Integrating Intel® Media SDK with FFmpeg for mux/demuxing and audio encode/decode usages

Introduction

The provided samples intend to illustrate how Intel® Media SDK can be used together with the popular FFmpeg suite of components to perform container muxing and demuxing (splitting). The samples also showcase integration of rudimentary FFmpeg audio decode and encode.

The sample projects are based on the Intel Media SDK 2012 R3 samples (http://software.intel.com/en-us/articles/vcsource-tools-media-sdk/) with only small modifications to original code such as adding new mux/demux command line directives.
Modified areas of the code with new integration code are tagged with
"// ========== ffmpeg … integration ===========".

FFmpeg integration functionality resides in the *FFMPEGWriter* and *FFMPEGReader* classes, which are subclasses of the generic Intel Media SDK sample file reader/writer *CSmplBitstreamWriter* and *CSmplBitstreamReader* classes, respectively.

To enable simplistic implementation of FFmpeg audio processing functionality, just enable the *DECODE_AUDIO* or *ENCODE_AUDIO* define directive.

Since the provided samples are based on the Intel Media SDK decode/encode samples where video stream processing is in focus, the integration of FFmpeg container handling and audio processing may seem somewhat artificial. However, the samples should sufficiently illustrate the required Intel Media SDK and FFmpeg integration points making it quite straightforward to adapt to real-life applications, which would likely entail more sophisticated approaches such as threading, etc.

The provided samples illustrate the following use cases:

1. Demux "mp4" container file containing AVC(H.264) video stream and any FFmpeg supported audio stream. Decode the AVC(H.264) video stream and audio stream.
   (using "-split" option)
2. Encode to AVC(H.264) video stream and AAC audio stream. Mux streams into "mp4" container.
   (using "-mux" option)

3. Demux “mpeg” container file containing MPEG-2 video stream and any FFmpeg supported audio stream. Decode the MPEG-2 video stream and audio stream.
   (using “-split” option)
4. Encode to MPEG-2 video stream and MPEG audio stream. Mux streams into “mpeg” container.
   (using “-mux” option)
5. Encode to AVC(H.264) or MPEG-2 video stream and Ogg Vorbis audio stream. Mux streams into “mkv” container (Matroska).
   (using “-mkv” option)

The audio encode part of the sample assumes input of raw PCM data as follows:

- Using “mp4” or “mpeg” container: 16-bit signed integer samples, 2 channels
- Using “mkv” container: 32-bit float samples, 2 channels

You can generate raw PCM audio input file by demuxing a 2 channel @ 44100Hz audio stream using the provided decode sample. The encode sample can also easily be modified to support other audio input configurations. You can use such tools as “Audacity” to convert to/from different raw formats.

The current set of samples were tested and integrated using build “2012-08-27” of FFmpeg from:
http://ffmpeg.zeranoe.com/builds/

Please understand that this set of samples provides a snapshot of FFmpeg integration. The FFmpeg interfaces may change at any time, thus requiring modifications to the integration code.

Project file structure

Folder	Content	Notes
sample_decode – ffmpeg	- sample_decode.sln - include - src	Contains decode/demux sample project. FFMPEGReader class located in pipeline_decode.h/cpp
sample_encode – ffmpeg	- sample_encode.sln - include - src	Contains encode/mux sample project. FFMPEGWriter class located in pipeline_encode.h/cpp
ffmpeg	- include - lib_win32 - lib_x64	Contains FFmpeg component include files and pre-built binaries. See below "Requirements" section for details
sample_common	- include - src	Contains common Intel® Media SDK sample code functionality (this is a copy of the Intel Media SDK 2012 R3 sample_common folder)

Requirements

Intel Media SDK

The sample projects depend on Intel Media SDK API include files and dispatcher library. To be able to build the provided sample code, Intel Media SDK 2012 R2 or later must be installed. The SDK can be found here: http://software.intel.com/en-us/articles/vcsource-tools-media-sdk/

For more details about the Intel Media SDK samples, and Media SDK specific requirements or limitations, please refer to documentation and manuals of the SDK package.

FFmpeg

The provided sample projects do not include FFmpeg include and binary files. To be able to build the projects the required FFmpeg files for Windows* must therefore be downloaded from http://ffmpeg.zeranoe.com/builds/ (other Windows builds of FFmpeg also exists, but these have not been verified with this integration code)

Download the following packages:

1. "<build_id>-<arch>-dev.7z" archive file (from "<arch>-bit Builds (Dev)" section on the webpage)
2. "<build_id>-<arch>-shared.7z" archive file (from "<arch>-bit Builds (Shared)" section on the webpage)

From the above packages:

- Copy content of "include" folder in (1) to "ffmpeg/include" folder

- Copy *.lib content of “lib” folder in (1) to “ffmpeg/lib_<arch>” folder
- Copy *.dll content of “bin” folder in (2) to “ffmpeg/lib_<arch>” folder

Note: “<arch>” is either win32 or x64.

msinttypes

The folder “ffmpeg/include/msinttypes-r26” contains parameter type bridge required to be able to build FFmpeg project in Microsoft Visual Studio.

Download the required include files from http://code.google.com/p/msinttypes/

Note: The solution/projects were created using Microsoft Visual Studio* 2010, but there is nothing preventing the environment to be back-ported to older versions of Visual Studio, if needed.

How to build

1. Open the solution (“.sln”) file in either “sample_decode – ffmpeg” or “sample_encode – ffmpeg” folder
2. Select desired build configuration: Debug/Release, Win32/x64
3. Build the solution

How to execute workloads

Below are some example command line workloads.

1. Demux mp4 container and decode the AVC(H.264) video stream. If the container includes audio stream, it will be decoded into ”audio.dat” (assuming sample has been built with “DECODE_AUDIO”)

   ```
   sample_decode.exe h264 –i file.mp4 -hw -d3d -split –o video.yuv
   ```

2. Demux mpeg container and decode the MPEG-2 video stream. If the container includes audio stream, it will be decoded into ”audio.dat” (assuming the sample was built with “DECODE_AUDIO”)

   ```
   sample_decode.exe mpeg2 –i file.mpg -hw -d3d -split –o video.yuv
   ```

3. Encode raw YUV video data into AVC(H.264) video stream. Mux into mp4 container. If the sample was built with “ENCODE_AUDIO” the audio will be encoded using AAC encoder (raw audio PCM data read from file “audio.dat”).

   ```
   sample_encode.exe h264 -i video.yuv -w 640 -h 480 -o out.mp4 -hw -d3d -mux
   -b 1000 -f 30
   ```

4. Encode raw YUV video data into MPEG-2 video stream. Mux into mpeg container. If the sample was built with “ENCODE_AUDIO” the audio will be encoded using mpeg encoder (raw audio PCM data read from file “audio.dat”).

   ```
   sample_encode.exe mpeg2 -i video.yuv -w 640 -h 480 -o out.mpg -hw -d3d -
   mux -b 1000 -f 30
   ```

5. Encode raw YUV video data into the AVC(H.264) video stream. Mux into Matroska (mkv) container. If the sample was built with “ENCODE_AUDIO” the audio will be encoded using Ogg Vorbis encoder (raw audio PCM data read from file “audio.dat”).

   ```
   sample_encode.exe h264 -i video.yuv -w 640 -h 480 -o out.mkv -hw -d3d -mkv
   -b 1000 -f 30
   ```

Known issues

- MPEG2 muxing results in “buffer underflow” warning. However, the warning does not seem to impact the content or validity of the resulting mpeg container.

References

- Intel Media SDK
  http://software.intel.com/en-us/articles/vcsource-tools-media-sdk/

- Intel Media SDK forum
  http://software.intel.com/en-us/forums/intel-media-sdk/

- FFmpeg
  http://ffmpeg.org/

- FFmpeg Windows builds
  http://ffmpeg.zeranoe.com/builds/

- FFmpeg tutorial
  http://www.codediesel.com/tutorial/ffmpeg-a-beginners-guide-part-1/

Notices