

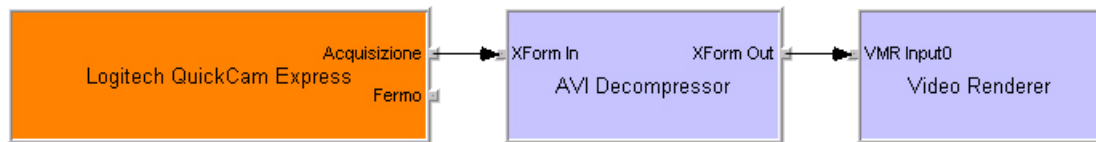
This paper focuses on the different approaches to RAW versus MPEG2 capture devices.

These two devices are from the same family (WDM capture devices) and will be enumerated in any application (Video Editing front-ends, Messenger, ..), but MPEG2 devices won't work with most of them. Why?

Because most application just add the selected capture filter to the graph builder and call *RenderStream* function, in most cases this will perform the task but not always, the problem is that MPEG2 capture filter have often to be configured (*IEncoderAPI*) and we need a MPEG-2 multiplexer (to have a video and an audio output) and two MPEG Decoder (audio and video) to convert encoded stream to uncompressed format (to link with a renderer for example).

Now we will see some simple examples..

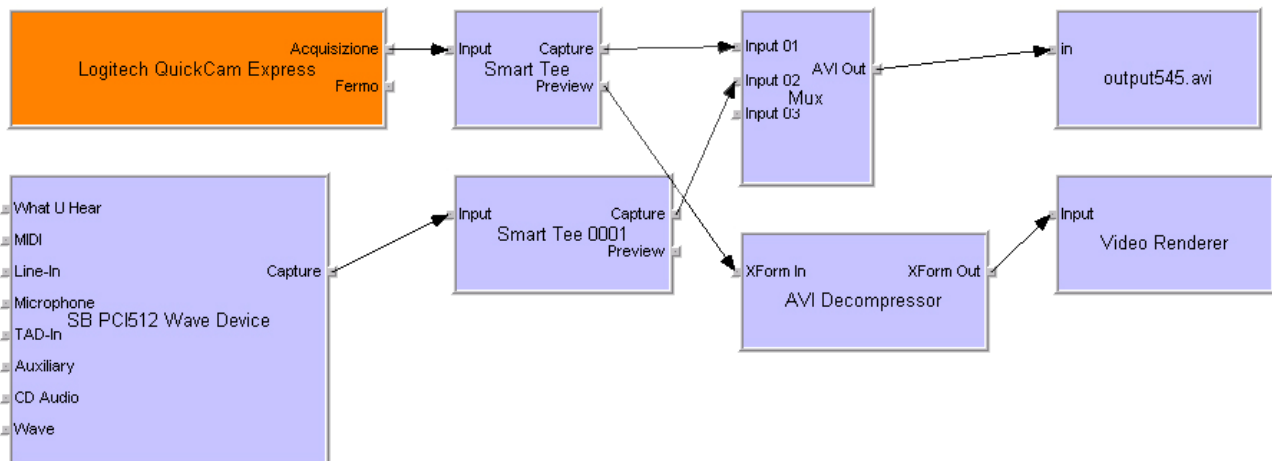
Usual video capture (from RAW device) (Ex. common USB/Firewire VideoIn for consumers)



Where the filter *AVI Decompressor* converts input format: I420 352x288 12bit (output of the Video Capture Filter) to YUV2 352x288 16bit (that is compatible with *Video Renderer*).

vetDirectXInput2 Graph

VETLib *vetDirectXInput2* module implements a graph just a bit more complex than the first case:

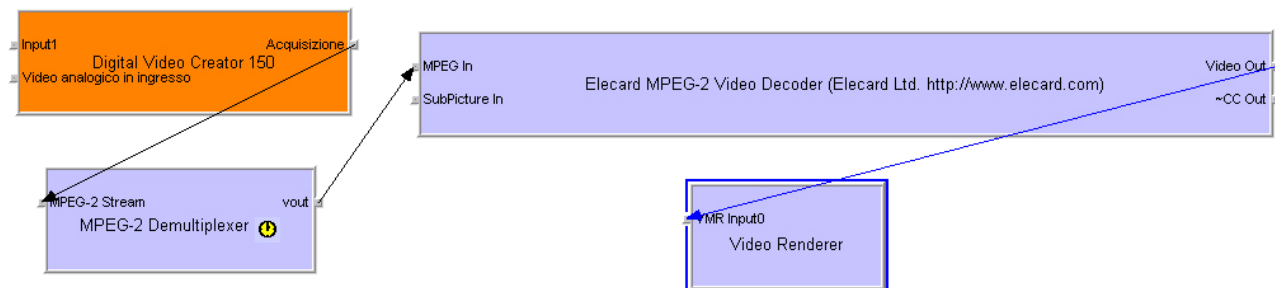


Here we have audio and video capture both, with video preview, the *AVI Multiplexer* filter sends output to the *File Writer* filter (*output545.avi*), *Smart Tee 0001* is not very useful here (we could link *Sound Blaster PCI 512 Wave Device* source to *AVI Mux* directly) but it remarks the need of an *hub* for previewing the stream while capturing (any stream).

Current version of *VETLib vetDirectXInput2* module still doesn't implement frame grabbing and MPEG2 device capture (does preview), hopefully soon it will be upgraded.

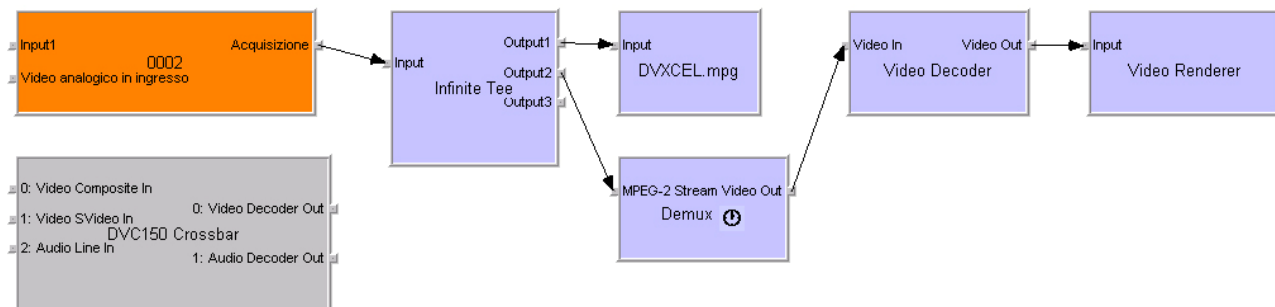
Encoded Video Input Graph (MPEG2 Capture devices):

In previous graph we had an uncompressed stream as source, in following examples i used a Dazzle Digital Video Creator 150 (USB2), video stream (from analog to..) is hardware encoded to MPEG2 format (video + audio), we obviously need to design a new graph:



Here we just render video to a window, audio is not processed and get lost after *MPEG-2 Demux*, the filter *Elecard MPEG-2 Video Decoder* converts stream to uncompressed YUV2 and then stream is rendered to a preview window (*Video Renderer*).

If we want to capture and preview the stream at same time:



Output Media type of capture filter (002 (it's the *DVC Capture Filter*) » "*Acquisizione*") is *MPEG2_PROGRAM*, so it can be redirected to an output file (*Infinite Tee* » *Output1*) or must be demultiplexed: stream is decoded (*Video Decoder*) into *RGB565* and can be rendered, here we are not really interested in audio but we just need to add an *MPEG Audio Decoder* to the *MPEG-2 Demultiplexer* filter and eventually a *Default DirectSound Device* filter for audio preview.

Elecard MPEG-2 Video Decoder (in second graph) and *Video Decoder* (in third graph) are different implementations but they do the same job (it is the same for *MPEG2 Demultiplexer* and *Demux*), in my test also the *MainConcept MPEG Video Decoder* worked fine (but i got problems with some other decoders).

We use again an *Infinite Tee* (or *Smart Tee*) as stream hub for previewing while capturing, the main difference between these two hubs is that *Smart Tee* is designed for capturing and overlay previewing (stream is not copied to both output, but preview share same buffer to speed up), while *Infinite Tee* just copies input stream to each output (lower performance).

Also note that these filters (*Tees*) are independent from data media type and you must se input pin first, then link output pins (because the filter need to update outputs data format setting it as input format).

MPEG Demultiplexer Notes:

This filter must be configured because output pins are “dynamic”:

1. Create a new output pin (“video”) with media type: *MPEG2 PROGRAM VIDEO*
2. Create a new output pin (“audio”) with media type: *MPEG2 PROGRAM AUDIO*
3. Setup both pins:
 1. set *stream_id* (usually 0xE0 for video and 0xC0 for audio);
 2. set *Elementary Stream (A/V only)*;
 3. map pin.
4. Connect Multiplexer into graph.

Used Filters Monikers:

AVI Decompressor:

@device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE86}\{CF49D4E0-1115-11CE-B03A-0020AF0BA770}

Smart Tee:

@device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE86}\{CC58E280-8AA1-11D1-B3F1-00AA003761C5}

File Writer:

@device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE86}\{8596E5F0-0DA5-11D0-BD21-00A0C911CE86}

Video Renderer:

@device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE86}\{70E102B0-5556-11CE-97C0-00AA0055595A}

AVI Mux:

@device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE86}\{E2510970-F137-11CE-8B67-00AA00A3F1A6}

MPEG2 Demultiplexer

@device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE86}\{AFB6C280-2C41-11D3-8A60-0000F81E0E4A}

References

Microsoft Developer Network:

<http://msdn.microsoft.com>

VETLib WebCenter:

<http://www.ewgate.net/vetlib>

[you may try vetDirectXInput2 module downloading (free) VETLib WorkShop]

Applies To

Microsoft DirectX 9.x (note that release 9.0b has some bugs about PAL)