence, “In addition, while in this mode, upstream facing ports (and only upstream facing ports) must respond to any IN token packet directed to the device (with an endpoint number supported by the device), and optionally to any IN token packet at all, with a NAK handshake (only if the packet CRC is determined to be correct) within the normal device response time.

Note: This means that the host should send test packets to a device under test with the device’s current address and endpoint number zero to ensure that the device will respond.”

Modify cable attenuation values to allow 5 meter cables:

Background: The USB 2.0 specification always intended to allow cables up to 5 meters in length. Analysis of the allowed cable attenuation values in the spec has shown that the value at 12.0 Mhz is transposed and should be .76 db not .67 db.

Change: p. 167, table 7-6: Change 12.0 Mhz attenuation value from .67 to .76. A portion of the corrected table is shown below.

Frequency Mhz	db
.512	0.13
.772	0.15
1.0	0.20
4.0	0.39
8.0	0.57
12.0	0.76
24.0	0.95
48.0	1.35
96.0	1.90
200.0	3.20
400.0	5.80

Chapter 8

Clarify the intended behavior and specification of NAK'd OUTs:

Background: There is a confusion over the correct usage and interpretation of NAK responses to High-speed bulk/control OUT transactions. This change and a corresponding change in Chapter 9 clarify the required aspects of this USB feature. The intent was 1) to require a device to specify the maximum number of OUTs it could NAK in some period (this would be tested as part of compliance) and 2) to allow the host to limit the attempted OUT transactions to the number specified by the device. The current specification language is inconsistent and somewhat confusing in describing this. This change impacts high-speed devices with bulk OUT and control endpoints.

Change: p. 217, section 8.5.1, 5th paragraph, after 3rd sentence: Break for new paragraph. Change sentence starting "If a device responds with a NAK in a (micro)frame, the host . . ." to "If a device responds with a NAK to an OUT in a microframe, the host . . ."

Chapter 9

Correct use of version 2.0 in the bcdUSB field of the device descriptor.

Background: There has been confusion over when 0200 should be used in the bcdUSB field in the device descriptor. Many people have assumed that 0200 is somehow associated with high speed peripherals and that full speed or low speed peripherals should not use it for their bcdUSB value. This is not correct. Any peripheral designed to the USB 2.0 specification should use 0200 as the bcdUSB value.

Change: Page 261, Section 9.6.1, 3rd paragraph**.** Change paragraph to the following:
"The DEVICE descriptor of a high-speed capable device has a version number of 2.0 (0200H). Full speed and low speed only devices designed to this specification should also use version number 2.0 (0200H). If the device is full-speed or low-speed only this version number indicates that it will respond correctly to a request for the device_qualifier descriptor (i.e., it will respond with a request error)."

Clarify the intended behavior and specification of NAK'd OUTs:

Background: There is a confusion over the correct usage and interpretation of NAK responses to High-speed bulk/control OUT transactions. This change and a corresponding change in Chapter 9 clarify the required aspects of this USB feature. The intent was 1) to require a device to specify the maximum number of OUTs it could NAK in some period (this would be tested as part of compliance) and 2) to allow the host to limit the attempted OUT transactions to the number specified by the device. The current specification language is inconsistent and somewhat confusing in describing this. This change impacts high-speed devices with bulk OUT and control endpoints.

Change: p.271, table 9-13, description of bInterval: change bulk/control description from "…at most 1 NAK each…" to "… after a NAK to an OUT that the next retry is after …". At end of paragraph, add "See Section 8.5.1 for more PING detail."

Chapter 10

Add host software requirement to suspend downstream ports before suspending a hub:

Background: There is a very unlikely possibility of a race condition between a remote wakeup event and the process of suspending a hub that could create unintended signaling events. This issue can be avoided if system software suspends a downstream facing hub ports before suspending a hub.

Change: p. 282, section 10.2.7, Add new paragraph, “The USB system should suspend all downstream facing ports of a hub before suspending the hub.”

Chapter 11

Allow orange in addition to amber LEDs for downstream facing hub ports:

Background: Some LED vendors do not provide any Green/Amber (bi-color) LEDs. However, almost all LED vendors provide Green/Orange LEDs. Allow either Amber or Orange bi-color LEDs.
Change: Change references from “amber” to “amber (or orange)”

Different port indicator behavior for hubs with/without power switching.

Background: The specification currently indicates that port indicator states are not necessarily consistent in automatic mode for hubs with/without power switching. This spec allowance is being removed.

Change: p. 316, Table 11-6. Remove the column for power switching and the row for without power switching.

Port indicator behavior through a hub suspend/resume cycle.

Background: There has been some confusion over how the port indicators on a hub should behave if the hub is suspended and then resumed. The core specification states that port indicators should be off when a hub is suspended. This is true whether the indicator is in automatic or manual mode. Port indicator behavior should be consistent and this behavior may be necessary for bus-powered hubs. Furthermore, when the hub is resumed the port indicator should return to its previous state before the hub was suspended.

Change: p. 316, 2nd paragraph. Add the following to the end of the paragraph. When the hub is suspended the port indicator must be off regardless of its state before the hub was suspended. When the hub is resumed, the port indicator must return to its state before the hub was suspended, unless the port indicator is in automatic mode and a port status change requires the indicator to change.

Change p. 317, Figure 11-11. Add the following note to the figure: When the hub is suspended the port indicator must be off regardless of its state before the hub was suspended. When the hub is resumed, the port indicator must return to its state before the hub was suspended, unless the port indicator is in automatic mode and a port status change requires the indicator to change.

Clarify the treatment of remote wake up (RWU) in the hub repeater:

Background: There is some potential confusion in the understanding of remote wake up in the hub repeater. These changes make more explicit some of the handling behavior of the hub repeater. This change may impact USB 2.0 hubs.

Change:
p. 308, section 11.4.4: replace "and SOP_FD…Repeater" with "TrueRWU from a downstream port".
p. 313, section 11.5.1.6: delete the third bullet "From the Suspended state…".
p. 315, sections 11.5.1.12 and .13 2nd paragraph, 2nd sentence: delete the phrase "…to the repeater…".
p.322, Figure 11-13: replace "Rptr_WFEOP" with "TrueRWUor".
p. 323, section 11.6.4, table 11-9: replace row "Rptr_WFEOP" with:

TrueRWUor	Downstream port	Logic OR of TrueRWU from Internal port or from downstream ports.

p.324, section 11.6.4.6: replace 1st sentence with. "The port enters this state from the Inactive state on TrueRWU from a downstream port."
p. 432, section 11.24.2.7.2.3: replace "Send_EOP" with "Send_EOR"

Clarify handling of complete-split isochronous IN DATAx PIDs:

Background: USB 2.0 specification is ambiguous about HC and TT requirements in expected DATA packet PIDs allowed during a high-speed complete-split isochronous IN DATA packet. Clarify that the DATAx packet PID returned by the TT for an isochronous complete split is the same DATAx packet PID returned by the TT for the corresponding full-speed transaction. An HC may therefore receive (and accept as valid) a DATA1 packet PID if the originating full-speed device sends a DATA1. No known impact on HC and Hub TT implementations in progress.

Note: It is also acceptable for the TT to always use DATA0 in this case – even if the device sends a DATA1.

Change: p. 397, Add an asterix in Figure 11-85 on the DATA0 (cd1 transition) "bubble". Add text "*TT accepts a DATA0 or DATA1 data packet received from downstream full-speed bus and returns that packet in the corresponding complete-split on the high-speed bus. A TT may choose to always send a DATA0, even if it receives a DATA1."

Clarification of Allowed Behavior for Data Carve Up Around Microframe Boundaries:

Background: Chapter 11 currently implies that Transaction Translators use a byte data engine and should respond with all data except for the last two bytes received in microframe X in response to the complete split in microframe X+1 for an isochronous or interrupt IN transaction that spans X/X+1 microframe boundary on the full/low speed bus. Examples of this behavior are shown in figure 11-82 and 11-93 in Chapter 11. However, the common industry design for USB 2.0 hubs have used a DWORD based data engine. With these designs a transaction translator "effectively" is not aware of data until it has a full DWORD. This means that up to 5.99999 . . . bytes could be carried forward to the next microframe (the two reserved potential CRC bytes plus anything up to another full DWORD of data). As a result of this behavior a Transaction Translator could return as many as 192 data bytes in response to an Isochronous IN complete split. The question has arisen as to whether this should be legal behavior. Several hub implementations have already been designed with this behavior. Changing the size of the data path would add significant delays and cost to USB 2.0 hub implementations. The issue was also not clearly discussed across the EHCI and core specifications. Therefore, the core spec is being modified to allow the transaction translator to carry forward up to (but not including) a full DWORD of data in addition to the 2 potential CRC bytes when an Isochronous or Interrupt IN transaction spans a microframe boundary.

A change is also being added to the EHCI specification to clearly specify that host must be able to handle up to 192 data bytes in response to an Isochronous IN complete split.

Note: If the transaction does end in microframe X (no matter how close to the microframe X+1 boundary) the transaction translator must respond with all data to the complete split in microframe X+1.

Change: Ch. 11, p. 378, section 11.18.5, 1st paragraph. Add the following sentences to the end of the paragraph: "The TT must hold back at least the last two data bytes to prevent returning the CRC for an interrupt or isochronous IN transaction that spans the microframe boundary. The TT is allowed to hold back up to (but not including) an additional DWORD (4 bytes). The TT must provide all remaining data to the complete split in the microframe immediately following the microframe where the transaction ended (no matter how close to the microframe boundary the transaction ends).

The approach used for full-speed isochronous INs and interrupt INs/OUTs ensures that there is always an opportunity for the TT to return data/results whenever it has something to return from the full/low-speed transaction. Then whenever the full-/low speed handler starts the full-/low speed transaction, it simply accumulates the results in each microframe and then returns it in response to a complete-split from the host. The TT acts similar to an isochronous device in that it uses the microframe boundary to "carve up" the full-/low speed data to be returned to the host. The TT does not do any computation on how much data to return at what time. In response to the "next" high-speed complete-split, the TT simply returns the endpoint data it has received from the full-/low-speed bus in a microframe. The TT must hold back at least the last two data bytes to prevent returning the CRC for an interrupt or isochronous IN transaction that spans the microframe boundary. The TT is allowed to hold back up to (but not including) an additional DWORD (4 bytes). -The TT must provide all remaining data to the complete split in the microframe immediately following the microframe where the transaction ended (no matter how close to the microframe boundary the transaction ends. The boundary is defined as the beginning of the micro SOF based on the hubs internal timer.).

Change: Ch. 11, p. 394, section 11.20.4, last paragraph. Add the following sentences as a new paragraph below figure 11-82. "The transaction translator must hold back at least the last 2 bytes for an interrupt IN transaction spanning a microframe boundary. It may hold back up to (but not including) the last 6 bytes (2 for the CRC and up to an additional DWORD to accommodate DWORD based data paths)."

Change: Ch. 11, p. 404, section 11.21.4, last paragraph. Add the following sentences as a new paragraph below figure 11.93. "The transaction translator must hold back at least the last 2 bytes for an isochronous IN transaction spanning a microframe boundary. It may hold back up to (but not including) the last 6 bytes (2 for the CRC and up to an additional DWORD to accommodate DWORD based data paths). Thus a TT could respond with up to 192 bytes in response to an isochronous IN complete split."

How should a TT handle packet babble from a device on a control or bulk transfer?

Background: The core spec never specifically addresses the case where a device responds with more bytes than the maximum allowed size for a control or bulk transfer. In any such case the goal of the TT should be to ensure the behavior from the device's point of view is the same is it would be below a UHCI or OHCI host. Since an OHCI or UHCI host would not retry in these cases – neither must the TT. One option for the TT is simply to return the bytes as they were sent by the device. The host can detect and deal with the babble. However, if the TT uses a minimal buffering solution this may not be possible. The other option for the TT is to return a STALL result for the transfer WITHOUT retrying it locally.

Change: Ch. 11, p. 372, Section 11.17.6. Add the following section below section 11.17.5.

11.17.6 Bulk/Control packet babble handling.
If a TT receives a bulk or control packet from a device that exceeds the maximum byte size allowed by the spec for that transfer type it has two options:

1. Return the data to the host as received.
2. Return STALL WITHOUT retrying the transaction locally.

How should a host handle a NAK response from the device to a setup performed using the split transaction protocol?

Background: The transaction sequence diagrams in the core specification are ambiguous with respect to how a host controller should handle a NAK response to the complete split for a control setup transaction. Immediately retrying the start split without incrementing err_count will be inefficient in what is almost certainly a device protocol error case. In this case the host should increment err_count and only retry the start split if err_count is less than three.

Change: Ch. 11, p. 363, Figure 11-49. Add the following to the Host response to the NAK response (currently says only Retry start split). If the complete split is for a Setup increment err_count. If err_count >= 3 endpoint halt, if err_count < 3 retry start split.

Meaning of bHubContrCurrent field in Hub Descriptor:

Background: There has been some confusion over the meaning of the bHubContrCurrent field in the Hub Descriptor. Some vendors have assumed the field should be current usage of the hub controller electronics (whether from VBUS or from an external power supply). They felt it would be redundant with the bMaxPower field in the configuration descriptor to report consumption from VBUS again. Other vendors have assumed that field should reflect the consumption of the hub controller electronics from VBUS without the current draw of any embedded devices (bMaxPower for a hub can include the draw from embedded devices - as long as the embedded devices don't also report current draw in their own bMaxPower fields). The core spec is being clarified to indicate the second interpretation must be used. This allows software to intelligently perform power management in cases where embedded devices have their power requirements included with the hub's bMaxPower field.

Change: Ch. 11, p. 418, offset 6 in table Change the Description of the bHubContrCurrent from "Maximum current requirements of the Hub Controller electronics in mA" to "Maximum current requirements of the Hub Controller electronics from VBUS. This value must not include current draw for any embedded devices."

Downstream port behavior after Stop_TT request.

Background: There have been questions over whether a hub should continue to generate SOFs on an enabled downstream port controlled by a TT that has been stopped via a Stop_TT request. The hub should continue to generate SOFs in this situation.

Change: Ch. 11, p. 434, section 11.24.2.11. 1st paragraph. Add the following sentence to the end of the first paragraph. “Any downstream port that was generating SOFs controlled by a transaction translator that is being stopped should continue to do so.”

USB 2.0 Spec Errata dated 12/07/2000

Standard Descriptors for Hub Class:

Background: The 12/07/2000 Errata for the USB 2.0 Specification contained several errata to the standard descriptors for the hub class. One additional change to these errata is still required. The Other_Speed_Configuration Descriptor for a full speed operating hub should always have bNumInterfaces set to 1 (not 1 for single TT and 2 for multiple TT). A multiple TT hub has two alternate settings not two interfaces.

Change: 12/07/2000 Errata, page 17. Change text in Other_Speed_Configuration Descriptor bNumInterfaces field to "1".