SDK Developer Reference for Multi-view Video Coding

API Version 1.8

LEGAL DISCLAIMER

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or by visiting Intel's Web Site.

MPEG is an international standard for video compression/decompression promoted by ISO. Implementations of MPEG CODECs, or MPEG enabled platforms may require licenses from various entities, including Intel Corporation.

Intel and the Intel logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

*Other names and brands may be claimed as the property of others.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel.

Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804

Table of Contents

Overview

The SDK (Software Development Kit) is a software development library that exposes the media acceleration capabilities of Intel platforms for decoding, encoding and video processing. The API library covers a wide range of Intel platforms.

This document describes SDK extension to support Multi-view Video Coding (MVC).

Document Conventions

The SDK API uses the Verdana typeface for normal prose. With the exception of section headings and the table of contents, all code-related items appear in the `Courier New` typeface. Examples relevant to this document are `mfxStatus` and `MFXInit`. Hyperlinks appear in underlined boldface, such as **`mfxStatus`**.

Acronyms and Abbreviations

SDK	Intel® Media Server Studio – SDK
API	Application Programming Interface
MVC	Multi-view Video Coding
H.264	ITU*-T H.264, Advanced Video Coding

Related Documents

ITU*-T H.264	The ITU-T H.264 specification: "Advanced video coding for generic audiovisual services"

Architecture & Programming Guide

SDK extension for multiple view video coding requires the application to use an additional include file `mfxmvc.h`, in addition to the regular SDK include files. No additional library is needed at the link time.

```
Include these files:
        #include "mfxvideo.h"        /* SDK functions in C */
        #include "mfxvideo++.h"      /* Optional for C++ development */
        #include "mfxmvc.h"          /* Multiview Video Coding development */
Link to this library:
        libmfx.lib                   /* The SDK dispatcher library */
```

The SDK supports MVC as a natural extension of the H.264 codec. The application can identify MVC encoding and decoding by the codec identifier **`MFX_CODEC_AVC`**, and one of the profiles below:

`MFX_PROFILE_AVC_MULTIVIEW_HIGH`	Multi-view high profile
`MFX_PROFILE_AVC_STEREO_HIGH`	Stereo high profile

The SDK considers each view (or temporal representation) of a frame picture a separate processing unit. The SDK decoder outputs one view at a time. The video processor and the encoder process one view at a time. The SDK maintains state within the library so that the SDK decoding, video processing and encoding functions process/generate all views of the current picture in sequence, before process/generate the next picture.

Decoding Procedure

The SDK MVC decoder operates on complete MVC streams that contain all view/temporal configurations. The application can configure the SDK decoder to generate a subset at the decoding output. To do this, the application needs to understand the stream structure and based on such information configure the SDK decoder for target views.

The decoder initialization procedure is as follows:

(1) The application calls the **`MFXVideoDECODE_DecodeHeader`** function to obtain the stream structural information. This is actually done in two sub-steps:

a. The application calls the **MFXVideoDECODE_DecodeHeader** function with the **mfxExtMVCSeqDesc** structure attached to the **mfxVideoParam** structure. Do not allocate memory for the arrays in the **mfxExtMVCSeqDesc** structure just yet. Set the `View`, `ViewId` and `OP` pointers to NULL and set `NumViewAlloc`, `NumViewIdAlloc` and `NumOPAlloc` to zero. The function parses the bitstream and returns `MFX_ERR_NOT_ENOUGH_BUFFER` with the correct values `NumView`, `NumViewId` and `NumOP`. This step can be skipped if the application is able to obtain the `NumView`, `NumViewId` and `NumOP` values from other sources.
b. The application allocates memory for the `View`, `ViewId` and `OP` arrays and calls the **MFXVideoDECODE_DecodeHeader** function again. The function returns the MVC structural information in the allocated arrays.

(2) The application fills the **mfxExtMvcTargetViews** structure to choose the target views, based on information described in the **mfxExtMVCSeqDesc** structure.

(3) The application initializes the SDK decoder using the **MFXVideoDECODE_Init** function. The application must attach both the **mfxExtMVCSeqDesc** structure and the **mfxExtMvcTargetViews** structure to the **mfxVideoParam** structure.

In the above steps, do not modify the values of the **mfxExtMVCSeqDesc** structure after the **MFXVideoDECODE_DecodeHeader** function, as the SDK decoder uses the values in the structure for internal memory allocation.

Once the application configures the SDK decoder, the rest decoding procedure remains unchanged. As illustrated in Example 1, the application calls the **MFXVideoDECODE_DecodeFrameAsync** function multiple times to obtain all target views of the current frame picture, one target view at a time. The target view is identified by the `FrameID` field of the **mfxFrameInfo** structure. See the *SDK Developer Reference* for additional details of the decoding procedure.

```
/* get sequence description */
mfxExtBuffer *eb[2];
mfxExtMVCSeqDesc  seq_desc;
mfxVideoParam init_param;

init_param.ExtParam=&eb;
init_param.NumExtParam=1;
eb[0]=&seq_desc;
MFXVideoDECODE_DecodeHeader(session, bitstream, &init_param);

/* select views to decode */
mfxExtMvcTargetViews tv;
init_param.NumExtParam=2;
eb[1]=&tv;

/* initialize decoder */
MFXVideoDECODE_Init(session, &init_param);

/* perform decoding */
for (;;) {
    MFXVideoDECODE_DecodeFrameAsync(session, bits, work, &disp,
                                    &syncp);
    MFXVideoCORE_SyncOperation(session, &syncp, INFINITE);
}

/* close decoder */
MFXVideoDECODE_Close();
```

Example 1: Pseudo Code of the Decoding Procedure

Video Processing Procedure

The SDK video processing supports processing multiple views. For video processing initialization, the application needs to attach the **mfxExtMVCSeqDesc** structure to the **mfxVideoParam** structure and call the **MFXVideoVPP_Init** function. The function saves the view identifiers.

During video processing, the SDK processes each view independently, one view at a time. The SDK refers to the **FrameID** field of the **mfxFrameInfo** structure to configure each view according to its processing pipeline. The application needs to fill the the **FrameID** field before calling the **MFXVideoVPP_RunFrameVPPAsync** function, if the video processing source frame is not the output from the SDK MVC decoder.

Example 2 shows the video processing procedure pseudo code. See the *SDK Developer Reference* for additional details of the video processing procedure.

```
/* create sequence description */
mfxExtBuffer *eb;
mfxExtMVCSeqDesc  seq_desc;
mfxVideoParam init_param;

init_param.ExtParam = &eb;
init_param.NumExtParam=1;
eb=&seq_desc;

/* init VPP */
MFXVideoVPP_Init(session, &init_param);

/* perform processing */
for (;;) {
    MFXVideoVPP_RunFrameVPPAsync(session,in,out,aux,&syncp);
    MFXVideoCORE_SyncOperation(session,syncp,INFINITE);
}

/* close VPP */
MFXVideoVPP_Close(session);
```

Example 2: Pseudo Code of the Video Processing Procedure

Encoding Procedure

Similar to the decoding and video processing initialization procedures, the application attaches the **mfxExtMVCSeqDesc** structure to the **mfxVideoParam** structure for encoding initialization. The **mfxExtMVCSeqDesc** structure configures the SDK MVC encoder to work in three modes:

- **Default dependency mode:** The application specifies **NumView** and all other fields zero. The SDK encoder creates a single operation point with all views (view identifier 0…NumView-1) as target views. The first view (view identifier 0) is the base view. Other views depend on the base view.
- **Explicit dependency mode:** The application specifies **NumView** and the **View** dependency array, and sets all other fields to zero. The SDK encoder creates a single operation point with all views (view identifier **View[0…NumView-1].ViewId**) as target views. The first view (view identifier **View[0].ViewId**) is the base view. The view dependencies follow the **View** dependency structures.
- **Complete mode:** The application fully specifies the views and their dependencies. The SDK encoder generates a bitstream with corresponding stream structures.

The SDK MVC encoder does not support importing sequence and picture headers via the **mfxExtCodingOptionSPSPPS** structure, or configuring reference frame list via the **mfxExtRefListCtrl** structure.

During encoding, the SDK encoding function **MFXVideoENCODE_EncodeFrameAsync** accumulates input frames until encoding of a picture is possible. The function returns **MFX_ERR_MORE_DATA** for more data at input or **MFX_ERR_NONE** if having successfully accumulated enough data for encoding of a picture. The generated bitstream contains the complete picture (multiple views).

The application can change this behavior and instruct encoder to output each view in a separate bitstream buffer. To do so the application has to turn on the **ViewOutput** flag in the **mfxExtCodingOption** structure. In this case, encoder returns **MFX_ERR_MORE_BITSTREAM** if it needs more bitstream buffers at output and **MFX_ERR_NONE** when processing of picture (multiple views) has been finished. It is recommended that the application provides a new input frame each time the SDK encoder requests new bitstream buffer.

The application must submit views data for encoding in the order they are described in the mfxExtMVCSeqDesc structure. Particular view data can be submitted for encoding only when all views that it depends upon have already been submitted.

Example 3 shows the encoding procedure pseudo code. See the *SDK Developer Reference* for additional details of the encoding procedure.

```
/* create sequence description */
mfxExtBuffer *eb;
mfxExtMVCSeqDesc  seq_desc;
mfxVideoParam init_param;

init_param.ExtParam=&eb;
init_param.NumExtParam=1;
eb=&seq_desc;

/* init encoder */
MFXVideoENCODE_Init(session, &init_param);

/* perform encoding */
for (;;) {
    MFXVideoENCODE_EncodeFrameAsync(session, NULL, surface2, bits,
                                    &syncp);
    MFXVideoCORE_SyncOperation(session,syncp,INFINITE);
}

/* close encoder */
MFXVideoENCODE_Close();
```

Example 3: Pseudo Code of the Encoding Procedure

Structure Reference

mfxFrameId

Definition

```
typedef struct {
      mfxU16      TemporalID;
      mfxU16      PriorityID;
      union {
            mfxU16      reserved[2];
            mfxU16      ViewID;
      };
} mfxFrameId;
```

Description

The `mfxFrameId` describes the view and layer of a frame picture.

Members

`TemporalID`	The temporal identifier as defined in the annex H of the ITU*-T H.264 specification.
`PriorityID`	Reserved and must be zero.
`ViewID`	The view identifier as defined in the annex H of the ITU-T H.264 specification.

Change History

This structure is available since SDK API 1.3.

mfxFrameInfo

Definition

```
typedef struct {
      mfxU32      reserved[6];
      mfxFrameId  FrameID;
      mfxU32      FourCC;
    ...
} mfxFrameInfo;
```

Description

The `mfxFrameInfo` structure is extended to describe additionally the frame view information. Other fields remain unchanged. See the *SDK Developer Reference* for additional structure descriptions.

Members

`FrameID`	The **`mfxFrameId`** structure to describe the frame view information. `FrameID` is ignored when used in the **`mfxVideoParam`** structure.

Change History

This structure is available since SDK API 1.0. SDK 1.3 extended the structure to include the frame view description.

mfxMVCViewDependency

Definition

```
typedef struct  {
      mfxU16      ViewId;

      mfxU16      NumAnchorRefsL0;
      mfxU16      NumAnchorRefsL1;
      mfxU16      AnchorRefL0[16];
      mfxU16      AnchorRefL1[16];

      mfxU16      NumNonAnchorRefsL0;
      mfxU16      NumNonAnchorRefsL1;
      mfxU16      NonAnchorRefL0[16];
      mfxU16      NonAnchorRefL1[16];
} mfxMVCViewDependency;
```

Description

This `mfxMVCViewDependency` structure describes MVC view dependencies.

Members

`ViewId`	View identifier of this dependency structure
`NumAnchorRefsL0`	Number of view components for inter-view prediction in the initial reference picture list RefPicList0 for anchor view components

NumAnchorRefsL1	Number of view components for inter-view prediction in the initial reference picture list RefPicList1 for anchor view components
AnchorRefL0	View identifiers of the view components for inter-view prediction in the initial reference picture list RefPicList0 for anchor view components
AnchorRefL1	View identifiers of the view components for inter-view prediction in the initial reference picture list RefPicList1 for anchor view components
NumNonAnchorRefsL0	Number of view components for inter-view prediction in the initial reference picture list RefPicList0 for non-anchor view components
NumNonAnchorRefsL1	Number of view components for inter-view prediction in the initial reference picture list RefPicList1 for non-anchor view components
NonAnchorRefL0	View identifiers of the view components for inter-view prediction in the initial reference picture list RefPicList0 for non-anchor view components
NonAnchorRefL1	View identifiers of the view components for inter-view prediction in the initial reference picture list RefPicList0 for non-anchor view components

Change History

This structure is available since SDK API 1.3.

mfxMVCOperationPoint

Definition

```
typedef struct {
      mfxU16      TemporalId;
      mfxU16      LevelIdc;

      mfxU16      NumViews;
      mfxU16      NumTargetViews;
      mfxU16      *TargetViewId;
} mfxMVCOperationPoint;
```

Description

The mfxMVCOperationPoint structure describes the MVC operation point.

Members

`TemporalId`	Temporal identifier of the operation point
`LevelIdc`	Level value signaled for the operation point
`NumViews`	Number of views required for decoding the target output views corresponding to the operation point
`NumTargetViews`	Number of target output views for the operation point
`TargetViewId`	View identifiers of the target output views for operation point

Change History

This structure is available since SDK API 1.3.

mfxExtMVCSeqDesc

Definition

```
typedef struct  {
      mfxExtBuffer Header;

      mfxU32      NumView;
      mfxU32      NumViewAlloc;
      mfxMVCViewDependency *View;

      mfxU32      NumViewId;
      mfxU32      NumViewIdAlloc;
      mfxU16      *ViewId;

      mfxU32      NumOP;
      mfxU32      NumOPAlloc;
      mfxMVCOperationPoint *OP;

      mfxU16      NumRefsTotal;

      mfxU32      Reserved[16];
} mfxExtMVCSeqDesc;
```

Description

The `mfxExtMVCSeqDesc` structure describes the MVC stream information of view dependencies, view identifiers, and operation points. See the ITU*-T H.264 specification

chapter H.7.3.2.1.4 for details.

Members

`Header.BufferId`	Must be set to **MFX_EXTBUFF_MVC_SEQUENCE_DESCRIPTION**
`NumView`	Number of views
`NumViewAlloc`	The allocated view dependency array size
`View`	Pointer to a list of the **mfxMVCViewDependency** structure
`NumViewId`	Number of view identifiers
`NumViewIdAlloc`	The allocated view identifier array size
`ViewId`	Pointer to a list of view identifier
`NumOP`	Number of operation points
`NumOPAlloc`	The allocated operation point array size
`OP`	Pointer to a list of the **mfxMVCOperationPoint** structure
`NumRefsTotal`	Total number of reference frames in all views required to decode the stream. This value is returned from the **MFXVideoDECODE_Decodeheader** function. Do not modify this value.

Change History

This structure is available since SDK API 1.3.

mfxExtMvcTargetViews

Definition

```
typedef struct {
      mfxExtBuffer      Header;
      mfxU16            TemporalID;
      mfxU32            NumView;
      mfxU16            ViewID[1024];
} mfxExtMvcTargetViews;
```

Description

The mfxExtMvcTargetViews structure configures views for the decoding output.

Members

`Header.BufferId`	Must be **`MFX_EXTBUFF_MVC_TARGET_VIEWS`**
`TemoporalID`	The temporal identifier to be decoded
`NumView`	The number of views to be decoded
`ViewID`	List of view identifiers to be decoded

Change History

This structure is available since SDK API 1.3.

Enumerator Reference

CodecProfile

Description

The `CodecProfile` enumerator is extended to support MVC profiles. See the *SDK Developer Reference* for additional profile definitions.

Name/Description

`MFX_PROFILE_AVC_MULTIVIEW_HIGH`	MVC profiles
`MFX_PROFILE_AVC_STEREO_HIGH`	

Change History

This enumerator is available since SDK API 1.0. SDK API 1.3 added MVC profiles.

ExtendedBufferID

Description

The `ExtendedBufferID` enumerator is extended to add MVC support. See the *SDK Developer Reference* for additional definitions.

Name/Description

`MFX_EXTBUFF_MVC_SEQUENCE_DESCRIPTION`	This extended buffer describes stream structures. See the **`mfxExtMVCSeqDesc`** structure for details. The application can attach this buffer to the **`mfxVideoParam`** structure for encoding, decoding and video processing initialization.
`MFX_EXTBUFF_MVC_TARGET_VIEWS`	This extended buffer defines target views at the decoder output. See the **`mfxExtMVCTargetViews`** structure for details. The application can attach this buffer to the **`mfxVideoParam`** structure for decoding initialization.

Change History

This enumerator is available since SDK API 1.0. See additional change history in the structure definitions.