VSS H.264 Encoder 4.6

Advanced Settings Description

Vanguard Software Solutions, Inc.
895 Jordan Ave.,
Los Altos, CA 94022
(650) 961-3098 (voice)
(650) 292-2340 (fax)
info@vsofts.com
http://www.vsofts.com

1. Version History

Version	Date	Comments
4.6	02/2011	initial
	02/2011	build 4608:
	03/2011	build 4611: interlace_flags 0x80, 0x100; rc.max_intra_frame_bytes; rc.min_intra_frame_bytes; rc.f
	03/2011	build 4612: rc.flags; rc.gop_bytes

2. Introduction

2.1 Overview

This document describes advanced encoding parameters for the 4th generation of VSS H.264/AVC Codec.

The codec may come either as an SDK with static/dynamic libraries or in the form of DirectShow filter/DMO object, still the set of encoding parameters is the same.

The advanced tuning is performed either with a *configuration file* having a standardized form described below or through API function calls.

2.2 What's New

2.2.1 Ver. 4.6

- NVIDIA hardware acceleration support (V1)
- ParallelStream V2 (much speedup, comparing to 4.5)
- ParallelStream H.264 to H.264 transcoder
- AVC-Intra (precise per-frame rate control, Panasonic compatibility)
- Internal support for different modes of 3D via AVC encoding
- Fixed GOP-size encoding
- Encoding quality enhancements:
    - Rate-control (AVC/SVC, CBR/VBR/Capped VBR, high bitrates, MT problems)
    - Mb-level encoding quality (ME, rounding, quantization, decisions, etc.)
    - Scene changes and fading handling
    - Weighted prediction usage
- More input raw frames formats
- Optimization, speedup
- Better Multithreading for 16+ cores

2.2.2 Ver. 4.5

Version 4.5 is an H.264 codec with MVC support and enhanced SVC and AVC coding.

Compared to v4.4 the key encoder features are:

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

- Multi-View Coding (MVC) encoding support (3D BluRay compliant)
- ParallelStream™ (simultaneous AVC encoding of several bitrates of a single video)
- 64-bit platform native support (Windows and Linux)
- 10 bit through 14 bit support
- Hierarchical B-frames
- Calculation of optional "hints" for decoder (film-grain and filtering SEIs)
- Support for frame packing SEI (3D via AVC coding, Side-by-Side or Top-and-Bottom)
- More ER features
- Better dual-pass encoding quality
- Rate-control enhancements (AVC/SVC, CBR/VBR)
- Encoding quality enhancements
- Optimization, speedup

New parameters in settings:

- *enc_flags* – special encoding flags (MVC reference decoder compatibility)
- *interlace_flags* – new flag value of 0x40
- *bit_depth_luma* – bit depth of the encoded luma plane (8-14)
- *bit_depth_chroma* – bit depth of the encoded chroma plane (8-14)
- *gop.flags* – enables hierarchical B-frames encoding
- *rc.dual_pass_param* – controls second-pass in dual pass encoding
- *frext.transform_8x8* – new value of 2
- *svc.multistream_mode* – controls multi-streaming mode (SVC/AVC/MV)
- *svc.flags* – MVC encoding flags
- *svc.layer[i].max_kbps* – sets maximum bitrate for a multi-stream layer in VBR coding (SVC/AVC/MVC)
- *sei.film_grain_flag, sei.post_filter_flag, sei.sbs_frame_packing_flag, sei.rec_point_flag* (plus a number of dependent parameters) – new encoder-generated SEI messages

2.3 Configuration file

Configuration file is a plain text file where parameters and their values are connected with equal sign ("***parameter = value***"). Each parameter takes up its own line of text. So the configuration file is to meet the following pattern:

```
// comment1
param1 = value1
// comment2
param2 = value2
```

Values could be of 3 types: text, integer and hexadecimal. Hexadecimal values should be written in "C" language notation like 0xNNNN.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

A typical codec delivery package contains *v4enc.cfg* configuration file with summary of all encoder settings and commentary blocks describing their meaning and range of possible values. The section below duplicates information from *v4enc.cfg* (although the latter is still to be used for reference since it's supposed to be the most up-to-date). More detailed description of the parameters are in the subsequent sections.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

3. Parameters Summary

Parameter	Range	Description
3.1 Input description		
input.width		Input frame width (pixels)
input.height		Input frame height (pixels)
input.colorspace	[0-8]	Input colorspace: 0=IYUV,I420; 1=YV12; 2=YUYV,YUY2; 3=YVYU; 4=UYVY; 5=RGB555; 6=RGB565; 7=RGB24; 8=RGB32, 9 = YUV 4:0:0 planar, 10 = YUV 4:2:2 planar
input.sample_size	[1, 2]	Bytes per sample
input.significant_bits	[8–14]	Significant bits per sample
3.2 Preprocessing		
preproc.intra_precision	[0–4]	Complexity analysis precision: 0= disabled, 4= maximum
preproc.chroma_format_idc	[0–2]	Internal chroma format: 0= YUV 4:0:0, 1(default)= YUV 4:2:0, 2= YUV 4:2:2, 3(not supported)= YUV 4:4:4
preproc.crop.enable	[0, 1]	Picture cropping: 0(default)= disabled, 1= enabled
preproc.crop.left		Left offset in pixels
preproc.crop.top		Top offset in pixels
preproc.crop.right		Right offset in pixels
preproc.crop.bottom		Bottom offset in pixels
preproc.me_denoise.level	[0–6]	ME temporal denoise: 0(default)= off, 1= weak, 2= moderate, 3= middle, 4= strong, 5= very strong, 6= maximum
preproc.me_denoise.skip_luma	[0, 1]	ME denoise: 0(default)= process luma, 1= skip luma
preproc.me_denoise.skip_chroma	[0, 1]	ME denoise: 0(default)= process chroma, 1= skip chroma
preproc.deinterlace	[0–3]	Deinterlace: 0(default)= none, 1= copy top field, 2= copy bottom field, 3= blend fields
preproc.step[*#N*].type		Filter type: 0x00= none, 0x10= BLUR_3x3, 0x11= BLUR_5x5, 0x20= SHARPEN_3x3, 0x21= SHARPEN_5x5, 0x30= MEDIAN_3x3, 0x31= MEDIAN_5x3, 0x40= RESIZE, 0x50= TEMPORAL_DENOISE

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

preproc.step[*#N*].param0 preproc.step[*#N*].param1 preproc.step[*#N*].param2 preproc.step[*#N*].param3		Parameters for the Nth filter step selected in *preproc.step[#N].type* (see Preprocessing section below for details)

3.3 Multi-Stream Encoding (SVC, AVC, MVC)

svc.num_layers	[0 – 7]	Number of multi-stream layers: 0= multi-stream mode OFF
svc.multistream_mode	[0, 1, 2]	Multi-stream mode: 0=SVC, 1=AVC ParallelStream™, 2=MVC
svc.key_picture_period		SVC MGS key pictures period (affects all layers): 0(default)= none
svc.temporal_mode	[0, 1]	SVC temporal scalability: 0(default)= off, 1= on (2 layers)
svc.flags		Multi-stream flags: 1=Put MVC prefix-nal units into stream, 2=Put MVC picture delimiter into stream, 4=Use fast version of ParallelStream, 8=Generate MVC SEI according to Blu Ray spec
svc.layer[*#N*].extend		Enhancement layer type: 0= spatial dyadic (2x), 1= quality CGS, 2= quality MGS, 15= spatial 1.5x, 100= spatial custom

svc.layer[*#N*].flags_i
svc.layer[*#N*].flags_p
svc.layer[*#N*].flags_b
		SVC inter-layer prediction flags (for I-, P- and B- pictures):
		`SVC_ADAPTIVE_BASEMODE_FLAG = 0x01,`
		`SVC_ADAPTIVE_RES_PRED_FLAG = 0x02,`
		`SVC_ADAPTIVE_MV_PRED_FLAG = 0x04,`
		`SVC_DEFAULT_BASEMODE_FLAG = 0x10,`
		`SVC_DEFAULT_RES_PRED_FLAG = 0x20,`
		`SVC_DEFAULT_MV_PRED_FLAG = 0x40`

svc.layer[*#N*].profile_idc	[83, 86]	Encoding profile SVC: 83= Scalable Baseline, 86= Scalable High AVC: see *profile_idc* below MVC: 118= Multiview High, 128= Stereo High
svc.layer[*#N*].level_idc	[0, 10,…]	Level selection: 0= auto, 10= 1.0, 12=1.2, 32=3.2, 40=4.0 etc.
svc.layer[*#N*].sym_mode	[0, 1]	Layer symbol mode: 0=CAVLC, 1=CABAC
svc.layer[*#N*].kbps		SVC/MVC: cumulative bitrate for current and all the lower (including base) layers (Kbps) AVC: individual layer target bitrate (Kbps)
svc.layer[*#N*].max_kbps		Maximum allowed bitrate for VBR (Kbps): 0(default)= not set
svc.layer[*#N*].qp_intra	[0 – 51]	QP value for intra frames (*rc.type = 0*)
svc.layer[*#N*].speed.i svc.layer[*#N*].speed.p svc.layer[*#N*].speed.b	[0 – 7]	Speed/quality mode (for I-, P- and B- pictures): 0=slowest, 7 =fastest

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

svc.layer[*#N*].vui_aspect_ratio_idc svc.layer[*#N*].vui_sar_width svc.layer[*#N*].vui_sar_height		Layer aspect ratio parameters. Valid only if *vui.aspect_ratio_info_present_flag = 1* (see Aspect Ratio IDC) svc.layer[*#N*].vui_aspect_ratio_idc: 0(default)= auto, 1–16= manual, 255= custom (Extended_SAR)
svc.layer[*#N*].slice.mode svc.layer[*#N*].slice.param svc.layer[*#N*].slice.i_param svc.layer[*#N*].slice.b_param		Slicing parameters for SVC layers (see AVC Slicing section below)
svc.layer[*#N*].num_mgs_slices	[1 – 4]	SVC: no. of slices to split coefficients for MGS layers: 1(default)= no splitting
svc.layer[*#N*].mgs_coeffs		SVC MGS transform coefficient splitting pattern: 0(default)= auto
svc.layer[*#N*].frame_width		Layer frame width (pixels) SVC: valid for *svc.layer[#N+1].extend=100* only
svc.layer[*#N*].frame_height		Layer frame height (pixels) SVC: valid for *svc.layer[#N+1].extend=100* only
3.4 General		
profile_idc	[66, 77, 100, 110, 122]	H.264 profile: 66= Baseline, 77= Main, 100= High, 110= High 10, 122= High 4:2:2
level_idc	[0, 10,…]	H.264 level selection: 0=auto, 12=1.2, 32=3.2, 40=4.0 etc.
sym_mode	[0, 1]	Symbol mode: 0=CAVLC, 1=CABAC
bit_depth_luma	[8 – 14]	Bit depth of the encoded luma plane: 8= default
bit_depth_chroma	[8 – 14]	Bit depth of the encoded chroma plane: 8= default

enc_flags		Special encoder flags (MVC ref. decoder compatibility):
		
		`ENC_DISABLE_VUI = 1,`
		`ENC_SLICE_TYPE_012 = 2,`
		`ENC_SPS_ONLY_ONCE = 4,`
		`ENC_REC_POINT_IDR = 8`

sei.pic_timing_flag	[0 – 2]	Picture timing and buffering period SEIs control: 0 - disable; 1 - put all picture SEIs in single NAL unit; 2 - Put each SEI in separate NAL unit
frame_width		Multi-Stream: base layer frame width (pixels) SVC: valid for *svc.layer[1].extend=100* only
frame_height		Multi-Stream: base layer frame height (pixels) SVC: valid for *svc.layer[1].extend=100* only

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

interlace_mode	[0, 3]	Interlace coding mode (not supported in baseline profile): 0 = disabled; 1 = all fields, top field first; 2 = all fields, bottom field first; 3 = MBAFF (mb-level adaptive frame/field coding)
interlace_flags	[0, 1, 2]	Interlace flags: 0=standard ME from bottom field (P) to top one (I), 0x01 = disable ME from bottom field to top one; 0x02 = encode both fields as intra (only top field is intra by default); 0x04 = show bottom field first when mbaff of frame coding; 0x08 = force decoder to play frame-encoded stream as interlaced; 0x10 = put zero POC offsets for both top & bottom fields (for mbaff coding); 0x20 = RD-opt MBAFF decision; 0x40 = disable preprocessing for bottom field; 0x80 = add telecine picture structure for frame-encoded video; 0x100 = "3-2" start with 3. Together with interlace bottom-field-first defines start position of telecine
direct_mode	[0, 1]	Direct mode type for B-frames: 0=temporal, 1=spatial
constrained_intra_pred	[0, 1]	constrained intra prediction flag
chroma_qp_offset	[-26, +26]	offset for chroma QP
weighted_pred_flag	[0, 1]	Enables weighted prediction tool
poc_type	[0, 2]	Picture Order Count (POC) type
gpu_acceleration	[0, 1]	Enable NVidia GPU acceleration: 0=disable, 1=enable
avc_intra_class	[0, 50, 100]	AVC-Intra class encoding: 0=off, 50=class 50, 100=class 100
avc_intra_flags		Bitwise combination of avc-intra encoding flags: 1=force all features for Panasonic compatibility

3.5 Group Of Pictures (GOP)

gop.idr_period		period of IDR frames (0=only first, N=on every N-th I-frame)
gop.keyframes	[0-300]	period of I-frames (0=only first)
gop.bframes	[0-4]	number of B-frames between P (0=no B-frames)
gop.min_bframes	[0 - gop.bframes]	minimum number of B-frames for adaptive mode (if equal to *gop.bframes* adaptive mode is disabled
gop.emulate_b	[0 – 2]	B-frame emulation: replaces B frames with non-reference Ps (requires non-zero *gop.bframes*): 0(default)= off, 1= on, 2= encode in natural order
gop.aud	[0 – 2]	enable Access Unit Delimiters on output: (default)=disable, 1=enable AUD+PPS, 2=enable AUD only
gop.num_units		num units per tick
gop.time_scale		time scale *framerate (fps) = gop.time_scale / (2*gop.num_units)*

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

gop.min_intra_period		minimal distance between intra frames for continuous scene change (frames)
gop.flags		GOP flags: 1= enable Hierarchical B-frames (valid if *gop.bframes* ≥ 3)
3.6 High Profile Tools		
frext.transform_8x8	[0, 1]	allows using 8x8 transform: 0(default)= off, 1= on (adaptive), 2= use whenever possible (faster)
frext.second_qp_offset	[-26 – +26]	offset for V-chroma QP
frext.scaling_matrix	[0, 1]	Switches on using alternative scaling matrix: 0(default)= off, 1= on.
3.7 Rate Control		
rc.type	[0-5]	Type of the Rate Control: 0=fixed QP, 1=VBR, 2(default)=CBR, 3=CBR+filler, 4=dual-pass: 1st pass, 5=dual-pass: 2nd-pass, 6=intra-only mode
rc.kbps	[1 – 100,000]	Target bitrate (Kbps): 0= use a predefined value (depending on the frame size)
rc.auto_qp	[0, 1]	Enables automatic QP selection: 1(default)= automatic first QP and range selection, 0= use manual settings (see below)
rc.qp_intra	[0-51]	Quant Parameter for I-frames (requires *rc.auto_qp*=0)
rc.qp_delta_p	[0-51]	base QP delta between I and P
rc.qp_delta_b	[0-51]	base QP delta between P and B
rc.qp_min	[0-51]	Minimum allowed QP for Rate Control
rc.qp_max	[0-51]	Maximum allowed QP for Rate Control
rc.scene_detect	[0 – 100]	Scene change detection threshold: 0= disable, 50= default, 100= most sensitive
rc.vbv_length	[0 – 8,000]	rate control buffer length in msecs; if set to 0 takes default value depending on *rc.type*: 1000 ms for CBR, 2000 ms for VBR
rc.qp_modulation	[0, 1]	QP modulation mode: 0= off, 1(default)= on
rc.mb_update	[0, 1]	enables MB-level rate-control: 0= off, 1(default)= on
rc.look_ahead	[0–8]	number of look-ahead frames: 0=off, 1= default
rc.max_kbps		maximum allowed peak bitrate (Kbps) for VBR mode: 0(default)=not set
rc.initial_cpb_removal_delay		Initial fullness for VBV buffer (1/90000 sec). (-1)(default) means it takes value of (90**rc.vbv_length*/2)

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

rc.dual_pass_param	[0 – 256]	dual-pass behavior: 0= CBR-like, 128= default, 256 - "fixed QP"-like
rc.max_intra_frame_bytes		maximum size of intra frames in bytes: 0(default)= no restriction
rc.min_intra_frame_bytes		minimum size of intra frames in bytes: 0(default)= no restriction
rc.gop_bytes		size of GOP in bytes (effective only with *rc.flag* of 0x40): 0= calculated automatically (based on *rc.max_kbps*)
rc.flags		Bitwise RC flags: 0x01= ignore buffer overflow for VBR coding, 0x02= use qp_delta_b as max for automatically calculated delta_b, 0x04= use qp_delta_b as min for automatically calculated delta_b, 0x10 - add filler NALs (can be set in VBR mode if max_kbps is specified), 0x20 - put cbr_flag into sps (effective only with the flag above), 0x40 - supports fixed number of bytes in GOP (see also *rc.gop_bytes*)
3.8 Motion Estimation		
me.subdiv	[1–127]	Macroblock subdivision mask: 1=16x16, 2=16x8, 4=8x16, 8=8x8, 16=8x4, 32=4x8, 64=4x4, 127=all
me.search_range	[1-64]	Maximum search range in full pels: -1(default)= automatic calculation from picture size
me.max_refs	[1-5]	number of pictures (frames or fields) used for motion search (reference pictures)
3.9 Speed Modes		
speed.i	[0–8]	speed/quality mode for I-frames: 0= slowest, 4= default, 8= fastest
speed.p	[0–8]	speed/quality mode for P-frames: 0= slowest, 4= default, 8= fastest
speed.b	[0–8]	speed/quality mode for B-frames: 0= slowest, 4= default, 8= fastest
speed.automatic	[0, 1]	enables automatic real time control: 0(default)= off, 1= on
3.10 Slicing		
slice.mode	[0–3]	selects slice mode (0=none, 1=#mbs per slice, 2=#bytes per slice; 3=#slices)
slice.param		provides appropriate number value for *slice.mode*
slice.i_param		provides appropriate number value for *slice.mode* for I-slices
slice.b_param		provides appropriate number value for *slice.mode* for B-slices

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

3.11 Deblocking		
deblock.flag	[0, 1]	Configures loop filter (0=parameter below ignored, 1=parameters used)
deblock.disable	[0, 1]	Disable loop filter in slice header (0=Filter, 1=No Filter)
deblock.alpha_c0	[-6-+6]	Alpha & C0 offset div. 2
deblock.beta_c0	[-6-+6]	Beta offset div. 2
3.12 Multithreading		
mt.disable	[0, 1]	Disables multithreading: 0(default)= MT on, 1= MT off
mt.num_threads		No. of worker threads to run: 0(default)= auto
mt.max_pict_tasks	[0 – 5]	Max no. of simultaneously coded pictures: (-1)(default)= auto
mt.max_raw_frames		Max value of frames to hold in async-feed encoding: 0(default)= calculate automatically
3.13 Error Resilience (ER)		
er.enable	[0, 1]	Enables ER tools: 0(default)= disabled, 1= enabled
er.initial_expected_loss_percent	[0 – 100]	Initial expected loss rate (percents): 0= no ER, 15= default
er.intra_update_method	[0 – 3]	Intra update method: 0= off, 1(default)= adaptive (motion tracking), 2= rolling intra MB rows, 3= random intra MBs
er.fast_motion_update_period		Short update period (fast motion) for adaptive method (frames): 0= disabled, 1, 2, 3= recommended
er.full_motion_update_period		Long update period (fast and slow motion) for adaptive method (frames): 0= disabled, 5, 6.. gop.keyframes/2= recommended
er.total_intra_update_period		Period for picture full intra update (frames) (valid only if *er.enable* = 1 and *er.initial_expected_loss_percent* > 0)

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

4. Detailed Description

This section describes parameters and their dependencies. For the complete list of settings, their default values and ranges look at the summary table in Section 3.

4.1 Input Description

Input Description set of parameters are to be used by the application level.

input.width *input frame width (pixels).*

Must be multiple of 2

input.height *input frame height (pixels).*

Must be multiple of 2 for progressive or 4 for interlaced content

input.colorspace *input colorspace format*

0(default)= IYUV,I420; 1= YV12; 2= YUYV,YUY2; 3= YVYU; 4= UYVY; 5= RGB555; 6= RGB565; 7= RGB24; 8= RGB32; 9= YUV 4:0:0 planar, 10= YUV 4:2:2 planar

The internal encoder's color representation format is specified by *preproc.chroma_format_idc* parameter (see below). The default colorspace is IYUV/I420 (YUV 4:2:0). Colorspace conversion (if needed) is performed on the preprocessing stage

input.sample_size *size of input pixels (bytes)*

1(default), 2

input.significant_bits *significant bits of input sample*

8= default, up to 14 (depending on the library version)

See also *bit_depth_luma* and *bit_depth_luma* parameters below

4.2 Preprocessing

preproc.intra_precision *complexity analysis precision*

0= disabled 1= 2 4x4 blocks per macroblock used, 2(default)= 4 blocks, 3= 8 blocks, 4= 16 blocks (full macroblock)

Picture complexity analysis is performed on the preprocessing stage. Higher modes can improve the rate control behavior giving better quality of encoding.

preproc.chroma_format_idc *Internal encoder chroma subsampling format*

0= YUV 4:0:0 (monochrome), 1(default)= YUV 4:2:0, 2= YUV 4:2:2, 3(not supported)= YUV 4:4:4

For input formats higher than the selected internal one chroma downsampling is produced what results in loosing some color information before encoding.

YUV 4:2:2 coding is supported in High Profile 4:2:2 (*profile_idc = 122*, see below) only. This will work for 4:2:2 and 4:4:4 input material only.

preproc.crop.enable *enables picture cropping*

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

0(default)= disabled, 1= enabled

preproc.crop.left *left offset in pixels*

preproc.crop.top *top offset in pixels*

preproc.crop.right *right offset in pixels*

preproc.crop.bottom *bottom offset in pixels*

preproc.me_denoise.level *ME temporal denoise*

0(default) = off, 1 = weak, 2 = moderate, 3 = middle, 4 = strong, 5 = very strong, 6 = maximum

This is a more complicated method of temporal denoise (compared to the simple temporal denoise filter- see below) and therefore more computationally demanding.

preproc.me_denoise.skip_luma *ME denoise: option to skip luma component*

0(default)= process, 1= skip

preproc.me_denoise.skip_chroma *ME denoise: option to skip chroma component*

0(default)= process, 1= skip

preproc.deinterlace *deinterlace*

0(default)= none, 1= duplicate top field, 2= duplicate bottom field, 3= adaptive field blend

Although interlaced video is best to be coded in one of the interlaced modes (see below) one can use deinterlace feature of the preprocessing stage to make it progressive (with some quality degradation, of course). After that progressive mode should be used for encoding. This provides better compatibility with 3rd-party decoders and frees from the need of decoder-side deinterlacing before playback.

Various picture filters can be applied in a user-specified order (following the preprocessing steps listed above). Seven stages (steps) can be used. Results of one filter step are used as the input for the next filter of the sequence. Filters of the same type can specified multiple times. A filter can have up to four parameters, the actual number depends on the filter type (see table below).

preproc.step[*#N*].type *Filter type*

0x00(default)= none, 0x10= BLUR_3x3, 0x11= BLUR_5x5, 0x20= SHARPEN_3x3, 0x21= SHARPEN_5x5, 0x30= MEDIAN_3x3, 0x31= MEDIAN_5x3, 0x40= RESIZE, 0x50= TEMPORAL_DENOISE

preproc.step[*#N*].param0 *Parameters for the corresponding filter of #N step*
preproc.step[*#N*].param1
preproc.step[*#N*].param2
preproc.step[*#N*].param3

the actual number of parameters for the preprocessing step depends on the filter type used (see table below)

where *#N*= [0–6] *preprocessing step index*

Filter	Parameters
0x00 = none	parameters are ignored

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

0x10 = BLUR_3x3	*param0*= filter luma, *param1*= filter chroma 0(default)= disabled, 1= enabled
0x11 = BLUR_5x5	*param0*= filter luma, *param1*= filter chroma 0(default)= disabled, 1= enabled provides better quality than blur 3x3
0x20 = SHARPEN_3x3	*param0*= filter luma, *param1*= filter chroma 0(default)= disabled, 1= enabled
0x21 = SHARPEN_5x5	*param0*= filter luma, *param1*= filter chroma 0(default)= disabled, 1= enabled
0x30 = MEDIAN_3x3	*param0*= filter luma, *param1*= filter chroma 0(default)= disabled, 1= enabled
0x31 = MEDIAN_5x3	*param0*= filter luma, *param1*= filter chroma 0(default)= disabled, 1= enabled
0x40 = RESIZE	*param0*= new picture width (pixels), *param1*= new picture height (pixels) The following condition must be met: picture dimensions can be changed by a factor of four or less
0x50 = TEMPORAL_DENOISE	*param0*= luma strength, *param1*= luma buffer length, *param2*= chroma strength, *param3*= chroma buffer length *strength*: [0-10], 0= off, 10= maximum *buffer length*: [2-7], no. of luma/chroma planes to be used for filtration In strong filtration modes moved objects and scene changes can cause artifacts.

Note: when combining the crop tool (*preproc.crop*) with any combination of the preprocessing filters, the cropping is done before the 1st stage filter is applied. That is, for example, input picture is cropped before it gets resized (with 0x40 resize filter).

Note: when combining the de-interlace tool (*preproc.deinterlace*) with any combination of the preprocessing filters, de-interlace is done after all the filters are applied.

4.3 Multi-Stream Coding (SVC, AVC, MVC)

In addition to SVC version 4.5 supports Multi View Coding (MVC) and so called ParallelStream™. That is simultaneous encoding of several AVC streams of different resolution and bitrate from one input video. All these encoding modes are controlled by the same set of parameters (*svc_settings_t* structure).

Generally multi-stream means encoding of several layers. 'Base' layer encoding is controlled by parameters of the general and rate control sections below. Other layers are governed by the *svc.layer[].* parameters described below. The third group of parameters (like the GOP structure) are common to all layers.

Encoder behavior is ruled either by setting fixing quant parameters (see *rc.layer[].kpbs* below) or in terms of target bitrate (using the rate-control mechanism). This is controlled by parameter *rc.type* (see Rate Control section below) and is common to all the multi-stream layers (including the base one).

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

The following limitations are currently set for the multi-stream coding:

- SVC, MVC and ParallelStream can't be used together with interlace coding
- Weighted prediction and transform 8x8 are disabled for SVC coding

svc.num_layers

number of multi-stream layers (not counting the base one)

```
0(default)= multi-stream off, N= no. of layers, 7= maximum
```

svc.multistream_mode

multi-stream mode

```
0(default)= SVC, 1= AVC ParallelStream, 2= MVC
```

Depending on this value other parameters of the svc_settings_t structure are interpreted in different ways. See details in appropriate fields.

svc.key_picture_period

SVC (MGS) and AVC key picture period (affects all layers)

```
0(default)= none
```

Parameter to control inserting of SVC key pictures, P-frames which are "more significant" than others. Key pictures use only previous key picture for reference.

The tradeoff between SVC drift effect[1] and enhancement layer coding efficiency is achieved with the SVC key picture concept. This is periodical inserting of inter coded pictures which use only the base layer and only previous key pictures for motion compensation.

Ver 4.5: Now it also works in AVC mode (*svc.num_layers*=0). This is referred to as *Hierarchical Frame Structure* (HFS). The period is measured in the number of reference frames. Note that currently it can't be used in combination with hierarchical B-frames and interlace encoding.

svc.temporal_mode

enables SVC temporal scalability mode (2 layers)

```
0(default)= disabled, 1= enabled
```

When enabled, layer with *temporal_id=0* refers to all the encoded frames and layer with *temporal_id=1* contains reference (I and P) pictures only. The number of non-reference frames is regulated with *gop.bframes* parameter. See also *gop.emulate_b* for the option to set-up non-reference P-frames (instead of Bs).

svc.flags

bitwise multi-stream encoding flags

```
0(default)= none
1= Put MVC prefix-nal units into stream
2= Put MVC picture delimiter into stream
4= Use fast version of ParallelStream (new!)
8= Generate MVC SEI according to Blu Ray spec
```

This parameter is introduced for better MVC encoding flexibility

[1] Drift describes the effect when motion-compensated prediction loops at encoder and decoder are not synchronized due to quality refinement packets being discarded from the SVC stream.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

svc.layer[*#N*].extend

SVC enhancement layer type

```
0= SVC_EXTEND_2X2= spatial dyadic 2x,
1= SVC_EXTEND_1X1= quality CGS,
2= SVC_MGS_EXTEND= quality MGS,
15= SVC_EXTEND_1_5= spatial 1.5x

100= SVC_EXTEND_CUSTOM= spatial arbitrary (custom)
```

Each SVC layer *#N* produces spatial or quality enhancement for the underlying one *#N*-1. Quality enhancement means the spatial resolution is not changed between the layers. Spatial enhancement means that the upper layer (with greater index) has a higher spatial resolution. The resolution of the uppermost layer with *#N* = (*svc.num_layers*-1) equals to the input frame size (*input.width*, *input.height*), provided no resize or crop operations at the preprocessing stage are performed.

Quality enhancement mode makes sense only when the bitrate/QP (depending on the *rc.type* mode) setting of the current layer is higher than that of the previous one.

For 1.5x spatial scalability the original dimensions must be divisible by three (e.g. 1920x1080).

For 2x and 1.5x spatial scalability modes resolution of the reduced layer will be calculated automatically.

For custom spatial scalability mode the user must specify dimensions of the underlying (*#N*-1 or base layer if *#N*=0) reduced layer explicitly (see parameters below).

svc.layer[*#N*].flags_i
svc.layer[*#N*].flags_p
svc.layer[*#N*].flags_b

SVC encoding flags for I-, P- and B- pictures

```
Bitwise combination of the following constants:
       SVC_ADAPTIVE_BASEMODE_FLAG = 0x01,
       SVC_ADAPTIVE_RES_PRED_FLAG = 0x02,
       SVC_ADAPTIVE_MV_PRED_FLAG = 0x04,

       SVC_DEFAULT_BASEMODE_FLAG = 0x10,
       SVC_DEFAULT_RES_PRED_FLAG = 0x20,
       SVC_DEFAULT_MV_PRED_FLAG = 0x40

0x07= default
```

The bitwise combination of the flags above regulate the usage of SVC inter-layer prediction tools (see SVC sub-section below for details).

If an *adaptive* flag is set (1) for some tool then the *default* value of this tool is ignored. If an *adaptive* flag is reset (0) then the corresponding *default* value (0 or 1) is used for all macroblocks (of that particular picture type).

The recommended setting for the parameters is 0x7 – adaptive usage for all tools. Non-adaptive mode can be used to speedup decoder or for some other reasons.

svc.layer[*#N*].profile_idc

H.264 encoding profile

```
SVC: 83= Scalable Baseline, 86= Scalable High
AVC: see profile_idc below, 100= default
MVC: 118= Multiview High, 128= Stereo High
```

svc.layer[*#N*].level_idc

H.264 level selection (level x 10)

```
0= auto, 12= 1.2, 32= 3.2, 40= 4.0 etc.
41= default
```

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

svc.layer[*#N*].sym_mode

layer encoding symbol mode

0=CAVLC, 1(default)=CABAC

Each layer can be encoding using either CABAC or VLC coding. Entropy coding mode can be set for each layer independently.

svc.layer[*#N*].kbps

target bitrate (Kbps)

0= automatic selection (depending on the picture size)

SVC: cumulative bitrate value for current and all the SVC layers below (including base – set by *rc.kbps*). That is, *svc.layer[#N].kbps* must be greater than *svc.layer[#N-1].kbps*

AVC: individual layer bitrate

MVC: summary bitrate value of current layer plus base layer (*rc.kbps*). Automatic bitrate setting gives (1.5**rc.kbps*)

These parameters make sense for non-zero *rc.type* selections only (see Rate Control section below)

svc.layer[*#N*].max_kbps

maximum allowed bitrate for current layer (Kbps)

0= default

Applicable for VBR mode only (*rc.type* = 1). It has the same meaning as *rc.max_kbps* parameter for plain AVC encoding. It can be used for MVC, SVC and ParallelStream™ encoding. In these cases *rc.max_kbps* parameter is related to base layer.

There are following restrictions on using this parameter:

It should be set to zero for all layers or to some values higher than corresponding kbps for all the layers. If this restriction is not met then encoder modifies settings by following algorithm:

1. If *rc.max_kbps* == 0 all max_kbps are set to zero for all layers.
2. If *rc.max_kbps* > 0 all max_kbps are checked for validity (they must be greater than corresponding kbps). For those layers (including base), where the above condition is not held it is set to *2*kbps*.

svc.layer[*#N*].qp_intra

QP value for intra-frame coding

This parameter is applicable for fixed quality mode (*rc.type* = 0) only.

Quant parameters for P and B slices are calculated using common *rc.qp_delta_p* and *rc.qp_delta_b* parameters (see below)

svc.layer[*#N*].speed.i
svc.layer[*#N*].speed.p
svc.layer[*#N*].speed.b

speed mode (for I-, P- and B- slice/frame correspondingly):

0=slowest, 4= default, 7 =fastest

This setting allows to achieve the desired speed/quality trade-off

svc.layer[*#N*].vui_aspect_ratio_idc

individual layer aspect ratio IDC

0(default)= auto, 1-16= manual, 255= Extended_SAR

Valid with *vui.aspect_ratio_info_present_flag* = 1 only (see Aspect Ratio IDC section below)

Automatic calculation are based on the settings for the base layer (taking into account the corresponding layer resize).

svc.layer[*#N*].vui_sar_width
svc.layer[*#N*].vui_sar_height

Extended_SAR width and height

See Aspect Ratio IDC section for more details

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

svc.layer[*#N*].slice.mode
svc.layer[*#N*].slice.param
svc.layer[*#N*].slice.i_param
svc.layer[*#N*].slice.b_param

slicing settings

see description in the corresponding section below

svc.layer[*#N*].num_mgs_slices

SVC: number of slices to split coefficients for MGS enhancement layer

`1(default)= no splitting, 4= maximum allowed`

This is to enable MGS partitioning of transform coefficients feature. See SVC sub-section below.

svc.layer[*#N*].mgs_coeffs

SVC (MGS) transform coefficient splitting pattern (for num_mgs_slices > 1)

`0(default)= auto`

Zero value means automatic partitioning selection to make approximately equal slices. If greater then 0, each hex digit of this parameter specifies the last coefficient index of each slice. The highest order (left-most) hex digit corresponding to the last mgs slice must be F. See Example 2 configuration below.

svc.layer[*#N*].frame_width

layer frame width (pixels)

SVC: to be set for custom spatial scalability mode only (*svc.layer[#N+1].extend* = 100*)*.

svc.layer[*#N*].frame_height

layer frame height (pixels)

SVC: to be set for custom spatial scalability mode only (*svc.layer[#N+1].extend* = 100).

where
#N= 0 to (*svc.num_layers*-1)

enhancement layer index

4.3.1 Scalable Video Coding (SVC)

SVC-encoded stream is composed of the base layer (AVC-compliant sub-stream) plus several spatial and/or quality enhancement layers above it. Enhancement layers are arranged in the order of increasing quality/resolution. That is, the base layer has the lowest resolution / poorest quality. Layer with [0] index is the base layer's enhancement. And so on, until the upmost [*svc.num_layers* – 1] layer having the highest resolution (the one of the original input picture plus preprocessing) and/or the highest quality (highest bitrate / lowest QP value). VSS SVC encoder also supports two layers of temporal scalability.

Each frame in H.264 SVC standard can be coded as a set of slices of different ***layers***. Different layers are marked with 2 parameters "Dependency ID" and "Quality ID". Typically (but not mandatory) different dependency IDs specify different spatial dimension. Different quality IDs (within the same dependency ID) specifies different encoding quality of the same spatial dimensions. All slices with the same dependency ID are called "Dependency Representation". Encoding several layers within one dependency representation we call *MGS* (medium-grain). The main advantage of MGS-encoded stream is flexibility of extracting legal stream to fit desired bitrate. With *CGS* (coarse-grain) encoding the only points where *MANE* (media-aware network element) can safely switch from one layer to another are IDR frames. With MGS encoded stream MANE can decide to send or skip some layers on each frame. Thus it can fit a big variety of desired bitrates, even if encoding was done only for few (say 3-4) bitrates for different quality layers. There is another SVC tool which helps to achieve the same goal. It is partitioning of transform coefficients into different slices.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

The main ***inter-layer prediction*** options (see *svc.layer[].flags* parameters above) on the macroblock level are:

- Base-mode macroblock type
- Inter-layer residual prediction
- Inter-layer motion prediction

SVC specification allows adaptive or non-adaptive usage of each one of these tools. Adaptive means that the decision whether to use this tool or not is done by the encoder on the macroblock level. Non-adaptive means that the tool is turned on/off for all macroblocks of the slice. In this latter case the default flag value is used (higher bits of the corresponding flag parameter). Note that if non-adaptive modes are selected with all zero default values (0x0 for the flag parameter), the slice will not take any advantage of the SVC-coding (traditional AVC coding is used only).

Note: Not all of the flags can be set independently: if macroblock type is base-mode then motion vectors are always derived via inter-layer prediction, thus, mv-prediction flag is ignored.

Partitioning of transform coefficients. Note that when *svc.layer[].num_mgs_slices* > 1 for some layer(s) the actual number of encoded SVC layers will be greater then the specified value of *svc.num_layers*. In the Example 2 configuration below encoder will generate one base layer (dep_id = q_id = 0) and 4 SVC layers with dep_id = 0 and q_id = 1,2,3,4. Rate control will control only the size of base layer and overall size of these 4 layers. Taking in account MGS stream extracting possibilities this shouldn't be a big disadvantage.

Example 1

As an example here are setting for 3-layer encoding with base layer, one spatial and one quality CGS SVC layers. Note, that not all parameters are defined. For missing parameters their default values will be used:

```
svc.num_layers = 2 //base + 2 SVC layers

svc.layer[1].extend   = 1  //quality scalability CGS
svc.layer[1].kbps     = 8000

svc.layer[0].extend   = 0 // spatial extension 2x
svc.layer[0].kbps     = 4000
svc.layer[0].flags_p  = 0x3 //don't use mv-pred flag
```

Example 2

```
svc.num_layers = 1
svc.key_picture_period = 3
svc.layer[0].extend   = 2
svc.layer[0].num_mgs_slices = 4
svc.layer[0].mgs_coeffs   = 0xFB73
```

This means to encode base layer and one SVC layer of the same size (quality MGS), encode every 3rd picture as "key" and separate coefficients of the SVC layer into four slices using following rule:

- Coefficients [0,3] – slice 0
- Coefficients [4,7] – slice 1
- Coefficients [8,11] – slice 2

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

- Coefficients [12,15] – slice 3

Example 3

```
input.width = 352 //original dimensions, - dimensions of last SVC layer
input.height = 288

svc.num_layers = 2

svc.layer[1].extend = 100

svc.layer[0].extend = 100
svc.layer[0].frame_width = 240 //dimensions of the 1st SVC layer
svc.layer[0].frame_height = 192

frame_width = 176 //base layer dimensions
frame_height = 144
```

With these settings encoder will generate 176x144 base layer; 240x192 first SVC layer and 352x288 last SVC layer video.

4.3.2 ParallelStream™ (AVC)

Encoder SDK since version 4.5 allows simultaneous AVC encoding of several bitrates from one video input. This encoding mode is called *ParallelStream*™. It should be much more efficient compared to separate encoding of different resized video from one input. Encoder will perform all necessary resize on-the-fly and it will guarantee the exact GOP structure of the output stream (GOP-aligned). This feature can be used by Network element to switch smoothly from one stream to another when network condition changes. In some sense this encoding mode is similar to SVC with custom spatial scalability encoding. The difference is that there is no interlayer prediction, and all "layers" can be decoded independently.

Version 4.6 adds new faster ParallelStream encoding mode. To achieve it encoder uses "fine-to-coarse" encoding technique. That means that video with highest bitrate and/or resolution is encoded first. All other encoding is done, using internal information of this encoding. Thus all next encoding can be done much faster. This encoding mode is switched on by special flag (*svc.flags = 4*).

Encoder uses the same `svc_settings_t` structure to control ParallelStream™ encoding. It can support up to 8 streams encoded simultaneously. Encoder settings are used practically in the same manner as for SVC encoding with custom spatial scalability. The only difference is that bitrate settings (*svc.layer[].`kbps`* and *svc.layer[].`max_kbps`*) for each layer is the individual bitrate value for this particular layer (rather than cumulative rate as for SVC coding). Another option is *automatic* resolution and bitrate calculation.

To open encoder for ParallelStream™ encoding caller must set the following parameters in svc structure:

```
svc.multistream_mode = MULTISTREAM_MODE_AVC;
svc.num_layer = (number_of_output_streams-1);
```

Output of ParallelStream™ encoding are NAL-units of all the streams.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

Example

Input resolution is 1280x720 pixels. We are to generate 3 streams with following characteristics:

1. 1280x720; 3Mbps
2. 592x336; 1025Kbps
3. 288x160; 350Kbps

To do this we prepare configuration file with following settings:

```
svc.num_layers = 2
svc.multistream_mode = 1

svc.layer[0].frame_width  = 1280
svc.layer[0].frame_height = 720
svc.layer[0].kbps = 3000

svc.layer[1].frame_width  = 592
svc.layer[1].frame_height = 336
svc.layer[1].kbps = 1025

frame_width = 288
frame_height = 160
rc.kbps = 350
rc.type = 2
```

Auto Settings Service

ParallelStream™ library provides automatic bitrates and resolutions calculations for each AVC channel in case they are missing for one or more layer.

Mandatory settings are:

```
svc.multistream_mode = 1
svc.num_layers = 1 … 7 (it means 2 … 8 AVC streams)
gop.time_scale and gop.num_units (to define framerate)
input.width and input.height  (as resolution of last channel – last layer)
rc.type > 0 (i.e. one of Rate Control algorithms should be activated)
```

All other settings are not mandatory and will be calculated automatically by H.264 library according to the model recommended by *Microsoft SmoothStreamingMBRCalculator*.

Auto-settings service is activated when at least one of bitrates or resolutions is not defined. It is possible to define or not define resolutions and bitrates in any combinations. Missing values will be calculated according to the model. The order of auto-calculations is as follows:

1. Min and Max Bitrates values are defined.
2. Rest Bitrates values are calculated.
3. Pixels per second (pps) values based on bitrate values are calculated.
4. Pictures widths and heights values based on pps, fps and aspect ratio are calculated.
5. Calculated widths and heights are refined to better fit with original aspect ratio.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

6. If width or height exceeds input picture size they are truncated to input values and `VSSH_WARN_SVC_LAYER` is returned. This can occur in not a last channel with too large bitrate setting of the last channel is ordered.
7. Check that resolutions are steadily increasing

Additional services:

1. Bitrate of the smallest channel (in case of its absence) is calculated by the following formula:

$$MinBitrate = MaxBitrate / Number\_of\_streams$$

2. Special case of automatic bitrates calculation is implemented. It is activated in case of defined base layer bitrate and undefined last layer bitrate. In this case *rc.kbps* setting is interpreted as Max bitrate (instead of Min Bitrate). This case is implemented to simplify auto-configuration from console application command line.

Automatic resolutions are always multiple of 16.

Automatic bitrates are set only if Rate Contol is in use.

Default profile is MAIN.

4.3.3 Multi View Coding (MVC)

Multi-view video sequences are captured by different cameras in different positions and through different angles; they are just different representations of the same scene. Therefore, there exists an additional redundancy in the view direction for 3D video called view redundancy. The MVC standard utilizes prediction between different views to exploit this redundancy and improve the compression ratio. Multi-view coding tools are introduced in H.264 standard (Annex H). According to this extension, the stream can contain one *base* view (fully compliant with AVC standard) and a set of *dependent* views encoded with slices of `NALU_TYPE_SLICE_EXT(20)` type. The concept is very similar to SVC. For dependant views special profiles are introduced:

- *Stereo High Profile* (2 views with possible interlace)
- *Multiview High Profile* (2 or more views without interlace)

Codec 4.5 currently doesn't support interlace coding with MVC.

Encoder uses the same `svc_settings_t` structure to control MVC encoding. It can support up to 7 dependant views encoding. To open encoder for MVC encoding caller must set the following parameters:

```
svc.multistream_mode = MULTISTREAM_MODE_MVC;
svc.num_layer = number_of_dependant_layers;
```

Each layer will be encoded with dependence on base layer only. Bitrates values of each layer are rate of this plus base layer. Parameters which are only SVC specific are ignored.

Output of MVC encoding is NAL-units stream including both Base View and Dependent views (similar to SVC). They can be separated by their types and headers structure in the same

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

manner as SVC layers separating.

The main practical usage of MVC encoding now is encoding of Stereo HD content for 3D enabled BluRay players. There are some specific restrictions on base and dependent views in this case. (For example frames of both views shall be spitted into 6 or more slices). It also demands to separate base and dependent view into different streams. An example of encoder configuration file with all BluRay restrictions (other parameters will be default):

```
profile_idc = 100
level_idc = 41

gop.keyframes = 23
gop.bframes = 2
gop.num_units = 1001
gop.time_scale = 50000
gop.aud = 1

rc.type = 2
rc.kbps = 8000
rc.vbv_length = 1000

slice.mode = 3
slice.param = 6

frext.transform_8x8 = 1

vui.aspect_ratio_info_present_flag = 1
vui.aspect_ratio_idc = 1

svc.num_layers = 1
svc.multistream_mode = 2
svc.flags = 2

svc.layer[0].profile_idc = 128
svc.layer[0].level_idc=41
svc.layer[0].kbps = 12000
svc.layer[0].slice.mode = 3
svc.layer[0].slice.param = 6
```

BluRay spec also demands to put special "dependant delimiters" NALs into dependant view. This can be switched on by setting `M_FLAG_MVC_DELIMITERS` flag in *`svc.flags`* parameter.

4.4 General

profile_idc *H.264 encoding profile*

`66= Baseline, 77= Main, 100= High, 110= High 10, 122= High 4:2:2`

Encoding profile specifies a set of bit stream features that can be used. Note that tools not included in the selected profile will be automatically disabled by the encoder. For example, if you set *profile_idc=66* then CABAC, interlace coding and B-frames will all be disabled.

YUV 4:2:2 coding is supported in High Profile 4:2:2 only. It requires also *preproc.chroma_format_idc = 2* setting (see above).

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

level_idc *H.264 level selection (level x 10)*

0= auto, 12=1.2, 32=3.2, 40=4.0 etc.

Encoding level specifies a set of constraints laid on bitstream parameters. It is the user's duty to select appropriate level or to use the auto level option. For example, *level_idc=32* is the default setting for SD (D1) resolution, *level_idc=42* – for full HD.

Encoder can calculate (*level_idc = 0*) minimum level value which will be enough to handle all the bitstream parameters. It depends on frame dimensions, framerate, bitrate, maximum number of reference frame, interlace/progressive mode. It doesn't affect either the encoding speed, bitrate, quality or other encoder behavior.

sym_mode *Symbol mode*

0=CAVLC, 1=CABAC

Selects entropy-coding mode. CAVLC is Context-Adaptive Variable-Length Coding; *CABAC* is Context-based Adaptive Binary Arithmetic Coding – more powerful (+10% compression), it is not supported in H.264 Baseline profile.

bit_depth_luma *Bit depth of the encoded luma plane (8-14)*

bit_depth_chroma *Bit depth of the encoded chroma plane (8-14)*

The following restrictions are currently applied:

For 8-bit only library: *bit_depth_luma = bit_depth_chroma = input.significant_bits* = 8

input.sample_size = 1

For 8-14-bit ibrary: *bit_depth_luma = bit_depth_chroma = input.significant_bits*

input.sample_size = 2

enc_flags *Bitwise special encoding flags*

0(default)= none,
1= ENC_DISABLE_VUI = don't put vui infromation in sps,
2= ENC_SLICE_TYPE_012 = encode slice types as 0,1 or 2 (default is 5,6,7),
4= ENC_SPS_ONLY_ONCE = put SPS only for the first frame of stream,
8= ENC_REC_POINT_IDR = put recovery point SEI for IDR picture too

This parameter is introduced for better encoding flexibility. One of the reasons is that reference MVC decoder has a restriction. It doesn't decode MVC streams if they are encoded without these flags. These flags can be expanded in future versions.

Note that ENC_DISABLE_VUI flag can't be used together with *sei.pic_timing_flag*, because timing information works only when bitrate, frame rate and HRD buffer parameters are present in VUI section of SPS.

sei.pic_timing_flag *Enables picture timing and buffering period SEI messages*

0(default)= disabled, 1= enabled, put all picture SEIs in single NAL unit, 2= enabled, put each SEI in separate NAL unit

Regulates whether to put buffering period and picture timing SEI messages into the encoded bitstream. This is required for *HRD* stream conformance (see also *rc.type*)

frame_width *Multi-Stream base layer frame width (pixels)*

SVC: to be set by user for custom spatial scalability mode only (*svc.layer[1].extend* = 100).

frame_height *Multi-Stream base layer frame height (pixels)*

SVC: to be set by user for custom spatial scalability mode only (*svc.layer[1].extend* = 100).

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

interlace_mode *Interlace coding mode (not supported in baseline profile)*

```
0(default) = disabled (progressive mode, frames);
1 = all fields, top field first;
2 = all fields, bottom field first;
3 = MBAFF (macroblock-level adaptive frame/field coding)
```

Interlaced video material (fields) is recommended to be coded in one of the interlaced modes (provides better quality).

Note: codec does not recognize interlaced content automatically. It is the user's duty to enable interlaced coding mode or to make appropriate de-interlace decision on the preprocessing stage.

interlace_flags *Interlace flags*

```
0(default)=standard ME from bottom field (P) to top one (I);
0x01 = disable ME from bottom field to top one;
0x02 = encode both fields as intra (only top field is intra by
default);
0x04 = show bottom field first in MBAFF coding;
0x08 = force decoder to play frame-encoded stream as interlaced;
0x10 = put zero POC offsets for both top & bottom fields (for
MBAFF coding);
0x20 = RDO MBAFF decision
0x40 = disable preprocessing for bottom field (new in v4.5)
0x80 = add telecine picture structure for frame-encoded video
0x100 = "3-2" start with 3. Together with interlace bottom-field-
first defines start position of telecine) (new in v4.6)
```

Options for interlaced modes (*interlace_mode* > 0). A bitwise combination of several flags can be used.

Ver. 4.5 encoder uses new enhanced algorithm of interlaces material encoding. That is, separate preprocessing and complexity analysis for each field. It makes default interlace encoding little bit slower, comparing to version 4.4 or 3.5. `0x40` flag disables this new processing. This will make interlace material encoding faster, but with some possible quality degradation.

Ver. 4.6 supports *telecine* structure signaling in picture timing SEI messages. To control it use following new flags in interlace_flags settings parameter:

INT_ADD_TELECINE_PICT_STRUCT = 0x80

INT_TELECINE_START_WITH_3 = 0x100

If flag _ADD_TELECINE_PICT_STRUCT is set, encoder generates telecine picture structures. Telecine picture structure is signaled by pic_struct value (*frame_info_t* structure) for each progressively encoded frame.

The following values are used:

3 - top field, bottom field, in that order

4 - bottom field, top field, in that order

5 - top field, bottom field, top field repeated, in that order

6 - bottom field, top field, bottom field repeated, in that order

For consecutive frames this value should be assigned as:

..., 3, 5, 4, 6, 3, 5, 4, 6, 3, 5, 4, 6, 3, 5, 4, 6, ...

Flags INT_BOTTOM_FIRST (0x04) and INT_TELECINE_START_WITH_3 (0x100) uniquely define from which value to start in this sequence:

INT_BOTTOM_FIRST	INT_TELECINE_START_WITH_3	pic_struct

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

0	0	3
0	1	5
1	0	4
1	1	6

direct_mode *Direct mode type for B-frames*

0(default)=temporal, 1=spatial

If a macroblock in a B-frame is coded as Skip or Direct, then no motion vector is coded, and instead a predicted MV is used. This takes fewer bits compared to the other inter types. This option controls how that MV is computed.

constrained_intra_pred *constrained intra prediction flag*

0(default)= disabled, 1= enabled

Constrained intra texture prediction is a error-resilience related encoder feature. It limits inter macroblocks of P and B (inter) frames to be predicted only from the neighboring inter-coded macroblocks. (This results in a special intra macro-block pattern when top and left neighbors of each intra MB are also intra.) This prevents from error-propagation.

chroma_qp_offset *QP offset for chroma components*

0(default)= off, [-26, +26]= QP offset

This parameter defines the difference between quant-parameters in luma and chroma texture coding. If for some reasons we prefer luma coding to be more accurate than chroma we should set *positive* value. If we prefer chroma coding to be more accurate than luma we should set *negative* value. (Related parameter is *frext.second_qp_offset* for High Profile)

weighted_pred_flag *Enables weighted prediction tool*[2]

0(default)= disabled, 1= enabled

Designed to increase compression efficiency on fading scenes

poc_type *Picture Order Count (POC) type (see H.264 standard)*

0(default), 1, 2

Encoder supports only 0 and 2 for Baseline profile

gpu_acceleration *Enables NVidia GPU acceleration*

0=disable, 1=enable

Encoder SDK version 4.6 allows using of GPU if it is detected on the computer. GPU-enabled encoding uses it for motion estimation and macroblock intra/inter decision. The rest of encoding still implemented on host CPU. Current version of GPU usage makes encoding much faster, but has its drawback. Because of the limited data bandwidth between GPU and host it is impossible to achieve the best CPU-only encoding quality. In any case, the speed/quality tradeoff for fast encoding is better.

Note, that the latest NVidia drivers must be installed.

[2] Does not work together with MBAFF (*interlace_mode = 3*, see above)

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

avc_intra_class *AVC-Intra class encoding*

`0=off, 50=class 50, 100=class 100`

Used to generate AVC-Intra streams compatible with SMPTE 2027-2007 class 50 and class 100 specifications. When enabled encoder automatically modifies *rc.type* parameter to 7 (see Rate Control) and assigns appropriate GOP structure (see Intra only encoding)

avc_intra_flags *Bitwise combination of avc-intra encoding flags*

`1=AVC_I_FORCE_PANASONIC - force all features for Panasonic compatibility`

4.5 Group Of Pictures (GOP)

gop.keyframes *period of I-frames*

`0=only first, 1=all frames are intra, N=every Nth frame is intra`

Key-frame period affects seeking performance (in file playback), decoder playback start-up time (for network streaming), recovery time from network errors and overall picture quality. We recommend to use the following values: 1-2 secs for network streaming, 5-10 secs for file playback.

Each I-frame of the VSS-encoded stream contains SPS/PPS NAL-units. That's why VSS H.264 Decoder is able to start playback from any I-frame (not only IDR).

When reducing key-frame period significantly (less than 1 sec approx) in CBR/VBR (as opposed to fixed QP mode) encoding codec fails to increase the bit rate to conform to the VBV buffer parameter. This can result in skipped frames in CBR mode and overall quality degradation.

Scene change detection tool can affect the GOP structure (reduce the I-frame period) by inserting IDR I-frame. Key frame count is reset on the scene change too. See also *rc.scene_detect*

gop.idr_period *period of IDR frames*

`0=only first, 1(default)=all I-frames are IDR, N=every Nth I-frame`

In H.264 IDR and non-IDR I-frames have different meaning. *IDRs* (Instantaneous Decoding Refresh) are 'traditional' I-frames the way they were known in previous standards. An IDR I-frame allows subsequent frames reference itself and the frames after it, thus marking a closed GOP. This behavior resets the decoder completely and thus guarantees reliable seeking.

Non-IDR I-frame can be understood as an intra-coded P-frame. They can be referenced by preceding B-frames what improves picture quality (especially in high motion scenes) and reduces I-frame flicker effect by smoothing the P-to-I frame transition. The side effect (when using 3rd-party H.264 decoders) of non IDR frames that they can reduce seeking precision or increase decoder start-up time .

Scene change detection tool can affect the GOP structure (reduce the IDR frame period) by inserting IDR I-frame. IDR frame count is reset on the scene change too. See also *rc.scene_detect*

gop.bframes *number of B-frames between two consecutive Ps*

`0= off, 2= default, N= no. of B-frames (max no. of B-frames for adaptive mode, see gop.min_bframes below)`

B (or bidirectional) frames are Main and High profile feature, they are not supported in Baseline. The usage of B-frames can increase compression ratio and picture quality significantly. The recommended value for general usage is 2.

Baseline profile: non-reference P frames can be used to emulate B-frames (see *gop.emulate_b* below)

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

gop.min_bframes *minimum number of B-frames (for adaptive GOP mode)*

`0= default, up to gop.bframes`

When *gop.min_bframes* is less than *gop.bframes* the so called adaptive GOP mode is enabled. Number of B-frames for the adaptive mode will vary between *gop.min_brames* and *gop.bframes* values. Encoder will reduce B-frame number (modifying the GOP structure) for higher motion scenes to achieve optimal quality for the predefined target bitrate.

When *gop.min_bframes* is equal to *gop.bframes* the number of B-frames if constant.

gop.emulate_b *Enables non-reference P frames in Baseline profile (requires non-zero gop.bframes)*

`0= off, 1(default)= on (bidirectional), 2= encode in natural order`

H.264 standard allows use of non-reference (incl. bidirectional) P-frames what gives the user a tool to increase picture quality in Baseline profile. They also have the following advantages over ordinary P-frames: (1) can be lost in network transmission without significant side effect (error propagation) since they are not referenced, (2) can be used to save bitrate (by increasing their individual QP) since the poor quality won't propagate to the subsequent frames, (3) give possibility to speed-up the decoder (by switching off deblocking or even skipping them completely). The minus of bidirectional P-frames is that once they appear the stream latency value increases by the length of one frame for each emulated B-frame.

gop.aud *enables Access Unit Delimiters in the output stream*

`0(default)=disable, 1= enable AUD+PPS, 2= enable AUD only`

gop.num_units *Num units per tick*

`10000= default`

Input frame rate (in fps) is specified with the use of two parameters: *frame_rate (fps) = gop.time_scale / (2*gop.num_units)*

gop.time_scale *Time scale*

`600000= default`

Examples:

```
gop.num_units = 10000
gop.time_scale = 600000
```

give 30 fps,

```
gop.num_units = 10010
gop.time_scale = 600000
```

give 29.97 fps

gop.min_intra_period *Minimal distance between intra frames for continuous scene change (frames)*

`4= default`

This is to prevent from continuous I-frame insertion on scene changing.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

gop.flags *Bitwise GOP flags*

```
0= default,
1= encode Hierarchical B-frames (if 3 or more B-frames specified
– see gop.bframes)
2= don't set IDR-slice on scene change
```

1) In hierarchical B-frame mode encoder will place reference B-frame inside the B-frame sequence. This is supposed to enhance overall encoding quality. Note that it is not supported for fields coding (only for progressive or MBAFF interlace).

2) Typically encoder inserts IDR I-frame when scene change is detected. If for some reasons this is behavior is undesirable this flag should be set. In this case IDR will be entered only as defined by *gop.idr_period* setting.

4.5.1 Intra only encoding

It is possible to generate intra-only encoding conformance to different Intra profiles specified in the H.264 standard. No new parameters are introduced to do this. It can be done just by setting appropriate GOP parameters:

```
gop.keyframes = 1
gop.idr_period = 1
```

For intra-only encoding of interlaced material, MBAFF encoding mode should be used:

```
interlace_mode = 3
```

To generate CAVLC-Intra profile conformance video the following must be set in addition:

```
sym_mode = 0
```

Other parameters (such as bitrate, speed, slice mode and others) can be set as designed.

Ver. 4.6 can also generate AVC-Intra streams compatible with SMPTE 2027-2007 class 50 and class 100 specifications. A new parameter is introduced: see *avc_intra_class* in General

4.6 High Profile Tools

frext.transform_8x8 *transform 8x8 tool*

```
0(default)= off, 1= adaptive, 2= use 8x8 transform whenever
possible without Intra16x16 (faster)
```

High profile H.264 standard introduces new texture encoding mode using transform 8x8. This mode is supposed to give encoding quality enhancements, especially for HD video.

frext.second_qp_offset *QP offset for V-chroma component (see also chroma_qp_offset above)*

```
0(default)= off, [-26, 26]= QP offset
```

If present, it defines the offset for V-chroma component only, while *chroma_qp_offset* is applied for U-component. If not, *chroma_qp_offset* is applied for both.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

frext.scaling_matrix *switches on using alternative scaling matrix*

0(default)= off, 1= on

High-profile H.264 standard introduces another new texture encoding mode – using scaling matrix. Typically it is used to increase precision of low-frequency coefficients compared to the one of high-frequency coefficients. This mode is also supposed to give subjective quality enhancements, especially for HD-video. The scaling matrices can be predefined and custom. It is different for intra and inter macroblocks, for 4x4 and 8x8 transform and for luma and chroma.

All in all there exist 8 types of scaling matrix:

- 4x4 - intra - Y, U and V
- 4x4 - inter - Y, U and V
- 8x8 - intra and inter (only Y)

For details please refer to the H.264 standard.

If one sets only the parameter above the predefined scaling matrix is used. To set custom scaling matrix one should call *v4e_set_scaling_matrix()* API function

4.7 Rate Control

rc.type *Type of the Rate Control (RC)*

0= fixed QP: fixed quality mode, no RC (see *rc.qp_intra*),
1= VBR: target bitrate exceeding is allowed,
2(default)= CBR: target bitrate exceeding is not allowed,
3= CBR+filler: strictly constant bitrate, bitrate variations are compensated with zero fillers
4= dual-pass: 1st pass (see *rc.dual_pass_param*)
5= dual-pass: 2nd pass
6= intra-only mode

For all modes, except the fixed QP, the *VBV* (Video Buffer Verifier) paradigm is implemented. The buffer length is defined by *rc.vbv_length* parameter. Buffer is protected from underflow and overflow by the rate control mechanism.

RC behavior in cases of overflow differs depending on the RC type: in VBR frame skipping is not allowed what can result in temporary target bitrate exceeding, in CBR modes skip frames are allowed and the bitrate is kept below. In both cases target bitrate value is kept on average. Bitrate peaks occur, especially on outcoming I-frames. It is the user's duty to do the bitrate smoothing, for example, for network streaming applications when a stable output is required.

HRD Compliance. Constant bitrate modes 2 and 3 are *HRD* (Hypothetical Reference Decoder)-compliant with VSS CBR mode corresponding to VBR HRD mode and VSS CBR+fller mode – to CBR HRD mode. CBR+filler (rc.type=3) will set CBR flag in SPS HRD-buffer params. See also: *sei.pic_timing_flag*, *rc.initial_cpb_removal_delay, rc.vbv_length*

Intra-only encoding. SMPTE specification demands exact number of encoded bytes per frame. So, new rate control modes are introduced to achieve this: 6, 7. These modes will try to encode all frames of the same size. The last mode will add filler NAL-untis to make them exactly of the same size. See also: *avc_intra_class* in General and Intra only encoding

rc.kbps *Target (average) bitrate in kbps*

0= set automatically (depending on the picture size)

Not applicable in fixed QP mode (*rc.type* = 0)

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

rc.auto_qp

Enables automatic QP selection mode

0= off, use manual settings (*rc.qp_intra*)
1(default)= on, automatic first QP and range selection

In automatic mode starting QP and QP range are selected depending on the bitrate and resolution, taking into account picture complexity analysis data. Subsequent intra QP and min/max values vary adaptively along encoder run (with min-max range kept constant).

rc.qp_intra

Quant Parameter for I-frames

30= default

This parameter have different meaning depending on the RC type and *rc.auto_qp* setting. For fixed QP mode it specifies the exact QP value for I-frames. For other RC types this is starting QP value for I-frames. If auto-QP mode is turned on (rc.auto_qp=1) the value is to be corrected by RC (according to the target bitrate, video resolution etc.) before use.

rc.qp_delta_p

Base delta between QP for I- and P-frames

N= (QP_P - QP_I), 2= default

A positive value means starting QP difference for I and P frames. Higher QP for P frames allows more bits spending on the I-frame and thus increasing the overall picture quality.

On still scenes I-frame flicker effect can be visible due to the difference in I- and P- frame texture coding. One can reduce the effect by setting this value to zero, although a better choice would be using of non-IDR I-frames (see *gop.idr_period* parameter).

rc.qp_delta_b

Base delta between QP for P- and B-frames

N= (QP_B - QP_P), 3= default

A positive value means starting QP difference for P and B frames. B-frames by their nature are not referenced, therefore one can save bits on them since quality degradation won't propagate **- Obsolete (see rc.flags)**

rc.qp_min

Minimum allowed QP for CBR/VBR Rate Control

12= default

Defines lower QP boundary in VBR/CBR modes.

rc.qp_max

Maximum allowed QP for CBR/VBR Rate Control

51= default

Defines upper QP boundary in VBR/CBR modes.

Notes:

The min-max range restricts the P-frames QP values. I- and B-frame QPs are calculated using the corresponding delta QP values (see *rc.qp_delta_p*, *rc.qp_delta_b* above).

rc.scene_detect

Scene change detection threshold

0=disable; 50=default; 100=most sensitive

When video scene change is detected IDR I-frame insertion is forced by the rate control. Key frame and IDR frame counts are also reset on the scene change (see also *gop.keyframes*, *gop.idr_period, gop.min_intra_period, gop.flags)*.

rc.vbv_length

Rate control buffer length in msecs

0= takes default value depending on rc.type: 1000 ms for CBR, 2000 ms for VBR

VBV buffer paradigm is used for rate control in CBR and VBR modes. The less is

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

the buffer size, the more strict is the rate control in terms of keeping the bit-rate stable in time. The larger is the buffer size, the higher quality one can get, especially in picture complexity changes.

Please note that too low buffer lengths (500 ms or so) can lead to frame skips (especially right after large I-frames) and overall quality degradation.

HRD-buffer support: this is exact value of CPB size set in SPS (see also *rc.type*)

rc.qp_modulation *QP modulation*

`0= off, 1(default)= on`

Adaptive QP modulation is a supplementary encoder tool. It allows having individual QP value for every macroblock of a picture. It increases the compression ratio (and thus the overall quality) by rearranging bits between macroblocks but can somehow reduce visibility of details since parts of the picture that are regarded as 'less important' get higher QP.

rc.mb_update *MB-level rate-control*

`0= off, 1(default)= on`

This parameter affects the macroblock-level QP update feature of the rate control mechanism. MB-level update helps VBV rate-control in preventing buffer overflow on complex scenes thus making it more precise. If a starting QP of a picture is selected incorrectly then this option will give the encoder a chance to adjust it on the fly.

rc.look_ahead *Number of look-ahead frames*[3]

`0= off, 1= default, N= number of frames`

For more precise bit allocation (more frequently referenced MBs get lower QP values and therefore more bits) it is helpful for the rate control to have information on the complexity of upcoming frames before actual current frame coding. This can be achieved by introducing some delay in encoding. Expected result is overall encoding quality enhancement, especially for video with different scenes. Higher values will cause higher delays and are supposed to give more quality gain.

rc.max_kbps *Maximum allowed peak bitrate (Kbps) for VBR mode*

`0=not set, N= bitrate in Kbps`

Do not confuse with *rc.kbps* that sets average target bitrate

rc.initial_cpb_removal_delay *Initial fullness of VBV buffer (1/90000 sec)*

`(-1)(default)= 90*`*`rc.vbv_length`*`/2`

This allows to avoid empty buffer at the encoder start. Useful when coding a few portions of a stream simultaneously (in parallel).

HRD-buffer support: this exact value will be set in first buffering period update SEI (see also *rc.type*)

rc.dual_pass_param *Regulates dual-pass RC behavior (see rc.type)*

`0= CBR-like, 128= default, up to 256= "fixed QP"-like`

This parameter smoothly tunes second pass encoding from CBR-like encoding (0) to fixed-QP like encoding (256)

rc.max_intra_frame_bytes *Maximum size of intra frames in bytes*

`0(default)= no restriction`

rc.min_intra_frame_bytes *Minimum size of intra frames in bytes*

[3] Currently values of 0 and 1 are supported only

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

0(default)= no restriction

rc.gop_bytes

Size of GOP in bytes (effective only with rc.flag of 0x40- see below)

0= calculated automatically (based on *rc.max_kbps*)

rc.flags

Bitwise RC flags

0x01= ignore buffer overflow for VBR coding,
0x02= use qp_delta_b as max for automatically calculated delta_b,
0x04= use qp_delta_b as min for automatically calculated delta_b
0x10= add filler NALs (can be set in VBR mode if *rc.max_kbps* is specified),
0x20= put cbr_flag into sps (effective only with the flag above),
0x40= supports fixed number of bytes in GOP (see also *rc.gop_bytes*)

Ver 4.6 includes VBR encoding enhancements. VBR encoding implies that predefined bitrate is not strictly obligatory for all the video. For complex scenes it can be greater to preserve visual quality. It is done automatically on some level. User can also define *rc.qp_max* explicitly to control this. Nevertheless, encoder tries to make average bitrate close to predefined *rc.kbps*. To achieve this encoder reduces bitrate on simple scenes. As a result we can see bitrate fluctuations and not good enough quality on simple scenes where defects become noticeable. In order to avoid this effect user can set *RC_IGNORE_VBR_BUFFER_OVERFLOW = 0x01* in rc.flags.

When encoding B-frames encoder uses automatically calculated QP to achieve balance between visual quality and bitrate saving. Typically it is greater than QP for P-frames. If for some reason the user is not satisfied with automatically calculated deltas they can use *rc.qp_delta_b* parameter together with following flags in rc.flags:

RC_BOUND_MAX_QP_DELTA_B = 0x02

RC_BOUND_MIN_QP_DELTA_B = 0x04

Thus setting both flags will disable automatic delta calculation at all. Encoder will use rc.qp_delta_b in any case.

Note that if none of these flags (0x02, 0x04) is set then *rc.qp_delta_b* has no effect.

RC_ADD_FILLER = 0x10

RC_SET_CBR_FLAG = 0x20

The two flags above take are set automatically in *rc.type=3* mode (CBR+filler). The user is allowed to enable them in VBR mode if needed.

RC_FIXED_GOP_SIZE = 0x40

Encoder supports fixed GOP size encoding switched on by the flag above. In this case it uses no scene change and no open GOP settings. Then it tries to limit each GOP size by some predefined number of bytes. Note, that the result is not 100% guaranteed (an overflow may occur). In such a case it is the upper level application's duty to re-encode the resulted GOPs. Encoder guarantees that it is safe to replace GOPs without loosing stream integrity. Typical usage of this feature is to set VBR mode with *rc.kbps* slightly (10%) less than desired and to set proper *rc.max_kbps* or *rc.gop_bytes* parameters (if *rc.max_kbps* is not set then it will be calculated based on *rc.gop_bytes* or vice versa, if both set then *rc.gop_bytes* dominates over *rc.max_kbps*).

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

4.8 Motion Estimation

me.subdiv *Macroblock subdivision mask*

```
1(0x01,mandatory)=16x16, 2(0x02)=16x8, 4(0x04)=8x16, 8(0x08)=8x8,
16(0x10)=8x4, 32(0x20)=4x8, 64(0x40)=4x4, 127(0x7f)=all
```

Macroblock subdivision flags enable macroblock subdivisions for motion estimation and encoding. Flags can be combined with arithmetic sum.

Currently maximum value is set to 7 (smaller subdivisions are disabled) for optimal performance on higher resolutions.

me.search_range *Maximum motion search range in full pels*

```
-1(default)= automatic calculation from picture size, N[1-64]=
search range in full pels
```

The higher is the range, the slower is the search.

me.max_refs *Number of reference pictures*

```
1= default, N[1-5]= number of reference pictures
```

Number of pictures (frames or fields) used for inter motion search. The higher the number, the slower is the search. For interlaced coding it's recommended to double this value in order to achieve proper quality level.

me.flags *ME flags*

```
0(default)= none,
0x10= ME_LIMIT_B_REFS - Set no. of reference for B-frames to 1 even
if me.max_refs > 1
0x20= ME_DISABLE_PREPROC_ME - Disable motion estimation by reduced
picture
0x40= ME_DISABLE_PREPROC_COMPLEXITY - Disable preproc complexity
calculation
0x1000= ME_USE_FULL_LOG_SEARCH_P - Use more detailed motion
estimation for P frames
0x2000= ME_USE_FULL_LOG_SEARCH_B - Use more detailed motion
estimation for B frames
```

This is a combination of bitwise flags to adjust Motion Estimation process. Generally the user doesn't need to care about it and may leave the default value of 0.

ME_LIMIT_B_REFS flag may present interest when working with interlaced material. For high encoding quality one needs to set *me.max_refs* to 2 (2 reference fields) but this can result in quite serious performance degradation (compared to progressive mode). To compensate for this one can set *me.flags* to 0x010 saving time on numerous B-frames.

Note that 0x20 and 0x40 flags are mostly experimental and are not recommended for usage. If however they are used, the following restrictions should be followed:
1) *rc.scene_change* should be <= 33

2) *gop.min_b_frames* should be equal to *gop.max_b_frames*

3) They shouldn't be used simultaneously

0x1000 and 0x2000 flags will decrease encoder speed but suppose to increase the quality

4.9 Speed modes

speed.i *Encoding speed for I-frames*

```
0= slowest/max. quality, 4= default, 8= fastest
```

speed.p *Encoding speed for P-frames*

```
0= slowest/max. quality, 4= default, 8= fastest
```

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

speed.b *Encoding speed for B-frames*

0= slowest/max. quality, 4= default, 8= fastest

speed.automatic *Enables automatic real time control (experimental)*

0(default)= off, 1= on

When turned on, varies speed settings on-the-fly to achieve optimal real-time performance (CPU load).

Slower speed modes imply using of the RDO (Rate-Distortion Optimization) encoder feature. RDO significantly improves picture quality at the same bitrate values. RDO algorithm optimizes decision-making process of the encoder by doing some sample coding and choosing the best variant among the others.

4.10 Slicing

Slice is a packaging unit for compressed bits. Slices are typically been used to divide a frame into several logical parts to allow for independent asynchronous encoding or network transmission. For example, for error resiliency of the transmitted video it is beneficial to set slice size equal (or a bit smaller) to the size of network packet. The side effect of using slices is that on some smaller resolutions (and lower bitrates) the edges of slices could be seen.

slice.mode *Selects a slice mode*

0(default)= none,
1= no. of MBs per slice,
2= no. of bytes per slice;
3= no. of slices per picture

slice.param *provides appropriate number value for slice.mode*

slice.i_param *provides appropriate number value for slice.mode for I-slices only*

optional, if omitted or zero then *slice.param* is used for I-slices instead

slice.b_param *provides appropriate number value for slice.mode for B-slices only*

optional, if omitted or zero then *slice.param* is used for B-slices instead

4.11 Deblocking

deblock.flag *Loop filter configuration*

0(default)=parameter below ignored, 1=parameters used

deblock.disable *Disables loop filter in slice header*

0= filter enabled, 1= filter disabled

deblock.alpha_c0 *Alpha & C0 offset div. 2*

[-6 - +6]

deblock.beta_c0 *Beta offset div. 2*

[-6 - +6]

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

4.12 Multithreading

This group of settings is for advanced usage only. We recommend to use the default settings when encoder is configured automatically for optimal performance at a given hardware (number of virtual CPUs).

mt.disable *Disables multithreading*

`0(default)= MT on, 1= MT off`

mt.num_threads *No. of worker threads to run*

`0(default)= auto`

mt.max_pict_tasks *Max no. of simultaneously coded pictures*

`(-1)(default)= auto, N[0, 5]= no. of pictures`

mt.max_raw_frames *Maximum number of input frames to hold for async mode*

`0(default)= calculate automatically`

4.13 Video Usability Information (VUI)

This is a supplemental group of parameters specified by the H.264 standard (see Annex E for details). They are written directly into the bitstream for enhanced usage of it. See the sample configuration file (*v4enc.cfg*) for the full VUI parameters list.

The following items are modified by the encoder:

- vui.timing_info_present_flag
- vui.fixed_frame_rate_flag
- vui.time_scale
- vui.num_units_in_tick
- vui.low_delay_hrd_flag
- vui.pic_struct_present_flag
- vui.vcl_hrd_parameters_present_flag
- vui.bitstream_restriction_flag

Alternatively the SDK user may invoke *v4e_set_vui_parameters()* API function to set their own configuration (in this case parameters listed above are ignored).

Encoder allows setting any values in VUI sections. By default it will not put not obligatory parameters in VUI and will calculate proper values for those which can be calculated. For instance, default value **`vui.bitstream_restriction_flag`** is zero. But you can set:

```
vui.bitstream_restriction_flag = 1
```

Encoder will calculate and put into VUI section all related parameters. Among them

```
vui.num_reorder_frames
vui.max_dec_frame_buffering
```

These parameters help decoder to define smallest possible output delay. So it can be very useful in video-conferencing applications.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

4.13.1 Aspect Ratio IDC

vui.aspect_ratio_info_present_flag

Enables aspect ratio IDC

0(default)= disable, 1= enable

vui.aspect_ratio_idc

Aspect ratio IDC (see table below)

0= none,
1-16= predefined values (see table below),
255= custom (vui.sar_width, vui.sar_height)

For custom IDC setting the effective aspect ratio formula takes the form:

*aspect_ratio= (picture_width * sar_width) : (picture_height * sar_height)*

vui.sar_width

Indicates the horizontal size of the sample aspect ratio (in arbitrary units)

Used when *vui.aspect_ratio_idc = 255*

vui.sar_height

Indicates the vertical size of the sample aspect ratio (in the same arbitrary units as sar_width)

Used when *vui.aspect_ratio_idc = 255*

aspect_ratio_idc	Sample aspect ratio	Usage examples
0	Unspecified	
1	1:1 ("square")	1280x720 16:9 frame without overscan 1920x1080 16:9 frame without overscan (cropped from 1920x1088) 640x480 4:3 frame without overscan
2	12:11	720x576 4:3 frame with horizontal overscan 352x288 4:3 frame without overscan
3	10:11	720x480 4:3 frame with horizontal overscan 352x240 4:3 frame without overscan
4	16:11	720x576 16:9 frame with horizontal overscan 540x576 4:3 frame with horizontal overscan
5	40:33	720x480 16:9 frame with horizontal overscan 540x480 4:3 frame with horizontal overscan
6	24:11	352x576 4:3 frame without overscan 480x576 16:9 frame with horizontal overscan
7	20:11	352x480 4:3 frame without overscan 480x480 16:9 frame with horizontal overscan
8	32:11	352x576 16:9 frame without overscan
9	80:33	352x480 16:9 frame without overscan
10	18:11	480x576 4:3 frame with horizontal overscan
11	15:11	480x480 4:3 frame with horizontal overscan
12	64:33	540x576 16:9 frame with horizontal overscan
13	160:99	540x480 16:9 frame with horizontal overscan

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

14	4:3	1440x1080 16:9 frame without horizontal overscan
15	3:2	1280x1080 16:9 frame without horizontal overscan
16	2:1	960x1080 16:9 frame without horizontal overscan
17..254	Reserved	
255	Extended_SAR	

4.14 Error Resilience (ER)

Error resilience refers to mechanisms in the VSS Encoder, that enhance the capability of the compressed bitstream to resist channel errors. Error resilience tools conflict with compression efficiency since they introduce redundancy in the data stream. Therefore proper design requires a tradeoff analysis to select the optimal blend. Please refer to *VSS ER-EC White Paper* document for description of the VSS proprietary ER tools.

er.enable

Enables ER tools

```
0(default)= disabled, 1= enabled
```

It is not possible to enable/disable ER tools on-the-fly

er.initial_expected_loss_percent

Initial expected loss rate (percents)

```
0= no ER, 15= default
```

This value influences the intra update intensity.

er.intra_update_method

Intra update method

```
0= off,
1(default)= adaptive intra refresh (AIR) by
distinguishing the motion areas of the picture,
2= update with rolling intra macroblock rows,
3= update with random intra macroblocks
```

To minimize possible overheads VSS encoder sprinkles intra coded macroblocks in P and B frames rather than inserting complete I-frames according to the method selected.

The recommended adaptive method (1) fights comet tail artifacts by inserting intra macroblocks more frequently to the fast motion regions.

In rolling intra MB rows method (2) one or more horizontal rows of the macroblocks are inserted in the inter frames. The selected lines move downwards from frame to frame by one row.

Random intra MB update method (3) inserts intra coded macroblocks in every inter coded frame, based on the expected packet loss probability estimate. Greater weight is assigned to the central portion of the frame because macroblocks in this area are more likely to be references for other predictions.

er.fast_motion_update_period

Short period (fast motion) for adaptive intra update (frames)

```
0= disabled,
1, 2, 3= recommended
```

Applicable when *er.intra_update_method* = 1 only

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

er.full_motion_update_period *Long period (fast and slow motion) for adaptive intra update (frames)*

```
0= disabled,
5, 6, up to (gop.keyframes/2)= recommended
```

Applicable when *er.intra_update_method* = 1 only

er.total_intra_update_period *period of picture full intra update (frames)*

```
0,1= off, 60= default
```

This parameter is intended to be used for encoding sequences with long key frame intervals (or without keyframes at all) as the addition to the Error Resilience functionality. This tool is also known as *Cyclic Intra Refresh* (CIR). It works only if *er.enable* = 1 and *er.initial_expected_loss_percent* > 0

It's guaranteed that every spatial macroblock position will be intra-refreshed at least once inside series of *er.total_intra_update_period* frames. So the average expected full intra refresh period is equal to *er.total_intra_update_period*. It is guaranteed that full-frame intra refresh period is not greater then 2*er.total_intra_update_period*

4.14.1 Frame slicing

It is recommended to set the slice size equal to network MTU (maximum transmission unit) size minus RTP, UDP and IP headers.

On Windows platform it is recommended to not exceed 1024 bytes for slice size plus RTP header. Standard RTP header is 12 bytes (without extensions). Some versions of Windows introduce quite significant overhead on IP packets larger than 1024 bytes. So 1000 bytes is a reasonable setting for Ethernet conditions.

This is managed by Slicing section parameters.

```
slice.mode = 2          // mode 2: slices are limited by size in bytes
slice.param = 1000      // <=1000 bytes in slice
```

4.14.2 Cyclic Intra Refresh (CIR)

It is recommended to avoid using I-frames with CIR. Please note that CIR period is measured in reference frames, i.e. if you are using B-frames (or emulated B-frames), recalculate the period accordingly taking into account the fact that B-frames are not referenced.

This is managed by Group Of Pictures (GOP) and Error Resilience (ER) sections in the configuration file.

```
gop.keyframes   = 0 // period of I-frames in number of frames (0=only first)
er.enable = 1                          // Enable error resilience
er.initial_expected_loss_percent = 10 // expected loss rate in percents
er.intra_update_method = 0             // 0 - don't use intra update
er.total_intra_update_period = 30   // period for picture full intra update,
                                       // measured in number of reference
frames
```

4.14.3 Adaptive Intra Refresh (AIR)

This is managed by Error Resilience (ER) section in the configuration file.

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.

```
er.enable = 1                          // Enable error resilience
er.initial_expected_loss_percent = 10 // expected loss rate in percents
er.intra_update_method = 1             // 1 = motion adaptive intra update
```

4.14.4 Hierarchical Frame Structure (HFS)

Key pictures are the pictures that reference another key pictures only. Key picture period is measured in reference frames. It is recommended to set the period to three as a good balance between robustness and compression efficiency.

It is recommended to use non-reference frames (B-frames, or emulated B-frames) for improved stream robustness, because their loss doesn't affect the other frames.

This is managed by Multi-Stream Coding (SVC, AVC, MVC) and Group Of Pictures (GOP) sections in the configuration file.

```
svc.num_layers = 0            // 0 = use AVC mode
svc.key_picture_period = 3    // key picture period, in number of reference
frames
gop.bframes     = 1     // number of B-frames between Ps
gop.min_bframes = 1     // adaptive mode is disabled
gop.emulate_b   = 2     // use non-reference uni-directional P frames
```

© Vanguard Software Solutions Inc 1995-2011, All Rights Reserved.