V4L2 API Ambiguities and Improvements

Hans Verkuil

Cisco Systems Norway

Split off controls

Split off the control part from videodev2.h. Controls are almost 30% of videodev2.h (and growing). Maintaining controls would be easier if they are moved to e.g. linux/v4l2-controls.h which is included by videodev2.h.

Should driver-specific controls have unique IDs?

Currently there are three types of controls: standard controls, controls that are specific to a chipset (e.g. cx2341x, mfc51) and driver-specific controls. The controls of the first two types have well defined and unique IDs. For driver-specific controls however there are no clear rules.

It all depends on one question: should driver-specific controls have a unique control ID as well, or can they overlap with other drivers?

If the answer is that they should be unique as well, then all driver-specific controls will have to be defined in a single header so you can be certain that there is no overlap.

If the answer is that they may overlap, then each driver can either define their controls inside their own driver, or in a driver-specific public header.

STREAMON/OFF

What should VIDIOC_STREAMON/OFF do if the stream is already started/stopped? I believe they should do nothing and just return 0. The main reason for that is it can be useful if an application can just call VIDIOC_STREAMOFF without having to check whether streaming is in progress.

Unsupported formats in TRY/S_FMT

What should a driver return in TRY_FMT/S_FMT if the requested format is not supported (possible behaviours include returning the currently selected format or a default format).

The spec says this: "Drivers should not return an error code unless the input is ambiguous", but it does not explain what constitutes an ambiguous input. In my opinion TRY/S_FMT should never return an error other than EINVAL (if the buffer type is unsupported) or EBUSY (for S_FMT if streaming is in progress).

Should we make a recommendation whether the currently selected format or a default format should be returned?

One proposal is to just return a default format if the requested pixelformat is unsupported. Returning the currently selected format leads to inconsistent results.

Empty bus_info?

VIDIOC_QUERYCAP allows bus_info to be empty. Since the purpose of bus_info is to distinguish multiple identical devices this makes no sense. I propose to make the spec more strict and require that bus_info is always filled in with a unique string.

Deprecate V4L2_BUF_TYPE_PRIVATE

None of the kernel drivers use it, and I cannot see any good use-case for this. If some new type of buffer is needed, then that should be added instead of allowing someone to abuse this buffer type.

Buftype checks against node type

A driver that has both a video node and a vbi node using the same struct v4l2_ioctl_ops for both nodes will have to check in e.g. vidioc_g_fmt_vid_cap or vidioc_g_fmt_vbi_cap whether it is called from the correct node (video or vbi) and return an error if it isn't.

Should such checks be added to the V4L2 core? And if so, should we add some additional VFL types? Currently we have GRABBER (video nodes), VBI, RADIO and SUBDEV. But if we want to do proper core checks, then we would also need OUTPUT, VBI_OUT and M2M.

We could also refuse ioctls based on the VFL_TYPE: e.g. VIDIOC_ENCODER_CMD makes no sense for radio/vbi nodes. Easy to implement with the lookup table.

Remove experimental tag

Remove the experimental tag from the following old drivers:

- VIDEO_TLV320AIC23B
- USB_STKWEBCAM
- VIDEO_CX18
- VIDEO_CX18_ALSA
- VIDEO_ZORAN_AVS6EYES
- DVB_USB_AF9005
- MEDIA_TUNER_TEA5761

Removing this tag from these drivers might be too soon, though:

- VIDEO_NOON010PC30
- VIDEO_OMAP3

Wrong timings API use for input

What should VIDIOC_G_STD/DV_PRESET/DV_TIMINGS return if the current input or output does not support that particular timing approach? EINVAL? ENODATA?

This is relevant for the case where a driver has multiple inputs/outputs where some are SDTV (and support the STD API) and others are HDTV (and support the DV_TIMINGS API).

I propose ENODATA.

Add new defines

Add these new defines:

```
#define V4L2_IN_CAP_TIMINGS V4L2_IN_CAP_CUSTOM_TIMINGS
#define V4L2_OUT_CAP_TIMINGS V4L2_OUT_CAP_CUSTOM_TIMINGS
```

Since DV_TIMINGS is now used for any HDTV timings and no longer just for custom, non-standard timings, the word "CUSTOM" is no longer appropriate.

Sequence and timestamp for output

What should video output drivers do with the sequence and timestamp fields when they return a v4l2_buffer from VIDIOC_DQBUF?

I think the spec is clear with respect to the timestamp: "The driver stores the time at which the first data byte was actually sent out in the timestamp field." For sequence the spec just says: "Set by the driver, counting the frames in the sequence."

So I think that output drivers should indeed set both sequence and timestamp.

Const arg for write-only ioctls

Make the argument of write-only ioctls const in v4l2-ioctls.h. This makes it obvious to drivers that they shouldn't change the contents of the input struct since it won't make it back to userspace. It also simplifies v4l2-ioctl.c since it can rely on the fact that after the ioctl call the contents of the struct hasn't changed. Right now the struct contents is logged (if debugging is on) before the ioctl call for write-only ioctls

How to set the timestamp?

What is the right/best way to set the timestamp? The spec says gettimeofday, but is it my understanding that ktime_get_ts is much more efficient. Some drivers are already using ktime_get_ts.

Options:

a) all drivers must comply to the spec and use gettimeofday

b) we change the spec and all drivers must use the more efficient ktime_get_ts

c) we add a buffer flag V4L2_BUF_FLAG_MONOTONIC to tell userspace that a monotonic clock like ktime_get_ts is used and all drivers that use ktime_get_ts should set that flag.

If we go for c, then we should add a recommendation to use one or the other as the preferred timestamp for new drivers.

Device Specified Clock

Some devices provide their own timestamp in device clock units. How should that be passed on to userspace?

Proposal: extend the struct v4l2_timecode in v4l2_buffer, either by just adding a new type, or by creating an anonymous union.

Currently there are no drivers that use v4l2_timecode.

It is questionable for v4l2_timecode to be in v4l2_buffer at all since such information will typically be carried over VBI.

V4L2_CAP_VIDEO_CAPTURE

If a driver supports only formats with more than one plane, should V4L2_CAP_VIDEO_CAPTURE still be defined? And if a driver also supports single-plane formats in addition to >1 plane formats, should V4L2_CAP_VIDEO_CAPTURE be compulsary?

The consensus seems to be 'No' and 'No'. The CAPs refer to whether the single and/or multiplanar API is implemented, not whether single and/or multiplanar formats are available.

V4L2_CROPCAP

VIDIOC_CROPCAP: the spec says that CROPCAP must be implemented by all capture and output devices (Section "Image Cropping, Inserting and Scaling"). In reality only a subset of the drivers support cropcap.

Should cropcap really be compulsory? Or only for drivers that can scale? And in that case, should we make a default implementation for those drivers that do not support it? (E.g.: call g_fmt and use the width/height as the default and bounds rectangles, and set the pixel aspect to 1/1)

Pixel Aspect Ratio

Pixel aspect: currently this is only available through VIDIOC_CROPCAP. It never really belonged to VIDIOC_CROPCAP IMHO. It's just not a property of cropping/composing. It really belongs to the input/output timings (STD or DV_TIMINGS). That's where the pixel aspect ratio is determined.

While it is possible to add it to the dv_timings struct, I see no way of cleanly adding it to struct v4l2_standard (mostly because VIDIOC_ENUMSTD is now handled inside the V4L2 core and doesn't call the drivers anymore).

An alternative is to add it to struct v4l2_input/output, but I don't know if it is possible to defined a pixelaspect for inputs that are not the current input.

What I am thinking of is just to add a new ioctl for this VIDIOC_G_PIXELASPECT.

Tuner Ownership

How to handle tuner ownership if both a video and radio node share the same tuner?

Obvious rules:

- Calling S_FREQ, S_TUNER, S_MODULATOR or S_HW_FREQ_SEEK will make the filehandle the owner if possible. EBUSY is returned if someone else owns the tuner and you would need to switch the tuner mode.
- Ditto for ioctls that expect a valid tuner configuration like QUERYSTD. This is likely to be driver dependent, though.
- Just opening a device node should not switch ownership

Tuner Ownership

But it is not clear what to do when any of these ioctls are called:

- G_FREQUENCY: could just return the last set frequency for radio or TV: requires that that is remembered when switching ownership. This is what happens today, so G_FREQUENCY does not have to switch ownership.
- G_TUNER: the rxsubchans, signal and afc fields all require ownership of the tuner. So in principle you would want to switch ownership when G_TUNER is called. On the other hand, that would mean that calling v4l2-ctl --all -d /dev/radio0 would change tuner ownership to radio for /dev/video0. That's rather unexpected.

  It is possible to just set rxsubchans, signal and afc to 0 if the device node doesn't own the tuner. I'm inclined to do that.
- Should closing a device node switch ownership? E.g. if nobody has a radio device open, should the tuner switch back to TV mode automatically? I don't think it should.
- How about hybrid tuners?