VCC310 Linux Software Programming Guide

Customized Property

2010.10.18

Our driver is built based on **Video 4 Linux 2 (V4L2)** and **Advanced Linux Sound Architecture (ALSA)**, video and audio respectively . Software would be easily implemented if you adopt this standard APIs. In addition, we could design some customized properties for the user to control these special functions.

V4L2 resource:
http://linux.bytesex.org/v4l2/

ALSA resource:
http://www.alsa-project.org/alsa-doc/alsa-lib/index.html

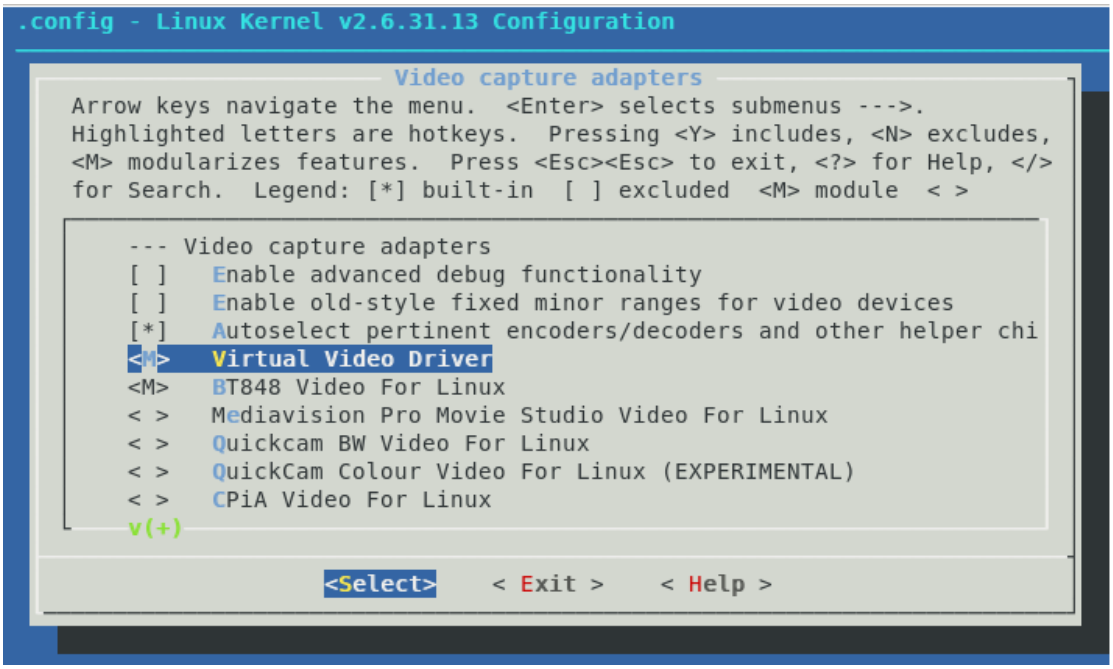
Contents

1. Module Dependancy 3
2. Module Installation and Uninstallation 4
3. Set/Get GPIO Property 5
4. Get/Set Decoder Property 7
5. Video and Audio Sample Program 8

1. Module Dependancy

Our module is built based on **V4L2**(video) and **ALSA**(audio), so you have to make sure your kernel have these modules built in.

Note. If you are using your customized linux kernel, you have to make Virtual Video Driver built in your kernel, which is located at Device Drivers -> Multimedia Devices -> Video capture adapters -> Virtual Video Driver.



2. Module Installation and Uninstallation

Our driver module name is LXV4L2D.ko. To reduce the possibility of failed installation, generally we will compile our driver in the environment as similar as what the user uses.

Module Installation:

Step1. Copy LXV4L2D.ko to your module directory.

```
cp LXV4L2D.ko /lib/modules/`uname -r`/
```

Step2. Add LXV4L2D.ko to modules.dep.

```
depmod -a
```

Step3. Load LXV4L2D.ko module.

```
modprobe LXV4L2D
```

If "invalid module format" error message occurs, try to use -f parameter.

```
modprobe -f LXV4L2D
```

Module Uninstallation

```
modprobe -r LXV4L2D
```

If you still have error message when loading the module, please provide your .config file to us, which is located in /usr/src/`uname -r`/.config.

3. Set/Get GPIO Property

```
#define V4L2_CID_GPIO_DIRECTION (V4L2_CID_BASE + 270)
#define V4L2_CID_GPIO_DATA (V4L2_CID_BASE + 271)
```

The property allows you to access CX25820/1's GPIO interface. The property V4L2_CID_GPIO_DIRECTION allows you to control its direction. Here, writing 1 to bit enables this pin as output pin. Usually, the GPIO is controlled by the first chipset in one board.

Support Value: 0 ~ 1 – Input ~ Output

The property V4L2_CID_GPIO_DATA allows you to access GPIO's data.

Support Value: 0 ~ 1 – Low ~ High

EXAMPLE#01: define GPIO as 16 output pins [0:15] and 16 input pins [16:31].

```
struct v4l2_control s_ctrl;
s_ctrl.id = V4L2_CID_GPIO_DIRECTION;
s_ctrl.value = 0x0000FFFF;
ioctl( fd, VIDIOC_S_CTRL, &s_ctrl );
```

EXAMPLE#02: Define GPIO as 32 output pints [0:31] then pull high for all.

```
struct v4l2_control s_ctrl;
s_ctrl.id = V4L2_CID_GPIO_DIRECTION;
s_ctrl.value = 0xFFFFFFFF;
ioctl( fd, VIDIOC_S_CTRL, &s_ctrl );
s_ctrl.id = V4L2_CID_GPIO_DATA;
s_ctrl.value = 0xFFFFFFFF;
ioctl( fd, VIDIOC_S_CTRL, &s_ctrl );
```

EXAMPLE#03: Define GPIO as 32 input pins [0:31] then read data from it.

```
struct v4l2_control s_ctrl;
```

```
s_ctrl.id = V4L2_CID_GPIO_DIRECTION;
s_ctrl.value = 0x00000000;
ioctl( fd, VIDIOC_S_CTRL, &s_ctrl );
s_ctrl.id = V4L2_CID_GPIO_DATA;
ioctl( fd, VIDIOC_G_CTRL, &s_ctrl );
```

4. Get/Set Decoder Property

You can use VIDIOC_S_CTRL / VIDIOC_G_CTRL to access analog video decoder property.

Currently, we had offered these properties for VDB300 series as below:

```
#define V4L2_CID_BRIGHTNESS (V4L2_CID_BASE + 0)
#define V4L2_CID_CONTRAST (V4L2_CID_BASE + 1)
#define V4L2_CID_SATURATION (V4L2_CID_BASE + 2)
#define V4L2_CID_HUE (V4L2_CID_BASE + 3)
#define V4L2_CID_SHARPNESS (V4L2_CID_BASE + 27)
```

EXAMPLE#01. Set brightness property of sub-channel#01.

```
struct v4l2_control s_ctrl;
s_ctrl.id = V4L2_CID_BRIGHTNESS;
s_ctrl.value = 128;
ioctl( fd, VIDIOC_S_CTRL, &s_ctrl );
```

EXAMPLE#02. Set contrast property of sub-channel#02.

```
struct v4l2_control s_ctrl;
s_ctrl.id = V4L2_CID_CONTRAST;
s_ctrl.value = 128;
ioctl( fd, VIDIOC_S_CTRL, &s_ctrl );
```

EXAMPLE#03. Set saturation property of sub-channel #03.

```
struct v4l2_control s_ctrl;
s_ctrl.id = V4L2_CID_SATURATION;
s_ctrl.value = 128;
ioctl( fd, VIDIOC_S_CTRL, &s_ctrl );
```

5. Video and Audio Sample Program

Video:

Ucview, which is built with v4l/v4l2 interface, is a public and free video capture software for linux. So you can use ucview to test our capture card.

In addition, we also provide our own sample program for you to reference.

Audio:

With our audio driver is built based on ALSA(Advanced Linux Sound Architecture), you can use "**arecord**" to record sound and "**aplay**" to play the recorded sound.

List your audio device.

```
arecord -l
```

Record channel 1 of capture card and play the sound instantly.

```
arecord -D hw:1,0 -r 48000 -c 1 -f S16_LE | aplay
```

(Usually hw:0,0 is your sound card device)

Record channel 2 of capture card to file CH2.wav

```
arecord -D hw:2,0 -r 48000 -c 1 -f S16_LE CH2.wav
```

You can type 'man areocrd' or go to here to see more detailed parameter setting.

In addition, we also provide our own sample program for you to reference.