Universal Serial Bus
Device Class Definition
for
Printing Devices

Version 1.1
January 2000

Contributors

Axiohn IPB	Kevin Butler	Kevin.Butler@axiohm.com
Canon	Sadahiko Sano	sano@cse.canon.co.jp
Canon	Naoki Shimada	shimada@cbs.canon.co.jp
Canon	Shigeru Ueda	ueda@cpdc.cannon.co.jp
Canon	Bill Russell	Bill_russel@cissc.canon.com
Hewlett Packard	Jeanette Coffren	Jeanette_coffren@vcd.hp.com
Hewlett Packard	Todd Fischer	todd_fischer@hp.com
Hewlett Packard	Karen Van Der Veer	Karenv@vcd.hp.com
Hewlett Packard	Paul Walrath	Paul_walrath@hp.com
Hewlett Packard	Erik Kilk	erik@vcd.hp.com
Hitachi	Richard W. Howell	howellr@halsp.hitachi.com
In-System Design	David Luke	luke@in-system.com
In-System Design	Eric Luttmann	eric@in-system.com
Intel	Sailesh Bissessur	SAILESH_X_BISSESSUR@ccm.ch.intel.com
Lexmark	Reed Owens	Reed@Lexmark.com
Lexmark	C. T. Wolfe	ctwolfe@Lexmark.com
Lucent Technologies	Kevin Lynch	kjlynch@lucent.com
Microsoft	John Fuller	jfuller@microsoft.com
Microsoft	Kenneth Ray	Kenray@microsoft.com
Microsoft	Thomas Nielsen	Tomnie@microsoft.com
Microsoft	Joby Lafky	Jobyl@microsoft.com
MicTron	Man Tam	mtam@mictron.com
Moore Computer Consultants	John Keys	johnk@mcci.com
NCR	Dale Lyons	dale.lyons@atlantaga.ncr.com
NEC	Naofumi Ota	ota@pr2.neec.fc.nec.co.jp
NEC	Takeshi Shinjo	snj@pr2.neec.fc.nec.co.jp
Philips	Sampath Suresh	Suresh.s@blr.sc.philips.com
Phoenix Technologies Ltd.	Mehran Ramezani	Mehran_Ramezani@Phoenix.Com
Seko-Epson	Hideki Morozumi	Morozumi.hideki@exc.epson.co.jp
Smart Technology Enablers, Inc.	David Lawrence	Dlawrence@enablers.com
Warp Nine Engineering	Larry A. Stein	Lstein@fapo.com

Technical Editor, version 1.0

Phoenix Technologies	Curtis Stevens, former Chair	curtis_stevens@phoenix.com

USB Communication Device Class Definition for Printer Devices
Copyright © 1996 - 2000, USB Implementers Forum.
All rights reserved.

INTELLECTUAL PROPERTY DISCLAIMER

THIS SPECIFICATION IS PROVIDED "AS IS" WITH NO WARRANTIES WHATSOEVER INCLUDING ANY WARRANTY OF MERCHANTABILITY, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION, OR SAMPLE.

A LICENSE IS HEREBY GRANTED TO REPRODUCE AND DISTRIBUTE THIS SPECIFICATION FOR INTERNAL USE ONLY. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY OTHER INTELLECTUAL PROPERTY RIGHTS IS GRANTED OR INTENDED HEREBY.

AUTHORS OF THIS SPECIFICATION DISCLAIM ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF PROPRIETARY RIGHTS, RELATING TO IMPLEMENTATION OF INFORMATION IN THIS SPECIFICATION. AUTHORS OF THIS SPECIFICATION ALSO DO NOT WARRANT OR REPRESENT THAT SUCH IMPLEMENTATION(S) WILL NOT INFRINGE SUCH RIGHTS.

All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

Please send comments via electronic mail to usbdevice@mailbag.intel.com

Table of Contents

List of Tables

1. Introduction

The Universal Serial Bus (USB) is a communications architecture that gives a PC the ability to interconnect a variety of devices via a simple four-wire cable. One such device is the printer. Traditionally, printers have been interfaced using the following technologies:

- Unidirectional parallel port
- Bi-directional parallel port
- Serial port
- SCSI port
- Ethernet/LAN

There are other, more sophisticated printer interfaces, but the ones previously listed are the most popular. USB offers a much greater throughput capability than the serial port and is comparable in speed to the parallel port. This makes both parallel and serial printers good candidates for interfacing with USB.

1.1 Scope

This document fully describes the Printer Class of USB devices, including:

- The Printer Class subclass and protocol.
- Device, configuration, interface, and endpoint descriptors.
- The USB standard requests used by printer devices.
- The USB class-specific requests and responses used by printer devices.

1.2 Purpose

The purpose of this document is to describe configuration, interface, and endpoint descriptors as well as a communications protocol for operating system, BIOS, and peripheral designers implementing support for USB printers. These definitions allow an operating system designer to design a single software package to support a given class or subclass of device. These definitions also provide a framework for designing the peripherals in each class or subclass.

Note This specification does not define Page Description Languages (PDL) or Printer Control Protocols (PCP). This specification defines *interfaces* that are intended to support existing PDLs and PCPs.

1.3 Related Documents

The Printer Class defines an architecture for delivering existing printer-command sets to a USB printer. The following documents include relevant USB publications as well as some popular printer PDLs and PCPs:

-
- *Universal Serial Bus Specification*, 1.0 revision 1.1 (also referred to as the *USB Specification*). In particular, see Chapter 9, "USB Device Framework." This specification is available on the World Wide Web site http://www.usb.org.
- *PostScript Language Reference Manual, Second Edition.* Addison-Wesley Publishing Company, Inc. ISBN: 0-201-18127-4.
- PCL / PJL Technical Reference Package. HP part number 5961-0937. Hewlett-Packard Company. 1-800-227-8164 or http://www.hp.com.
  - Includes: PCL 5 Language Technical Reference Manual and Printer Job Language Technical Reference Manual
- IEEE 1284-1994 Standard Signaling Method for a Bidirectional Parallel Peripheral Interface for Personal Computers. This specification is available from the IEEE (www.ieee.org).

- `IEEE P1284.4 Standard for Data delivery and logical channels for IEEE std. 1284 interfaces.` This specification is available from the IEEE (www.ieee.org).

2. Management Overview

Printing devices are unique because they convert a Page Description Language (PDL) into a human-readable printed page. Printers require a mechanism for sending these PDLs, in the form of data, to the printer; and a mechanism for returning status information from the printer. Conventional parallel printers use a bi-directional printer port to support these languages. USB uses a Bulk OUT endpoint to send the data to the printer, and a Bulk IN endpoint for status and other data received from the printer. The Bulk IN endpoint is sufficient for status data retrieval because in today's existing implementations, the parallel status information is retrieved via polling mechanisms in the host. Some PDLs allow the printer to return large amounts of data, such as font definitions. If the printer implements this capability, it returns the data on the same Bulk IN endpoint which is typically used for status.

Some examples of PDLs are *PostScript* and *Hewlett Packard Graphics Language (HPGL)*. Examples of Printer Control Protocols (PCP) are *Hewlett Packard PJL*, *IEEE 1284.1*, and *Microsoft Windows Printing System (WPS)*. The *USB Device Class Definition for Printing Devices* provides a mechanism for administering all of the previously listed capabilities.

USB printers report their capabilities via the class-specific command GET_DEVICE_ID which returns a device ID string that is compatible with IEEE-1284. This string lists all the PDLs and PCPs supported by the printer.

3. Functional Characteristics

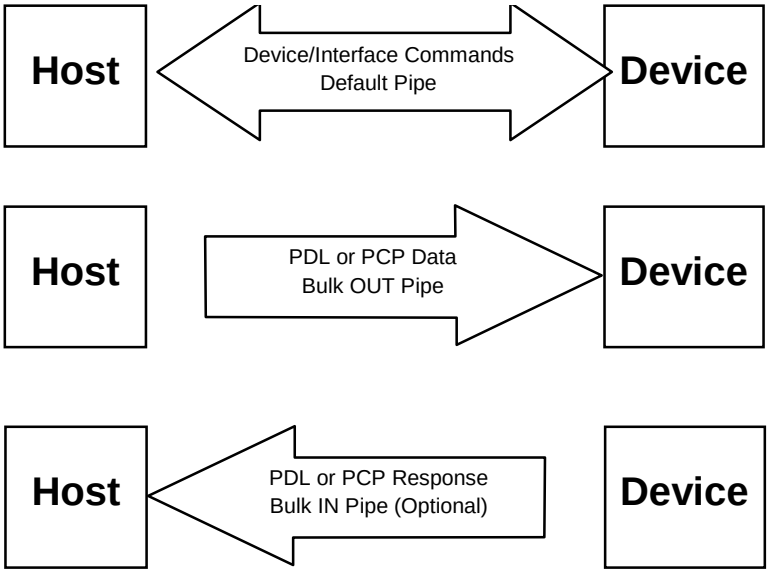
This section describes the functional characteristics of printers, including:

- The operational model
- Interfaces

3.1 Operational Model

Printers have two different types of commands: those that transfer data, and those that control the USB interface or printer interface. The host prints something on a printer by delivering data on the Bulk OUT endpoint. This data may take the form of *PostScript*, *HP PCL*, or any other PDL. This data may also be encapsulated in a PCP, such as *IEEE 1284.1*, or something that is vendor-specific. In addition, the data may also be simple text, or it may be a proprietary PDL. The printer can respond periodically on the Bulk IN endpoint with status regarding the data it is receiving, or because of an asynchronous event. A typical printed page takes the following sequence:

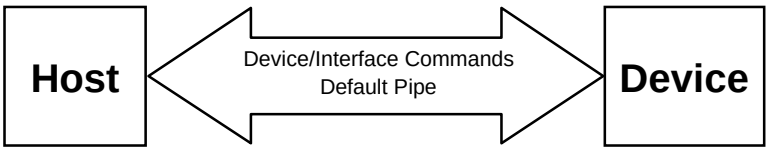
Figure 1 Printing a Page



The PDL data is sent to the device on the Bulk OUT pipe. If the device uses a PCP, then the PDL is encapsulated in the PCP; the Bulk IN pipe is used for any responses, such as errors and printer status as defined in the PCP. For a unidirectional interface, the status is retrieved by issuing a GET_PORT_STATUS on the default pipe, and the status is returned on the default pipe.

The second type of command controls the USB interface. These commands include the standard requests, such as GET_DESCRIPTOR and SET_CONFIGURATION. These requests are described in Chapter 9, "USB Device Framework," in the *Universal Serial Bus Specification*. Printer-specific commands, such as SOFT_RESET or GET_PORT_STATUS, are included as well. The following diagram shows a control command sequence of a typical interface:

Figure 2 Interface Control Command



Commands sent on the default pipe report information back to the host on the default pipe. If an error occurs, they generate a standard USB error status. This applies to all class-specific requests in addition to the general requests described in Chapter 9, "USB Device Framework," in the *Universal Serial Bus Specification*.

3.2 Interfaces

Printers support three possible interfaces, but only one can be enabled at any given time. The interfaces are defined as follows:

- Unidirectional Interface. The unidirectional interface supports only the sending of data to the printer via a Bulk OUT endpoint. Status data that is compatible with a standard PC parallel port is retrieved via the class-specific command GET_PORT_STATUS over the default pipe. For more information about status data, see Section 4.2.2, "GET_PORT_STATUS."

- Bi-directional Interface. The bi-directional interface supports sending data to the printer via a Bulk OUT endpoint, and receiving status and other information from the printer via a Bulk IN endpoint. Status data that is compatible with a standard PC parallel port is also available when this interface is in use via the GET_PORT_STATUS class-specific command over the default pipe. For more information about status data, see Section 4.2.2, "GET_PORT_STATUS."

- IEEE 1284.4 compatible Bi-directional Interface. The 1284.4 interface supports sending data to the printer via a Bulk OUT endpoint, and receiving status and other information from the printer via a Bulk IN endpoint, just as the bi-directional interface does. The 1284.4 interface additionally specifies that data will be transmitted to and from the device using the 1284.4 protocol. The 1284.4 protocol is defined by the " IEEE P1284.4 Standard for Data delivery and logical channels for IEEE std. 1284 interfaces, version 1.0." This specification is pending final approval.

This specification is intended only for the definition of USB Printer devices using 1284.4 for communications. The definition of multi-function devices using 1284.4 for communications is outside the scope of this specification.

One of these interfaces shall be supported. If multiple printer interfaces are supported, they shall be implemented as alternate interfaces. The Protocol field in the interface descriptor informs the host of the interface type: 01 is the unidirectional interface, 02 is the bi-directional interface. and 03 is the 1284.4 interface.

3.3 Status Reporting

USB printers have several ways of reporting status information; the host selects the method of status retrieval based on the type of status desired.

3.3.1 Compatibility with a Centronics Parallel Port

The GET_PORT_STATUS class-specific command returns 1 byte of information that is compatible with the standard PC parallel port, as described in Section 4.2.2 "GET_PORT_STATUS."

3.3.2 Printer Capabilities

The GET_DEVICE_ID command returns a variable-length string that describes the printer. The capabilities include PDLs, such as *PostScript* or *HPGL*, and manufacturer ID strings. Host software reads this string to determine the supported PCPs.

3.3.3 Protocol-Specific Status

PCPs support extensive methods of status reporting. Status information for PCPs is returned on the Bulk IN endpoint.

4. Requests

A printer can respond to two different types of requests:

- Standard USB device requests, which perform general functions for supporting the bus and bus-related functions.
- Class-specific requests, which cause the device to transfer command data to or from the host.

4.1 Standard Requests

A printer device supports all of the standard device requests described in Chapter 9, "USB Device Framework," of the *Universal Serial Bus Specification*:

- Clear Feature
- Get Configuration
- Get Descriptor
- Get Interface
- Get Status
- Set Address
- Set Configuration
- Set Descriptor (optional)
- Set Interface
- Set Feature

4.2 Class-Specific Requests

Printers support a variety of class-specific requests. The following table is a comprehensive list of all the class-specific requests supported by printers:

Table 1 - Class-Specific Requests

Label	bmRequestType	bRequest	wValue	wIndex	wLength	Data
GET_DEVICE_ID	10100001B	0	Config Index	Interface & Alternate Setting	Maximum Length	1284 Device ID String
GET_PORT_STATUS	10100001B	1	Zero	Interface	1	BYTE
SOFT_RESET	00100001B	2	Zero	Interface	Zero	[None]

4.2.1 GET_DEVICE_ID (bRequest = 0)

This class-specific request returns a device ID string that is compatible with IEEE 1284. See IEEE 1284 for syntax and formatting information. A printer with multiple configurations, interfaces, or alternate settings may contain multiple IEEE 1284 device ID strings. The wValue field is used to specify a zero-based configuration index. The high-byte of the wIndex field is used to specify the zero-based interface index. The low-byte of the wIndex field is used to specify the zero-based alternate setting. The device ID string is returned in the following format:

Table 2 - Capabilities String

Offset	Type	Description
0...n-1	Data	IEEE 1284 device ID string (including length in the first two bytes in big endian format).

4.2.2 GET_PORT_STATUS (bRequest = 1)

This class-specific request returns the printer's current status, in a format which is compatible with the status register of a standard PC parallel port. The following table defines the data returned.

Note Some USB printers may not always be able to determine this information. In this case, they should return benign status of "Paper Not Empty," "Selected," and "No Error."

Table 3 - Printer Port Status

7	6	5	4	3	2	1	0
Reserved		Paper Empty	Select	Not Error	Reserved		

Bit(s)	Field	Description
7 .. 6	Reserved	Reserved for future use; device shall return these bits reset to zero.
5	Paper Empty	1 = Paper Empty, 0 = Paper Not Empty
4	Select	1 = Selected, 0 = Not Selected
3	Not Error	1 = No Error, 0 = Error
2 .. 0	Reserved	Reserved for future use; device shall return these bits reset to zero.

4.2.3 SOFT_RESET (bRequest = 2)

This class-specific request flushes all buffers and resets the Bulk OUT and Bulk IN pipes to their default states. This request clears all stall conditions. This reset does NOT change the USB addressing or USB configuration.

Note: Version 1.0 of the specification incorrectly stated that the bmReqestType for SOFT_RESET was 00100011B. Version 1.1 Host software implementers should be prepared for USB printers that expect this request code, and version 1.1 device implementers should be prepared for host software that issues this request code.

5. Standard Descriptors

Printer Class devices support the following standard USB descriptors:

- **Device**. Each printer has one device descriptor.
- **Configuration**. Each device has one default configuration descriptor which supports at least one interface.
- **Interface**. A printer device has a single data interface with possible alternates.
- **Endpoint**. A printer device supports the following endpoints:
  - Bulk OUT endpoint. Used for transfer of PDL/PCP data.
  - Optional Bulk IN endpoint. Provides status and other return information.

Printers have no class-specific descriptors.

The rest of this section describes the standard USB device, configuration, interface, and endpoint descriptors for printer devices. For information about other standard descriptors, see Chapter 9, "USB Device Framework," of the *Universal Serial Bus Specification*.

5.1 Device Descriptor

There is only one device descriptor for each USB device. This descriptor contains the definitions of the device class and the device subclass, among other following device descriptor information:

Table 4 - Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	Byte	12h	Size of this descriptor, in bytes.
1	bDescriptorType	Byte	01h	DEVICE descriptor type.
2	bcdUSB	Word	????h	USB Specification Release Number, in Binary-Coded Decimal (BCD).
4	bDeviceClass	Byte	00h	See Section 5.3, "Interface Descriptors."
5	bDeviceSubClass	Byte	00h	See Section 5.3, "Interface Descriptors."
6	bDeviceProtocol	Byte	00h	See Section 5.3, "Interface Descriptors."
7	wMaxPacketSize0	Byte	??h	Maximum packet size for endpoint 0.
8	idVendor	Word	????h	Vendor ID (assigned by the USB).
9	idProduct	Word	????h	Product ID (assigned by the manufacturer).
10	bcdDevice	Word	????h	Device release number, in BCD.
14	iManufacturer	Byte	??h	Index of string descriptor describing manufacturer.
15	iProduct	Byte	??h	Index of string descriptor describing this product.
16	iSerialNumber	Byte	??h	Index of string descriptor describing the device's serial number.
17	bNumConfigurations	Byte	??h	Number of possible configurations. There must be at least one configuration.

5.2 Configuration Descriptor

A printer has one default configuration descriptor. This descriptor has one interface, called the Data interface, which has one or two endpoints: Bulk OUT, and the optional Bulk IN.

Table 5 - Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	Byte	09h	Size of this descriptor, in bytes.
1	bDescriptorType	Byte	02h	CONFIGURATION descriptor type.
2	bTotalLength	Word	????h	Number of bytes in this configuration. This includes the configuration descriptor plus all of the interface and endpoint descriptors.
4	bNumInterfaces	Byte	??h	Printers have at least one interface.
5	bConfigurationValue	Byte	01h	Value used as an argument to the SetConfiguration() request to select this configuration.
6	iConfiguration	Byte	??h	Index of string descriptor describing this configuration.
7	bmAttributes	Byte	??h	Configuration characteristics: Bit Description 7 Reserved (set to one). 6 Self-powered. 5 Remote wakeup. 4-0 Reserved, set to 0. Printers are usually self-powered, but may be both self-powered and bus-powered.
8	MaxPower	Byte	??h	Maximum power consumption of this configuration.

5.3 Interface Descriptors

All printers support a Data interface for transferring data to and/or from the device:

Table 6 - Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	Byte	09h	Size of this descriptor, in bytes.
1	bDescriptorType	Byte	04h	INTERFACE descriptor type.
2	bInterfaceNumber	Byte	00h	Zero-based value identifying the number of this interface.
3	bAlternateSetting	Byte	??h	Value used to select an alternate interface.
4	bNumEndpoints	Byte	??h	Number of endpoints used by this descriptor. This is 01h or 02h for printers
5	bInterfaceClass	Byte	07h	Base class for printers.
6	iInterfaceSubClass	Byte	01h	The subclass codes for Printer devices are: 01 Printers
7	bInterfaceProtocol	Byte	??h	Printer Interface Type: 00 Reserved, undefined. 01 Unidirectional interface. 02 Bi-directional interface. 03 1284.4 compatible bi-directional interface. 04-FEh Reserved for future use. FFh Vendor-specific printers do not use class-specific protocols.
8	iInterface	Byte	??h	Index to string that describes this interface.

5.4 Endpoint Descriptors

Printers support one or two endpoints. In addition to the Default endpoint, printers are required to support the Bulk OUT endpoint, or both the Bulk OUT and the Bulk IN endpoints.

Table 7 - Endpoint Descriptor

Attributes	IN/OUT	Type	Description
Bulk OUT	OUT	Mandatory	Bulk OUT endpoint.
Bulk IN	IN	Optional	Bulk IN endpoint.

5.4.1 Bulk OUT Endpoint

The Bulk OUT endpoint is used for transferring PDL and PCP data from the host to the printer:

Table 8 - Bulk OUT Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	Byte	09h	Size of this descriptor, in bytes.
1	bDescriptorType	Byte	05h	ENDPOINT descriptor type.
2	bEndpointAddress	Byte	0?h	The address of this endpoint on the USB device. This address is an endpoint number between 1 and 15. Bit 0..3 Endpoint number. Bit 4..6 Reserved, must be 0. Bit 7 0 = Out, 1 = In
3	bmAttributes	Byte	02h	This is a Bulk OUT endpoint.
4	wMaxPacketSize	Word	????h	Maximum data transfer size.
6	bInterval	Byte	00h	Does not apply to Bulk endpoints.

5.4.2 Bulk IN Endpoint

The Bulk IN endpoint is used to return any data generated by the PDL or PCP to the host. If the printer supports a PCP, such as IEEE-1284.1 or IEEE-1284.4, this endpoint will return status or other printer-related information:

Table 9 - Bulk IN Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	Byte	09h	Size of this descriptor, in bytes.
1	bDescriptorType	Byte	05h	ENDPOINT descriptor type.
2	bEndpointAddress	Byte	8?h	The address of this endpoint on the USB device. This address is an endpoint number between 1 and 15. Bit 0..3 Endpoint number. Bit 4..6 Reserved, must be 0. Bit 7 0 = Out, 1 = In
3	bmAttributes	Byte	02h	This is a Bulk endpoint.
4	wMaxPacketSize	Word	????h	Maximum data transfer size.
6	bInterval	Byte	00h	Does not apply to Bulk endpoints.