# Table of Contents

# Encoding and decoding clips for the DVEVM on a Windows host

**Note!** This topic discusses version 1.10 of the DVEVM, as it comes with codecs with a certain feature set. If you are trying to use these instructions with another version of the DVEVM, consult the codec data sheets to see which features are supported in your version and adapt the encoder parameters accordingly. The process described should be the same however.

We will show how to encode clips for use with the DVEVM decode demo, and how to play clips from the DVEVM encode demo on a Windows host. The instructions should work on a Linux host too, as the open source tools used are cross platform. The easiest platform to encode on is actually MacOS X because a very good interface (ffmpegX, see below) is available for the open source command line tools used in this doc.

A big thanks to the codec team for their assistance in finding the right encoder parameters for the clips!

## Playback of DVEVM clips on a Windows host

You cannot use Windows Media Player to play the clips used by the DVEVM demos (unless plugins are available for download somewhere that lets you). This because the DVEVM demos are using elementary streams as opposed to transport streams. A full fledged multimedia application will use transport streams to facilitate (de)muxing of audio and video and other recovery and seek features, but the basic DVEVM demos only use elementary streams, which are essentially just the encoded frames stored one after the other in a file. An elementary stream can be extracted from a transport stream (like an AVI file) though, and a transport stream can be created around an elementary stream. See the tips and tricks section below for more information.

To play back the MPEG2 and H.264 elementary streams on a Windows (or Linux) host, the VideoLan player (http://www.videolan.org), or VLC as it's often called, is preferred. It has an easy to use GUI. It does not however play back MPEG4 elementary streams.

Another option which plays back all elementary streams supported by the DVEVM is mplayer. Make sure you select the GUI version and download it from http://www.mplayerhq.hu/design7/news.html. It is available for many operating systems, including Linux and Windows.

A shareware alternative for MPEG4 elementary streams is the "Elecard MPEG Player" available from http://www.elecard.com/download/index.php.

## Encoding clips for use with the DVEVM demos and codecs

To manipulate the raw video data we will use the VirtualDub application (http://virtualdub.org). It can be downloaded from SourceForge using http://virtualdub.sourceforge.net. We used version 1.6.15.

VirtualDub can be extended through external plugins. Probably the biggest collection is through another project called AviSynth. This is a scripted technology that uses external DLLs that perform various conversions or I/O. Download the AviSynth installer from http://sourceforge.net/project/showfiles.php?group_id=57023. We used version 2.5.6a. There is a list of available filters at http://avisynth.org/warpenterprises.

As our Davinci effect raw clips were originally in a MOV container we needed a plugin for this format. There is one available from http://forum.doom9.org/showthread.php?t=104293.

As the DVEVM 1.10 clips only support Top Field First (TFF) clips, a useful filter to convert to TFF from Bottom Field First (BFF) is available from http://www.geocities.com/siwalters_uk/reversefielddominance.html.

Once the raw data is in the correct format, we will use the open source command line tool ffmpeg to encode our clips. This is a command line tool with a lot of options and very little documentation, which can make it a little scary at first. Unfortunately we cannot seem to find a good and stable GUI front end for ffmpeg for Windows, but if you own a Mac you can use ffmpegX (http://www.ffmpegx.com) which is quite easy to use.

The ffmpeg application is a swiss army knife when it comes to video, and it has a lot of advanced options. You should be able to get clips which work with the DVEVM using the information in this document, but for more detailed information the application documentation is available from http://ffmpeg.mplayerhq.hu/ffmpeg-doc.html.

Building ffmpeg for Windows is unnecessary as a prebuilt binary is available from http://ffdshow.faireal.net/mirror/ffmpeg (I'm using rev. 6830). Wherever you download ffmpeg from, execute `ffmpeg -formats` and make sure that the 'E' flag is set for "xvid", "mpeg2video" and "h264" under the "Codecs" list (signaling that encode is available for those algorithms), and 'E' is set for "m4v", "m2v" and "h264" under "File Formats" (signaling writing to these elementary stream types is available as options). The ffmpeg tool can be compiled with various support form other open source libraries, and the above flags are necessary to be able to encode the formats used in the DVEVM.

In order to get more control of the parameters used to encode our H.264 clips we use the x264 tool directly (as opposed to through ffmpeg). The homepage is http://www.x264.nl and you can download a precompiled version from http://www.free-codecs.com/download/x264_Video_Codec.htm. Make sure your x264 binary is compiled with AviSynth support by executing `x264.exe --help`. You should see the following:

```
Infile can be raw YUV 4:2:0 (in which case resolution is required),
  or YUV4MPEG 4:2:0 (*.y4m),
  or AVI or Avisynth if compiled with AVIS support (yes).
```

For MPEG4 we will use XviD (but invoke it through ffmpeg). It's homepage is http://www.xvid.org. If you want to install the codec (for use with VirtualDub for instance) you can download a precompiled binary from http://www.koepi.org/xvid.shtml.

For RPM based Linux you can find ffmpeg, XviD and x264 binary and source from http://freshrpms.net. But note that this can be a cumbersome install (unless you point yum at the freshrpms repository which is quite slick) as there are many dependencies. All dependencies should be available from that site though, it's just a little time consuming.

## Manipulating the raw clip

VirtualDub prefers to work with AVI files, but can be exteded through AviSynth to read and write MOV files (and others), see above for download details. You will want to save an AVI file as ffmpeg works well reading these, but depending on which file your original content is in you might have to apply various filters to VirtualDub or AviSynth (VirtualDub uses AviSynth to gain scripting capabilities).

The decoders shipped with the DVEVM 1.10 only support Top Field First (TFF). You can see if you are trying to play a Bottom Field First (BFF) clip using the demos, as there will be a noticeable studder. This is easier to observe on a CRT screen than on an LCD TV. If so you can apply the "ReverseFieldDominance" script downloaded above. The below AviSynth example loads a MOV file and applies the field switching filter and finally saves this back as a MOV file on the file system:

```
QTInput("c:\test_BFF.mov")
ReverseFieldDominance()
```

```
QTOutput("c:\test_TFF.mov")
```

Load the script file, or the AVI file, into VirtualDub using "File->Open Video File". You should now see two display windows in VirtualDub. The left one is showing the frames before the VirtualDub (not AviSynth!) filters are applied, and the right one is showing the frames after the filters have been applied. You can play the clip and preview the effects or filters you have selected by pressing the play button at the bottom of the application.

Now you can apply the VirtualDub filters from "Video->Filters". Note that the AviSynth filters in your script file will be applied before the VirtualDub filters. If you are using a progressive codec like MPEG4 SP or H.264 BP you will likely want to deinterlace your content (if it is interlaced to begin with). Do this by selecting the "Add" in the filter window and adding the "Deinterlacer" filter (accept default parameters).

Some encoders, like x264, are picky about the data format of the raw clip. This can be changed using "Video->Color Depth..." and setting "Output format to decompressor / display" to "4:2:0 planar YCbCr (YV12)".

Now do "File->Save as AVI..." and write the file to disk. The raw video data is now ready to be encoded. If none of these steps were necessary, i.e. you already had the raw data in a file which ffmpeg accepts as input, your raw data was progressive (i.e. non-interlaced) or top field first, you don't need VirtualDub but can go straight to encoding the data below.

## MPEG2 Main Profile

As the TI MPEG2 decoder supports most if not all main profile features, this procedure is quite simple:

```
ffmpeg -i raw.avi -vcodec mpeg2video -b 8000000 encoded.m2v
```

This example creates an 8Mbps MPEG2 Main Profile encoded elementary stream (encoded.m2v) from the raw.avi file. Note that there is no issue with your raw video file being interlaced, as MPEG2 Main Profile supports interlaced content.

## MPEG4 Simple Profile

MPEG4 Simple Profile doesn't support interlaced content, and therefore you get a better quality encoded clip if you deinterlace the clip prior to encoding it (see above).

The open source XviD encoder gives good quality MPEG4 clips, so we will instruct ffmpeg to use XviD using:

```
ffmpeg -i raw.avi -vcodec xvid -b 4000000 encoded.m4v
```

This example creates an 4 Mbps MPEG4 Advanced Simple Profile encoded clip (encoded.m4v) from the raw.avi file. The encoded.m4v file will have to be renamed encoded.mpeg4 before being played with the DVEVM demos, as this is the file extension it expects.

Alternatively you can plug in XviD directly into VirtualDub using the instructions available at http://www.doom9.org/index.html?/xvid.htm. Note that you have to use "Advanced Simple Profile @ Level 5" and h.263 profile when configuring XviD using the GUI to make DVEVM 1.10 compatible clips.

In order to get better control of the XviD encoding parameters you can use the `xvid_encraw.exe` example application which comes with XviD. For some reason this application is not provided with the precompiled versions of XviD we found, but it's available in the source package so you can compile it yourself. It's a little difficult to use however, so if you need more control the VirtualDub XviD integration is preferred.

### H.264 Base Profile

H.264 Base Profile doesn't support interlaced content, and therefore you get a better quality encoded clip if you deinterlace the clip prior to encoding it (see above).

We will use the x264 utility directly in order to get better control of the encoding parameters. This tool is picky about it's input format, and you can set the format to YUV 420 in VirtualDub using the instructions above. Another way of accomplishing this is to use the AviSynth script support (which needs to be selected for inclusion when x264 is compiled):

```
AviSource("c:\raw.avi")
ConvertToYV12()
```

Put the above code in a file (c:\input.avs) and execute the command below to get a 3Mbps H.264 Baseline Profile encoded clip (encoded.264) from the input file (c:\raw.avi).

```
x264.exe --level 3 --no-cabac -r 2 -p 2 -B 3000000 --progress --merange 64 -o c:\encoded.264 c:\i
```

The most important parameters to make DVEVM 1.10 compatible clips are `--no-cabac` and `--level 3`. The rest can be tweaked.

## Transcoding and other ffmpeg tips and tricks

If you have encoded content, like mpeg2, which you want to show in another format, like h.264, ffmpeg can transcode the data for you. Execute `ffmpeg -formats` and make sure the input file format and codec is marked with 'D' (decode) and that your destination file format and codec is marked with 'E' (encode).

**Note!** By transcoding the data you normally lose quality versus the original content and compared to encoding from raw content.

This for example transcodes from MPEG1 encoded content in an ASF file to xvid (mpeg4) for the dvevm demos:

```
ffmpeg -i mpeg1.asf -vcodec xvid -b 1000000 encoded.m4v
```

Another useful thing ffmpeg can be used for is to strip elementary streams out of their containers (transport streams). The file formats m4v, m2v and h264 are elementary streams. This would for instance extract an m4v elementary stream from an AVI file by using the ffmpeg copy codec:

```
ffmpeg -i tranport.avi -v codec copy elementary.m4v
```

Now you should have an MPEG4 video elementary stream (note that this conversion will strip the audio) which you can play on the DVEVM using the demos.

-- NiclasAnderberg - 13 Nov 2006