Universal Serial Bus
Device Class Definition
for
Video Devices:
Frame Based Payload

Revision 1.1

June 1, 2005

Contributors

Abdul R. Ismail	Intel Corp.
Akihiro Tanabe	Canon Inc.
Anand Ganesh	Microsoft Corp.
Andy Hodgson	STMicroelectronics
Anshuman Saxena	Texas Instruments
Bertrand Lee	Microsoft Corp.
Charng Lee	Sunplus Technology Co., Ltd
David Goll	Microsoft Corp.
Eric Luttmann	Cypress Semiconductor Corp.
Fernando Urbina	Apple Computer Inc.
Geert Knapen	Philips Electronics
Geraud Mudry	Logitech Inc.
Hiro Kobayashi	Microsoft Corp.
Jean-Michel Chardon	Logitech Inc.
Jeff Zhu	Microsoft Corp.
Ken-ichiro Ayaki	Fujifilm
Mitsuo Niida	Canon Inc.
Nobuo Kuchiki	Sanyo Electric Co., Ltd
Olivier Lechenne	Logitech Inc.
Paul Thacker	STMicroelectronics
Remy Zimmermann	Logitech Inc.
Shinichi Hatae	Canon Inc.
Steve Miller	STMicroelectronics
Tachio Ono	Canon Inc.
Takashi Sato	Philips Electronics
Yoichi Hirata	Matsushita Electric Industrial Co., Ltd

Revision History

Version	Date	Description
1.1	June 1st, 2005	Initial release

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This document defines a general extensibility mechanism by which vendors can support Frame Based payload formats not defined by the *USB Device Class Definition for Video Devices* standard payload format specifications. A Frame Based payload format is any format where the video data is transferred as a sequence of individual video images (or frames), where each frame shares properties such as aspect ratio, bit depth, dimensions, etc.

1.2 Scope

The payload format and associated header information are fully specified in this document. This includes:

- USB Video Class payload header
- List of format descriptors

1.3 Related Documents

USB Specification Revision 2.0, April 27, 2000, www.usb.org
USB Device Class Definition for Video Devices www.usb.org

2 Video Class-Specific Information

2.1 Payload Header

The following is a description of the payload header format for Frame Based formats.

Table 2-1 Header Format for Frame Based Streams

HLE	Header Length							
BFH [0]	EOH	ERR	STI	RES	SCR	PTS	EOF	FID
PTS	PTS [7:0]							
	PTS [15:8]							
	PTS [23:16]							
	PTS [31:24]							
SCR	SCR [7:0]							
	SCR [15:8]							
	SCR [23:16]							
	SCR [31:24]							
	SCR [39:32]							
	SCR [47:40]							

Header length field

The header length field specifies the length of the header, in bytes.

Bit field header field

FID: Frame Identifier

This bit toggles at each frame start boundary and stays constant for the rest of the frame.

EOF: End of Frame

This bit indicates the end of a video frame and is set in the last video sample belonging to a frame. The use of this bit is an optimization to reduce latency in completion of a frame transfer, and is optional.

PTS: Presentation Time Stamp

This bit, when set, indicates the presence of a PTS field.

SCR: Source Clock Reference

This bit, when set, indicates the presence of a SCR field.

RES: Reserved.

Set to 0.

STI: Still Image

This bit, when set, identifies a video sample as belonging to a still image.

ERR: Error Bit

This bit, when set, indicates an error in the device streaming.

EOH: End of Header

This bit, when set, indicates the end of the BFH fields.

PTS: Presentation Time Stamp, Size: 4 bytes, Value: Number
The PTS field is present when the PTS bit is set in the BFH[0] field. See Section 2.4.3.3 "Video and Still Image Payload Headers" in the *USB Device Class Definition for Video Devices* specification.

SCR: Source Clock Reference, Size: 6 bytes, Value: Number
The SCR field is present when the SCR bit is set in the BFH[0] field. See Section 2.4.3.3 "Video and Still Image Payload Headers" in the *USB Device Class Definition for Video Devices* specification.

3 Payload-Specific Information

The Color Matching descriptor is mandatory for Frame Based video formats. For detailed information, see section 3.9.2.6, "Color Matching Descriptor" in the *USB Device Class Definition for Video Devices* specification.

3.1 Descriptors

This section provides detailed information about the following descriptors:

- Frame Based Payload Video Format Descriptor
- Frame Based Payload Frame Descriptor

3.1.1 Frame Based Payload Video Format Descriptor

The Frame Based Payload Video Format descriptor defines the characteristics of a specific video stream. It is used for formats that carry Frame Based Payload video information.

A Terminal corresponding to a USB IN or OUT endpoint, and the interface it belongs to, supports one or more format definitions. To select a particular format, host software sends control requests to the corresponding interface.

The **bFormatIndex** field contains the one-based index of this format descriptor, and is used by requests from the host to set and get the current video format.

The **guidFormat** field uniquely identifies the video data format that shall be used when communicating with this interface at the corresponding format index. For a video source function, the host software will deploy the corresponding video format decoder (if necessary) based on the format specified in this field.

The **bAspectRatioX** and **bAspectRatioY** fields specify the X and Y dimensions of the picture aspect ratio, respectively. For example, **bAspectRatioX** will be 16 and **bAspectRatioY** will be 9 for a 16:9 display.

A Frame Based Payload Video Format Descriptor is followed by one or more Frame Based Payload Video Frame Descriptor(s); each Video Frame Descriptor conveys information specific to a frame size supported for the format.

A Frame Based Payload Video format descriptor identifies the following.

Table 3-1 Frame Based Payload Video Format Descriptor

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this descriptor in bytes: 28
1	**bDescriptorType**	1	Constant	CS_INTERFACE descriptor type
2	**bDescriptorSubtype**	1	Constant	VS_FORMAT_FRAME_BASED descriptor subtype
3	**bFormatIndex**	1	Number	Index of this format descriptor

4	**bNumFrameDescriptors**	1	Number	Number of frame descriptors following that correspond to this format
5	**guidFormat**	16	GUID	Globally Unique Identifier used to identify stream-encoding format
21	**bBitsPerPixel**	1	Number	Number of bits per pixel used to specify color in the decoded video frame. May be zero if not applicable.
22	**bDefaultFrameIndex**	1	Number	Optimum Frame Index (used to select resolution) for this stream
23	**bAspectRatioX**	1	Number	The X dimension of the picture aspect ratio.
24	**bAspectRatioY**	1	Number	The Y dimension of the picture aspect ratio.
25	**bmInterlaceFlags**	1	Bitmap	Specifies interlace information. If the scanning mode control in the Camera Terminal is supported for this stream, this field shall reflect the field format used in interlaced mode. (Top field in PAL is field 1, top field in NTSC is field 2.): D0: Interlaced stream or variable. 1 = Yes D1: Fields per frame. 0= 2 fields, 1 = 1 field D2: Field 1 first. 1 = Yes D3: Reserved D5..4: Field pattern 00 = Field 1 only 01 = Field 2 only 10 = Regular pattern of fields 1 and 2 11 = Random pattern of fields 1 and 2 D7..6: Reserved. Do not use.
26	**bCopyProtect**	1	Boolean	Specifies whether duplication of the video stream is restricted: FALSE (0): No restrictions TRUE (1): Restrict duplication

27	**bVariableSize**	1	Boolean	Specifies whether the data within the frame is of variable length from frame to frame. TRUE (1): Variable Size FALSE (0): Fixed Size If **bVariableSize** is TRUE (1), then **dwBytesPerLine** (below) must be set to zero (0).

3.1.2 Frame Based Payload Frame Descriptor

Frame Based Payload Video Frame descriptors (or Frame descriptors for short) are used to describe the decoded video and still-image frame dimensions and other frame-specific characteristics supported by a particular stream. One or more Frame descriptors follow the Frame Based Payload Video Format descriptor they correspond to. The Frame descriptor is also used to determine the range of frame intervals supported for the frame size specified.

The Frame Based Payload Video Frame descriptor is used only for video formats for which the Frame Based Payload Video Format descriptor applies (see section 3.1.1, "Frame Based Payload Video Format Descriptor").

The **bFrameIndex** field contains the one-based index of this frame descriptor, and is used by requests from the host to set and get the current frame index for the format in use. This index is one-based for each corresponding format descriptor supported by the device.

The range of frame intervals supported can be either a continuous range or a discrete set of values. For a continuous range, **dwMinFrameInterva**l, **dwMaxFrameInterval** and **dwFrameIntervalStep** indicate the limits and granularity of the range. For discrete values, the **dwFrameInterval(x)** fields indicate the range of frame intervals (and therefore frame rates) supported at this frame size. The frame interval is the average display time of a single decoded video frame in 100ns units.

A Frame descriptor identifies the following.

Table 3-2 Frame Based Payload Video Frame Descriptors

Offset	Field	Size	Value	Description
0	**bLength**	1	Number	Size of this descriptor in bytes when bFrameIntervalType is 0: 38 Size of this descriptor in bytes when bFrameIntervalType > 0: 26+(4*n)
1	**bDescriptorType**	1	Constant	CS_INTERFACE descriptor type

2	**bDescriptorSubtype**	1	Constant	VS_FRAME_FRAME_BASED descriptor subtype
3	**bFrameIndex**	1	Number	Index of this frame descriptor
4	**bmCapabilities**	1	Number	D0: Still image supported Specifies whether still images are supported at this frame setting. This is only applicable for VS interfaces with an IN video endpoint using Still Image Capture Method 1, and should be set to 0 in all other cases. D1: Fixed frame-rate Specifies whether the device provides a fixed frame rate on a stream associated with this frame descriptor. Set to 1 if fixed rate is enabled; otherwise, set to 0. D7..2: Reserved, set to 0.
5	**wWidth**	2	Number	Width of decoded bitmap frame in pixels
7	**wHeight**	2	Number	Height of decoded bitmap frame in pixels
9	**dwMinBitRate**	4	Number	Specifies the minimum bit rate at the longest frame interval in units of bps at which the data can be transmitted.
13	**dwMaxBitRate**	4	Number	Specifies the maximum bit rate at the shortest frame interval in units of bps at which the data can be transmitted.
17	**dwDefaultFrameInterval**	4	Number	Specifies the frame interval the device would like to indicate for use as a default. This must be a valid frame interval described in the fields below.
21	**bFrameIntervalType**	1	Number	Indicates how the frame interval can be programmed: 0: Continuous frame interval 1..255: The number of discrete frame intervals supported (n)

22	**dwBytesPerLine**	4	Number	Specifies the number of bytes per line of video for packed fixed frame size formats, allowing the receiver to perform stride alignment of the video. If the **bVariableSize** value (above) is TRUE (1), or if the format does not permit such alignment, this value shall be set to zero (0).
26…				See the following frame interval tables.

Table 3-3 Continuous Frame Intervals

Offset	Field	Size	Value	Description
26	**dwMinFrameInterval**	4	Number	Shortest frame interval supported (at highest frame rate), in 100 ns units.
30	**dwMaxFrameInterval**	4	Number	Longest frame interval supported (at lowest frame rate), in 100 ns units.
34	**dwFrameIntervalStep**	4	Number	Indicates granularity of frame interval range, in 100 ns units.

Table 3-4 Discrete Frame Intervals

Offset	Field	Size	Value	Description
26	**dwFrameInterval**(1)	4	Number	Shortest frame interval supported (at highest frame rate), in 100 ns units.
…	…	…	…	…
26+(4*n)-4	**dwFrameInterval**(n)	4	Number	Longest frame interval supported (at lowest frame rate), in 100 ns units.

3.2 Video Samples

Each Frame Based Payload frame is considered a single video sample. A video sample is made up of one or more *payload transfers* (as defined in the USB Device Class Specification for Video Devices).

For an isochronous pipe, each (micro) frame will contain a single payload transfer. Each payload transfer will consist of a payload header immediately followed by payload data in one or more data transactions (up to 3 data transactions for high speed high bandwidth endpoints).

For a bulk pipe, the first bulk data packet of each payload transfer shall contain a payload header at the beginning of the packet, followed by payload data, extending through additional bulk data transactions as needed.

4 Examples

4.1 Isochronous Transfer IN

The following example shows the relationship between Video Samples, Payload Transfers and the token and data packets when receiving isochronous transfers from the device. This example shows high-speed, high-bandwidth transfers, but this is only illustrative and not a requirement of the Frame Based Paylaod payload format. The actual video sample size and bandwidth usage will vary according to the requirements of the device.

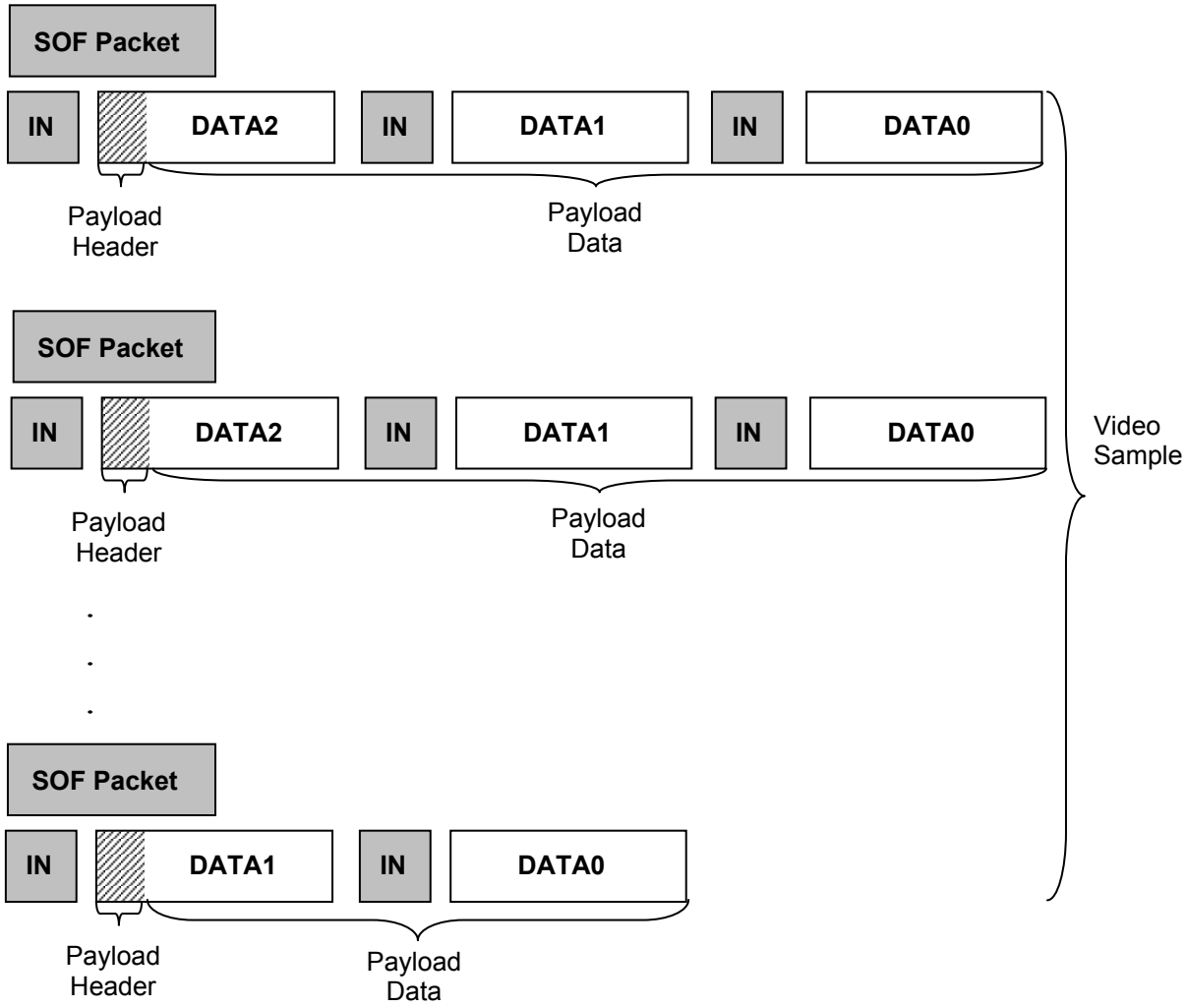

Figure 4-1 Example Frame Based Payload Isochronous Transfer, IN Endpoint

4.2 Isochronous Transfer OUT

The following example shows the relationship between Video Samples, Payload Transfers and the token and data packets when sending isochronous transfers to the device. This example shows high-speed, high-bandwidth transfers, but this is only illustrative and not a requirement of the Frame Based Paylaod payload format. The actual video sample size and bandwidth usage will vary according to the requirements of the device.

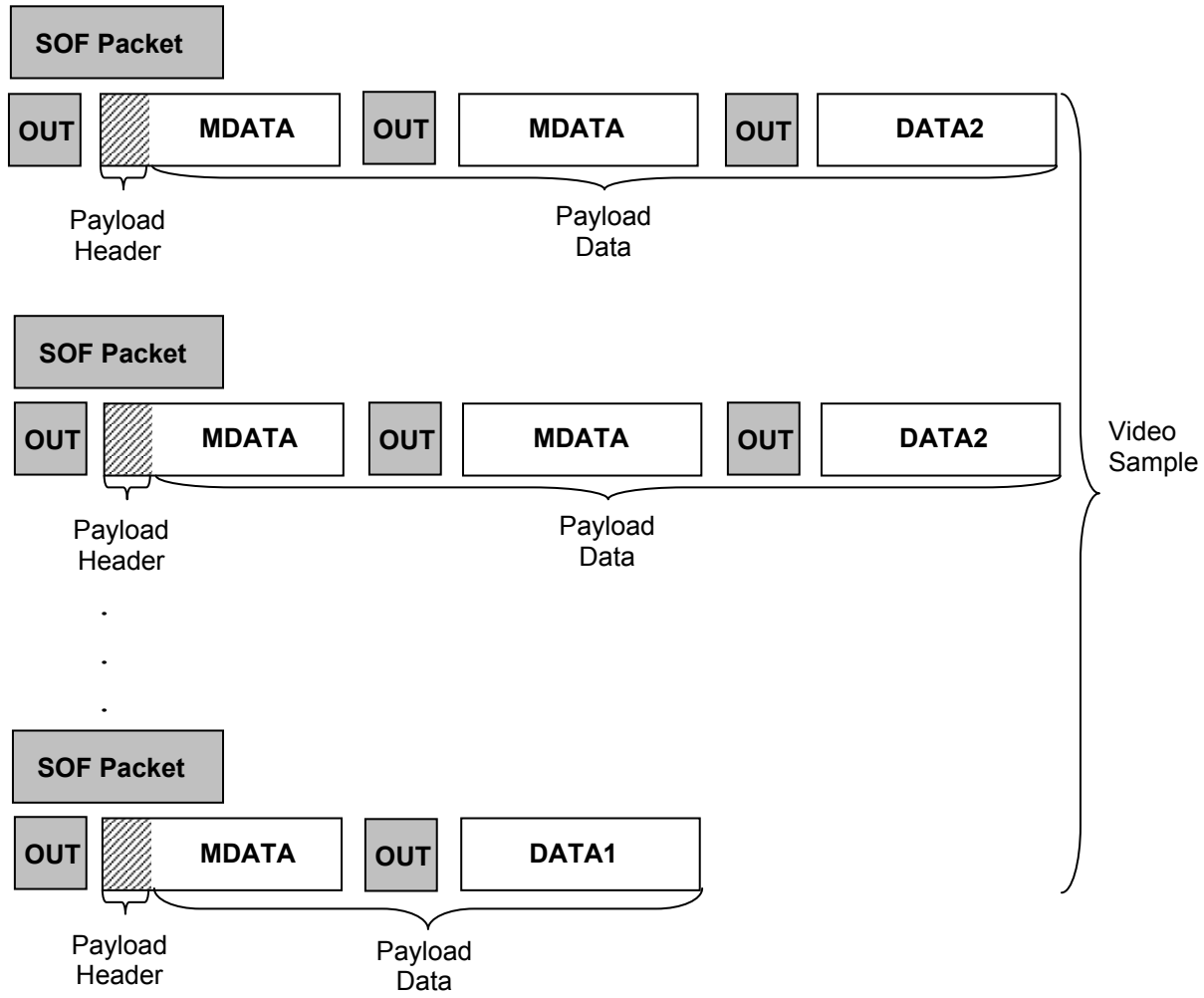

Figure 4-2 Example Frame Based Payload Isochronous Transfer, OUT Endpoint

4.3 Bulk Transfer IN

The following example shows the relationship between Video Samples, Payload Transfers and the token and data packets when receiving bulk transfers from a device. Handshake packets are not shown for the sake of clarity.

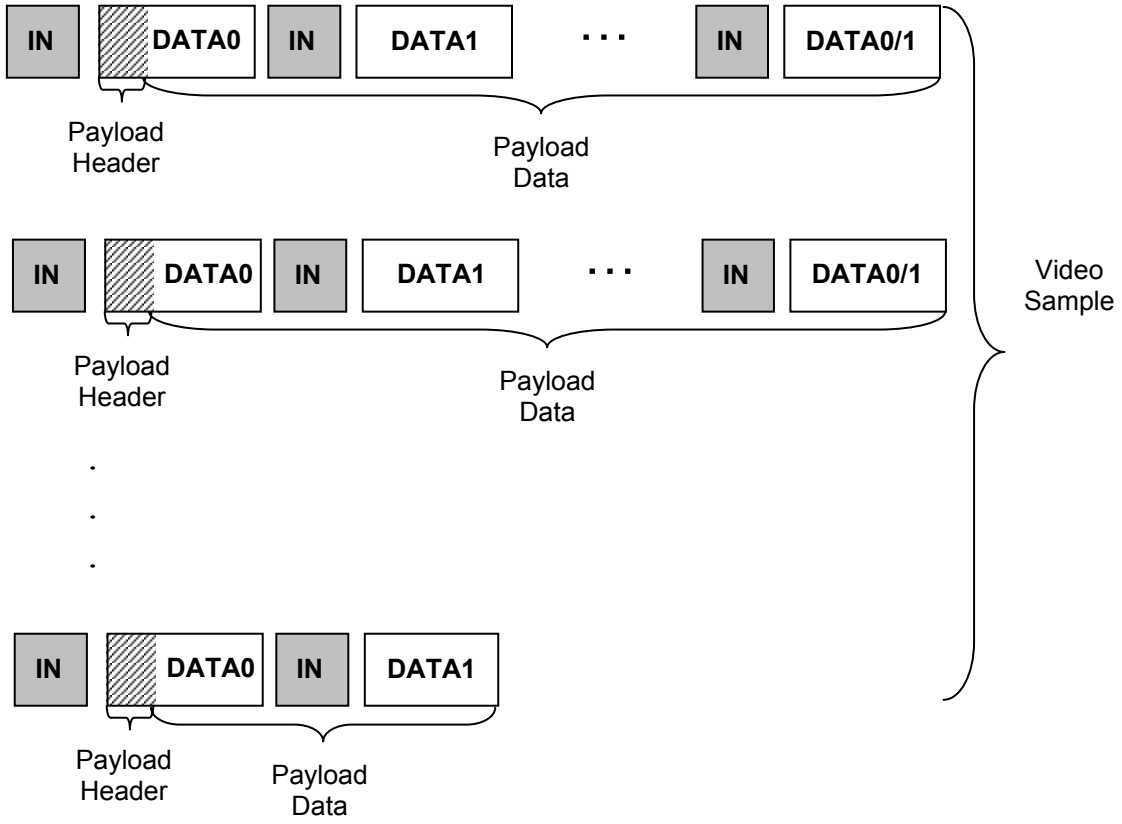

Figure 4-3 Example Frame Based Payload Bulk Transfer, IN Endpoint

4.4 Bulk Transfer OUT

The following example shows the relationship between Video Samples, Payload Transfers and the token and data packets when sending bulk transfers to the device. Handshake packets are not shown for the sake of clarity.

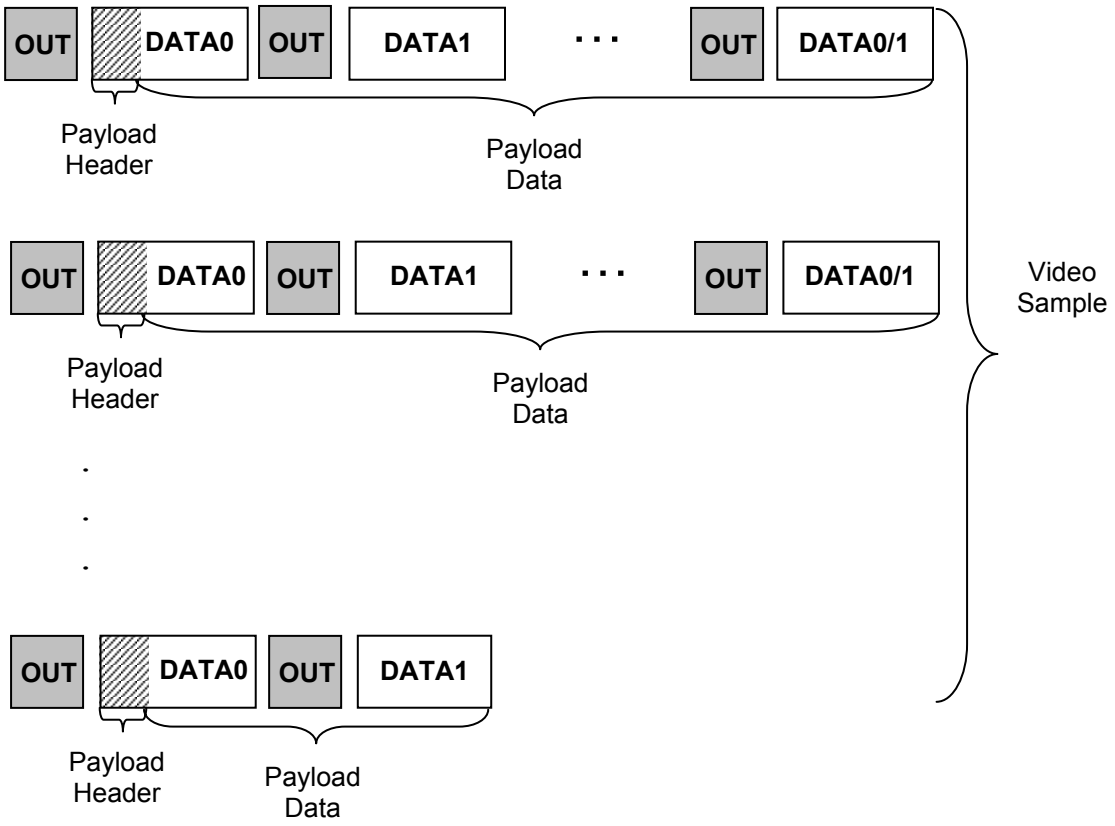

Figure 4-4 Example Frame Based Payload Bulk Transfer, OUT Endpoint