



ENCIRIS
TECHNOLOGIES

Enciris Technologies LT-XXX Framegrabber Directshow Framework Description Manual

LT-101, LT-102, LT-122, LT-124, LT-200

November 2016

Contents

1.	INTRODUCTION	3
2.	LT-XXX Directshow Driver Description	6
3.	Visual tool for building and testing filter graphs for DirectShow	7
4.	LT-XXX Filter Property Page Description	9
1.1.	COMMON CONFIG VC-1/H264	11
1.2.	BITRATE ADJUSTMENT	13
1.3.	VIDEO RESIZE	16
1.4.	NOISE FILTER	19
1.5.	COLOR ADJUSTMENT	20
1.6.	INTERLACE	21
1.7.	TIME STAMPS	23
1.8.	OSD (On Screen Display)	24
1.9.	MISCELLANEOUS	26
1.10.	ERROR HANDLING	28
1.11.	CUSTOM RESOLUTION	29
1.12.	INFO	30
5.	Available API's in Directshow SDK	32
6.	LT-XXX Directshow Filter Custom API	33
7.	ASF Recording Filter Description (Itasfmux.ax) (available for VC-1 compression only)	59
8.	ASF Recording Filter API (Itasfmux.ax) (available for VC-1 compression only)	61
9.	Streaming Server Filter Description (asfNetworkSink.ax). (available for VC-1 compression only)	63
10.	Streaming Server Filter API (asfNetworkSink.ax) (available for VC-1 compression only)	65
11.	C++ Examples Applications:	67
12.	Filter Graph Examples	74
1.1.	Record to AVI File	74
1.2.	Record to ASF and Preview Uncompressed Video	74
1.3.	Decode and Preview Compressed Video	74
1.4.	Send Compressed Video over Network via HTTP	75
13.	Conclusion	76

1.INTRODUCTION

All LT-XXX devices can capture and compress H.264 or VC-1 video. The device can be configured on the fly for each compression type.

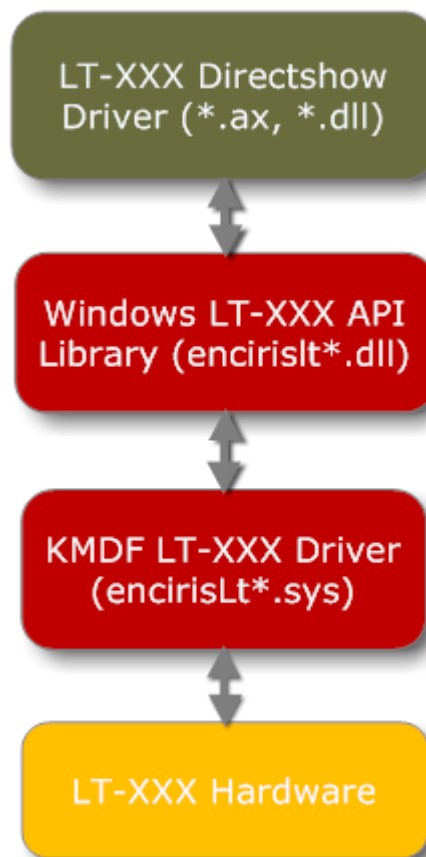
Important note: The VC-1 driver is a legacy driver and is only provided on demand. From driver version 2.87 on, only H.264 feature will be supported. VC-1 driver can be provided separately but no updates are planned for it.

Windows applications based on the LT-XXX video capture boards can either use the DLL based API described in “encirisLtAPI_vxxx.pdf” or directly connect to dedicated Directshow filter (also refer to as “Enciris LT-XXX” in e.g. Graphedit). LT-XXX capture device appears as Directshow capture device in the device list.

Numerous examples in the LT-XXX SDK show how to use Directshow filter programmatically. You need to install the Enciris LT-XXX SDK in order to experience them.

No “AVStream” driver has been developed for the LT-XXX video capture board. The “Enciris LT-XXX” Directshow (wrapper) driver is directly built on top of the LT-XXX middleware library.

The software hierarchy is depicted in the picture below.



A Directshow driver is usually based on “AVstream wdm-minidriver”. We use a different approach for LT-XXX Directshow driver. The LT-XXX Directshow driver directly builds on a user space windows middleware library (encirisLT*.dll). The Directshow driver is implemented as a push source (or Live Source) filter that exposes its capture/preview interfaces to any Directshow capture/streaming applications. For that reason, Directshow driver will be referred to as Directshow filter since it is the same instance in our context.

If one has used default location for driver installation, LT-XXX Directshow filter will be installed in following directory:

C:\Program Files\Enciris Technologies\LT-XXX driver\directshow

*.dll and *.manifest files in this directory are Microfoft Visual Studio runtime libraries that are needed by the filter.

All available Directshow (*.ax or *.dll) filters are stored in this directory:

Two special filters are also included in this directory (these filters are used for VC-1 compression only):

1. Itasfmux.ax: An ASF audio video multiplexer and dump filter
2. asfNetworkSink.ax: An HTTP streaming server

Itasfmux.ax and asfNetworkSink.ax filter can be directly connected to audio and video pin of LT-XXX Directshow filter. These filters will be described later.

*.bat files register or unregister the filters into Windows Registry. These files are used once during installation and are usually not needed later.

Any Directshow based application that can capture from a VC-1/H264 compressed or UYVY uncompressed video source should be able to interface with LT-XXX driver.

Here is a list of programs (among others) that interface to our LT-XXX Directshow driver/filter:

- AmCap
- VLC
- Unreal Media Server
- Windows media encoder
- etc...

2.LT-XXX Directshow Driver Description

The LT-XXX Directshow filter has 3 pins:

1. A compressed VC-1 or H.264 pin that outputs VC-1/H.264 video elementary stream. VC-1 is an SMPTE (SMPTE 421M 2006) based video standard from Microsoft. H.264/MPEG-4 Part 10 or AVC (Advanced Video Coding) is a ISO/IEC standard for video compression.
2. A Raw pin that outputs raw uncompressed video in FOURCC: "UYVY" YUV4:2:2 format. This pin is usually used for preview. User can simultaneously record compressed stream and preview live video. Note that video frame-rate may be limited by USB/PCI/PCIe bandwidth. E.g. up to 1080@15fps is feasible through PCI. You can also use downscaling feature in order to capture full uncompressed video framerate.
3. An audio pin that can output one of 2 different audio formats
 - a. Uncompressed 16 bit stereo PCM audio at 48Khz
 - b. WMA8 compressed stereo audio at 48Khz

3. Visual tool for building and testing filter graphs for DirectShow

There are visual tools for building and testing filter graphs for DirectShow, such as:

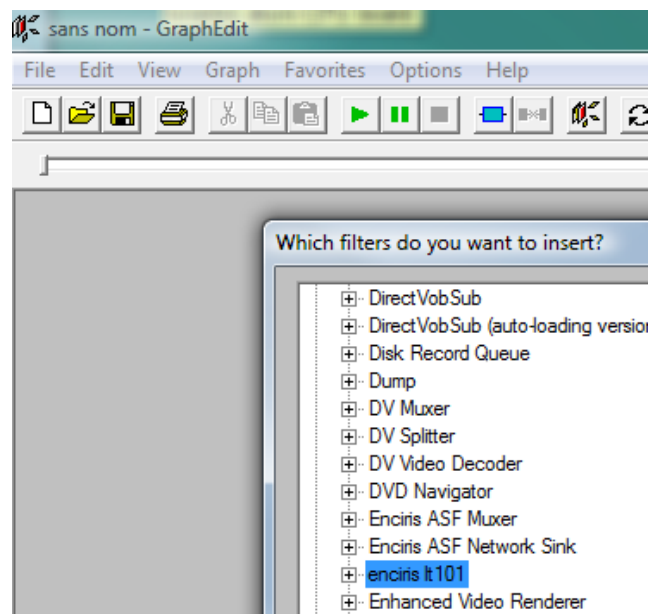
- GraphEdit
- GraphStudio
- GraphEditPlus
- Etc...

The most commonly used tool is GraphEdit.

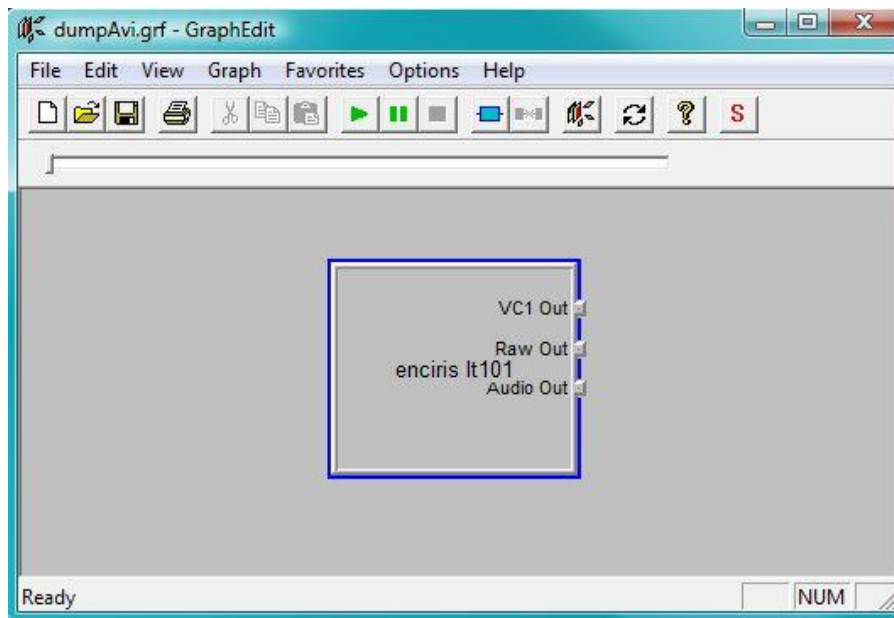
GraphEdit is a utility which is part of the Microsoft DirectShow SDK. It is a windows utility that can be used to load and connect Directshow filters to each-other.

The LT-XXX Directshow filter can be selected from Graphedit's

Graph->Insert Filters menu as depicted bellow:



A simple filtergraph only containing “Enciris LT-XXX” filter edited with Graphedit tool is shown in the picture bellow.

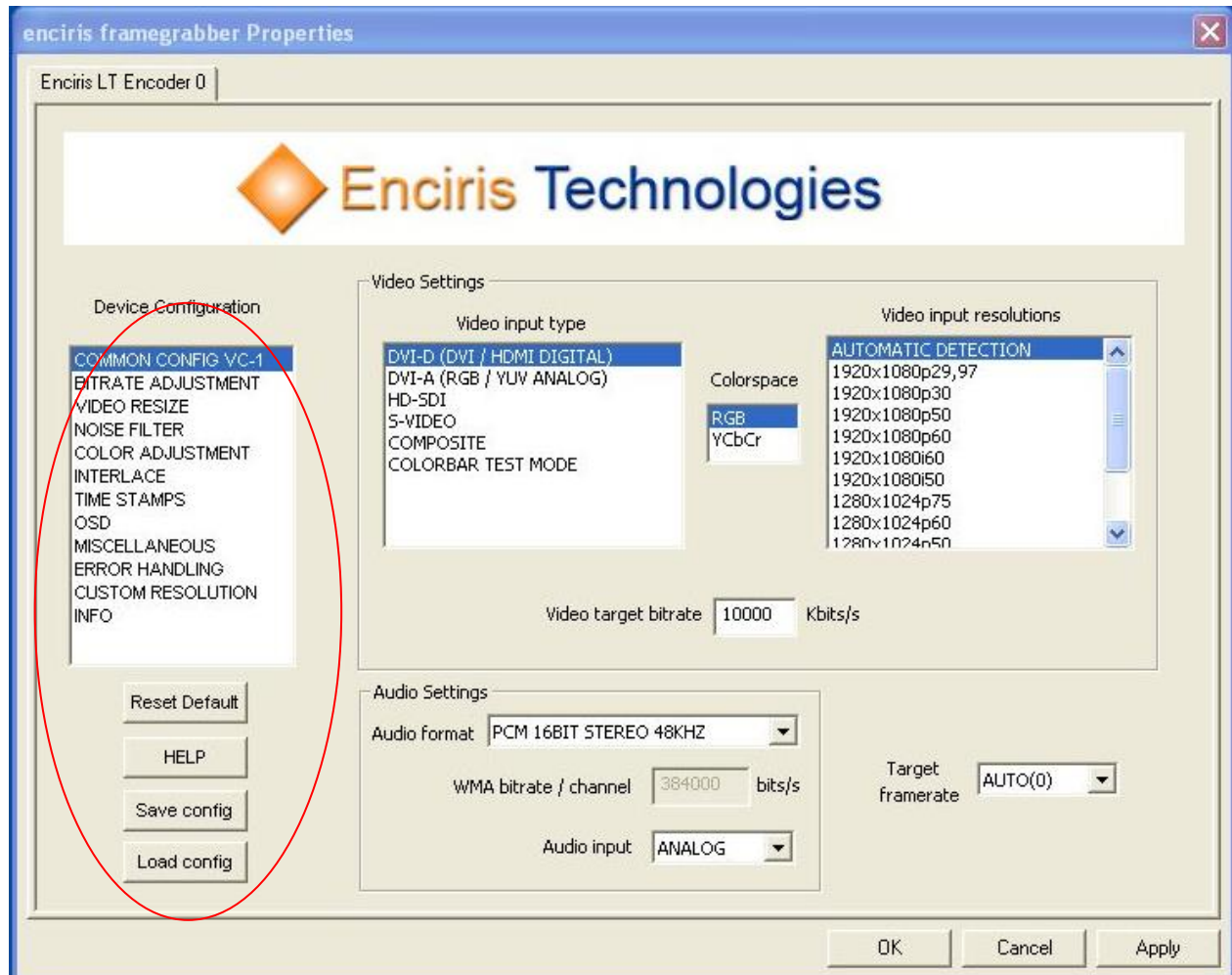


This filter appears in all Directshow compliant applications as “Enciris LT-XXX”.

4.LT-XXX Filter Property Page Description

“Enciris LT-XXX” supports directshow property page .

The property page’s Graphical User Interface (GUI) exposes a subset of all available interfaces to the user and transmits the user commands to the windows Middleware API.



Directshow Property Page

Some generic tools are made available in order to facilitate the configuration

- Reset Default
 - Will put the board into a known initial state
- Help
 - Will provide help on activated *configuration menu*.
- Save Config

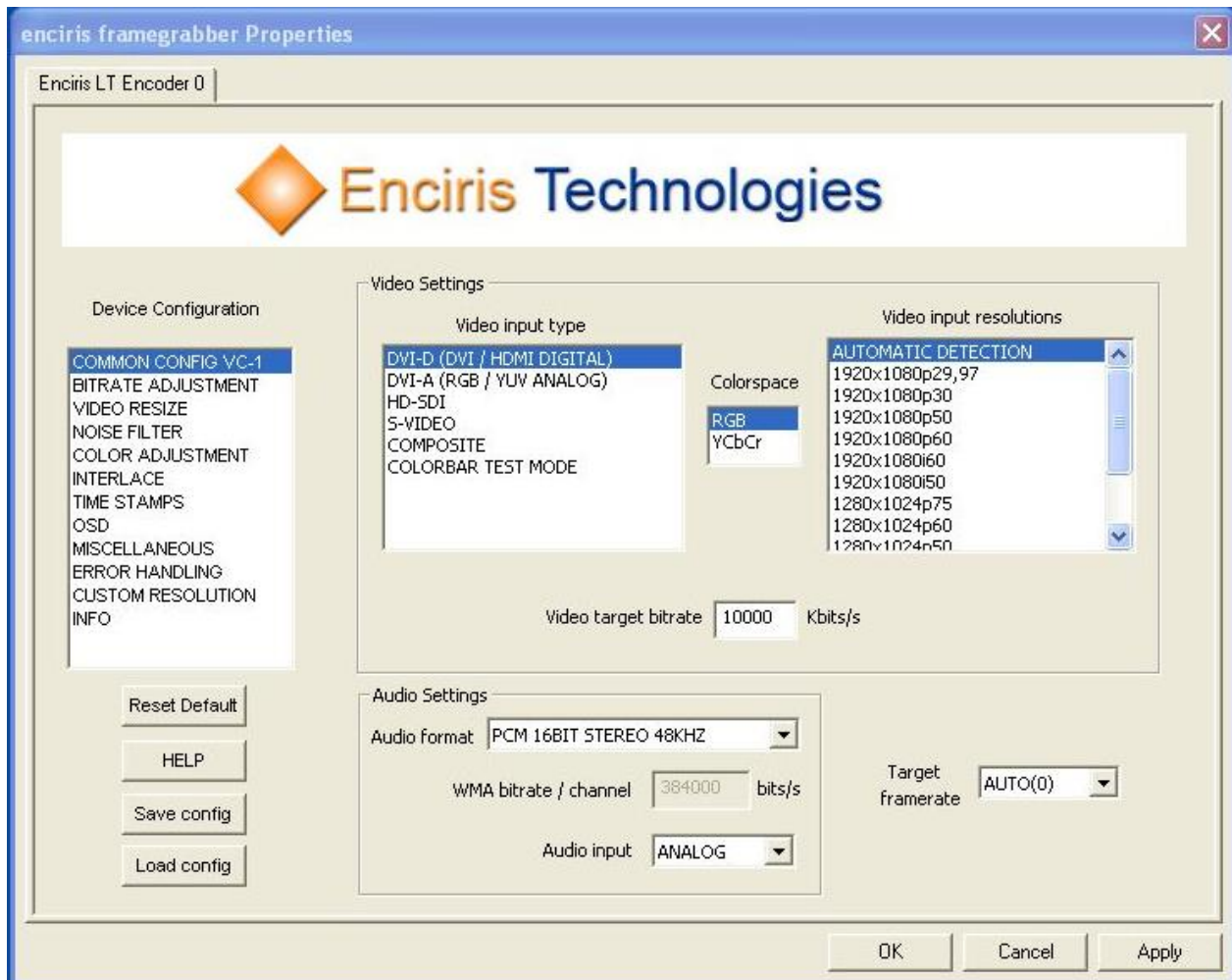
- Saves actual configuration to a file. The last configuration is always written to Windows registry
- Load Config
 - Loads the configuration that has been written to a file

Furthermore, one can select among 12 device *configuration menus*:

- COMMON CONFIG VC-1 or COMMON CONFIG H264
- BITRATE ADJUSTMENT
- VIDEO RESIZE
- NOISE FILTER
- COLOR ADJUSTMENT
- INTERLACE
- TIME STAMPS
- OSD
- MISCELLANEOUS
- ERROR HANDLING
- CUSTOM RESOLUTION
- INFO

The device configuration menus are described as follows:

1.1. COMMON CONFIG VC-1/H264



- Video input type
This sub-menu specifies the video input
- Video input resolution
Video is normally automatically detected. If this is not the case, one can force the corresponding mode manually.
 - Colorbar test mode
This mode is usually used to test if the LT-XXX board is working properly without the need of having actual video input connected
- Colorspace
No auto-detection is possible yet for the video colorspace and one must set this manually. If the wrong colorspace is chosen, the image appears to be green.
- Video target bitrate

This is the target bitrate of the compressed video.

- Audio format

Selects between 48Khz 16bit stereo PCM and WMA8 (Windows Media Audio) format with user defined target bitrate

- WMA bitrate/channel

Choose Wma target bitrate

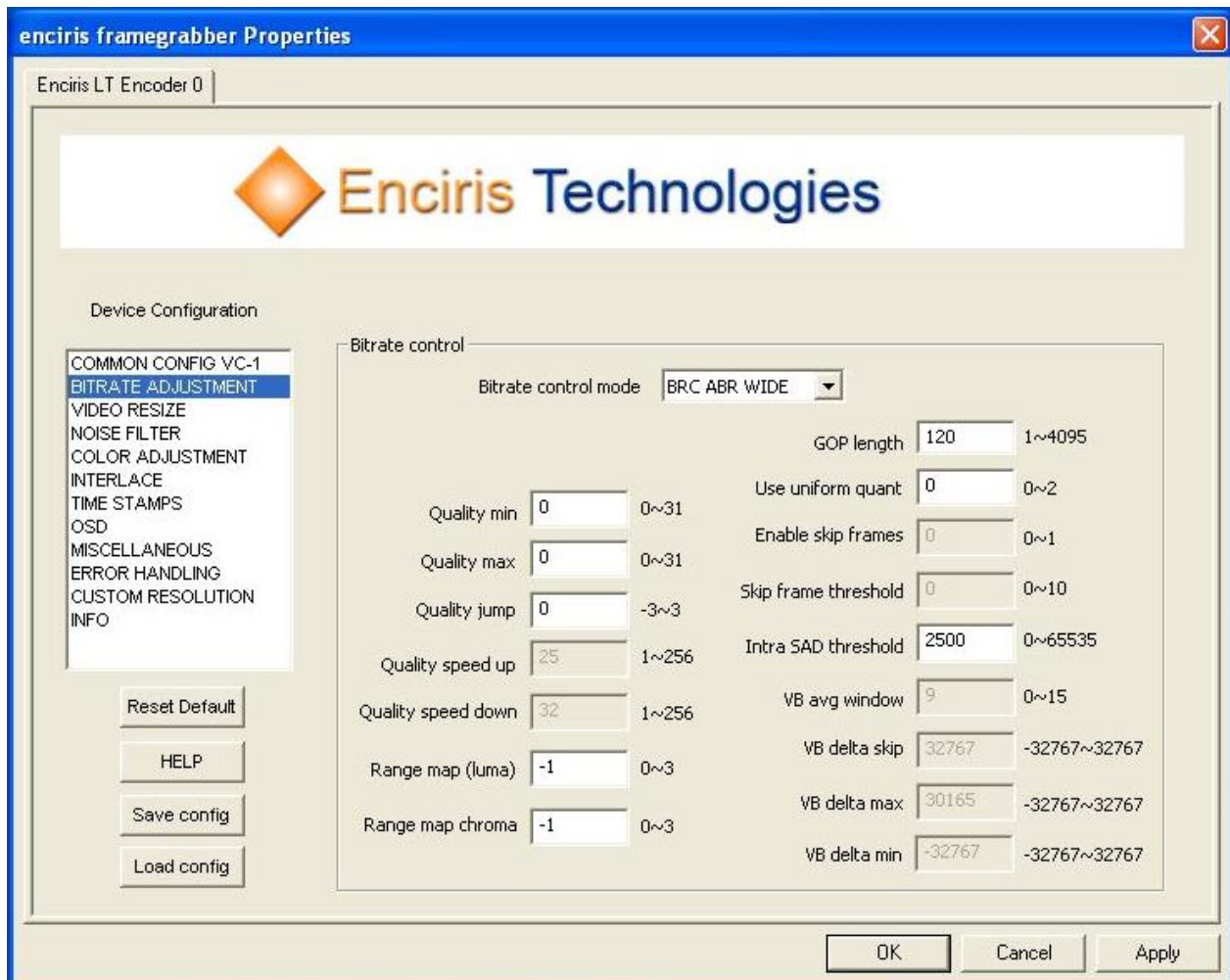
- Audio input

Selects between analog or embedded audio signal. Audio can be embedded on HDMI or SDI signal if this video input type is supported.

- Target framerate

If “Auto” mode is selected, the LT-XXX will try to provide the video at the maximum possible frame rate allowed by the bus type (USB, PCI, PCIe) or the video input resolution (1080p60) may be reduced to 1080p30 depending on the LT-XXX model used. If > 0, framerate decimation will be used to reduce the framerate

1.2. BITRATE ADJUSTMENT



- **Bitrate Control Mode**

This parameter sets the Bitrate Control policy to be applied.

- Following modes exist:

- **BRC ABR WIDE (ABR = Average BitRate)**
 - VBR with wide convergence window. Used in recording application. There can be big jump around the target bitrate
- **BRC ABR MEDIUM**
 - VBR with medium convergence window. For recording application. There can be medium jump around the targeted bitrate
- **BRC VBR**
 - For networking application. The maximum bitrate should not exceed target bitrate by more than 10%,

unless target bitrate is too low. Forced skip frame can occur

- BRC CONSTANTQ

- This is a constant Quality (or constant Quantizer) mode. Use “constant quality factor” menu to set wished quality

- GOP Length

GOP (Group Of Picture) length. This is the amount of P-frames within two I-frames. E.g. IPPPPIPPPPI... Has a Gop length of 5

- Use uniform quant (only for VC-1)

Selects between uniform and non-uniform quantization according to VC-1 specification (SMPTE 421M-2006). If set to 0, an automatic selection will be performed.

- Enable skip frames (only for VC-1)

Enables Skipped frames in the VC-1 stream. The decision to skip frames is made internally but can be influenced by changing the "VB delta xxx" parameters.

- Skip frames threshold

Not used yet.

Intra SAD threshold (only for VC-1)

This affects the number of Intra coded macroblock within an inter coded picture (P-frames).

VB avg window

Virtual Buffer sliding window size. This affects the reaction time of the bitrate control algorithm.

This value is expressed as power of two.

Quality min

minimum Quantizer (PQIndex) for given BRC algorithm. In constant quality mode ("BRC CONSTANTQ") this sets the quality factor.

The "Quality min" parameter is coded as follows:

0: Auto (internally decided):

1: Best quality:

31: Worst quality:

Quality jump

Is the offset between Intra frame quantizer and Inter frame quantizer.

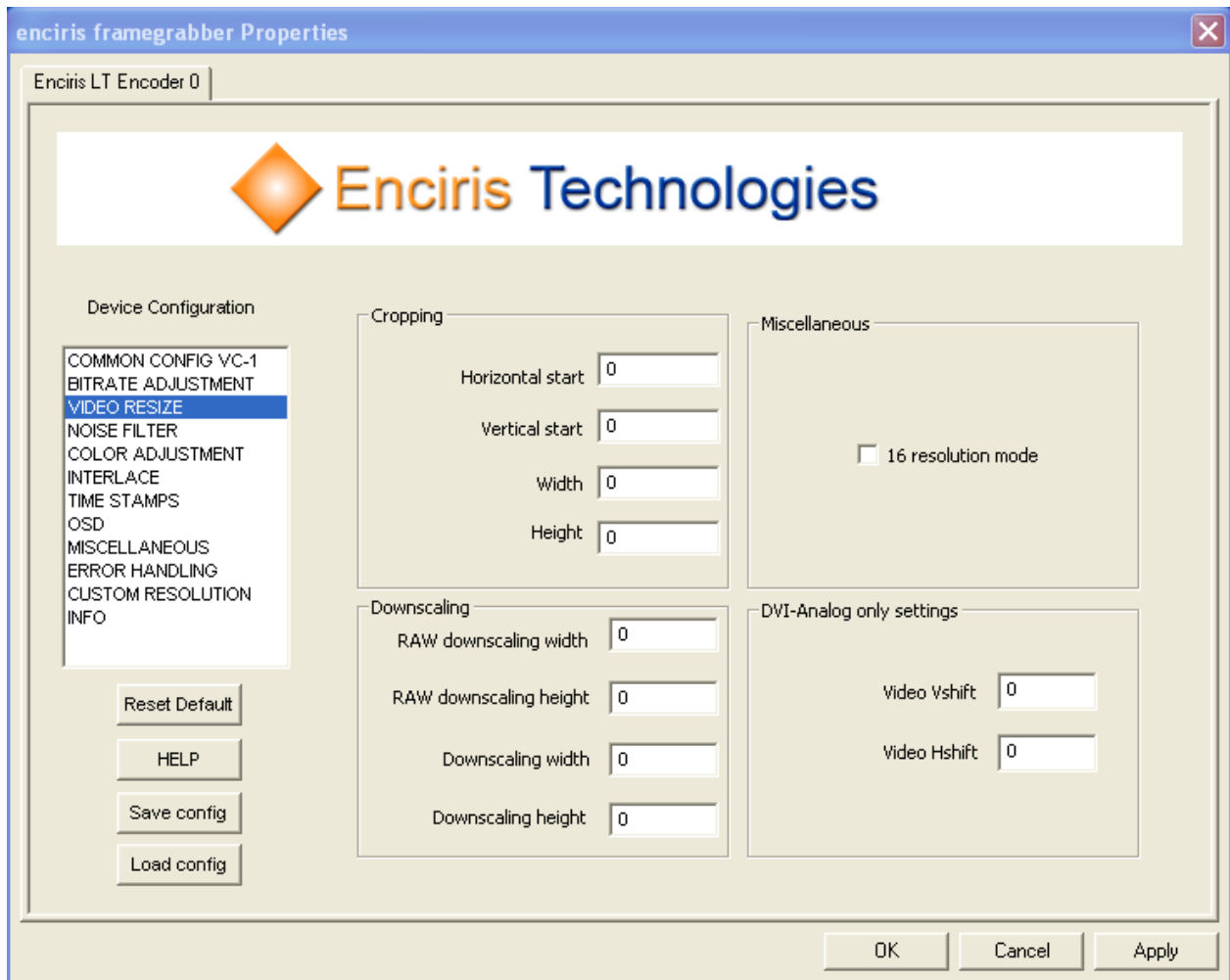
Quality speed up/down:

Adjusts the BRC algorithm reaction speed to scene changes.

Range map luma/chroma (only for VC-1)

Is as described in VC-1 specification (SMPTE 421M-2006). This is basically an additional quantization of the color components.

1.3. VIDEO RESIZE



- Horizontal start
 - Defines horizontal starting point of image
- Vertical start
 - Defines vertical starting point of image
- Width
 - Defines width of the image (must be lower than original width)
- Height
 - Defines height of the image (must be lower than original height)
- Raw Downscaling width
 - Horizontal downscaling factor for uncompressed video

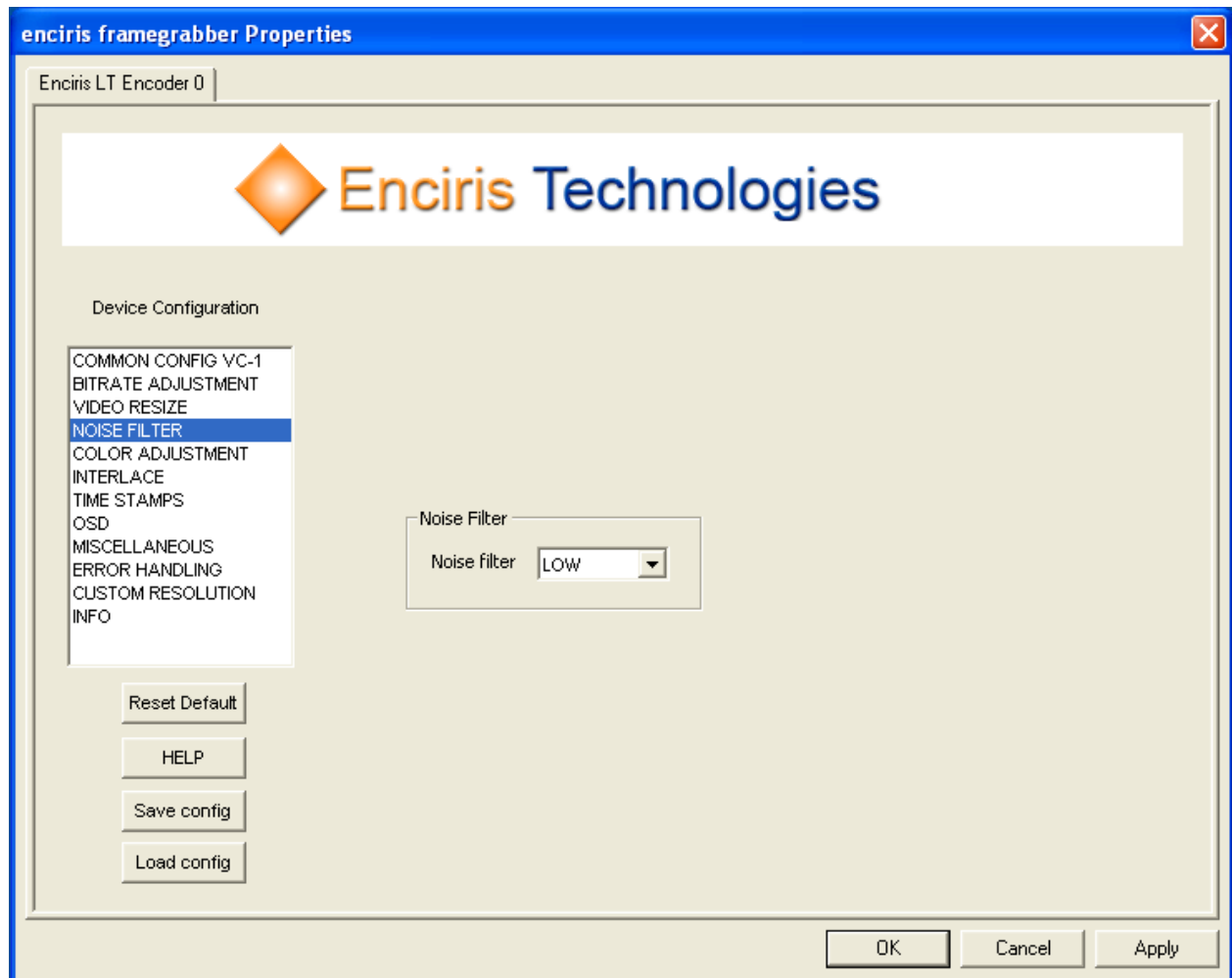
- New width. Must be inferior or equal original width. If this value is different from 0, downscaling will be done in Hardware.
- Raw Downscaling height
 - Vertical downscaling factor for uncompressed video
 - New height. Must be inferior or equal original height. If this value is different from 0, downscaling will be done in Hardware.
- Downscaling width (VC-1/H.264)
 - Horizontal downscaling factor for compressed video
 - New width. Must be inferior or equal original width. If this value is different from 0, downscaling will be done in Hardware.
- Downscaling height (VC-1/H.264)
 - Vertical downscaling factor for compressed video
 - New height. Must be inferior or equal original height. If this value is different from 0, downscaling will be done in Hardware.

The downscaled resolution will be further rounded to match hardware internal criteria. See encirisLtAPI_vxxx.pdf for more details about downscaling.

- Video Vshift/Hshift
 - These settings set the horizontal and vertical shift for (DVI-A) analog video.
- 16 resolution mode:
 - This parameter will govern the rounding of 1080p resolution to either 1920x1072 or 1920x1088 pixels.
 - Some video resolutions such as 1920x1080 cannot be processed natively as they are not a multiple of 16x16pixels (1080/16 is not an integer).

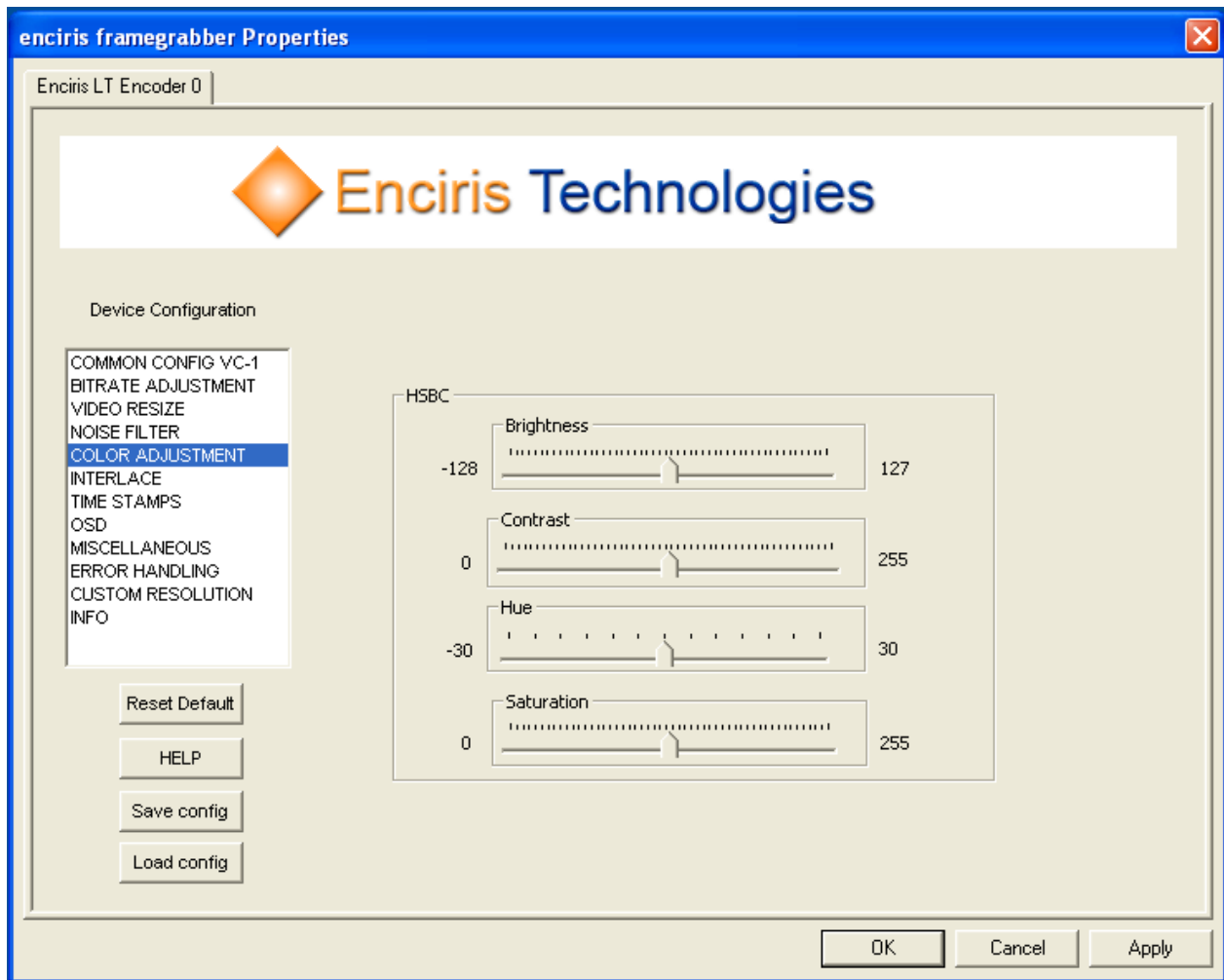
- Only multiple of 16 sized images can be compressed. One can do either black line/column insertion or remove 8 lines/column to accommodate for that.
- In the case of 1080, the rounding can be done to 1072 (0) or to 1088 (1).

1.4. NOISE FILTER



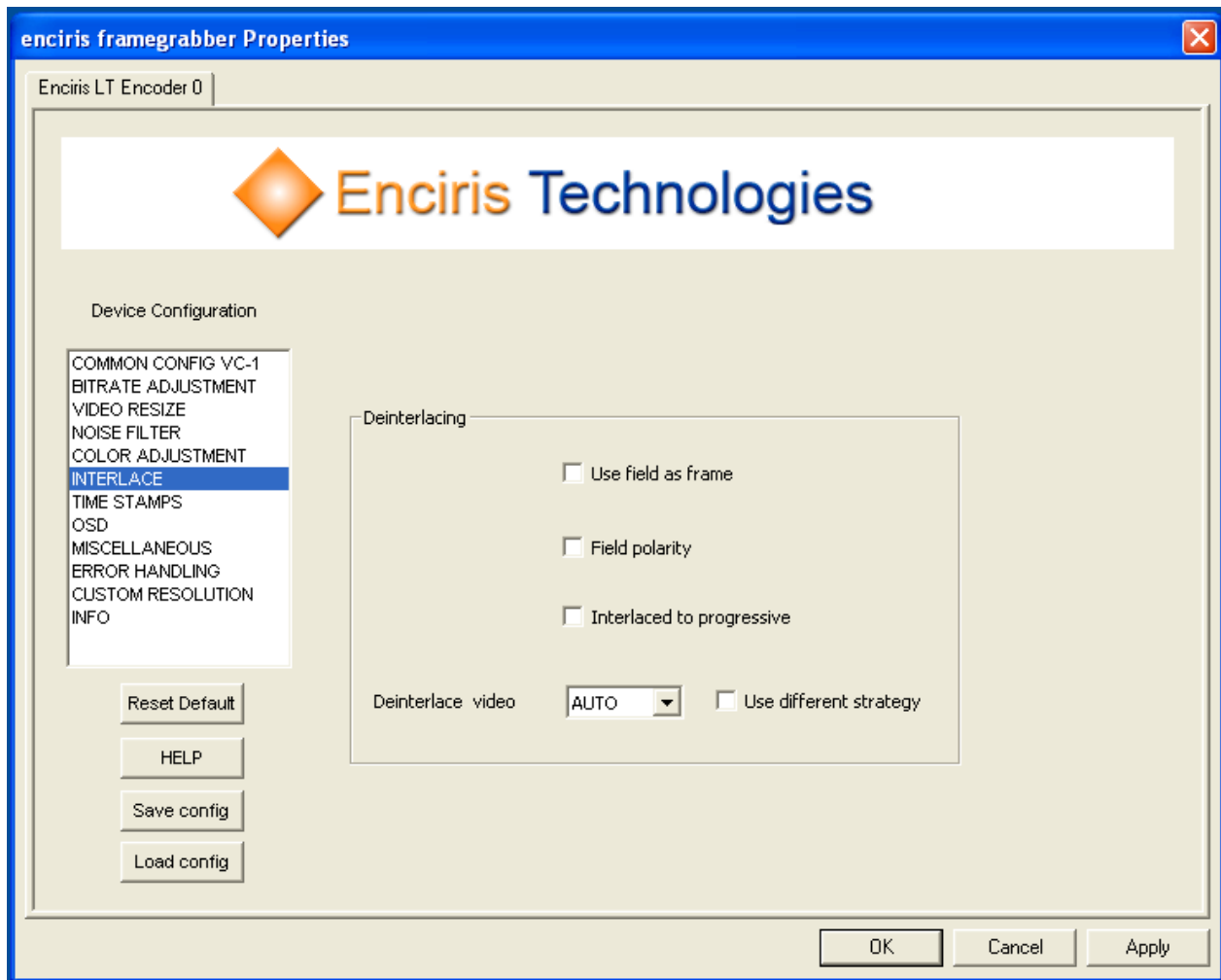
- Noise filter (following values are allowed)
 - NOT USED
 - LOW
 - MEDIUM
 - STRONG

1.5. COLOR ADJUSTMENT



- CBHS (Contrast, Brightness, Hue and Saturation)
 - Regulate image quality. These parameters can be changed at live time (during capture)

1.6. INTERLACE

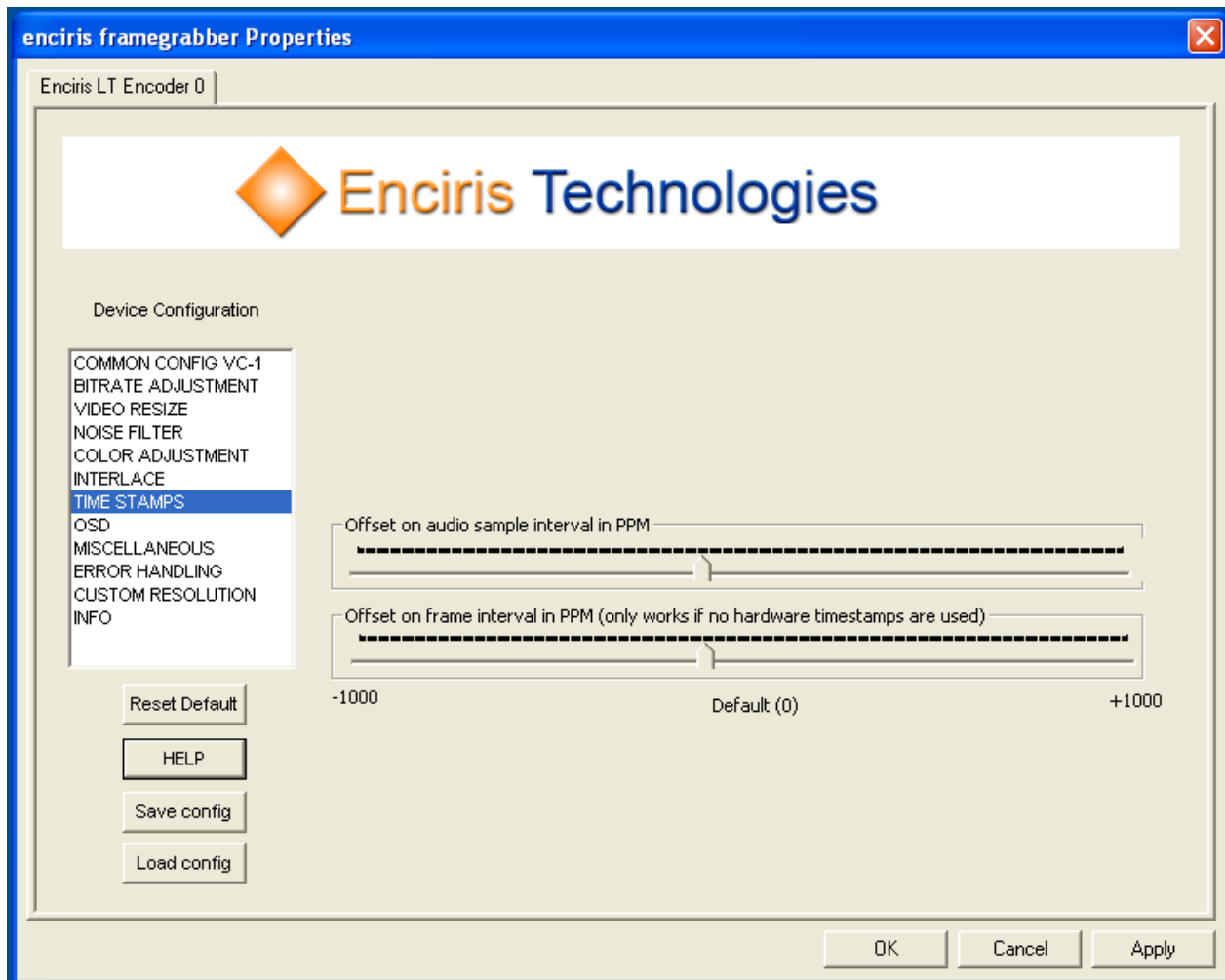


- Use field as frame:
 - Uses each interlaced field as a full frame. Image resolution is reduced but temporal resolution is increased
- Field polarity:
 - If set, top field will be transmitted first, otherwise bottom field is transmitted first.
- Interlaced to progressive:
 - This will double the framerate and perform interpolation for each field.
- Deinterlace video:

- Activates or deactivates the adaptive deinterlacing filter.
- Use different strategy:
 - if set, additional simple motion estimation will be applied.
- NOTE: True interlaced compression as specified in (VC-1/H.264 standard) is not supported at this time.

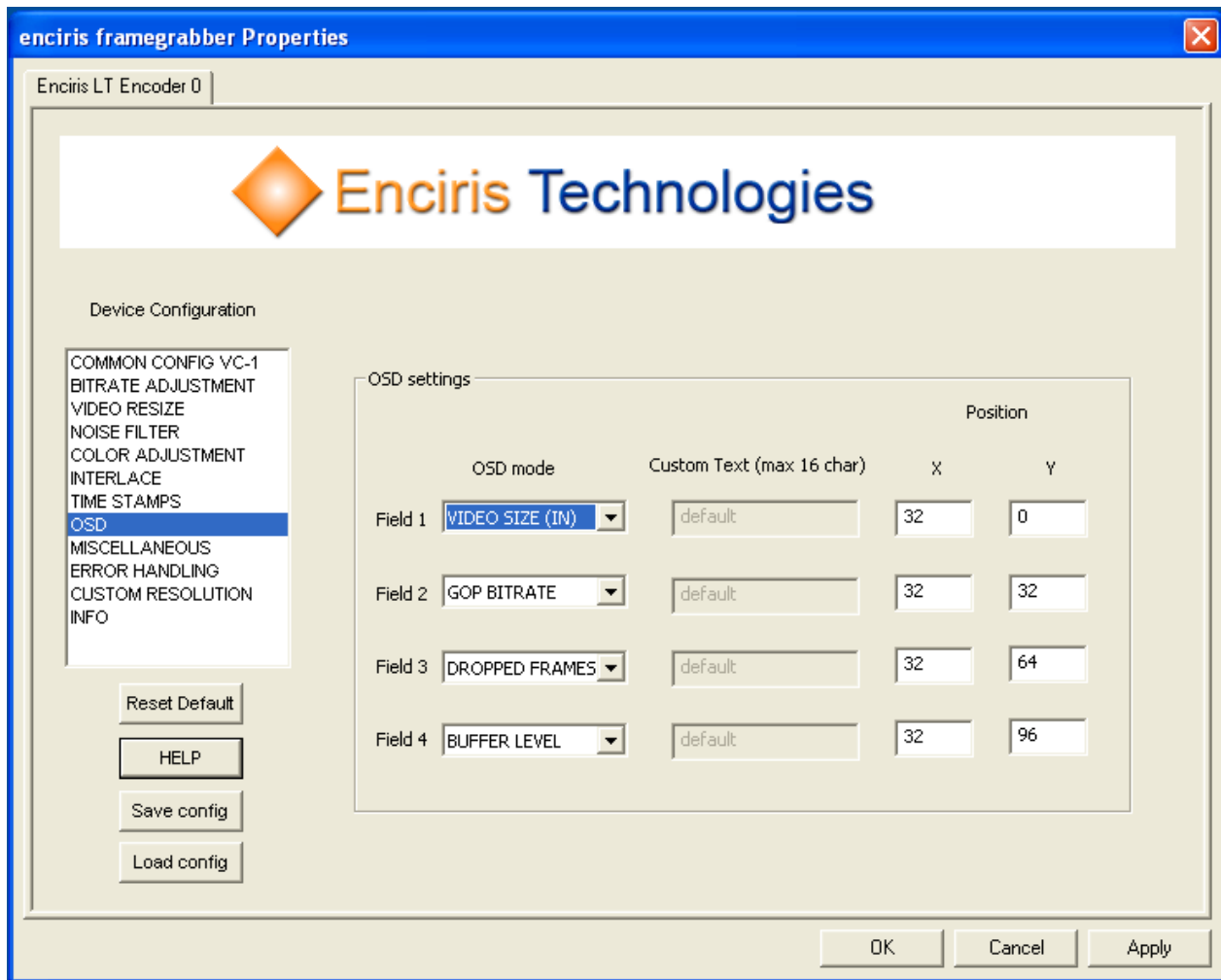
1.7. TIME STAMPS

These parameter could be used in order to fine tune audio/video synchronization.



- Offset on audio sample interval in PPM:
 - Offset on audio sample interval ($1/48000$ Sec) in Parts Per Million. This offset can be negative.
- Offset on frame interval in PPM:
 - Offset on frame interval in Parts Per Million. This offset can be negative.

1.8. OSD (On Screen Display)

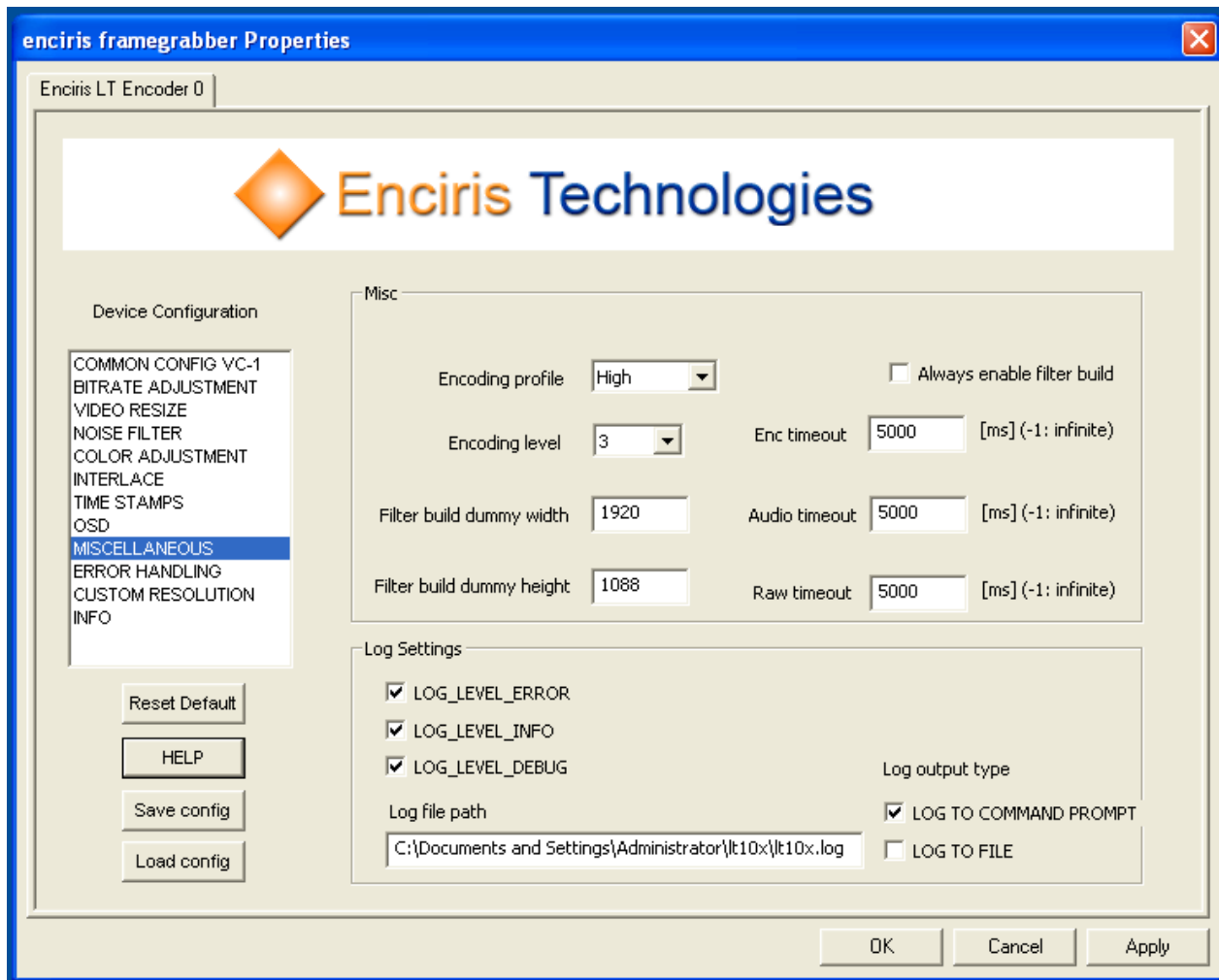


Four OSD fields can be defined and positioned over the video, before it is compressed by specifying their coordinates.

- OSD mode (they are listed as follows):
 - "NOT USED": OSD disabled.
 - "CUSTOM": Custom mode allowed. User can insert custom text with maximum length of 16 characters.
 - "VIDEO SIZE (IN)": Size of the input video before compression.
 - "VIDEO SIZE (OUT)": Size of the output video after compression.
 - "TIME / DATE": Current time and date.
 - "VIDEO BITRATE": Actual video bitrate averaged over one second.

- "GOP BITRATE": Actual video bitrate averaged over a GOP (Group Of Pictures).
- "DROPPED FRAMES": Number of dropped frames at acquisition.
- "BUFFER LEVEL": Hardware internal buffer level.
- Custom Text (max 16 char):
 - Custom text with a maximum of 16 characters.
- Position:
 - x and y position of the OSD characters. This should be a multiple of 16.

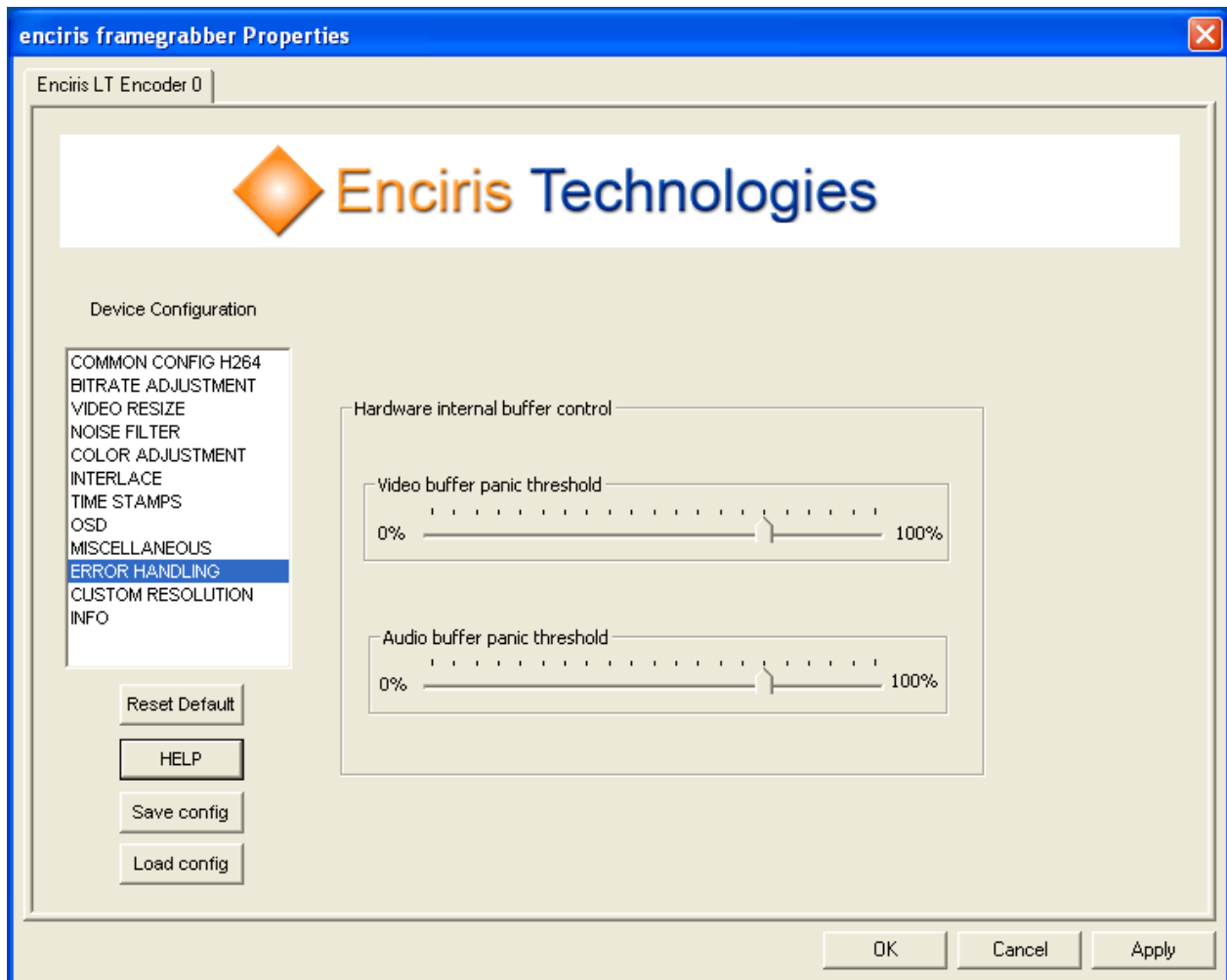
1.9. MISCELLANEOUS



- Encoding profile
 - Specifies the profile. This is High for VC-1 or Baseline, main or High for H.264
- Encoding level
 - VC-1/H.264 level.
- Filter build dummy width/height
 - This is a special mode where directshow filter gets build even if no input video is connected. This mode only works if “Always enable filter build” is checked.
- Always enable filter build
 - Directshow filter gets build even if no actual video source is connected
- Audio/Video/Raw timeout:

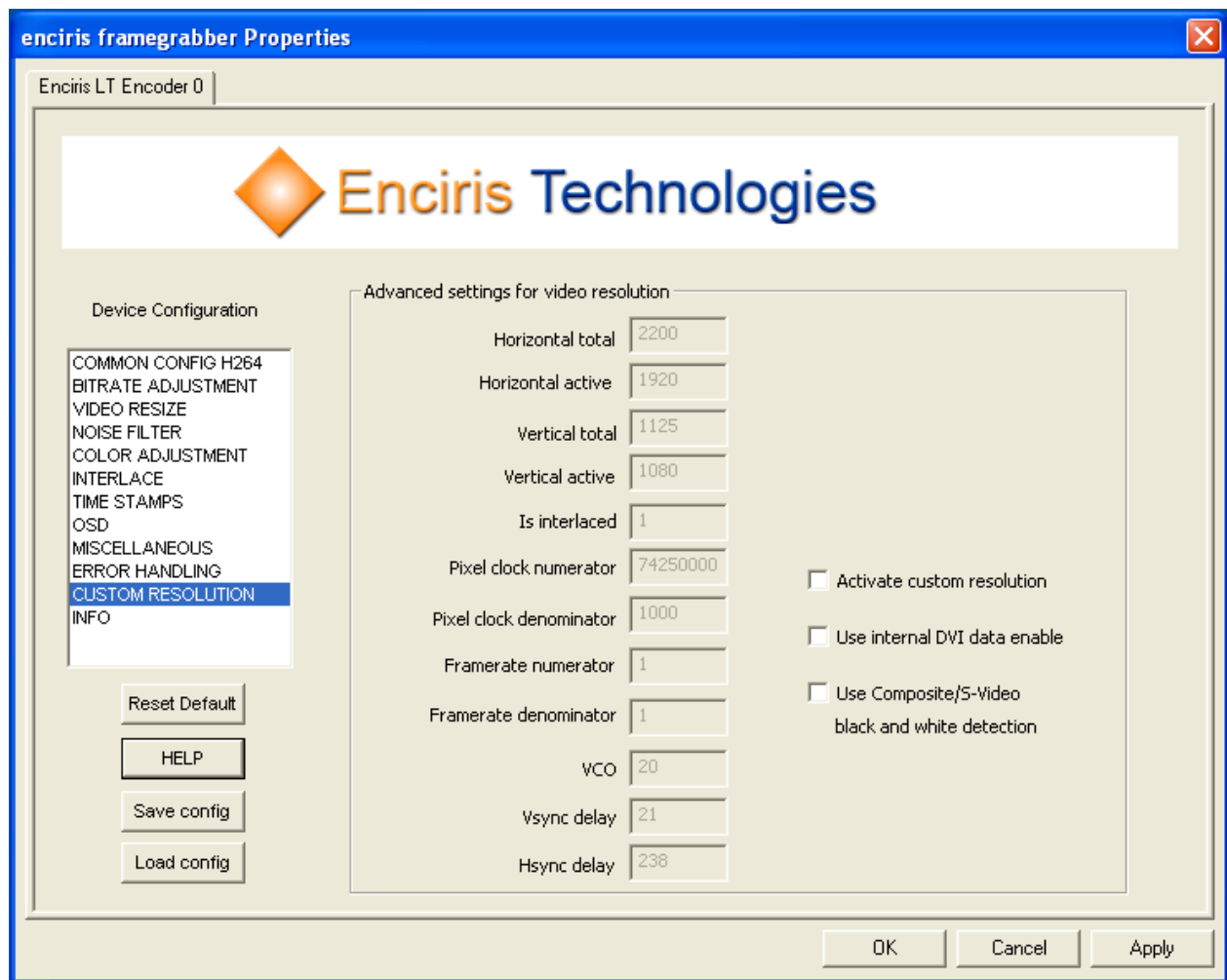
- Timeout if video data is unavailable.
- Log Settings
 - Enables debug logging into a file or on a command prompt

1.10. ERROR HANDLING



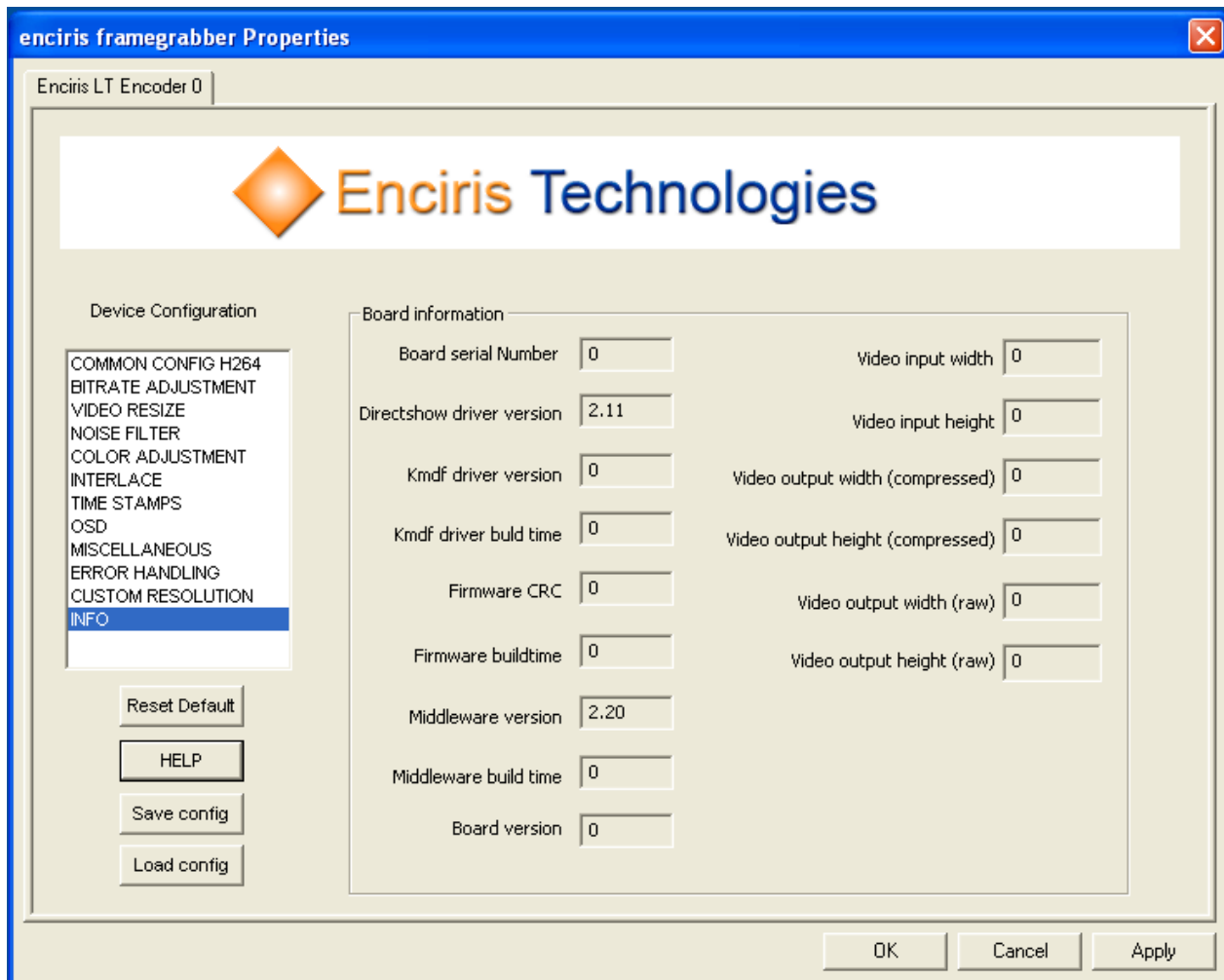
- Video buffer panic threshold:
 - Controls the internal video hardware buffer level. If this buffer overflows, frames will be dropped.
- Audio buffer panic threshold:
 - Controls the internal audio hardware buffer level. If this buffer overflows, audio samples will be dropped.

1.11. CUSTOM RESOLUTION



NOTE: This is a special debug mode only valid for DVI-A (DVI-Analog) input. This mode should normally not be used. Please contact Enciris Technologies for more details.

1.12. INFO



- Board serial number
 - Each board can be identified by its unique serial number written on its PC
- Directshow driver version
 - Version of the directshow source filter
- Kmdf driver version (not yet implemented)
- Kmdf driver build time (not yet implemented)
- Firmware CRC (not yet implemented)
- Firmware build time (not yet implemented)
- Middleware version
 - Version of the underlying middleware dll driver
- Board version (not yet implemented)
- Video input width

- Width of the incoming video
- Video input height
 - Height of the incoming video
- Video output width (compressed)
 - Width of the compressed video
- Video output height (compressed)
 - Height of the compressed video
- Video output width (raw)
 - Width of the uncompressed video
- Video output height (raw)
 - Height of the uncompressed video

5. Available API's in Directshow SDK

The LT-XXX Directshow API uses a custom interface defined in the IRaw.h file. All parameters that can be set in the property page can also be set directly through the custom interface. This interface will be described in the next chapter.

Enciris Technologies has also developed other Directshow filters for VC-1 streaming and recording: Itasfmux.ax (an ASF multiplexer filter) and asfNetworkSink.ax (an HTTP streaming server filter)

IDump.h contains the interface for LT-XXX ASF multiplexer filter and INetWrite.h contains the interface for the HTTP server filter. These together with enciris_lt.h and enciris_ltlb.h are the only files needed to develop a Directshow applications based on Enciris Technologies own Directshow filters.

IRaw.h, IDump.h and INetWrite.h contain GUID of corresponding filters as well as interface methods used to program the LT-XXX board.

Please also refer to the Directshow examples in our SDK package.

The Directshow examples will be placed in the following location after SDK installation (if default path was used during SDK installation):

C:\Program Files\Enciris Technologies\LT-XXX SDK\directX_examples\

Please refer to SDK description document for more details about the SDK.

6.LT-XXX Directshow Filter Custom API

There are only two methods in order to interface with LT-XXX board through Directshow:

1. Property pages as described above
2. LT-XXX Directshow proprietary custom API

Standard Directshow interfaces have been knowingly skipped to avoid undesired interaction with the custom API.

The LT-XXX API has a set of user parameters that can be set to program the LT-XXX hardware.

User parameters are contained in a C++ structure called `pCmd_t`. Not all configuration parameters are passed through this structure. Other parameters are passed through dedicated functions described later in this document. This splitting is done for backward compatibility reasons.

The `pCmd_t` structure used to set/get user data is as follows:

```
typedef struct _cmd_t
{
    int kmdf_drv_ver;
    int kmdf_drv_buildtime;
    int fw_crc;
    int fw_buildtime;
    int dll_version;
    int dll_buildtime;
    int board_version;
    int video_in_width;
    int video_in_height;
    int vcl_width;
    int vcl_height;
    int raw_width;
    int raw_height;
    int target_framerate;
    int profile_and_level;
    int denoise;
    int skip_frame_thr;
    int intra_sad_thr;
    int gop_length;
    int adapt_deint;
    int avg_bitrate;
    int img_res;
    int device_no;
    int use_colorbar;
    int audio_mode;
    int aud_bitrate;
    int squared_pix;
    int video_vshift;
```

```

int video_hshift;
int contrast;
int brightness;
int hue;
int saturation;
int video_16resmode;
int vcl_timeout;
int audio_timeout;
int raw_timeout;
int ppm_fr_interv_offs;
int ppm_samp_interv_offs;
int cust_h_total;
int cust_h_active;
int cust_v_total;
int cust_v_active;
int cust_interlaced;
int cust_p_clocknum;
int cust_p_clockden;
int cust_fr_numerator;
int cust_fr_denominator;
int cust_vco;
int cust_vsync_delay;
int cust_hsync_delay;
int brc_mode;
int audio_input;
int crop_h_start;
int crop_v_start;
int crop_width;
int crop_height;
int board_serial_no;
int dshow_drv_version;
int osd_setting1;
int osd_setting2;
int osd_setting3;
int osd_setting4;
int osd_x1;
int osd_x2;
int osd_x3;
int osd_x4;
int osd_y1;
int osd_y2;
int osd_y3;
int osd_y4;
char osd_text1[32];
char osd_text2[32];
char osd_text3[32];
char osd_text4[32];
int skipframe_enable;
int quality_min;
int quality_max;
int quality_jump;
int quality_speed_up;
int quality_speed_down;
int quality_uniform;
int vb_avg_window;
int vb_deltamin;
int vb_deltamax;
int vb_deltaskip;
int rangemap_luma;
int rangemap_chroma;
int use_field_as_frame;
int field_polarity;
int interlaced2prog;
int video_panic_lev;
int audio_panic_lev;
int raw_scaling_w;

```

```

int raw_scaling_h;
int vcl_scaling_w;
int vcl_scaling_h;
int videoInput;
int videoResolution;
int videoColorspace;
int activate_cust_res;
int internal_dvi_de;
int cust_res_idx;
bool stream_error_flag;
int enable_build;
int dummy_width;
int dummy_height;
int usr_sr_epl;
int reg_is_valid;
int mv_timeout;
int use_exp_mode;
int scaling_raw;
int scaling_comp;
int pq_index;
int aac_quant_qual;
int aac_output_format;
int aac_mpeg_version;
int aac_object_type;
int debug;
int vcl_burst_length;
int asf_enable_asf;
int asf_packet_size;
int asf_preroll;
int asf_use_padding;
double frame_interval;
double sample_interval;
double frame_rate;
}cmd_t, *pCmd_t;

```

The parameters are as follow:

kmdf_drv_ver: (not yet available)

Kernel Mode Driver Foundation version. Not yet implemented.

kmdf_drv_buildtime: (not yet available)

Kernel Mode Driver Foundation version. Not yet implemented.

fw_crc: (not yet available)

Hardware firmware file checksum. Not yet implemented.

fw_buildtime: (not yet available)

Firmware build time. Not yet implemented.

dll_version: (read only)

Underlying middleware driver version. E.g 220 means version 2.20

dll_buildtime: (not yet available)

Underlying middleware driver build time. Not yet implemented.

board_version: (not yet available)

Board version. Not yet implemented.

video_in_width: (read only)

Input video horizontal resolution.

video_in_height: (read only)

Input video vertical resolution.

target_framerate: (default 0)

Video target framerate. Suppose you have PAL video input at 25 images/sec. You can set this parameter e.g. to 10 in order to reduce framerate to 10 pictures per second. Note that this parameter will not guaranty the whished output framerate. A variation of one image/sec may occur. If set to 0, no framerate decimation will be done unless hardware internal framerate limitation is reached.

profile_and_level: (default 3)

Video elementary stream level. The actual profile is transmitted through SetProfile() function. This parameter sets level from 0 to 3 for VC1 and up to level 4.2 for H.264. For H.264 level*10 must be provided (e.g 42 will set level 4.2).

denoise: (default 1)

Noise filter that allows for better compression in case of low bitrate.

0: not used,

1: LOW

2:MEDIUM

3:STRONG

skip_frame_thr: (default 0)

This feature has not yet been implemented for H.264.

Skip frame threshold. Activity threshold beyond what frames will be skipped if this feature is allowed. This feature is not activated yet

intra_sad_thr: (default 2500)

This is only valid for VC-1 compression.

Intra macroblock threshold. Activity threshold beyond which macroblocks will be coded as "INTRA". Good value is 2500. This value has been empirically established

gop_length: (default 120)

Length of Group Of Picture (GOP). E.g. IPPPPPIPPPPPIPP.... has a GOP length of 7. A Good value is 120 (default)

adapt_deint: (default 0)

Content adaptive deinterlacing filter for interlaced video.

0: automatic mode. Deinterlace depending on video input

1: Deinterlacing forced

2:Deinterlacing disabled

avg_bitrate: (default 10000000)

Video average target bitrate in bits/sec

img_res: (default cfg_auto_dvrgb)

Image resolution index defined in i2cldx_t and described as follows:

Video input Connector (Composite/S-Video/DVI etc..), input video color (YUV/RGB) and resolution are set via **img_res** parameter that can have following values:

```

//image resolution index
typedef enum _i2cIdx_t
{
    cfg_dummy0           = 0,
    cfg_dummy1           = 1,
    cfg_1920x1080p299_dvrgb = 2,
    cfg_1920x1080p30_dvrgb = 3,
    cfg_1920x1080p50_dvrgb = 4,
    cfg_1920x1080p60_dvrgb = 5,
    cfg_1920x1080i60_dvrgb = 6,
    cfg_1920x1080i50_dvrgb = 7,
    cfg_1920x1080i60_dvyuv = 8,
    cfg_1920x1080p60_dvyuv = 9,
    cfg_1920x1080i50_dvyuv = 10,
    cfg_1920x1080i60_avrgb = 11,
    cfg_1920x1080i50_avrgb = 12,
    cfg_1920x1080p60_avrgb = 13,
    cfg_1920x1080i50_avyuv = 14,
    cfg_1920x1080i60_avyuv = 15,
    cfg_1920x1080p60_avyuv = 16,
    cfg_dummy2           = 17,
    cfg_dummy3           = 18,
    cfg_1280x1024p75_dvrgb = 19,
    cfg_1280x1024p60_dvrgb = 20,
    cfg_1280x1024p50_dvrgb = 21,
    cfg_1280x1024p75_avrgb = 22,
    cfg_1280x1024p60_avrgb = 23,
    cfg_1280x1024p50_avrgb = 24,
    cfg_dummy4           = 25,
    cfg_dummy5           = 26,
    cfg_1280x720p60_dvrgb = 27,
    cfg_1280x720p60_dvyuv = 28,
    cfg_1280x720p60_avrgb = 29,
    cfg_1280x720p60_avyuv = 30,
    cfg_dummy6           = 31,
    cfg_dummy7           = 32,
    cfg_ntsc_svideo      = 33,
    cfg_ntsc_composite    = 34,
    cfg_pal_svideo       = 35,
    cfg_pal_composite     = 36,
    cfg_dummy8           = 37,
    cfg_dummy9           = 38,
    cfg_720x480p60_dvrgb = 39,
    cfg_dummy10          = 40,
    cfg_dummy11          = 41,
    cfg_auto_dvrgb       = 42,
    cfg_auto_dvyuv       = 43,
    cfg_auto_avrgb       = 44,
    cfg_auto_avyuv       = 45,
    cfg_auto_svideo      = 46,
    cfg_auto_composite    = 47,
    cfg_auto_sdi         = 48,
    cfg_dummy12          = 49,
    cfg_dummy13          = 50,
    cfg_custom_avrgb     = 51,
    cfg_custom_avyuv     = 52,
    cfg_custom_dvyuv     = 53,
    cfg_custom_dvrgb     = 54
} i2cIdx_t, *pI2cIdx_t;

```

Note that input video resolution **auto-detection** can be activated using:

- `cfg_auto_dvrgb`

- o DVI digital RGB
- cfg_auto_dvyuv
 - o DVI digital YUV
- cfg_auto_avrgb
 - o DVI analog RGB
- cfg_auto_avyuv
 - o DVI analog YUV
- cfg_auto_svideo
 - o s-video input
- cfg_auto_composite
 - o composite input
- cfg_auto_sdi
 - o SD-SDI/HD-SDI input

Please also refer to the SDK for usage examples of these parameters.

device_no: (default 0)

LT-XXX device number (0 to 32). Up to 32 LT-XXX framegrabber boards can be plugged (pci/usb) into the same PC

use_colorbar: (default 0)

Use colorbar flag. Sets colorbar mode for debugging. This mode allows to grab for video even if no video input is connected. 0: not used, 1: used

audio_mode: (default 1)

Audio format:

0: None

1: PCM 16 bit stereo at 48Khz

2: WMA8 at 48Khz and variable bitrate.

aud_bitrate: (default 384000)

bitrate / channel of audio pin. Expressed in bits/sec. This is only used if WMA8 compression is used.

squared_pix: (default 0)

Use squared pixel in case PAL is used at Composite or S-Video input.

The output resolution will be 768x576

video_vshift: (default 0)

These key allow for the shifting of the image position within the active window. 0 to 2048 (default: 0) Vertical shift. only for analog input through DVI-A

video_hshift: (default 0)

These key allow for the shifting of the image position within the active window. 0 to 2048 (default: 0) Horizontal shift. Only for analog input through DVI-A

Contrast: (default 128)

These key is used to adjust the contrast for the respective color components for DVI input. 0 to 255 (default: 128)

Brightness: (default 0)

These key is used to adjust the brightness or offset for the respective color components for DVI input. -128 to 127 (default: 0)

Hue: (default 0)

These key is used to adjust the video Hue. -30 to 30, (default: 0)

Saturation: (default 128)

These key is used to adjust the video saturation. -128 to 127, (default: 0)

video_16resmode: (default 0)

Chooses between 1072 or to 1088 for 1080 rounding modes (see also Windows API doc for more details).

In video compression, resolutions are always a multiple of 16pixels because of macroblock organization.

1080 is not a multiple of 16 and must therefore be rounded up or down. "video_16resmode" allows for rounding.

0: (default) rounds down, 1: rounds up

vc1_timeout: (default 5000)

Sets the time-out of a VC-1/H.264 video input in millisecond after which the acquisition returns an error. -1 sets infinite time, 0 to $2^{31}-1$ sets timeout in milliseconds

mv_timeout: (default 5000)

This parameter is deprecated.

audio_timeout: (default 5000)

Sets the time-out of an audio input in millisecond after which the acquisition returns an error. -1 sets infinite time, 0 to $2^{31}-1$ sets timeout in milliseconds

raw_timeout: (default 5000)

Sets the time-out of an uncompressed video input in millisecond after which the acquisition returns an error. -1 sets infinite time, 0 to $2^{31}-1$ sets timeout in milliseconds

ppm_fr_interval_offset: (default -1000)

Integer offset in PPM (part per million) to frame interval. This parameter can be used to fine-tune audio/video synchronization.

cust_h_total: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_h_active: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_v_total: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_v_active: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_interlaced: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_p_clocknum: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_p_clockden: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_fr_numerator: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_fr_denominator: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_vco: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_vsync_delay: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

cust_hsync_delay: (This is a special custom mode. Contact tech support for more details)

This can only be used if `cfg_custom_avrgb` or `cfg_custom_avyuv` are selected.

brc_mode: (default 0)

Bitrate control mode. Following settings are allowed:

0: BRC_ABR_WIDE.

For recording application. There can be big jump around the targeted bitrate.

1: BRC_ABR_MEDIUM

For recording application. There can be medium jump around the targeted bitrate.

2: BRC_VBR

For networking application. The maximum bitrate should not exceed target bitrate by more than 10%, unless target bitrate is too low.
Forced skip frame may occur.

2: BRC_CONSTANTQ

This is a constant Quality (or constant Quantizer) mode. Use **pq_index** to set wished constant quality.

0: automatically set within lower level driver

1: best quality

31:worse quality

audio_input: (default 0)

Audio input selection:

0: analog input

1: Digital embedded input through DVI or HD-SDI

crop_h_start: (default 0)

Horizontal cropping. Defines horizontal starting point of image

crop_v_start: (default 0)

Vertical cropping. Defines vertical starting point of image

crop_width: (default 0)

Defines width of the image section (must be lower than original width)

crop_height: (default 0)

Defines height of the image section (must be lower than original height)

board_serial_no: (Read only)

Board unique serial number. This is a read only parameter

dshow_drv_version: (Read only)

Directshow driver version.

osd_setting1-4:

Chose up to 4 different On Screen Display tags (OSD tags).

- 0: OSD disabled
- 1: custom 16character text to be provided in “osd_textX[32]”. Please refer to “osd_textX[32]” bellow
- 2: Print video input resolution
- 3: Print video output resolution
- 4: print the capture time
- 5: Print the video bitrate (Audio bitrate is not added)
- 7: Print the amount of dropped frames since the beginning of capture
- 8: Print hardware internal buffer fullness in units of 2048 bytes.

osd_x1-4:

Horizontal position of the OSD tags.

osd_y1-4:

Vertical position of the OSD tags

char osd_text1-4[32]:

OSD custom text. 16 characters can be set at maximum. This is only valid if corresponding osd_setting has been set to 1.

skipframe_enable: (default 0)

This is only valid for VC-1 compression. If set, skip frames may be present in the compressed stream.

quality_min: (default 0)

Sets the minimum quantization parameter to be used. If set to 0 automatic mode will be used. The quantizer varies from 1 to 31 for VC-1 and from 0 to 51 for H.264.

quality_max: (default 0)

Sets the maximum quantization parameter to be used. If set to 0 automatic mode will be used. The quantizer varies from 1 to 31 for VC-1 and from 0 to 51 for H.264.

quality_jump: (default 0)

This parameter is deprecated and should not be used.

quality_speed_up: (default 0)

This parameter is deprecated and should not be used.

quality_speed_down: (default 0)

This parameter is deprecated and should not be used.

quality_uniform: (default 0)

This parameter is deprecated and should not be used.

vb_avg_window: (default 0)

0 to 15: Bitrate control Buffer sliding window size. This affects the reaction time of the bitrate control algorithm

vb_deltamin: (default 0)

-32768 to +32767: Virtual buffer control parameter

vb_deltamax: (default 0)

-32768 to +32767: Virtual buffer control parameter

vb_deltaskip: (default 0)

-32768 to +32767: Virtual buffer control parameter that decides when to insert skip frames into the VC-1 stream

rangemap_luma, rangemap_chroma: (default -1)

Range map parameter as specified in VC-1 decoding standard (SMPTE 421M-2006).

use_field_as_frame: (default 0)

If set to 1, each interlaced field is used as a full frame. Image resolution is reduced but temporal resolution is increased

field_polarity: (default 0)

If set to 1, top field will be transmitted first, otherwise bottom field is transmitted first.

interlaced2prog: (default 0)

If set to 1, this will double the framerate and perform interpolation for each field.

video_panic_lev: (default 75)

This parameter set the video panic threshold in terms of percentage of input buffer fullness. If achieved, input video/audio samples will be discarded at acquisition.

audio_panic_lev: (default 75)

This parameter set the audio panic threshold in terms of percentage of input buffer fullness. If achieved, input video/audio samples will be discarded at acquisition.

raw_scaling_w, raw_scaling_h: (default 0. disabled)

Enables and programs generic video downscaling parameters for the uncompressed video. Note that only values below original values can be set. No upscaling is feasible yet.

Can be set up to input video width.

A maximum value of 64 is accepted.

Needs to be a multiple of 16.

Parameter is rounded to nearest valid value.

vc1_scaling_w, vc1_scaling_h: (default 0. disabled)

Enables and programs generic video downscaling parameters for the compressed video. Note that only values below original values can be set. No upscaling is feasible yet.

Up to input video width.

A maximum value of 128 is accepted.

Need to be a multiple of 16

Parameter is rounded to nearest valid value

videoInput:

This parameter is deprecated and should not be used. It is not connected internally.

videoResolution:

This parameter is deprecated and should not be used. It is not connected internally.

videoColorspace:

This parameter is deprecated and should not be used. It is not connected internally.

activate_cust_res:

This parameter is deprecated and should not be used.

internal_dvi_de:

Use internal DVI-D data enable flag instead of the one provided by A/D chip.

This is an advanced setting and should only be used by expert users.

cust_res_idx:

This parameter is deprecated and should not be used.

stream_error_flag:

This parameter is deprecated and should not be used.

enable_build:

This parameter is deprecated and should not be used.

dummy_width:

This parameter is deprecated and should not be used.

dummy_height:

This parameter is deprecated and should not be used.

usr_sr_epl:

This parameter is deprecated and should not be used.

reg_is_valid:

This interface is deprecated and should not be used.

mv_timeout:

This interface is deprecated and should not be used.

use_exp_mode:

This interface is deprecated and should not be used.

scaling_raw:

This interface is deprecated and should not be used.

scaling_comp:

This interface is deprecated and should not be used.

pq_index:

This interface is deprecated and should not be used.

aac_quant_qual: (available upon demand)

This parameter is deprecated and should not be used.

aac_output_format: (available upon demand)

This parameter is deprecated and should not be used.

aac_mpeg_version: (available upon demand)

This parameter is deprecated and should not be used.

aac_object_type: (available upon demand)

This parameter is deprecated and should not be used.

debug:

This parameter is deprecated and should not be used.

vc1_burst_length:

This parameter is deprecated and should not be used.

asf_enable_asf:

This parameter is deprecated and should not be used.

asf_packet_size:

This parameter is deprecated and should not be used.

asf_preroll:

This parameter is deprecated and should not be used.

asf_use_padding:

This parameter is deprecated and should not be used.

frame_interval: (read only)

Frame interval between two images expressed in 100 nanosecond units

sample_interval: (read only)

Audio sample interval expressed in 100 nanosecond units

frame_rate: (read only)

Video framerate

Following methods are used to set and get user parameters that are contained in `pCmd_t`:

1. `STDMETHOD (SetCommands) (pCmd_t pCmd);`
 - a. This interface will set user parameters
2. `STDMETHOD (GetCommands) (pCmd_t pCmd);`
 - a. This interface will retrieve user parameters
3. `STDMETHOD (SetLiveParams) (pCmd_t pCmd);`
 - a. This interface will set OSD and BCHS live parameters: Brightness, Contrast, Hue, saturation. These values are the only values that can be set during recording or playback.
4. `STDMETHOD (ResetCommands) (pCmd_t pCmd);`
 - a. This interface will reset user parameters to their default parameters

Further helper functions can be used to configure Directshow source filter.

- `STDMETHOD (IsDeviceConnected) (int *is_connected, int dev_no);`

This interface will retrieve device connected status

Is_connected: (output)

Connected status). 1: device is connected.

dev_no: (input)

Index of the device we want to check)

- `STDMETHOD (ForceSkipFrames) (int on);`

This interface will insert Skipped frames until `ForceSkipFrames (0)` is called. This is only valid for VC-1 compression.

On: (input)

1: insert skip frames. 0: stop inserting skip frames

- `STDMETHOD (GetVideoInputStatus) (THIS_ int VideoInputType, int * pVideoInputStatus);`

This is a helper function that returns a detailed status of the connected video sources.

This function can also be used in a separate thread. Don't call it too frequently (200ms interval should be plenty)

VideoInputType: (input)

Video input type.

VideoInputType can be any of the following values (to be retrieved from **enciris_lt.h**):

VID_BOOTSCREEN¹

VID_SDI

VID_SDI1 (same than VID_SDI)

VID_SDI2²

VID_HD_DIGITAL

VID_HD_DIGITAL1(same as VID_HD_DIGITAL

VID_HD_DIGITAL2³

VID_HD_ANALOG

VID_HD_ANALOG1(same as VID_HD_ANALOG)

VID_HD_ANALOG2⁴

VID_COMPOSITE

VID_COMPOSITE1(same as VID_COMPOSITE)

VID_COMPOSITE2⁵

VID_SVIDEO

²Only valid for LT-122

³Only valid for LT-124 and LT-125

⁴ Only valid for custom board

⁵ Only valid for custom board

pVideoInputStatus (out)

This is a 5 bit mask where each bit represents a particular video status.

The returned "pVideoInputStatus" 5 bit mask is decoded as follows (refer to **enciris_lt.h**):

```
#define VIDSTATUS_NULL (0)
```

```
#define VIDSTATUS_INPUT_AVAILABLE (1 << 0)
```

```
#define VIDSTATUS_INPUT_ACTIVE      (1 << 1)
#define VIDSTATUS_SIGNAL_PRESENT    (1 << 2)
#define VIDSTATUS_SIGNAL_STABLE     (1 << 3)
#define VIDSTATUS_SIGNAL_CHANGED    (1 << 4)
```

Where:

VIDSTATUS_INPUT_AVAILABLE :

Video input exists and is available (Example : If DVI-D active in others threads/channels DVI-A become unavailable)

VIDSTATUS_INPUT_ACTIVE :

The video input is currently used in other thread/channel.

VIDSTATUS_SIGNAL_PRESENT :

Video is present on selected input.

VIDSTATUS_SIGNAL_STABLE :

Video is stable on selected input.

VIDSTATUS_SIGNAL_CHANGED :

Video resolution change between now and the last video input configuration.

- **STDMETHOD (InsertIFrame) () ;**

This interface will insert a key frames into the compressed stream. This is important when the compressed video stream is split into segments. Each segment must start with a key frame.

During a record for example you may want to close current file and create a new one seamlessly. You need a Key Frame (also sometime referred to as Intra-Frame) to begin this new file. Use this function to force next frame to be a Key Frame.

This function can also be used to recover after a stream error knowing that key frames do not need any backward reference frame in time in order to be decoded.

Note: There can be a delay of one or two frames according to the target bitrate and the time when function call occur.

- **STDMETHOD (LoadRegistryData) () ;**

This interface will reconfigure the board with the configuration parameters stored in Windows registry.

- **STDMETHOD (SetInitLTDllLog)** (int Mask, char *FilePath, void *FilePointer);

This interface will initialize the debug log mechanism. User can specify log file location and content accuracy.

It is recommended to use LOG_LEVEL_ALL in case you request tech support from Enciris Technologies in order to provide as much information as possible.

Note:

LOG_LEVEL_NULL will disables all logs

LOG_LEVEL_ALL will enables all logs on standard output. This is the default configuration (same as | LOG_LEVEL_INFO | LOG_LEVEL_DEBUG | LOG_LEVEL_ERROR).

Mask: (input). Default: (LOG_TYPE_PRINTF | LOG_LEVEL_INFO | LOG_LEVEL_DEBUG | LOG_LEVEL_ERROR)

Sets 16bit log location and log level mask. (LOG_TYPE_PRINTF, LOG_TYPE_FILE, LOG_LEVEL_NULL, LOG_LEVEL_INFO, LOG_LEVEL_DEBUG, LOG_LEVEL_ERROR and LOG_LEVEL_ALL) are valid values (refer to **enciris_it.h**):. One can “OR” between several values.

FilePath: (input). Default: “Current user profile path\lt10x\lt10x.log”

Path to the file containing the log if file destination has been chosen

FilePointer: (input)

This is a reserved parameter that should be set to NULL.

- **STDMETHOD (GetInitLTDllLog)** (int *pMask, char *FilePath, void *FilePointer);

This interface will retrieve the debug log information.

pMask: (output).

Gets log location and log level mask.

FilePath: (output)

Path to the file containing the log if file destination has been chosen

FilePointer: (output)

This is a reserved parameter.

- **STDMETHOD (SetChannelType) (int ChannelType);**

This interface becomes important when user wants to use board resources in different applications/threads. User can e.g reserve one resource/process freeing remainder resources for other processes.

ChannelType: (inout). Default: CHANNEL_TYPE_ALL

Valid values are:

CHANNEL_TYPE_NULL nothing to do

CHANNEL_TYPE_ENC VC-1 or H.264 video

CHANNEL_TYPE_RAW uncompressed video

CHANNEL_TYPE_AUD audio

CHANNEL_TYPE_ALL all data types (default)

- **STDMETHOD (GetLastError) (int *pLastError);**

This interface will retrieve the last error occurred. The error is already reported in the log file specified in **SetInitLTD11Log()**.

In order to get the right error message, please refer to the error codes in **enciris_lt.h**

pLastError: (output).

Last error.

- **STDMETHOD (SetDeintStrategy) (int deint_strategy);**

Switch between a simple and a more advanced deinterlacing algorithms

deint_strategy: (input). Default: 0

0 Median based (default)

1 Median and simple Motion Estimation mix

- **STDMETHOD (GetDeintStrategy) (int *deint_strategy);**
read deinterlacing strategy parameter back.

deint_strategy: (output).

0 Median based (default)

1 Median and simple Motion Estimation mix

- **STDMETHOD (SetProfile) (int profile);**

H.264 profile indication

For VC-1 encoder advanced profile is always used.

profile: (input). Default: 0 (VC-1), 1(H.264)

0: H.264 Baseline

1: H.264 Main

3: H.264 Advanced (default)

- **STDMETHOD (GetProfile) (int *p_profile);**
read profile parameter back.

P_profile: (output).

0: H.264 Baseline

1: H.264 Main

3: H.264 Advanced (default)

- **STDMETHOD (SetSdbwMode) (int sdbwMode);**

Use black and white detection mode on standard video input

Allows for black and white input signal types.

sdbwMode: (input). Default: 0

0: off

1: on

- `STDMETHOD(GetSdbwMode) (int *sdbwMode);`

read sdbwMode parameter back.

sdbwMode: (output).

0: off

1: on

- `STDMETHOD(SetColorShift) (int colorshift);`

Interchange U and V chroma components in the YUV input video color space. This can be used to achieve special visual effects.

colorshift: (input). Default: 0

0: off

1: on

- `STDMETHOD(GetColorShift) (int *colorshift);`

read colorshift parameter back.

colorshift: (output).

0: off

1: on

- `STDMETHOD(GetStillImageUYVY) (unsigned char *p_still);`

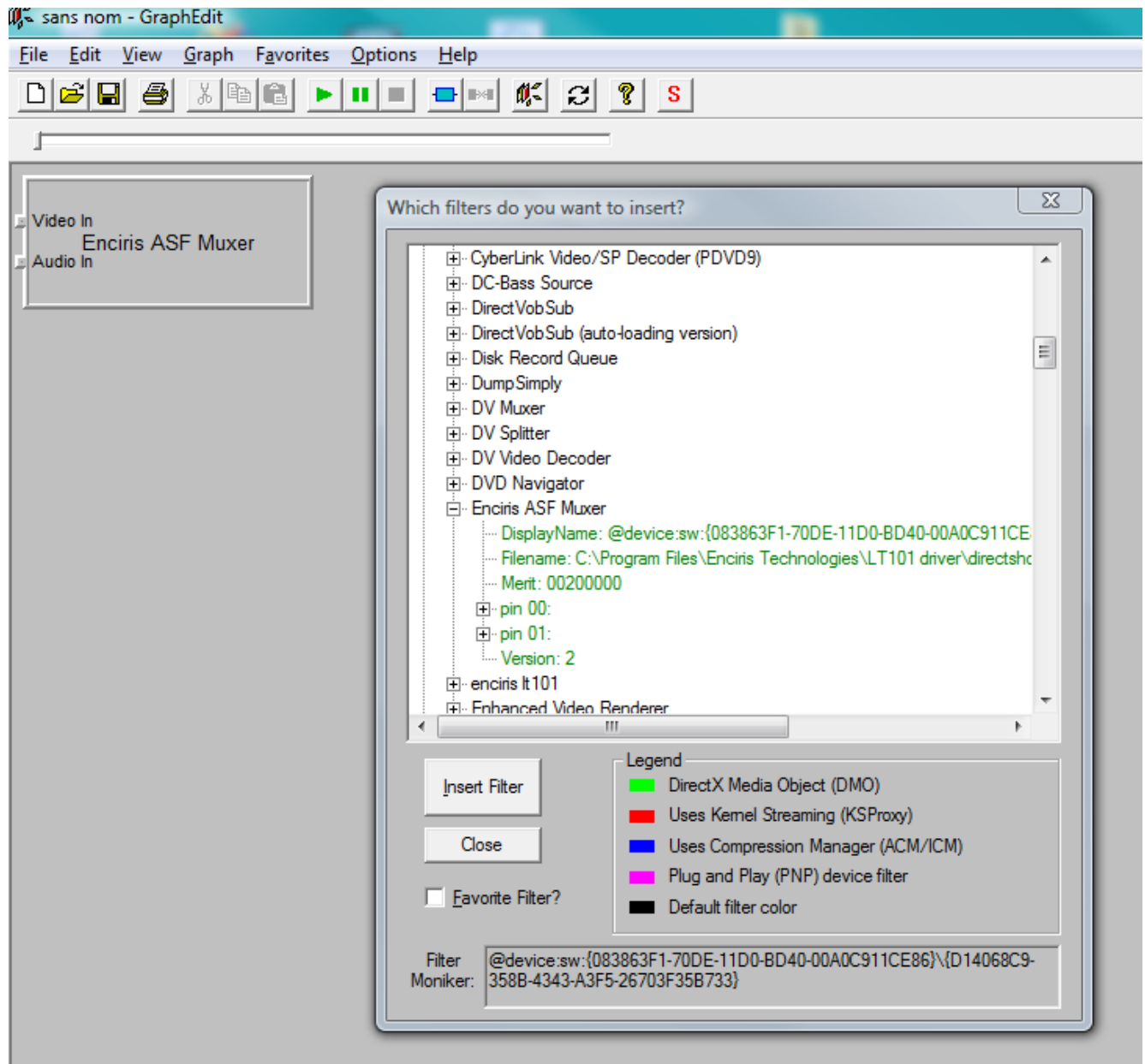
This function returns a still image in raw YUV format. The FOURCC used is UYVY. User must provide a properly allocated buffer of size `image_width * image_height * 2`.

P_still: (output). User provided pointer to still image capture

Enciris can provide code example for BMP, JPEG etc. conversion upon demand.

7.ASF Recording Filter Description (ltasfmux.ax) (available for VC-1 compression only)

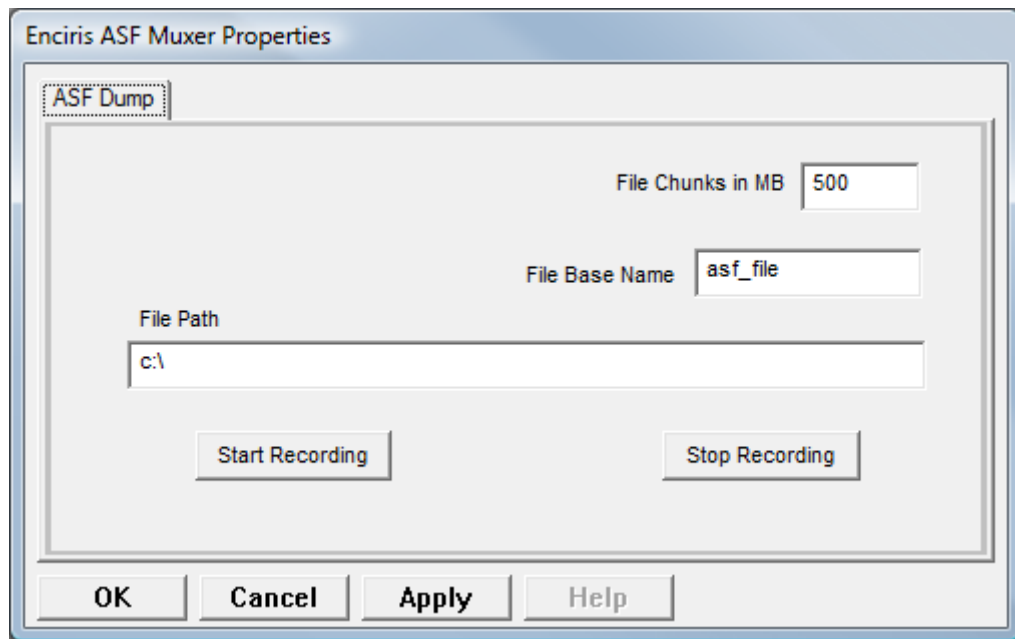
LT-XXX driver provides an ASF multiplexer and dump filter named ltasfmux.ax. This filter builds on WMSDK (Windows Media SDK). It is referred as “Enciris ASF Muxer” as can be seen in picture bellow.



“Enciris ASF Muxer” Filter

ltasfmux.ax can be used to record captured video to hard disk in versatile ASF format. Playback is possible through almost any media player (e.g. WMP11, VLC...).

This filter exposes following user configurable property page:



“Enciris ASF Muxer” Filter Property Page

The filter can be configured to record ASF files of variable chunk size at any user defined place.

A base-name is provided and auto file-split will be performed. Resulting recording folder will e.g. contain following files:

asf_file_0000000.asf

asf_file_0000001.asf

...

User can start and stop recording independently from stream start and stop.

Please refer to LT-XXX SDK for C++ usage examples of Itasfmux.ax.

8.ASF Recording Filter API (ltasfmux.ax) (available for VC-1 compression only)

This interface allows for ASF audio/video multiplexing and file recording using WMSDK (Windows Media SDK). Please also refer to “LT-XXX ASF Multiplexer Filter Description” section below.

Three methods are used to set and get the user parameters:

```
DECLARE_INTERFACE_(IDumpAsf, IUnknown)
{
    STDMETHOD(get_UserParams) (THIS_
        int *fileSize,
        int *packetSize,
        char *baseName,
        char *filePath
    ) PURE;

    STDMETHOD(set_UserParams) (THIS_
        int fileSize,
        int packetSize,
        char *baseName,
        char *filePath
    ) PURE;

    STDMETHOD(enable_Run) (THIS_
        bool is_run
    ) PURE;
};
```

The methods are detailed as follows:

1. STDMETHOD(set_UserParams) (...) PURE;
 - a. This will set user parameters
2. STDMETHOD(get_UserParams) (...) PURE;
 - a. This will retrieve user parameters
3. STDMETHOD(enable_Run) (...) PURE;
 - a. This will start and stop recording while video is being streamed

The function parameters are as follows:

fileSize:

Desired file chunk size in which the stream should be splitted.

packetSize: (This parameter is obsolete)

ASF packet size

baseName:

File base name

filePath:

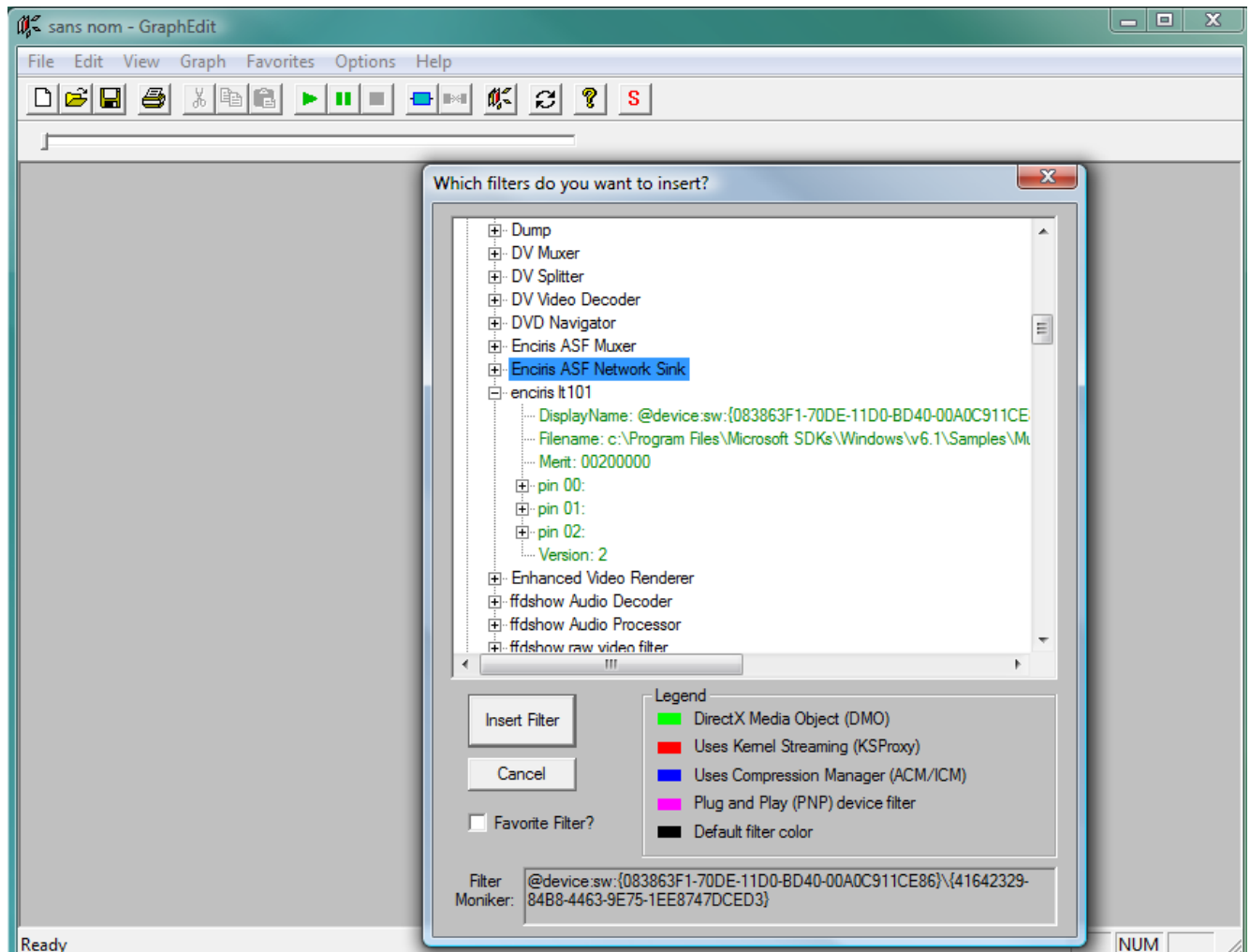
Absolute path of the recorded files

is_run:

Start/stop flag

9.Streaming Server Filter Description (asfNetworkSink.ax). (available for VC-1 compression only)

LT-XXX driver provides an HTTP streaming server filter named asfNetworkSink.ax. This filter builds on WMSDK (Windows Media SDK). It is referred as “Enciris ASF Network Sink” as can be seen in picture bellow.



“Enciris ASF Network Sink” Filter

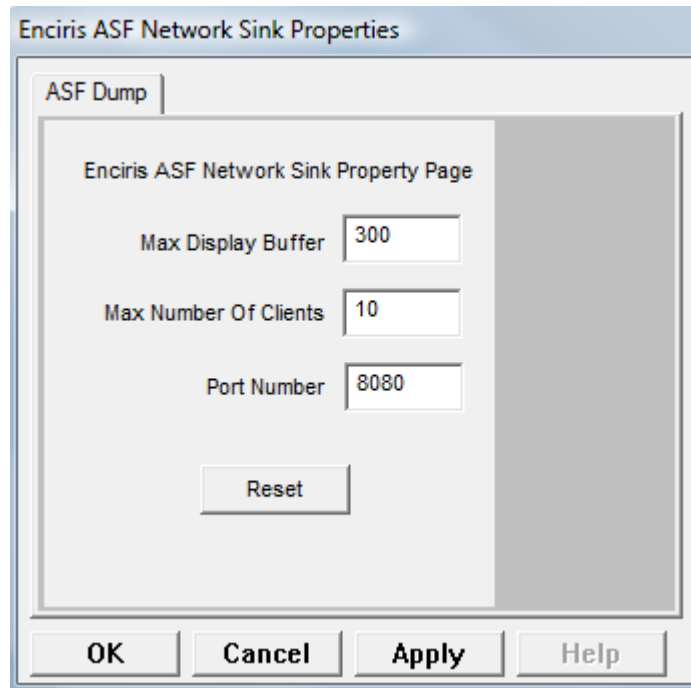
asfNetworkSink.ax can be used to send in an ASF audio/video stream over network. Playback is possible through Windows Media player version 11 (WMP11). When playing back in WMP11, choose file->open url and enter following sample URL:

<http://192.168.1.50:8080>

In this case 192.168.1.50 is the IP address of the server (you can use “localhost” if you use server and client on the same machine). 8080 is the port number of the service.

This filter is highly inspired from WMSDK’s WMVNetWrite example.

The filter exposes following user configurable property page:



“Enciris ASF Network Sink” Filter Property Page

The filter can be configured to send to a **maximum number of clients**.

A **Maximum Display buffer** size can be provided in Milliseconds.

A **Port number** must also be provided for the service.

All values are stored in windows registry.

Please refer to LT-XXX SDK for C++ usage examples of asfNetworkSink.ax.

10. Streaming Server Filter API (asfNetworkSink.ax) (available for VC-1 compression only)

This interface allows for ASF audio/video multiplexing and streaming over HTTP using WMSDK (Windows Media SDK). Please also refer to Microsoft's WMVNetWrite example from which the code is inspired.

Two methods are used to set and get the user parameters:

```
DECLARE_INTERFACE_(INetWrite, IUnknown)
{
    STDMETHOD(get_UserParams) (THIS_
        int *bufferSize,
        int *maxNoClients,
        int *portNumber
    ) PURE;

    STDMETHOD(set_UserParams) (THIS_
        int bufferSize,
        int maxNoClients,
        int portNumber
    ) PURE;
};
```

The methods are detailed as follows:

1. `STDMETHOD(set_UserParams) (...);`
 - a. This will set user parameters
2. `STDMETHOD(get_UserParams) (...);`
 - a. This will retrieve user parameters

The function parameters are as follows:

bufferSize:

This is an `IWMStreamConfig` interface parameter from Windows Media SDK (WMSDK). It specifies the maximum latency between when a stream is received and when it begins to be displayed (expressed in milliseconds). Default is 3000 ms

maxNoClements:

Maximal number of clients. Default is 10

portNumber:

Streaming port number. Default is 8080

Clients connect to session using e.g following URL: <http://192.168.1.50:8080> where 192.168.1.50 is the servers IP address.

11. C++ Examples Applications:

There are tools (like GraphEditPlus) that automatically generates source code from a filtergraph. These kinds of tools will accelerate development of custom filters.

Nevertheless, following example source code shows a basic Directshow application that used It_framegrabber.ax driver API. The examples in this document are only for academic purposes and may not be 100% up to date. Please refer to our official SDK in order to get current examples.

This example corresponds to “custom_renderer” example in SDK:

C:\Program Files\Enciris Technologies\LT-XXX\SDK\directX_examples\samples\custom_renderer

Note that this path is the default installation path.

```
// LT-XXX_renderer.cpp : Defines the entry point for the console application.
//
#include "stdafx.h"
#include <dshow.h>
#include "../api/IRaw.h"

// {37555C9E-28CC-4bd8-84FA-9FE011DDEF41}
DEFINE_GUID(CLSID_LT-XXXFrameGrabber,
0x37555c9e, 0x28cc, 0x4bd8, 0x84, 0xfa, 0x9f, 0xe0, 0x11, 0xdd, 0xef, 0x41);

IBaseFilter *GetVideoDevice ()
{
    ICreateDevEnum *pDevEnum = NULL; // Create the system device enumerator.
    IEnumMoniker *pClassEnum = NULL; // Create an enumerator for cap. dev
    IBaseFilter *pSrc = NULL;
    IMoniker *pMoniker = NULL;
    LPCWSTR sTemp;
    VARIANT varName;
    HRESULT hr;

    LPCWSTR deviceFriendlyName = L"enciris LT-XXX";

    CoCreateInstance(CLSID_SystemDeviceEnum, NULL, CLSCTX_INPROC,
        IID_ICreateDevEnum, (void **) &pDevEnum);

    pDevEnum->CreateClassEnumerator(CLSID_VideoInputDeviceCategory, &pClassEnum,
0);

    while (pClassEnum->Next(1, &pMoniker, NULL) == S_OK)
    {
        IPropertyBag *pPropBag;
        hr = pMoniker->BindToStorage(0, 0, IID_IPropertyBag,
            (void **) (&pPropBag));
        if (FAILED(hr))
        {
            pMoniker->Release();
            continue; // Skip this one, maybe the next one will work.
        }
        // Find the description or friendly name.
        VariantInit(&varName);
        hr = pPropBag->Read(L"Description", &varName, 0);
        if (FAILED(hr))
        {
            hr = pPropBag->Read(L"FriendlyName", &varName, 0);
        }
    }
}
```

```

    }
    sTemp = (LPCWSTR) varName.bstrVal;
    if(_wcsnicmp(sTemp, deviceFriendlyName, 14) == 0)
    {
        pMoniker->BindToObject(0, 0, IID_IBaseFilter, (void**)&pSrc);
        pMoniker->Release();
        pClassEnum->Release();
        pDevEnum->Release();
        return pSrc;
    }
    pPropBag->Release();
    pMoniker->Release();
}

pClassEnum->Release();
pDevEnum->Release();
return pSrc;
}

HRESULT AddFilterByCLSID(
    IGraphBuilder *pGraph,    // Pointer to the Filter Graph Manager.
    const GUID& clsid,        // CLSID of the filter to create.
    LPCWSTR wszName,          // A name for the filter.
    IBaseFilter **ppF)        // Receives a pointer to the filter.
{
    if (!pGraph || ! ppF) return E_POINTER;
    *ppF = 0;
    IBaseFilter *pF = 0;
    HRESULT hr = CoCreateInstance(clsid, 0, CLSCTX_INPROC_SERVER,
        IID_IBaseFilter, reinterpret_cast<void**>(&pF));
    if (SUCCEEDED(hr))
    {
        hr = pGraph->AddFilter(pF, wszName);
        if (SUCCEEDED(hr))
            *ppF = pF;
        else
            pF->Release();
    }
    return hr;
}

int main(int argc, char* argv[])
{
    IGraphBuilder          *pGraph = NULL;
    ICaptureGraphBuilder2 *pBuilder = NULL;
    IBaseFilter             *pSrc = NULL;
    IMediaControl           *pMC   = NULL;
    cmd_t                   mCmd;
    HRESULT                  hr;

    CoInitialize (NULL);
    // Create the filter graph.
    CoCreateInstance(CLSID_FilterGraph, NULL, CLSCTX_INPROC,
        IID_IGraphBuilder, (void **)&pGraph);

    // Create the capture graph builder.
    CoCreateInstance(CLSID_CaptureGraphBuilder2, NULL, CLSCTX_INPROC,
        IID_ICaptureGraphBuilder2, (void **)&pBuilder);

    pBuilder->SetFiltergraph(pGraph);

    // add the video filter
    AddFilterByCLSID(
        pGraph,                                // Pointer to the Filter Graph Manager.
        CLSID_LT-XXXFrameGrabber,              // CLSID of the filter to create.
        L"LT-XXX grabber",                     // A name for the filter.

```

```

&pSrc); // Receives a pointer to the filter.

//***** Begin custom interface *****
IRaw *pIRaw = NULL;
hr = pSrc->QueryInterface(IID_IRaw, (void **)&pIRaw);
//get last settings
pIRaw->GetCommands(&mCmd);
//set my own commands
mCmd.use_colorbar = 1;
mCmd.target_framerate = 0;
mCmd.profile_and_level = 4;
mCmd.usr_sr_epl = 1;
mCmd.denoise = 1;
mCmd.skip_frame_thr = 0;
mCmd.intra_sad_thr = 17000;
mCmd.gop_length = 14;
mCmd.scaling_raw = 0;
mCmd.scaling_comp = 0;
mCmd.adapt_deint = 1;
mCmd.avg_bitrate = 0; //allow constant PQIndex
mCmd.img_res = cfg_auto_avyuv;
mCmd.pq_index = 10;
mCmd.device_no = 0; //only 1 LT-XXX board connected
mCmd.video_vshift = 200;
mCmd.video_hshift = 200;
mCmd.brightness = 0;
mCmd.contrast = 128;
mCmd.hue = 0;
mCmd.saturation = 0;
mCmd.video_16resmode = 0;
mCmd.vcl_timeout = 5000;
mCmd.mv_timeout = 5000;
mCmd.audio_timeout = 5000;
mCmd.raw_timeout = 5000;
mCmd.vcl_burst_length = 2048;
mCmd.ppm_fr_interval_offset = 1000;
mCmd.asf_enable_asf = 0;
mCmd.asf_packet_size = 2048;
mCmd.asf_preroll = 0;
mCmd.asf_use_padding = 0;
mCmd.cust_h_total = 2200; // HTotal
mCmd.cust_h_active = 1920; // HActive
mCmd.cust_v_total = 1125; // VTotal
mCmd.cust_v_active = 1080; // VActive
mCmd.cust_interlaced = 1; // Interlaced
mCmd.cust_p_clocknum = 74250000; // PClockNum
mCmd.cust_p_clockden = 1000; // PClockDen
mCmd.cust_fr_numerator = 1; // FRNumerator
mCmd.cust_fr_denominator = 1; // FRDenominator
mCmd.cust_vco = 20; // Vco
mCmd.cust_vsync_delay = 21; // VsyncDelay
mCmd.cust_hsync_delay = 238; // HsyncDelay
pIRaw->SetCommands(&mCmd);

mCmd.brightness = 0;
mCmd.contrast = 128;
mCmd.hue = 0;
mCmd.saturation = 128;
pIRaw->SetLiveParams(&mCmd);
//***** End custom interface *****

pBuilder->RenderStream(
    &PIN_CATEGORY_CAPTURE, // Pin category
    &MEDIATYPE_Video, // Media type
    pSrc, // Capture filter
    NULL, // Compression filter (optional)

```

```

        NULL //ppf                // Multiplexer or renderer filter
    );

    pGraph->QueryInterface (IID_IMediaControl, (void **) &pMC);

    //run graph
    pMC->Run ();

    MessageBox(NULL, "To stop Player click here!", "Directshow", MB_OK);

    pMC->Stop ();

    pIRaw->Release();
    pSrc->Release();
    pMC->Release();
    pBuilder->Release();
    pGraph->Release();
    CoUninitialize ();

    return 0;
}

```

The next example is a “asfDump_application” example that is delivered with SDK:

C:\Program Files\Enciris Technologies\LT-XXX\SDK\directX_examples\samples\ asfDump_application

Note that the above path is the default installation path.

```

#include "stdafx.h"
#include <DShow.h>
#include <atlbase.h>
#include <initguid.h>
#include <dvdmedia.h>
#include "../..api/IRaw.h"
#include "../..api/idump.h"

// {37555C9E-28CC-4BD8-84FA-9FE011DDEF41}
DEFINE_GUID(CLSID_encirisframegrabber,
0x37555C9E, 0x28CC, 0x4BD8, 0x84, 0xFA, 0x9F, 0xE0, 0x11, 0xDD, 0xEF, 0x41);
//lt_framegrabber.ax

// {D14068C9-358B-4343-A3F5-26703F35B733}
DEFINE_GUID(CLSID_encirisdumpASF,
0xd14068c9, 0x358b, 0x4343, 0xa3, 0xf5, 0x26, 0x70, 0x3f, 0x35, 0xb7, 0x33);
//ltasfmux.ax

BOOL hrcheck(HRESULT hr, TCHAR* errtext)
{
    if (hr >= S_OK)
        return FALSE;
    TCHAR szErr[MAX_ERROR_TEXT_LEN];
    DWORD res = AMGetErrorText(hr, szErr, MAX_ERROR_TEXT_LEN);
    if (res)
        printf("Error %x: %s\n%s\n",hr, errtext,szErr);
    else
        printf("Error %x: %s\n", hr, errtext);
    return TRUE;
}

//change this macro to fit your style of error handling
#define CHECK_HR(hr, msg) if (hrcheck(hr, msg)) return hr;

CComPtr<IPin> GetPin(IBaseFilter *pFilter, LPCOLESTR pinname)

```

```

{
    CComPtr<IEnumPins> pEnum;
    CComPtr<IPin> pPin;

    HRESULT hr = pFilter->EnumPins(&pEnum);
    if (hrcheck(hr, "Can't enumerate pins. "))
        return NULL;

    while(pEnum->Next(1, &pPin, 0) == S_OK)
    {
        PIN_INFO pinfo;
        pPin->QueryPinInfo(&pinfo);
        BOOL found = !wcsicmp(piname, pinfo.achName);
        if (pinfo.pFilter) pFilter->Release();
        if (found)
            return pPin;
        pPin.Release();
    }
    printf("Pin not found!\n");
    return NULL;
}

HRESULT SetCommand(IBaseFilter *pSrc, IBaseFilter *pDest)
{
    int fileSize;
    int packetSize;
    cmd_t mCmd;
    HRESULT hr = S_OK;

    //***** Begin grabber custom interface *****
    IRaw *pIRaw = NULL;
    hr = pSrc->QueryInterface(IID_IRaw, (void **)&pIRaw);
    CHECK_HR(hr, "Can't QueryInterface IRaw");
    //get last settings
    pIRaw->GetCommands(&mCmd);
    //set my own commands
    mCmd.target_framerate = 0;
    mCmd.profile_and_level = 4;
    mCmd.usr_sr_epl = 1;
    mCmd.denoise = 1;
    mCmd.skip_frame_thr = 0;
    mCmd.intra_sad_thr = 17000;
    mCmd.gop_length = 14;
    mCmd.scaling_raw = 1;
    mCmd.scaling_comp = 1;
    mCmd.adapt_deint = 1;
    mCmd.avg_bitrate = 0; //allow constant PQIndex
    mCmd.img_res = cfg_1920x1080p30_dvrgb;
    mCmd.pq_index = 10;
    mCmd.device_no = 0; //only 1 LT-XXX board connected
    mCmd.use_colorbar = 1;
    pIRaw->SetCommands(&mCmd);
    //***** End grabber custom interface *****

    //***** Begin grabber custom interface *****
    IDumpAsf *pIDump = NULL;
    hr = pDest->QueryInterface(IID_IDumpAsf, (void **)&pIDump);
    CHECK_HR(hr, "Can't QueryInterface IDump");
    fileSize = 5; //file chunk size in MegaByte
    packetSize = 2048; //ASF packet size in Bytes
    char baseName[] = "asf_testFile";
    char filePath[] = "c:\\";
    pIDump->set_UserParams(fileSize, packetSize, baseName, filePath);
    //***** End grabber custom interface *****

    pIRaw->Release();
}

```

```

        pIDump->Release();

        return S_OK;
    }

HRESULT BuildGraph(IGraphBuilder *pGraph)
{
    HRESULT hr = S_OK;

    //graph builder
    CComPtr<ICaptureGraphBuilder2> pBuilder;
    hr = pBuilder.CoCreateInstance(CLSID_CaptureGraphBuilder2);
    CHECK_HR(hr, "Can't create Capture Graph Builder");
    hr = pBuilder->SetFiltergraph(pGraph);
    CHECK_HR(hr, "Can't SetFiltergraph");

    //add enciris framegrabber
    CComPtr<IBaseFilter> pencirisframegrabber;
    hr = pencirisframegrabber.CoCreateInstance(CLSID_encirisframegrabber);
    CHECK_HR(hr, "Can't create enciris framegrabber");
    hr = pGraph->AddFilter(pencirisframegrabber, L"enciris framegrabber");
    CHECK_HR(hr, "Can't add enciris framegrabber to graph");

    //add enciris dump ASF
    CComPtr<IBaseFilter> pencirisdumpASF;
    hr = pencirisdumpASF.CoCreateInstance(CLSID_encirisdumpASF);
    CHECK_HR(hr, "Can't create enciris dump ASF");
    hr = pGraph->AddFilter(pencirisdumpASF, L"enciris dump ASF");
    CHECK_HR(hr, "Can't add enciris dump ASF to graph");

    //connect enciris framegrabber and enciris dump ASF
    hr = pGraph->ConnectDirect(GetPin(pencirisframegrabber, L"VC1 Out"),
    GetPin(pencirisdumpASF, L"Video In"), NULL);
    CHECK_HR(hr, "Can't connect enciris framegrabber and enciris dump ASF");

    SetCommand(pencirisframegrabber, pencirisdumpASF);

    return S_OK;
}

int main(int argc, char* argv[])
{
    CoInitialize(NULL);
    CComPtr<IGraphBuilder> graph;
    graph.CoCreateInstance(CLSID_FilterGraph);

    printf("Building graph...\n");
    HRESULT hr = BuildGraph(graph);
    if(hr != S_OK)
    {
        printf("Check if your LT-XXX board is connected and properly
configured...\n");
        printf("Please also check in device manager (devmgmt.msc) if encirisLT
board is present in video controller section\n");
        getchar();
    }
    CHECK_HR(hr, "Can't build the graph");
    printf("Running");
    CComQIPtr<IMediaControl, &IID_IMediaControl> mediaControl(graph);
    hr = mediaControl->Run();
    CHECK_HR(hr, "Can't run the graph");
    CComQIPtr<IMediaEvent, &IID_IMediaEvent> mediaEvent(graph);

```

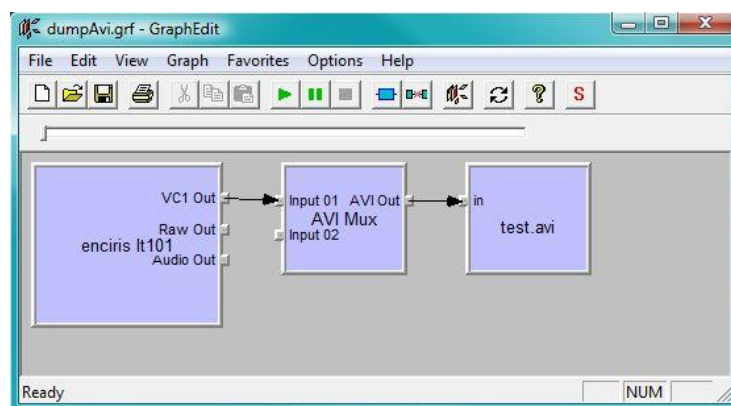


```
        MessageBox(NULL, "To stop recording to c:/asf_testFile_0000000.asf click  
here!", "Directshow", MB_OK);  
  
        CoUninitialize();  
        return 0;  
    }
```

12. Filter Graph Examples

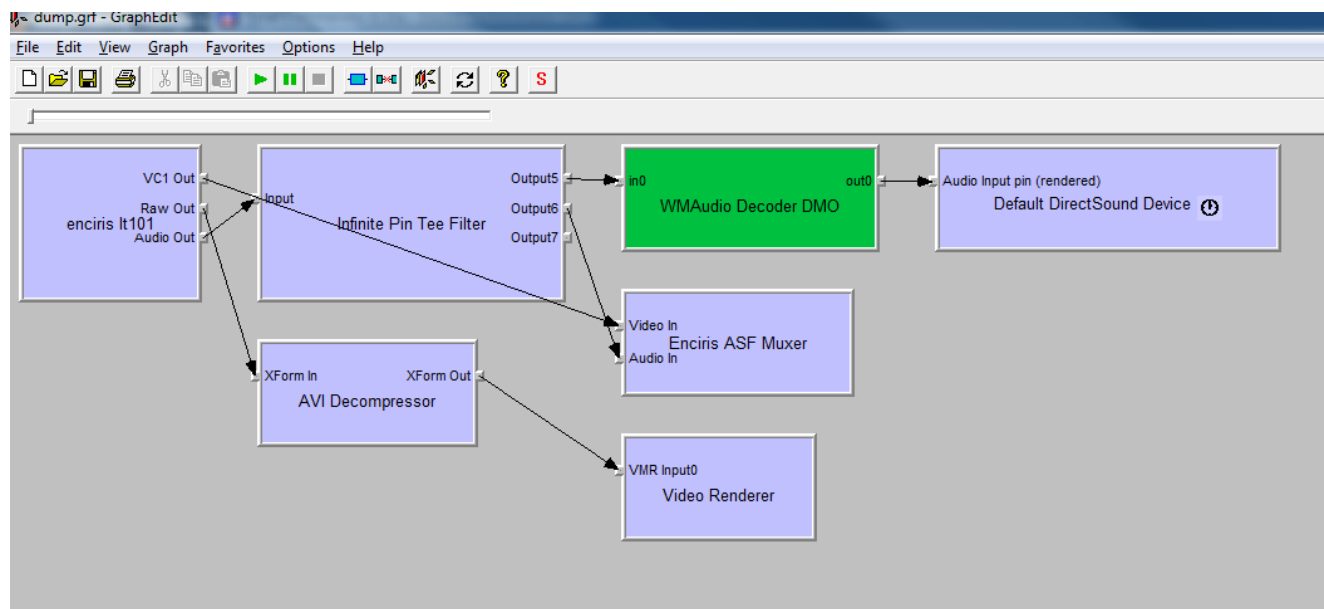
In the following, we show some examples how Enciris LT-XXX filter can be connected within Directshow to accomplish different tasks:

1.1. Record to AVI File



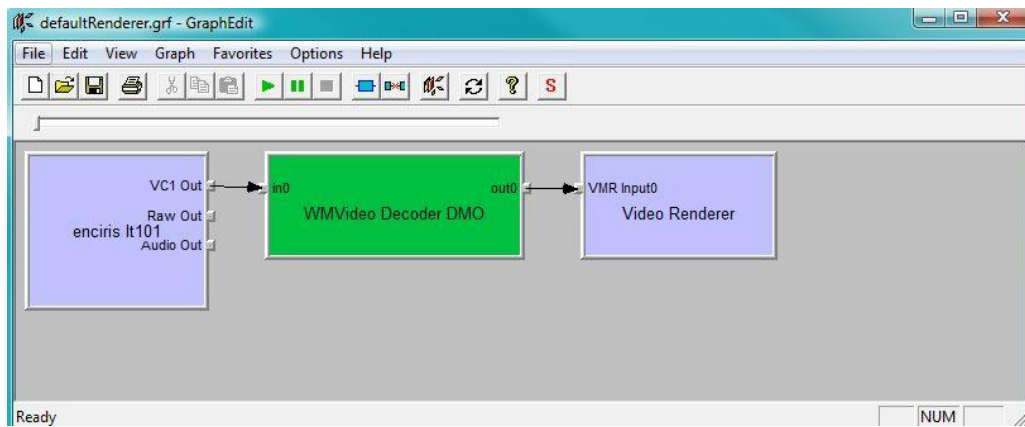
Records video data to an *.avi file using “avi mux” and “file writer” filters

1.2. Record to ASF and Preview Uncompressed Video



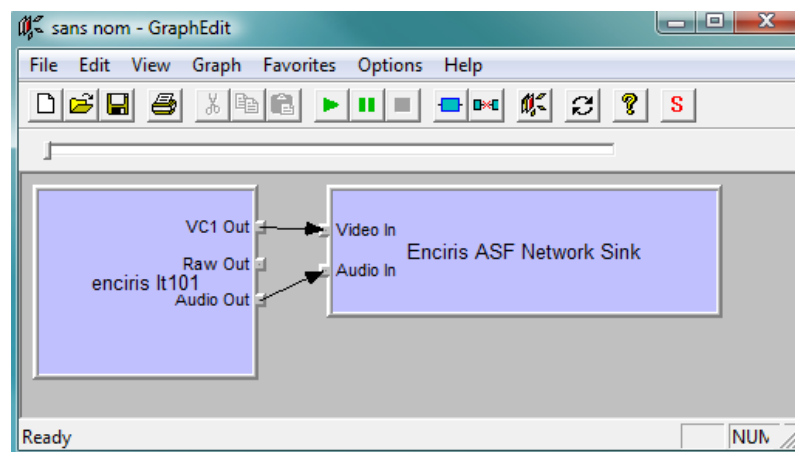
This example shows how to dump video data to an ASF file while watching a raw preview and hearing audio.

1.3. Decode and Preview Compressed Video



This is a very basic rendering application that decodes and renders the encoded captured stream.

1.4. Send Compressed Video over Network via HTTP



LT-XXX Filter connection showing connection to ASF Network Sink Filter. This configuration is an HTTP streaming server.

13. Conclusion

For more information, please also refer to our numerous examples provided in the LT-XXX Software Development Kit (SDK) and the directshow documentation.

Note that Enciris Technologies will be pleased to provide further examples showing special use cases of our API not covered in the SDK.

ENCIRIS TECHNOLOGIES

22 Avenue de l'Europe
81600 Gaillac
France

Tel : +33(0)5 81 18 01 12

info@enciris.com

<http://www.enciris.com>

Copyright © 2016 Enciris Technologies, SAS.

Other companies' product names that may be used in this publication are for identification purposes only and may be trademarks of their respective companies.