Errata for "Universal Serial Bus Communications Class Subclass Specifications for Network Control Model Devices Revision 1.0 April 30, 2009" as of November 19, 2010

Issue A

Related to: Table 6-3: NTB Parameter Structure; Table entry for bmNtbFormatsSupported

Issue: Bits 2-15 are reserved bits but it is not declared that these must be set to zero. (Compare to Table 5-2 bmNetworkCapabilities, which also has reserved bits and does declare that they must be zero)

Resolution Description: Update to "Bits 2 – 15: reserved (reset to zero; must be ignored by host)"

Correct as shown below:

Table 6-1: NTB Parameter Structure

Offset	Field	Size	Value	Description
0	wLength	2	Number	Size of this structure, in bytes = 1Ch.
2	bmNtbFormatsSupported	2	Bitmap	Bit 0: 16-bit NTB supported (set to 1) Bit 1: 32-bit NTB supported Bits 2 – 15: reserved (reset to zero; must be ignored by host)

....

Compatibility Impact: No change in intent

Issue B

Related to: Table 6-1: Networking Control Model Requests; Tables for GetNtbFormat & SetNtbFormat

Issue: Commands for manipulation of NTB format are declared as optional, but they may be mandatory when 32bit NTB formats are available. No explicit indication is available to indicate/determine if these are supported.

Resolution Description: Add text to sections 6.2.4 and 6.2.5 declaring that these must be supported by the device if it declares support for an NTB size other than 16bit, and that the host must not issue these commands to a device that only supports the 16bit NTB size. Also add text to 6.2.1 to cover this question.

Correct 6.2.4 (GetNtbFormat) as below:

> To get the full response, the host should set wLength to at least 2. The function shall never return more than 2 bytes in response to this command.
>
> This command must be supported by the function if it declares support for an NTB size other than 16bit in bmNtbFormatsSupported.

If the function does not support NTB sizes other than 16bit, then the host must not issue this command to the function.

Correct 6.2.5 (SetNtbFormat) as below:

If the value passed in wValue is not supported, the function shall return an error response (a STALL PID) and shall not change the format it is using to send and receive NTBs.

This command must be supported by the function if it declares support for an NTB size other than 16bit in bmNtbFormatsSupported.

If the function does not support NTB sizes other than 16bit, then the host must not issue this command to the function.

Correct 6.2.1 (GetNtbParameters) as below:

All NCM functions shall support 16-bit NTBs. Therefore, bit 0 of bmNtbFormatsSupported shall always be set to 1. NCM functions may support 32-bit NTBs. If 32-bit NTBs are supported, then GetNtbFormat and SetNtbFormat must be supported.

Change first paragraph of section 6.2 as follows:

Table 6-1 lists all of the class-specific requests that are valid for an NCM Communications Interface. This table includes those defined in [USBECM12]. Commands marked as "required" must be implemented by any conforming NCM function. Commands marked as "optional" must be implemented in some circumstances; see the Reference section for each command for details.

Compatibility Impact: No change in intent

Issue C

Related to: 3.7 Null NCM Datagram Pointer Entries; First sentence of section

Issue: Sentence discusses the "offset" field, but offset field does not exist.

Resolution Description: Change text to "Any NCM Datagram pointer entry with an index field of zero or with a length field of zero, or with both index and length fields set to zero, shall be treated as a Null entry."

Correct as below:

Any NCM Datagram pointer entry with an index field (wDatagramIndex, dwDatagramIndex) of zero or with a length field (wDatagramLength, dwDatagramLength) of zero, or with both index and length fields set to zero, shall be treated as a Null entry. Receivers shall process datagram pointer entries sequentially from the first entry in the NTB. The first Null entry shall be interpreted as meaning that all following NCM Datagram Pointer Entries in the NDP are to be ignored.

Deleted: offset

Deleted: offset

Compatibility Impact: No change in intent

Issue D

Related to: 5.2.3 Command Set Detail Functional Descriptor

Issue: Text in NCM 5.2.3 is incorrect

Resolution: Fix text.

Correct as below:

> If a Command Set Functional Descriptor appears, it may be followed by one or more Command Set Detail Functional Descriptors, as described in [USBWMC11], section 8.1.2.3. If the Command Set Functional Descriptor is not present, Command Set Detail Functional Descriptors shall not appear, and the host shall ignore any such descriptors.

Deleted: Detail

Deleted: Detail

Compatibility Impact: No change in intent

Issue E

Related to: Table 3-1 NTH16: Offset 10, wFpIndex
Table 3-2 NTH32: Offset 12, dwFpIndex
Table 3-3 NDP16: Offset 6, wNextFpIndex

Issue: In all three places, it would be more correct to name these fields “NdpIndex” instead of “FpIndex”.

Note: Table 3-4, NDP32 offset 8 is titled dwNextNdpIndex, and does not need this change.

Resolution: Correct as follows:

Table 3-1: Sixteen Bit NCM Transfer Header (NTH16)

Offset	Field	Size	Value	Description
0	dwSignature	4	Number (0x484D434E)	Signature of the NTH16 Header. This is transmitted in little-endian form, i.e., as 0x4E, 0x43, 0x4D, 0x48, or as the character sequence “NCMH”
4	wHeaderLength	2	Number (0x000C)	Size in bytes of this NTH16 structure, in little-endian format.
6	wSequence	2	Number	Sequence number. The transmitter of a block shall set this to zero in the first NTB transferred after every “function reset” event, and shall increment for every NTB subsequently transferred. The effect of an out-of-sequence block on the receiver is not specified. The specification allows the receiver to decide whether to check the sequence number, and to decide how to respond if it’s incorrect. The sequence number is primarily supplied for debugging purposes.
8	wBlockLength	2	Number	Size of this NTB in bytes (“L” in Figure 3-1). Represented in little-endian form. NTB size (IN/OUT) shall not exceed *dwNtbInMaxSize* or *dwNtbOutMaxSize* respectively; see Table 6-3 in 6.2.1. If *wBlockLength* = 0x0000, the block is terminated by a short packet. In this case, the USB transfer must still be shorter than *dwNtbInMaxSize* or *dwNtbOutMaxSize*. If exactly *dwNtbInMaxSize* or *dwNtbOutMaxSize* bytes are sent, and the size is a multiple of *wMaxPacketSize* for the given pipe, then no ZLP shall be sent. *wBlockLength*= 0x0000 must be used with extreme care, because of the possibility that the host and device may get out of sync, and because of test issues. *wBlockLength* = 0x0000 allows the sender to reduce latency by starting to send a very large NTB, and then shortening it when the sender discovers that there’s not sufficient data to justify sending a large NTB.
10	wNdpIndex	2	Number	Offset, in little endian, of the first NDP16 from byte zero of the NTB. This value must be a multiple of 4, and must be >= 0x000C.

Deleted: wFpIndex

Table 3-2: Thirty-two bit NCM Transfer Header (NTH32)

Offset	Field	Size	Value	Description
0	dwSignature	4	Number (0x686D636E)	Signature of the NTH32 Header. This is transmitted in little-endian form, i.e., as 0x6E, 0x63, 0x6D, 0x68, or as the character sequence "ncmh"
4	wHeaderLength	2	Number (0x0010)	Size in bytes of this NTH32 structure, in little-endian format.
6	wSequence	2	Number	Sequence number. The transmitter of a block shall set this to zero in the first NTB transferred after every "function reset" event, and shall increment for every NTB subsequently transferred. The effect of an out-of-sequence block on the receiver is not specified. The specification allows the receiver to decide whether to check the sequence number, and to decide how to respond if it's incorrect. The sequence number is primarily supplied for debugging purposes.
8	dwBlockLength	4	Number	Size of this NTB in bytes ("L" in Figure 3-2). Represented in little-endian form. NTB size (IN/OUT) shall not exceed *dwNtbInMaxSize* or *dwNtbOutMaxSize* respectively; see Table 6-3 in 6.2.1. If *dwBlockLength* = 0x0000, the block is terminated by a short packet. In this case, the USB transfer must still be shorter than *dwNtbInMaxSize* or *dwNtbOutMaxSize*. If exactly *dwNtbInMaxSize* or *dwNtbOutMaxSize* bytes are sent, and the size is a multiple of *wMaxPacketSize* for the given pipe, then no ZLP shall be sent. *dwBlockLength* = 0x0000 must be used with extreme care, because of the possibility that the host and device may get out of sync, and because of test issues. *dwBlockLength* = 0x0000 allows the sender to reduce latency by starting to send a very large NTB, and then shortening it when the sender discovers that there's not sufficient data to justify sending a large NTB.
12	dwNdpIndex	4	Number	Offset, in little endian, of the first NDP32 from byte zero of the NTB. This value must be a multiple of 4, and must be >= 0x0010 (because the first NDP32 had to be after the end of the NTH32).

Deleted: dwFpIndex

Table 3-3: Sixteen-bit NCM Datagram Pointer Table (NDP16)

Offset	Field	Size	Value	Description
0	dwSignature	4	Number (0x304D434E, 0x314D434E)	Signature of this NDP16. This is transmitted in little-endian form, i.e., as 0x4E, 0x43, 0x4D, 0x30 or 0x4E, 0x43, 0x4D, 0x31. or as the character sequences "NCM0",or "NCM1 where "0" or '1" has the meaning given in Table 3-5.
4	wLength	2	Number	Size of this NDP16, in little-endian format. This must be a multiple of 4, and must be at least 16 (0x0010).
6	wNextNdpIndex	2	Reserved (0)	Reserved for use as a link to the next NDP16 in the NTB
8	wDatagramIndex[0]	2	Number	Byte index, in little endian, of the first datagram described by this NDP16. The index is from byte zero of the NTB. This value must be must be >= the value stored in *wHeaderLength* of the NTH16 (because it must point past the NTH16).
10	wDatagramLength[0]	2	Number	Byte length, in little endian, of the first datagram described by this NDP16. For Ethernet frames, this value must be >= 14.
12	wDatagramIndex[1]	2	Number	Byte index, in little endian, of the second datagram described by this NDP16. If zero, then this marks the end of the sequence of datagrams in this NDP16.
14	wDatagramLength[1]	2	Number	Byte length, in little endian, of the second datagram described by this NDP16. If zero, then this marks the end of the sequence of datagrams in this NDP16.
…				
wLength-4	wDatagramIndex[(wLength-8) / 4 - 1]	2	Number (0)	Always zero
wLength – 2	wDatagramLength[(wLength-8) / 4 - 1]	2	Number(0)	Always zero

Deleted: wNextFpIndex

Compatibility Impact: No change in intent

Issue F

Related to: 6.2.9 SetMaxDatagramSize

Issue: Text says, "see section 7.1 for details." However, Section 7.1 is "Notification Sequencing".

Resolution: Correct cross-reference in last paragraph of 6.2.9 to refer to Section 7.2, as follows:

> The function's maximum datagram size is set to a default value by events outside the scope of this command; see section 7.2 for details.

Deleted: 7.1

Compatibility Impact: No change in intent.

Issue G

Related to: 7.2 Using Alternate Settings to Reset an NCM Function

Issue: Bullet item says “reset the current Maximum Segment Size to a function-specific default." This is inconsistent with rest of the document, which refers to a Segment as a Datagram. In addition, reviewers requested an explicit statement that (if SetMaxDatagramSize is supported), the host must query to determine the default size, or explicitly set a size prior to selecting the second alternate interface setting of the data interface.

Correct as below:

- reset the current Maximum Datagram Size to a function-specific default. If *SetMaxDatagramSize* is not supported, then the maximum datagram size shall be the same as the value in *wMaxSegmentSize* of the Ethernet Networking Functional Descriptor. If *SetMaxDatagramSize* is supported by the function, then the host must either query the function to determine the current effective maximum datagram size, or must explicitly set the maximum datagram size. If the host wishes to set the Maximum Datagram Size, it may do so prior to selecting the second alternate interface setting of the data interface. Doing so will ensure that the change takes effect prior to send or receiving data.

Deleted: Segment

Compatibility Impact: No change in intent.

Issue H

Related to: Table 5-2, Offset 1:

Issue: Value of 'bDescriptorType' is mentioned as '0x22'; this is incorrect, as it doesn't match the value of CS_INTERFACE defined in the CDC 1.2 specification.

Resolution: Change value of bDescriptorType to '0x24'.

Table 5-1: NCM Functional Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	6	Size of Descriptor in bytes
1	bDescriptorType	1	Constant	CS_INTERFACE (0x24)
2	bDescriptorSubtype	1	Constant (1Ah)	NCM Functional Descriptor subtype, as defined in Table 4-4
3	bcdNcmVersion	2	Number 0x0100	Release number of this specification in BCD, with implied decimal point between bits 7 and 8. 0x0100 == 1.00 == 1.0. This is a little-endian constant, so the bytes will be 0x00, 0x01.
5	bmNetworkCapabilities	1	Bitmap	Specifies the capabilities of this function. A bit value of zero indicates that the capability is not supported. D7..D5: Reserved (zero) D4: Function can process *SetCrcMode* and *GetCrcMode* requests D3: Function can process *SetMaxDatagramSize* and *GetMaxDatagramSize* requests. D2: Function can process *SendEncapsulatedCommand* and *GetEncapsulatedResponse* requests. D1: Function can process *GetNetAddress* and *SetNetAddress* requests. D0: Function can process *SetEthenetPacketFilter* requests, as defined in [USBECM12]. If not set, broadcast, directed and multicast packets are always passed to the host.

Deleted: 0x22

Compatibility Impact: No change in intent. However, implementers of class drivers should be aware that some NCM functions may have been implemented with the wrong code value in bDescriptorType.

Issue I

Issue: Table 6-1: several sections have incorrect section cross-references.

Resolution: Correct cross-references as shown.

Table 6-1: Networking Control Model Requests

Request	Description	Req'd/Opt	Reference
SendEncapsulatedCommand	Issues a command in the format of the supported control protocol. The intent of this mechanism is to support networking functions (e.g. host-based cable modems) that require an additional vendor-defined interface for media specific hardware configuration and management.	Optional	[USBCDC12]
GetEncapsulatedResponse	Requests a response in the format of the supported control protocol.	Optional	[USBCDC12]
SetEthernetMulticastFilters	Controls the receipt of Ethernet frames that are received with "multicast" destination addresses.	Optional	[USBECM12]
SetEthernetPowerManagement-PatternFilter	Some hosts are able to conserve energy and stay quiet in a "sleeping" state while not being used. NCM functions may provide special pattern filtering hardware that enables the function to wake up the attached host on demand when something is attempting to contact the host (e.g. an incoming web browser connection). This command allows the host to specify the filter values that detect these special frames.	Optional	[USBECM12]
GetEthernetPowerManagement-PatternFilter	Retrieves the status of the above power management pattern filter setting	Optional	[USBECM12]
SetEthernetPacketFilter	Controls the types of Ethernet frames that are to be received via the function.	Optional	[USBECM12]
GetEthernetStatistic	Retrieves Ethernet statistics such as frames transmitted, frames received, and bad frames received.	Optional	[USBECM12]
GetNtbParameters	Requests the function to report parameters that characterize the Network Control Block	Required	6.2.1
GetNetAddress	Requests the current EUI-48 network address	Optional	6.2.2
SetNetAddress	Changes the current EUI-48 network address	Optional	6.2.3
GetNtbFormat	Get current NTB Format	Optional	6.2.4
SetNtbFormat	Select 16 or 32 bit Network Transfer Blocks	Optional	6.2.5
GetNtbInputSize	Get the current value of maximum NTB input size	Required	6.2.6
SetNtbInputSize	Selects the maximum size of NTBs to be transmitted by the function over the bulk IN pipe.	Required	6.2.7
GetMaxDatagramSize	Requests the current maximum datagram size	Optional	6.2.8
SetMaxDatagramSize	Sets the maximum datagram size to a value other than the default	Optional	6.2.9
GetCrcMode	Requests the current CRC mode	Optional	6.2.10
SetCrcMode	Sets the current CRC mode	Optional	6.2.11

Deleted: 6.3.1

Deleted: 0

Deleted: 6.2.4

Deleted: 6.2.6

Compatibility Impact: No change in intent.

Issue J

Related to: Table 3-3, Table 3-4

Issue: duplicated text

Resolution: correct as shown below:

Table 3-1: Sixteen-bit NCM Datagram Pointer Table (NDP16)

Offset	Field	Size	Value	Description
			...	
8	wDatagramIndex[0]	2	Number	Byte index, in little endian, of the first datagram described by this NDP16. The index is from byte zero of the NTB. This value must be >= the value stored in *wHeaderLength* of the NTH16 (because it must point past the NTH16).
			...	

Deleted: must be

Table 3-2: Thirty-two bit NCM Datagram Pointer Table (NDP32)

Offset	Field	Size	Value	Description
			...	
16	dwDatagramIndex[0]	4	Number	Byte index, in little endian, of the first datagram described by this NDP32. The index is from byte zero of the NTB. This value must be >= 0x0010 (because it must point past the NTH).

Deleted: must be

Compatibility Impact: No change in intent.

Issue K

Related to: Table 6-3: NTB Parameter Structure; Table entry for –reserved at offset 26

Issue: Some devices need to inform the host about a limitation on number of datagrams that may be provided by the host in an NTB sent to the device. This limitation was inadvertently omitted from the 1.0 specification.

Resolution Description: Rename "reserved" at offset 26 of the NTB Parameter Structure to "*wNtbOutMaxDatagrams*", with Description "Maximum number of output datagrams within the NTB. Zero means no limit".

Correct as shown below:

Table 6-1: NTB Parameter Structure

Offset	Field	Size	Value	Description
...				
26	wNtbOutMaxDatagrams	2	Number	Maximum number of datagrams that the host may pack into a single OUT NTB. Zero means that the device imposes no limit

Deleted: -reserved

Deleted: Padding. Shall be transmitted as zero by function and ignored by host.

Compatibility Impact: No change in intent. Old devices which had no limitation continue to impose no limitation on the host. Devices constructed using this erratum, and using a non-zero value in this field, may receive extra datagrams from older host drivers; such devices must be prepared for this situation and adapt in a vendor-specific fashion (for example, by dropping datagrams if too many are received).

Issue L

Related to: Table 5-2, Section 6.2.6 and Section 6.2.7.

Issue: Some embedded hosts need to inform the device about a limitation on number of datagrams that may be provided by the device in an NTB sent to the host. This limitation was inadvertently omitted from the 1.0 specification.

Resolution:

- In Table 5-2, use D5 in bmNetworkCapabilities to signal that the device can transfer 8 bytes of data when processing the SetNtbInputSize and GetNtbInputSize requests.
- In Section 6.2.7, Add NTB Input Size structure to SetNtbInputSize, and add wNtbInMaxDatagrams to this structure
- In Section 6.2.6, reference the NTB Input Size structure for the response of GetNtbInputSize.

- **Table 5-1: NCM Functional Descriptor**

Offset	Field	Size	Value	Description
0	bFunctionLength	1	6	Size of Descriptor in bytes
1	bDescriptorType	1	Constant	CS_INTERFACE (0x22)
2	bDescriptorSubtype	1	Constant (1Ah)	NCM Functional Descriptor subtype, as defined in Table 4-4
3	bcdNcmVersion	2	Number 0x0100	Release number of this specification in BCD, with implied decimal point between bits 7 and 8. 0x0100 == 1.00 == 1.0. This is a little-endian constant, so the bytes will be 0x00, 0x01.
5	bmNetworkCapabilities	1	Bitmap	Specifies the capabilities of this function. A bit value of zero indicates that the capability is not supported. D7..D6: Reserved (zero) D5: Function can process 8-byte forms of *GetNtbInputSize* and *SetNtbInputSize* requests D4: Function can process *SetCrcMode* and *GetCrcMode* requests D3: Function can process *SetMaxDatagramSize* and *GetMaxDatagramSize* requests. D2: Function can process *SendEncapsulatedCommand* and *GetEncapsulatedResponse* requests. D1: Function can process *GetNetAddress* and *SetNetAddress* requests. D0: Function can process *SetEthenetPacketFilter* requests, as defined in [USBECM12]. If not set, broadcast, directed and multicast packets are always passed to the host.

Deleted: D5

6.2.6 GetNtbInputSize

This request returns NTB input size currently being used by the function.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100001B	GET_NTB_INPUT_SIZE	Zero	NCM Communications Interface	Number of bytes to read	The NTB input size structure, as defined in *SetNtbInputSize* (6.2.7).

To get the NTB input size, the host should set *wLength* to at least 4. To get the full NTB input size structure, the host should set *wLength* to at least 8. If bit D5 is set in field *bmNetworkCapabilities* of the function's NCM Functional Descriptor, the function shall never return more than 8 bytes in response to this command. If bit D5 is reset, the function shall never return more than 4 bytes in response to this command. The fields in the input size structure are returned in little-endian order.

Deleted: full

Deleted: response

Deleted: is

6.2.7 SetNtbInputSize

This request selects the maximum size of NTB that the device is permitted to send to the host, and optionally the maximum number of datagrams that the device is permitted to encode into a single NTB.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	SET_NTB_INPUT_SIZE	Zero	NCM Communications Interface	4 or 8	If wLength is 8, then this is the NTB Input Size Structure. If wLength is 4, then this is the dwNtbInMaxSize field of the NTB Input Size Structure.

Deleted: Maximum NTB size, in bytes, in little-endian order

NTB Input Size Structure

Offset	Field	Size	Value	Description
0	dwNtbInMaxSize	4	Number	IN NTB Maximum size in bytes. The host shall select a size that is at least 2048, and no larger than the maximum size permitted by the function, according to the value given in the NTB Parameter Structure
4	wNtbInMaxDatagrams	2	Number	Maximum number of datagrams within the IN NTB. Zero means no limit
6	-reserved	2	Number	Shall be transmitted as zero and ignored upon receipt.

If bit D5 is set in the *bmNetworkCapabilties* field of the function's NCM Functional Descriptor, the host may set wLength either to 4 or to 8. If *wLength* is 4, the function shall assume that *wNtbInMaxDatagrams* is to be set to zero. If *wLength* is 8, then the function shall use the provided value as the limit. The function shall return an error response (a STALL PID) if *wLength* is set to any other value.

Deleted: The host shall select a size that is at least 2048, and no larger than the maximum size permitted by the function, according to the value given in the NTB Parameter Structure.¶

If bit D5 is reset in the *bmNetworkCapabilties* field of the function's NCM Functional Descriptor, the host shall set *wLength* to 4. The function shall return an error response (a STALL PID) if *wLength* is set to any other value.

If the value passed in the data phase is not valid, or if *wLength* is not valid, the function shall return an error response (a STALL PID) and shall not change the value it uses for preparing NTBs.

Compatibility Impact: No change in intent. Old hosts which have no limitation continue to impose no limitation on the device. New hosts which want to impose a limitation can detect devices that support such limitations. Devices constructed without this feature continue to operate; hosts must be prepared for this situation and adapt in a vendor-specific fashion (for example, by dropping datagrams if too many are received).

Issue M

Related to: sections 4.3, 5.3 and 7.2.

Issue: although the specification is consistent, some readers missed the statement in section 5.3 that the interface descriptors for alternate settings 0 and 1 shall both have bInterfaceProtocol set to 01h.

Resolution:

- Add clarifying language to section 4.3
- Delete confusing language from section 7.2

In section 4.3:

[USBCDC12] defines Data Class Protocols. Alternate settings 0 and 1 of an NCM function's Data Interface shall use the protocol code defined in Table 4-3.

In section 7.2:

To place the network aspects of a function in a known state, the host shall:

- select alternate setting 0 of the NCM Data Interface (this is the setting with no endpoints). This can be done explicitly using *SetInterface*, or implicitly using *SetConfiguration*. See [USB30] for details.
- select the NCM operational parameters by sending commands to the NCM Communication Interface, then
- select the second alternate interface setting of the NCM Data Interface (this is the setting with a bulk IN endpoint and a bulk OUT endpoint).

Deleted: , and with *bInterfaceProtocol* set to 01h, as given in Table 4-3

Compatibility Impact: No change in intent. Hosts that wish to support devices made before the publication of these errata may choose to ignore the bInterfaceProtocol of alternate setting zero.

Issue N

Related to: Interface Association Descriptors

Issue: during interoperability testing, we discovered that some devices used unexpected values in the IAD. For clarity, the group decided it would be good to emphasize the values to be used in the IAD.

Resolution: add new section 7.4

7.4 Using the Interface Association Descriptor

USB Devices containing NCM functions may include an Interface Association Descriptor (IAD) to indicate that the Communication Class interface and the Data Class Interface are to be treated as a single function. (This is an alternative to using the WMC WHCM Union descriptor.)

If an IAD is used, the IAD should appear before the Communication Class Interface. The IAD's *bFunctionClass* shall be set to 02h (Communication Class), which is the same value used in *bInterfaceClass* of the Communication Class interface; *bFunctionSubClass* shall be set to 0Dh (Network Control Model), which is the same value used in *bInterfaceSubClass* of the Communication Class interface; and *bFunctionProtocol* shall be set to the same value as is used in *bInterfaceProtocol* of the Communication Class interface for the function. In accordance with the IAD ECN, *bInterfaceCount* shall be set to 2, and *bFirstInterface* shall be set to the interface number of the Communication Class Interface.

When using an IAD, the general order of descriptors for an NCM function should be:

- Interface Association Descriptor (IAD)
- Communications Class Interface Descriptor (interface n)
- Functional descriptors for the Communication Class Interface
- Endpoint descriptors for the Communication Class Interface
- Data Class Interface (interface n+1, alternate setting 0)
- Functional descriptors for Data Class Interface (interface n+1, alternate setting 0)
- Data Class Interface (interface n+1, alternate setting 1)
- Functional descriptors for Data Class Interface (interface n+1, alternate setting 1)
- Endpoint descriptors for Data Class Interface (interface n+1, alternate setting 1)

Compatibility issues: none. This simply repeats the information from the Interface Association Descriptor ECN for the benefit of the reader.

Issue O

Related to: Notification Sequencing

Issue: the text does not sufficiently emphasize the fact that the connection sequencing is different than that used for Ethernet Control Model (ECM).

Resolution: add following clarification text as a new paragraph at the end of section 7.1.

Developers should be aware that this sequence of notifications (speed before connection) is different than the sequence specified in [USBECM12].

Compatibility issues: none. This simply emphasizes a difference between ECM and NCM for the benefit of the reader.

Issue P

Typographical error in section 7.3.2

Resolution: change text as follows to clarify seemingly spurious word “in”:

While the link is suspended, functions shall observe changes in network connectivity and connection speed, but these changes shall not cause a remote wakeup to be signaled.

Compatibility issues: none.

Issue Q

Related to: frame alignment

Issue: some readers have requested extra information in graphic form to clarify the purpose and use of the wNdp{In,Out}{Divisor,PayloadRemainder} parameters.

Resolution: add the following figures and text at the end of section 3.3.4:

Two use cases are anticipated (although the specification doesn't restrict the user to these two cases).

In one use case, the function wants to align the beginning of an IP packet to a cache line boundary. Cache lines are generally much smaller than the maximum Ethernet frame size. In this case, the *wNdpXxDivisor* is set to the size of a cache line (in bytes), and the *wNdpXxPayloadRemainder* is set to zero. The effect is shown schematically in Figure 3-3.

In another use case, the function wants to place each IP packet in a fixed sized buffer. (This is primarily intended for use on the OUT pipe.) For this to work, each buffer must be bigger than the maximum Ethernet frame size. In this case, *wNdpXxDivisor* is set to the size of the buffer, and *wNdpXxPayloadRemainder* is set to the desired offset of the IP packet in the fixed size buffer. This situation is shown schematically in Figure 3-4.

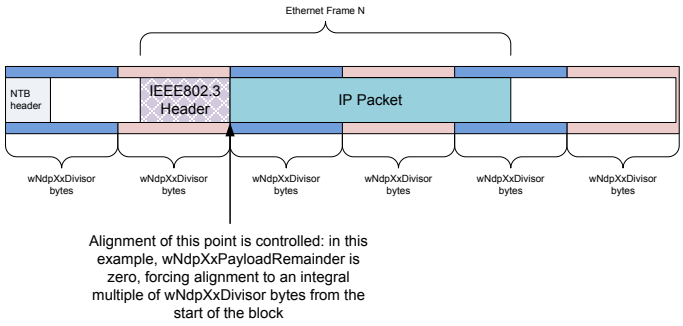

Figure 3-3 – Alignment to a cache line

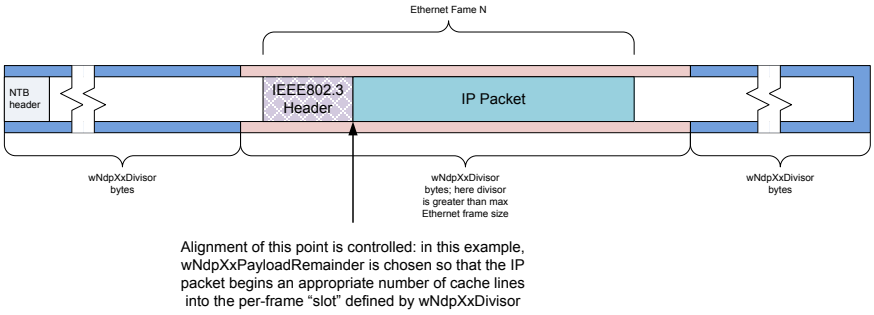

Figure 3-4 – Alignment for fixed-size internal buffers

Compatibility issues: none.

Issue R

Related to: miscellaneous typos

Issue: typographical errors were found while preparing the final documents

Resolution:

In table 3-4, correct as follows:

28	dwDatagramLength[1]	4	Number	Byte length, in little endian, of the second datagram described by this NDP32. If zero, then this marks the end of the sequence of datagrams in this NDP32.

Deleted: NDP16

In section 3.4, correct as follows:

For OUT pipes, the host determines the actual size of the NTB data structures sent to the device. Hosts can discover the maximum size supported by the device from the NTB Parameter Structure, and shall not send NTBs larger than the device can support.

Deleted: NTP

For IN pipes, the host tells the device the size of NTB data structures that it wishes the device to send using the *SetNtbInputSize* command. Hosts can discover the maximum size supported by the device from the field *dwNtbInMaxSize* in the NTB Parameter Structure (Table 6-3). Devices shall not send NTBs larger than the host has requested. The host shall not select a maximum NTB size that is not supported by the device.

Deleted: NTP

Compatibility issues: none.