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

SDK Developer Reference for JPEG/Motion JPEG ................................................. 1
Table of Contents ................................................................................................. 4
Overview .............................................................................................................. 5
  Document Conventions ...................................................................................... 5
  Acronyms and Abbreviations ............................................................................. 5
Architecture & Programming Guide ...................................................................... 5
  Decoding Procedure .......................................................................................... 5
    Figure 1: Pseudo Code of the JPEG Decoding Procedure ............................... 5
  Encoding Procedure .......................................................................................... 6
    Figure 2: Pseudo Code of the JPEG encoding Procedure ............................... 6
Structure Reference Extension ............................................................................. 7
  mfxInfoMFX ...................................................................................................... 7
  mfxExtJPEGQuantTables ................................................................................. 8
  mfxExtJPEGHuffmanTables .............................................................................. 8
Enumerator Reference Extension .......................................................................... 8
  CodecFormatFourCC ....................................................................................... 9
  CodecProfile .................................................................................................... 9
  ChromaFormatIdc ........................................................................................... 9
  Rotation ............................................................................................................ 9
  ExtendedBufferID ............................................................................................ 9
  JPEG Color Format ......................................................................................... 9
  JPEG Scan Type ............................................................................................... 9
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.

Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SDK</td>
<td>Intel® Media Server Studio – SDK</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>DECODE</td>
<td>Video decoding</td>
</tr>
<tr>
<td>EXIF*</td>
<td>An image file format used by digital cameras</td>
</tr>
<tr>
<td>JFIF*</td>
<td>An image file format used by digital cameras</td>
</tr>
<tr>
<td>JPEG*</td>
<td>A picture compression algorithm</td>
</tr>
<tr>
<td>Motion JPEG</td>
<td>A motion picture compression algorithm utilizing JPEG</td>
</tr>
<tr>
<td>NV12</td>
<td>A YCbCr 4:2:0 color format for raw video frames</td>
</tr>
<tr>
<td>RGB4</td>
<td>A RGB color format for raw photo pictures, or RGB32</td>
</tr>
</tbody>
</table>

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:

```c
#include "mfxvideo.h"  /* SDK functions in C */
#include "mfxvideo++.h"  /* Optional for C++ development */
#include "mfxjpeg.h"  /* JPEG development */
```

Link this library:

```c
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.

Figure 1: Pseudo Code of the JPEG Decoding Procedure

```c
// 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(...);
```

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 matrices
- 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 JPBBColorFormat fields in mfxInfoMFX structure. The MFXVideoDECODE_DecodeHeader function usually fills
The application should allocate big enough buffer to hold encoded picture. Roughly, its upper limit may be calculated using next equation:

\[
2 \text{ for chroma Cb (U)} \text{ and } 3 \text{ for chroma Cr (V).}
\]

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).

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 mfxExtJPEGHuffmanTables and mfxExtJPEGQuantTables 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.

**Figure 2: Pseudo Code of the JPEG encoding Procedure**

```c
// encoder initialization
MFXVideoENCODE_Init (...);
// single frame/picture encoding
MFXVideoENCODE_EncodeFrameAsync (...);
MFXVideoCORE_SyncOperation (...);
// close down
MFXVideoENCODE_Close (...);
```

**ENCODER** supports JPEG baseline profile encoding as follows:

- **DCT-based process**
- **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 mfxExtJPEGHuffmanTables and mfxExtJPEGQuantTables 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 BRCPParamMultiplier;
    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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JPEGChromaFormat</td>
<td>Specify the chroma sampling format that has been used to encode JPEG picture. See the ChromaFormat enumerator in SDK Developer Reference for details.</td>
</tr>
<tr>
<td>Rotation</td>
<td>Rotation option of the output JPEG picture; see the Rotation enumerator for details.</td>
</tr>
<tr>
<td>JPEGColorFormat</td>
<td>Specify the color format that has been used to encode JPEG picture. See the JPEG Color Format enumerator for details.</td>
</tr>
<tr>
<td>InterleavedDec</td>
<td>Specify JPEG scan type for decoder. See the JPEG Scan Type enumerator for details.</td>
</tr>
<tr>
<td>Interleaved</td>
<td>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.</td>
</tr>
<tr>
<td>Quality</td>
<td>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.</td>
</tr>
<tr>
<td>RestartInterval</td>
<td>Specifies the number of MCU in the restart interval. “0” means no restart interval.</td>
</tr>
<tr>
<td>SamplingFactorH</td>
<td>Sampling factor.</td>
</tr>
<tr>
<td>SamplingFactorV</td>
<td></td>
</tr>
</tbody>
</table>

**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**

```c
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>
<thead>
<tr>
<th>table ID</th>
<th>number of tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Y, U, V</th>
<th>Y, U, V</th>
<th>Y, U, V</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Y, U, V</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Y, U, V</td>
<td>Y, U, V</td>
<td></td>
</tr>
</tbody>
</table>

**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**

```c
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 DCTable 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.

<table>
<thead>
<tr>
<th>table ID</th>
<th>number of tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Y, U, V</th>
<th>Y, U, V</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Y, U, V</td>
<td></td>
</tr>
</tbody>
</table>

**Members**

- **Header.BufferId** Must be MFX_EXTBUFF_JPEG_HUFFMAN
- **NumDCTable** Number of DC quantization table in DCTable 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**
- **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_SCAN_TYPE_NONINTERLEAVED Non-interleaved scan.

This enumerator is available since SDK API 1.7.