Universal Serial Bus
IrDA Bridge
Device Definition

AMP Incorporated
SystemSoft Corporation

Revision 0.9b
December 7, 1998

Revision History

Rev	Date	Filename	Comments
0.9b	Dec 7, 1998	USBIR09b.DOC	Permit Multiple Configurations. Change direction bit in 6.2.5, and changed 5 to 6 BOFs in 5.4.2.2 and 7.2 tables. Removed Outstanding Issues paragraph 8.1.
0.9a	Jul 14, 1998	USBIR09a.DOC	Clarification of Inbound & Outbound Header functionality. Section 7.2, modifications to field names & minimum bLength changed to 12. General text clarifications. Outstanding issues added to Section 8.
0.9	Jun 30, 1998	USBIR09.DOC	Modify fields to use IrDA formats. Add header on outbound messages.
0.8	Jan 29, 1998	USBIR08.DOC	Incorporate comments
0.7c	Jan 6, 1998	USBIR07c.DOC	Specify that USB device performs IrLAP frame wrapper function
0.7b	Dec 4, 1997	USBIR07b.DOC	Incorporate comments, add Class-Specific Requests and Descriptors
0.7a	Oct 23, 1997	USBIR07a.DOC	Initial draft

Please send comments via electronic mail to: ramsland@amp.com

USB IrDA Bridge Device Definition

INTELLECTUAL PROPERTY DISCLAIMER

Contributors

A. Bruce Ramsland	AMP Incorporated
David G. Lawrence	SystemSoft Corporation
Homn Lin	KC Technology
Matt Katagiri	NEC
Mike Kienzle	IBM
Paul E. Berg	SystemSoft Corporation
Vanessa Hutchinson	Extended Systems Incorporated
Wayne Alvarez	Sigmatel

Table of Contents

1. Introduction

1.1. Purpose

This document specifies the behavior of a USB device containing an IrDA transceiver.

1.2. Scope

This document is intended to provide enough information for the following:

- To allow IHVs to build compliant device containing an IrDA transceiver
- To allow software developers to create a device driver capable of connecting any compliant USB IrDA Bridge device to host-based IrDA software layers.

1.3. Related Documents

Compaq, Digital Equipment Corporation, IBM PC Company, Intel, Microsoft, NEC, Northern Telecom, *USB Specification*, Version 1.0, January 19, 1996, www.usb.org

IBM Corporation, Hewlett-Packard Company, Apple Computer, Inc., Counterpoint Systems Foundry, Inc., *Serial Infrared Link Access Protocol (IrLAP)*, Version 1.1, June 16, 1996, Infrared Data Association, www.irda.org

1.4. Terms and Abbreviations

Term	Description
Configuration	A collection of one or more interfaces that may be selected on a USB device
Descriptor	Data structure used to describe a USB device capability or characteristic
Device	A USB peripheral
Driver	Host software that connects other drivers, DLLs or applications to USBDI.
Endpoint	Source or sink of data on a USB device
HCD	Acronym for Host Controller Driver, the Driver used to manage a host controller
HCDI	Acronym for HCD Interface, the programming interface used by USBD to interact with HCD
Host	A computer system where a Host Controller is installed
Host Controller	Hardware that connects a Host to USB

Term	Description
Host Software	Generic term for a collection of drivers, DLLs and/or applications that provide operating system support for a Device
IHV	Acronym for Independent Hardware Developer
Interface	Collection of zero or more endpoints that present functionality to a host
OHCI	Acronym for Open Host Controller Interface, a hardware register specification defined by Compaq and Microsoft for a Host Controller
UHCI	Acronym for Universal Host Controller Interface, a hardware register specification defined by Intel for a Host Controller
USB	Acronym for Universal Serial Bus, a bus used to connect devices to a host
USBD	Acronym for Universal Serial Bus Driver, the Driver used to manage and use Devices among multiple Device Drivers
USBDI	Acronym for USBD Interface, the USBD programming interface

2. Management Overview

This document specifies a USB IrDA Bridge device. Such devices are expected to replace or supplement motherboard based transceiver implementations. It should be possible to support any existing IrDA-compliant device that works with a motherboard based transceiver with this USB device, providing the appropriate software is installed on the host.

The USB device exchanges IrLAP frames between the host and the IrDA device. On the host, a USB driver routes the IrLAP frames to an IrLAP software layer. IrDA applications should not be able to detect that the IrDA device is connected using USB.

The USB driver strips header information added by the USB device to IrDA device supplied frames before delivering it to the host.

The device identifies itself as an application-specific USB device. All application-specific USB devices use the same class code [TBD]. The USB-IF maintains/assigns each application-specific USB device a unique sub-class code. Application-specific USB device specifications assign protocol codes as they require.

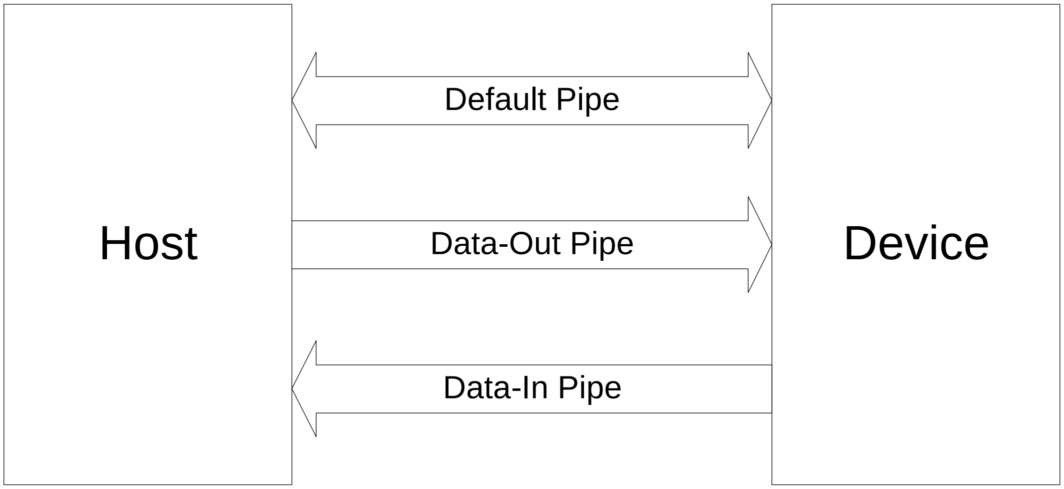

Figure 1 - Device Configuration

The Device uses a single configuration with a single interface explicitly defining two endpoints. (The default endpoint is included implicitly.) Data is exchanged between the Host and the Device through two data endpoints (one in and one out).

Section 3, Assumptions and Constraints, describes factors that affect the overall design of the Device.

Section 4, Functional Overview, presents an overview of the Device.

Section 5, Functional Characteristics, describes the physical layout of the Device.

Section 6, Device Requests, describes each of the Device Requests supported by the Device.

Section 7, Descriptors, describes each of the Descriptors required.

3. Assumptions and Constraints

3.1. Compliance

This section describes assumptions and constraints related to compliance.

3.1.1. USB Specification

The Device shall be compliant with the USB Specification.

3.1.2. Device Framework

The Device shall support standard and device-specific requests as described in the USB Specification and in the proposed Common Class Specification

3.1.3. USB Application-Specific Device Class

The Device shall support the proposed USB Application-Specific Device Class.

4. Functional Overview

Traditional IrDA host systems use a motherboard mounted IrDA transceiver to exchange data with an IrDA device. Communications between the host and device are half duplex. The IrDA device and the host negotiate connection establishing communication parameters. These include items such as baud rate and turn-around time.

The transmission medium uses a communications protocol to provide reliable data exchange. The IrDA Serial Link Access Protocol (IrLAP) exchanges data between the host and an IrDA device in units called frames. These IrLAP frames vary in size from two bytes to over two Kbytes. All IrLAP frames contain address and control bytes and optional information (data). The address, control and optional information field portions of an IrLAP frame are sometimes called the IrLAP Payload. The IrLAP Payload does not include any beginning of frame flags, the end of frame flag, or the frame check sequence fields. The process of adding these fields to an IrLAP Payload prior to transmission and removing them from an IrLAP frame on receipt is performed by an IrLAP Frame Wrapper. All USB-based IrDA Bridge Devices perform the IrLAP Frame Wrapper processing in the Device. The only IrLAP frame data carried by USB is the IrLAP Payload portion of the IrLAP frame.

USB uses the term frame to represent 1 ms of bus time. This is different from an IrLAP frame and simply an unfortunate re-use of the term frame. To avoid confusion, this document uses the terms IrLAP frame and USB frame to distinguish between the two uses of the term frame.

USB exchanges data between a host and a USB device in units called packets. These packets are combined into transactions and transactions are combined into data transfers. USB supports four types of data transfers: control, bulk, interrupt and isochronous. The different types of data transfers have different characteristics. Control transfers are bi-directional and allow the exchange of messages between host and device. All other data transfers are unidirectional and exchange data as a stream of bytes.

USB-based IrDA Bridge devices use bulk transfers to exchange data between a host and the USB device. Bulk transfers guarantee data integrity, but do not provide any guarantee of throughput. Bulk transfers are made on a bandwidth-available basis. The maximum packet size of a USB bulk transfer is 8, 16, 32 or 64 bytes. An endpoint (the source or destination for data on a USB device) has a fixed maximum packet size for a particular configuration and/or alternate setting for the interface which contains the endpoint.

While each data packet of a bulk endpoint is limited to the maximum packet size defined in the associated endpoint descriptor, it should be noted that a host might request multiple bulk packets within a single USB frame. For maximum throughput, a USB IrDA Bridge device must be prepared to transfer multiple bulk packets within a single USB frame.

Since the native transfer size of IrLAP frames may exceed the maximum packet size of a USB bulk endpoint, it is necessary to delineate the beginning and ending of IrLAP frames within the data stream delivered by a bulk endpoint. This positive delineation of IrLAP frames is done using a USB short packet mechanism. When an IrLAP frame spans *N* USB packets, the first packet through packet *N-1* shall be the maximum packet size defined for the USB endpoint. If the *N*th packet is less than maximum packet size the USB transfer of this short packet will identify the end of the IrLAP frame. If the *N*th packet is exactly maximum packet size, it shall be followed by a zero-length packet; the zero length packet identifies the end of the IrLAP frame.

The USB device adds a header to IrLAP frames received from the IrDA device for delivery to the host. This requires the USB IrDA Bridge device buffer at least one IrLAP frame so the header can be prepared prior to transmitting the frame to the host. The header is stripped before it is delivered to the ultimate consumer of the IrLAP frame, the client of the USB driver on the host.

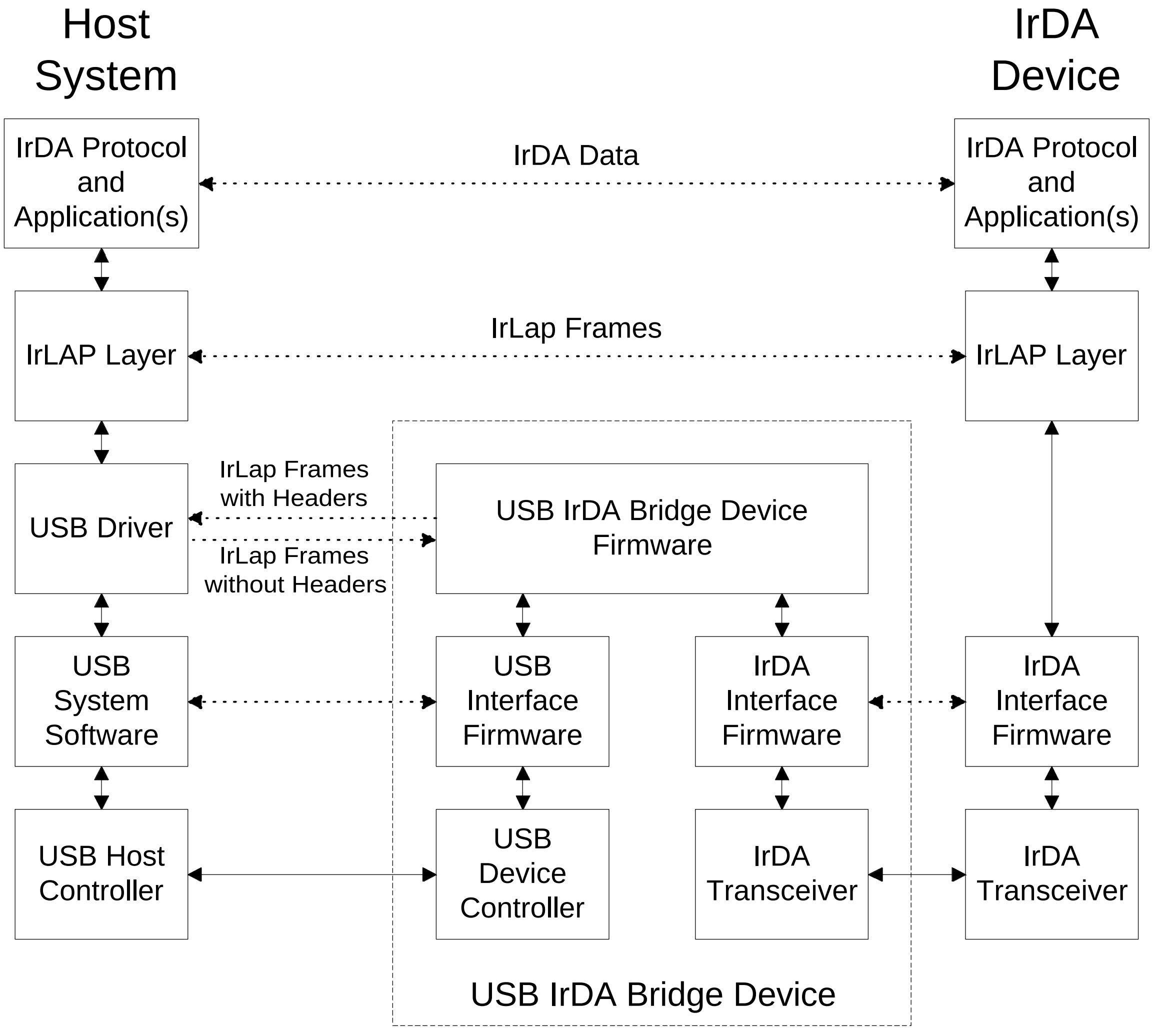

Figure 2 - USB IrDA Bridge Device Example

A USB IrDA Bridge device uses two bulk endpoints (one in and one out) to exchange IrLAP frames between the host and an IrDA device. Thus, the half duplex IrLAP protocol is actually transmitted over what appears to be a full duplex communications channel. Due to this fact, the USB IrDA Bridge device optimizes turn-around time between itself and the IrDA device. IrLAP frames are exchanged based on USB bus availability rather than forcing the USB driver client to track IrDA device turn-around time.

In order to support data integrity, reception of IrLAP frames must be acknowledged by the receiver. To optimize the use of the half-duplex IrDA medium, the protocol defines a concept of window size. Rather than requiring acknowledgement of each IrLAP frame before the next frame is sent, window size is the number of IrLAP frames a transmitter can send without an acknowledgement of an earlier frame.

The USB IrDA Bridge device communicates implementation choices to the USB driver using class specific descriptors. These descriptors describe USB IrDA Bridge device characteristics like maximum possible window size and supported transceiver baud rates. The host software adapts its operation based on the information conveyed by the descriptors.

5. Functional Characteristics

5.1. Configuration

The Device shall support a single configuration containing a single interface. The device class code, device sub-class code and device protocol code in the Device Descriptor shall all be set to zero.

5.2. Interface

The single interface within the single configuration shall contain two endpoints: data-in and data-out. As with all USB devices, the default endpoint shall be included in the interface by implication.

The interface descriptor shall use a device class code of application specific ([TBD]), a sub-class code of [TBD] and a protocol code of zero (0x00).

5.3. Endpoints

The device shall contain three endpoints: default, Data-In and Data-Out.

5.3.1. Default

The default endpoint uses control transfers as defined in the USB Specification. The default endpoint is used to send standard and application-specific requests to the device, the interface and/or an endpoint. The endpoint number must be zero (0).

5.3.2. Data-In

The Data-In endpoint shall be used to receive data from the device intended for delivery to a communications application on the host. The Data-In endpoint shall use bulk transfers. The endpoint number may be any value between one (1) and fifteen (15) that is not used by another endpoint on the device. The direction shall be IN. The maximum packet size is implementation specific.

5.3.3. Data-Out

The Data-Out endpoint shall be used to send data to the device from a communications application on the host. The Data-Out endpoint shall use bulk transfers. The endpoint number may be any value between one (1) and fifteen (15) that is not used by another endpoint on the device. The direction shall be OUT. The maximum packet size is implementation specific.

5.4. Data Format

The USB IrDA Bridge device shall exchange complete IrLAP frames between itself and the communicating (remote) IrDA device. As noted in the Functional Overview section, the IrLAP frames exchanged between the USB IrDA Host driver and the USB IrDA Bridge device are actually the IrLAP Payload portions of the IrLAP frame. Only the unwrapped IrLAP frame consisting of the address, control, and optional information fields are sent over USB.

XBOFs or STA or 16 PA	BOF or STA	IrLAP Payload	FCS	EOF or STO

A	C	I

5.4.1. Inbound IrLAP Frame Delivery

5.4.1.1 IrDA device to USB IrDA Bridge device

The USB IrDA Bridge device shall receive IrLAP frames from a communicating IrDA device and extract the IrLAP Payload portion. Only valid IrLAP frames will be used by the USB IrDA Bridge device. An IrLAP frame is invalid when it is incomplete, fails frame check sequence (FCS) validation, or contains an abort sequence. Invalid frames will be abandoned.

5.4.1.2 Inbound Header

Inbound Header	IrLAP Payload

The USB IrDA Bridge device shall add an Inbound Header to the extracted IrDA Payload. This header identifies the status of Media_Busy and Link_Speed when requested.

The format for the Inbound Header is:

Field	Size	Description
bmStatus	1	Status indicators D7 Media_Busy (1 bit value) 0 = ignore 1 = traffic detected D6..4 Reserved – set to 0 by the device, ignored by the host D3..0 Link_Speed (4 bit value) 0 = speed ignored 1 = 2,400 bps 2 = 9,600 bps 3 = 19,200 bps 4 = 38,400 bps 5 = 57,600 bps 6 = 115,200 bps 7 = 576,000 bps 8 = 1.152 Mbps 9 = 4 Mbps 10..15 = Reserved

The bmStatus field indicators shall be set by the Device as follows:

Media_Busy

- Media_Busy shall indicate zero (0) if the Device:
  - has not received a Check Media Busy class-specific request
  - has detected no traffic on the infrared media since receiving a Check Media Busy class-specific request
  - has returned a header with Media_Busy set to one (1) since receiving a Check Media Busy class-specific request.
-
- Media_Busy shall indicate one (1) if the Device has detected traffic on the infrared media since receiving a Check Media Busy class-specific request. Note that

Media_Busy shall indicate one (1) in exactly one header following receipt of each Check Media Busy class-specific request.

-

Link_Speed

- If no IrLAP frame follows this header, Link_Speed shall indicate zero (0).
- If the Device does not support variable rate sniffing as indicated in the bIrdaRateSniff field of the class-specific descriptor, Link_Speed shall indicate zero (0).
- If variable rate sniffing is currently enabled by a Set IrDA Rate Sniff class-specific request and an IrLAP frame follows this header, Link_Speed shall indicate the speed used to receive the IrLAP frame following this header.

5.4.1.3 USB IrDA Bridge device to USB IrDA Host driver

The USB IrDA Bridge device shall send the prepended IrLAP Payload to the USB IrDA Host driver via Data-In packets. The end of a valid IrDA Payload shall be indicated by either a short or a zero length packet.

5.4.1.4 USB IrDA Host driver to IrDA Host driver

The USB host driver shall separate the Inbound Header from IrLAP Payload received from the USB IrDA Bridge. The IrLAP Payload will be delivered to the host IrLAP layer. The contents of the Inbound Header will be delivered to the appropriate IrDA host functions.

5.4.2. Outbound IrLAP Frame Delivery

5.4.2.1 IrDA Host driver to USB IrDA Host driver

The USB IrDA Host driver shall accept IrLAP frames from the host IrLAP layer. These IrLAP frames will consist of the IrLAP Payload portion of a complete IrLAP frame.

5.4.2.2 Outbound Header

Outbound Header	IrLAP Payload

The USB IrDA Host driver shall prepend an Outbound Header to the IrLAP Payload. This header indicates a new requested number of Extra BOFs and/or new Link Speed.

The purpose of using this in-band header is to enable the USB IrDA Bridge device to receive IrDA frames at the new negotiated link speed immediately after the the IrDA negotiation messages have been transmitted to the communicating IrDA device.

If there is no IrDA Payload to deliver when a change request is required, the USB IrDA Host driver will prepend an appropriate header to a zero length IrDA Payload. Zero length IrDA Payload messages will not be transmitted over the infrared media.

The number of BOFs required vary as a function of mode and negotiated capabilities. See *Serial Infrared Link Access Protocol (IrLAP)*, Version 1.1 referenced in section 1.3 Related Documents for details.

The format for the Outbound Header is:

Field	Size	Description
bmChange	1	Change indicators D7..4 Extra_BOFs (4 bit value) 0 = No Change (BOF ignored) 1 = 48 BOF's 2 = 24 BOF's 3 = 12 BOF's 4 = 6 BOF's 5 = 3 BOF's 6 = 2 BOF's 7 = 1 BOF's 8 = 0 BOF's 9..15 = Reserved D3..0 Link_Speed (4 bit value) 0 = No Change (speed ignored) 1 = 2,400 bps 2 = 9,600 bps 3 = 19,200 bps 4 = 38,400 bps 5 = 57,600 bps 6 = 115,200 bps 7 = 576,000 bps 8 = 1.152 Mbps 9 = 4 Mbps 10..15 = Reserved

The bmChange field indicators shall be set by the Host Driver as follows:

Extra_BOFs

- Extra_BOFs shall be changed if a value other than zero (0) is indicated:
  - Number of Extra BOF's shall be changed upon completion of transmitting IrDA outbound frame.

Link_Speed

- Link Speed shall changed if a value other than zero (0) is indicated:
  - Link Speed shall be changed upon completion of transmitting IrDA outbound frame.

5.4.2.3 USB IrDA Host driver to USB IrDA Bridge device

The USB IrDA Host driver will send the prepended IrLAP Payload to the USB IrDA Bridge device via Data-Out packets. The end of a valid IrDA Payload being sent from the host to the device shall be indicated by either a short or a zero length packet.

5.4.2.4 USB IrDA Host driver to IrDA device

The USB IrDA Bridge device shall separate the Outbound Header from the IrDA Payload.

The USB IrDA Bridge device shall prepare the IrDA Payload for transmission by constructing an IrLAP frame. The appropriate frame check sequence shall be calculated and the appropriate frame wrapper fields shall be added. The constructed IrLAP frame shall be transmitted over the infrared media.

XBOFs or STA or 16 PA	BOF or STA	IrLAP Payload	FCS	EOF or STO

See Appendix D of *Serial Infrared Link Access Protocol (IrLAP)*, Version 1.1 referenced in section 1.3 Related Documents for details on IrLAP Frame Wrappers.

After the IrLAP frame has been transmitted over the infrared media, the USB IrDA Bridge device shall change to the requested settings contained within the Outbound Header.

Except for the case when variable rate sniffing is active, the speed of outgoing frames and the speed of incoming frames are simultaneously changed. The Device shall return STALL if the requested speed and/or number of Extra BOFs is not supported.

6. Device Requests

6.1. Standard Requests

The Device shall support the standard USB device requests as described below. The device shall return STALL if any unrecognized or unsupported standard request is received.

6.1.1. Clear Feature

The Device shall support the following Clear Feature request:

- When directed to an endpoint recipient for ENDPOINT_STALL

The Device may support additional Clear Feature requests. The device shall return STALL if any unrecognized or unsupported Clear Feature request is received.

6.1.2. Get Configuration

The Device shall support the Get Configuration request. The Device shall return zero if the device is unconfigured or the bConfiguration value defined in the Configuration Descriptor is configured.

6.1.3. Get Descriptor

The Device shall support Get Descriptor requests for standard descriptors (Device, Configuration and String). The Device shall return STALL if a Get Descriptor request is received for a class-specific descriptor. The Device may support Get Descriptor requests for vendor-specific descriptors. The Device shall return STALL if a Get Descriptor request is made for an unrecognized or unsupported descriptor.

6.1.4. Get Interface

The Device shall support a Get Interface request for Interface 0 when configured by returning an Alternate Setting of zero. The Device shall return STALL for a Get Interface request for any other Interface or any Get Interface request before the Device is configured.

6.1.5. Get Status

The Device shall support a Get Status request directed at the device, Interface 0 or any defined endpoint (default, Data-In, or Data-Out). The Device shall return STALL if a Get Status request is received for Interface 0 or any defined Endpoint before the Device is configured. The Device shall return STALL if a Get Status request is received for any unrecognized or unsupported recipient.

6.1.6. Set Address

The Device shall support a Set Address request to change the Device Address from the default address (zero) to a unique address. The Device may return STALL if any subsequent Set Address request is received to change the Device Address from a non-zero value to any value (including zero).

6.1.7. Set Configuration

The Device shall support Set Configuration requests to set the Device Configuration to zero (unconfigured) or the bConfiguration value defined in the Configuration Descriptor. The Device shall return STALL if a Set Configuration request is received with any other value.

6.1.8. Set Descriptor

The Device may support Set Descriptor requests for any defined Descriptor (Device, Configuration, Interface, Endpoint, String, Class or Vendor-Specific). The Device shall return STALL if a Descriptor may not be updated, is unrecognized or unsupported.

6.1.9. Set Feature

The Device shall return STALL for any unrecognized or unsupported Set Feature request.

6.1.10. Set Interface

When configured, the Device shall support a Set Interface request to Interface 0 for Alternate Setting 0. The Device shall return STALL for any other Set Interface request.

6.1.11. Synch Frame

The Device shall return STALL for any Synch Frame request.

6.2. Class-Specific Requests

The Device shall support the following class-specific requests. The Device shall return STALL if an unrecognized or unsupported class-specific request is received.

6.2.1. Receiving

This class-specific request asks the Device if it is currently receiving an IrLAP frame. If a frame is being received, the Device shall return a value of one (1); if no frame is being received, the value returned shall be zero (0).

bmRequestType	bRequest	wValue	wIndex	wLength	Data
10100001B	1	Zero	Interface	1	0 or 1

6.2.2. Check Media Busy

This class-specific request instructs the Device to look for a media busy condition. If infrared traffic of any kind is detected by this Device, the Device shall set the Media_Busy field in the bmStatus field in the next Data-In packet header sent to the host. In the case where a Check Media Busy command has been received, a media busy condition detected, and no IrLAP frame traffic is ready to transmit to the host, the Device shall set the Media_Busy field and send it in a Data-In packet with no IrLAP frame following the header.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	3	Zero	Interface	Zero	[None]

6.2.3. Set IrDA Rate Sniff

This class-specific request tells the Device what speed infrared frames it shall accept. When the wValue field is set to one (1), the Device shall dynamically accept frames transmitted at any speed. When the wValue field is set to zero (0), the Device shall only accept frames transmitted at the speed set by Set IrDA Link Speed request or the default speed of 9600 bps if no Set IrDA

Link Speed request has been received. The Device shall return STALL if support for this capability is not indicated in the class-specific descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	4	0 or 1	Interface	Zero	[None]

6.2.4. Set IrDA Unicast List

This class-specific request sets a list of one byte addresses. The Device shall only accept frames from addresses in the list. If the list of addresses is empty (wLength is set to zero) or if any of the addresses in the list are outside the range 1..0x7F, the Device shall accept all valid frames. The Device shall return STALL if the number of addresses provided exceeds the number of addresses supported as indicated in the class-specific descriptor.

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00100001B	5	Zero	Interface	number of addresses in list or Zero	array of addresses or [None]

6.2.5. Get Class Specific Descriptors

This class-specific request instructs the Device to return its class-specific descriptor. The descriptor shows the Device's capabilities which are used by IrLAP when the infrared connection is established with the other infrared device. This request is mandatory and every USB-IrDA Device shall accept it.

bmRequestType	bRequest	WValue	wIndex	wLength	Data
10100001B	6	Zero	Interface	0x000C	[None]

6.3. Vendor-Specific Requests

The Device may support vendor-specific requests. The device shall return STALL if an unrecognized or unsupported vendor-specific request is received.

7. Descriptors

7.1. Standard Descriptors

The Device shall support the standard USB descriptors as described below. The device shall return STALL if a request is received for any unrecognized or unsupported standard descriptor.

7.1.1. Device

The device shall return a Device Descriptor with the following values:

Field	Value	Description
bLength	??	The length of this descriptor in bytes. Length must be at least 18.
bDescriptorType	1	Device Descriptor Type
bcdUSB	??	USB Specification Release Number. The value for Version 1.0 will be 0x100.
bDeviceClass	0	Class information may be found at the interface level
bDeviceSubClass	0	bDeviceClass is zero
bDeviceProtocol	0	bDeviceClass is zero
bMaxPacketSize0	??	Implementation specific, may be set to 8, 16, 32 or 64
idVendor	??	Vendor ID assigned to IHV by USB-IF
idProduct	??	Product ID assigned by IHV to this device model
bcdDevice	??	Device release number in BCD assigned by IHV to this release of model
iManufacturer	??	Index of string descriptor describing IHV. If set to zero (0), there is no manufacturer string.
iProduct	??	Index of string describing this product. If set to zero (0), there is no product string.
iSerialNumber	??	Index of string descriptor with this specific device's serial number. If set to zero (0), this device does not have a serial number.
bNumConfigurations	??	Number of Device configurations.

7.1.2. Configuration

The device shall return a Configuration Descriptor with the following values:

Field	Value	Description
bLength	??	The length of this descriptor in bytes. The length must be at least 9.
bDescriptorType	2	Configuration Descriptor Type
wTotalLength	????	Total length of data returned for this configuration. Includes this descriptor and all of the descriptors that follow (interface, endpoint, class specific and vendor specific, if present). If no vendor specific descriptors are present, this value is 0x002B (43).
bNumInterfaces	1	This configuration has only one interface
bConfigurationValue	??	Implementation specific value used as an argument to Set Configuration to select this configuration
iConfiguration	??	Index of string describing this configuration. If set to zero (0), there is no configuration string.
bmAttributes	??	Configuration characteristics as defined by USB Specification
MaxPower	??	Maximum power draw from the bus by this device when this configuration is selected.

7.1.3. Interface

The device shall return an Interface Descriptor with the following values:

Field	Value	Description
bLength	??	The length of this descriptor in bytes. The length must be at least 9.
bDescriptorType	4	Interface Descriptor Type
bInterfaceNumber	??	Interface in configuration
bAlternateSetting	0	Default (and only) setting for this interface
bNumEndpoints	2	Number of endpoints in this interface, not including the default endpoint.

Field	Value	Description
bInterfaceClass	TBD	Application-specific interface
bInterfaceSubClass	TBD	USB IrDA Bridge device
bInterfaceProtocol	0	Only supported protocol
iInterface	??	Index of string describing this interface. If set to zero (0), there is no interface string.

7.1.4. Endpoint

The device shall return two endpoint descriptors with the following values. The endpoint descriptors may be included in any order.

Field	Value	Description
bLength	??	The length of this descriptor in bytes. Length must be at least 7.
bDescriptorType	5	Endpoint Descriptor Type
bEndpointAddress	0x8?	Any endpoint number not used by another endpoint. The direction shall be set to IN.
bmAttributes	0x02	The endpoint uses bulk transfers
wMaxPacketSize	????	Maximum packet size used by this endpoint. Must be set to 8, 16, 32 or 64.
bInterval	??	Ignored

Field	Value	Description
bLength	??	The length of this descriptor in bytes. Length must be at least 7.
bDescriptorType	5	Endpoint Descriptor Type
bendpointAddress	0x0?	Any endpoint number not used by another endpoint. The direction shall be set to OUT.
bmAttributes	0x02	The endpoint uses bulk transfers

wMaxPacketSize	????	Maximum packet size used by this endpoint. Must be set to 8, 16, 32 or 64.
bInterval	??	Ignored

7.1.5. String

The device may include strings describing the manufacturer, product, serial number, configuration and interface. All string descriptors use the following format:

Field	Value	Description
bLength	??	The length of this descriptor in bytes.
bDescriptorType	3	String Descriptor Type
bString	??	UNICODE string

7.2. Class-Specific Descriptors

The device shall return a class-specific descriptor with the following values:

Field	Value	Description
bLength	??	The length of this descriptor in bytes. The length must be at least 12.
bDescriptorType	0x21	USB IrDA Bridge Device class-specific descriptor
bcdSpecRevision	????	Revision of USB-IrDA Bridge Specification

bmDataSize	??	The maximum number of bytes allowed in any frame for the duration of the connection. See IrLAP Specification.
		D7..0 IrDA Data Size
		D7..6 = Reserved (must be zero)
		D5 = 2048 bytes
		D4 = 1024 bytes
		D3 = 512 bytes
		D2 = 256 bytes
		D1 = 128 bytes
		D0 = 64 bytes

Field	Value	Description

bmWindowSize	??	The maximum number of unacknowledged IrDA frames that can be received before an acknowledgement is sent. See IrLAP Specification for details.
		D7..0 IrDA Window Size
		D7 = Reserved (must be 0)
		D6 = 7 Frames
		D5 = 6 Frames
		D4 = 5 Frames
		D3 = 4 Frames
		D2 = 3 Frames
		D1 = 2 Frames
		D0 = 1 Frame

bmMinTurnaroundTime	??	The length of time in microseconds required for recovery time between the moment the device stops transmitting on the infrared media and the moment it is capable of receiving. See IrLAP Specification for details.
		D7..0 Min Turnaround Time
		D7 = 0 ms
		D6 = 0.01 ms
		D5 = 0.05 ms
		D4 = 0.1 ms
		D3 = 0.5 ms
		D2 = 1 ms
		D1 = 5 ms
		D0 = 10 ms

wBaudRate	????	The transmission/receiving speeds supported by this Device. See IrLAP Specification for details. Each supported speed is indicated by setting the corresponding bit position as identified below:
		D15..D0 IrDA Baud Rate
		D15..D9 = Reserved (must be zero)
		D8 = 4 Mbps
		D7 = 1.152 Mbps
		D6 = 576,000 bps
		D5 = 115,200 bps
		D4 = 57,600 bps
		D3 = 38,400 bps
		D2 = 19,200 bps
		D1 = 9,600 bps
		D0 = 2,400 bps

Field	Value	Description
bmAdditionalBOFs	??	The number of extra BOFs to be added to a frame for proper reception at 115,200 bps. If the Device does not support speeds at or below 115,200 bps this field shall be zero (0). See IrLAP Specification for details. D7..0 Additional BOFs bit 7 = 0 BOFs at 115,200 bps bit 6 = 1 BOFs at 115,200 bps bit 5 = 2 BOFs at 115,200 bps bit 4 = 3 BOFs at 115,200 bps bit 3 = 6 BOFs at 115,200 bps bit 2 = 12 BOFs at 115,200 bps bit 1 = 24 BOFs at 115,200 bps bit 0 = 48 BOFs at 115,200 bps
bIrdaRateSniff	??	This field shall be one (1) if the Device can dynamically accept frames transmitted at any speed. This field shall be zero (0) if the Device can only accept frames transmitted at a pre-specified speed.
bMaxUnicastList	??	The maximum number of addresses that can be specified in a class-specific Set IrDA Unicast List request. If the device does not support the capability of filtering valid frames by their addresses, this field shall be set to zero (0).

7.3. Vendor-Specific Descriptors

The device may support vendor-specific descriptors. The device shall return STALL if a request is received for an unrecognized or unsupported vendor-specific descriptor.

8. Outstanding Issues

This section describes issues that are outstanding in this version of the specification. These issues are to be resolved before Version 1.0 of the specification is released.