SDK API
Reference Manual
for Premium Telecine Interlace Reverser

API Version 1.15
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.

Copyright © 2014-2015, Intel Corporation. All Rights reserved.
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 ............................................................................................................................................... 1  
Document Conventions ......................................................................................................................... 1  
Acronyms and Abbreviations .................................................................................................................. 1  
Architecture ............................................................................................................................................ 2  
Deinterlacing/Inverse Telecine Processing ............................................................................................... 2  
Programming Guide ............................................................................................................................... 4  
  Video Processing Procedures using VPP plug-in ...................................................................................... 4  
      Configuration....................................................................................................................................... 6  
  Transcoding Procedures ......................................................................................................................... 9  
Function Reference ................................................................................................................................. 11  
  MFXVideoVPP ....................................................................................................................................... 11  
      MFXVideoVPP_RunFrameVPPAsyncEx ................................................................................................. 11
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 Premium Telecine Interlace Reverser.

Document Conventions

The Intel Media Server Studio – PTIR 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 (mxfStatus and MFXInit). All class-related items appear in all cap boldface, such as decode and encode. Member functions appear in initial cap boldface, such as Init and Reset, and these refer to members of all three classes, decode, encode and VPP. Hyperlinks appear in underlined boldface, such as MFXVideoVPP_RunFrameVPPAsyncEx.

Acronyms and Abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>SDK</td>
<td>Intel® Media Server Studio – SDK and Intel® Integrated Native Developer Environment Media SDK for Windows</td>
</tr>
<tr>
<td>video memory</td>
<td>memory used by hardware acceleration device, also known as GPU, to hold frame and other types of video data</td>
</tr>
<tr>
<td>VPP</td>
<td>Video Processing</td>
</tr>
<tr>
<td>PTIR</td>
<td>Premium Telecine Interlace Reverser</td>
</tr>
<tr>
<td>NV12</td>
<td>A color format for raw video frames</td>
</tr>
</tbody>
</table>
Architecture

SDK functions fall into the following categories:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VPP</td>
<td>Perform video processing on raw video frames</td>
</tr>
<tr>
<td>CORE</td>
<td>Auxiliary functions for synchronization</td>
</tr>
<tr>
<td>Misc</td>
<td>Global auxiliary functions</td>
</tr>
</tbody>
</table>

Deinterlacing/Inverse Telecine Processing

The PTIR takes raw interlaced or telecined frames in NV12 format as input and provides raw NV12 progressive frames as output.

The actual conversion process is a chain operation with many single-function filters. Figure 1 illustrates deinterlacing input frame sequence into progressive frame sequence. The application specifies the input and output picture structure and frame rate, and the PTIR configures the pipeline accordingly. Unless specifically instructed, the PTIR builds the pipeline in a way that best utilizes hardware acceleration or generates the best video processing quality.

Table 1 shows the PTIR supported features. The application can configure supported video processing feature through the video processing I/O parameters. The application can also
configure optional features through hints. See “Video Processing procedure / Configuration” for more details on how to configure optional filters.

Table 1: Video Processing Features

<table>
<thead>
<tr>
<th>Video Processing Features</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic Telecine and Interlace Reversal of Mixed Content automatically selecting Telecine, or Interlace Reversal (double frame rate)</td>
<td>I/O parameters and/or hint</td>
</tr>
<tr>
<td>Automatic Telecine and Interlace Reversal of Mixed Content automatically selecting Telecine, or Interlace Reversal (normal frame rate).</td>
<td>I/O parameters and/or hint</td>
</tr>
<tr>
<td>Interlace Reversal (Deinterlacing) at double frame rate</td>
<td>I/O parameters and/or hint</td>
</tr>
<tr>
<td>Interlace Reversal (Deinterlacing) at normal frame rate</td>
<td>I/O parameters and/or hint</td>
</tr>
<tr>
<td>Automatic Telecine pattern identification and Telecine Reversal</td>
<td>I/O parameters and/or hint</td>
</tr>
<tr>
<td>Selected Telecine pattern Reversal (for a specified Telecine pattern)</td>
<td>hint</td>
</tr>
<tr>
<td>Automatic Telecine and Interlace Reversal of Mixed Content automatically selecting Telecine, or Interlace Reversal with a constant 30 FPS output and input frame sequence interlace detection.</td>
<td>hint</td>
</tr>
<tr>
<td>Input frame sequence interlace detection with sequence pass through from input to output</td>
<td>hint</td>
</tr>
</tbody>
</table>
Programming Guide

This chapter describes the concepts used in programming the SDK and the Premium Telecine Interlace Reverser.

The application must use the include file, `mfxvideo.h` (for C programming), or `mfxvideo++.h` (for C++ programming), and link the SDK static dispatcher library, `libmfx.lib`.

Include these files:
```c
#include “mfxvideo.h” /* The SDK include file */
#include “mfxvideo++.h” /* Optional for C++ development */
#include “mfxplugin.h” /* Plugin development */
```

Link this library:
```c
libmfx.lib /* The SDK static dispatcher library */
```

Video Processing Procedures using VPP plug-in

Example 1 shows the pseudo code of the video processing procedure using plugin. The following describes a few key points:

- The application uses the `MFXVideoVPP_QueryIOSurf` function to obtain the number of frame surfaces needed for input and output. The application must allocate two frame surface pools, one for the input and the other for the output.

- The video processing function `MFXVideoVPP_RunFrameVPPAsyncEx` is asynchronous. The application must synchronize to make the output result ready, through the `MFXVideoCORE_SyncOperation` function.

- The body of the video processing procedures covers three scenarios as follows:
  - If sufficient number of frames is passed to VPP, VPP returns `MFX_ERR_NONE` when an output is ready. The application must process the output frame after synchronization, as the `MFXVideoVPP_RunFrameVPPAsyncEx` function is asynchronous. At the end of a sequence, the application must provide a NULL input to drain any remaining frames.
  - SDK legacy VPP API function `MFXVideoVPP_RunFrameVPPAsync` function is not supported by PTIR plug-in. `MFXVideoVPP_RunFrameVPPAsyncEx` must be used.
  - If the number of frames consumed at input is more than the number of provided work frames, VPP returns `MFX_ERR_MORE_DATA` for additional input until an output is ready. Application must update input frame after receiving this status, input frame is consumed by frame processor only and if only previous call of this function returns this status. When sufficient resources were provided, VPP returns `MFX_ERR_NONE` and updates pointer to the output frame. The application must synchronize to make the output result ready, through the `MFXVideoCORE_SyncOperation` function. The application must
process the output frame after synchronization and provide a NULL input at the end of sequence to drain any remaining frames.

- If the number of frames consumed at input is less than the number of required frames for output, VPP returns MFX_ERR_MORE_SURFACE. The application must call this function again with new free surfaces from out surface pool.

```c
mfxSession session;
MFXInit(MFX_IMPL_HARDWARE,1.11,&session);
/* Initialize a handle to a system’s hardware acceleration device */
MFXVideoCORE_SetHandle(session, type, hdl);
MFXVideoUSER_Load(session, MFX_PLUGINID_ITELECINE_HW, version);
mfxVideoParam *in, *out; // input and output parameters structure
/* allocate structures and fill input parameters structure, zero unused fields */
MFXVideoVPP_Query(session, in, out);
/* check supported parameters */
MFXVideoVPP_QueryIOSurf(session, &init_param, response);
allocate_pool_of_surfaces(in_pool, response[0].NumFrameSuggested);
allocate_pool_of_surfaces(work_pool, response[1].NumFrameSuggested);
mfxVideoParam init_param;
/* configure init_param structure within supported parameters according to response from Query */
MFXVideoVPP_Init(session, &init_param);
in=find_unlocked_surface_and_fill_content(in_pool);
work=find_unlocked_surface_from_the_pool(work_pool);
bool update_in = false;
for (;;) {
    if (update_in) {
        update_in = false;
        in=find_unlocked_surface(in_pool);
        fill_content_for_video_processing(in);
        if (end_of_input_sequence()) in=NULL;
        continue;
    }
    sts=MFXVideoVPP_RunFrameVPPAsyncEx(session,in,work,&out,&syncp);
    if (sts==MFX_ERR_MORE_DATA && in==NULL) break;
    if (sts==MFX_ERR_MORE_DATA) {
        if (end_of_input_sequence()) {
            in=NULL;
        } else {
            update_in = true;
        }
    } else {
        process_output_frame(out);
    }
}
MFXVideoVPP_Close(session);
free_pool_of_surfaces(in_pool);
free_pool_of_surfaces(work_pool);
MFXVideoUSER_UnLoad(session, MFX_PLUGINID_ITELECINE_HW)
MFXClose(session);
```

Example 1: Video Processing Pseudo Code
Configuration

The PTIR plug-in configures the video processing pipeline operation based on parameters specified in the mfxVideoParam structure. Example 2 shows pseudo code with configuration parameters for deinterlacing without frame rate change. Table 2 represents supported PTIR modes of operation. Table 3 represents mfxVideoParam structure parameters configuration for different modes.

```c
/* configure the mfxVideoParam structure */
mfxVideoParam conf;
memset(&conf,0,sizeof(conf));
conf.IOPattern=MFX_IOPATTERN_IN_SYSTEM_MEMORY|
              MFX_IOPATTERN_OUT_SYSTEM_MEMORY;


conf.vpp.In.FrameRateExtN==conf.vpp.Out.FrameRateExtN=30;
conf.vpp.In.FrameRateExtD==conf.vpp.Out.FrameRateExtD=1;

conf.vpp.In.PicStruct==MFX_PICSTRUCT_FIELD_TFF;
conf.vpp.Out.PicStruct==MFX_PICSTRUCT_PROGRESSIVE;

/* video processing initialization */
MFXVideoVPP_Init(session, &conf);
```

### Example 2: Configure Video Processing

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Deinterlacing mode enumerator</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Automatic telecine and interlace reversal of mixed content automatically selecting telecine, or interlace reversal (double frame rate)</td>
<td>MFX_DEINTERLACING_AUTO_DOUBLE</td>
</tr>
<tr>
<td>1</td>
<td>Automatic telecine and interlace reversal of mixed content automatically selecting telecine, or interlace reversal (normal frame rate).</td>
<td>MFX_DEINTERLACING_AUTO_SINGLE</td>
</tr>
<tr>
<td>2</td>
<td>Interlace reversal (deinterlacing) at double frame rate</td>
<td>MFX_DEINTERLACING_FULL_FR_OUT</td>
</tr>
<tr>
<td>3</td>
<td>Interlace reversal (deinterlacing) at normal frame rate</td>
<td>MFX_DEINTERLACING_HALF_FR_OUT</td>
</tr>
<tr>
<td>4</td>
<td>Automatic telecine pattern identification and telecine reversal</td>
<td>MFX_DEINTERLACING_24FPS_OUT</td>
</tr>
</tbody>
</table>
Table 3: PTIR modes configuration by mfxVideoParam structure fields

<table>
<thead>
<tr>
<th>Mode id</th>
<th>Input picture structure</th>
<th>Output picture structure</th>
<th>Input frame rate</th>
<th>Output frame rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unknown</td>
<td>Progressive</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>Unknown</td>
<td>Progressive</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>2</td>
<td>Interlaced TFF or BFF</td>
<td>Progressive</td>
<td>30</td>
<td>60</td>
</tr>
<tr>
<td>3</td>
<td>Interlaced TFF or BFF</td>
<td>Progressive</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>Interlaced TFF or BFF</td>
<td>Progressive</td>
<td>29.97 or 30</td>
<td>23.976 or 24</td>
</tr>
</tbody>
</table>

Note, that in this table input rate is frame not field rate like in SDK legacy VPP. For example, 30i to 60p conversion, it should be described in mfxVideoParam as input frame rate equal to 30 and output 60.

Note, that mode 5 (Selected telecine pattern reversal) can be configured only with an extended buffer `mfxExtVPPDeinterlacing` provided. Attach this structure as part of the `mfxVideoParam` extended buffers to configure the SDK VPP during `MFXVideoVPP_Init`.

Table 4 represents supported telecine patterns. Fields order column represents the example of field order of original progressive sequence, in this example; all patterns have a top field first structure. Telecine location represents a position inside a sequence of 5 frames where the telecine artifacts start (for a given field order pattern):
Table 4: Supported telecine patterns reversal

<table>
<thead>
<tr>
<th>Telecine Pattern enumerator</th>
<th>Fields order example</th>
<th>Telecine location for a given example</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFX_TELECINE_PATTERN_32</td>
<td>T0B0 T1B1 T1B2 T2B3 T3B3</td>
<td>2</td>
</tr>
<tr>
<td>MFX_TELECINE_PATTERN_2332</td>
<td>T0B0 T1B1 T2B1 T2B2 T3B3</td>
<td>2</td>
</tr>
<tr>
<td>MFX_TELECINE_PATTERN_FRAME_REPEAT</td>
<td>T0B0 T0B0 T1B1 T2B2 T3B3</td>
<td>1</td>
</tr>
<tr>
<td>MFX_TELECINE_PATTERN_41</td>
<td>T0B0 T0B1 T1B2 T2B3 T3B0</td>
<td>1</td>
</tr>
</tbody>
</table>

Example 3 shows pseudo code with configuration parameters for reverse telecine operation for a fixed telecine pattern with cadence 3:2.
Example 3: Configure Video Processing

Note that there are additional limitations with comparison to SDK legacy VPP described by following key points:

- PTIR supports only AsyncDepth = 1;
- Input frame width and height must match output frame width and height;
- Hardware acceleration device handle must be provided prior PTIR initialization.

Transcoding Procedures

The application can use other SDK components and PTIR plugin together for transcoding operations. For example, video processing functions to resize and render to a display with

```c
/* configure the mfxVideoParam structure */
mfxVideoParam conf;
memset(&conf,0,sizeof(conf));

mfxExtVPPDeinterlacing extdeint;
memset(&extdeint,0,sizeof(extdeint));

//attach extended buffer to the mfxVideoParam structure
mfxExtBuffer* extbuf = (mfxExtBuffer*) &extdeint;
conf.ExtParam = &extbuf;

conf.IOPattern=MFX_IOPATTERN_IN_SYSTEM_MEMORY| MFX_IOPATTERN_OUT_SYSTEM_MEMORY;


conf.vpp.In.FrameRateExtN=30000;
conf.vpp.In.FrameRateExtD=1001;
conf.vpp.Out.FrameRateExtN=24000;
conf.vpp.Out.FrameRateExtD=1001;

conf.vpp.In.PicStruct=MFX_PICSTRUCT_FIELD_TFF;
conf.vpp.Out.PicStruct=MFX_PICSTRUCT_PROGRESSIVE;

extdeint.Mode = MFX_DEINTERLACING_FIXED_TELECINE_PATTERN;
extdeint.TelecinePattern = MFX_TELECINE_PATTERN_32;
extdeint.TelecineLocation = 2; //valid values from 0 to 4

/* video processing initialization */
MFXVideoVPP_Init(session, &conf);
```
resolution less than initial content, and then encode to H.264 or H.265. For more details on building transcoding pipelines please see SDK API Reference Manual.

**Important note**: to use SDK legacy VPP the application must create a separate SDK session and use `MFXJoinSession` method, see details in SDK API Reference Manual.
One new SDK VPP API function was introduced with Premium Telecine Interlace Reverser, please check SDK API Reference Manual and SDK API Reference Manual Extensions for User-Defined Functions for the full list of SDK functions description.

### MFXVideoVPP

This class of functions performs video processing before encoding.

### MFXVideoVPP_RunFrameVPPAsyncEx

#### Syntax

```c
mfxStatus MFXVideoVPP_RunFrameVPPAsyncEx(mfxSession session,
                                        mfxFrameSurface1 *surface_in, mfxFrameSurface1 *surface_work,
                                        mfxFrameSurface1 **surface_out, mfxSyncPoint *syncp);
```

#### Parameters

- `session`: SDK session handle
- `surface_in`: Pointer to the input video surface structure
- `surface_work`: Pointer to the work video surface structure
- `surface_out`: Pointer to the output frame in the input order
- `syncp`: Pointer to the output sync point

#### Description

This function processes a single input frame to a single output frame. Input frame is expected **only and if only** previous call of this function had returned **MFX_ERR_MORE_DATA**.

The video processing process may not generate an instant output given an input. See section Video Processing Procedures for details on how to correctly send input and retrieve output.

At the end of the stream, call this function with the input argument `in=NULL` to retrieve any remaining frames, until the function returns **MFX_ERR_MORE_DATA**.

This function is asynchronous.

#### Return Status

- **MFX_ERR_NONE**: The output frame is ready after synchronization.
MFX_ERR_MORE_DATA

Need more input frames before VPP can produce an output.

MFX_ERR_MORE_SURFACE

Need more surfaces at output for additional output frames available.

MFX_WRN_DEVICE_BUSY

Hardware device is currently busy. Call this function again in a few milliseconds.

Change History

This function is available since SDK API 1.11.
Structure Reference

mfxExtVPPDeinterlacing

Definition

typedef struct {
    mfxExtBuffer    Header;
    mfxU16  Mode;
    mfxU16  TelecinePattern;
    mfxU16  TelecineLocation;
    mfxU16  reserved[9];
} mfxExtVPPDeinterlacing;

Description

The mfxExtVPPDeinterlacing structure is used by the application to specify different deinterlacing algorithms.

Members

Header.BufferId Must be MFX_EXTBUFF_VPP_DEINTERLACING

Mode Deinterlacing algorithm. See the DeinterlacingMode enumerator for details.

TelecinePattern Specifies telecine pattern when Mode = MFX_DEINTERLACING_FIXED_TELECINE_PATTERN. See the TelecinePattern enumerator for details.

TelecineLocation Specifies position inside a sequence of 5 frames where the artifacts start when TelecinePattern = MFX_TELECINE_POSITION_PROVIDED.

Change History

This structure is available since SDK API 1.8.

The SDK API 1.13 adds TelecinePattern and TelecineLocation fields.
## Enumerator Reference

### DeinterlacingMode

**Description**

The `DeinterlacingMode` enumerator itemizes VPP deinterlacing modes.

**Name/Description**

<table>
<thead>
<tr>
<th>Name/Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFX_DEINTERLACING_BOB</td>
<td>BOB deinterlacing mode.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_ADVANCED</td>
<td>Advanced deinterlacing mode.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_AUTO_DOUBLE</td>
<td>Auto mode with deinterlacing double framerate output.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_AUTO_SINGLE</td>
<td>Auto mode with deinterlacing single framerate output.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_FULL_FR_OUT</td>
<td>Deinterlace only mode with full framerate output.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_HALF_FR_OUT</td>
<td>Deinterlace only Mode with half framerate output.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_24FPS_OUT</td>
<td>24 fps fixed output mode.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_FIXED_TELECINE_PATTERN</td>
<td>Fixed telecine pattern removal mode.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_30FPS_OUT</td>
<td>30 fps fixed output mode.</td>
</tr>
<tr>
<td>MFX_DEINTERLACING_DETECT_INTERLACE</td>
<td>Only interlace detection.</td>
</tr>
</tbody>
</table>

**Change History**

This enumerator is available since SDK API 1.13.

### TelecinePattern

**Description**

The `TelecinePattern` enumerator itemizes telecine patterns.

**Name/Description**
MFX_TELECINE_PATTERN_32
3:2 telecine

MFX_TELECINE_PATTERN_2332
2:3:3:2 telecine

MFX_TELECINE_PATTERN_FRAME_REPEAT
One frame repeat telecine

MFX_TELECINE_PATTERN_41
4:1 telecine

MFX_TELECINE_POSITION_PROVIDED
User must provide position inside a sequence of 5 frames where the artifacts start.

**Change History**

This enumerator is available since SDK API 1.13.