DirectX Video Acceleration Specification for H.264/MPEG-4 AVC Multiview Video Coding (MVC), Including the Stereo High Profile

Gary J. Sullivan and Yongjun Wu
Microsoft Corporation
March 2011

Applies to:
DirectX Video Acceleration

Summary: Defines extensions to DirectX Video Acceleration (DXVA) to support Multiview Video Coding (MVC) according to the H.264/AVC specification of MVC. Support of the H.264/AVC Stereo High Profile is an important special case. This document describes high-level design concepts and specific MVC extensions to the DXVA interfaces and data structures for H.264/AVC decoding. Only off-host VLD profiles for H.264/AVC-based MVC decoding are specified.

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, AS TO THE INFORMATION IN THIS DOCUMENT.

Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred.

Microsoft does not make any representation or warranty regarding specifications in this document or any product or item developed based on these specifications. Microsoft disclaims all express and implied warranties, including but not limited to the implied warranties or merchantability, fitness for a particular purpose and freedom from infringement. Without limiting the generality of the foregoing, Microsoft does not make any warranty of any kind that any item developed based on these specifications, or any portion of a specification, will not infringe any copyright, patent, trade secret or other intellectual property right of any person or entity in any country. It is your responsibility to seek licenses for such intellectual property rights where appropriate. Microsoft shall not be liable for any damages arising out of or in connection with the use of these specifications, including liability for lost profit, business interruption, or any other damages whatsoever. Some states do not allow the exclusion or limitation of liability or consequential or incidental damages; the above limitation may not apply to you.

Contents

1. Introduction

This document defines extensions to DirectX® Video Acceleration (DXVA) to support decoding of the Multiview Video Coding (MVC) extensions to the H.264/MPEG-4 Advanced Video Coding (AVC) specification. This includes support of the H.264/AVC Stereo High Profile as an important special case. This document is currently in draft form.

In this document, the term "ordinary H.264/AVC bitstream" refers to an H.264/AVC bitstream that does not use the SVC and MVC extensions that are specified in Annexes G and H of the H.264/AVC specification.

This document assumes that the reader is familiar with the H.264/AVC standard specification and its MVC extensions in Annex H, and the DXVA specification for ordinary H.264/AVC bitstream decoding.

2. Normative references

2.1 The H.264/AVC standard

The Multiview Video Coding (MVC) extensions to the H.264/AVC standard (referred to hereafter as MVC) are specified in Annex H of ITU-T Rec. H.264, ISO/IEC 14496-10: *Advanced video coding for generic audiovisual services*. The standard is available at http://www.itu.int/rec/T-REC-H.264. Unless otherwise specified, this document refers to the edition approved by ITU-T in March 2010, which is the version given in the URL.

2.2 The DXVA Specification for Ordinary H.264/AVC Bitstream Decoding

The DXVA specification for ordinary H.264/AVC bitstream decoding is found in the Microsoft publication *DirectX Video Acceleration Specification for H.264/AVC Decoding,*" available at http://go.microsoft.com/fwlink/?LinkId=107379. Unless otherwise specified, this document refers to the edition that includes updates as of December 2010.

3. Overview of the MVC Extensions of H.264/AVC

This section provides a general overview of the MVC extensions.

3.1 Basic Design Structure

The MVC design is specified primarily in Annex H of the H.264/MPEG-4 Advanced Video Coding (AVC) standard. It closely follows the design of the ordinary, non-MVC form of the H.264/AVC standard. The data for a coded video sequence is structured into *access units*. These units are sent in the bitstream in decoding order. Within an access unit, one “base view” component is formatted as an ordinary H.264/AVC coded picture. This component represents the base view, and can be decoded by an ordinary non-MVC decoder that supports the H.264/AVC High Profile. Within the same access unit, there are one or more additional view components, each of which represents an additional non-base view of a multiview encoding for the same instant in time.

Other than minor differences in the high-level syntax, the encoding format for a non-base view is basically the same as that of the base view. The decoding process for each view component is also basically the same as the decoding process for an ordinary non-MVC picture.

For each non-base view, the view components of other views within the same access unit, and the preceding view components for the same non-base view (in decoding order), can be used as reference pictures for inter-picture prediction during the decoding process. Therefore, the design supports both temporal prediction within a single view across different access units, and inter-view prediction across different views within the same access unit. For the syntax and decoding process at the macroblock level and levels below that, the design is the same as in the ordinary non-MVC design. Only the high-level header formats are different.

Two MVC profiles are specified in the standard. They are described next.

3.2 The Multiview High Profile

The Multiview High Profile was the first specified MVC profile. It was basically complete in 2008. This profile supports an arbitrary number of views. It supports all coding features of the H.264/AVC High Profile except the interlaced coding tools (that is, it requires the frame_mbs_only_flag syntax element of the sequence parameter set to be equal to 1).

3.3 The Stereo High Profile

Soon after the Multiview High Profile was created, a second MVC profile was created to serve near-term requirements for the important special case of stereoscopic video coding. The result was the Stereo High Profile. It differs from the Multiview High Profile in two primary ways:

- The number of supported views is limited to two.
- The interlaced coding tools associated with frame_mbs_only_flag equal to 0 are also supported.

At approximately the same time that the standardization drafting work was completed (late 2009), the Stereo High Profile was adopted by the Blu-ray Disc Association for use by 3D-enabled Blu-ray disc players.

4. Basic DXVA design approach

4.1 Consistency with DXVA design for ordinary H.264/AVC bitstream decoding

This section provides an overview of the design for DXVA decoding of MVC bitstreams. It is intended as background overview information, and may be helpful for understanding the following sections. In the case of conflicts, later sections of this document override this section.

MVC DXVA support provides consistency with the DXVA design for ordinary H.264/AVC bitstream decoding. As much as possible, it uses the same basic data flow and similar data structures.

4.2 Support for Multiview High Profile versus Stereo High Profile

In the near term, support for the Stereo High Profile may be a higher priority than support for the more general Multiview High Profile. However, for design consistency,

we consider it desirable to use a consistent approach for both Stereo High Profile and Multiview High Profile.

4.3 Basic Handling of Frame Buffers

Because an MVC view component has basically the same decoding process as an ordinary H.264/AVC coded picture, it will be treated the same way in DXVA decoding. That is, each decoded frame buffer contains one decoded MVC view component, instead of a complete multi-view access unit, and is the decoding target identified in one DXVA picture parameters data structure. Frame buffers that hold view components used as inter-view or temporal prediction references in the decoding process are identified in a decoded picture buffer (DBP) list in the picture parameters buffer. As with the current DXVA 2 design, the accelerator is responsible for tracking the sequential dependencies between the decoding operations. In particular, the accelerator ensures that read operations used in prediction processes are not performed until the write operations to fill the frame buffer with the correct decoded data are completed.

4.4 Differences In Relation To Ordinary H.264/AVC DXVA Decoding

An MVC-enabled DXVA software decoder or hardware accelerator will use approximately *NumViews* times the number of frame buffers that would be used for the ordinary decoding of an H.264/AVC bitstream with similar temporal referencing characteristics, where *NumViews* is the number of views to be decoded (per sub-clause H.10.2 of the H.264/AVC standard).

The memory capacity requirement for support of a given level is approximately doubled, as specified by the *mvcScaleFactor* variable that is set equal to 2 in sub-clause H.10.2 of the H.264/AVC standard. For example, at level 4.1 or 4.2, the maximum number of occupied frame buffers in the DPB is 8 at a resolution of 1920x1088 when *NumViews* is greater than or equal to 2.

As with ordinary H.264/AVC, the number of frame buffers can increase if the decoder is operated with a picture size that is smaller than the maximum picture size supported by the level. However, our current understanding, based on the outcome of the October 2010 meetings of MPEG and VCEG, is that for the currently-specified MVC profiles (the Multiview High and Stereo High Profiles), the total number of view components held in the decoded picture buffer for use as references for predicting other pictures can never exceed 16 frames (32 fields). This maximum limit is the same as for ordinary (non-MVC) H.264/AVC decoding.

The host software decoder requires modification to track and control the current DPB state for the various views and to parse the sequence parameter set extensions, picture parameter set extensions, slice headers, prefix NAL units, reference picture list initialization and reordering, and (possibly) associated additional SEI messages. Accelerator-side modifications primary relate to header format issues and the increased memory requirement.

Correct rendering of multiview video will obviously differ from monoscopic video. However, the rendering process is outside the scope of this specification.

4.5 Support for Off-Host VLD Operation Only

Previous DXVA specifications have defined modes of DXVA operation other than off-host VLD operation, such as DXVA_ModeH264_MoComp_NoFGT and

DXVA_ModeH264_IDCT_NoFGT profiles for ordinary H.264/AVC bitstreams. Over time, the level of industry interest in supporting such modes seems to have waned. Therefore, we do not plan to specify such modes for MVC decoding.

4.6 No Planned Support for Film-Grain Synthesis

Over time, the level of industry interest in supporting film-grain synthesis (as in the DXVA_ModeH264_VLD_FGT profile for ordinary H.264/AVC bitstreams) seems to have waned. Therefore, we do not plan to specify such modes of operation for MVC decoding. Feedback is especially requested on this topic, because it should not be difficult to add DXVA modes to support this functionality.

5. DXVA Extensions for MVC

This section specifies the MVC extensions to the data flow and data structures for DXVA H.264/AVC decoding. This section assumes that the reader is already familiar with the DXVA specification for ordinary H.264/AVC bitstream decoding. The DXVA process for MVC decoding follows the DXVA specifications for ordinary H.264/AVC decoding except as specified in the following sections.

5.1 Extensions to DXVA Decoding Operations

Section 1.5 of the DXVA specification for ordinary H.264/AVC bitstream decoding specifies the basic sequence of operations for H.264/AVC decoding. For MVC view component decoding, the **BeginFrame**, **Execute**, and **EndFrame** calls shall be used to perform decoding of each individual view component (picture), instead of a complete primary coded picture. (In the MVC definition, an access unit is "*a set of NAL units* that are consecutive in *decoding order* and contain exactly one *primary coded picture* consisting of one or more *view components*.") This design is intended to minimize the API/DDI changes, and to make MVC view component decoding consistent with H.264/AVC picture decoding.

The decoding process for each view component (picture) is similar to that for ordinary H.264/AVC DXVA decoding, with the input of a picture parameters buffer, a quantization matrix buffer, and a slice control buffer. The picture parameters buffer uses a **DXVA_PicParams_H264_MVC** structure (specified in section 5.4), instead of the **DXVA_PicParams_H264** that is used in ordinary H.264/AVC DXVA decoding.

Off-host VLD profiles for MVC decoding involves sending the following buffers to the accelerator,

- One picture parameters buffer of type **DXVA_PicParams_H264_MVC**.
- One quantization matrix buffer.
- One slice control buffer.
- One or more bitstream data buffers.

5.2 Extensions to Configuration Parameters

This section describes the extensions to the configuration parameters for MVC decoding as previously specified in section 2.0 of the DXVA specification for ordinary H.264/AVC bitstream decoding .

bConfigBitstreamRaw

Equal to 3 for MVC long slice header format in VLD DXVA profiles. Equal to 4 for MVC short slice header formats in VLD DXVA profiles.

bConfigResidDiffHost

Equal to 0 for MVC VLD DXVA profiles (with **bConfigBitstreamRaw** equal to 3 or 4).

bConfigResidDiffAccelerator

Equal to 1 for MVC VLD DXVA profiles (with **bConfigBitstreamRaw** equal to 3 or 4).

5.3 Extensions to the DXVA_PicEntry_H264 Structure

This section describes the use of the DXVA picture entry data structure for MVC decoding, as previously specified in section 3.0 of the DXVA specification for ordinary H.264/AVC bitstream decoding.

```
typedef struct _DXVA_PicEntry_H264 {
union {
    struct {
           UCHAR Index7Bits : 7;
           UCHAR AssociatedFlag : 1;
    };
    UCHAR bPicEntry;
    };
} DXVA_PicEntry_H264, *LPDXVA_PicEntry_H264;
```

Index7Bits is an index that identifies an uncompressed surface for the **CurrPic** or **RefFrameList** member of the picture parameters structure (defined in section 4.0 of the DXVA specification for ordinary H.264/AVC bitstream decoding) or the **RefPicList** member of the slice control data structure (defined in section 6.0 of the DXVA specification for ordinary H.264/AVC bitstream decoding).

When **Index7Bits** is used in the **CurrPic** and **RefFrameList** members of the picture parameters structure, the value directly specifies the DXVA index of an uncompressed surface.

When **Index7Bits** is used in the **RefPicList** member of the slice control data structure, the value identifies the surface indirectly, as an index into the **RefFrameList** array of the associated picture parameters structure.

In all cases, when **Index7Bits** does not contain a valid index, it shall be equal to 127.

The valid values for **Index7Bits** shall be in the range of 0 to 126, inclusive. Because the maximum number of reference frames is 16, the length of **Index7Bits** could hypothetically support up to 111 views. However, for purposes of this specification, it is assumed that 16 views are sufficient for practical purposes, and software decoders shall not send video data to the accelerator that reflects the use of more than 16 views.

Note The constraint not to use more than 16 views is not a constraint on the value of **Index7Bits**. Instead, it is a constraint on the number of views that are actually active in the bitstream.

5.4 Picture Parameters Data Structure

The **DXVA_PicParams_H264_MVC** structure provides the picture-level parameters of a compressed picture for H.264/AVC MVC decoding.

5.4.1 Syntax

```
/* H.264 MVC picture parameters structure */
typedef struct _DXVA_PicParams_H264_MVC {
  USHORT  wFrameWidthInMbsMinus1;
  USHORT  wFrameHeightInMbsMinus1;
  DXVA_PicEntry_H264  CurrPic; /* flag is bot field flag */
  UCHAR   num_ref_frames;

  union {
    struct {
      USHORT  field_pic_flag                : 1;
      USHORT  MbaffFrameFlag                : 1;
      USHORT  residual_colour_transform_flag : 1;
      USHORT  sp_for_switch_flag            : 1;
      USHORT  chroma_format_idc             : 2;
      USHORT  RefPicFlag                    : 1;
      USHORT  constrained_intra_pred_flag   : 1;

      USHORT  weighted_pred_flag            : 1;
      USHORT  weighted_bipred_idc           : 2;
      USHORT  MbsConsecutiveFlag            : 1;
      USHORT  frame_mbs_only_flag           : 1;
      USHORT  transform_8x8_mode_flag       : 1;
      USHORT  MinLumaBipredSize8x8Flag      : 1;
      USHORT  IntraPicFlag                  : 1;
    };
    USHORT  wBitFields;
  };
  UCHAR  bit_depth_luma_minus8;
  UCHAR  bit_depth_chroma_minus8;

  USHORT Reserved16Bits;
  UINT   StatusReportFeedbackNumber;

  DXVA_PicEntry_H264  RefFrameList[16]; /* flag LT */
  INT    CurrFieldOrderCnt[2];
  INT    FieldOrderCntList[16][2];

  CHAR   pic_init_qs_minus26;
  CHAR   chroma_qp_index_offset;   /* also used for QScb */
  CHAR   second_chroma_qp_index_offset; /* also for QScr */
  UCHAR  ContinuationFlag;

/* remainder for parsing */
  CHAR   pic_init_qp_minus26;
  UCHAR  num_ref_idx_l0_active_minus1;
  UCHAR  num_ref_idx_l1_active_minus1;
  UCHAR  Reserved8BitsA;

  USHORT FrameNumList[16];
  UINT   UsedForReferenceFlags;
  USHORT NonExistingFrameFlags;
  USHORT frame_num;
```

```
    UCHAR  log2_max_frame_num_minus4;
    UCHAR  pic_order_cnt_type;
    UCHAR  log2_max_pic_order_cnt_lsb_minus4;
    UCHAR  delta_pic_order_always_zero_flag;

    UCHAR  direct_8x8_inference_flag;
    UCHAR  entropy_coding_mode_flag;
    UCHAR  pic_order_present_flag;
    UCHAR  num_slice_groups_minus1;

    UCHAR  slice_group_map_type;
    UCHAR  deblocking_filter_control_present_flag;
    UCHAR  redundant_pic_cnt_present_flag;
    UCHAR  Reserved8BitsB;

    USHORT slice_group_change_rate_minus1;
    /* SliceGroupMap is not needed for MVC, as MVC is for high profile
       only */

    /* Following are H.264 MVC Specific parameters */
    UCHAR   num_views_minus1;
    USHORT  view_id[16];
    UCHAR   num_anchor_refs_l0[16];
    USHORT  anchor_ref_l0[16][16];
    UCHAR   num_anchor_refs_l1[16];
    USHORT  anchor_ref_l1[16][16];
    UCHAR   num_non_anchor_refs_l0[16];
    USHORT  non_anchor_ref_l0[16][16];
    UCHAR   num_non_anchor_refs_l1[16];
    USHORT  non_anchor_ref_l1[16][16];

    USHORT curr_view_id;
    UCHAR  anchor_pic_flag;
    UCHAR  inter_view_flag;
    USHORT ViewIDList[16];

} DXVA_PicParams_H264_MVC, *LPDXVA_PicParams_H264_MVC;
```

5.4.2 Semantics

Except as noted in this section, all **DXVA_PicParams_H264_MVC** structure members have the same semantics as the corresponding members of the **DXVA_PicParams_H264** structure. The **DXVA_PicParams_H264** structure is defined in section 4.0 of the DXVA specification for ordinary H.264/AVC bitstream decoding. The following **DXVA_PicParams_H264_MVC** structure members have semantics that differ from the **DXVA_PicParams_H264** structure**:**

num_slice_groups_minus1

Must be equal to 0.

Note For both multi-view high profile and stereo high profile, this parameter must be zero. For this reason, the **DXVA_PicParams_H264_MVC** structure does not contain a **SliceGroupMap** array, as found in the **DXVA_PicParams_H264** structure.

num_views_minus1

The value **num_views_minus1** plus 1 specifies the maximum number of coded views in the coded video sequence. The value of **num_view_minus1** shall be in the range of 0 to 15, inclusive, because the maximum number of supported views is restricted to 16 in this specification. The actual number of views in the coded video sequence may be less than or equal to **num_views_minus1** plus 1.

view_id[i]

Specifies the view_id of the view with *VOIdx* equal to *i*. The array size is 16, which is the maximum number of views supported in this specification. The value of **view_id**[i] shall be in the range of 0 to 1023, inclusively. The array shall contain valid values from index *i* = 0 to **num_views_minus1**, inclusively. The host shall set the unused elements from index *i* = **num_views_minus1**+1 to 15, inclusively, to the invalid value 0xFFFF.

num_anchor_refs_l0[i]

Specifies the number of view components for inter-view prediction in the initialized RefPicList0 in decoding anchor view components with *VOIdx* equal to *i*. The value of **num_anchor_refs_l0**[i] shall not be greater than 15. The value of **num_anchor_refs_l0**[0] shall be 0. The array size is 16, which is the maximum number of views supported in this specification.

anchor_ref_l0[i][j]

Specifies the view_id of the *j*th view component for inter-view prediction in the initialized RefPicList0 in decoding anchor view components with *VOIdx* equal to *i*. The array size is 16x16, because the maximum number of views supported in this specification is 16 and the value of **num_anchor_refs_l0**[*i*] shall not exceed 15.

num_anchor_refs_l1[i]

Specifies the number of view components for inter-view prediction in the initialized RefPicList1 in decoding anchor view components with *VOIdx* equal to *i*. The value of **num_anchor_refs_l1**[i] shall not be greater than 15. The value of **num_anchor_refs_l1**[0] shall be 0. The array size is 16, which is the maximum number of views supported in this specification.

anchor_ref_l1[i][j]

Specifies the view_id of the *j*th view component for inter-view prediction in the initialized RefPicList1 in decoding an anchor view component with *VOIdx* equal to *i*. The array size is 16x16, because the maximum number of supported views is 16 and the value of num_anchor_refs_l1[i] shall not exceed 15.

num_non_anchor_refs_l0[i]

Specifies the number of view components for inter-view prediction in the initialized RefPicList0 in decoding non-anchor view components with *VOIdx* equal to *i*. The value of **num_non_anchor_refs_l0**[i] shall not be greater than 15. The value of **num_non_anchor_refs_l0**[0] shall be 0. The array size is 16, which is the maximum number of views supported in this specification.

non_anchor_ref_l0[i][j]

Specifies the view_id of the *j*th view component for inter-view prediction in the initialized RefPicList0 in decoding non-anchor view components with *VOIdx* equal to *i*. The array size is 16x16, because the maximum number of supported views is 16 and the value of **num_non_anchor_refs_l0**[i] shall not exceed 15.

num_non_anchor_refs_l1[i]

Specifies the number of view components for inter-view prediction in the initialized RefPicList1 in decoding non-anchor view components with *VOIdx* equal to *i*. The value of **num_non_anchor_refs_l1**[i] shall not be greater than 15. The value of **num_non_anchor_refs_l1**[0] shall be 0. The array size is 16, which is the maximum number of views supported in this specification.

non_anchor_ref_l1[i][j]

Specifies the view_id of the *j*th view component for inter-view prediction in the initialized RefPicList1 in decoding non-anchor view components with *VOIdx* equal to *i*. The array size is 16x16, the maximum number of supported views is 16 and the value of **num_non_anchor_refs_l1**[i] shall not exceed 15.

Note The **num_views_minus1**, **view_id**, **num_anchor_refs_l0**, **anchor_ref_l0**, **num_anchor_refs_l1**, **anchor_ref_l1**, **num_non_anchor_refs_l0**, **non_anchor_ref_l0**, **num_non_anchor_refs_l1**, and **non_anchor_ref_l1** parameters are used primarily for short slice-header mode with **bConfigBitstreamRaw** equal to 4, described in Section 5.5. In this mode, reference list initialization and reordering for each slice need all these parameters. For long slice-header mode with **bConfigBitstreamRaw** equal to 3, the accelerator might decide to ignore all the parameters, because the final reference lists for each slice after initialization and reordering are sent in the **DXVA_Slice_H264_Long** data structure.

---

curr_view_id

Specifies the view identifier for current picture.

anchor_pic_flag

If 1, specifies that the current picture is an anchor picture.

inter_view_flag

If 0, the current view component is not used for inter-view prediction by any other view component in the current access unit. If 1, the current view component may be used for inter-view prediction by other view components in the current access unit.

ViewIDList[i]

Specifies the view_id of each entry in **RefFrameList**. The array size is 16.

Clarifications on the use of individual parameters from the **DXVA_PicParams_H264** structure are as follows:

num_ref_frames

Refers to the max_num_ref_frames syntax element in the MVC specification. This specifies the maximum number of short-term and long-term reference frames, complementary reference field pairs, and non-paired reference fields that may be used by the decoding process for inter prediction of any view component in the coded video sequence. The max_num_ref_frames syntax element also determines the sliding window size of the sliding window operation, as specified in sub-clause H.8.3 of the H.264/AVC specification. The value of max_num_ref_frames shall be in the range of 0 to 16, inclusive, for MVC decoding. However, **num_ref_frames** could be changed by the host software decoder to some value larger than the value of max_num_ref_frames that may be present within the video bitstream. The hardware accelerator shall still decode the video correctly if **num_ref_frames** is larger than the max_num_ref_frames value encoded in the video bitstream (up to a maximum value of 16).

RefFrameList

This list of reference uncompressed buffers shall include all view components that may be used for inter-view prediction or temporal inter-picture prediction of the current view component and of any subsequent view component of any view in the bitstream. Pictures that are potentially used for inter-view prediction may have the same **PicOrderCnt** value as the current view component but a different **view_id** value.

FrameNumList

For each entry in **RefFrameList**, the corresponding entry in **FrameNumList** contains the value of FrameNum or LongTermFrameIdx, depending on the value of **AssociatedFlag** in the **RefFrameList** entry. Becayse **RefFrameList** may contain pictures in different views, several entries in **FrameNumList** may have the same FrameNum or LongTermFrameIdx. The **ViewIDList** array is used to associate each entry with the FrameNum or LongTermFrameIdx.

5.5 Extensions to Slice Control Data Structure

This section describes the use of the slice control data structure for MVC decoding as previously specified in section 6.0 of the DXVA specification for ordinary H.264/AVC bitstream decoding.

bConfigBitstreamRaw	Slice Data Control Structure
3	**DXVA_Slice_H264_ Long**
4	**DXVA_Slice_H264_ Short**

When **bConfigBitstreamRaw** is 3 or 4, the slice control buffer is accompanied by a bitstream data buffer. When **bConfigBitstreamRaw** is 3, the decoder uses the same **DXVA_Slice_H264_Long** slice control data structure as specified in section 6.1 of the DXVA specification for ordinary H.264/AVC bitstream decoding. When **bConfigBitstreamRaw** is 4, MVC decoding uses the same **DXVA_Slice_H264_Short** slice control data structure as specified for H.264/AVC DXVA decoding in section 6.1 of the DXVA specification for ordinary H.264/AVC bitstream decoding. For both structures, the meaning of the flags is the same as for ordinary H.264/AVC bitstream decoding.

5.6 Extensions to Restricted Mode Profiles

This section describes restricted-mode profiles for MVC decoding, as previously specified in section 13.0 of the DXVA specification for ordinary H.264/AVC bitstream decoding.

The GUIDs that identify these profiles will be defined in the header file dxva.h and are included here for reference.

5.6.1 DXVA_ModeH264_VLD_Stereo_High_Progressive Profile

This profile supports the features necessary for a decoder that conforms to the Stereo High Profile, except for imposing the additional constraint that **frame_mbs_only_flag** shall be 1. In this profile, the accelerator performs bitstream parsing, inverse quantization scaling, inverse transform processing, motion compensation, and deblocking, without film-grain synthesis.

1. Configuration parameters:
   - **bConfigBitstreamRaw** = 3 for long slice header mode or 4 for short slice header mode.

- **bConfigMBcontrolRasterOrder** = 0
- **bConfigResidDiffHost** = 0
- **bConfigSpatialResid8** = 0
- **bConfigResid8Subtraction** = 0
- **bConfigSpatialHost8or9Clipping** = 0
- **bConfigSpatialResidInterleaved** = 0
- **bConfigIntraResidUnsigned** = 0
- **bConfigResidDiffAccelerator** = 1
- **bConfigHostInverseScan** = 1
- **bConfigSpecificIDCT** = 2
- **bConfig4GroupedCoefs** = 0

2. All data buffers shall contain only data that is consistent with the constraints specified for the Stereo High Profile, with the additional constraint of **frame_mbs_only_flag** equal to 1.
3. **num_views_minus1** shall be 0 or 1.
4. Film-grain synthesis is not supported in this profile.

The number of supported views is limited to two in the DXVA_ModeH264_VLD_Stereo_High_Progressive profile. The interlaced coding tools associated with **frame_mbs_only_flag** equal to 0 are not supported.

The GUID for this profile is currently not defined in the Windows SDK. To use this profile, use the following declaration:

```
// {D79BE8DA-0CF1-4c81-B82A-69A4E236F43D}
DEFINE_GUID( DXVA_ModeH264_VLD_Stereo_Progressive_NoFGT,
0xd79be8da, 0xcf1, 0x4c81, 0xb8, 0x2a, 0x69, 0xa4, 0xe2, 0x36, 0xf4,
0x3d);
```

5.6.2 DXVA_ModeH264_VLD_Stereo_High Profile

This profile supports the features necessary for a decoder that conforms to the Stereo High Profile. In this profile, the accelerator performs bitstream parsing, inverse quantization scaling, inverse transform processing, motion compensation, and deblocking, without film-grain synthesis.

1. Configuration parameters:
   - **bConfigBitstreamRaw** = 3 for long slice header mode or 4 for short slice header mode.
   - **bConfigMBcontrolRasterOrder** = 0
   - **bConfigResidDiffHost** = 0
   - **bConfigSpatialResid8** = 0
   - **bConfigResid8Subtraction** = 0
   - **bConfigSpatialHost8or9Clipping** = 0
   - **bConfigSpatialResidInterleaved** = 0
   - **bConfigIntraResidUnsigned** = 0
   - **bConfigResidDiffAccelerator** = 1
   - **bConfigHostInverseScan** = 1
   - **bConfigSpecificIDCT** = 2
   - **bConfig4GroupedCoefs** = 0
2. All data buffers shall contain only data that is consistent with the constraints specified for the Stereo High Profile of H.264/AVC.
3. **num_views_minus1** shall be 0 or 1.
4. Film-grain synthesis is not supported in this profile.

The number of supported views is limited to two in the Stereo High Profile. The interlaced coding tools associated with **frame_mbs_only_flag** equal to 0 are also supported.

The GUID for this profile is currently not defined in the Windows SDK. To use this profile, use the following declaration:

```
// {F9AACCBB-C2B6-4cfc-8779-5707B1760552}
DEFINE_GUID( DXVA_ModeH264_VLD_Stereo_NoFGT,
0xf9aaccbb, 0xc2b6, 0x4cfc, 0x87, 0x79, 0x57, 0x7, 0xb1, 0x76, 0x5,
0x52);
```

Hardware accelerator drivers that expose support for this capability declaration GUID should also expose the previously-specified GUID of DXVA_ModeH264_VLD_NoFGT, as decoders conforming to the Stereo High Profile at a specific level shall also be able to decode all bitstreams of the High Profile, Main Profile, and Constrained Baseline Profile. Therefore DXVA_ModeH264_VLD_Stereo_NoFGT covers all feaures defined for the DXVA_ModeH264_VLD_NoFGT profile of DXVA.

Hardware accelerator drivers that expose support for this capability declaration GUID should also expose the GUID of DXVA_ModeH264_VLD_Stereo_Progressive_NoFGT, as decoders conforming to the Stereo High Profile at a specific level shall be able to decode all bitstreams of the Stereo High Profile in which frame_mbs_only_flag is equal to 1.

5.6.3 DXVA_ModeH264_VLD_Multiview_High Profile

This profile supports the features necessary for a decoder that conforms to the Multiview High Profile. In this profile, the accelerator performs bitstream parsing, inverse quantization scaling, inverse transform processing, motion compensation, and deblocking, without film-grain synthesis.

Up to 16 views are supported in the DXVA_ModeH264_VLD_Multiview_High_Profile. This profile requires the frame_mbs_only_flag syntax element of the sequence parameter set to be equal to 1.

1. Configuration parameters:
   - **bConfigBitstreamRaw** = 3 for long slice header mode or 4 for short slice header mode.
   - **bConfigMBcontrolRasterOrder** = 0
   - **bConfigResidDiffHost** = 0
   - **bConfigSpatialResid8** = 0
   - **bConfigResid8Subtraction** = 0
   - **bConfigSpatialHost8or9Clipping** = 0
   - **bConfigSpatialResidInterleaved** = 0
   - **bConfigIntraResidUnsigned** = 0
   - **bConfigResidDiffAccelerator** = 1
   - **bConfigHostInverseScan** = 1
   - **bConfigSpecificIDCT** = 2
   - **bConfig4GroupedCoefs** = 0
2. All data buffers shall contain only data that is consistent with the constraints specified for the Multiview High Profile of the H.264/AVC specification.
3. **num_views_minus1** shall be in the range of 0 to 15, inclusively.
4. Film-grain synthesis is not supported in this profile.

The associated GUID definition for the corresponding entry in the dxva.h header file is as follows:

The GUID for this profile is currently not defined in the Windows SDK. To use this profile, use the following declaration:

```
// {705B9D82-76CF-49D6-B7E6-AC8872DB013C}
DEFINE_GUID(DXVA_ModeH264_VLD_Multiview_NoFGT,
0x705b9d82, 0x76cf, 0x49d6, 0xb7, 0xe6, 0xac, 0x88, 0x72, 0xdb, 0x1,
0x3c);
```

Hardware accelerator drivers that expose support for this capability declaration GUID should also expose the GUID of DXVA_ModeH264_VLD_Stereo_Progressive_NoFGT, as decoders conforming to Multiview High Profile at a specific level shall be able to decode all bitstreams of the Stereo High Profile in which the frame_mbs_only_flag is equal to 1.

Annex A Compatibility with H.264/AVC Decoding and Display

Over time, the industry has released products supporting DXVA modes for H.264/AVC decoding. In some instances, a software decoder may use available DXVA modes for H.264/AVC to decode MVC bitstreams, by performing a conversion process. Section A.1 describes one such conversion process.

The industry has also released products supporting single-view display. If the hardware is capable of MVC DXVA decoding, an MVC bitstream may be decoded in a single-view mode, by performing a conversion to make it compatible with single-view display hardware. Section A.2 describes one such conversion process.

A.1 Host Conversion from H.264/AVC-based MVC to H.264/AVC

When hardware is capable of DXVA operation for ordinary H.264/AVC decoding but not for MVC decoding, the host software decoder may take the compressed data of the base view of an MVC bitstream and send it through the DXVA interface, using H.264/AVC data structures with correct parameter settings. This process enables the host software decoder to use the available hardware acceleration for single-view decoding.

In this case, the host software decoder is responsible for converting the MVC bitstream to the form of an ordinary H.264/AVC bitstream, and for using the correct settings of the H.264/AVC DXVA data structures. Hardware accelerators capable of ordinary H.264/AVC DXVA decoding might not be aware of the host conversion process.

The base view in an MVC bitstream can be decoded independently of all other views, both for Stereo High Profile and Multiview High Profile. Moreover, if other views in the bitstream are coded without inter-view dependencies, it might be possible for the host to extract the compressed data of some other view and convert it into an ordinary H.264/AVC bitstream.

A.2 Single-View Display with H.264/AVC-based MVC

When a hardware accelerator is only capable of single-view display, but is capable of MVC DXVA decoding, the MVC bitstream may be decoded in a single-view mode as follows: The host software decoder sends the compressed data of one independently-

decodable view for the Stereo High Profile or Multiview High Profile for MVC DXVA decoding through MVC DXVA interfaces and data structures, following the DXVA_ModeH264_VLD_Stereo_High_Progressive profile, DXVA_ModeH264_VLD_Stereo_High profile or DXVA_ModeH264_VLD_Multiview_High profile. The compressed data may be the base view or some other view, as long as it is independently decodable.

When the host sends one independently decodable view through MVC DXVA interfaces, an MVC DXVA accelerator shall be able to decode the compressed data of an independently-encoded view, and shall output the decoded YUV data accordingly.

A.3 Single-View H.264/AVC Decoding with MVC DXVA

When a hardware accelerator is configured to support MVC DXVA, the hardware shall also be able to decode ordinary single-view (non-MVC) H.264/AVC bitstreams with corresponding constraints. Moreover, the accelerator shall be able to switch between MVC bitstream decoding and ordinary single-view H.264/AVC bitstream decoding (and vice-versa) without re-creating the DXVA decoder instance. This requirement ensures a smooth playback experience if the source content switches between MVC and single-view coded video sequences.

The following are specific examples of implications of this requirement.

- Accelerators supporting the DXVA_ModeH264_VLD_Stereo_High_Progressive profile shall be capable of decoding any non-MVC H.264/AVC bitstream that is compatible with High Profile decoders in which the **frame_mbs_only_flag** is equal to 1 in the active sequence parameter set.
- Accelerators supporting the DXVA_ModeH264_VLD_Stereo_High profile shall be able to decode any non-MVC H.264/AVC bitstream that is compatible with High Profile decoders.
- Accelerators supporting the DXVA_ModeH264_VLD_Multiview_High profile shall be able to decode any non-MVC H.264/AVC bitstream that is compatible with High Profile decoders in which the **frame_mbs_only_flag** is equal to 1 in the active sequence parameter set.

For More Information

- DXVA 1.0 specification: http://go.microsoft.com/fwlink/?LinkId=93647
- DXVA interfaces: http://go.microsoft.com/fwlink/?LinkId=210876
- DXVA configuration: http://go.microsoft.com/fwlink/?LinkId=210877
- Windows Driver Kit (WDK) documentation for DXVA: http://msdn.microsoft.com/en-us/library/ff55386.aspx
- DXVA 2.0: http://go.microsoft.com/fwlink/?LinkId=94771