Universal Serial Bus
Device Class Definition
for
Video Devices:
H.264 Payload

Revision 1.00

April 26, 2011

Contributors

Ross Cutler	Microsoft Corporation
Ming-Chieh Lee	Microsoft Corporation
Stephen Cooper	Microsoft Corporation
Maribel Figuera	Microsoft Corporation
Richard Webb	Microsoft Corporation
Andrei Jefremov	Skype
Remy Zimmermann	Logitech Inc.
Venkatesh Tumatikrishnan	Logitech Inc.
Oliver Hoheisel	Logitech Inc.
Chandrashekhar Rao	Logitech Inc.
Michael Cheng	Logitech Inc.

AVC/H.264 Disclaimer

Any implementation of the specification described herein would require a MPEG LA AVC/H.264 Patent Portfolio license to essential patent rights for the AVC/H.264 (MPEG-4 Part 10) digital video coding standard. See http://www.MPEGLA.com.

Revision History

Version	Date	Description
0.1	July 12, 2010	Initial Draft
0.2	July 15, 2010	Updated after review, added Slice mode, size and format
0.3	July 16, 2010	Added format MJPEG, Preview flipped info Removed wWidth and wHeight modulo 16 Added bPreviewFlipped Removed P and B from Picture type control Removed noise filtering Added Crop configuration
0.40	July 29, 2010	Added application group based configuration
0.41	August 10, 2010	Clarification of H.264+YUY2 and CABAC option
0.42	August 16, 2010	Update based on meeting comments
0.43	August 26, 2010	Added multiple configurations on query support.
0.44	August 31, 2010	Removed fast config as per meeting discussion
0.45	September 09,2010	Added examples updated Table 2
0.46	September 16,2010	Clean up, clarification for Multiplexed Payload and profile_idc
0.47	September 22, 2010	Added detail on use of GET_MAX
0.48	September 23, 2010	Added UCIF types to bUsageType for UCIF approved usage types
0.49	September 27,2010	Examples in Visio, added text for table, profile_id, App4 and 4CC
0.50	September 29,2010	Updated flow chart examples with GET_MAX, added mux option clarifications.
0.51	October 4,2010	Figure replaced with Visio. Table text updates
0.52	October 13, 2010	Update Table 7, 9 and 2 Idr, B frame I frame periodicity
0.53	October 14, 2010	Added picture timing SEI messages requirement
0.54	October 14, 2010	Added wEstimatedVideoDelay

0.55	October18, 2010	Added wDelay for payload, frame rate update and QP min/max Updated txt and visio
0.56	October 26,2010	Text updates
0.60	October 27,2010	Table 2 PROBE/COMMIT options , Table 8 NumLayers reserve field and Updated examples,
0.61	October 27, 2010	Added buffering period SEI messages and HRD conformance
0.62	November 3,2010	Updated Table 9 for all the frames. Added little endian to 3.3 and added GET_MAX field requirement
0.63	November 10,2010	Updated for default config after stream ends. Note for bStreamMuxOption. Added reference.
0.64	November 11, 2010	Added wEstimatedMaxConfigDelay to Table 2
0.80	November 12,2010	Table 2 bMaxLayer changed to bMaxSpatialLayer Table 2 split into probe table 2 and commit Table 3 Table 9 wNumLayer has option of I and P settings Added Example 5.4 for SVC Added Text to Table 5
0.81	November 30,2010	Added clarification text to Table 5
0.82	December 3,2010	Updated bitrate Table 8, UCCONFIG, header format with PTS bInterFrmNum to bNumOfReorderFrames added bView in Table 2, bMinQp/bMaxQp changed to signed and version format.
0.83	December 15, 2010	Added Audio Video Synchronization
0.84	December 15, 2010	Merged the changes and comments made by the technical writer into v0.83 and created v0.84. Updated the Buffering Period and Picture Timing SEI messages section.
0.85	December 16,2010	Resolved the comments, add table 14, added bit 4 and bit 7 bStreamMuxOption for Simulcast and Max.
0.86	December 17,2010	For release
0.87	January 17, 2011	Clarification for payload and updated SVC example

0.87a	January 20, 2011	Added NV12 Definition, Remove Spatial_Rewrite modes from PROBE and COMMIT table 2 and table 6. Minor Typos and syntax fixes.
0.87b	February 07,2011	Updated based on Raleigh F2F, changed tables 1, 2, 3, 8, 9 and added table 14. Add more comments. LayerID has been added.
0.87c	February 08, 2011	Added bStreamID in Table 2 and updated bStreamMuxOption
0.88	February 12, 2011	Updated examples and comments.
088a	February 22,2011	StreamID and LayerID has been updated after discussion and aligned with H.264 specification. Added SVC info.
0.88b	February 28, 2011	wLayerId comments updates, in/out updates, removed crop added clarification GET_CUR for table 8 and table 9
0.88c	March 15,2011	Added GET_MAX, GET_MIN to dynamic control. Example updated for SVC.
0.88d	March 18, 2011	Added LTR proposal, Added UVCX_LTR_BUFFER_SIZE_CONTROL, UVCX_PICTURE_LTR_CONTROL and UVCX_ENCODER_RESET, Table 10 LTR removed. bStreamMuxOption bit 6 used.
0.90	April 4, 2011	Updated LTR, Frame Interval clarifications, Added comments for QP and bitrate GET_CUR and config index.
0.91	April 8, 2011	Reykjavik F2F review and updates
0.92	April 15, 2011	Review and updates
0.93	April 21, 2011	Updates after CC , Added wLayerID to dynamic tables.
0.94	April 22, 2011	Updated based on SVC and LTR review. UVCX_ENCODER_RESET is removed from dynamic control.
1.00	April 26, 2011	Removed reserve fields from XU Control, Aligned the order of XU control with the table 1.

Table of Contents

List of Tables

List of Figures

1 Introduction

1.1 Purpose

This specification describes H.264 (ISO/IEC 14496 Part 10/ITU-T H.264 AVC, SVC & MVC) specific UVC device payload and interface. Devices supporting H.264 encoding are able to interface with the host using defined controls and video streaming interface(s). The document describes the method of getting capabilities of the device and configuring it. It further describes the supported video streaming payload formats: Frame based Payload and Stream based Payload.

In order to address current and future capabilities and limitations, this specification supports different payload types, as follows:

- H.264 Payload Format
- Multiplexed Payload Format (MPF)

Support of multiple video streaming interfaces follows the UVC specification allowing different use cases. The following example (Figure 1) shows a USB H.264 device with an uncompressed video streaming interface for Preview and one H.264 video streaming interface for network communication.

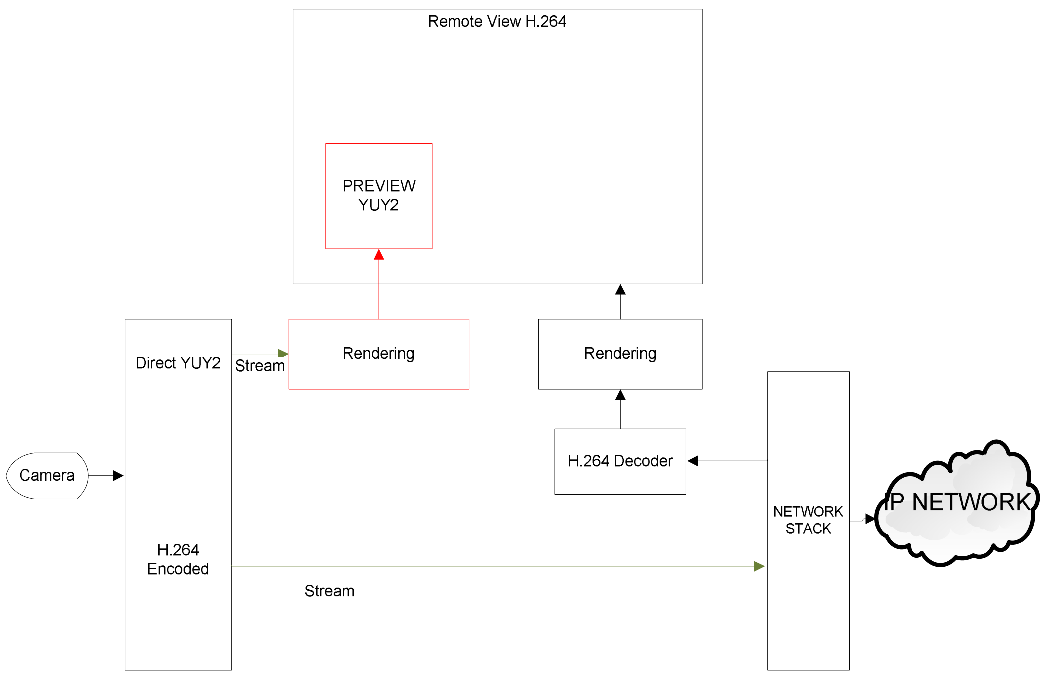


Figure 1 Overview

1.2 Scope

The control and payload specifications are described in this document. This includes:

- Stream Based H.264 Payload Format
- Frame Based H.264 Payload Format
- MJPEG Based Payload for Multiplexed Payload Format
- H.264 Encoder Extension Unit and Associated Controls

1.3 Related Documents

[1] USB Video Class 1.1 (http://www.usb.org/developers/devclass_docs#approved)

[2] USB_Video_Payload_Frame_Based_1.1

[3] USB_Video_Payload_Stream_Based_1.1

[4] USB_Video_Payload_MJPEG_1.1

[5] RTP Payload for H.264 (http://tools.ietf.org/html/rfc3914)

[6] ITU H.241 (http://www.itu.int/itu-t/recommendations/index.aspx?ser=H)

[7] ITU T.81 (http://www.itu.int/itu-t/recommendations/index.aspx?ser=T)

[8]The H.264/MPEG-4 AVC standard (http://www.itu.int/rec/T-REC-H.264) (referred to hereafter simply as H.264) is specified in the following document:

a. ITU-T Rec. H.264 | ISO/IEC 14496-10 Advanced video coding for generic audiovisual services. The standard is available at. Unless otherwise specified, this document refers to the edition approved by ITU-T in March 2010 (posted at the ITU-T web site link above).
b. The Scalable Video Coding (SVC) extensions to the H.264/MPEG-4 AVC standard (referred to hereafter simply as SVC) are specified in Annex G of the above document.
c. The Multiview Video Coding (MVC) extensions to the H.264/MPEG-4 AVC standard (referred to hereafter simply as MVC) are specified in Annex H of the above document.

[9] When supported, the use of SVC and simulcast of multiple streams in the context of this specification shall additionally conform to the following specification:

a. Unified Communication Specification and Interfaces for H.264/MPEG-4 AVC and SVC Encoder Implementation.
b. The specification is available at http://technet.microsoft.com/en-us/lync (Unified Communication Specification for H.264 AVC and SVC Encoder Implementation). Unless otherwise

specified, this document refers to the edition of version 1.01 (posted at the Microsoft web site link above).

1.4 Glossary

Term	Definition
AVC	Advanced Video Coding (see H.264)
CABAC	Context-based Adaptive Binary Arithmetic Coding
CAVLC	Context-based Adaptive Variable Length Coding
CBR	Constant Bit Rate
CPB	Coded Picture Buffer
DPB	Decoded Picture Buffer
H.264	ISO/IEC 14496 Part 10
IDR	Instantaneous Decoder Refresh. Intraframe with no past reference.
LTR	Long Term Reference
MB	Macroblock
MJPG	Motion JPEG. See UVC standard reference payload specification.
MPF	Multiplexed Payload Format
MVC	Multiview Video Coding
NAL	Network Abstract Layer
NALU	Network Access Layer Unit
NV12	Planar 4:2:0 format with Y-plane followed by plane of interleaved U/V (see http://www.fourcc.org/yuv.php#NV12)
PPS	Picture Parameter Set
QP	Quantization Parameter
SCR	Source Clock Reference
SEI	Supplemental Enhancement Information

SPS	Sequence Parameter Set
SVC	Scalable Video Coding
USB	Universal Serial Bus
UVC	USB Video Class
VBR	Variable Bit Rate
VC	Video Control
VS	Video Streaming
VUI	Video Usability Information
XU	Extension Unit
YUY2	Interleaved 16-bit YUV data. Y, U, Y, V.

2 Functional Characteristics

2.1 H.264 Payload Format

The H.264 Payload Format is exposed through a standard UVC Video Streaming Interface according to the Stream Based or Frame Based payload specifications (see [2] or [3]). The devices can have additional streams to support other video payload formats (see Figure 2); the example configuration below uses one video control (VC) and three video streaming (VS) interfaces: Uncompressed, MJPEG and H.264.

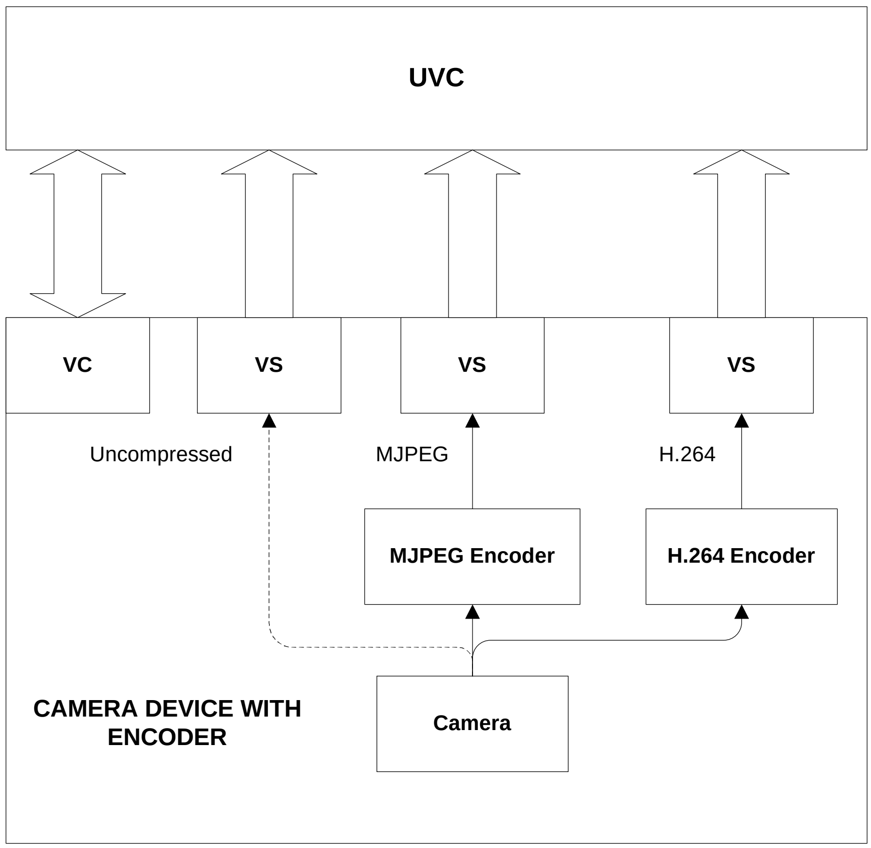


Figure 2 Video Stream Interfaces

2.2 Multiplexed Payload Format

The Multiplexed Payload Format allows supporting multiple payloads formats on a single video Streaming Interface; the MPF is exposed to the video streaming interface as a MJPEG Payload (see [4]) and optionally encapsulates H.264 and/or Uncompressed.

The example configuration shown in Figure 3 has one video control and one video streaming interface for MPF.

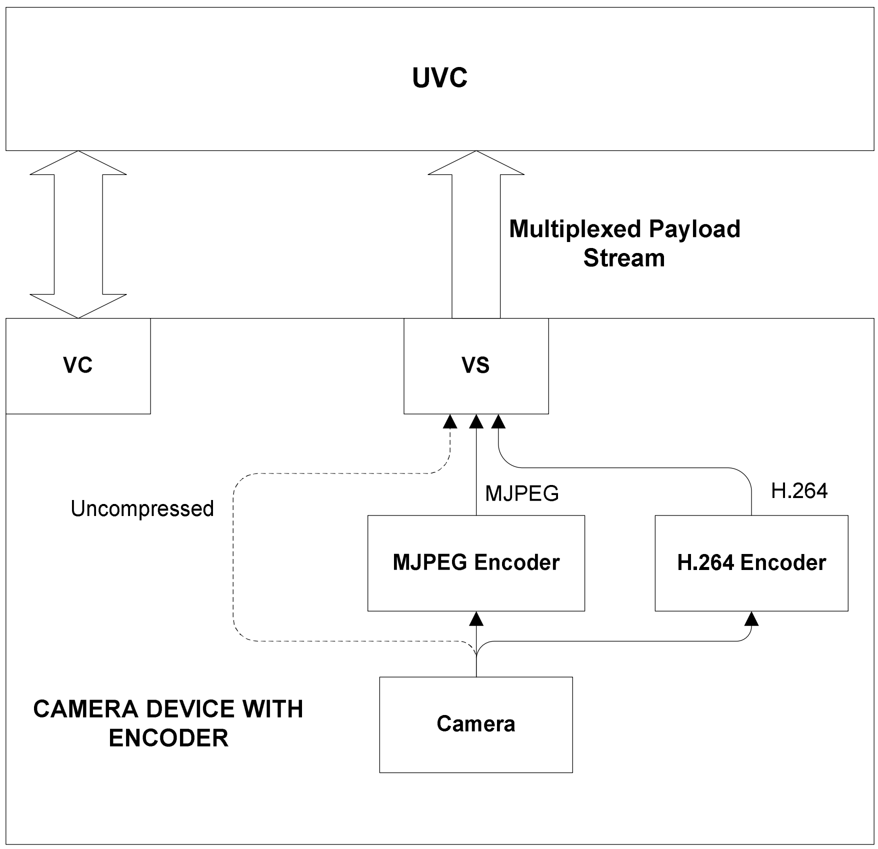


Figure 3 Multiplexed Stream

3 H.264 Interface

3.1 UVC Probe and Commit

3.1.1 Format Negotiation

The regular UVC Probe and Commit negotiation uses only parameters needed for Frame Based payloads; stream-based parameters wKeyFrameRate, wPFrameRate, wCompQuality, wCompWindowSize and wDelay are ignored and are instead defined and implemented using H.264 extension units.
The device may support multiple stream and the configurations methods are as follows:

3.1.1.1 H.264 Payload Format

In this scenario, it is assumed that the Video Streaming Interface supports only the H.264 Payload Format.

- Use UVCX_VIDEO_CONFIG to find a set of parameters that the VS interface is known to support. The individual parameters (e.g., frame rate, resolution, temporal scalability mode, etc.) can be degraded but not upgraded. The UVCX_VIDEO_CONFIG_PROBE/ UVCX_VIDEO_CONFIG_COMMIT setting for **bStreamMuxOption** value shall be set to 0.
- Proceed with regular UVC probe & commit.
- Use dynamic control (3.3.2) to change H.264 encoding parameters.

3.1.1.2 Multiplexed Payload Format

In this scenario, it is assumed that Video Streaming Interface can support multiplexed payloads simultaneously.

The VS Interface negotiation shall follow the sequence below:

- Use UVCX_VIDEO_CONFIG with GET_MAX and GET_CUR to find a set of parameters that the VS interface is known to support. The individual parameters (e.g., frame rate, resolution, temporal scalability mode, etc.) can be degraded but not upgraded. The UVCX_VIDEO_CONFIG_PROBE/ UVCX_VIDEO_CONFIG_COMMIT setting for **bStreamMuxOption** value shall be set to non-zero value. Bits 1-7 represent one or more preferred auxiliary streams to enable. If each embedded stream requires different settings then inform the device by calling UVCX_VIDEO_CONFIG multiple times with the required bit mask for bits 1-7. The second time, calling the function shall configure the secondary stream configuration and it shall not change the configuration of the primary stream. And so on for any additional streams.
- Proceed with regular UVC probe & Commit.
- Use dynamic controls (3.3.2) to change H.264 encoding parameters.

3.1.1.3 Scalable Video Coding

Scalable Video Coding (SVC) is primarily specified in Annex G of the H.264/MPEG-4 Advanced Video Coding (AVC) standard. Within a picture, there is one “base layer” that is formatted as an ordinary H.264/AVC coded picture and one or more additional scalable layer representations which each represent an additional “enhancement layer” of a SVC encoded bitstream for the same instant in time.

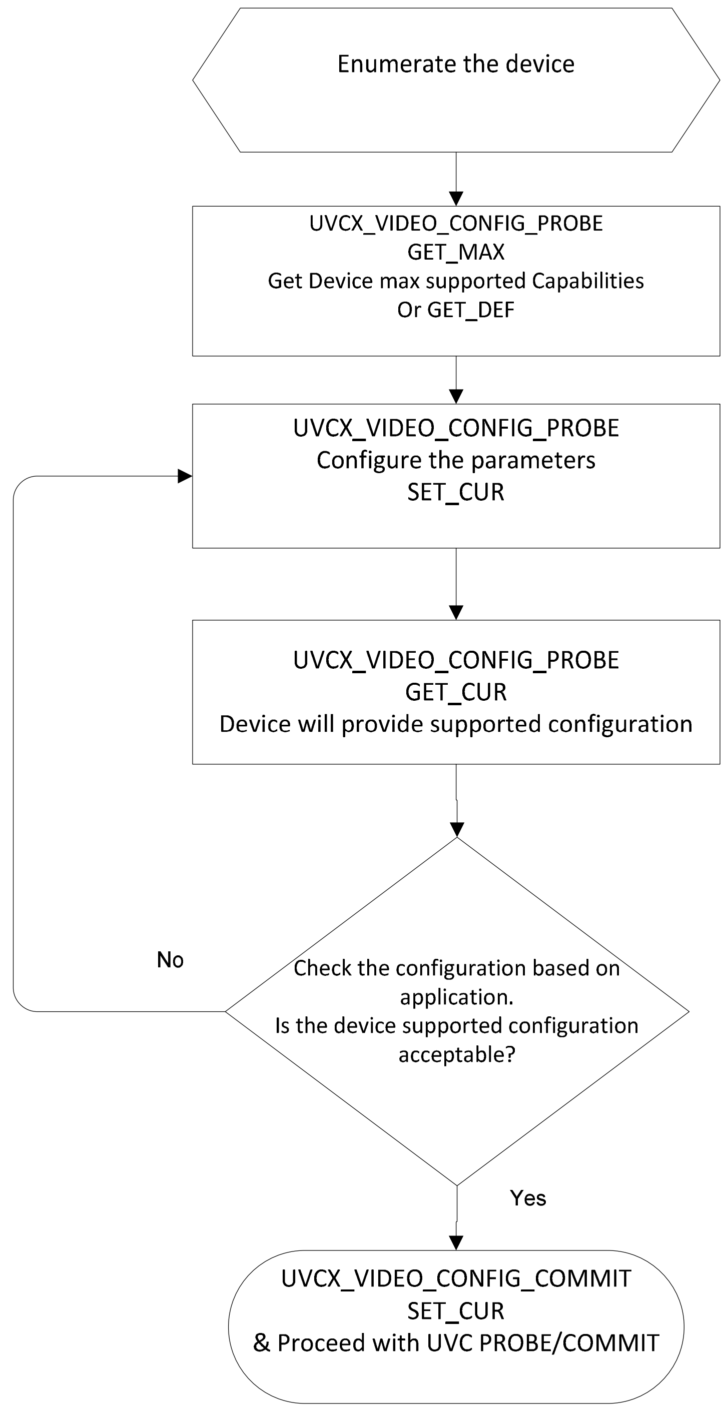
SVC supports three main types of classes of scalability: temporal, quality (or SNR), and spatial scalability where quality scalability can be further classified into Coarse Grained Scalability (CGS) and Medium Grained Scalability (MGS). A SVC bitstream may contain arbitrary combinations of these three classes of scalability. To simplify the design, this specification only considers the most commonly used layering structures as defined in the UCConfig Specification, summarizes as below:

- Temporal scalability is applied first in layering a SVC bitstream. A temporal layer is identified by the syntax element temporal_id for an H.264 NALU. The value of temporal_id must be assigned starting from 0 and increased continuously.
- Quality scalability is applied next in layering a SVC bitstream. A quality layer is identified by the syntax element dependency_id in CGS mode and quality_id in MGS mode for an H.264 NALU. The values of quality_id and dependency_id must be assigned starting from 0 and increased continuously. When MGS is used, an MGS layer is split into multiple sublayers by means of transform coefficient partitioning. CGS is effectively a special case of spatial scalability when two successive spatial layers have identical spatial resolutions.
- Spatial scalability is applied last in layering a SVC bitstream. A spatial layer is identified by the syntax element dependency_id in an H.264 NALU. Additional quality scalable layers may be applied in a spatial enhancement layer.

With the above constraints, for a particular layering structure the values of temporal_id, dependency_id, and quality_id associated with a layer can be determined without ambiguity and used as an unique identifier for that layer.

3.2 Programming Model

3.2.1 Configuration Model



The UVCX_VIDEO_CONFIG structure (See Table 2) shall be used to configure the H.264 encoder; however, the required configuration might not be supported by the device. GET_MAX shall provide the maximum capability of individual features defined in the UVCX_VIDEO_CONFIG, assuming those other features have been specified. GET_MAX does not return a supported configuration of the VS interface, but a summary of the maximum capabilities of the camera when each feature is considered separately. The GET_CUR shall provide a configuration that is supported by the VS interface. This configuration can subsequently be used in UVCX_VIDEO_CONFIG_COMMIT or in UVCX_VIDEO_CONFIG_PROBE for further negotiation.

The configuration of the stream shall be set to the default configuration after at the end of stream. The process should provide clearing the old negotiated configuration.

3.3 H.264 UVC Extensions Units (XUs)

The control and parameters shall be in Little Endian byte ordering.

Control Selector	Value	Comments
UVCX_VIDEO_UNDEFINED	0x00	Reserved
UVCX_VIDEO_CONFIG_PROBE	0x01	Negotiate encoding parameters without altering current streaming state
UVCX_VIDEO_CONFIG_COMMIT	0x02	Sets the current configuration of the encoder
UVCX_RATE_CONTROL_MODE	0x03	Configuration of the encoder in bitrate/quality mode.
UVCX_TEMPORAL_SCALE_MODE	0x04	Number of layers
UVCX_SPATIAL_SCALE_MODE	0x05	Setting the spatial mode
UVCX_SNR_SCALE_MODE	0x06	Setting the quality mode
UVCX_LTR_BUFFER_SIZE_CONTROL	0x07	LTR Buffer usage
UVCX_LTR_PICTURE_CONTROL	0x08	LTR Control
UVCX_PICTURE_TYPE_CONTROL	0x09	I , IDR frame requests
UVCX_VERSION	0x0A	Spec. version supported from the device
UVCX_ENCODER_RESET	0x0B	Encoder Reset
UVCX_FRAMERATE_CONFIG	0x0C	Dynamic frame rate configuration
UVCX_VIDEO_ADVANCE_CONFIG	0x0D	Configuration for level_idc
UVCX_BITRATE_LAYERS	0x0E	Bitrate per layer
UVCX_QP_STEPS_LAYERS	0x0F	Minimum/Maximum QP Configuration per layers

Table 1: Extension unit control selectors

3.3.1 UVCX_VIDEO_CONFIG_PROBE & UVCX_VIDEO_CONFIG_COMMIT

The UVCX_VIDEO_CONFIG_PROBE control shall be used to query the device to get supported configurations and negotiate the individual parameters.

The UVCX_VIDEO_CONFIG_COMMIT control is used to configure the device for streaming operation.

<table>
<tr><td colspan="2">Control Selector</td><td colspan="3">UVCX_VIDEO_CONFIG_PROBE

UVCX_VIDEO_CONFIG_COMMIT</td></tr>
<tr><td colspan="2">Mandatory Requests</td><td colspan="3">UVCX_VIDEO_CONFIG_PROBE valid options SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN

UVCX_VIDEO_CONFIG_COMMIT valid option SET_CUR</td></tr>
<tr><td colspan="2">wLength</td><td colspan="3">46</td></tr>
<tr><td>Offset</td><td>Field</td><td>Size</td><td>Value</td><td>Description</td></tr>
<tr><td>0</td><td>dwFrameInterval</td><td>4</td><td>Number</td><td>In 100ns frame interval

Note: This shall not be lower than the UVC_PROBE/COMMIT dwFrameInterval.</td></tr>
<tr><td>4</td><td>dwBitRate</td><td>4</td><td>Number</td><td>Average bits per second</td></tr>
<tr><td>8</td><td>bmHints</td><td>2</td><td>Bitmap</td><td>Advises what configuration parameter(s) should be maintained.

0x0001: Resolution (wHeight and wWidth)

0x0002: Profile (wProfile)

0x0004: Rate Control Mode (bRateControlMode)

0x0008: Usage Type (bUsageType)

0x0010: Slice Mode (wSliceMode)

0x0020: Slice Unit (wSliceUnits)

0x0040: MVC View (bView)

0x0080: Temporal (bTemporalScaleMode)

0x0100: SNR (bSNRScaleMode)

0x0200: Spatial (bSpatialScaleMode)

0x0400: Spatial Layer Ratio (bSpatialLayerRatio)

0x0800: Frame interval (dwFrameInterval)

0x1000: Leaky Bucket Size (wLeakyBucketSize)

0x2000: Bit Rate (dwBitRate)

0x4000: Entropy CABAC (bEntropyCABAC)

0x8000: I FramePeriod (wIFramePeriod)</td></tr>
</table>

10	wConfigurationIndex	2	Number	Configuration index, an increasing number from 1 to max wConfigurationIndex that increments for each subsequent GET_CUR. Note: The device shall return first index =1 on the first GET_CUR from host. If it wants to scan the next configuration, it sends GET_CUR again; SET_CUR selects any valid configuration index.
12	wWidth	2	Number	Encoder input image width in pixels. The resolution for SVC shall be set for the highest layer.
14	wHeight	2	Number	Encoder input image height in pixels. The resolution for SVC shall be set for the highest layer.
16	wSliceUnits	2	Number	The parameter defines the units of the wSliceMode. wSliceMode=0x0000: wSliceUnits ignored wSliceMode=0x0001**:** wSliceUnits in bits/slice wSliceMode=0x0002**:** wSliceUnits in MBs/slice wSliceMode=0x0003**:** wSliceUnits in slices/frame
18	wSliceMode	2	Number	0x0000 -> no multiple slices 0x0001 -> multiple slices - bits/slice, 0x0002 -> multiple slices-MBs/slice, 0x0003 -> number of slices per frame 0x0004-0xFFFF = Reserved

20	wProfile	2	Number	profile_idc as defined in H.264 specification. **Profiles** (Bits 8-15) 0x4200 -> Baseline Profile 0x4D00 -> Main Profile 0x6400 -> High Profile 0x5300 -> Scalable Baseline Profile 0x5600 -> Scalable High Profile 0x7600 -> Multiview High Profile 0x8000 -> Stereo High Profile **Constrained flags** (Bits 0-7) 0x0080 -> constraint_set0_flag 0x0040 -> constraint_set1_flag 0x0020 -> constraint_set2_flag 0x0010 -> constraint_set3_flag 0x0008 -> constraint_set4_flag 0x0004 -> constraint_set5_flag 0x0002 ->Reserved 0x0001 ->Reserved Example: **Profile using Constrained flags** 0x4240 -> Constrained Baseline
22	wIFramePeriod	2	Number	The time between IDR frames in milliseconds. 0x0000= No periodicity requirements for IDR frames.

24	wEstimatedVideoDelay	2	Number	Estimated time between the end of exposure and the presentation on the USB interface, in milliseconds.
26	wEstimatedMaxConfigDelay	2	Number	Estimated maximum time to change configuration modes, in milliseconds.
28	bUsageType	1	Number	Encoder Configuration based on the host configured usage type. 0x00: Reserved 0x01: Real-time (video conf) 0x02: Broadcast 0x03: Storage 0x04-0x0F: UCCONFIG MODES 0x10-0xFF = Reserved
29	bRateControlMode	1	Number	Bits 0-3 Modes: 0x00: Reserved 0x01: CBR 0x02: VBR 0x03: Constant QP Bits 4-7 Flags: 0x10: fixed_frame_rate_flag 0x20: Reserved set to zero 0x40: Reserved set to zero 0x80: Reserved set to zero
30	bTemporalScaleMode	1	Number	0x00: No Temporal enhancement layer 0x01- 0x07: Number of Temporal enhancement layers 0x08-0xFF = Reserved Note: Constrained by bUsageType.

31	bSpatialScaleMode	1	Number	0x00: No Spatial Enhancement Layer 0x01-0x08: Number of Spatial enhancement layers 0x09-0xFF = Reserved Note: Constrained by bUsageType.
32	bSNRScaleMode	1	Number	0x00: No SNR Enhancement Layer 0x01: Reserved 0x02: CGS_NonRewrite_TwoLayer 0x03: CGS_NonRewrite_ThreeLayer 0x04: CGS_Rewrite_TwoLayer 0x05: CGS_Rewrite_ThreeLayer 0x06: MGS_TwoLayer 0x07-0xFF = Reserved Note: Constrained by bUsageType.

33	bStreamMuxOption	1	Bitmap	Auxiliary stream control **Bit 0**: Enable/Disable auxiliary stream 0: auxiliary stream disabled. Bits 1-7 ignored. 1: auxiliary stream enabled. PROBE/COMMIT fields apply to streams indicated by bits 1-7. **Bit 1**: Embed H.264 auxiliary stream. **bStreamID** identifies the simulcast stream to be configured. **Bit 2**: Embed YUY2 auxiliary stream. **Bit 3**: Embed NV12 auxiliary stream. **Bit 4-5**: Reserved **Bit 6**: MJPEG payload used as a container. **Bit 7:** Reserved Note: For SET_CUR operation, only one auxiliary stream bit shall be set.
34	bStreamFormat	1	Number	0x00 – Output data in Byte stream format (H.264 Annex- B) 0x01 – Output data in NAL stream format 0x02-0xFF = Reserved
35	bEntropyCABAC	1	Number	0x00=CAVLC 0x01=CABAC 0x02-0xFF = Reserved

36	bTimestamp	1	Bool	0x00=picture timing SEI disabled 0x01=picture timing SEI enabled 0x02-0xFF = Reserved
37	bNumOfReorderFrames	1	Number	Number of B frames between the reference frames.
38	bPreviewFlipped	1	Bool	0x00 = No Change 0x01 = Horizontal Flipped Image for non H.264 streams. 0x02-0xFF = Reserved
39	bView	1	Number	Number of additional MVC Views. 0x00: none
40	bReserved1	1		Reserved- set to zero
41	bReserved2	1		Reserved-set to zero
42	bStreamID	1	Number	0x00-0x06 = Simulcast stream index 0x07-0xFF = Reserved
43	bSpatialLayerRatio	1	Number	Specifies the ratio between each spatial layer. The high nibble is defined for the integer part and low nibble is for the fractional part. It is represented in fixed point. Example: For 1.5 ratio → bSpatialLayerRatio = 0x18 For 2.0 ratio → bSpatialLayerRatio = 0x20
44	wLeakyBucketSize	2	Number	In milliseconds

Table 2: UVCX_VIDEO_CONFIG_PROBE/UVCX_VIDEO_CONFIG_COMMIT

The bmHints field indicates the host application's preference to "lock" some of the parameters in the PROBE/COMMIT structure. Those parameters with their corresponding bmHints bit clear are considered for adjustment, in the order from lowest priority (I FramePeriod) to highest priority (Resolution). If the device cannot generate a valid configuration after considering adjusting just these parameters, then it must set wWidth and wHeight to zero.

The GET_MAX command shall return the PROBE/COMMIT structure with maximum field performance independent of each other. i.e. No field is restricted by any other field.

For Multiplexed streams, the PROBE/COMMIT sequences shall be completed one stream at a time.

3.3.2 Dynamic Controls

Dynamic controls allow changing VS interface parameters while the VS interface is active.

The dynamic controls are: UVCX_RATE_CONTROL_MODE, UVCX_TEMPORAL_SCALE_MODE, UVCX_SPATIAL_SCALE_MODE, UVCX_SNR_SCALE_MODE, UVCX_LTR_BUFFER_SIZE_CONTROL, UVCX_LTR_PICTURE_CONTROL, UVCX_PICTURE_TYPE_CONTROL, UVCX_VERSION, UVCX_FRAMERATE_CONFIG, UVCX_VIDEO_ADVANCE_CONFIG, UVCX_BITRATE_LAYERS and UVCX_QP_STEPS_LAYERS.

3.3.2.1 wLayerID Structure

wLayerID				
Reserved (3 bits)	Stream ID (3 bits)	Quality ID (3 bits)	Dependency ID (4 bits)	Temporal ID (3 bits)
15 13	12 10	9 7	6 3	2 0

Table 3: wLayerID Structure

StreamID:

The StreamID provides specification of a specific H.264 stream in the case of a simulcast sequence. The StreamID has 3 bits (bits 12-10 in wLayerID) to support 7 streams (0-6). A value of 7 shall be used to simultaneously refer to all streams. In the case of a single H.264 stream, stream_id is always 0. Non-zero StreamID only appears in cases of simulcast of two or more H.264 streams.

QualityID:

The QualityID provides specification of a specific Quality layer in a multi-layer SVC stream. The QualityID has 3 bits (bits 9-7 in wLayerID) to support 7 Quality layers (0 enhancements – 6 enhancements layers). A value of 7 shall be used to simultaneously refer to all quality layers. In the case of a single-layer H.264 stream, QualityID shall always be 0. In the case of a SVC stream not using MGS mode SNR scalability, QualityID shall always be 0. A non-zero QualityID shall only appear in SVC streams using MGS mode SNR scalability where 1 indicates the first quality enhancement layer, up to the maximum quality Enhancement layer. The MSG mode of SNR scalability partitions transform coefficients into separate Quality layers.

DependencyID:

The DependencyID provides specification of a specific Dependency Layer in a multi-layer SVC stream. The DependencyID has 4 bits (bits 6-3 in wLayerID) to support 15 dependency layers (0 enhancements – 14 enhancements layers). A value of 15 shall be used to simultaneously refer to all Dependency layers. In the case of a single-layer H.264 stream, DependencyID shall always be 0. In the case of a SVC stream not using either CGS mode SNR scalability or Spatial scalability mode, DependencyID shall always be 0. A non-zero DependencyID shall only appear in SVC streams using either CGS mode SNR scalability or Spatial scalability where 1 indicates the first SNR or spatial enhancement layer, up to the maximum SNR or spatial Enhancement layer defined as the sum of bSpatialScaleMode and the number of CGS mode SNR scalable enhancement layers identified in table 8.

TemporalID:

The TemporalID provides specification of a specific Temporal Layer in a multi-layer SVC stream. The TemporalID has 3 bits (bits 2-0 in wLayerID) to support 7 temporal layers (0 enhancements – 6 enhancements layers). A value of 7 shall be used to simultaneously refer to all temporal layers. In the case of a single-layer H.264 stream, TemporalID shall always be 0. In the case of a SVC stream not using temporal scalability, TemporalID shall always be 0. A non-zero TemporalID shall only appear in SVC streams using temporal scalability where 1 indicates the first temporal enhancement layer, up to the maximum temporal Enhancement layer bTemporalScaleMode set in the UVCX_TEMPORAL_SCALE_MODE control.

Reserved:

The Reserved field has 3 bits (bits 15-13 in wLayerID) and shall always be 0.

3.3.3 UVCX_RATE_CONTROL_MODE

This control allows the application to dynamically switch between rate control modes.

Control Selector		UVCX_RATE_CONTROL_MODE		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		3		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast The wLayerID structure is defined in section 3.3.2.1.
2	bRateControlMode	1	Number	Bits 0-3 Modes: 0x00: Reserved 0x01: CBR 0x02: VBR 0x03: Constant QP Bits 4-7 Flags: 0x10: fixed_frame_rate_flag 0x20: Reserved set to zero 0x40: Reserved set to zero 0x80: Reserved set to zero

Table 4: Rate Control mode

3.3.4 UVCX_TEMPORAL_SCALE_MODE

The UVCX_TEMPORAL_SCALE_MODE control dynamically queries and configures the number of temporal layers.

Control Selector		UVCX_TEMPORAL_SCALE_MODE		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		3		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast The wLayerID structure is defined in section 3.3.2.1.
2	bTemporalScaleMode	1	Number	0x00: No Temporal Enhancement Layer 0x01- 0x07: Number of Temporal Enhancement Layers 0x08-0xFF = Reserved

Table 5: Temporal scale mode control

The dwFrameInterval parameter, defined in UVCX_VIDEO_CONFIG_COMMIT (Table 2), establishes the upper boundary on the frame rate of the highest layer.

3.3.5 UVCX_SPATIAL_SCALE_MODE

The UVCX_ SPATIAL _SCALE_MODE control is used to dynamically query and configure the number of spatial layers.

Control Selector		UVCX_SPATIAL_SCALE_MODE		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		3		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast The wLayerID structure is defined in section 3.3.2.1.
2	bSpatialScaleMode	1	Number	0x00: No Spatial Enhancement Layer 0x01-0x08: Number of Spatial Enhancement Layers 0x09-0xFF = Reserved

Table 6: Spatial scale mode control

The bSpatialScaleMode parameter configures the number of spatial layers in the stream. The wWidth and wHeight parameters, defined in UVCX_VIDEO_CONFIG_COMMIT (Table 2), establishes the upper boundary on resolution. Similarly, the bSpatialLayerRatio defines the resolution ratio for lower spatial layers.

3.3.6 UVCX_SNR_SCALE_MODE

The UVCX_ SNR_SCALE_MODE control is used to dynamically query and configure the number of SNR layers.

Control Selector		UVCX_SNR_SCALE_MODE		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		4		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast The wLayerID structure is defined in section 3.3.2.1.
2	bSNRScaleMode	1	Number	0x00: No SNR Enhancement Layer 0x01: Reserved 0x02: CGS_NonRewrite_TwoLayer 0x03: CGS_NonRewrite_ThreeLayer 0x04: CGS_Rewrite_TwoLayer 0x05: CGS_Rewrite_ThreeLayer 0x06: MGS_TwoLayer 0x07-0xFF = Reserved
3	bMGSSublayerMode	1	Number	MGS Sublayer Partition index 0x00: Reserved for non-MGS case 1-15: Number of transform coefficient units allocated to quality layer 1. 16-0xff: Reserved Note: if bSNRMode does not equal 6, then this field must be set to zero. Note: The second quality layer will contain all of the remaining transform coefficients

Table 7: SNR scale mode control

bSNRScaleMode	Description	Number of SNR Scalable Enhancement Layers	Number of Quality Layers	CGS Mode	Rewrite Mode
0x00	None	0	0	0	0
0x01	Reserved	NA	NA	NA	NA
0x02	CGS_NonRewrite_TwoLayer	1	0	1	0
0x03	CGS_NonRewrite_ThreeLayer	2	0	1	0
0X04	CGS_Rewrite_TwoLayer	1	0	1	1
0X05	CGS_Rewrite_ThreeLayer	2	0	1	1
0x06	MGS_TwoLayer	0	2	0	0
0x07-0xFF	Reserved	0	0	0	0

Table 8: bSNRScaleMode

3.3.7 UVCX_LTR_BUFFER_SIZE_CONTROL

The UVCX_LTR_BUFFER_SIZE_CONTROL should provide the control to device's Long term reference buffer usage. The host should check the device's long term buffer availability for the control.

Control Selector		UVCX_LTR_BUFFER_SIZE_CONTROL		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		4		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast The only base layer is valid for SVC. The wLayerID structure is defined in section 3.3.2.1
2	bLTRBufferSize	1	Number	Total Number of Long Term Reference Frames for current setup 0x00 – none 0x01 – one 0x02 – two Up to 0xFF
3	bLTREncoderControl	1	Number	Number of Long Term Reference Frames the device can control. 0 – none. Device will not control any LTRs. 1 – Device will control one LTR. Etc.

Table 9: Long term buffer Size control

The UVCX_LTR_BUFFER_SIZE_CONTROL controls the allocation of long term reference (LTR) frames of the device. Additionally, the control provides for a subset of the total buffer to be allocated for device control, and the remainder shall be allocated for host control using UVCX_LTR_PICTURE_CONTROL. If the device does not have enough memory to allow use of long term reference at the current resolution, then the GET_MAX shall return bLTRBufferSize equal to 0. Once the number of controllable buffers is known the host then sets the actual number which device should reserve for device control via bLTREncoderControl. The bLTREncoderControl shall be less or equal to bLTRBufferSize read from the device. The number of LTR buffers allocated for Host control is implicitly set to bLTRBufferSize – bLTREncoderControl. If the device does not allow the host to manage any LTB buffers, then the device shall set bLTRBufferSize equal to 0.

If the device allows the host to manage the LTR buffers, it shall assign continuous index space starting from 0 for the host controlled LTR frames.

The device is responsible for signaling appropriate Decoder picture buffer parameters in SPS. It shall make sure that buffer size stays within the limits given the assigned level. The device may generate IDR if necessary.

Note: The device expected behavior is explained in FAQ (USB_Video_Payload_H.264_FAQ)

3.3.8 UVCX_LTR_PICTURE_CONTROL

The UVCX_LTR_PICTURE_CONTROL should provide host to limit and/or change which long term reference frame will be used for next frame encoding.

Control Selector		UVCX_LTR_PICTURE_CONTROL		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		4		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast. The wLayerID structure is defined in section 3.3.2.1.
2	bPutAtPositionInLTRBuffer	1	Number	Next frame should be put at certain position in Long Term Reference Buffer (LTRB) 0 - Encoder is free to choose where to save the frame except that it cannot be saved at host controlled part of LTRB (positions 0..N-1) 1 - position 0 2 - position 1 … N – position N-1 (maximum) Note: N = bLTRBufferSize – bLTREncoderControl: Number of LTR Buffers under Host control (valid indexes are 0 through N-1).
3	bEncodeUsingLTR	1	Bitmap	Next frame should only be referring a certain set of frames from LTR 0x00 – Request an I frame 0x01 - LTR frame from position 0 0x02 - LTR frame from position 1 0x04 - LTR frame from position 2 0x08 - LTR frame from position 3

				Etc. possible combined in a bitmap 0xFF - Encoder may use any of valid frames in DPB. The previous calls with bEncodeUsingLTR not equal 0xFF may invalidate some or all frames in DPB.

Table 10: Picture Long term reference control

Note: The device expected behavior is explained in FAQ (USB_Video_Payload_H.264_FAQ).

3.3.8.1 bPutAtPositionInLTRBuffer

The max number in bPutAtPositionInLTRBuffer is equal to bLTRBufferSize – bLTREncoderControl (from UVCX_LTR_BUFFER_SIZE_CONTROL). i.e. frames 0 to bLTRBufferSize – bLTREncoderControl -1 are assigned to being controlled by the host.

bPutAtPositionInLTRBuffer = 0 means that encoder has freedom to where to save the frame (save in short term buffer, its own section of LTRB i.e. with index N through bLTRBufferSize-1).

3.3.8.2 bEncodeUsingLTR

The parameter bEncodeUsingLTR specifies that the only specific subset host controlled long term reference frames of all possible frames in decoded picture buffer can be used for encoding a next frame. If bEncodeUsingLTR>0 no short term frames should be used by encoder for encoding the current frame.

a. The encoder is not required to utilize all (or any) the frames in the LTR buffer unless explicitly asked to (using bEncodeUsingLTR bitmap). The encoder processing power limitation could force encoder to use only one frame as a reference.
b. Free Choice Mode: mode of initial operation of the encoder between the first IDR frame (which goes into location 0) and when the first UVCX_LTR_PICTURE_CONTROL with bEncodeUsingLTR>0 is received by encoder. Encoder may use one, some or all frames from the decoded picture buffer in Free Choice Mode.
c. Limited Choice Mode: mode of operation of the encoder after reception of a UVCX_LTR_PICTURE_CONTROL with bEncodeUsingLTR>0. Note, once Encoder has entered a Limited Choice Mode it expected to remain in such mode until a new IDR frame is generated.
d. Once a command with bEncodeUsingLTR > 0 is executed at frame N. Encoder shall not have a free choice of frames to use as references (Limited Choice Mode). For encoding frames N+1 and future the following rules apply
    I. It shall NOT use frames from short term reference buffer older than N (N, N+1 etc are usable. N-1, N-2 etc are not usable)
    II. It shall NOT use any frames from LTR buffer other than the set described by most recent bEncodeUsingLTR and it applies to the encoder controlled portion of LTR buffer as well.

III. LTR frames updated after frame N was encoded can be used as reference (similar to #I case)

IV. Encoder is free to update own portion of LTR buffer with newer frames and use those in future encoding.

e. It is expected in case UVCX_ LTR_PICTURE_ CONTROL with bEncodeUsingLTR>0 is executed then in order to improve coding efficiency and network control logic:

I. Reference Picture Re-Ordering command is inserted to slice header by the encoder with frames actively used for encoding moved at beginning of the list. The semantics of a command is described in “7.4.3.1 Reference picture list modification semantics“ in H.264 standard.

II. The actual number of active reference frames signaled via num_ref_idx_l0_active_minus1 as described in “7.4.3 Slice header semantics” in H.264 standard.

3.3.9 UVCX_PICTURE_TYPE_CONTROL

The UVCX_PICTURE_TYPE_CONTROL is used for requesting the next frame as a requested Picture and thereafter the stream goes back to normal frames.

Control Selector		UVCX_PICTURE_TYPE_CONTROL		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		4		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast The wLayerID structure is defined in section 3.3.2.1
2	wPicType	2	Number	0x0000: I-Frame 0x0001: Generate an IDR frame 0x0002: Generate an IDR frame with new SPS and PPS 0x0003-0xFFFF=Reserved

Table 11: Picture type control

3.3.10 UVCX_VERSION

The UVCX_VERSION control is used to dynamically query and negotiate the device version.

Control Selector		UVCX_VERSION		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		2		
Offset	Field	Size	Value	Description
0	wVersion	2	Number	Version 1.00 0x0100 for this version BCD format Examples: 1.10 =0x0110 10.01=0x1001

Table 12: Version control

3.3.11 Encoder Configuration Reset

3.3.11.1 UVCX_ENCODER_RESET

The UVCX_ENCODER_RESET should provide the option of initialization of each or all streams. The command shall set all the dynamic and static control parameters to default state.

Control Selector		UVCX_ENCODER_RESET		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		2		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast. The wLayerID structure is defined in section 3.3.2.1.

Table 13: Encoder Configuration Reset

3.3.12 UVCX_FRAMERATE_CONFIG

The UVCX_FRAMERATE_CONFIG control is used to dynamically query and configure the frame interval.

Control Selector		UVCX_FRAMERATE_CONFIG		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		6		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Bit mask for StreamID ,QualityID, DependencyID, and TemporalID, The wLayerID structure is defined in section 3.3.2.1.
2	dwFrameInterval	4	Number	In 100 ns frame interval

Table 14: Dynamic frame rate configuration

3.3.13 UVCX_VIDEO_ADVANCE_CONFIG

The UVCX_VIDEO_ADVANCE_CONFIG control is used to dynamically query the dwMb_max of the device. It is also used to dynamically query and configure the blevel_idc.

Control Selector		UVCX_VIDEO_ADVANCE_CONFIG		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		8		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Only StreamID is used for Simulcast The wLayerID structure is defined in section 3.3.2.1
2	dwMb_max	4	Number	The number of macroblocks per second processing rate. The parameter is provided by the device for its maximum processing rate.
6	blevel_idc	1	Number	As specified level_idc in H.264 specification. For example, 0x1F = level 3.1 0x28 = level 4.0
7	bReserved	1	Number	Reserved

Table 15: Advance configuration

3.3.13.1 dwMb_max

The dwMb_max should provide the device's maximum macroblock per second processing power.

3.3.13.2 blevel_idc

The blevel_idc parameter provides option to ensure the usage of the decoder capabilities.

3.3.14 UVCX_BITRATE_LAYERS

The UVCX_BITRATE_LAYERS control is used to dynamically query and configure the bitrates of the individual layer.

Control Selector		UVCX_BITRATE_LAYERS		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		10		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Bit mask for StreamID ,QualityID, DependencyID, and TemporalID, The wLayerID structure is defined in section 3.3.2.1.
2	dwPeakBitrate	4	Number	Peak Bitrate in bits/sec for the specified wLayerID. To set the wLayerID for subsequent get operations, set this field to zero in a SET_CUR command.
6	dwAverageBitrate	4	Number	Average Bitrate in bits/sec for the specified wLayerID. To set the wLayerID for subsequent get operations, set this field to zero in a SET_CUR command.

Table 16: Bitrate control

3.3.15 UVCX_QP_STEPS_LAYERS

The UVCX_QP_STEPS_LAYERS control is used to dynamically query and configure the Minimum/Maximum QP of the individual layer.

Control Selector		UVCX_QP_STEPS_LAYERS		
Mandatory Requests		SET_CUR, GET_CUR, GET_DEF, GET_INFO, GET_LEN, GET_MAX, GET_MIN		
wLength		5		
Offset	Field	Size	Value	Description
0	wLayerID	2	Bitmap	Bit mask for StreamID ,QualityID, DependencyID, and TemporalID, The wLayerID structure is defined in section 3.3.2.1.
2	bFrameType	1	Bitmap	Bitmap of frame types 0x00 = Reserved 0x01 = I frame 0x02 = P frame 0x04 = B frame 0x07 = all types 0x08 = Reserved 0xF0 = Reserved
3	bMinQp	1	Signed	Minimum Quantization step size To set the wLayerID and bFrameType for subsequent get operations, set this field to zero in a SET_CUR command
4	bMaxQp	1	Signed	Maximum Quantization step size To set the wLayerID and bFrameType for subsequent get operations, set this field to zero in a SET_CUR command.

Table 17: Quantization control

3.4 Packetization

- H.264 elementary stream format, extended to support multiplexed payload.

3.5 Stream Multiplexing

If the device supports data multiplexing (as defined in section 3.1.2.2), primary UVC probe/commit format shall be MJPG and the auxiliary format shall be delivered to the host by injecting the additional stream into the application-specific data segments of the JPEG payload as described below.

3.5.1 Payload Header

As the device supports more than one stream that can be injected into the payload, the header information as described below shall be added at the beginning of each stream.

Header Format

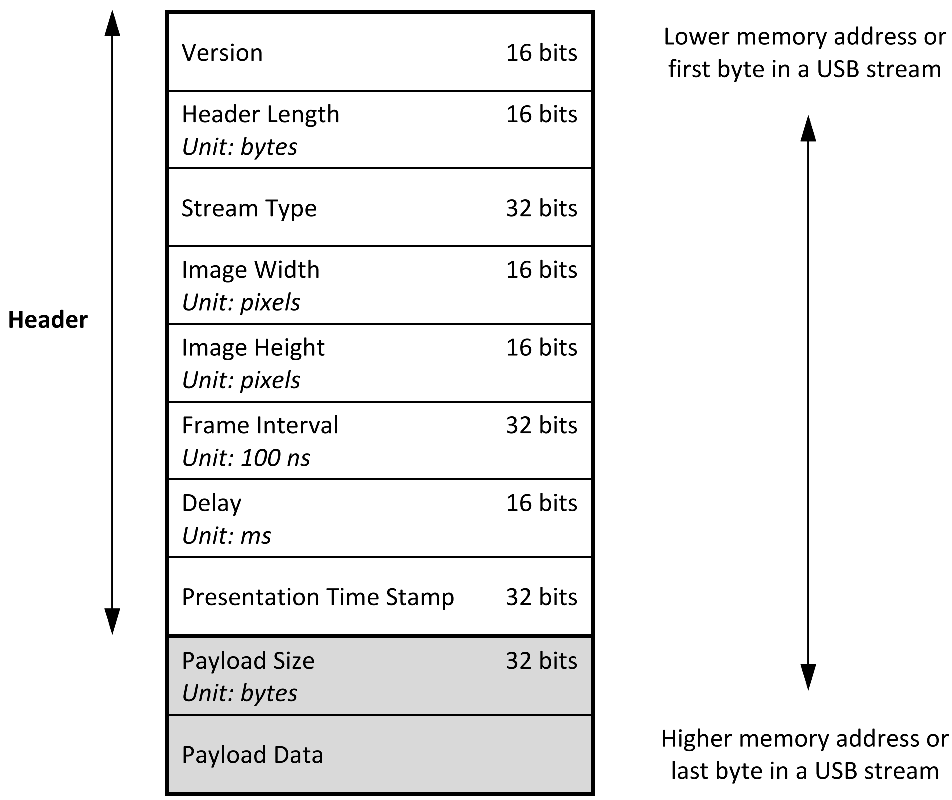


Figure 4 Header Format

Notes:

a. All fields containing integer values are in little endian byte order. This is in contrast to any JPEG specific fields (e.g. the 16-bit length field that follows the JPEG APP marker is in big endian byte order).
b. Header Length: The Header Length provides the bytes offset to Payload Size field from start of the Payload header. The *Payload Size* field and the *Payload Data* are not considered part of the header and do therefore *not* count towards the *Header Length* field. For example, the header length is 22 bytes in the example (Figure 4)
c. The *Stream Type* field contains a 4-byte FourCC code denoting the format contained in the payload.
d. The frame rate of the auxiliary stream is described by means of the *Frame Interval* field in units of 100 nanoseconds. For example, 25 fps would be 400,000 (0x00061A80).
e. The *Delay* field describes the dynamic encoding delay introduced by the device, measured from end of exposure to data send on USB. The field may be different and dynamic for each stream.
f. The Presentation Time Stamp field provides the frame capture time.
g. The *Payload Size* field contains the total size of the payload data that is contained in the current JPEG frame (the payload data in the current APP segment and remaining application segments), hence its 32-bit size. The value does *not* include the 4 bytes that the *Payload Size* field occupies. (Example as Figure 5 Payload Size includes the payload data of first application segment and remaining full application segments size. Payload Size also includes marker and length of remaining application segments.)

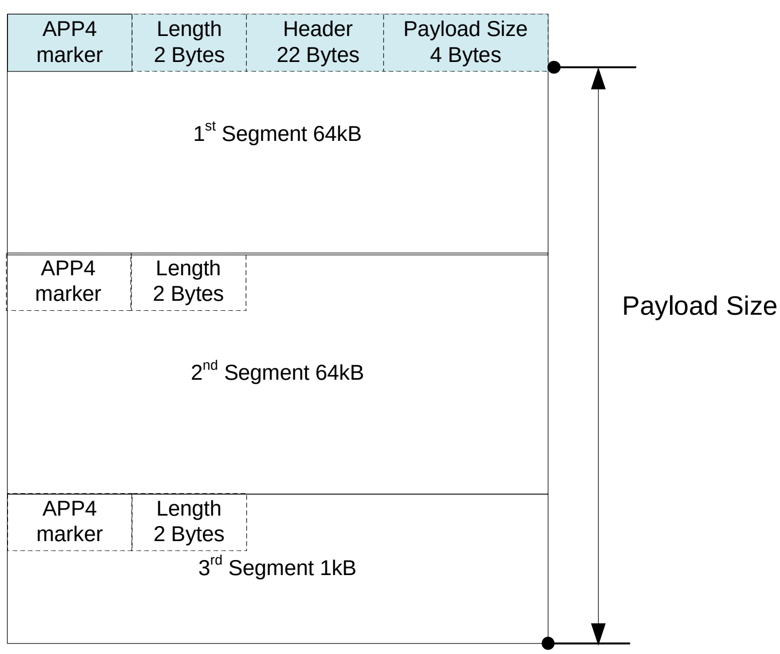


Figure 5 Payload Size

The *Version* field contains one of the values from the following table:

Version	Version field contents	Header length
1.0 (described in this specification)	0x0100	22 bytes (for the current example)

3.5.2 Multiplexed Payload

a. Assume 'x' bytes of H.264 encoded data to be inserted (always including the header).

b. Assume 'y' bytes of YUY2 data to be inserted (always including the header).

c. Create the data in memory as mentioned below:

Payload Header	Payload Data	Payload Header	Payload data

d. Break them into segments, each not more than 64K in length.

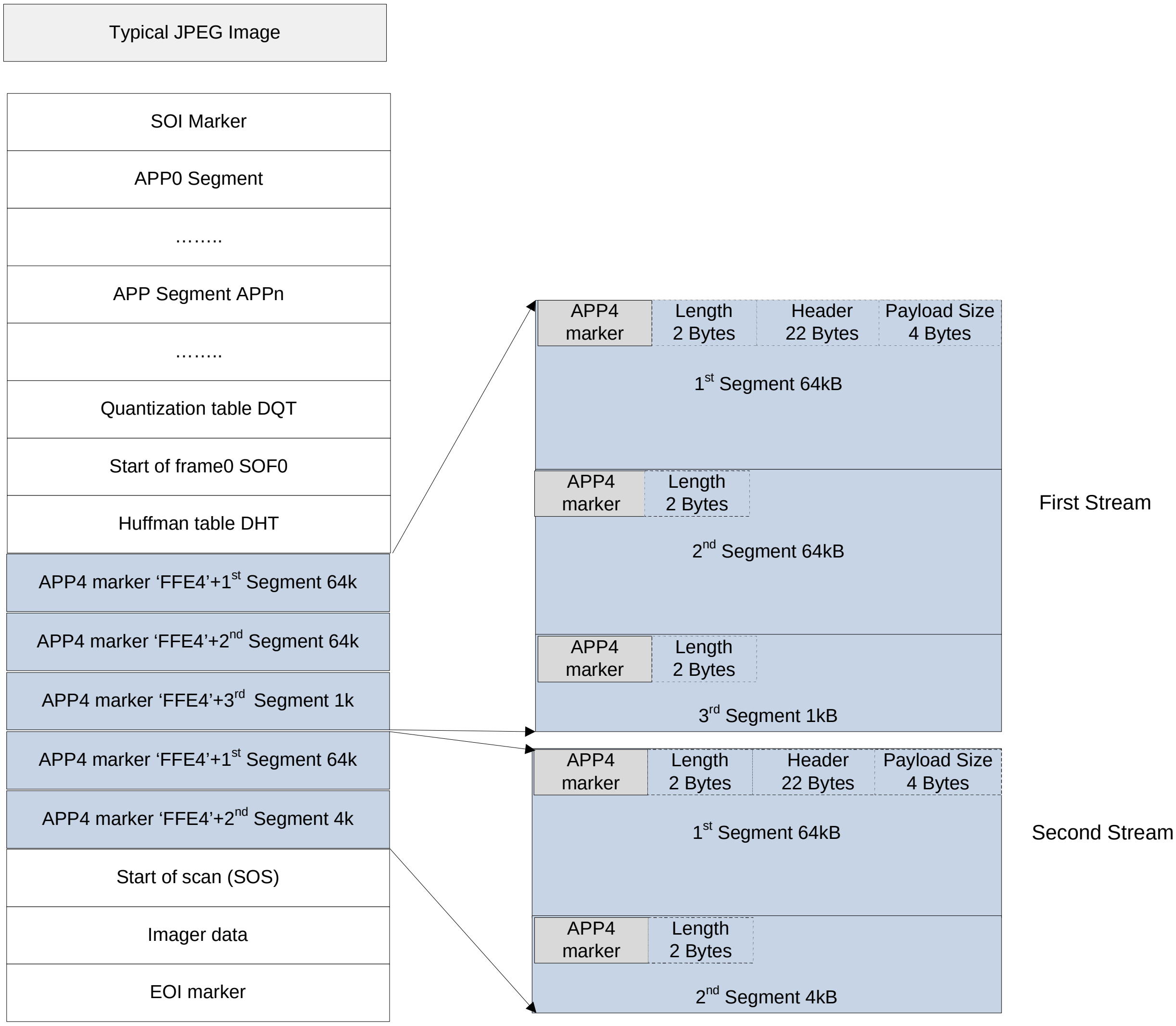
e. Scan for the marker 'FFDA' (SOS) in the original JPEG image/frame, between SOI and EOI. Remember to skip any image/actual data byte 0xFF, if followed by 'zero'. They are not markers. Also some markers such as 'restart/resync' don't have 'length' fields. Refer to JPEG specifications for further information.

f. Insert each segment before SOS segment, one by one, with an application marker prefix 'FFE4'. The app marker shall be followed by the application data segment length field of 2 bytes, as required for JPEG compliant. For example, if 'size of x' = payload+header=129K, we have 3 segments of 64K, 64K and 1K. For example, if 'size of y' =142K, 64+64+14. So, total 6 segments will be created

The following block diagram describes original MJPEG data and MJPEG data with H.264 stream- injected into it for the above scenario. Shown only for 'x'

Typical JPEG Image
SOI Marker
APP0 Segment
……..
APP Segment APPn
……..
Quantization table DQT
Start of frame0 SOF0
Huffman table DHT
Start of scan (SOS)
Imager data
EOI marker

Figure 6 Typical JPEG Image



Note:
1. **APP4 Marker (2 bytes) and Length (2 bytes) are in network byte order as per JPEG requirements.**
2. The header is included in only first segment of the stream.
3. New stream starts with next application segment.
4. The header and Payload size fields are defined in Figure 4

Figure 7 Example Payload+header

For 'y' similar to Figure 6 and 7 should be considered.

Assumptions:

- The frame rate of the primary UVC stream should typically be greater than the auxiliary stream.

- Not all MJPG payloads shall contain auxiliary video data. Clients should not assume the availability of auxiliary stream in MJPG payload as it is completely dependent on the encoder rate control or other system dependencies and availability of data.

3.6 Buffering Period and Picture Timing SEI messages

Buffering period (BP) and picture timing (PT) supplemental enhancement information (SEI) NALUs can be used to carry additional timing information in the elementary bitstream. When present, a NALU containing a BP or PT SEI message must contain only one SEI message. When present, a NALU containing a BP SEI message must be the first SEI NALU of the picture. When present, a NALU containing a PT SEI message must be the first SEI NALU of the picture other than (when present) a NALU containing a BP SEI message. When present, decoders should use this timing information to understand relative frame capture times when the video comes from a variable frame rate source. When such timing information is present, random-access I frames (as well as IDR frames) shall have an associated BP SEI message.

If the client enables picture timing SEI messages by setting bTimestamp to 1 in Table 2, BP and PT SEI messages must be present in the bitstream.

4 Appendix-A

4.1 GUIDs:

4.1.1 Extension Unit GUIDs

Extension Unit	GUID
Codec (H.264) Control	{A29E7641-DE04-47e3-8B2B-F4341AFF003B}

4.1.2 H.264 Streams GUIDs

Extension Unit	GUID
MEDIASUBTYPE_H264	{34363248-0000-0010-0x8000-00aa00389b71}

5 Appendix-B

Usage Examples

The examples are provided to configure the UVC H.264 device. The application shall use the UVC XU control to configure the device. The configuration process involves getting the device capabilities. The process also addresses the application requirement and device capabilities negotiation.

Configuration Data Structure: As per Table 2

```
struct {
Word32      dwFrameInterval;
Word32      dwBitRate;
Word16      bmHints;
Word16      wConfigurationIndex;
Word16      wWidth;
Word16      wHeight;
Word16      wSliceUnits;
Word16      wSliceMode;
Word16      wProfile;
Word16      wIFramePeriod;
Word16      wEstimatedVideoDelay;
Word16      wEstimatedMaxConfigDelay;
UChar       bUsageType;
UChar       bRateControlMode;
UChar       bTemporalScaleMode;
UChar       bSpatialScaleMode;
UChar       bSNRScaleMode;
UChar       bStreamMuxOption;
UChar       bStreamFormat;
UChar       bEntropyCABAC;
UChar       bTimestamp;
UChar       bNumOfReorderFrames;
UChar       bPreviewFlipped;
UChar       bView;
UChar       bReserved1;
UChar       bReserved2;
UChar       bStreamID;
UChar       bSpatialLayerRatio;

Word16      wLeakyBucketSize;
} struct_ UVCX_VIDEO_CONFIG;
```

Note: The mixing of decimal and hexadecimal are done to make it more readable.

5.1 Programming Example for Single Payload based configuration

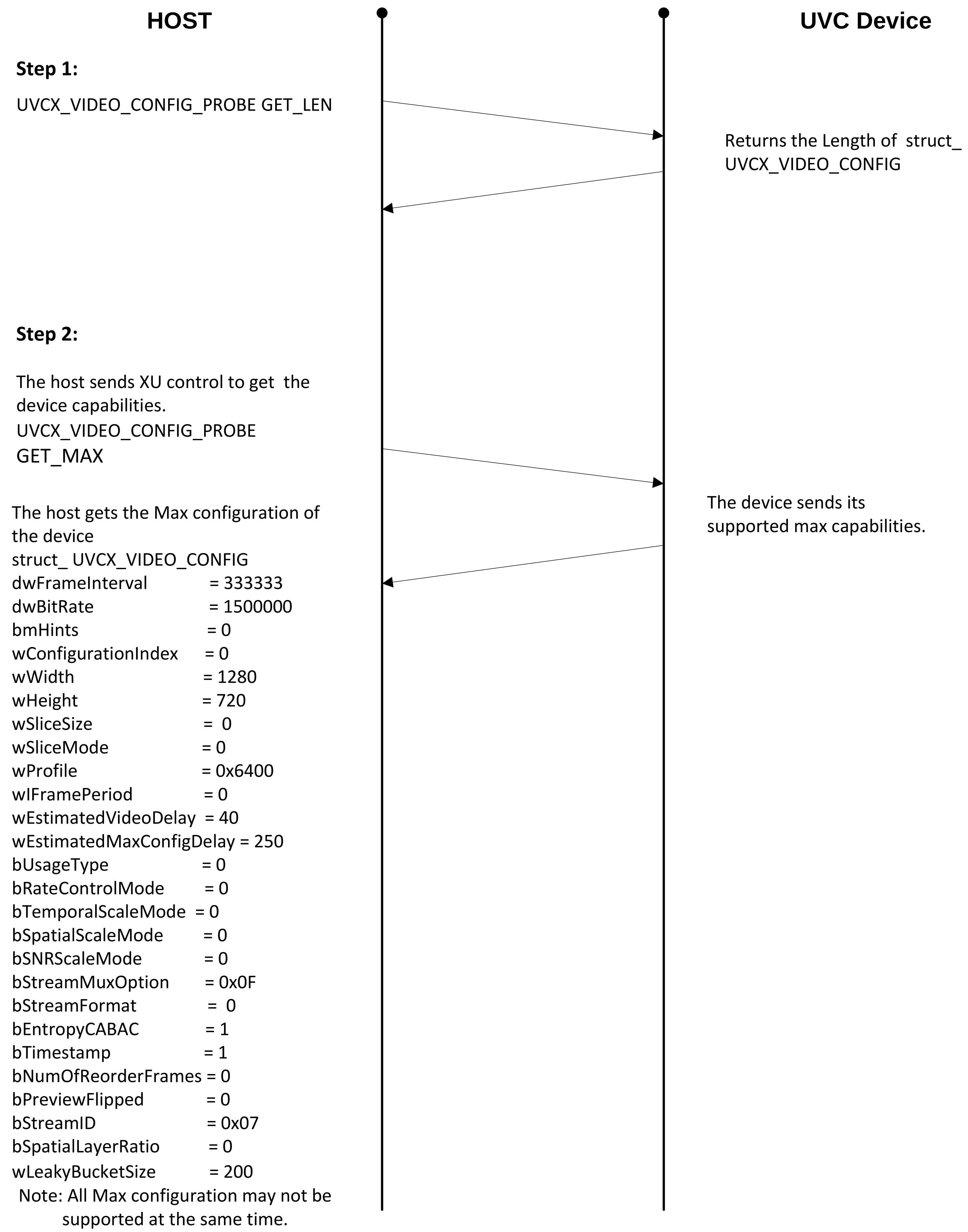
Device Capabilities:

- Single Payload
- H.264 Baseline Profile, Constrained Baseline, High Profile.
- 1280x720
- 15 and 30 Frames per second
- Single slice support
- CAVLC support only

Host Requested Configuration:

- H.264 Payload Format
- H.264 Baseline Profile
- 1280x720
- 30 Frames per second
- Real-time use case
- CBR mode
- 512K bits per second

Note: The program will have to start from the Step 1 (defined in the example) in the event of command error.


HOST
UVC Device
Step 1:
UVCX_VIDEO_CONFIG_PROBE GET_LEN
Returns the Length of struct_ UVCX_VIDEO_CONFIG
Step 2:
The host sends XU control to get the device capabilities.
UVCX_VIDEO_CONFIG_PROBE
GET_MAX
The device sends its supported max capabilities.
The host gets the Max configuration of the device
struct_ UVCX_VIDEO_CONFIG
dwFrameInterval = 333333
dwBitRate = 1500000
bmHints = 0
wConfigurationIndex = 0
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x6400
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x0F
bStreamFormat = 0
bEntropyCABAC = 1
bTimestamp = 1
bNumOfReorderFrames = 0
bPreviewFlipped = 0
bStreamID = 0x07
bSpatialLayerRatio = 0
wLeakyBucketSize = 200
Note: All Max configuration may not be supported at the same time.

HOST	UVC Device

Step 3:
This point the host is aware of the maximum configuration supported by the device.
The host configures the required parameters in configuration structure.

struct_ UVCX_VIDEO_CONFIG
dwFrameInterval = 333333
dwBitRate = 512000
wWidth = 1280
wHeight = 720
bUsageType = 1 (Real Time)
bRateControlMode = 1
bStreamMuxOption = 0x03
wProfile = 0x4200
bStreamID =0x00
wLeakyBucketSize =200

The host sends XU control UVCX_VIDEO_CONFIG_PROBE SET_CUR

The device evaluates the SET_CUR parameters based on its capabilities. The device updates the structure for the supported configuration.

struct_ UVCX_VIDEO_CONFIG
dwFrameInterval = 333333
dwBitRate = 512000
bmHints = 0
wConfigurationIndex = 1
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x4200
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x03
bStreamFormat = 0
bEntropyCABAC = 1
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x00
bSpatialLayerRatio = 0
wLeakyBucketSize = 200

HOST | **UVC Device**

Step 4:

The host shall request current parameters of the device.
UVCX_VIDEO_CONFIG_PROBE
GET_CUR

The device sends back the present configuration, which is done in **step 3**.

struct_ UVCX_VIDEO_CONFIG
dwFrameInterval = 333333
dwBitRate = 512000
bmHints = 3
wConfigurationIndex = 1
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x4200
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x03
bStreamFormat = 0
bEntropyCABAC = 1
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x00
bSpatialLayerRatio = 0
wLeakyBucketSize = 200

The host validates the configuration. The host has an option of changing the parameters again by following Step 2:

Step 5:

The host will send xu control to start streaming based on

UVCX_VIDEO_CONFIG_COMMIT

SET_CUR

The device configures the encoder and starts the stream.

Step 6:

The host shall proceed with UVC PROBE/ COMMIT

5.2 Programming Example for Multiplexed Payload

Device Capabilities:

- Support Multiplexed Payload Format
- H.264 Baseline Profile, Constrained Baseline, High profile. MJPEG, YUY2 and NV12
- 1280x720, 640x480
- 15, 24 and 30 Frames per second
- Single slice support
- CABAC and CAVLC support

Host Requested Configuration:

- Multiplexed Payload
- H.264 High Profile 1280x720
- NV12 640x480
- 30 Frames per second
- Real-time use case
- CBR mode
- 1000K bits per second

The device needs to be configured twice, once for the H.264 stream and once for the NV12 stream. Each stream must be configured using the associated mux option as defined in section 3.1.1.2 "Multiplexed Payload Format".

Note: The following parameters are not applicable for YUY2 and NV12.

```
Word16      wRateControlMode;
Word16      wSliceSize;
Word16      wSliceMode;
Word16      wBitRate;
Word16      wProfile;
UChar       bTemporalScaleMode
UChar       bSpatialScaleMode;
UChar       bSNRScaleMode;
UChar       bStreamFormat;
UChar       bEntropyCABAC;
UChar       bSpatialLayerRatio;

Word16      wLeakyBucketSize;
```

The program will have to start from the Step 1 (defined in the example) in the event of command error.

HOST | **UVC Device**

Step 1:

UVCX_VIDEO_CONFIG_PROBE GET_LEN

Returns the Length of struct_ UVCX_VIDEO_CONFIG

Step 2:

The host sends XU control to get the device capabilities.
UVCX_VIDEO_CONFIG_PROBE
GET_MAX

The device sends its supported max capabilities.

The host gets the Max configuration of the device.
struct_ UVCX_VIDEO_CONFIG

```
dwFrameInterval          = 333333
dwBitRate                = 1500000
bmHints                  = 0
wConfigurationIndex      = 0
wWidth                   = 1280
wHeight                  = 720
wSliceSize               = 0
wSliceMode               = 0
wProfile                 = 0x6400
wIFramePeriod            = 0
wEstimatedVideoDelay     = 40
wEstimatedMaxConfigDelay = 250
bUsageType               = 0
bRateControlMode         = 0
bTemporalScaleMode       = 0
bSpatialScaleMode        = 0
bSNRScaleMode            = 0
bStreamMuxOption         = 0x0F
bStreamFormat            = 0
bEntropyCABAC            = 1
bTimestamp               = 1
bNumOfReorderFrame       = 0
bPreviewFlipped          = 0
bStreamID                = 0x06
bSpatialLayerRatio       = 0
wLeakyBucketSize         = 200
```

Note: All Max configuration may not be supported at the same time.

HOST | **UVC Device**

Step 3:
This point the host is aware of the maximum configuration supported by the device
The host configures the required parameters in configuration structure for primary payload (H.264).
struct_ UVCX_VIDEO_CONFIG

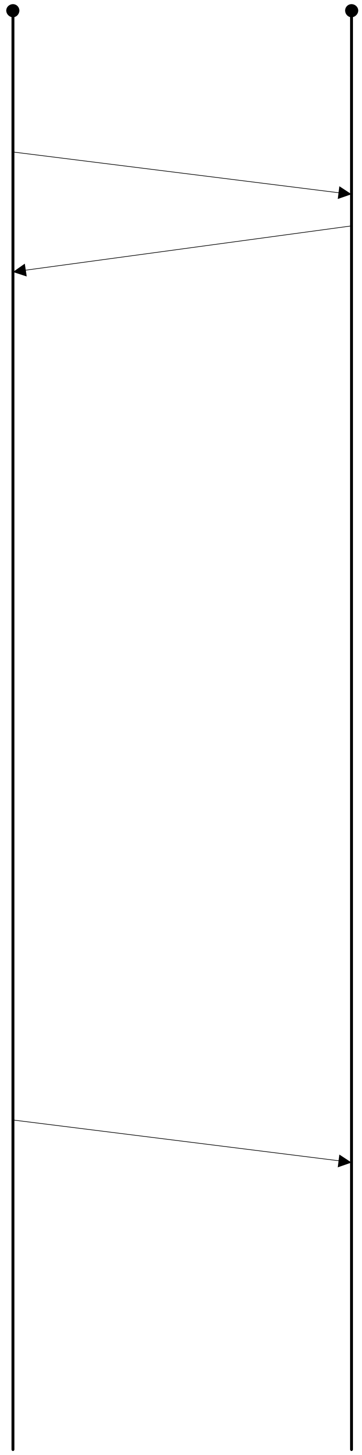
dwFrameInterval	= 666667
wBitRate	= 1000000
wWidth	= 1280
wHeight	= 720
wProfile	= 0x6400
bUsageType	= 1 (Real Time)
bRateControlMode	= 1
bStreamMuxOption	= 0x03 (H.264)
bStreamID	= 0x00

The host sends XU control UVCX_VIDEO_CONFIG_PROBE SET_CUR

The device evaluates the SET_CUR parameters based on its capabilities. The device updates the structure for the capable configuration.

dwFrameInterval	= 666667
dwBitRate	= 1000000
bmHints	= 0
wConfigurationIndex	= 1
wWidth	= 1280
wHeight	= 720
wSliceSize	= 0
wSliceMode	= 0
wProfile	= 0x6400
wIFramePeriod	= 0
wEstimatedVideoDelay	= 40
wEstimatedMaxConfigDelay	= 250
bUsageType	= 1
bRateControlMode	= 0
bTemporalScaleMode	= 0
bSpatialScaleMode	= 0
bSNRScaleMode	= 0
bStreamMuxOption	= 0x03
bStreamFormat	= 0
bEntropyCABAC	= 1
bTimestamp	= 1
bNumOfReorderFrame	= 0
bPreviewFlipped	= 0
bStreamID	= 0x00
bSpatialLayerRatio	= 0
wLeakyBucketSize	= 200

HOST | **UVC Device**



Step 4:

The host shall request current set parameters of the device.
UVCX_VIDEO_CONFIG_PROBE
GET_CUR

The device sends back the present configuration, which is done in **step 3**.

The host validates the configuration and takes care in application. The host has an option of changing the parameters again by following Step 2:

dwFrameInterval = 666667
dwBitRate = 1000000
bmHints = 0
wConfigurationIndex = 1
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x6400
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 1
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x03
bStreamFormat = 0
bEntropyCABAC = 1
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x00
bSpatialLayerRatio = 0
wLeakyBucketSize = 200

Step 5:

The host will send xu control to start streaming based on

UVCX_VIDEO_CONFIG_COMMIT

SET_CUR

The device configures the encoder and starts the stream.

HOST | **UVC Device**

Step 6:
The host configures the required parameters in configuration structure for the second payload (NV12).
struct_ UVCX_VIDEO_CONFIG
dwFrameInterval = 666667
wWidth = 640
wHeight = 480
bStreamMuxOption =0x09 (NV12)

The host sends XU control
UVCX_VIDEO_CONFIG_PROBE SET_CUR

The device evaluates the SET_CUR parameters based on its capabilities. The device updates the structure for the capable configuration.
dwFrameIntreval = 666667
bmHints = 0
wConfigurationIndex = 2
wWidth = 640
wHeight = 480
wSliceSize = 0
wSliceMode = 0
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x09
bStreamFormat = 0
wProfile = 0

Step 7:
The host shall request current parameter set of the device.
UVCX_VIDEO_CONFIG_PROBE
GET_CUR

The device sends back the present configuration, which is done in **step 6**.

The host validates the configuration and takes care in application.

Step 8:

The host will send xu control.

UVCX_VIDEO_CONFIG_COMMIT

SET_CUR

The device configures the encoder.

Step 9:

The host shall proceed with UVC PROBE/
COMMIT

5.3 Programming Example for Configuration Negotiation

Device Capabilities:

- Single Payload
- H.264 Baseline Profile, Constrained Baseline Profile, High Profile
- 1280x720
- 30 Frames per second
- Single slice support
- CAVLC support only

Host Requested Configuration:

- Single Payload
- H.264 High Profile
- 1280x720
- 30 Frames per second
- Real-time use case
- CBR mode
- 512K bits per second
- CABAC

Host Negotiated Configuration:

- Single Payload
- H.264 Baseline profile.
- 1280x720
- 30 Frames per second
- Real-time use case
- CBR mode
- 512K bits per second
- CAVLC

Note: The program will have to start from the Step 1 (defined in the example) in the event of command error.

HOST | **UVC Device**

Step 1:

UVCX_VIDEO_CONFIG_PROBE GET_LEN

Returns the Length of struct_ UVCX_VIDEO_CONFIG

Step 2:

The host sends XU control to get the device capabilities.
UVCX_VIDEO_CONFIG_PROBE
GET_MAX

The device sends its supported max capabilities.

The host gets the Max configuration of the device

dwFrameInterval	= 333333
dwBitRate	= 1500000
bmHints	= 0
wConfigurationIndex	= 0
wWidth	= 1280
wHeight	= 720
wSliceSize	= 0
wSliceMode	= 0
wProfile	= 0x6400
wIFramePeriod	= 0
wEstimatedVideoDelay	= 40
wEstimatedMaxConfigDelay	= 250
bUsageType	= 0
bRateControlMode	= 0
bTemporalScaleMode	= 0
bSpatialScaleMode	= 0
bSNRScaleMode	= 0
bStreamMuxOption	= 0x0F
bStreamFormat	= 0
bEntropyCABAC	= 1
bTimestamp	= 1
bNumOfReorderFrame	= 0
bPreviewFlipped	= 0
bStreamID	= 0x07
bSpatialLayerRatio	= 0
wLeakyBucketSize	= 200

Note: All Max configuration may not be supported at the same time.

HOST

Step 3:

This point the host is aware of the maximum configuration supported by the device

The host configures the required parameters in config structure.

struct_ UVCX_VIDEO_CONFIG
dwFrameRate = 333333
wBitRate = 512000
wUsageType = 1 (Real Time)
wWidth = 1280
wHeight = 720
wProfile = 0x6400
bRateControlMode = 1
bStreamMuxOption = 0x03
bEntropyCABAC = 1
bStreamID =0x00

The host sends XU control UVCX_VIDEO_CONFIG_PROBE SET_CUR

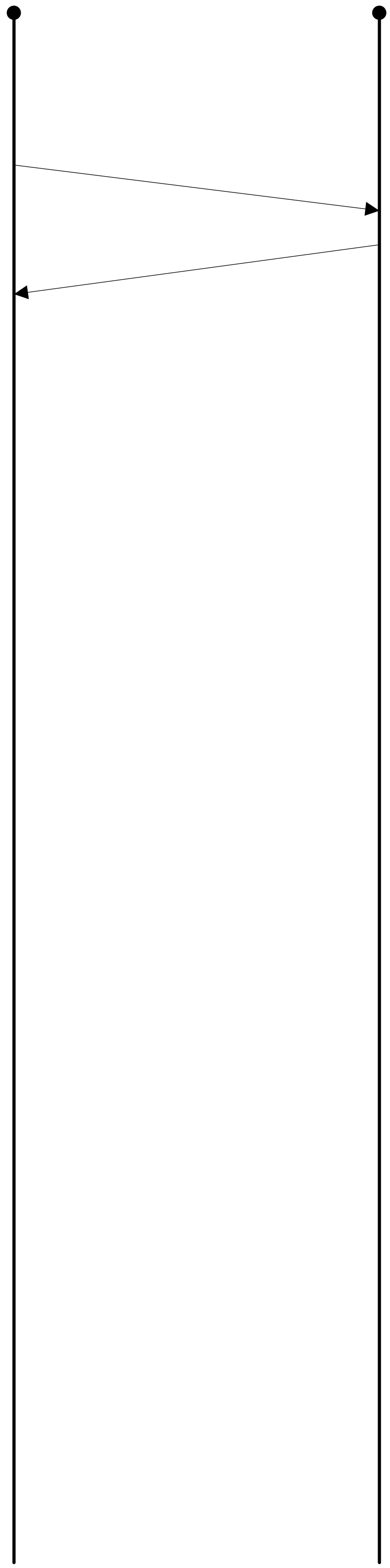
UVC Device

The device evaluates the SET_CUR parameters based on its capabilities. The device updates the structure for the capable configuration.

dwFrameInterval = 333333
dwBitRate = 512000
bmHints = 0
wConfigurationIndex = 1
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x6400
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x03
bStreamFormat = 0
bEntropyCABAC = 0
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x00
bSpatialLayerRatio = 0
wLeakyBucketSize = 200

HOST

UVC Device



Step 4:
The host shall request current parameter set of the device.
UVCX_VIDEO_CONFIG_PROBE
GET_CUR

The device sends back the present configuration, which is done in **step 3**.

struct_ UVCX_VIDEO_CONFIG
dwFrameInterval = 333333
dwBitRate = 512000
bmHints = 0
wConfigurationIndex = 1
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x6400
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x03
bStreamFormat = 0
bEntropyCABAC = 0
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x00
bSpatialLayerRatio = 0
wLeakyBucketSize = 200

The host validates the configuration and decides to go for other configuration as CABAC is not supported.

HOST

UVC Device

Step 5:
The host updates the configuration structure for new parameters
wProfile = 0x4200
Host sends XU control UVCX_VIDEO_CONFIG_PROBE SET_CUR

The device evaluates the SET_CUR parameters based on its capabilities. The device updates the structure for the capable configuration.

dwFrameInterval = 333333
dwBitRate = 512000
bmHints = 0
wConfigurationIndex = 1
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x4200
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 0
bSpatialScaleMode = 0
bSNRScaleMode = 0
bStreamMuxOption = 0x03
bStreamFormat = 0
bEntropyCABAC = 0
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x00
bSpatialLayerRatio = 0
wLeakyBucketSize = 200

Step 6:
The host shall request the current parameter set of the device. device.UVCX_VIDEO_CONFIG_PROBE GET_CUR

The device sends back the present configuration, which is negotiated in **step 5**.

The host checks the configuration and updates application for the change in requested config.

Step 7:
The host will send XU control UVCX_VIDEO_CONFIG_COMMIT SET_CUR

The device configures the encoder.

Step 8:
The host shall proceed with UVC PROBE/ COMMIT

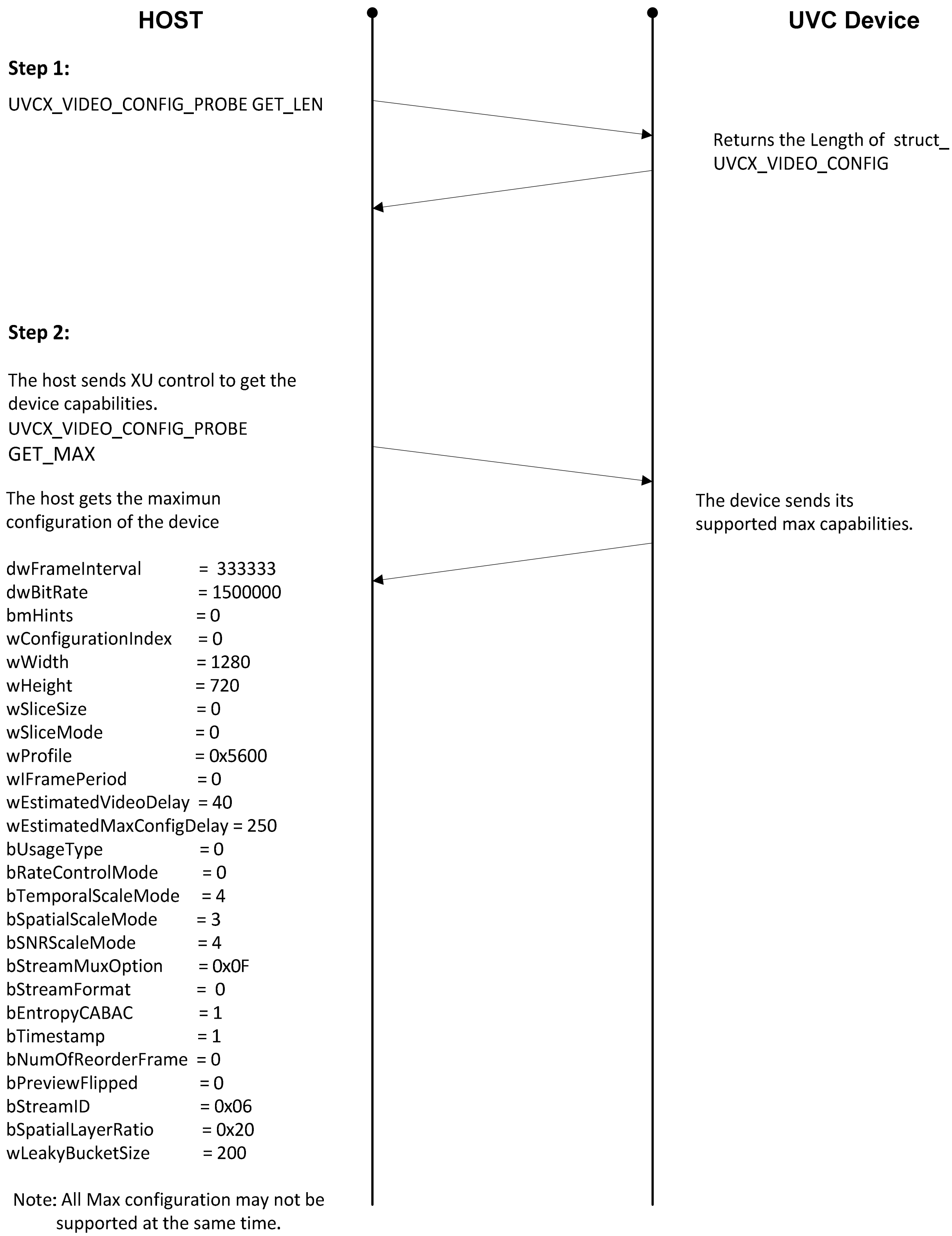
5.4 Programming Example for SVC

Device Capabilities:

- Single Payload
- H.264 Baseline Profile, Constrained Baseline Profile, High Profile, Scalable Baseline Profile
- 1280x720
- 30 Frames per second
- Single slice support
- CAVLC support only

Host Requested Configuration:

- Single Payload
- H.264 Scalable Baseline Profile
- 1280x720 (720p, 360p, and 180p)
- 7.5, 15, and 30 Frames per second
- Real-time use case


HOST
UVC Device
Step 1:
UVCX_VIDEO_CONFIG_PROBE GET_LEN
Returns the Length of struct_
UVCX_VIDEO_CONFIG
Step 2:
The host sends XU control to get the
device capabilities.
UVCX_VIDEO_CONFIG_PROBE
GET_MAX
The host gets the maximun
configuration of the device
The device sends its
supported max capabilities.
dwFrameInterval = 333333
dwBitRate = 1500000
bmHints = 0
wConfigurationIndex = 0
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x5600
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 0
bRateControlMode = 0
bTemporalScaleMode = 4
bSpatialScaleMode = 3
bSNRScaleMode = 4
bStreamMuxOption = 0x0F
bStreamFormat = 0
bEntropyCABAC = 1
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x06
bSpatialLayerRatio = 0x20
wLeakyBucketSize = 200
Note: All Max configuration may not be
supported at the same time.

HOST

UVC Device

Step 3:
This point the host is aware of the maximum configuration supported by the device.
The host configures the required parameters in configuration structure for primary payload (H.264).

struct_ UVCX_VIDEO_CONFIG
dwFrameInterval = 333333
wBitRate = 1500000
wWidth = 1280
wHeight = 720
wProfile = 0x5600
bUsageType = 1 (Real Time)
bRateControlMode = 1
bTemporalScaleMode = 3
bSpatialScaleMode = 3
bStreamMuxOption = 0x03
bSpatialLayerRatio = 0
wLeakyBucketSize = 200

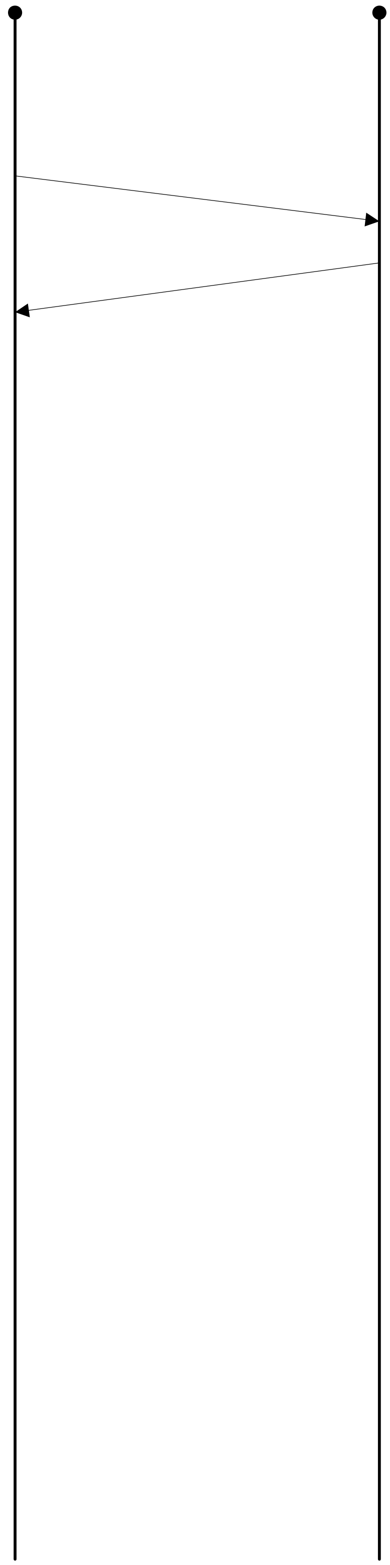
The host sends XU control UVCX_VIDEO_CONFIG_PROBE SET_CUR

The device evaluates the SET_CUR parameters based on its capabilities. The device updates the structure for the capable configuration.

dwFrameInterval = 333333
dwBitRate = 1500000
bmHints = 0
wConfigurationIndex = 1
wWidth = 1280
wHeight = 720
wSliceSize = 0
wSliceMode = 0
wProfile = 0x5600
wIFramePeriod = 0
wEstimatedVideoDelay = 40
wEstimatedMaxConfigDelay = 250
bUsageType = 1
bRateControlMode = 1
bTemporalScaleMode = 3
bSpatialScaleMode = 3
bSNRScaleMode = 0
bStreamMuxOption = 0x03
bStreamFormat = 0
bEntropyCABAC = 0
bTimestamp = 1
bNumOfReorderFrame = 0
bPreviewFlipped = 0
bStreamID = 0x00
bSpatialLayerRatio = 0x20
wLeakyBucketSize = 200

HOST

UVC Device



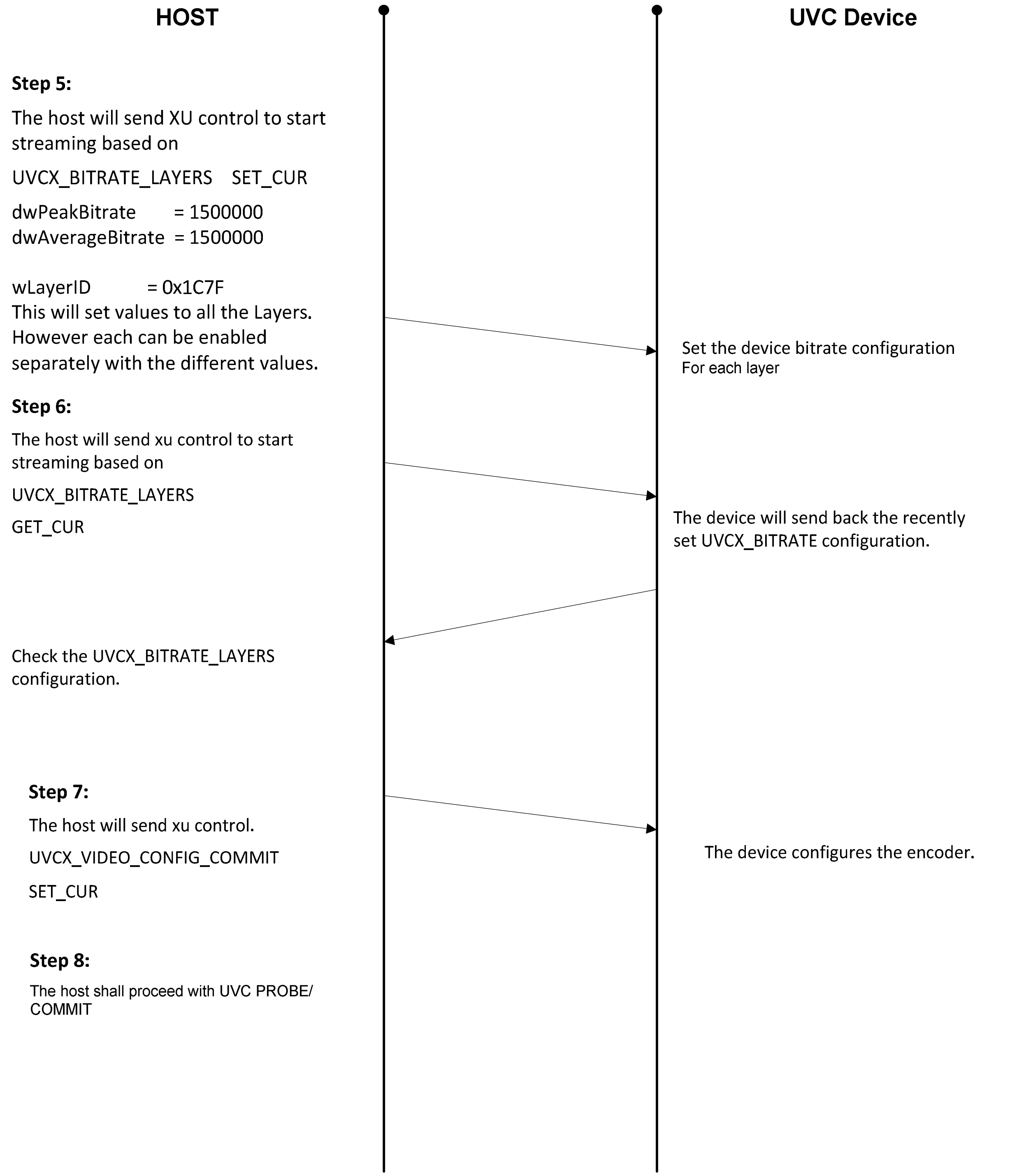
Step 4:
The Host shall request to get device capabilities.
UVCX_VIDEO_CONFIG_PROBE
GET_CUR

The device sends back the present configuration, which is done in **step 3**.

struct_ UVCX_VIDEO_CONFIG

dwFrameInterval	= 333333
dwBitRate	= 1500000
bmHints	= 0
wConfigurationIndex	= 1
wWidth	= 1280
wHeight	= 720
wSliceSize	= 0
wSliceMode	= 0
wProfile	= 0x5600
wIFramePeriod	= 0
wEstimatedVideoDelay	= 40
wEstimatedMaxConfigDelay	= 250
bUsageType	= 1
bRateControlMode	= 1
bTemporalScaleMode	= 3
bSpatialScaleMode	= 3
bSNRScaleMode	= 0
bStreamMuxOption	= 0x03
bStreamFormat	= 0
bEntropyCABAC	= 0
bTimestamp	= 1
bNumOfReorderFrame	= 0
bPreviewFlipped	= 0
bStreamID	= 0x00
bSpatialLayerRatio	= 0x20
wLeakyBucketSize	= 200

The Host validates the configuration..
The host has an option of changing the parameters again by following Step 2:


HOST
UVC Device
Step 5:
The host will send XU control to start streaming based on
UVCX_BITRATE_LAYERS SET_CUR
dwPeakBitrate = 1500000
dwAverageBitrate = 1500000
wLayerID = 0x1C7F
This will set values to all the Layers. However each can be enabled separately with the different values.
Set the device bitrate configuration For each layer
Step 6:
The host will send xu control to start streaming based on
UVCX_BITRATE_LAYERS
GET_CUR
The device will send back the recently set UVCX_BITRATE configuration.
Check the UVCX_BITRATE_LAYERS configuration.
Step 7:
The host will send xu control.
UVCX_VIDEO_CONFIG_COMMIT
SET_CUR
The device configures the encoder.
Step 8:
The host shall proceed with UVC PROBE/ COMMIT

6 Appendix-C

Audio Video Synchronization

An H.264 encoding webcam will induce significant latency in the video pipeline. This, in turn, exposes a new risk of A/V synch issues for both real-time streaming and file saving scenarios. The following section describes a solution that is derived from existing UVC 1.0 MJPEG payload header data and Probe & Commit data.

The solution below relies on two major features. First, pipeline delay must be calculated for use by the audio and video drivers when they timestamps the packets. Second, the clocks used to timestamp audio and video need to be correlated. Optimally, they are the same clock.

6.1 Calculating Video Delay

Video delay between sensor capture and driver timestamp is calculated in two parts. The delay on the camera due to pipeline processing and encoding, and the delay caused by USB transport and host processing.

The webcam generates two pieces of data that aid in calculating these two delays, Presentation Time Stamp (PTS) and Source Clock Reference (SCR). PTS and SCR are attached to the MJPEG payload header as described in the USB_Video_Payload_MJPEG_1.1 specification. PTS should be attached to every frame and SCR at the frequency required to address clock drift. An abbreviated definition is as follows:

Presentation Time Stamp (PTS)

The Source Time Clock (STC) in native device clock units when the raw frame capture begins. The PTS is in the same units as specified in the **dwClockFrequency** field of the Video Probe Control response.

Source Clock Reference (SCR)

The SCR contains two fields that enable the host to correlate between the device clock and the USB clock.

- STC: device's Source Time Clock value in units of the dwClockFrequency field of the Probe and Commit response of the device
- SOFTC: Start-of-Frame (SOF) token counter for USB, expressed in units of the 1KHz USB host controller clock.

Both these clocks are sampled at the SOF boundary when the video frame is sent over USB. While the UVC 1.1 specification states that the SOF is not required to match the 'current' frame number, *for this solution, the SOF must be the same frame number as that of the USB packet to which the SCR is attached*.

The delay of the video frame on the camera is calculated as:

DeviceDelay = (SCR_STC) - PTS Equation 1

This delay is expressed in units of dwClockFrequency, where dwClockFrequency is provided by the webcam as part of Probe & Commit. The delay caused by USB transport and processing is calculated as the difference between the SOF marker when the driver receives the video payload and the SOF in the SCR from the device:

TransportDelay = SOF_Driver – SOF_SCR Equation 2

TransportDelay is expressed in units of the 1 KHz USB host controller clock.

The total delay for each video frame between capture and the video class driver is calculated as the sum of the two delays calculated in Equation 1 and Equation 2 above.

Total Video Delay = DeviceDelay + TransportDelay Equation 3

6.1.1 Correlating between Device and PC clocks

Since the capture time of the video frame (PTS) is indicated by the device using the STC, and A/V sync will rely on PC clock values, we need to correlate the two clocks. The correlation 'constant' between PTS and QPC can be calculated as the most recent Total Video Delay.

Clock Correlation Constant (CCC) = Total Video Delay Equation 4

6.1.2 Video Time Stamping

The timestamp applied by the video driver to the current video frame is calculated as the timestamp for the current frame – CCC.

Timestamp for current frame = PTS - CCC Equation 5

The timestamp calculated above is applied to all NAL Units belong to the same picture. The camera indicates a new picture by toggling the FID between 0 and 1 on the UVC payload header.

6.2 Audio Time Stamping

The USB audio class driver performs the final audio time stamp. For this solution to work the audio timestamp is the current PC clock time minus the delay declared by the audio device (if available). The delay parameter is important if the audio path includes delay on the device, or on the host before the audio driver sees the data.