[sys-base]

V4L2 Stateless Video Encoding uAPI Progress Update

Paul Kocialkowski

paulk@sys-base.io

Linux Media Summit
Tuesday May 26th 2026

Stateless Video Codecs Introduction

Types of hardware video codecs:

- **Stateful**: driven by a dedicated microcontroller
  - Mailbox interface for (proprietary) firmware
  - High-level configuration parameters
  - In-firmware (proprietary) rate control implementation
- **Stateless**: driven by the main processor
  - Low-level (register-level) configuration
  - Bitstream generation (all parameters), with hardware limitations
  - On-CPU rate control implementation
  - Memory management (reconstruction, scratch/metadata)

V4L2 support:

- **Stateful encoders and decoders**: using V4L2 controls (simple) and M2M API
- **Stateless decoders**: using V4L2 controls (compound), M2M and Media Request APIs
- **Stateless encoders**: missing upstream support

Current Status

[PATCH 00/14] media: Add V4L2 H.264 stateless encode and VC8000E support

Paul Kocialkowski, 2026-05-22

```
drivers/media/v4l2-core/Kconfig               |    4 +
drivers/media/v4l2-core/Makefile              |    2 +
drivers/media/v4l2-core/v4l2-ctrls-core.c     |   62 +
drivers/media/v4l2-core/v4l2-ctrls-defs.c     |    4 +
drivers/media/v4l2-core/v4l2-h264-enc-rbsp.c  | 1173 +++++++++
drivers/media/v4l2-core/v4l2-h264-enc-rc.c    |  558 +++++
drivers/media/v4l2-core/v4l2-h264-enc.c       | 1322 ++++++++++
drivers/media/v4l2-core/v4l2-h264.c           |   69 +
include/media/v4l2-ctrls.h                    |    2 +
include/media/v4l2-h264-enc-rbsp.h            |   72 +
include/media/v4l2-h264-enc-rc.h              |  108 +
include/media/v4l2-h264-enc.h                 |  135 ++
include/media/v4l2-h264.h                     |  146 ++
include/uapi/linux/v4l2-controls.h            |   33 +
include/uapi/linux/videodev2.h                |    1 +
```

Current Status

[PATCH 00/14] media: Add V4L2 H.264 stateless encode and VC8000E support

Paul Kocialkowski, 2026-05-22

```
arch/arm64/boot/dts/freescale/imx8mp.dtsi     |   11 +
drivers/media/platform/verisilicon/Kconfig    |    1 +
drivers/media/platform/verisilicon/Makefile   |    2 +
drivers/media/platform/verisilicon/hantro.h   |   17 +
.../media/platform/verisilicon/hantro_drv.c   |  180 +-
.../media/platform/verisilicon/hantro_h264.c  |    6 +-
.../media/platform/verisilicon/hantro_hw.h    |   28 +
.../media/platform/verisilicon/hantro_v4l2.c  |  123 +-
.../platform/verisilicon/hantro_vc8000e.c     |   68 +
.../verisilicon/hantro_vc8000e_h264_enc.c     |  883 +++++++
.../verisilicon/hantro_vc8000e_regs.h         | 2129 ++++++++++++++++++
.../media/platform/verisilicon/imx8m_vpu_hw.c |  113 +
```

General Philosophy

Design Goals:

- Reuse the existing stateful uAPI as much as possible
- Hardware-agnostic uAPI
- Accomodation rather than combinatory capabilities reporting

Architecture:

- Bitstream headers generated by the kernel
- Rate control implemented in the kernel
- References marked by userspace, tracked by the kernel
- Hardware-specific limitations handled by the kernel
- Final parameters returned to userspace

uAPI: Warmup

Capabilities:

- Fully compatible with stateful uAPI
- `VIDIOC_ENUM_FMT`
- `VIDIOC_ENUM_FRAMESIZES`
- `VIDIOC_ENUM_FRAMEINTERVALS`

Initialization:

- Fully compatible with stateful uAPI
- `VIDIOC_S_FMT` capture (coded) then output (raw)
- `VIDIOC_S_PARM` `timeperframe` for frame rate
- `VIDIOC_S_SELECTION` for unaligned crop

uAPI: Pixel Format

Existing formats:

- V4L2_PIX_FMT_H264:
  - *The encoder generates one Access Unit per buffer.*
  - Good fit for stateless encode, but...
  - Stateful decoders are detected from this pixel format
- V4L2_PIX_FMT_H264_SLICE:
  - *H264 parsed slice data, including slice headers, either with or without the start code*
  - V4L2_CID_STATELESS_H264_DECODE_MODE, V4L2_CID_STATELESS_H264_START_CODE
  - Not good for stateless encode, but...
  - Currently used and abusing definition

Proposal for discussion:

- Codec pixel format indicates mode of operation, not just data type
- Keep V4L2_PIX_FMT_H264 for stateful
- Rename V4L2_PIX_FMT_H264_SLICE to V4L2_PIX_FMT_H264_SL
  some codecs don't have slices anyway

uAPI: Parameters

- Core:
  - `V4L2_CID_STATELESS_H264_SPS`: decode SPS
  - `V4L2_CID_STATELESS_H264_PPS`: encode PPS
  - `V4L2_CID_STATELESS_H264_ENCODE_PARAMS`: slice header and more (new)
- RBSP:
  - `V4L2_CID_MPEG_VIDEO_H264_VUI_*`
  - `V4L2_CID_STATELESS_H264_START_CODE`
  - `V4L2_CID_MPEG_VIDEO_PREPEND_SPSPPS_TO_IDR`
  - `V4L2_CID_MPEG_VIDEO_AU_DELIMITER`
- Rate control:
  - `V4L2_CID_MPEG_VIDEO_FRAME_RC_ENABLE`
  - `V4L2_CID_MPEG_VIDEO_BITRATE_MODE`
  - `V4L2_CID_MPEG_VIDEO_BITRATE,CONSTANT_QUALITY`
  - `V4L2_CID_MPEG_VIDEO_H264_{I,P,B}_FRAME_(MIN/MAX_)QP`

uAPI: Encode

Operation:

- Mostly compatible with stateful uAPI
- Use of media request for buffer/ctrl association
- Not necessarily 1:1 output to capture buffer (P frames, etc)
- `VIDIOC_{QBUF,DQBUF}`

Feedback:

- Controls updated with actual values used in the bitstream
- Best attached to the incoming media request (missing support)
- Buffer flags:
  - `V4L2_BUF_FLAG_{KEYFRAME,PFRAME,BFRAME}`
  - `V4L2_BUF_FLAG_ERROR`

uAPI: References

Process:

- Sliding window reference marking process (simple)
- Reference indication with `nal_ref_idc`, long-term indication with `V4L2_H264_ENCODE_FLAG_LONG_TERM_REFERENCE` in encode params
- Kernel-side tracking of DPB and L0/L1

Discussion:

- Adaptive reference management (`MMCO`) and list modification (`RPLM`) are useful
- May not be supported by hardware (e.g. hard slice header)
- Explicit DPB and L0/L1 needs to be validated and translated to `MMCO` + `RPLM`
- Array of operations is easier (userspace can diff and validate)
- Feedback of used DPB and L0/L1

uAPI: Missing Features

- Attaching controls to incoming media request for feedback
- Average QP return: `V4L2_CID_MPEG_VIDEO_AVERAGE_QP` (mutex)
- Reference modification: `MMCO` (adaptive), `RPLM`
- Reference feedback: DPB and L0/L1
- Slice type support indication:
  - Maybe read-only `V4L2_CID_MPEG_VIDEO_B_FRAMES`
  - Maybe dedicated control with P/B support flag and number
- Multi-slice support: `V4L2_CID_MPEG_VIDEO_MULTI_SLICE_*`
  - Parameters from `V4L2_CID_STATELESS_H264_ENCODE_PARAMS` for all slices
- Frame skipping:
  - `V4L2_CID_MPEG_VIDEO_FRAME_SKIP_MODE`
  - `V4L2_CID_MPEG_VIDEO_TEMPORAL_DECIMATION`
  - `V4L2_CID_MPEG_VIDEO_VBV_SIZE`
- Rate control implementation selection
- HRD conformance (no controls?)
- Drain and resolution change: `VIDIOC_ENCODER_CMD`
- Documentation

Implementation: Core

- Files:
  - `drivers/media/v4l2-core/v4l2-h264-enc.c`
  - `include/media/v4l2-h264-enc.h`
- Functions:
  - `v4l2_h264_enc_init`: Init from driver at `start_streaming` with: feature flags, format, ops, timeperframe, ctrl handler pointers
  - `v4l2_h264_enc_exit`: Exit from driver at `stop_streaming`
  - `v4l2_h264_enc_step`: Frame step from driver at `device_run`
  - `v4l2_h264_enc_complete`: Frame complete from driver (IRQ context), set buffer flags
- Hardware feature flags:
  - `V4L2_H264_ENC_FLAG_INTER_PRED/BIPRED`
  - `V4L2_H264_ENC_FLAG_HW_AUD/SPS/PPS/SLICE_HEADER`
  - `V4L2_H264_ENC_FLAG_CHROMA_QP_CR_OFFSET`

Implementation: Core

- Structures:
  - `struct v4l2_h264_enc`: Top-level, pre-filled by drivers
  - `struct v4l2_h264_enc_state`: Ready-to-go parameters
  - `struct v4l2_h264_enc_ops`: Driver callbacks
- State management:
  - Next state prepared from controls and cross-validated at each step
  - Next state constrained via `state_constrain` driver callback
  - Next state commited as new active state
  - Active state available to driver for parameters

Implementation: RBSP

- Files:
  - drivers/media/v4l2-core/v4l2-h264-enc-rbsp.c
  - include/media/v4l2-h264-enc-rbsp.h
  - include/media/v4l2-h264.h
- Functions:
  - v4l2_h264_enc_rbsp_init
  - v4l2_h264_enc_rbsp_{bytes,bits}_count
  - v4l2_h264_enc_exit: Exit by driver at stop_streaming
  - v4l2_h264_enc_rbsp_{start_code,aud}
  - v4l2_h264_enc_rbsp_{sps,pps,slice_header}

Implementation: RBSP

- Structures:
  - `struct v4l2_h264_enc_rbsp`: Top-level
  - `struct v4l2_h264_enc_rbsp_ops`: Driver callbacks for hardware operations `bits_raw` (no EPTB), `bits` (EPTB), `ue`, `se` (exp-Golomb), `align`
- Usage:
  - Not tied to H.264 encode core, can be used by stateful drivers
  - Optional low-level hardware operation or generic virtual memory access
  - Managed by H.264 encode core, ops provided by driver
  - Produced when needed from active vs next state comparison and controls
  - Supports SPS video part including VUI

Implementation: References

- Files:
  - `drivers/media/v4l2-core/v4l2-h264-enc.c`
  - `include/media/v4l2-h264-enc.h`
- Structures:
  - `struct v4l2_h264_enc_ref`: DPB, L0/L1 and buffers
  - `struct v4l2_h264_enc_rec_buffer`: Reconstruction buffer
- Reference tracking:
  - Kernel-side DPB maintenance following the sliding window reference marking process
  - Existing stateless decode reflist builder for L0/L1, new generic entrypoint
- Reconstruction buffers management:
  - Allocated and freed via `rec_buffer_{alloc,free}` driver callbacks
  - Initial allocation of slots from `ref_slots_count_init` driver parameter or `V4L2_H264_NUM_DPB_ENTRIES / 2`
  - Only increased during stream lifetime

Implementation: Rate Control

- Files:
  - `drivers/media/v4l2-core/v4l2-h264-enc-rc.c`
  - `include/media/v4l2-h264-enc-rc.h`
- Functions:
  - `v4l2_h264_enc_rc_init`
  - `v4l2_h264_enc_rc_exit`
  - `v4l2_h264_enc_rc_mode_update`: Detected and called by core
  - `v4l2_h264_enc_rc_step`: Apply rate control strategy and determine QP
  - `v4l2_h264_enc_rc_complete`: Keep track of stats
- Structures:
  - `struct v4l2_h264_enc_rc`: Top-level
  - `struct v4l2_h264_enc_rc_mode`: Common rate control mode operations
    `init`, `exit`, `measure` (size), `estimate` (QP), `complete` (size)
  - `struct v4l2_h264_enc_rc_ops`: Driver-specific rate control mode operations
  - `struct v4l2_h264_enc_rc_stats`: Tracking of previous frames (type, QP, size, target)

Implementation: Rate Control

Direct mode:

- Return QP from controls

Constant Quality (CQ):

- Linear quality to QP mapping, per-type range
- Non-linear LUT is probably better

Constant Bitrate (CBR):

- Measurement:
  - Frame size from remaining buget of sliding window average bitrate target
  - Boost for intra frames ($\times\ 8$), penalty for inter frames ($\times\ 0.8$), overshoot protection ($\times\ 0.8$)
- Estimation:
  - Initial size to QP from static table
  - Normal size to QP from stats, QP delta from target to captured size ratio
  - Average of last 3 captures with penalty weighting (temporal, size diff)

Implementation: Missing Features

- Rate control implementation selection
  - Driver-specific implementations
  - Generic implementations, maybe different approaches
  - eBPF, maybe modules too?
- Reflist builder API rework
- Profile/level features validation/conformance
- Interlaced field support
- Multiple SPS/PPS support
- `pic_order_cnt_type > 0`
- `gaps_in_frame_num_allowed_flag` validation
- RBSP sequence/stream and filler NALUs
- Documentation

Hantro VC8000E Driver

Integration:

- Target hardware: NXP i.MX8MP
- Registers in memory (as structs) and copied
  HEVC and JPEG definitions included
- Pixel formats: `YUYV/UYVY`, `NV12/NV12M`, `YUV420/YUV420M`

Missing features:

- Second reference slot
- QP-based lambda tables calculation
- Intra area (cyclic intra refresh)
- ROI QP delta
- More pixel formats

Userspace

- GStreamer implementation: `gst-plugins-bad`, `v4l2codecs`
  `https://github.com/paulkocialkowski/gstreamer.git`
- Working and usable, more testing welcome
- Example pipelines:

```
$ gst-launch-1.0 videotestsrc ! v4l2slh264enc ! ! fakesink
$ gst-launch-1.0 videotestsrc pattern=smpte num-buffers=150 !
  v4l2slh264enc rate-control=cbr qp-min=8 bitrate=8000000 !
  h264parse ! matroskamux ! filesink location=encode.mkv
```

- Debug:

```
$ export GST_DEBUG=4,v4l2codecs-h264enc:6,v4l2codecs-encoder:7,
  videoencoder:6
```