M O O R E
C O M P U T E R
C O N S U L T A N T S
I N C O R P O R A T E D

MCCI

3520 Krums Corners Road

Ithaca, New York 14850
USA

Phone +1-607-277-1029

Fax +1-607-277-6844

www.mcci.com

Universal Serial Bus MDLM Semantic Model for **SAFE Communication Devices**

bGuid = {5d34cf66-1118-11d6-a21a-000102ca9a7f}

MCCI Engineering Report 950198

Revision 0.8a

January 27, 2002

Revision History

Rev	Date	Filename	Comments
0.8a	1/27/2002	950178-0_8a (SAFE-Network-Protocol).doc	Convert to WMC-compatible form
-	1/23/2002	SAFE.txt	Original version from Stuart Lynne

Please send comments via electronic mail to tmm@mcci.com ***and*** sl@lineo.com

Contributors

Stuart Lynne	Lineo
Terry Moore	MCCI

Table of Contents

List of Tables

1 Introduction

1.1 Purpose

USB device hardware is commonly integrated into much larger system-on-chip (SOC) integrated circuits. Because USB functionality is a small part of the overall function of the device, business pressures often lead to USB devices which have significant limitations relative to the full USB core specification.

This specification is intended to define functional semantics and data plane data transfer conventions which allow such devices to be used reliably for certain kinds of data transport.

1.2 Scope

This document specifies new device subclasses intended for use with Wireless Mobile Communication devices, based on [USBWMC1.1] and in turn [USBCDC1.1].

Because this specification is based on the [USBWMC1.1] specification, the question arises of how to resolve conflicts between that specification and this one. The intention of this specification is that all material presented here be based on the MDLM functional model provided in that specifications. New numeric codes are defined for UUIDs, protocol codes, management elements, and notification elements.

In some cases material from [USBWMC1.1] or [USBCDC1.1] is repeated for clarity. In such cases, the originating document shall be treated as the controlling document.

1.3 Related Documents

[LEACH1998]	*UUIDs and GUIDs,* Paul J. Leach et al, IETF draft draft-leach-uuids-guids-01.txt., February 4, 1998.
[USB2.0]	*Universal Serial Bus Specification*, revision 2.0 (also referred to as the *USB Specification*). This specification is available on the World Wide Web site **http://www.usb.org**.
[USBCDC1.1]	*Universal Serial Bus Class Definitions for Communication Devices*, Version 1.1. This specification is available on the World Wide Web site **http://www.usb.org**.
[USBWMC1.0]	*Universal Serial Bus CDC Subclass Specification for Wireless Mobile Communication Devices*, Version 1.0. This specification is available on the World Wide Web site **http://www.usb.org**.

1.4 Terms and Abbreviations

ACM	Abstract Control Model, a way of representing data/fax modem capabilities in USB devices, defined by [USBCDC1.1].
MDLM	Mobile Direct Line Model, a way of migrating some of the protocol functions of wireless terminal adapters to the USB host system. A mechanism for implementing

	MDLM transport is defined by this document.
SAFE	The name of this MDLM semantic model, which is intended to be safe to use with a variety of real-world embedded USB device functions.

2 Management Overview

The SAFE device class semantic model is intended to be a simple, single data plane, configuration that can be used to support serial or network data on USB Device hardware that has limited scope and possible hardware problems, while allowing for a single class driver to discover and accommodate any conforming device. The specification is loosely based on the semantics of the USB Communication Device Class Abstract Control and Ethernet Control Models, adapted to meet the needs of devices with limited capabilities.

Following the conventions of the USB core document and the USB WMC 1.0 specification, this specification also allows for multi-function devices. The rest of this document uses the term “function” to refer to a single instance that conforms to the SAFE device class, possibly combined with device functions from the same or other device classes

Specifically device functions in accordance with this model can enable the following options as their overall semantic model:

Serial Transport	the function is transporting character data, similar in concept to a serial port. (However, this semantic model deos not include modeling of RS-232 baudrate or handshaking lines over the USB interface.)
Ethernet Packet Transport	the function is transporting Ethernet packet data.

The following independent variables modify the data plane:

CRC	a CRC is appended to data to ensure that received data can be dropped if incorrect. The CRC is appended both on transmit and receive.
Padded	force all USB packets to be either the USB packetsize or the USB packetsize less one in length

3 Assumptions and Constraints

3.1 Compliance

Compliance testing with SAFE is based on the following checklist:

In addition to meeting all the requirements of the USB core specification [USB2.0], devices conforming to this specification must meet the following requirements. Some of these requirements are easily testable, others must be enforced by design.

The testable requirements focus on the content of the descriptors.

1. The Device descriptor shall have bDeviceClass set to Communication class, and bDeviceSubclass and bDeviceProtocol set to zero.

2. Each SAFE Communication class interface must be followed by the appropriate functional descriptors for that interface. These shall include the HEADER functional descriptor, and the MDLM functional descriptor.

3. The GUID of the MDLM functional descriptor shall be {5d34cf66-1118-11d6-a21a-000102ca9a7f}, encoded as the byte sequence 5d, 34, cf, 66, 11, 18, 11, d6, a2, 1a, 00, 01, 02, ca, 9a, 7f.

4. If a SAFE Communication class interface has associated Data class interface, a Union descriptor must appear, specifying the Communication class interface as the primary interface, and the Data class interface(s) as subordinate interfaces. (Note, however, that

5. No Data class interface should appear that is not mentioned in a Union descriptor.

6. If a SAFE function is part of a WHCM handset, each Data class interface that is part of that function shall be mentioned in exactly two Union descriptors. It shall be mentioned in the Union descriptor that follows the primary Communication interface for the function to which it belongs. It shall also be mentioned in the Union descriptor for the WHCM interface that describes the handset to which the Data interface belongs.

   Otherwise, if the SAFE function is not part of a WHCM handset, each Data class interface shall be mentioned in exactly one Union descriptor. It shall be mentioned in the Union descriptor that follows the primary Communication interface for the function to which it belongs.

7. If a SAFE Communication class interface appears with multiple alternate settings, all alternate settings for that interface must have the same bInterfaceClass, bInterfaceSubclass and bInterfaceProtocol codes.

8. The class descriptors associated with a given SAFE Communication class interface must appear sequentially in the configuration bundle *after* the interface descriptor to which they apply, and *before* the next interface descriptor in the bundle (even if that interface descriptor is for an alternate setting for the same interface.

9. If a SAFE Communication class interface appears with multiple alternate settings, functional descriptors must be repeated after each such interface and the Union descriptors must be the same for each alternate setting.

10. Bit zero of the first octet of any Ethernet address associated with a SAFE function must always be zero. (See section 6.1.2.3.)

Non-testable requirements

1. The first three octets of any Ethernet address associated with the device shall be the OUI of the organization assigning the Ethernet address, and the last three octets shall be assigned in such a way as to guarantee uniqueness and consistency.

The intent of this specification is to allow SAFE functions be represented in certain ways. However, this specification is not restrictive. The class-compliant functions of a device must be compliant with the governing class specification (and in accordance with the general requirements given above).

4 Functional Overview

4.1 Device Organization

4.2 Device Operation

4.3 Function Models

4.4 Interface Definitions

4.5 Endpoint Requirements

The minimal requirements are a single configuration with a single interface. The interface must describe a minimum of the following endpoints:

Bulk IN

Bulk Out

It may optionally describe:

Interrupt

Any valid endpoint number may be specified.

The default pipe can be used, as usual, to send USB standard requests and class-specific requests to the function.

For full-speed devices, the maximum USB wMaxPacketSize that may be specified for a Bulk Endpoint is sixty four (64) bytes. For high-speed devices, the USB wMaxPacketSize must be five hundred twelve (512) bytes.

4.6 Device Models

The SAFE device model can be used by itself as the only USB function in a configuration, as part of a multi-function device, or as one of several configurations. For example, a configuration containing a single SAFE device model function could be combined with a configuration containing a CDC Ethernet Control Model function allowing the host to select between the two.

In a multi-function device, a SAFE function may be combined with other functions, for example a DFU interface or a HID-class interface.

5 Class Specific Codes

5.1 MDLM UUID Codes

Table 5-1: MDLM UUID Codes

UUID	Meaning
{5d34cf66-1118-11d6-a21a-000102ca9a7f}	SAFE Networking Model
TBD	SAFE Serial Model

5.2 MDLM Functional Detail Descriptor Codes

Table 5-2: SAFE Class Functional Detail Descriptor Codes

Code	Name	Descriptor
00	SAFE_NET_FUNC_DESC_CODE	SAFE Networking Functional Detail Descriptor

5.3 SAFE Class Management Element Request Codes

Table 5-3: SAFE Class Management Element Request Codes

Code	Subclass
60-7F	Reserved SAFE Semantic-Model specific Requests (32 in all)

5.4 SAFE Class Notification Element Request Codes

Table 5-4: Communication Class Notification Element Request Codes

Code	Subclass
40-5F	Reserved SAFE Semantic-Model-specific Notifications (32 in all).

6 Functional Characteristics

6.1 SAFE Networking Model

The SAFE Networking Model allows for the transport of Ethernet LAN frames over the bulk IN and bulk OUT pipes. It is based on the CDC Ethernet Control Model with the following deviations:

1. No separate Data Class interface is used

2. Packet filtering, multicast filtering and statistics are all intended to be performed on the host

6.1.1 Functional Topology

A LAN frame traffic facility is consists of:

1. A Communications Class/Ethernet Control Model interface with a notification endpoint.

2. A Data Class interface with two endpoints, one BULK IN, the other BULK OUT.

This is just as defined for a single-function Ethernet adapter in [USBCDC1.1]. However, the notification endpoint is required.

6.1.2 Descriptors

6.1.2.1 MDLM Interface Descriptor

One interface descriptor with bInterfaceClass == COMM, bInterfaceSubClass == Ethernet Control Model, and bInterfaceProtocol == 00 shall be embedded in the configuration bundle for each data/fax.

Table 6-5: Communications Class Ethernet Control Model Interface Descriptor

Offset	Field	Size	Value	Description
4	bNumEndpoints	1	Constant (??)	Indicates the number of endpoints in this interface. Should be either 2 or 3.
5	bInterfaceClass	1	Constant (02)	Communication class
6	bInterfaceSubClass	1	Constant (0A)	Mobile Direct Line Model, as defined in [WMC1.0].
7	bInterfaceProtocol	1	Constant (00)	No specific protocol.

Notice that the interface protocol is 00, which indicates “no protocol”.

Following the Mobile Direct Line Model interface descriptor, a number of functional descriptors appear.

6.1.2.2 Communications Class Header Descriptor

This is as described in [USBCDC1.1].

This descriptor is mandatory, and must be first.

6.1.2.3 SAFE Networking MDLM Functional Descriptor

This descriptor is mandatory, and identifies this interface as a SAFE Networking interface. It must appear after the Header descriptor and before the endpoint descriptors.

Table 6-6: SAFE Networking MDLM Functional Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (0x15)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (24h)	CS_INTERFACE
2	bDescriptorSubType	1	Constant (0x12)	Indicates that this is an MDLM functional descriptor.
3	bcdVersion	2	Constant	Version of this SAFE specification used for controlling this device, as a BCD number with the implied decimal point between bits 7 and 8
5	bGuid[16]	16	Bytes	The GUID that identifies SAFE Networking, in network byte order: 0x5d, 0x34, 0xcf, 0x66, 0x11, 0x18, 0x11, 0xd6, 0xa2, 0x1a, 0x00, 0x01, 0x02, 0xca, 0x9a, 0x7f

6.1.2.4 SAFE Networking MDLM Detail Descriptor

This descriptor is mandatory, and must appear after the SAFE Networking MDLM Functional Descriptor.

Table 6-7: SAFE Networking MDLM Detail Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (06h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (24h)	CS_INTERFACE [USBCDC1.1]
2	bDescriptorSubType	1	Constant (0x13)	Indicates that this is an MDLM detail descriptor. [USBWMC1.0]
3	bGuidDescriptorType	1	Constant	SAFE_NET_FUNC_DESC_CODE, as defined in table Table 5 -2.
4	bmNetworkCapabilites	1	Bitmap	Specifies the capabilities of this function. A bit value of zero indicates that the capability is not supported. D7..D1: Reserved (zero) D0: Function can process SET_ETHERNET_PACKET_FILTER requests. If not set, broadcast,

Offset	Field	Size	Value	Description
				directed and multicast packets possibly affecting the host are always passed to the host.
5	bmDataCapabilities	1	Bitmap	Specifies the data-plane capabilities of this function. A value of zero indicates that the capability is not supported. Data plane properties are the same in both directions (IN and OUT). D7..D2: Reserved (zero) D1: Pad PDUs so that last packet of PDU is a multiple of wMaxPacketSize - 1. If this bit is set, D0 must also be set. D0: Device expects host to append Ethernet CRCs to OUT-pipe PDUs; and will append Ethernet CRC to IN-pipe PDUs.

6.1.2.5 Ethernet Networking Functional Descriptor

This descriptor is mandatory for LAN frame facilities. For informative purposes, the definition is repeated from [USBCDC1.1].

Table 6-8: Ethernet Networking Functional Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (24h)	CS_INTERFACE
2	bDescriptorSubtype	1	Constant (0Fh)	Ethernet Networking Functional Descriptor subtype, as defined in Table 25 of [USBCDC1.1]
3	iMacAddress	1	Index	Index of string descriptor giving the Ethernet MAC address for this facility. Must not be zero. The MAC address must be formatted in UNICODE as specified in [USBCDC1.1].
4	bmEthernetStatistics	4	Bitmask	Mask of supported statistics. Stored in little-endian order. If zero, host must calculate its own statistics.
8	wMaxSegmentSize	2	Number	The maximum segment size that the LAN frame facility can support, normally 1514 bytes. Stored in little-endian order
10	wNumberMCFilters	2	Bitmask	Indicates the number of multicast filters supported, as defined by table 41 of [USBCDC1.1]. If zero, host must perform all multicast filtering.
12	bNumberPowerFilters	1	Number	Indicates the number of power filters implemented by the function. If zero, then the device doesn't support power filters for remote wakeup of host.

This specification requires that the Ethernet address specified by the string at iMacAddress be the same no matter which (valid) language code is used with GET_DESCRIPTOR to retrieve it. After conversion, the first three octets of the address must be the OUI assigned by the IEEE to the authority assigning the address. The remaining three octets must be unique to this physical device.

6.1.2.6 Endpoint Descriptors

Two endpoints must be provided.

1. A bulk IN endpoint; this endpoint is used to transfer PDUs from the function to the host. Each PDU is delineated by a short packet.

2. A bulk OUT endpoint, used to transer PDUs from the host to the function.

3. Optionally, an INTERRUPT-IN endpoint, with a wMaxPacketSize of at least 8 bytes. If this feature is to be used, the INTERRUPT IN hardware in the function must be able to transmit packets of any size (from zero to wMaxPacketSize bytes), as required by the protocol layers.

These endpoint descriptors may appear in any order.

6.1.3 Management Elements

The management elements for LAN frame facilities are as defined by [USBCDC1.1], Table 10.

6.1.4 Notifications

The notifications for LAN frame facilities are as defined by [USBCDC1.1], Table 11.

6.1.5 Padding and CRC generation

There are three modes of PDU transportation: raw mode, PDU + CRC, and PDU + padding + CRC. We consider each in turn.

6.1.5.1 Raw Mode

In raw mode, PDUs are transmitted over pipes, delineated by short packets.

Many hosts and devices are unable to process zero-length packets correctly. For this reason, this spec allows either host or device is permitted to append one byte (value 0x00) to a PDU, if the PDU would otherwise be a multiple of wMaxPacketSize for that PDU. Because protocols transported in IEEE 802.3 frames are supposed to be insensitive to PDU extension, this causes no problem in practice. Test procedures that are sensitive to packet length may have to be modified to accommodate this situation, however.

6.1.5.2 PDU + CRC

In PDU+CRC mode (bDataCapabilities[1:0] == 01), PDUs are transmitted over pipes, with appended Ethernet CRC, delineated by short packets.

If the PDU is shorter than 64 bytes, the host shall pad the PDU to 64 bytes, by appending bytes containing 0x00, and then shall append the 4-byte CRC, calculated over the entire PDU including padding.

If the size of the PDU, reduced modulo wMaxPacketSize for the conveying pipe, is exactly wMaxPacketSize-4, then the PDU+CRC message would be exactly a multiple of wMaxPacketSize. This would require sending a ZLP. Since some hosts cannot reliably send a ZLP, padding may be inserted. Padding SHALL be inserted by appending one byte of 0x00 to the PDU **before** the CRC is calculated and appended. (I.e., padding SHALL NOT be appended after the CRC.)

(The reason for requiring padding prior to CRC is to accommodate hardware bridges that require the host to calculate the Ethernet CRC for reliability. By using this algorithm, the hardware bridge can simply transmit the PDU+CRC unit as an Ethernet frame.)

6.1.5.3 PDU + Padding + CRC

Some device silicon is unable to transmit or receive short packets reliably. In this case, the transmitting agent is required to pad all PDUs so that the size of each PDU (modulo wMaxPacketSize) is exactly (wMaxPacketSize – 5). Then the Ethernet CRC shall be calculated over the PDU+Padding, and shall be appended to form a message consisting of PDU+Padding+CRC. This message shall be transmitted over the pipe.

This encoding causes each message to be transferred as a sequence of packets. The packets will have lengths (wMaxPacketSize)* wMaxPacketSize-1.

Please be aware that this may cause a message (PDU+Padding) larger than 1514 bytes to be generated! The receiving code must be careful, after stripping the CRC, to further discard any PDU bytes after the 1514'th byte.

6.2 SAFE Serial Functions

To be supplied. Here are Stuart's notes:

```
Serial

******

The serial protocol simply sends the serial data as a single packet of data.

If CRC is enabled the the maximum amount of serial data that can be sent is
the USB packetsize less two.

If padding is enabled additional NULL bytes (0x00) are appended to the serial
data to bring the length of the sent data to the USB packetsize less two.

The CRC is a 10bit CRC that is computed over the data being sent and a single
additional byte of data containing the number of bytes of valid data. The
first byte of the computed CRC (top two bits) is OR'd into the last byte, the
second byte (bottom eight bits) is appended to the data transfer.
```

7 Device Requests

No additional class-level requests are defined by this specification. Subclass-specific requests are given as defined in section 6.1, “SAFE Networking Model”.

8 Device Descriptors

8.1 Standard USB Descriptors

8.1.1 Device Descriptor

The device descriptor is a standard device descriptor, as specified in Chapter 9 of the USB 2.0 specification. The device class shall be 0x02 (Communication class). The device subclass and protocol codes shall be 0x00. The default pipe maximum packet size may be any value permitted for the speed of the device by [USB2.0].

It is probable that the Device Descriptor fields, idVendor and idProduct will be used by the USB host to select a driver for the device.

8.1.2 Configuration Bundle

The configuration bundle starts with a configuration descriptor, which is as specified in Chapter 9 of the USB 2.0 specification.

There are no Configuration Descriptor requirements imposed by this specification.

9 Network Example

These are the SAFE descriptors for the Sharp IRIS network configuration.

9.1 Network Example Device Descriptor

Table 9-9: Network Example Device Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (12h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (01h)	DEVICE descriptor
2	bcdUSB	2	BCD (00h, 02h)	The USB spec version claimed is 2.0
4	bDeviceClass	1	Constant (02h)	Communication class
5	bDeviceSubClass	1	Constant (00h)	No subclass
6	bDeviceProtocol	1	Constant (00h)	No protocol
7	bMaxPacketSize0	1	Constant (08h)	Control endpoint max packet size is 8 bytes
8	idVendor	2	Constant (0DDh, 04h)	Vendor ID is 04DDh.
10	idProduct	2	Constant (80h, 00h)	Product ID is 0080h
12	bcdDevice	2	BCD (00h, 01h)	Device version is 1.0
14	iManufacturer	1	String Index (01h)	Manufacturer string index is 0x01
15	iProduct	1	String Index (00h)	No product string ID
16	iSerialNumber	1	String Index (02h)	Serial number string index is 0x02
17	bNumConfigurations	1	Constant (01h)	Only one configuration on this device.

9.2 Network Example Configuration Bundle

9.2.1 Configuration Descriptor

Table 9-10: Network Example Configuration Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (09h)	Size of Descriptor in bytes

Offset	Field	Size	Value	Description
1	bDescriptorType	1	Constant (02h)	CONFIGURATION descriptor
2	wTotalLength	2	Constant (????)	Total size of the configuration bundle, in bytes.
4	bNumInterfaces	1	Constant (01h)	Only one interface on this device
5	bConfigurationValue	1	Constant (01h)	Use this value in SET_CONFIG to select this configuration
6	iConfiguration	1	String Index (00h)	No string description for this configuration
7	bmAttributes	1	Bit map (C0h)	D7: reserved, must be set D6: set -> self powered D5: clear -> no remote wakeup D4..D0 reserved
8	MaxPower	1	Number (00h)	Indicates that the device draws no power from Vbus

9.2.2 Interface Descriptor

Table 9-11: Network Example Configuration Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (09h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (04h)	INTERFACE descriptor
2	bInterfaceNumber	1	Constant (00h)	Interface number: this is interface 0.
3	bAlternateSetting	1	Constant (00h)	Alternate setting: this is alt setting 0.
4	bNumEndpoints	1	Constant (03h)	Interface has three endpoints.
5	bInterfaceClass	1	Constant (02h)	Communication class
6	bInterfaceSubClass	1	Constant (0Ah)	MDLM subclass
7	bInterfaceProtocol	1	Constant (00h)	No special protocol
8	iInterface	1	String Index (00h)	No string description for this interface

9.2.3 Communication Class Header Descriptor

Table 9-12: Network Example Communication Class Header Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (05h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (24h)	CS_INTERFACE
2	bDescriptorSubType	1	Constant (0x00)	Indicates that this is a header descriptor.
3	bcdCDC	2	Constant (10h, 01h)	Version of the governing CDC specification is 1.1

9.2.4 Example SAFE Networking MDLM Functional Descriptor

Table 9-13: Example SAFE Networking MDLM Functional Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (0x15)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (24h)	CS_INTERFACE
2	bDescriptorSubType	1	Constant (12h)	Indicates that this is an MDLM functional descriptor.
3	bcdVersion	2	Constant (00h, 01h)	Specification version 1.0
5	bGuid[16]	16	Bytes	The GUID that identifies SAFE Networking, in network byte order: 0x5d, 0x34, 0xcf, 0x66, 0x11, 0x18, 0x11, 0xd6, 0xa2, 0x1a, 0x00, 0x01, 0x02, 0xca, 0x9a, 0x7f

9.2.5 Example SAFE Networking MDLM Detail Descriptor

Table 9-14: Example SAFE Networking MDLM Detail Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (06h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (24h)	CS_INTERFACE [USBCDC1.1]
2	bDescriptorSubType	1	Constant (13h)	Indicates that this is an MDLM detail descriptor. [USBWMC1.0]
3	bGuidDescriptorType	1	Constant (00h)	SAFE_NET_FUNC_DESC_CODE, as defined in table Table 5 -2.
4	bmNetworkCapabilites	1	Bitmap (00h)	Specifies the capabilities of this function. A bit value of zero indicates that the capability is not supported. D7..D1: Reserved (zero)

Offset	Field	Size	Value	Description
				D0: Function can **not** process SET_ETHERNET_PACKET_FILTER requests.
5	bmDataCapabilities	1	Bitmap (03h)	Specifies the data-plane capabilities of this function. D7..D2: Reserved (zero) D1: Pad PDUs so that last packet of PDU is a multiple of wMaxPacketSize - 1. D0: Device expects host to append Ethernet CRCs to OUT-pipe PDUs; and will append Ethernet CRC to IN-pipe PDUs.

9.2.6 Example Ethernet Networking Functional Descriptor

Table 9-15: Example Ethernet Networking Functional Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (0Dh)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (24h)	CS_INTERFACE
2	bDescriptorSubtype	1	Constant (0Fh)	Ethernet Networking Functional Descriptor subtype, as defined in Table 25 of [USBCDC1.1]
3	iMacAddress	1	Index (02h)	Index of string descriptor giving the Ethernet MAC address for this facility.
4	bmEthernetStatistics	4	Bitmask (00h, 00h)	No statistics are supported.
8	wMaxSegmentSize	2	Number (EAh, 05h)	The maximum segment size that the LAN frame facility can support, 1514 bytes. Note that this doesn't include the CRC or padding!
10	wNumberMCFilters	2	Bitmask (00h, 00h)	No multicast filters are supported by the function.
12	bNumberPowerFilters	1	Number (00h)	No power filters are supported by the function.

9.2.7 Network Example Bulk Out Endpoint Descriptor

Table 9-16: Network Example Bulk Out Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (07h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (05h)	ENDPOINT descriptor
2	bEndpointAddress	1	Endpoint (01h)	This is ENDPOINT 1 OUT

Offset	Field	Size	Value	Description
3	bmAttributes	1	Bitmap (02h)	Bulk endpoint
4	wMaxPacketSize	2	Number (20h, 00h)	Max Packet Size is 32 bytes
6	bInterval	1	Number (00h)	Reserved for Bulk endpoints, must be zero.

9.2.8 Network Example Bulk In Endpoint Descriptor

Table 9-17: Network Example Bulk In Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (07h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (05h)	ENDPOINT descriptor
2	bEndpointAddress	1	Endpoint (82h)	This is ENDPOINT 2 IN
3	bmAttributes	1	Bitmap (02h)	Bulk endpoint
4	wMaxPacketSize	2	Number (20h, 00h)	Max Packet Size is 32 bytes
6	bInterval	1	Number (00h)	Reserved for Bulk endpoints, must be zero.

9.2.9 Network Example Interrupt In Endpoint Descriptor

Table 9-18: Network Example Interrupt In Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bFunctionLength	1	Number (07h)	Size of Descriptor in bytes
1	bDescriptorType	1	Constant (05h)	ENDPOINT descriptor
2	bEndpointAddress	1	Endpoint (83h)	This is ENDPOINT 3 IN
3	bmAttributes	1	Bitmap (03h)	Interrupt endpoint
4	wMaxPacketSize	2	Number (10h, 00h)	Max Packet Size is 16 bytes
6	bInterval	1	Number (00h)	Reserved for Bulk endpoints, must be zero.