User Controls

Devices typically have a number of user-settable controls such as brightness, saturation and so on, which would be presented to the user on a graphical user interface. But, different devices will have different controls available, and furthermore, the range of possible values, and the default value will vary from device to device. The control ioctls provide the information and a mechanism to create a nice user interface for these controls that will work correctly with any device.

All controls are accessed using an ID value. V4L2 defines several IDs for specific purposes. Drivers can also implement their own custom controls using `V4L2_CID_PRIVATE_BASE` [1] and higher values. The pre-defined control IDs have the prefix `V4L2_CID_`, and are listed in :ref:`control-id`. The ID is used when querying the attributes of a control, and when getting or setting the current value.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst, line 18); *backlink***
>
> Unknown interpreted text role "ref".

Generally applications should present controls to the user without assumptions about their purpose. Each control comes with a name string the user is supposed to understand. When the purpose is non-intuitive the driver writer should provide a user manual, a user interface plug-in or a driver specific panel application. Predefined IDs were introduced to change a few controls programmatically, for example to mute a device during a channel switch.

Drivers may enumerate different controls after switching the current video input or output, tuner or modulator, or audio input or output. Different in the sense of other bounds, another default and current value, step size or other menu items. A control with a certain *custom* ID can also change name and type.

If a control is not applicable to the current configuration of the device (for example, it doesn't apply to the current video input) drivers set the `V4L2_CTRL_FLAG_INACTIVE` flag.

Control values are stored globally, they do not change when switching except to stay within the reported bounds. They also do not change e. g. when the device is opened or closed, when the tuner radio frequency is changed or generally never without application request.

V4L2 specifies an event mechanism to notify applications when controls change value (see :ref:`VIDIOC_SUBSCRIBE_EVENT`, event `V4L2_EVENT_CTRL`), panel applications might want to make use of that in order to always reflect the correct control value.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst, line 48); *backlink***
>
> Unknown interpreted text role "ref".

All controls use machine endianness.

Control IDs

`V4L2_CID_BASE`

First predefined ID, equal to `V4L2_CID_BRIGHTNESS`.

`V4L2_CID_USER_BASE`

Synonym of `V4L2_CID_BASE`.

`V4L2_CID_BRIGHTNESS (integer)`

Picture brightness, or more precisely, the black level.

`V4L2_CID_CONTRAST (integer)`

Picture contrast or luma gain.

`V4L2_CID_SATURATION (integer)`

Picture color saturation or chroma gain.

`V4L2_CID_HUE (integer)`

Hue or color balance.

`V4L2_CID_AUDIO_VOLUME (integer)`

Overall audio volume. Note some drivers also provide an OSS or ALSA mixer interface.

V4L2_CID_AUDIO_BALANCE (integer)

Audio stereo balance. Minimum corresponds to all the way left, maximum to right.

V4L2_CID_AUDIO_BASS (integer)

Audio bass adjustment.

V4L2_CID_AUDIO_TREBLE (integer)

Audio treble adjustment.

V4L2_CID_AUDIO_MUTE (boolean)

Mute audio, i. e. set the volume to zero, however without affecting V4L2_CID_AUDIO_VOLUME. Like ALSA drivers, V4L2 drivers must mute at load time to avoid excessive noise. Actually the entire device should be reset to a low power consumption state.

V4L2_CID_AUDIO_LOUDNESS (boolean)

Loudness mode (bass boost).

V4L2_CID_BLACK_LEVEL (integer)

Another name for brightness (not a synonym of V4L2_CID_BRIGHTNESS). This control is deprecated and should not be used in new drivers and applications.

V4L2_CID_AUTO_WHITE_BALANCE (boolean)

Automatic white balance (cameras).

V4L2_CID_DO_WHITE_BALANCE (button)

This is an action control. When set (the value is ignored), the device will do a white balance and then hold the current setting. Contrast this with the boolean V4L2_CID_AUTO_WHITE_BALANCE, which, when activated, keeps adjusting the white balance.

V4L2_CID_RED_BALANCE (integer)

Red chroma balance.

V4L2_CID_BLUE_BALANCE (integer)

Blue chroma balance.

V4L2_CID_GAMMA (integer)

Gamma adjust.

V4L2_CID_WHITENESS (integer)

Whiteness for grey-scale devices. This is a synonym for V4L2_CID_GAMMA. This control is deprecated and should not be used in new drivers and applications.

V4L2_CID_EXPOSURE (integer)

Exposure (cameras). [Unit?]

V4L2_CID_AUTOGAIN (boolean)

Automatic gain/exposure control.

V4L2_CID_GAIN (integer)

Gain control.

Primarily used to control gain on e.g. TV tuners but also on webcams. Most devices control only digital gain with this control but on some this could include analogue gain as well. Devices that recognise the difference between digital and analogue gain use controls V4L2_CID_DIGITAL_GAIN and V4L2_CID_ANALOGUE_GAIN.

V4L2_CID_HFLIP (boolean)

Mirror the picture horizontally.

V4L2_CID_VFLIP (boolean)

Mirror the picture vertically.

V4L2_CID_POWER_LINE_FREQUENCY (enum)

Enables a power line frequency filter to avoid flicker. Possible values for enum v4l2_power_line_frequency are:

V4L2_CID_POWER_LINE_FREQUENCY_DISABLED	0
V4L2_CID_POWER_LINE_FREQUENCY_50HZ	1
V4L2_CID_POWER_LINE_FREQUENCY_60HZ	2
V4L2_CID_POWER_LINE_FREQUENCY_AUTO	3

V4L2_CID_HUE_AUTO (boolean)

Enables automatic hue control by the device. The effect of setting `V4L2_CID_HUE` while automatic hue control is enabled is undefined, drivers should ignore such request.

`V4L2_CID_WHITE_BALANCE_TEMPERATURE (integer)`

This control specifies the white balance settings as a color temperature in Kelvin. A driver should have a minimum of 2800 (incandescent) to 6500 (daylight). For more information about color temperature see Wikipedia.

`V4L2_CID_SHARPNESS (integer)`

Adjusts the sharpness filters in a camera. The minimum value disables the filters, higher values give a sharper picture.

`V4L2_CID_BACKLIGHT_COMPENSATION (integer)`

Adjusts the backlight compensation in a camera. The minimum value disables backlight compensation.

`V4L2_CID_CHROMA_AGC (boolean)`

Chroma automatic gain control.

`V4L2_CID_CHROMA_GAIN (integer)`

Adjusts the Chroma gain control (for use when chroma AGC is disabled).

`V4L2_CID_COLOR_KILLER (boolean)`

Enable the color killer (i. e. force a black & white image in case of a weak video signal).

`V4L2_CID_COLORFX (enum)`

Selects a color effect. The following values are defined:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst, line 203)

Unknown directive type "tabularcolumns".

```
.. tabularcolumns:: |p{5.7cm}|p{11.8cm}|
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst, line 205)

Unknown directive type "flat-table".

```
.. flat-table::
    :header-rows:  0
    :stub-columns: 0
    :widths: 11 24

    * - ``V4L2_COLORFX_NONE``
      - Color effect is disabled.
    * - ``V4L2_COLORFX_ANTIQUE``
      - An aging (old photo) effect.
    * - ``V4L2_COLORFX_ART_FREEZE``
      - Frost color effect.
    * - ``V4L2_COLORFX_AQUA``
      - Water color, cool tone.
    * - ``V4L2_COLORFX_BW``
      - Black and white.
    * - ``V4L2_COLORFX_EMBOSS``
      - Emboss, the highlights and shadows replace light/dark boundaries
        and low contrast areas are set to a gray background.
    * - ``V4L2_COLORFX_GRASS_GREEN``
      - Grass green.
    * - ``V4L2_COLORFX_NEGATIVE``
      - Negative.
    * - ``V4L2_COLORFX_SEPIA``
      - Sepia tone.
    * - ``V4L2_COLORFX_SKETCH``
      - Sketch.
    * - ``V4L2_COLORFX_SKIN_WHITEN``
      - Skin whiten.
    * - ``V4L2_COLORFX_SKY_BLUE``
      - Sky blue.
    * - ``V4L2_COLORFX_SOLARIZATION``
      - Solarization, the image is partially reversed in tone, only color
        values above or below a certain threshold are inverted.
    * - ``V4L2_COLORFX_SILHOUETTE``
      - Silhouette (outline).
```

```
* - ``V4L2_COLORFX_VIVID``
  - Vivid colors.
* - ``V4L2_COLORFX_SET_CBCR``
  - The Cb and Cr chroma components are replaced by fixed coefficients
    determined by ``V4L2_CID_COLORFX_CBCR`` control.
* - ``V4L2_COLORFX_SET_RGB``
  - The RGB components are replaced by the fixed RGB components determined
    by ``V4L2_CID_COLORFX_RGB`` control.
```

`V4L2_CID_COLORFX_RGB (integer)`

Determines the Red, Green, and Blue coefficients for `V4L2_COLORFX_SET_RGB` color effect. Bits [7:0] of the supplied 32 bit value are interpreted as Blue component, bits [15:8] as Green component, bits [23:16] as Red component, and bits [31:24] must be zero.

`V4L2_CID_COLORFX_CBCR (integer)`

Determines the Cb and Cr coefficients for `V4L2_COLORFX_SET_CBCR` color effect. Bits [7:0] of the supplied 32 bit value are interpreted as Cr component, bits [15:8] as Cb component and bits [31:16] must be zero.

`V4L2_CID_AUTOBRIGHTNESS (boolean)`

Enable Automatic Brightness.

`V4L2_CID_ROTATE (integer)`

Rotates the image by specified angle. Common angles are 90, 270 and 180. Rotating the image to 90 and 270 will reverse the height and width of the display window. It is necessary to set the new height and width of the picture using the :ref:`VIDIOC_S_FMT <VIDIOC_G_FMT>` ioctl according to the rotation angle selected.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst, line 267); *backlink*

Unknown interpreted text role "ref".

`V4L2_CID_BG_COLOR (integer)`

Sets the background color on the current output device. Background color needs to be specified in the RGB24 format. The supplied 32 bit value is interpreted as bits 0-7 Red color information, bits 8-15 Green color information, bits 16-23 Blue color information and bits 24-31 must be zero.

`V4L2_CID_ILLUMINATORS_1 V4L2_CID_ILLUMINATORS_2 (boolean)`

Switch on or off the illuminator 1 or 2 of the device (usually a microscope).

`V4L2_CID_MIN_BUFFERS_FOR_CAPTURE (integer)`

This is a read-only control that can be read by the application and used as a hint to determine the number of CAPTURE buffers to pass to REQBUFS. The value is the minimum number of CAPTURE buffers that is necessary for hardware to work.

`V4L2_CID_MIN_BUFFERS_FOR_OUTPUT (integer)`

This is a read-only control that can be read by the application and used as a hint to determine the number of OUTPUT buffers to pass to REQBUFS. The value is the minimum number of OUTPUT buffers that is necessary for hardware to work.

`V4L2_CID_ALPHA_COMPONENT (integer)`

Sets the alpha color component. When a capture device (or capture queue of a mem-to-mem device) produces a frame format that includes an alpha component (e.g. :ref:`packed RGB image formats <pixfmt-rgb>`) and the alpha value is not defined by the device or the mem-to-mem input data this control lets you select the alpha component value of all pixels. When an output device (or output queue of a mem-to-mem device) consumes a frame format that doesn't include an alpha component and the device supports alpha channel processing this control lets you set the alpha component value of all pixels for further processing in the device.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst, line 300); *backlink*

Unknown interpreted text role "ref".

`V4L2_CID_LASTP1`

End of the predefined control IDs (currently `V4L2_CID_ALPHA_COMPONENT` + 1).

`V4L2_CID_PRIVATE_BASE`

ID of the first custom (driver specific) control. Applications depending on particular custom controls should check the driver name and version, see :ref:`querycap`.

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst`, line 317); *backlink*
>
> Unknown interpreted text role "ref".

Applications can enumerate the available controls with the :ref:`VIDIOC_QUERYCTRL` and :ref:`VIDIOC_QUERYMENU <VIDIOC_QUERYCTRL>` ioctls, get and set a control value with the :ref:`VIDIOC_G_CTRL <VIDIOC_G_CTRL>` and :ref:`VIDIOC_S_CTRL <VIDIOC_G_CTRL>` ioctls. Drivers must implement `VIDIOC_QUERYCTRL`, `VIDIOC_G_CTRL` and `VIDIOC_S_CTRL` when the device has one or more controls, `VIDIOC_QUERYMENU` when it has one or more menu type 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)control.rst`, line 321); *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)control.rst`, line 321); *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)control.rst`, line 321); *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)control.rst`, line 321); *backlink*
>
> Unknown interpreted text role "ref".

Example: Enumerating all controls

```
struct v4l2_queryctrl queryctrl;
struct v4l2_querymenu querymenu;

static void enumerate_menu(__u32 id)
{
    printf("  Menu items:\\n");

    memset(&querymenu, 0, sizeof(querymenu));
    querymenu.id = id;

    for (querymenu.index = queryctrl.minimum;
         querymenu.index <= queryctrl.maximum;
         querymenu.index++) {
        if (0 == ioctl(fd, VIDIOC_QUERYMENU, &querymenu)) {
            printf("  %s\\n", querymenu.name);
        }
    }
}

memset(&queryctrl, 0, sizeof(queryctrl));

queryctrl.id = V4L2_CTRL_FLAG_NEXT_CTRL;
while (0 == ioctl(fd, VIDIOC_QUERYCTRL, &queryctrl)) {
    if (!(queryctrl.flags & V4L2_CTRL_FLAG_DISABLED)) {
        printf("Control %s\\n", queryctrl.name);

        if (queryctrl.type == V4L2_CTRL_TYPE_MENU)
```

```
            enumerate_menu(queryctrl.id);
    }

    queryctrl.id |= V4L2_CTRL_FLAG_NEXT_CTRL;
}
if (errno != EINVAL) {
    perror("VIDIOC_QUERYCTRL");
    exit(EXIT_FAILURE);
}
```

Example: Enumerating all controls including compound controls

```
struct v4l2_query_ext_ctrl query_ext_ctrl;

memset(&query_ext_ctrl, 0, sizeof(query_ext_ctrl));

query_ext_ctrl.id = V4L2_CTRL_FLAG_NEXT_CTRL | V4L2_CTRL_FLAG_NEXT_COMPOUND;
while (0 == ioctl(fd, VIDIOC_QUERY_EXT_CTRL, &query_ext_ctrl)) {
    if (!(query_ext_ctrl.flags & V4L2_CTRL_FLAG_DISABLED)) {
        printf("Control %s\\n", query_ext_ctrl.name);

        if (query_ext_ctrl.type == V4L2_CTRL_TYPE_MENU)
            enumerate_menu(query_ext_ctrl.id);
    }

    query_ext_ctrl.id |= V4L2_CTRL_FLAG_NEXT_CTRL | V4L2_CTRL_FLAG_NEXT_COMPOUND;
}
if (errno != EINVAL) {
    perror("VIDIOC_QUERY_EXT_CTRL");
    exit(EXIT_FAILURE);
}
```

Example: Enumerating all user controls (old style)

```
memset(&queryctrl, 0, sizeof(queryctrl));

for (queryctrl.id = V4L2_CID_BASE;
     queryctrl.id < V4L2_CID_LASTP1;
     queryctrl.id++) {
    if (0 == ioctl(fd, VIDIOC_QUERYCTRL, &queryctrl)) {
        if (queryctrl.flags & V4L2_CTRL_FLAG_DISABLED)
            continue;

        printf("Control %s\\n", queryctrl.name);

        if (queryctrl.type == V4L2_CTRL_TYPE_MENU)
            enumerate_menu(queryctrl.id);
    } else {
        if (errno == EINVAL)
            continue;

        perror("VIDIOC_QUERYCTRL");
        exit(EXIT_FAILURE);
    }
}

for (queryctrl.id = V4L2_CID_PRIVATE_BASE;;
     queryctrl.id++) {
    if (0 == ioctl(fd, VIDIOC_QUERYCTRL, &queryctrl)) {
        if (queryctrl.flags & V4L2_CTRL_FLAG_DISABLED)
            continue;

        printf("Control %s\\n", queryctrl.name);

        if (queryctrl.type == V4L2_CTRL_TYPE_MENU)
            enumerate_menu(queryctrl.id);
    } else {
        if (errno == EINVAL)
            break;

        perror("VIDIOC_QUERYCTRL");
        exit(EXIT_FAILURE);
    }
}
```

Example: Changing controls

```
struct v4l2_queryctrl queryctrl;
```

```
struct v4l2_control control;

memset(&queryctrl, 0, sizeof(queryctrl));
queryctrl.id = V4L2_CID_BRIGHTNESS;

if (-1 == ioctl(fd, VIDIOC_QUERYCTRL, &queryctrl)) {
    if (errno != EINVAL) {
        perror("VIDIOC_QUERYCTRL");
        exit(EXIT_FAILURE);
    } else {
        printf("V4L2_CID_BRIGHTNESS is not supportedn");
    }
} else if (queryctrl.flags & V4L2_CTRL_FLAG_DISABLED) {
    printf("V4L2_CID_BRIGHTNESS is not supportedn");
} else {
    memset(&control, 0, sizeof (control));
    control.id = V4L2_CID_BRIGHTNESS;
    control.value = queryctrl.default_value;

    if (-1 == ioctl(fd, VIDIOC_S_CTRL, &control)) {
        perror("VIDIOC_S_CTRL");
        exit(EXIT_FAILURE);
    }
}

memset(&control, 0, sizeof(control));
control.id = V4L2_CID_CONTRAST;

if (0 == ioctl(fd, VIDIOC_G_CTRL, &control)) {
    control.value += 1;

    /* The driver may clamp the value or return ERANGE, ignored here */

    if (-1 == ioctl(fd, VIDIOC_S_CTRL, &control)
        && errno != ERANGE) {
        perror("VIDIOC_S_CTRL");
        exit(EXIT_FAILURE);
    }
/* Ignore if V4L2_CID_CONTRAST is unsupported */
} else if (errno != EINVAL) {
    perror("VIDIOC_G_CTRL");
    exit(EXIT_FAILURE);
}

control.id = V4L2_CID_AUDIO_MUTE;
control.value = 1; /* silence */

/* Errors ignored */
ioctl(fd, VIDIOC_S_CTRL, &control);
```

[1] The use of V4L2_CID_PRIVATE_BASE is problematic because different drivers may use the same V4L2_CID_PRIVATE_BASE ID for different controls. This makes it hard to programmatically set such controls since the meaning of the control with that ID is driver dependent. In order to resolve this drivers use unique IDs and the V4L2_CID_PRIVATE_BASE IDs are mapped to those unique IDs by the kernel. Consider these V4L2_CID_PRIVATE_BASE IDs as aliases to the real IDs.

Many applications today still use the V4L2_CID_PRIVATE_BASE IDs instead of using :ref:`VIDIOC_QUERYCTRL` with the V4L2_CTRL_FLAG_NEXT_CTRL flag to enumerate all IDs, so support for V4L2_CID_PRIVATE_BASE is still around.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\userspace-api\media\v4l\(linux-master)(Documentation)(userspace-api)(media)(v4l)control.rst, line 514); *backlink*

Unknown interpreted text role "ref".