USB Feature Specification: Shared Endpoints

SYSTEMSOFT® CORPORATION
INTEL® CORPORATION

Revision 1.0

October 27, 1999

Revision History

Revision	Issue Date	Comments
1.0	10/27/1999	Version 1.0 release.

Contributors

Shelagh Callahan	Intel Corporation
Steve McGowan	Intel Corporation
John S. Howard	Intel Corporation
Dave Lawrence	SystemSoft Corporation

Please send comments via electronic mail to ccscomments@usb.org

Table of Contents

1. Introduction

The *Universal Serial Bus Specification* allows only the default endpoint to be shared between interfaces on a device. This was a simplifying assumption to reduce coupling between interfaces. However, as additional classes were defined, it became clear that several interfaces on a device could have very similar requirements on an endpoint, which would allow it to be shared among those interfaces. Such sharing would reduce the overall number of endpoints required, and thus the overall cost of the device.

For instance, the specification of synchronization mechanisms and reuse of class specifications to allow easy driver binding could substantially increase the number of endpoints required for some devices. Devices which require feedback would need a substantial number of endpoints since each endpoint requiring synchronization may require a separate endpoint to report feedback information.

In addition, some devices may actually be collections of interfaces that use very simple and low bandwidth data reporting mechanisms, such as an interrupt endpoint, for a common activity such as event notification. If this interrupt endpoint could be shared by multiple interfaces, the device could use fewer endpoints and fewer endpoints would need to be scheduled.

1.1 Purpose

The purpose of this document is to provide a common specification of how a physical pipe resource should be shared across multiple interfaces.

1.2 Scope

This document fully describes the Common Class Shared Endpoint Feature extension for USB Devices. It describes:

- Host and Device requirements for shared endpoints.
- Descriptors for representing shared endpoints in the device framework
- Data addressing structures

1.3 Related Documents

USB Specification, Version 1.1, available at http://www.usb.org.

USB Common Class Specification, Version 1.0, available at http://www.usb.org.

1.4 Terms and Abbreviations

CCS	Common Class Specification
DEVICE ENDPOINT	See Universal Serial Bus Specification Rev. 1.1
LOGICAL ENDPOINT	A uniquely addressable portion of a Universal Serial Bus Device Endpoint that is the source or sink of information in a communication flow between the host and the device. Is identified via an Universal Serial Bus Endpoint Address and a Logical Endpoint Number.
LOGICAL ENDPOINT NUMBER	A number in the range 00H to FFH that is used to uniquely identify a Logical Endpoint within a Device Endpoint.
LOGICAL PACKET INFO RECORD	The repeating group of members in a Logical Packet Descriptor that describe individual logical packets. The “Info record” includes the members: bLogicalPacketID, bmLogicalPacketFlags and bLogicalPacketSize.
LOGICAL PIPE	A logical abstraction representing the association between a logical endpoint on a device and software on the host.
LOGICAL PIPE CONTROL	A uniquely identifiable flow of control information over a shared pipe.
LOGICAL PIPE DATA	A uniquely identifiable flow of data over a shared pipe.
LOGICAL PIPE PACKET	A control header and data payload. Logical endpoint packets are used to move logical endpoint data and control information.
SHARED ENDPOINT	A physical Device Endpoint which supports multiple Logical Device Endpoints.
SHARED PIPE	A physical pipe that supports multiple Logical Pipes.

2. Management Overview

This document provides a common definition of how an endpoint resource should be shared across multiple interfaces. It specifies the representation and requirements of the Host and Devices for consistent operation.

Several types of pipes recur in many different interface definitions. For instance, many interfaces need an interrupt pipe for event notification. This interrupt pipe is used relatively rarely and yet, if individually implemented, could take up many different endpoints. Also, synchronization pipes are required for many different isochronous pipes.

Device drivers on the Host system must be able to use pipes without knowledge that the underlying pipe resource is shared. This requires an additional software abstraction on the host (and potentially on the device) to manage the dedicated resource abstraction. *Host software* is assumed to be the provider of this abstraction (See Figure 2-1).

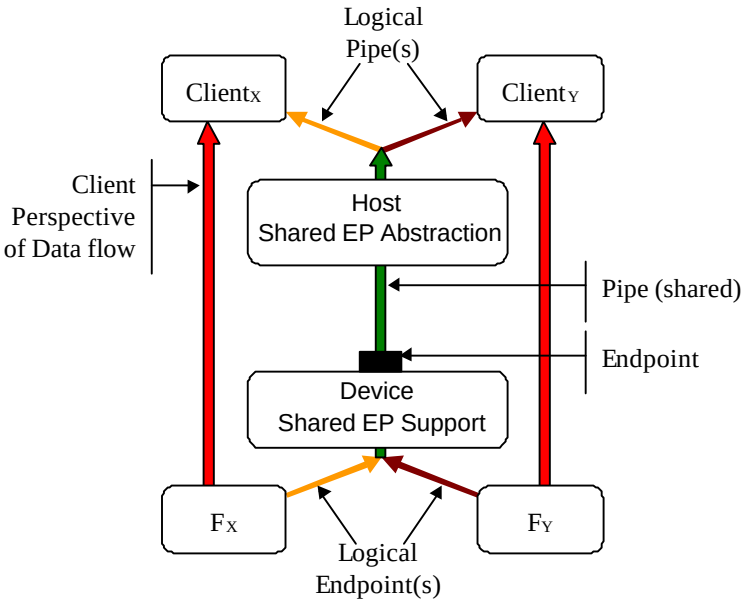

Figure 2-1: Mapping of Logical Resources to Physical Resources

This document defines a method to describe how to associate a pipe with more than one interface. It defines a method of indicating the source or target of individual data packets so that information from such a pipe can be routed appropriately. The basic Host/Device requirements are:

- Host software must provide transparent sharing of pipes for client drivers.
- Shared pipes must support all operations and have identical behavior as physical pipes.
- Shared Interrupt and Bulk pipes provide logical flow control in order to prevent the loss of samples from a fast logical source overrunning a slow logical target.

3. Functional Characteristics

The shared pipe service provided by host software hides the sharing feature from a client driver. A driver is need not be aware that one or more of the pipes in an interface is actually a shared resource. Host software must ensure all logical pipe semantics and operations are identical to those of a core specification pipe.

Each unique source and target using the shared pipe represents a separate *flow of* data that is multiplexed over a single pipe. The logical data flow is uniquely identified using a header for each separately sourced or targeted data. The header provides information, which allows routing to occur. The header plus the data payload is referred to below as a logical packet. Figure 3-1 illustrates the possible mapping of logical pipe packets to core pipe packets.

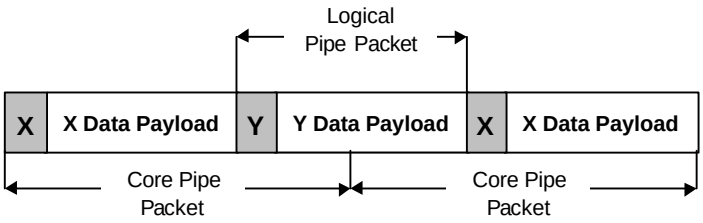

Figure 3-1: Physical vs Logical Packets

Shared pipes introduce a new phenomenon to USB. **Error! Reference source not found.** illustrates. Assume $Client_X$ and $Client_Y$ are logical targets which share a pipe resource. $Client_X$ consumes bytes at rate ***M*** bytes/ms and $Client_Y$ consumes bytes at ***N*** bytes/ms. If ***N***<<***M***, then the condition may exist where the $Client_Y$ target does not consume data as fast as the F_Y source produces data. In Universal Serial Bus Rev 1.0, the consumer has explicit control over flow control. For Device to Host transfers, the client simply does not make requests until buffer space is available. For Host to Device transfers, the device can NAK until there is buffer space on the device. Shared pipes are different because the Shared Endpoint layer maintains the pipe, initiating requests as long as one client driver is making requests. For Device to Host transfers, this means that data may arrive faster than clients may be asking. For Host to Device transfers, the device should not NAK requests unless all sharing logical endpoints agree to NAK (i.e. because it is inappropriate to starve unrelated logical data flows).

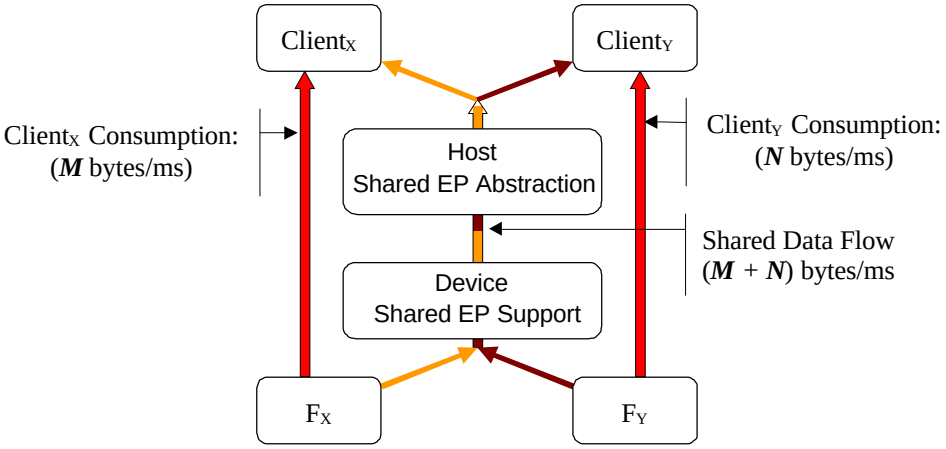

Figure 3-2: Data Overrun Due to Slow Consumers

Shared pipes as a feature provides mechanism for logical pipe flow control. The flow control mechanism is explained in detail in Section 4.

4. Operational Model

This section describes the operational model for shared pipes.

4.1 Device Components

Logical endpoints are represented in the device framework with Logical Endpoint descriptors (see Section 5.2). The configuration data does not contain standard endpoint descriptors for the shared endpoints. Rather, only the logical endpoints are described in the configuration data. The characteristics of the physical endpoint are derived from the combination of Logical Endpoint descriptors. The rules for deriving physical endpoint characteristics are provided in Section 4.4.1.

Interfaces that use shared pipes use Logical Endpoint descriptors in the position where a standard endpoint descriptor would normally be placed (see Figure 4-1). Each Logical Endpoint descriptor represents a placeholder for the actual endpoint descriptor. The Logical Endpoint descriptor has a footprint identical to a standard endpoint descriptor, but contains information allowing the host software to intercept the configuration, identify the logical endpoint number, map to and derive the attributes of the physical endpoint. The form of the Logical Endpoint descriptor is illustrated in Figure 5-1. An interface may have more than one Logical Endpoint descriptor. Each Logical Endpoint descriptor declared by a device must have a unique logical endpoint number. Refer to Section 5.2 for the rules devices must use for assigning logical endpoint numbers.

Configuration Descriptor
- Interface Descriptor (Interface 0)
  - Endpoint Descriptor (81H)
  - *Logical Endpoint Descriptor (82H, 00H)*
  - *Logical Packet Descriptor (82H, 00H)*
  - Endpoint Descriptor (03H)
- Interface Descriptor (Interface 1)
  - *Logical Endpoint Descriptor (82H, 01H)*
  - *Logical Packet Descriptor (82H, 01H)*
  - Endpoint Descriptor (84H)
  - *Logical Endpoint Descriptor (82H, 02H)*
  - *Logical Packet Descriptor (82H, 02H)*

Figure 4-1: Illustration of Logical Endpoint and Packet Descriptors in Configuration

In Figure 4-1 the notation in the parentheses represents the value found in the respective bEndpointAddess field of the Logical Endpoint descriptor and the bLogicalEPNumber field of the Logical Packet descriptor.

4.2 Addressing Requests to Logical Endpoints

Standard, Class or Vendor-specific requests can be addressed to Logical endpoints. Only requests applicable to standard endpoints may be issued to logical endpoints. Logical endpoints receive requests over the default pipe using an endpoint recipient (in the *bmRequestType* field), the endpoint number in the low byte of *wIndex* and the logical endpoint number in the high byte of *wIndex*.

4.3 Isochronous Shared Pipes

Shared isochronous endpoints must have a *maxPacketSize* large enough so that no data item ever spans frames. In particular, shared synchronization endpoints must have a *maxPacketSize* large enough to accommodate the worst case situation. This occurs when all sharing isochronous logical data endpoints, which are isochronous synchronization data endpoints, send their synchronization data in the same frame. All synchronization data and associated wrappers for all the isochronous data pipes must be bundled in one physical pipe packet.

A Logical Endpoint descriptor specifies a maximum packet size for the physical pipe (*bMaxPacketSize*). The packet size used by the Logical Endpoint is defined in the Logical Packet Descriptor.

Shared Endpoint flow control is not allowed on isochronous pipes.

4.4 Bulk and Interrupt Shared Pipes

The Logical Endpoint descriptor specifies a maximum packet size for the logical endpoint (*bMaxPacketSize*). Bulk and interrupt logical pipe packets may span multiple physical pipe packets.

4.4.1 Flow Control

Devices that can accept data loss on their shared Bulk or Interrupt pipes can reduce their complexity by not implementing flow control.

A flow control mechanism requires a pipe in the opposite direction through which flow control information can move. The flow control protocol defined here is optimized for applications where the data stream of a particular type is inherently bi-directional (e.g. a bulk data stream includes both an input and output pipe and data flows regularly in both directions). Flow control packets are multiplexed into the data stream in the opposite direction and since data endpoints exist in each direction, flow control endpoints do not need to be declared. If flow control is implemented on a device with a unidirectional data stream then a dedicated flow control endpoint in the opposite direction must be implemented however no endpoint descriptor shall be declared.

If any Logical Packet Info member enables flow control then the flow control pipe is inferred by host software to be the same physical endpoint number (but with the opposite direction) as the physical endpoint number of logical endpoint data pipe. When inference of the flow control pipe occurs, the *wMaxPacketSize* of the flow control endpoint is 8. The flow control pipe must always be of the same transfer type as the logical data pipe.

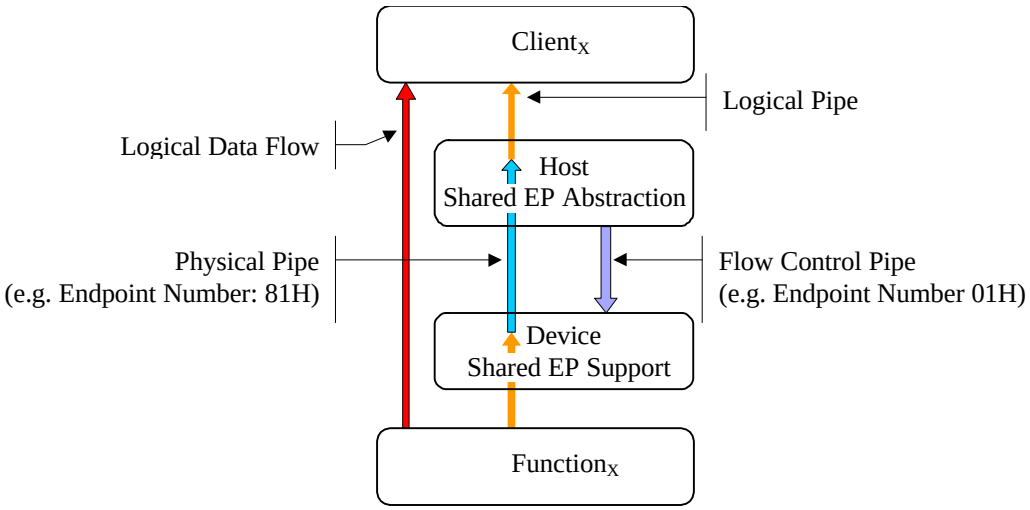

Figure 4-2: Simple Shared Pipe Flow Control Model

A pipe can be a flow control pipe. A special data header is used to distinguish Logical data packets from Logical flow control packets.

Flow control information is sent from a target to a source. A source must not send data on the logical pipe until the target has provided a grant (*to send*). A grant represents the amount of space currently available at the target to receive data. Space is expressed in terms of an integral number of logical data packets of buffering available. For

example, take a logical packet size of 64 and a host client provides a 2K byte destination buffer. This would yield a grant with a value of 32 (2K/64). A grant is issued by the source for each buffer, as the buffer is made available by the target. A grant has two parts: a data wrapper, which identifies the channel number and a data payload, which is the value of the grant (see Figure 5-6).

A target sends a grant for each buffer made available. A source sends packets on the pipe until it has exhausted the grant (i.e. sent all the packets specified in the grant value). The source of each logical pipe on a physical pipe is required to support two simultaneous grants. This allows a data transfer to maintain streaming by having one grant active while another is pending.

4.4.1.1 Managing Grants

As mentioned above, a target sends one grant for each buffer. There are several events, which effect how and when grant value goes to zero and it is discarded.

Short Packet The USB Specification provides short packet behavior which forces the retirement of a buffer whenever the target sends a short packet, indicating a requested *transfer unit* is complete. Similarly, a source can send a short logical data packet, which has two effects.

- Short logical data packet forces retirement of client buffer associated with the logical pipe data.
- Source cancels the current grant. If the grant value associated with the current transfer is non-zero when the short logical data packet is delivered, the remaining grant is canceled the same way as if the grant value went to zero.

Abort Pipe Host software may provide a client the ability to abort a pipe. Aborting a pipe is described in Chapter 10 of the USB Specification. Briefly, aborting a pipe essentially means that all buffers queued for the endpoint are immediately returned to the client. No effects to either host or endpoint state.

The effect on the grants depends on the direction of the pipe.

Target: Host	**Target: Device**
Host has issued grants to the device. To prevent lost data, the Host should notify logical endpoint to cancel all grants.	Host has received grants from the device. Host retains memory of all grants not used.

Reset Pipe

Host software may provide a client the ability to reset a pipe. Resetting a pipe is described in Chapter 10 of the USB Specification. Breifly, resetting a pipe aborts the buffers, cancels all grants and may involve a request to transition the endpoint state to active.

The effect on the grants depends on the direction of the pipe.

Target: Host	**Target: Device**
Host has issued grants to the device. To prevent lost data, the Host should notify logical endpoint to cancel all grants. Any adjustments to the endpoint state via explicit requests implicitly cancel grants queued at the device for the logical endpoint.	Host has received grants from the device. Host loses memory of all grants not used.

Normal Completion

In many cases, the data stream will span buffers and grants. This is the simple case where grants are discarded when their value reaches zero as a result of moving all the data.

4.4.2 Halting Logical Endpoints

Interrupt and Bulk logical endpoints on the device are allowed to signal the Host the state of the logical endpoint is halted. A device sends a STALL flow control packet to the Host to communicate the Halt condition. See Section 5.5 for format of this packet.

Stall notifications are sent on Interrupt or Bulk logical pipes in response to catastrophic errors on the device. No assumptions can be made about the state of the logical data stream and the time when the stall notification is sent by the device. Recovery may involve significant host or user interaction.

A ClearFeature(HALT) request is sent to a logical endpoint to recover from Abort Pipe and Reset Pipe on the related logical pipe. Receipt of this request causes an endpoint/pipe to return to its starting state (as specified by the USB Specification) and additionally invalidates all outstanding grants.

4.4.3 Latency

A device (or host) must assume that a large buffer that can contain many Logical Packets is allocated for receiving data. If prompt delivery of a particular packet is important then the device (or host) must use standard USB methods to force the retirement of the buffer.

5. Descriptors

This section describes the special descriptors use to represent shared endpoints in the device framework and also defines the format of the data wrapper used to identify logical data and control.

5.1 Interface Descriptor

To invoke the Common Class driver and provide shared endpoint support the device must declare a CCS Interface Descriptor. See the CCS Logical-Device Feature Specification for a detailed description of the CCS Interface Descriptor.

5.2 Version Descriptor

A device that employs Shared Endpoint functionality must declare a CCS Version Descriptor. The CCS Version descriptor will contain a Version Information Record for the CCS Shared Endpoint feature.

The Feature ID (*bFeatureID*) for the Shared Endpoint feature is 0x03.

All Feature Flags (*bmFeatureFlags*) are reserved and set to 0.

See the CCS Version Descriptor Feature Specification for a detailed description of the CCS Version Descriptor.

5.3 Logical Endpoint Framework Representation

The endpoint descriptor is defined in Chapter 9 of the Universal Serial Bus Specification Rev. 1.0. Logical endpoint descriptors have exactly the same footprint.

Following is the format of a Logical Endpoint Descriptor.

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	LOGICAL ENDPOINT Descriptor Type
2	*bEndpointAddress*	1	Endpoint	See chapter 9 of the Universal Serial Bus Specification
3	*bmAttributes*	1	Bit Map	See chapter 9 of the Universal Serial Bus Specification
4	*wMaxPacketSize*	2	Number	Maximum *Physical* packet size this logical endpoint is capable of sending or receiving when the configuration is selected. For Interrupt and bulk endpoints smaller data payloads may be sent, but will retire the current logical host buffer and may or may not require intervention to restart. See Section 4.4.1 for more information.
6-8	*No Change*	3	Variable	These fields are the same as a physical endpoint descriptor, see chapter 9 of the Universal Serial Bus Specification.

Figure 5-1: Logical Endpoint Descriptor Format

Note that the *bEndpointAddress* field in the Logical Endpoint Descriptor contains the physical endpoint address and the logical endpoint number is declared in the *bLogicalEPNumber* field of the Logical Packet Descriptor that immediately follows the Logical Endpoint Descriptor. This address pair is used to access logical endpoints in the wIndex field of a USB request.

5.4 Logical Packet Framework Representation

A Logical Packet Descriptor consists of a 4 byte header followed by one or more Logical Packet Info Records. A Logical Packet Info Record contains the 3 members: bLogicalPacketID, bmLogicalPacketFlags and bLogicalPacketSize. A record is defined for each Packet ID generated or expected by a logical pipe.

Following is the format of a Logical Packet Info Record.

Offset	Field	Size	Value	Description
0	*bLogicalPacketID*	1	Number	Bit 7 Reserved Bit 0..6 Logical Packet ID
1	*bmLogicalPacketFlags*	1	Bitmap	Logical Packet Flags Bit 3..7 Reserved Bit 2 Flow Control Required 1 = True 0 = False Bit 1 Fixed/Variable Header 1 = Variable 0 = Fixed Bit 0 Pass Header 1 = Pass 0 = Strip (See details below)
2	*wLogicalPacketSize*	2	Number	Logical Packet size associated with the Logical Packet ID. If this Logical Packet Info member is used to define a fixed size Logical Packet then this is the exact size of the data payload section of that Logical Packet . If this Logical Packet Descriptor is used to define a variable sized Logical Packet then this is the maximum size of the data payload section of the Logical Packet . The wLength field of the Logical Packet header identifies the actual size. For Interrupt and bulk endpoints smaller data payloads may be sent, but will retire the current logical host buffer and may or may not require intervention to restart. See Section 4.4.1 for more information.

Figure 5-2: Logical Packet Info Record Format

bLogicalPacketID

Bits 0..6 Logical Packet ID – This number is used as a unique identifier for logical Packet identification. This number, plus the bEndpointAddress must be unique on the device

(across all combinations of active interfaces and alternate settings). The Message ID can be between 1 and 127, 0 is reserved.

BmLogicalPacketFlags

Bit 2 Flow Control – If this bit is set (1) then the device does not support flow control for this logical packet. If this bit is set (0) then flow control is required by the device for this logical packet.

Bit 1 Fixed/Variable – If this bit is set (1) then the message is a variable size where the wPayloadSize field concatenated to the Logical Packet header determines the number of bytes in the packet. If this bit is set (0) then the message is a fixed size where the associated wLogicalPacket Size determines the number of bytes in the packet.

Bit 0 Pass Header – If this bit is set (1) then the message is transmitted to the target driver with the message header included (bControl and optionally wPayloadSize if it is a variable size message). Otherwise the message header, bControl and optionally wPayloadSize are stripped before passing the message to the target driver.

Following is the format of a Logical Packet Descriptor.

Offset	Field	Size	Value	Description
0	*bLength*	1	Number	Size of this descriptor in bytes
1	*bDescriptorType*	1	Constant	LOGICAL PACKET Descriptor Type
2	*bLogicalEPNumber*	1	Number	Number used as a unique identifier for logical endpoint identification. This number, plus the *bEndpointAddress* of the associated Logical Endpoint Descriptor must be unique on the device (across all combinations of active interfaces and alternate settings of the selected configuration).
3	*bLogicalPacketCount*	1	Number	The number of Logical Packet Info members (Logical Packet ID/Flags/Size) to follow.
4-n	*Logical Packet Info Record [n]*	4	Logical Packet Info Records	Additional Logical Packet info records.

Figure 5-3: Logical Packet Descriptor Format

5.5 Logical Control & Data Addressing

The following figures define the format of the various data and flow control packet formats. The same general format is used for both channel data and flow control information. The first byte contains the Packet ID and control information. The most significant bit is a flag to indicate whether the data payload is data or flow control information. If set, the payload is flow control. If not set, the payload is channel data. Three figures display the format of fixed data, variable data and flow control packets, respectively.

5.5.1 Fixed Length Data Packet

If the Flow Control bit in the bmControl member indicates that this is a Data packet and the Fixed/Variable Data flag in the bmLogicalPacketFlags member of the Logical Packet descriptor indicates that the associated Packet ID represents a Fixed sized packet then the following format will be used.

Offset	Field	Size	Value	Description
0	*bmControl*	1	Bit Mask	Bit mask used to identify function of header. Bit 7 Flow Control 0 = Data packet (See below for details) Bit 6..0 Logical Packet ID (See below for details).
1-n	*Packet Data payload*	Variable	Byte Stream	Data payload associated with the preceding logical packet header.

Figure 5-4: Fixed Length Logical Data Packet Format

5.5.2 Variable Length Data Packet

If the Flow Control bit in the bmControl member indicates that this is a Data packet and the Fixed/Variable Data flag in the bmLogicalPacketFlags member of the Logical Packet descriptor indicates that the associated Packet ID represents a Variable sized packet then the following format will be used.

Offset	Field	Size	Value	Description
0	*bmControl*	1	Bit Mask	Bit mask used to identify function of header. Bit 7 Flow control 0 - Data packet (See below for details) Bit 6..0 Logical Packet ID (See below for details).
1	*wPayloadSize*	2	Number	If the Logical Packet ID identifies this packet as a variable size message then this field indicates the length of the data payload.
3-n	*Packet Data payload*	Variable	Byte Stream	The Data payload associated with the preceding logical packet header.

Figure 5-5: Variable length Logical Data Packet Format

5.5.3 Flow Control Packet

If the Flow Control bit in the bmControl member indicates that this is a Flow Control packet then the following format will be used.

Offset	Field	Size	Value	Description
0	*bmControl*	1	Bit Mask	Bit mask used to identify function of header. Bit 7 Flow Control 1 - Flow Control packet (See below for details) Bit 6..0 Logical Packet ID (See below for details).
1	*bOpCode*	1	Number	If this is a Flow Control Packet then this field contains a single byte Flow Control OpCode. See Figure 5-7.
2-n	*Flow Control payload*	Variable	Byte Stream	The Flow Control payload associated with the preceding Flow Control OpCode.

Figure 5-6: Flow Control Packet Format

bmControl

Bits 0..6 Message ID – This number used as a unique identifier for message identification. This number, plus the bEndpointAddress must be unique on the device (across all combinations of active interfaces and alternate settings). Message ID can be between 1 and 127, 0 is reserved.

Bit 7 Flow Control – If this bit is set (1) then this packet conforms to the Flow Control packet format. If this bit is set (0) then this packet conforms to a Data packet format. The Fixed/Variable Data flag in the bmLogicalPacketFlags member of the Logical Packet descriptor of the associated Packet ID indicates whether the Variable or Fixed Data packet format is appropriate.

Note: if there is more than one recipient in a given transfer, each set of data has its own wrapper and all of the data that will fit is concatenated into one physical packet.

Figure 5-7 defines the opcodes for the Flow Control packet version of the logical packet header defined in Figure 5-4, Figure 5-5 and Figure 5-6. The unit of *wPayloadSize* is bytes.

OpCode	Name	Payload Size	Description
0	*Reserved*	0	This code is reserved.
1	*Buffer_Grant*	2	This opcode is used to communicate grants from a target to a source. Units of the grant value is integral number of logical packets.
2	*Stall*	0	This opcode is used by a device to communicate a Halt condition on a logical endpoint. The payload size must be zero
3-255	*Reserved*	n/a	These codes are reserved for future use.

Figure 5-7: Logical Flow Control OpCode Definitions