SDK Developer Reference for JPEG*/Motion JPEG

API Version 1.20

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 5

Document Conventions 5

Acronyms and Abbreviations 5

Architecture & Programming Guide 6

Decoding Procedure 6

Encoding Procedure 8

Structure Reference Extension 10

mfxInfoMFX 10

mfxExtJPEGQuantTables 12

mfxExtJPEGHuffmanTables 13

Enumerator Reference Extension 15

CodecFormatFourCC 15

CodecProfile 15

ChromaFormatIdc 15

Rotation 16

ExtendedBufferID 16

JPEG Color Format 16

JPEG Scan Type 17

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 the extension to the SDK for JPEG* processing.

Document Conventions

The SDK 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
DECODE	Video decoding
EXIF*	A image file format used by digital cameras
JFIF*	A image file format used by digital cameras
JPEG*	A picture compression algorithm
Motion JPEG	A motion picture compression algorithm utilizing JPEG
NV12	A YCbCr 4:2:0 color format for raw video frames
RGB4	A RGB color format for raw photo pictures, or RGB32

Architecture & Programming Guide

The SDK extension for JPEG*/motion JPEG requires the application to use an additional include file, `mfxjpeg.h`, in addition to the regular SDK include files. No additional library is required at link time.

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

The SDK extends the codec identifier **MFX_CODEC_JPEG** for JPEG and motion JPEG processing.

Decoding Procedure

The application can use the same decoding procedures for JPEG/motion JPEG decoding, as illustrated in Figure 1. See the *SDK Developer Reference* for the description of the decoding procedures.

```
// optional; retrieve initialization parameters
MFXVideoDECODE_DecodeHeader(…);
// decoder initialization
MFXVideoDECODE_Init(…);
// single frame/picture decoding
MFXVideoDECODE_DecodeFrameAsync(…);
MFXVideoCORE_SyncOperation(…);
// optional; retrieve meta-data
MFXVideoDECODE_GetUserData(…);
// close down
MFXVideoDECODE_Close(…);
```

Figure 1: Pseudo Code of the JPEG Decoding Procedure

DECODE supports JPEG baseline profile decoding as follows:

- DCT-based process
- Source image: 8-bit samples within each component
- Sequential
- Huffman coding: 2 AC and 2 DC tables
- 3 loadable quantization matrixes
- Interleaved and non-interleaved scans
- Single and multiple scans
- chroma subsampling ratios:
  - Chroma 4:0:0 (grey image)
  - Chroma 4:1:1
  - Chroma 4:2:0
  - Chroma horizontal 4:2:2
  - Chroma vertical 4:2:2
  - Chroma 4:4:4
- 3 channels images

The **MFXVideoDECODE_Query** function will return **MFX_ERR_UNSUPPORTED** if the input bitstream contains unsupported features.

For still picture JPEG decoding, the input can be any JPEG bitstreams that conform to the ITU-T* Recommendation T.81, with an EXIF* or JFIF* header. For motion JPEG decoding, the input can be any JPEG bitstreams that conform to the ITU-T Recommendation T.81.

Unlike other SDK decoders, JPEG one supports three different output color formats - NV12, YUY2 and RGB32. This support sometimes requires internal color conversion and more complicated initialization. The color format of input bitstream is described by **JPEGChromaFormat** and **JPEGColorFormat** fields in **mfxInfoMFX** structure. The **MFXVideoDECODE_DecodeHeader** function usually fills them in. But if JPEG bitstream does not contains color format information, application should provide it. Output color format is described by general SDK parameters - **FourCC** and **ChromaFormat** fields in **mfxFrameInfo** structure.

Motion JPEG supports interlaced content by compressing each field (a half-height frame) individually. This behavior is incompatible with the rest SDK transcoding pipeline, where SDK requires that fields be in odd and even lines of the same frame surface.) The decoding procedure is modified as follows:

(a) The application calls the **MFXVideoDECODE_DecodeHeader** function, with the first field JPEG bitstream, to retrieve initialization parameters.
(b) The application initializes the SDK JPEG decoder with the following settings:
   a. Set the **PicStruct** field of the **mfxVideoParam** structure with proper interlaced type, **MFX_PICSTRUCT_TFF** or **MFX_PICSTRUCT_BFF**, from motion JPEG header.
   b. Double the **Height** field of the **mfxVideoParam** structure as the value returned by the **MFXVideoDECODE_DecodeHeader** function describes only the first field. The actual frame surface should contain both fields.
(c) During decoding, application sends both fields for decoding together in the same **mfxBitstream**. Application also should set **DataFlag** in **mfxBitstream** structure to **MFX_BITSTREAM_COMPLETE_FRAME**. The SDK decodes both fields and combines them into odd and even lines as in the SDK convention.

SDK supports JPEG picture rotation, in multiple of 90 degrees, as part of the decoding operation. By default, the **MFXVideoDECODE_DecodeHeader** function returns the **Rotation**

parameter so that after rotation, the pixel at the first row and first column is at the top left. The application can overwrite the default rotation before calling **`MFXVideoDECODE_Init`**.

The application may specify Huffman and quantization tables during decoder initialization by attaching **`mfxExtJPEGQuantTables`** and **`mfxExtJPEGHuffmanTables`** buffers to **`mfxVideoParam`** structure. In this case, decoder ignores tables from bitstream and uses specified by application. The application can also retrieve these tables by attaching the same buffers to **`mfxVideoParam`** and calling **`MFXVideoDECODE_GetVideoParam`** or **`MFXVideoDECODE_DecodeHeader`** functions.

Encoding Procedure

The application can use the same encoding procedures for JPEG/motion JPEG encoding, as illustrated in Figure 12. See the *SDK Developer Reference* for the description of the encoding procedures.

```
// encoder initialization

MFXVideoENCODE_Init (…);

// single frame/picture encoding

MFXVideoENCODE_EncodeFrameAsync (…);

MFXVideoCORE_SyncOperation(…);

// close down

MFXVideoENCODE_Close(…);
```

Figure 2: Pseudo Code of the JPEG encoding Procedure

ENCODE supports JPEG baseline profile encoding as follows:

- DCT-based process
- Source image: 8-bit samples within each component
- Sequential
- Huffman coding: 2 AC and 2 DC tables
- 3 loadable quantization matrixes
- Interleaved and non-interleaved scans
- Single and multiple scans
- chroma subsampling ratios:
  - Chroma 4:0:0 (grey image)
  - Chroma 4:1:1
  - Chroma 4:2:0
  - Chroma horizontal 4:2:2
  - Chroma vertical 4:2:2
  - Chroma 4:4:4
- 3 channels images

The application may specify Huffman and quantization tables during encoder initialization by attaching **`mfxExtJPEGQuantTables`** and **`mfxExtJPEGHuffmanTables`** buffers to **`mfxVideoParam`**

structure. If the application does not define tables then the SDK encoder uses tables recommended in ITU-T* Recommendation T.81. If the application does not define quantization table it has to specify **Quality** parameter in **mfxInfoMFX** structure. In this case, the SDK encoder scales default quantization table according to specified **Quality** parameter.

The application should properly configured chroma sampling format and color format. **FourCC** and **ChromaFormat** fields in **mfxFrameInfo** structure are used for this. For example, to encode 4:2:2 vertically sampled YCbCr picture, the application should set **FourCC** to **MFX_FOURCC_YUY2** and **ChromaFormat** to **MFX_CHROMAFORMAT_YUV422V**. To encode 4:4:4 sampled RGB picture, the application should set **FourCC** to **MFX_FOURCC_RGB4** and **ChromaFormat** to **MFX_CHROMAFORMAT_YUV444**.

The SDK encoder supports different sets of chroma sampling and color formats on different platforms. The application has to call **MFXVideoENCODE_Query** function to check if required color format is supported on given platform and then initialize encoder with proper values of **FourCC** and **ChromaFormat** in **mfxFrameInfo** structure.

The application should not define number of scans and number of components. They are derived by the SDK encoder from **Interleaved** flag in **mfxInfoMFX** structure and from chroma type. If interleaved coding is specified then one scan is encoded that contains all image components. Otherwise, number of scans is equal to number of components. The SDK encoder uses next component IDs - "1" for luma (Y), "2" for chroma Cb (U) and "3" for chroma Cr (V).

The application should allocate big enough buffer to hold encoded picture. Roughly, its upper limit may be calculated using next equation:

```
BufferSizeInKB = 4 + (Width * Height * BytesPerPx + 1023) / 1024;
```

where `Width` and `Height` are weight and height of the picture in pixel, `BytesPerPx` is number of byte for one pixel. It equals to 1 for monochrome picture, 1.5 for NV12 and YV12 color formats, 2 for YUY2 color format, and 3 for RGB32 color format (alpha channel is not encoded).

Structure Reference Extension

mfxInfoMFX

Definition

```
typedef struct {
    mfxU32  reserved[7];

    mfxU16  reserved4;
    mfxU16  BRCParamMultiplier;

    mfxFrameInfo    FrameInfo;
    mfxU32  CodecId;
    mfxU16  CodecProfile;
    mfxU16  CodecLevel;
    mfxU16  NumThread;

    union {
        struct {    /* MPEG-2/H.264 Encoding Options */
        ...
        };
        struct {    /* H.264, MPEG-2 and VC-1 Decoding Options */
        ...
        };
        struct {    /* JPEG Decoding Options */
            mfxU16  JPEGChromaFormat;
            mfxU16  Rotation;
            mfxU16  JPEGColorFormat;
            mfxU16  InterleavedDec;
            mfxU8   SamplingFactorH[4];
            mfxU8   SamplingFactorV[4];
            mfxU16  reserved3[5];
        };
        struct {    /* JPEG Encoding Options */
            mfxU16  Interleaved;
            mfxU16  Quality;
            mfxU16  RestartInterval;
            mfxU16  reserved5[10];
        };
    };
} mfxInfoMFX;
```

Description

The `mfxInfoMFX` structure is extended to include JPEG* decoding options. Other fields remain unchanged. See the *SDK Developer Reference* for additional structure descriptions.

Members

`JPEGChromaFormat`	Specify the chroma sampling format that has been used to encode JPEG picture. See the **`ChromaFormat`** enumerator in *SDK Developer Reference* for details.
`Rotation`	Rotation option of the output JPEG picture; see the **`Rotation`** enumerator for details.
`JPEGColorFormat`	Specify the color format that has been used to encode JPEG picture. See the **`JPEG Color Format`** enumerator for details.
`InterleavedDec`	Specify JPEG scan type for decoder. See the **`JPEG Scan Type`** enumerator for details.
`Interleaved`	Non-interleaved or interleaved scans. If it is equal to **`MFX_SCANTYPE_INTERLEAVED`** then the image is encoded as interleaved, all components are encoded in one scan. See the **`JPEG Scan Type`** enumerator for details.
`Quality`	Specifies the image quality if the application does not specified quantization table. This is the value from 1 to 100 inclusive. "100" is the best quality.
`RestartInterval`	Specifies the number of MCU in the restart interval. "0" means no restart interval.
`SamplingFactorH` `SamplingFactorV`	Sampling factor.

Remarks

The application must specify the JPEG initialization parameters before rotation.

Change History

The JPEG decoding options are available since SDK API 1.3. Encoding options since SDK API 1.5.

The SDK API 1.6 added **`JPEGColorFormat`** field.

The SDK API 1.7 added **`InterleavedDec`** field.

The SDK API 1.19 added `SamplingFactorH` and `SamplingFactorV` fields.

mfxExtJPEGQuantTables

Definition

```
typedef struct {
    mfxExtBuffer    Header;

    mfxU16  reserved[7];
    mfxU16  NumTable;

    mfxU16    Qm[4][64];
} mfxExtJPEGQuantTables;
```

Description

The structure specifies quantization tables. The application may specify up to 4 quantization tables. The SDK encoder assigns ID to each table. That ID is equal to table index in **Qm** array. Table "0" is used for encoding of Y component, table "1" for U component and table "2" for V component. The application may specify fewer tables than number of components in the image. If two tables are specified, then table "1" is used for both U and V components. If only one table is specified then it is used for all components in the image. Table below illustrate this behavior.

table ID / number of tables	0	1	2
1	Y, U, V		
2	Y	U, V	
3	Y	U	V

Members

Header.BufferId	Must be **MFX_EXTBUFF_JPEG_QT**.
NumTable	Number of quantization tables defined in **Qm** array.
Qm	Quantization table values.

Change History

This structure is available since SDK API 1.5.

mfxExtJPEGHuffmanTables

Definition

```
typedef struct {
    mfxExtBuffer    Header;

    mfxU16  reserved[2];
    mfxU16  NumDCTable;
    mfxU16  NumACTable;

    struct {
        mfxU8   Bits[16];
        mfxU8   Values[12];
    } DCTables[4];

    struct {
        mfxU8   Bits[16];
        mfxU8   Values[162];
    } ACTables[4];
} mfxExtJPEGHuffmanTables;
```

Description

The structure specifies Huffman tables. The application may specify up to 2 quantization table pairs for baseline process. The SDK encoder assigns ID to each table. That ID is equal to table index in **DCTables** and **ACTables** arrays. Table "0" is used for encoding of Y component, table "1" for U and V component. The application may specify only one table in this case it will be used for all components in the image. Table below illustrate this behavior.

number of tables \ table ID	0	1
1	Y, U, V	
2	Y	U, V

Members

`Header.BufferId`	Must be **MFX_EXTBUFF_JPEG_HUFFMAN**.
`NumDCTable`	Number of DC quantization table in **DCTables** array.

`NumACTable`	Number of AC quantization table in **`ACTables`** array.
`Bits`	Number of codes for each code length.
`Values`	List of the 8-bit symbol values.

Change History

This structure is available since SDK API 1.5.

Enumerator Reference Extension

CodecFormatFourCC

Description

Additional `CodecFormatFourCC` enumerator itemizes the JPEG* codec. See the *SDK Developer Reference* for additional enumerator definitions.

Name/Description

`MFX_CODEC_JPEG`	JPEG codec

CodecProfile

Description

Additional `CodecProfile` enumerator itemizes the supported JPEG profile. See the *SDK Developer Reference* for additional enumerator definitions.

Name/Description

`MFX_PROFILE_JPEG_BASELINE`	JPEG baseline profile

ChromaFormatIdc

Description

Additional `ChromaFormatIdc` enumerator itemizes the JPEG* color-sampling formats. See the *SDK Developer Reference* for additional enumerator definitions.

Name/Description

`MFX_CHROMAFORMAT_JPEG_SAMPLING`	Color sampling specified via **`mfxInfoMFX`**::`SamplingFactorH` and `SamplingFactorV`

Available since SDK API 1.19.

Rotation

Description

The `Rotation` enumerator itemizes the JPEG rotation options.

Name/Description

`MFX_ROTATION_0`	No rotation
`MFX_ROTATION_90`	90 degree rotation
`MFX_ROTATION_180`	180 degree rotation
`MFX_ROTATION_270`	270 degree rotation

ExtendedBufferID

Description

Additional **`ExtendedBufferID`** were added for JPEG support. See the *SDK Developer Reference* for additional enumerator definitions.

Name/Description

Encoding Configuration

`MFX_EXTBUFF_JPEG_QT`	This extended buffer defines quantization tables for JPEG encoder.
`MFX_EXTBUFF_JPEG_HUFFMAN`	This extended buffer defines Huffman tables for JPEG encoder.

JPEG Color Format

Description

This enumerator itemizes the JPEG color format options.

Name/Description

`MFX_JPEG_COLORFORMAT_UNKNOWN`	Unknown color format. The SDK decoder tries to determine color format from available in bitstream information. If such information is not present, then **`MFX_JPEG_COLORFORMAT_YCbCr`** color format is assumed.

MFX_JPEG_COLORFORMAT_YCbCr	Bitstream contains Y, Cb and Cr components.
MFX_JPEG_COLORFORMAT_RGB	Bitstream contains R, G and B components.

This enumerator is available since SDK API 1.6.

JPEG Scan Type

Description

This enumerator itemizes the JPEG scan types.

Name/Description

MFX_SCANTYPE_UNKNOWN	Unknown scan type.
MFX_SCANTYPE_INTERLEAVED	Interleaved scan.
MFX_SCANTYPE_NONINTERLEAVED	Non-interleaved scan.

This enumerator is available since SDK API 1.7.