



NVIDIA VIDEO CODEC SDK SAMPLES GUIDE

SDK Samples Guide



REVISION HISTORY

Revision	Date	Author	Description
1.0	Nov 14, 2014	YG	Initial release.
2.0	Nov 25, 2015	EY	Update to NVIDIA Video Codec SDK 6.0 Added NVCUVID decode samples
3.0	June 10, 2016	SM/VU/GJ	Update to NVIDIA Video Codec SDK 7.0

TABLE OF CONTENTS

NVIDIA Video Codec SDK Samples Guide	1
Introduction	1
1. BUILDING SAMPLES	2
Windows	2
Linux	2
2. SAMPLES REFERENCE	3
2.1 NvEncoder	3
2.2 NvEncoderCudaInterop	4
2.3 NvEncoderD3DInterop	5
2.4 NvEncoderLowLatency	5
2.5 NvEncoderPerf	7
2.6 NvTranscoder.....	8
2.7 NvDecodeD3D9	9
2.8 NvDecodeD3D11	10
2.9 NvDecodeGL.....	11

NVIDIA VIDEO CODEC SDK SAMPLES GUIDE

INTRODUCTION

NVIDIA Video Codec SDK contains the following samples. The sample applications provided in the package are for demonstration purposes only and may not be fully tuned for quality and/or performance. Hence the users are advised to do their independent evaluation for quality and/or performance.

NvEncoder

This sample demonstrates the usage of basic encoding functionality.

NvEncoderCudaInterop

This sample demonstrates the usage of encoding with CUDA surfaces.

NvEncoderD3DInterop

This sample demonstrates the usage of encoding with D3D9 surfaces.

NvEncoderLowLatency

This sample demonstrates the usage of low latency features such as Intra Refresh and Reference Picture Invalidations.

NvEncoderPerf

This sample demonstrates the maximum achieved encoding performance.

NvTranscoder

This sample demonstrates the transcoding capabilities of NVENC.

NvDecodeD3D9

This sample demonstrates video decode with D3D9 visualization.

NvDecodeD3D11

This sample demonstrates video decode with D3D11 visualization.

NvDecodeGL

This sample demonstrates video decode and OpenGL visualization.

1. BUILDING SAMPLES

Windows

The Windows SDK samples are built using the Visual Studio IDE. Solution files (*.sln) are provided for Visual Studio 2010 and 2013.

Complete samples solution files exist at:

SDK7.0\Samples

Each individual sample has its own set of solution files at:

SDK7.0\Samples\<sample_dir>

To build/examine all the samples at once, the complete solution files should be used. To build/examine a single sample, the individual sample solution files should be used

Linux

The Linux samples are built using makefiles. To use the makefiles, change the current directory to the sample directory you wish to build, and run make:

```
$ cd <sample_dir>
$ make
```

2. SAMPLES REFERENCE

2.1 NvEncoder

The NvEncoder application demonstrates the code for doing a basic encoding using NVENC. It supports both H.264 and HEVC encoding with different presets. The application allows to configure bitrate, frame rate, number of B frames and allows the user to select from the given Rate Control Modes.

The following are the options that may be specified for NvEncoder Application.

```
-i <string> : Specifies the input YUV File that has to be encoded
-o <string> : Specifies the output bitstream file
-size <integer integer> : Specifies the input resolution width and height

-codec <integer>: Specifies the codec (0) - H264 and (1) - HEVC
-preset <string> :  hq - High Quality Preset
                   hp - High Performance Preset
                   lowLatencyHP - Low Latency High Performance Preset
                   lowLatencyHQ - Low Latency High Quality Preset
-startf <integer> : Specifies the starting frame Index for encoding.
Default value is zero
-endf <integer> : Specifies the end frame Index for encoding. Default
value is zero
-fps <integer> : Specifies the encoding frame rate
-gopLength <integer> : Specifies the GOP (Group of Pictures) Length
-numB <integer> : Specifies the number of B frames
-bitrate <integer> : Specifies the encoding average bitrate
-vbvMaxBitrate <integer> : Specifies the VBV Maximum Bitrate
-vbvSize <integer> : Specifies the Encoding VBV/HRD Buffer Size
-rcmode <integer> : Specifies the Rate Control Mode.
                   0 : Constant QP
                   1 : Single Pass VBR
                   2 : Single Pass CBR
                   4 : Single Pass VBR with Minimum QP
                   8 : Two Pass Frame Quality
                   16 : Two Pass Frame Size Cap
                   32 : Two Pass VBR

-qp <integer> : Specifies the qp value for Constant QP Rate Control Mode
-deviceType <integer> : 0 - DX9 Device Type
                       1 - DX10 Device Type
                       2 - DX11 Device Type
                       3 - CUDA Device Type
-inputFormat <integer> : Specify the input format
                        0: YUV 420
```

```
1: YUV 444
2: YUV 420 10-bit
3: YUV 444 10-bit
```

-help : Prints Help Information

2.2 NvEncoderCudaInterop

The NvEncoderCudaInterop application demonstrates the interoperability of the NVENC hardware encoder with CUDA surfaces.

The following are the options that may be specified for NvEncoderCudaInterop Application.

```
-i <string> : Specifies the input YUV File that has to be encoded
-o <string> : Specifies the output bitstream file
-size <integer integer> : Specifies the input resolution width and height

-startf <integer> : Specifies the starting frame Index for encoding.
Default value is zero
-endf <integer> : Specifies the end frame Index for encoding. Default
value is zero
-codec <integer>: Specifies the codec (0) - H264 and (1) - HEVC

-preset <string> :  hq - High Quality Preset
                     hp - High Performance Preset
                     lowLatencyHP - Low Latency High Performance Preset
                     lowLatencyHQ - Low Latency High Quality Preset
-fps <integer> : Specifies the encoding frame rate
-gopLength <integer> : Specifies the GOP (Group of Pictures) Length
-numB <integer> : Specifies the number of B frames
-bitrate <integer> : Specifies the encoding average bitrate
-vbvMaxBitrate <integer> : Specifies the VBV Maximum Bitrate
-vbvSize <integer> : Specifies the Encoding VBV/HRD Buffer Size
-rcmode <integer> : Specifies the Rate Control Mode.
                  0 : Constant QP
                  1 : Single Pass VBR
                  2 : Single Pass CBR
                  4 : Single Pass VBR with Minimum QP
                  8 : Two Pass Frame Quality
                 16 : Two Pass Frame Size Cap
                 32 : Two Pass VBR
-qp <integer> : Specifies the qp value for Constant QP Rate Control Mode
-deviceID <integer> : Specifies the GPU Device on which encoding will take
place
-help : Prints Help Information
```

2.3 NvEncoderD3DInterop

The NvEncoderD3DInterop application shows the interoperability with DX Surfaces. This application takes a directory of BMP files as an input and generates the output encoded file.

The following are the options that may be specified for NvEncoderD3DInterop Application.

```
-bmpfilePath <string> : Specifies the input RGB BMP file path
-o <string> : Specifies the output bitstream file

-size <integer integer> : Specifies the input resolution width and height
-codec <integer>: Specifies the codec (0) - H264 and (1) - HEVC

-preset <string> :  hq - High Quality Preset
                   hp - High Performance Preset
                   lowLatencyHP - Low Latency High Performance Preset
                   lowLatencyHQ - Low Latency High Quality Preset

-fps <integer> : Specifies the encoding frame rate
-gopLength <integer> : Specifies the GOP (Group of Pictures) Length
-numB <integer> : Specifies the number of B frames
-bitrate <integer> : Specifies the encoding average bitrate
-vbvMaxBitrate <integer> : Specifies the VBV Maximum Bitrate
-vbvSize <integer> : Specifies the Encoding VBV/HRD Buffer Size
-rcmode <integer> : Specifies the Rate Control Mode.
                   0 : Constant QP
                   1 : Single Pass VBR
                   2 : Single Pass CBR
                   4 : Single Pass VBR with Minimum QP
                   8 : Two Pass Frame Quality
                   16 : Two Pass Frame Size Cap
                   32 : Two Pass VBR

-qp <integer> : Specifies the qp value for Constant QP Rate Control Mode
-help : Prints Help Information
```

2.4 NvEncoderLowLatency

The NVEncoderLowLatency application demonstrates the encoding for low latency streaming. The application shows the usage of features such as Intra Refresh and Reference Picture Invalidation, Dynamic Resolution Change and Dynamic Bitrate Change that are extremely useful in error prone streaming environments.

The following are the options that may be specified for NvEncoderLowLatency Application.

```
-i <string> : Specifies the input YUV File that has to be encoded
-o <string> : Specifies the output bitstream file
-size <integer integer> : Specifies the input resolution width and height
-codec <integer>: Specifies the codec (0) - H264 and (1) - HEVC
-preset <string> :  hq - High Quality Preset
                   hp - High Performance Preset
                   lowLatencyHP - Low Latency High Performance Preset
                   lowLatencyHQ - Low Latency High Quality Preset
-startf <integer> : Specifies the starting frame Index for encoding.
Default value is zero
-endf <integer> : Specifies the end frame Index for encoding. Default
value is zero
-fps <integer> : Specifies the encoding frame rate
-bitrate <integer> : Specifies the encoding average bitrate
-vbvSize <integer> : Specifies the Encoding VBV/HRD Buffer Size
-rcmode <integer> : Specifies the Rate Control Mode.
                   0 : Constant QP
                   1 : Single Pass VBR
                   2 : Single Pass CBR
                   4 : Single Pass VBR with Minimum QP
                   8 : Two Pass Frame Quality
                  16 : Two Pass Frame Size Cap
                  32 : Two Pass VBR
-encCmdFile <string> : Specifies the name of the encode command file. The
commands can be given in the following format.
<encode command> <frame number> <param0> < param1>...<param15>
The following commands can be given in the encode command file:
0 : Dynamic Resolution Command <param0 = new Width> <param1 = new Height>
1 : Dynamic Bitrate Change <param0 = new bitrate> <param1 = new vbv size>
2 : Force IDR Frame
3 : Force Intra Refresh <param0 = intra refresh duration>
4. Invalidate Refrence Frame <param 0 = ref frame 0> <param1 = ref frame
1>..

```

2.5 NvEncoderPerf

The NvEncoderPerf application demonstrates the maximum encoding performance that may be achieved using NVENC. The application buffers a large number of input frames to prevent Disk I/O from being a bottleneck. The execution may be constrained by the Video Memory available on a system. The MAX_FRAMES_TO_PRELOAD compile time variable determines the number of frames that are buffered and may be reduced on systems with constrained Video Memory. The performance numbers give an indication of the Encode Compute Power available on NVENC. Applications need to be pipelined/multi-threaded in order to achieve maximum encode performance in practice.

The following are the options that may be specified for NvEncoderPerf Application.

```
-i <string> : Specifies the input YUV File that has to be encoded
-o <string> : Specifies the output bitstream file
-size <integer integer> : Specifies the input resolution width and height
-codec <integer>: Specifies the codec (0) - H264 and (1) - HEVC
-preset <string> :  hq - High Quality Preset
                   hp - High Performance Preset
                   lowLatencyHP - Low Latency High Performance Preset
                   lowLatencyHQ - Low Latency High Quality Preset
-startf <integer> : Specifies the starting frame Index for encoding.
Default value is zero
-endf <integer> : Specifies the end frame Index for encoding. Default
value is zero
-fps <integer> : Specifies the encoding frame rate
-gopLength <integer> : Specifies the GOP (Group of Pictures) Length
-numB <integer> : Specifies the number of B frames
-bitrate <integer> : Specifies the encoding average bitrate
-vbvMaxBitrate <integer> : Specifies the VBV Maximum Bitrate
-rcmode <integer> : Specifies the Rate Control Mode.
                   0 : Constant QP
                   1 : Single Pass VBR
                   2 : Single Pass CBR
                   4 : Single Pass VBR with Minimum QP
                   8 : Two Pass Frame Quality
                   16 : Two Pass Frame Size Cap
                   32 : Two Pass VBR
-qp <integer> : Specifies the qp value for Constant QP Rate Control Mode
-deviceType <integer> : 0 - DX9 Device Type
                       1 - DX10 Device Type
                       2 - DX11 Device Type
                       3 - CUDA Device Type
-inputFormat <integer> : Specify the input format
                        0: YUV 420
                        1: YUV 444
                        2: YUV 420 10-bit
```

3: YUV 444 10-bit

-help : Prints Help Information

2.6 NvTranscoder

The NvTranscoder application demonstrates transcoding using NVENC. The transcoder application supports 8-bit depth H.264 encoded files for input that may be transcoded to H.264 or HEVC files.

The following are the options that may be specified for NvTranscoder Application.

-i <string> : Specifies the input file that has to be transcoded
-o <string> : Specifies the output bitstream file
-size <integer integer> : Specifies the input resolution width and height for encoding. If not specified, it will use the width and height of the input file.
-codec <integer>: Specifies the codec (0) - H264 and (1) - HEVC
-preset <string> : hq - High Quality Preset
 hp - High Performance Preset
 lowLatencyHP - Low Latency High Performance Preset
 lowLatencyHQ - Low Latency High Quality Preset
-fps <integer> : Specifies the encoding frame rate. If not specified, it will use the fps of the input file.
-gopLength <integer> : Specifies the GOP (Group of Pictures) Length
-numB <integer> : Specifies the number of B frames
-bitrate <integer> : Specifies the encoding average bitrate
-vbvMaxBitrate <integer> : Specifies the VBV Maximum Bitrate
-vbvSize <integer> : Specifies the Encoding VBV/HRD Buffer Size
-rcmode <integer> : Specifies the Rate Control Mode.
 0 : Constant QP
 1 : Single Pass VBR
 2 : Single Pass CBR
 4 : Single Pass VBR with Minimum QP
 8 : Two Pass Frame Quality
 16 : Two Pass Frame Size Cap
 32 : Two Pass VBR
-qp <integer> : Specifies the qp value for Constant QP Rate Control Mode
-deviceID <integer> : Specifies the GPU Device on which encoding will take place
-help : Prints Help Information

2.7 NvDecodeD3D9

The NvDecodeD3D9 application demonstrates the code for handling video decode of MPEG-2, VC-1, H.264, HEVC, VP8 and VP9 with NVDEC. The application takes a video file as input and renders the result to D3D9 window for display. The sample will use an included MPEG-2 file if no video file is provided.

The following are the options that may be specified for NvDecodeD3D9 Application.

NVDecodeD3D9 [parameters]

-i=source	- Input file for decoding
-o=output.yuv	- Output YUV file
-decodecuda	- Use CUDA for MJPEG (Available with 64+ CUDA cores)
-decodedxva	- Use NVDEC for decode.
-decodecuvid	- Use NVDEC for decode (optimized)
-vsync	- Enable vertical sync.
-novsync	- Disable vertical sync.
-repeatframe	- Enable frame repeats.
-updateall	- always update CSC matrices.
-displayvideo	- display video frames on the window
-nointerop	- create the CUDA context w/o using graphics interop
-readback	- enable readback of frames to system memory
-device=n	- choose a specific GPU device to decode video with
-nframestart=n	- set the start frame number
-nframeend=n	- set the end frame number

2.8 NvDecodeD3D11

The NvDecodeD3D9 application demonstrates the code for handling video decode of MPEG-2, VC-1, H.264, HEVC, VP8 and VP9 with NVDEC. The application takes a video file as input and renders the result to D3D11 window for display. The sample will use an included MPEG-2 file if no video file is provided.

The following are the options that may be specified for NvDecodeD3D11 Application.

NVDecodeD3D9 [parameters]

-i=source	- Input file for decoding
-o=output.yuv	- Output YUV file
-decodecuda	- Use CUDA for MJPEG (Available with 64+ CUDA cores)
-decodedxva	- Use NVDEC for decode.
-decodecuvid	- Use NVDEC for decode (optimized)
-vsync	- Enable vertical sync.
-novsync	- Disable vertical sync.
-repeatframe	- Enable frame repeats.
-updateall	- always update CSC matrices.
-displayvideo	- display video frames on the window
-nointerop	- create the CUDA context w/o using graphics interop
-readback	- enable readback of frames to system memory
-device=n	- choose a specific GPU device to decode video with
-nframestart=n	- set the start frame number
-nframeend=n	- set the end frame number

2.9 NvDecodeGL

The NvDecodeGL application demonstrates the code for handling video decode of MPEG-2, VC-1, H.264, HEVC, VP8 and VP9 with NVDEC. The application takes a video file as input and renders the result to an OpenGL window for display. The sample will use an included MPEG-2 file if no video file is provided.

The following are the options that may be specified for NvDecodeGL Application.

NvDecodeGL [parameters]

-i=source	- Input file for decoding
-o=output.yuv	- Output YUV file
-decodecuda	- Use CUDA for MJPEG (Available with 64+ CUDA cores)
-decodecuvid	- Use NVDEC for decode (optimized)
-vsync	- Enable vertical sync.
-novsync	- Disable vertical sync.
-repeatframe	- Enable frame repeats.
-updateall	- always update CSC matrices.
-displayvideo	- display video frames on the window
-nointerop	- create the CUDA context w/o using graphics interop
-readback	- enable readback of frames to system memory
-device=n	- choose a specific GPU device to decode video with
-nframestart=n	- set the start frame number
-nframeend=n	- set the end frame number

Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA, the NVIDIA logo, GeForce, Quadro, Tesla, and NVIDIA GRID are trademarks and/or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2011-2016 NVIDIA Corporation. All rights reserved.