Extended Controls API

Introduction

The control mechanism as originally designed was meant to be used for user settings (brightness, saturation, etc). However, it turned out to be a very useful model for implementing more complicated driver APIs where each driver implements only a subset of a larger API.

The MPEG encoding API was the driving force behind designing and implementing this extended control mechanism: the MPEG standard is quite large and the currently supported hardware MPEG encoders each only implement a subset of this standard. Further more, many parameters relating to how the video is encoded into an MPEG stream are specific to the MPEG encoding chip since the MPEG standard only defines the format of the resulting MPEG stream, not how the video is actually encoded into that format.

Unfortunately, the original control API lacked some features needed for these new uses and so it was extended into the (not terribly originally named) extended control API.

Even though the MPEG encoding API was the first effort to use the Extended Control API, nowadays there are also other classes of Extended Controls, such as Camera Controls and FM Transmitter Controls. The Extended Controls API as well as all Extended Controls classes are described in the following text.

The Extended Control API

Three new ioctls are available: :ref:`VIDIOC_G_EXT_CTRLS <VIDIOC_G_EXT_CTRLS>`, :ref:`VIDIOC_S_EXT_CTRLS <VIDIOC_G_EXT_CTRLS>` and :ref:`VIDIOC_TRY_EXT_CTRLS <VIDIOC_G_EXT_CTRLS>`. These ioctls act on arrays of controls (as opposed to the :ref:`VIDIOC_G_CTRL <VIDIOC_G_CTRL>` and :ref:`VIDIOC_S_CTRL <VIDIOC_G_CTRL>` ioctls that act on a single control). This is needed since it is often required to atomically change several controls at once.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l)extended-controls.rst`, **line 41**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l)extended-controls.rst`, **line 41**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l)extended-controls.rst`, **line 41**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l)extended-controls.rst`, **line 41**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l)extended-controls.rst`, **line 41**); ***backlink***

Unknown interpreted text role "ref".

Each of the new ioctls expects a pointer to a struct :c:type:`v4l2_ext_controls`. This structure contains a pointer to the control array, a count of the number of controls in that array and a control class. Control classes are used to group similar controls into a single class. For example, control class `V4L2_CTRL_CLASS_USER` contains all user controls (i. e. all controls that can also be set using the old :ref:`VIDIOC_S_CTRL <VIDIOC_G_CTRL>` ioctl). Control class `V4L2_CTRL_CLASS_CODEC` contains controls relating to codecs.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 51); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 51); *backlink*

Unknown interpreted text role "ref".

All controls in the control array must belong to the specified control class. An error is returned if this is not the case.

It is also possible to use an empty control array (`count` == 0) to check whether the specified control class is supported.

The control array is a struct :c:type:`v4l2_ext_control` array. The struct :c:type:`v4l2_ext_control` is very similar to struct :c:type:`v4l2_control`, except for the fact that it also allows for 64-bit values and pointers to be passed.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 67); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 67); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 67); *backlink*

Unknown interpreted text role "c:type".

Since the struct :c:type:`v4l2_ext_control` supports pointers it is now also possible to have controls with compound types such as N-dimensional arrays and/or structures. You need to specify the `V4L2_CTRL_FLAG_NEXT_COMPOUND` when enumerating controls to actually be able to see such compound controls. In other words, these controls with compound types should only be used programmatically.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 73); *backlink*

Unknown interpreted text role "c:type".

Since such compound controls need to expose more information about themselves than is possible with :ref:`VIDIOC_QUERYCTRL <VIDIOC_QUERYCTRL>` the :ref:`VIDIOC_QUERY_EXT_CTRL <VIDIOC_QUERYCTRL>` ioctl was added. In particular, this ioctl gives the dimensions of the N-dimensional array if this control consists of more than one element.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 80); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master) (Documentation) (userspace-api) (media) (v4l) extended-controls.rst`, line 80); *backlink*

Unknown interpreted text role "ref".

Note

1. It is important to realize that due to the flexibility of controls it is necessary to check whether the control you want to set actually is supported in the driver and what the valid range of values is. So use :ref:`VIDIOC_QUERYCTRL` to check this.

   System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)extended-controls.rst, line 88); *backlink*

   Unknown interpreted text role "ref".

2. It is possible that some of the menu indices in a control of type `V4L2_CTRL_TYPE_MENU` may not be supported (`VIDIOC_QUERYMENU` will return an error). A good example is the list of supported MPEG audio bitrates. Some drivers only support one or two bitrates, others support a wider range.

All controls use machine endianness.

Enumerating Extended Controls

The recommended way to enumerate over the extended controls is by using :ref:`VIDIOC_QUERYCTRL` in combination with the `V4L2_CTRL_FLAG_NEXT_CTRL` flag:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)extended-controls.rst, line 105); *backlink*

Unknown interpreted text role "ref".

```
struct v4l2_queryctrl qctrl;

qctrl.id = V4L2_CTRL_FLAG_NEXT_CTRL;
while (0 == ioctl (fd, VIDIOC_QUERYCTRL, &qctrl)) {
    /* ... */
    qctrl.id |= V4L2_CTRL_FLAG_NEXT_CTRL;
}
```

The initial control ID is set to 0 ORed with the `V4L2_CTRL_FLAG_NEXT_CTRL` flag. The `VIDIOC_QUERYCTRL` ioctl will return the first control with a higher ID than the specified one. When no such controls are found an error is returned.

If you want to get all controls within a specific control class, then you can set the initial `qctrl.id` value to the control class and add an extra check to break out of the loop when a control of another control class is found:

```
qctrl.id = V4L2_CTRL_CLASS_CODEC | V4L2_CTRL_FLAG_NEXT_CTRL;
while (0 == ioctl(fd, VIDIOC_QUERYCTRL, &qctrl)) {
    if (V4L2_CTRL_ID2CLASS(qctrl.id) != V4L2_CTRL_CLASS_CODEC)
        break;
    /* ... */
    qctrl.id |= V4L2_CTRL_FLAG_NEXT_CTRL;
}
```

The 32-bit `qctrl.id` value is subdivided into three bit ranges: the top 4 bits are reserved for flags (e. g. `V4L2_CTRL_FLAG_NEXT_CTRL`) and are not actually part of the ID. The remaining 28 bits form the control ID, of which the most significant 12 bits define the control class and the least significant 16 bits identify the control within the control class. It is guaranteed that these last 16 bits are always non-zero for controls. The range of 0x1000 and up are reserved for driver-specific controls. The macro `V4L2_CTRL_ID2CLASS(id)` returns the control class ID based on a control ID.

If the driver does not support extended controls, then `VIDIOC_QUERYCTRL` will fail when used in combination with `V4L2_CTRL_FLAG_NEXT_CTRL`. In that case the old method of enumerating control should be used (see :ref:`enum_all_controls`). But if it is supported, then it is guaranteed to enumerate over all controls, including driver-private controls.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)extended-controls.rst, line 151); *backlink*

Unknown interpreted text role "ref".

Creating Control Panels

It is possible to create control panels for a graphical user interface where the user can select the various controls. Basically you will have to iterate over all controls using the method described above. Each control class starts with a control of type `V4L2_CTRL_TYPE_CTRL_CLASS`. `VIDIOC_QUERYCTRL` will return the name of this control class which can be used as the title of a tab page within a control panel.

The flags field of struct :ref:`v4l2_queryctrl <v4l2-queryctrl>` also contains hints on the behavior of the control. See the :ref:`VIDIOC_QUERYCTRL` documentation for more details.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)extended-controls.rst`, **line 170**); ***backlink***

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)extended-controls.rst`, **line 170**); ***backlink***

Unknown interpreted text role "ref".