Media Samples Guide

Overview

Samples work with **Intel® Media Server Studio 2018 - SDK for Linux* Server** (hereinafter referred to as "**SDK**").

They demonstrate how to incorporate the **SDK** API into various applications.

Some samples are able to work with **HEVC Decoder & Encoder** (hereinafter referred to as "**HEVC Encoder**", "**HEVC Decoder**", "**HEVC**").

Not all of the samples listed below might be applicable and supported for a particular product. Make sure to check the respective release notes document for potential limitations.

What's New

- Due to migration of SDK to new libva infrastructure, samples are now bound to libva.so.2 library and can be used with the latest version of SDK only. To use samples with previous version of SDK, sample_common/src/vaapi_utils.cpp file should be modified: all references to libva.so.2, libva-drm.so.2, libva-x11.so.2 should be replaced with corresponding file names ending with ".so.1"
- sample_plugin_opencl is modified to be compatible with OpenCL NEO
- Sample_multi_transcode is extended with VPP output memory type selection option
- Sample_multi_transcode is extended with option enabling external MBQP map
- Major fixed issues:
  - HEVC interlaced support in sample_encode
  - Sample_multi_transcode writes incorrect last field in case of interlaced HEVCe and -n option
  - Incorrectly calculated bitstream buffer size in sample_multi_transcode (in particular cases)
  - sample_multi_transcode doesn't stop writing out bitstream after some time
  - sample_vpp advanced FRC conversion gains only 1 frame as result
  - sample_multi_transcode sometimes crashes while using cpu rotation plugin
- New samples added: sample_hevc_fei, sample_hevc_fei_abr, metrics monitor

Package contents

Full list of available samples:

- **Video Decoding Sample**

  Console application which performs decoding of elementary compressed video stream to raw frames. Includes the following features:

  - decoding of HEVC (High Efficiency Video Coding) video via **HEVC Decoder**
  - decoding with video post processing (color conversion) of raw video sequences
- **Video Encoding Sample**

  Console application which performs encoding of raw video frames into elementary compressed stream. Includes the following features:

---

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

  - video resizing
  - video rotation via User Plug-in Sample
  - video rotation via User Plug-in Sample using OpenCL™
  - encoding HEVC video via **HEVC Encoder**
- **Video Processing Sample**

  Console application which performs various video processing algorithms on raw frames.
- **Video Transcoding Sample**

  Console application which performs transcoding of elementary video stream from one compressed format to another. Includes the following features:

  - multiple video streams transcoding
  - video resizing, de-interlacing
  - video rotation via User Plug-in Sample
  - video rotation via User Plug-in Sample using OpenCL
  - video processing using VPP algorithms
- **OpenCL Video Motion Estimation Sample and OpenCL Advanced Video Motion Estimation Samples**

  Console application which provides step-by-step guidelines on the using Intel's motion estimation extension for OpenCL standard. The motion estimation extension includes a set of host-callable functions for frame-based Video Motion Estimation.
- **FEI Encoding Sample**

  Console application that uses SDK FEI API (Flexible Encoder Interface) and demonstrates capability to stream internal encoder information during encoding process to specified output. Intel® Media Server Studio 2018 - SDK for Linux* Server is required for this sample.

  Supported interfaces are:

  - PreENC
  - FEI ENCODE
  - ENC
  - PAK
  - DecodeStreamOut
- **sample_hevc_fei**

  Sample that showcases usage of HEVC FEI for regular encodings and transcodings. Includes basic constructions for HEVC FEI enabling.
- **sample_hevc_fei_abr**

  Sample that emonstrates how to construct Look Ahead BRC for transcoding scenarios using HEVC FEI. Includes statistics stream out from encoded stream which influence low-level HEVC FEI encoder decision for quality tuning. Also supports 1:N transcoding.
- **Metrics Monitor**

  Metrics Monitor is a user space shared library and sample for Linux that provides applications access to a number of metrics from the GPU kernel mode driver to aid in understanding the state of the Intel GPU for Media workloads.

  The Metrics Monitor library collects the following i915 kernel mode driver performance counters data:

  - Amount of time each GPU Engine spent executing tasks
  - Average actual GPU frequency

  Metrics Monitor allows to monitor the following GPU hardware units (engines):

  - Render engine (execution units)
  - Multi-Format CODEC (MFX)
  - Video Quality Engine

- Blitter engine

Each sample includes:

- a readme file for each sub-sample
- source and header files for each sub-sample

Samples package has one installer for all sub-samples.

Software & Hardware Requirements

Hardware:

- Hardware requirements are the same as described in **SDK** Release Notes
- (Optional) HDMI* 1.4, eDP* 1.1 or similar based monitor/TV as primary display
- (Optional) Active shutter glasses

Software:

- See `<msdk_install-folder>/media_server_studio_sdk_release_notes.pdf` for **SDK** general requirements. To build **Samples** you additionally need the following components to be installed and properly configured on the system:
- **For CentOS*:**

```
$ sudo yum install gcc g++ make cmake perl libX11-devel mesa-libGL-devel
```

- Samples can be built with GCC/G++ compiler version 4.6 and CMake* version 2.8.0 or higher. It is strongly recommended to use GCC version 6 or later since that's the first GCC version which has non-experimental support of C++11. Note: Samples should be built with the same version of compiler that was used for building dispatcher library.
- For samples with OpenCL (**Video Encoding, Video Transcoding, Video Motion Estimation, Interoperability**) it is required to install **Intel® Media Server Studio – Intel® SDK for OpenCL™ Applications** and **Intel® Media Server Studio – Graphics Drivers**.

Build Instructions

To build samples the following environment variable should be setup:

```
$ export MFX_HOME=/mediasdk/installation/folder
```

Go to the samples directory and execute `build.pl` script without arguments to see the help:

```
$ ./build.pl
Copyright (c) 2012-2018 Intel Corporation. All rights reserved.
This script performs Samples projects creation and build.
Usage: perl build.pl --cmake=ARCH,GENERATOR,CONFIG [--clean] [--build]
Possible variants:
 ARCH = intel64
 GENERATOR = make
 CONFIG = debug | release
Environment variables:
 MFX_HOME=/path/to/mediasdk/package # required
 MFX_VERSION="0.0.000.0000" # optional
Optional flags:
 --clean - clean build directory before projects generation / build
 --build - try to build projects before generation (requires
cmake>=2.8.0)
Examples:
 perl build.pl --cmake=intel64,make,debug [ only
```

```
generate projects ]
 perl build.pl --cmake=intel64,make,debug --build [ generate
and then build ]
 perl build.pl --cmake=intel64,make,debug --build --clean [ generate,
clean and build ]
```

Note that optional flag --enable-x11 temporary does not affect compilation process (temporary limitation) - so you should have libx11-devel installed, as it is described in Software and Hardware Requirements section.

Script invokes specified CMake* projects generator and optionally builds them (option available for cmake>=2.8.0). At the moment only make files generator for UNIX-like systems is supported. Project files will be placed in the folder named by the requested configuration; for example:

```
/__cmake
intel64.make.release
intel64.make.debug
```

To build generated project files use generator-specific approaches. For example, to build samples from make files invoke:

```
$ make -C <install-folder>/__cmake/intel64.make.release
```

With CMake older than 2.8.0 all samples can be built at once with the following command:

```
$ ./build.pl --cmake=intel64,make,release --clean --build
```

Binaries will appear in the following folder:

```
$ ls -1 __cmake/intel64.make.release/__bin/release/
sample_decode
sample_encode
sample_multi_transcode
sample_vpp
```

Running the Software

DRM backend specific notes:

For application to perform rendering through DRM it should be authorized to access graphics card:

- You should have no any other application blocking DRM mastership (like Xserver) to be running. You need root privileges to run:

```
$ sudo LD_LIBRARY_PATH=$MEDIASDK_INSTALL_FOLDER/bin/x64 \
$ sample_decode h264 -i input.264 -o output.yuv –vaapi –hw
```

X11 backend specific notes:

To use this backend for rendering user should be sure that he is logged into X server or is allowed to make connections to the X server:

- If user is allowed to use X and logged into machine remotely (thru SSH) he needs `DISPLAY` environment variable properly set. For example:

```
$ export DISPLAY=:0.0
$ sample_decode h264 -i input.264 -o output.yuv –vaapi –hw
```

NOTE: It has been found that in very rare cases (~20-30 machine hours to reproduce) system usleep/nanosleep calls with 1-10 milliseconds timeouts may take 2-30 minutes to complete. Since several usleep/nanosleep function calls are present in Samples, this issue may cause random hangs or an undefined behavior. Workaround: load Linux kernel with "nohz=off" parameter.

License

This software is distributed under the BSD-3 clause license, full text of license is reproduced below:

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

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 SSE3 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

---

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

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.