USB-Ethernet Preliminary Software Spec

Revision History:

0.0 First pass
0.1 7/23/98
Corrected errors in description of vendor commands
Added three vendor commands
Added Device and Configuration Descriptors
0.2 Updated vendor commands
0.3 Added language clarifying WORD format in descriptors.

1. Overview

The USB-Ethernet adapter provides an interface between the USB port on a PC and an Ethernet network.

The dafault Vendor ID for the adapter is 0x03e8, and the default product ID is 0x0008

Three endpoints are configured in the following manner

Endpoint 0 is used for standard commands, as well as for Ethernet-specific commands and requests.
Endpoint 1 is used to by the USB-Ethernet adapter to send data down to the host.
Endpoint 2 is used to carry data from the host to the USB-Ethernet adapter.v
Endpoint 3 is not currently used
Subsequent sections of this document will describe:

- Device, configuration and endpoint descriptors.
- Vendor-specific commands
- Data Formats
- API Calls

Please note that references to the “USB Spec” refer to Revision 1.0 of the Universal Serial Bus Specification.

2. Device and Configuration Descriptors

The USB-Ethernet adapter implements standard device and configuration descriptors in accordance with Chapter 9 of the USB spec. This section describes briefly the details specific to this adapter.
Note: All of the WORD-sized (16-bit) fields are stored in little-endian format for compatibility with Intel architecture.

2.1 Device Descriptor

Table 1: Device Descriptor

Offset	Field	Size	Value	Description
0	Blength	1	18	Size of this descriptor in bytes
1	BdescriptorType	1	1	DEVICE descriptor type
2	BcdUSB	2	0x100	BCD-encoded USB revision number
4	BdeviceClass	1	0	Class code: Please refer to USB specification for interpretation.
5	BdeviceSubClass	1	0	SubClass code: Please refer to USB specification for interpretation.
6	BdeviceProtocol	1	0	Protocol code: : Please refer to USB specification for interpretation.
7	bMaxPacketSize0	1	8	Maximum packet size for Endpoint 0
8	IdVendor	2	????	Vendor ID (Manufacturer specific)
10	IdProduct	2	????	Product ID (Manufacturer specific)
12	BcdDevice	2	0x0002	Device release number
14	Imanufacturer	1	2	Index of manufacturer string descriptor
15	Iproduct	1	3	Index of product string descriptor
16	IserialNumber	1	1	This string is a text representation of the MAC address
17	BnumConfigurations	1	1	Number of configurations

2.2 Configuration Descriptor

Note that a SET_CONFIGURATION command will also cause the MAC to be reset.

Table 2: Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	9	Size of this descriptor in bytes
1	bDescriptorType	1	2	CONFIGURATION descriptor type
2	wTotalLength	2	39	Total length of data returned for this configuration
4	bNumInterfaces	1	1	Number of interfaces supported by this configuration.
5	bConfigurationValue	1	1	Use this value in Set Configuration to select this configuration.
6	iConfiguration	1	0	This string descriptor not implemented
7	bmAttributes	1	0x80	Device is bus powered
8	MaxPower	1	250	Maximum power consumption of device in 2-mA units.

2.3 Interface Descriptor

Table 3: Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	9	Size of this descriptor in bytes
1	bDescriptorType	1	4	INTERFACE descriptor type
2	bInterfaceNumber	1	0	Number of this interface
3	bAlternateSetting	1	0	Alternate setting for this interface
4	bNumEndpoints	1	3	Number of endpoints used by this interface
5	bInterfaceClass	1	0	Class code: for interpretation, refer to USB spec.
6	bInterfaceSubClass	1	0	SubClass code: for interpretation, refer to USB spec.
7	bInterfaceProtocol	1	0	Protocol code: for interpretation, refer to USB spec.
8	iInterface	1	0	This string descriptor not implemented

2.4 Endpoint Descriptors

2.4.1 Endpoint 1 Descriptor

Table 4: Endpoint 1 Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	7	Size of this descriptor in bytes
1	bDescriptorType	1	5	ENDPOINT descriptor type
2	bEndpointAddress	1	0x81	Endpoint #1, IN Endpoint
3	bmAttributes	1	2	Transfer mode is BULK
4	wMaxPacketSize	2	64	Maximum packet size for this endpoint
6	bInterval	1	0	Ignored for BULK endpoints.

2.4.2 Endoint 2 Descriptor

Table 5: Endpoint 2 Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	7	Size of this descriptor in bytes
1	bDescriptorType	1	5	ENDPOINT descriptor type
2	bEndpointAddress	1	0x02	Endpoint #2, OUT Endpoint
3	bmAttributes	1	2	Transfer mode is BULK
4	wMaxPacketSize	2	64	Maximum packet size for this endpoint
6	bInterval	1	0	Ignored for BULK endpoints.

2.4.3 Endpoint 3 Descriptor

Table 6: Endpoint 3 Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	7	Size of this descriptor in bytes
1	bDescriptorType	1	5	ENDPOINT descriptor type
2	bEndpointAddress	1	0x83	Endpoint #3, IN Endpoint
3	bmAttributes	1	3	Transfer mode is INTERRUPT
4	wMaxPacketSize	2	8	Maximum packet size for this endpoint
6	bInterval	1	1	Interval for polling endpoint for data transfers.

3. Vendor Commands

This section defines the vendor commands for the USB-Ethernet Adapter.
The table below summarizes these commands, and subsequent sections define selected commands in detail.

Table 7: Vendor Commands

Command	Value
GET_ETHERNET_DESCRIPTOR	00H
SET_ETHERNET_MULTICAST_FILTERS	01H
SET_ETHERNET_PACKET_FILTER	02H
GET_ETHERNET_STATISTICS	03H
GET_AUX_INPUTS	04H
SET_AUX_OUTPUTS	05H
SET_TEMP_MAC	06H
GET_TEMP_MAC	07H
SET_URB_SIZE	08H
SET_SOFS_TO_WAIT	09H
SET_EVEN_PACKETS	0AH
RESERVED (future use)	0BH-FEH
SCAN	0FFH

3.1 GetEthernetDescriptor

This Ethernet Networking functional descriptor describes those operational modes supported by the KLSI USB-Ethernet adapter.

Table 8: Ethernet Functional Descriptor

Offset	Field	Size	Value	Description
0	*bFunctionLength*	1	Number (18)	Size of this functional descriptor, in bytes.
1	*Reserved*	1	Number	
2	*Reserved*	1	Number	
3	*bMACAddress*	6	Number	Contains the 48 bit Ethernet MAC address (encoded using network byte order).
9	*bmEthernetStatistics*	4	Bitmap	Indicates which Ethernet statistics functions the device collects. If a bit is set to 0, the host network driver is expected to keep count for the corresponding statistic (if able). See Table 9 for a detailed listing of possible Ethernet statistics. Support for any of these statistics is optional. If none of these bits are set, the device does not support the GetEthernetStatistics request.
13	*wMaxSegmentSize*	2	Number (1514)	The maximum segment size that the Ethernet device is capable of supporting.
15	*wNumberMCFilters*	2	Number (128)	Contains the number of multicast filters that can be configured by the host. D15: 0 - The device performs perfect multicast address filtering (no hashing). 1- The device uses imperfect multicast address filtering (hashing). Here, the host software driver must perform further qualification itself to achieve perfect filtering. D14..0: Indicates the number of multicast address filters supported by the device (0 to 32767). If the host finds the number of filters supported by the device to be inadequate, it may choose to set the device's Ethernet Packet Filter to forward all multicast frames to the host, performing all multicast filtering in software instead. If this value is 0, the device does not support the SetEthernetMulticastFilters request.

Offset	Field	Size	Value	Description
17	*Reserved*	1	0	

Note: where a value is given in parentheses in a “Value” column, this number is the value assigned in the current implementation

Table 9 assigns meanings to the bits for the Ethernet Statistics bitmap field.

Note: Not all statistics capabilities will be available in this implementation.

Table 9: Ethernet Statistics Capabilities

Offset	Field	Description
D0	XMIT_OK	Frames transmitted without errors
D1	RVC_OK	Frames received without errors
D2	XMIT_ERROR	Frames not transmitted, or transmitted with errors
D3	RCV_ERROR	Frames received with errors that are not delivered to the USB host..
D4	RCV_NO_BUFFER	Frame missed, no buffers
D5	DIRECTED_BYTES_XMIT	Directed bytes transmitted without errors
D6	DIRECTED_FRAMES_XMIT	Directed frames transmitted without errors
D7	MULTICAST_BYTES_XMIT	Multicast bytes transmitted without errors
D8	MULTICAST_FRAMES_XMIT	Multicast frames transmitted without errors
D9	BROADCAST_BYTES_XMIT	Broadcast bytes transmitted without errors
D10	BROADCAST_FRAMES_XMIT	Broadcast frames transmitted without errors
D11	DIRECTED_BYTES_RCV	Directed bytes received without errors
D12	DIRECTED_FRAMES_RCV	Directed frames received without errors
D13	MULTICAST_BYTES_RCV	Multicast bytes received without errors
D14	MULTICAST_FRAMES_RCV	Multicast frames received without errors
D15	BROADCAST_BYTES_RCV	Broadcast bytes received without errors
D16	BROADCAST_FRAMES_RCV	Broadcast frames received without errors
D17	RCV_CRC_ERROR	Frames received with circular redundancy check (CRC) or frame check sequence (FCS) error
D18	TRANSMIT_QUEUE_LENGTH	Length of transmit queue
D19	RCV_ERROR_ALIGNMENT	Frames received with alignment error
D20	XMIT_ONE_COLLISION	Frames transmitted with one collision
D21	XMIT_MORE_COLLISIONS	Frames transmitted with more than one collision
D22	XMIT_DEFERRED	Frames transmitted after deferral
D23	XMIT_MAX_COLLISIONS	Frames not transmitted due to collisions
D24	RCV_OVERRUN	Frames not received due to overrun
D25	XMIT_UNDERRUN	Frames not transmitted due to underrun
D26	XMIT_HEARTBEAT_FAILURE	Frames transmitted with heartbeat failure
D27	XMIT_TIMES_CRS_LOST	Times carrier sense signal lost during transmission
D28	XMIT_LATE_COLLISIONS	Late collisions detected
D29-D31	RESERVED	Must be set to zero

3.2 SetEthernetMulticastFilters

This request sets the Ethernet device multicast filters as specified in the sequential list of 48 bit addresses Ethernet multicast addresses . Note that if the host wishes to change a single multicast filter in the device, it must reprogram the entire list of filters using this request

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000000B	SET_ETHERNET_ MULTICAST_ FILTERS	Number of filters (N)	0	N * 6	A list of N 48 bit Multicast addresses, in network byte order

3.3 SetEthernetPacketFilter

This request is used to configure device Ethernet packet filter settings. The Packet Filter is the inclusive OR of the bitmap shown in Table 10

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000000B	SET_ETHERNET_PACKET_ FILTER	Packet Filter Bitmap	0	Zero	None

Table 10 : Ethernet Packet Filter Bitmap

Bit position	Description
D15..D5	RESERVED (Reset to zero)
D4	PACKET_TYPE_MULTICAST 1: All multicast packets enumerated in the device's multicast address list are forwarded up to the host. (required) 0: Disabled. The ability to disable forwarding of these multicast packets is optional. ***
D3	PACKET_TYPE_BROADCAST 1: All broadcast packets received by the networking device are forwarded up to the host. (required) 0: Disabled. The ability to disable forwarding of broadcast packets is optional. ***
D2	PACKET_TYPE_DIRECTED 1: Directed packets received containing a destination address equal to the MAC address of the networking device are forwarded up to the host (required) 0: Disabled. The ability to disable forwarding of directed packets is optional. ***
D1	PACKET_TYPE_ALL_MULTICAST 1: ALL multicast frames received by the networking device are forwarded up to the host, not just the ones enumerated in the device's multicast address list (required) 0: Disabled.
D0	PACKET_TYPE_PROMISCUOUS: 1: ALL frames received by the networking device are forwarded up to the host (required) 0: Disabled.

3.4 GetEthernetStatistics

This request is used to retrieve the statistics based on the feature selector. The value returned indicates the number of frames of matching with the specified statistic that have occurred since the device has been powered on. This number is a 32 bit unsigned integer, which is incremented at each occurrence, and will be wrapped to 0 if reaching the maximum value.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
11000000B	GET_ETHERNET_ STATISTICS	Feature Selector	0	4	32 bit unsigned integer

Table 11: Ethernet Statistics Feature Selector Codes

Feature selector	Code	Targets	Length of Data	Description
RESERVED	00h	None	None	Reserved for future use
XMIT_OK	1	Interface or Endpoint	4	Frames transmitted without errors
RVC_OK	2	Interface or Endpoint	4	Frames received without errors
XMIT_ERROR	3	Interface or Endpoint	4	Frames not transmitted, or transmitted with errors
RCV_ERROR	4	Interface or Endpoint	4	Frames received with errors
RCV_NO_BUFFER	5	Interface or Endpoint	4	Frames missed, no buffers
DIRECTED_BYTES_XMIT	6	Interface or Endpoint	4	Directed bytes transmitted without errors
DIRECTED_FRAMES_XMIT	7	Interface or Endpoint	4	Directed frames transmitted without errors
MULTICAST_BYTES_XMIT	8	Interface or Endpoint	4	Multicast bytes transmitted without errors
MULTICAST_FRAMES_XMIT	9	Interface or Endpoint	4	Multicast frames transmitted without errors
BROADCAST_BYTES_XMIT	10	Interface or Endpoint	4	Broadcast bytes transmitted without errors

BROADCAST_FRAMES_XMIT	11	Interface or Endpoint	4	Broadcast frames transmitted without errors
DIRECTED_BYTES_RCV	12	Interface or Endpoint	4	Directed bytes received without errors
DIRECTED_FRAMES_RCV	13	Interface or Endpoint	4	Directed frames received without errors
MULTICAST_BYTES_RCV	14	Interface or Endpoint	4	Multicast bytes received without errors
MULTICAST_FRAMES_RCV	15	Interface or Endpoint	4	Multicast frames received without errors
BROADCAST_BYTES_RCV	16	Interface or Endpoint	4	Broadcast bytes received without errors
BROADCAST_FRAMES_RCV	17	Interface or Endpoint	4	Broadcast frames received without errors
RCV_CRC_ERROR	18	Interface or Endpoint	4	Frames received with circular redundancy check (CRC) or frame check sequence (FCS) error
TRANSMIT_QUEUE_LENGTH	19	Interface or Endpoint	4	Length of transmit queue
RCV_ERROR_ALIGNMENT	20	Interface or Endpoint	4	Frames received with alignment error
XMIT_ONE_COLLISION	21	Interface or Endpoint	4	Frames transmitted with one collision
XMIT_MORE_COLLISIONS	22	Interface or Endpoint	4	Frames transmitted with more than one collision
XMIT_DEFERRED	23	Interface or Endpoint	4	Frames transmitted after deferral
XMIT_MAX_COLLISIONS	24	Interface or Endpoint	4	Frames not transmitted due to collisions
RCV_OVERRUN	25	Interface or Endpoint	4	Frames not received due to overrun
XMIT_UNDERRUN	26	Interface or Endpoint	4	Frames not transmitted due to underrun
XMIT_HEARTBEAT_FAILURE	27	Interface or Endpoint	4	Frames transmitted with heartbeat failure

XMIT_TIMES_CRS_LOST	28	Interface or Endpoint	4	Times carrier sense signal lost during transmission
XMIT_LATE_COLLISIONS	29	Interface or Endpoint	4	Late collisions detected

3.5 GetAuxInputs

Reads four auxiliary input pins from the USB-Ethernet Controller chip.

bmRequestType	bRequest	Wvalue	wIndex	wLength	Data
11000000B	GET_AUX_INPUTS	None	0	1	The four LS bits represent the state of the pins.

3.6 SetAuxInputs

Sets four auxiliary output pins on the USB-Ethernet Controller chip.

bmRequestType	bRequest	Wvalue	wIndex	wLength	Data
01000000B	SET_AUX_OUTPUTS	The four LS bits represent the state to which the pins are to be set.	0	Zero	None

3.7 GetMacAddress

Obtains the MAC Address currently used by the Ethernet adapter. If a SetMacAddress command has not been issued, the address returned will be whatever is in the I2C EEPROM on the device.

BmRequestType	bRequest	Wvalue	wIndex	wLength	Data
11000000B	GET_TEMP_MAC	0	0	6	MAC Address

3.8 SetMacAddress

Sets the MAC address to be used by the Ethernet adapter. Does *not* alter the default MAC address set in the I2C EEPROM

BmRequestType	bRequest	Wvalue	wIndex	wLength	Data
01000000B	SET_TEMP_MAC	0.	0	6	MAC address

3.9 SetURBSize

Sets the size of the USB Request Block to be used by the Ethernet adapter. To increase data throughput in the system, the USB Ethernet adapter does not terminate every Ethernet packet on USB with a Zero Length Packet (ZLP). Ethernet packets are sent down the USB cable as they are received but the USB driver does not notice them until the current URB is fulfilled. The URB is fulfilled when a ZLP is sent on USB. Setting the URB size in the Ethernet adapter allows the adapter to know when to send a ZLP.

bmRequestType	bRequest	WValue	wIndex	WLength	Data
01000000B	SET_URB_SIZE	URB size.	0	0	None

3.10 SetSOFsToWait

Sets the number of Start Of Frames to wait while filling a URB before sending a ZLP. See above.

bmRequestType	BRequest	wValue	wIndex	wLength	Data
01000000B	SET_SOFS_TO_WAIT	SOFs.	0	0	None

3.11 SetEvenPackets

Microsoft's UHCI driver code in Windows 95 has a bug in keeping track of the Data Toggle bit. This vendor command causes all Ethernet packets sent on USB to be padded out to have an even number of USB packets to get around this bug.

BmRequestType	bRequest	wValue	wIndex	wLength	Data
01000000B	SET_EVEN_PACKETS	0 to turn off. 1 to turn on..	0	0	None

3.12 Send Scan Data

This request can be used to modify the I2C EEPROM on the device at a specified address, or to reset the device.

This mechanism is documented in detail in the QT ROM BIOS specification, in the section on Int 67.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
01000000B	SCAN	0	0	Variable	See below

3.12.1 To modify the contents of EEPROM:

Note that the wLength value is the total length of this data structure:

Data	
0xC3B6	Signature (used by ROM BIOS)
0x*nnnn*	Length of data
0x08	Type byte (specifies write to an interrupt)
0x40	Specified small I2C interface (up to 24lc16)
0x*nnnn*	I2C Address to which data is to be written
nnnnnnnnnn	Data to be written

Because an address must be supplied, the host program needs to know at what address in the I2C it wants to write.
If the data changes size (such as a string change) the host program modifying the data must know the structure of data in the I2C. It is assumed that, since the host program programmed the I2C in the first place, it will know this structure.

3.12.2 To reset the device:

Resetting the QT processor can be accomplished by forcing a jump to the adress 0xFFF0:

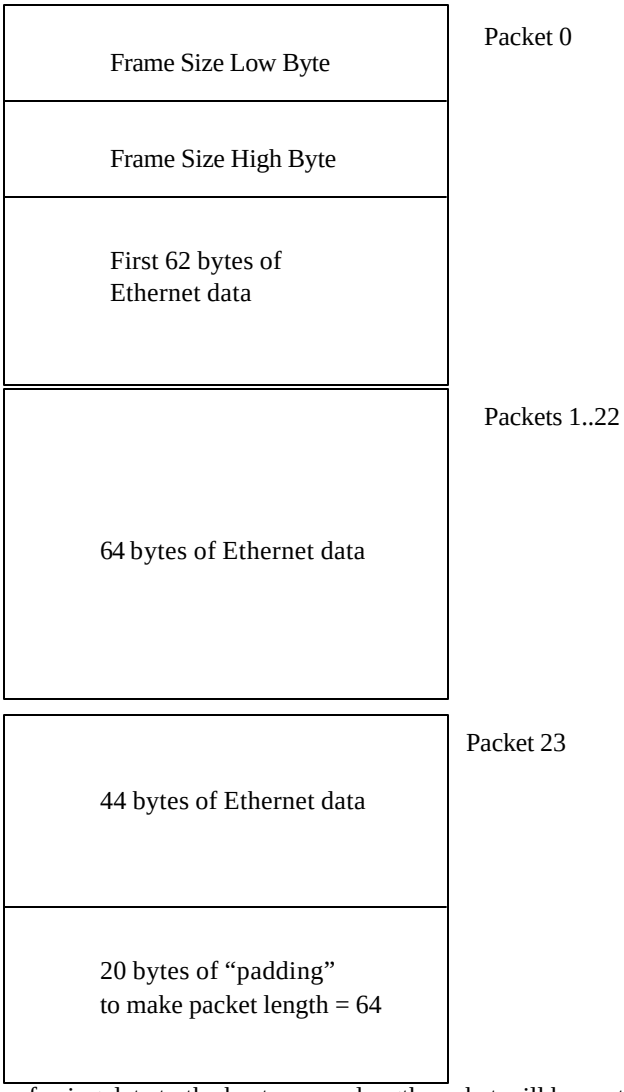
Data	
0xC3B6	Signature (used by ROM BIOS)
0x02	Length of data
0x04	Type byte (specifies jump to address)
0xFFF0	Address to which to jump to reset

4. Data Formats

4.1 USB Packet Format

This section defines the mapping of Ethernet frames onto USB packets.
Each Ethernet frame is transferred to-from the host as a series of up to 24 64-byte USB packets
Packet 0 contains two bytes which give the length (in bytes) of the entire frame, and then the first 62 bytes of Ethernet data. Subsequent packets contain Ethernet data, and the final packet is padded to make its length 64 bytes.
For example, a full 1514-byte Ethernet frame is transferred as 24 USB packets in the following manner:

Figure 1: USB Packet Format



When the USBMAC is transferring data to the host, a zero-length packet will be sent when there are no more Ethernet packets to be sent. (see SET_URB_SIZE and SET_SOFS_TO_WAIT).

Note: The Preamble, Start Frame Delimiter, and CRC fields of the Ethernet frame are not sent down to the host. If a CRC error is detected, none of the frame is sent to the host.