

# USB Audio/Video Codec Model 2253

## Linux Software Manual

Ver. 1.2.2 | January 2012

SENSORAY | embedded electronics



Designed and manufactured in the U.S.A

SENSORAY | p. 503.684.8005 | email: [info@SENSORAY.com](mailto:info@SENSORAY.com) | [www.SENSORAY.com](http://www.SENSORAY.com)

7313 SW Tech Center Drive | Portland, OR 97203

# Table of Contents

OPERATING SYSTEM SUPPORT.....	3
INSTALLATION.....	4
BASIC OPERATION.....	5
Video Capture and Output Driver.....	5
Troubleshooting.....	6
Demo Program.....	7
Notes on the MP4 format.....	7
GUI Demo Program.....	8
Playing Recorded Files.....	9
Using ALSA for audio capture and playback.....	10
Using the Video Output Device.....	11
Using Loopback on Video Output Device.....	11
Using the Video Overlay.....	12
GPIO Device.....	12
Known Limitations.....	13
SDK REFERENCE.....	14
General Notes.....	14
Driver ioctl Reference.....	14
VIDIOC_G_FMT, VIDIOC_S_FMT, VIDIOC_TRY_FMT.....	15
VIDIOC_G_JPEGCOMP, VIDIOC_S_JPEGCOMP.....	16
VIDIOC_ENUM_INPUT, VIDIOC_G_INPUT, VIDIOC_S_INPUT.....	16
VIDIOC_ENUM_OUTPUT, VIDIOC_G_OUTPUT, VIDIOC_S_OUTPUT.....	16
VIDIOC_QUERYCTRL, VIDIOC_G_CTRL, VIDIOC_S_CTRL, VIDIOC_TRY_EXT_CTRL, VIDIOC_G_EXT_CTRL, VIDIOC_S_EXT_CTRL.....	16
VIDIOC_ENUMAUDIO, VIDIOC_G_AUDIO, VIDIOC_S_AUDIO.....	19
VIDIOC_ENUMAUDOUT, VIDIOC_G_AUDOUT, VIDIOC_S_AUDOUT.....	19
S2253_VIDIOC_OSD.....	19
S2253_VIDIOC_CLOCK.....	20
S2253_VIDIOC_OVERLAY.....	20
S2253_VIDIOC_SET_USER_DATA.....	22
REVISION HISTORY.....	23

# Operating System Support

The SDK has been developed on Linux Ubuntu LTS and support is provided for this distribution. The SDK may work on other Linux distributions, but this is not guaranteed. The lowest required kernel version is 2.6.25. An effort has been made to support version 2.6.18, although some features may be limited.

# Installation

The software may be distributed on a CD or downloaded from Sensoray's web site. If the file is downloaded, it will need to be unzipped into a folder on the local drive prior to connecting the 2253 to the USB port.

Setup is performed as follows.

- 1) Unpack the `tgz` file:

```
tar xjf sdk_2253_X.Y.Z_linux.tar.bz2,
```

where `X.Y.Z` is the version of the SDK.

- 2) Change the current directory to the one where the unpacked SDK is:

```
cd sdk_2253_X.Y.Z_linux
```

- 3) Build the driver and demo application:

```
make all
```

- 4) Install the modules and firmware:

```
sudo make install
```

- 5) Plug the 2253 into an available USB port. The module will be loaded automatically and the device should be ready to use.

- 6) Optional: If the 2253 was previously plugged in, unplug it and return to step 5, or load the module manually with:

```
modprobe s2253
```

Note: These steps only need to be performed once. After reboot, the module should always be loaded automatically when the 2253 is present. If the kernel version on your system changes, steps 3 and 4 may need to be performed again to reinstall the modules.

# Basic Operation

When the driver is loaded, there are three video device nodes and one GPIO char device created in the /dev directory. The video nodes are named “videoX” where X depends on the number of video devices present in the system. The first two video device nodes are for video capture/preview and the third video device node is for video output.

## ***Video Capture and Output Driver***

The driver supports Video4Linux 2 (V4L2) ioctls. The V4L2 API is well documented at the LinuxTV web site (<http://www.linuxtv.org/downloads/v4l-dvb-apis/>). V4L2 operation is not supported for kernels below 2.6.25. The v4l-dvb hg or git tree is not required (and not recommended) to use this driver.

The three video devices can be used with applications that support V4L2 API. Video can be captured using uncompressed YUV422 (packed YUYV or UYVY) or YUV420SP (NV12 semi-planar, Y plane and interleaved CrCb plane) or encoded in compressed formats JPEG, MPEG4 ASP or H.264 elementary streams. The MPEG4 or H.264 streams can also be muxed with AAC audio (with A/V sync) in a MP4 container or a MPEG transport stream. Both capture devices record video from a single source, and each capture device can be started, stopped, and configured independently. Some video options cannot be configured independently, such as interpolation, brightness, hue, contrast, and saturation.

One example of operation is capturing compressed video to a storage device while simultaneously previewing the uncompressed video on the display. Another example is capturing high-bandwidth compressed video to a storage device while simultaneously streaming low-bandwidth compressed video over a network interface.

The video output device can play compressed streams containing MPEG4 or H.264 elementary streams. MP4 muxed fragmented streams or MPEG transport streams are also supported with A/V sync. MJPEG decoding is not supported at this time.

There are internal limitations in the device that may prevent use of both input streams and the output stream simultaneously. It is recommended to limit use to both input streams simultaneously, or a single input stream and single output stream simultaneously. The limitation may appear as stuttering video and audio during playback.

## **Troubleshooting**

If the driver fails to load, or if after loading the driver, there are no video device nodes created, this may indicate a kernel compatibility problem in the driver. Please send the output from “`uname -r`” and “`dmesg`” with the name and version number of the Linux distribution to [support@sensoray.com](mailto:support@sensoray.com). If you have a custom built kernel, please also provide the kernel configuration file.

The driver file is a kernel module that resides in  
`/lib/modules/`uname -r`/extra/s2253.ko`.

If this file is missing, which can happen if your Linux kernel is updated, please reinstall the driver, starting at Step 4 in the setup instructions. If you have the “`lsusb`” program installed, you can see if the device firmware was loaded. When Model 2253 is plugged in, it initially has the USB device id 1943:a253 and the red LED (D1) on the 2253TA board is not lit. The driver loads the firmware file on the 2253 from `/lib/firmware/s2253.fw`. If this file is missing, please reinstall the driver, Step 4 in the setup instructions. When the 2253 firmware is loaded, the USB device id should change to 1943:2253 and the red LED on 2253TA board should be lit.

If the LED fails to light and the `lsusb` device id 1942:a253 indicates the firmware hasn't loaded, change to the driver directory “`cd driver`” and do “`sudo make unload`” and “`sudo make load`”. This will reload the driver with some debug messages enabled, and “`dmesg`” should provide some extra detail about the problem.

The red LED on the 2253TA board will also indicate the status of the composite video input signal. The LED will blink when the signal is absent or doesn't match the requested video standard parameter. The LED blinking feature can be controlled by using a v4l2 control (see API.)

## **Demo Program**

Make sure video source is connected and turned on.

1. Make demo with command “make demo”
2. “./capture” runs the demo application.
3. Type “./capture -h” for help menu.
4. For example, type “./capture -4 -b 2000000 -o output.m4v” to save a MPEG4 video at 2 Mbps in the file `output.m4v`.

Mplayer could be used to display live video captured by the 2253. Below are some example Mplayer commands to display various formats:

```
mplayer tv:// -tv outfmt=yuy2:width=320:height=240:norm=NTSC:device=/dev/video1
mplayer tv:// -tv outfmt=uyvy:width=320:height=240:norm=NTSC:device=/dev/video1
mplayer tv:// -tv outfmt=mjpg:width=320:height=240:norm=NTSC:device=/dev/video1
MPEG4: mplayer tv:// -tv outfmt=0x5634504d:width=640:height=480:norm=NTSC
H264: mplayer tv:// -tv outfmt=0x34363248:width=640:height=480:norm=NTSC
```

The options for changing the format use the `outfmt` setting. The width and height settings control the image size. The `norm` setting can take NTSC or PAL. The device setting can select which capture device to play from, allowing you to use two mplayer commands to display both streams simultaneously.

## **Notes on the MP4 format**

The MP4 container format produced by the 2253 features audio and video sync muxed in a single stream. The device itself creates a “fragmented” MP4 stream, which makes it possible to stream incrementally over USB or network. A fragmented file is always valid, in case of power loss or interruption. Unfortunately the fragmented format has limited application support and may only play with Quicktime, MPlayer and FFmpeg. To ensure the recorded MP4 files are playable in most players, the capture demo transparently transcodes the fragmented stream in memory to a normal MP4 file.

To capture in normal MP4 format, use the `-m` option and the `-4` or `-x` option with the capture demo. The output must be to a file.

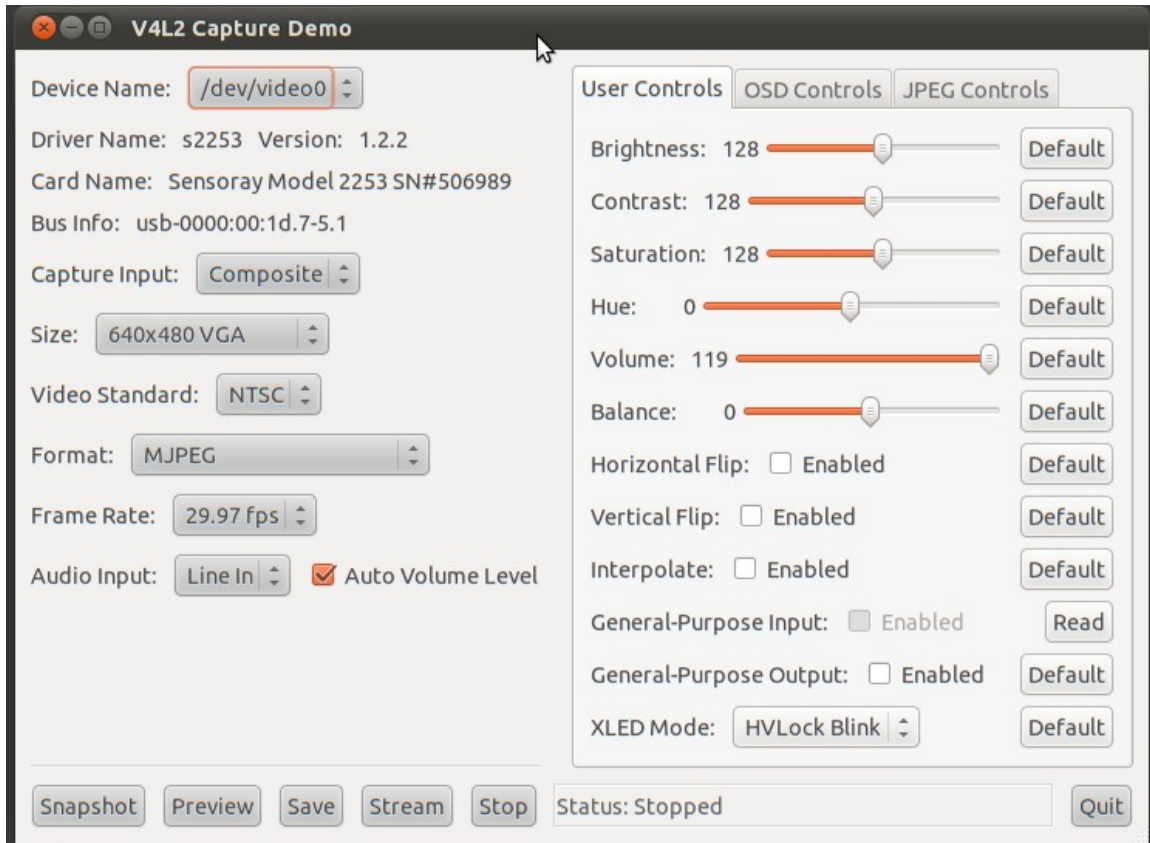
To capture in fragmented MP4 format, use the `-z` option and the `-4` or `-x` option. Note that this file may not play in some players.

To play a MP4 file with the video output device, it must be in the fragmented format. For example, “`cat output.mp4f > /dev/video2`” will play the file on the device.

To play a MP4 file that is non-fragmented, use the playback demo program, which can convert the file on-the-fly to the fragmented format and play it on the device.

## GUI Demo Program

Make sure video source is connected and turned on. Execute the application "demo.py" in the driver directory. The application window will look similar to the following image:



Use the Device Name selector to choose which device to capture from. The buttons on the lower left can be used to take a single JPEG snapshot in the current directory, launch a preview window using the mplayer application (if installed,) or capture stream data to a file, or network stream over UDP. The left side controls are used only when starting the stream, and the right side tabbed controls can be adjusted while streaming. The mplayer preview or file capture will continue to run while another Device Name is selected, allowing multiple streams to be captured simultaneously.

## **Playing Recorded Files**

Examples here are for VideoLAN Player (VLC) MPlayer/MEncoder and FFmpeg.

VLC web site: <http://www.videolan.org/vlc/>

MPlayer web site: <http://www.mplayerhq.hu/>

FFmpeg web site: <http://www.ffmpeg.org/>

When fps is required, use 29.97 fps for NTSC, and 25 fps for PAL. Also, substitute the actual image size used to capture, instead of of 640x480.

### **VLC examples:**

MJPEG: vlc output.mjpg --mjpeg-fps 29.97

MPEG4: vlc output.m4v

h264: vlc output.h264

MP4: vlc output.mp4

### **MPlayer examples:**

MJPEG: mplayer output.mjpg -fps 29.97

UYVY: mplayer -rawvideo format=uyvy:w=640:h=480 -demuxer rawvideo output uyvy  
-fps 29.97

YUYV: mplayer -rawvideo format=yuy2:w=640:h=480 -demuxer rawvideo output yuy2  
-fps 29.97

NV12: mplayer -rawvideo format=nv12:w=640:h=480 -demuxer rawvideo output nv12  
-fps 29.97

grey: mplayer -rawvideo format=y800:w=640:h=480 -demuxer rawvideo output grey  
-fps 29.97

MPEG4: mplayer output.m4v

h264: mplayer output.h264 -fps 29.97

MP4: mplayer output.mp4

### **FFmpeg conversion to AVI format:**

MJPEG: ffmpeg -f mjpeg -r 29.97 -i output.mjpg -vcodec copy output\_mjpg.avi

UYVY: ffmpeg -f rawvideo -pix\_fmt uyvy422 -s 640x480 -r 29.97 -i output uyvy  
-vcodec copy output uyvy.avi

YUYV: ffmpeg -f rawvideo -pix\_fmt yuyv422 -s 640x480 -r 29.97 -i output yuyv  
-vcodec copy output yuyv.avi

NV12: ffmpeg -f rawvideo -pix\_fmt nv12 -s 640x480 -r 29.97 -i output nv12  
-vcodec copy output nv12.avi

grey: ffmpeg -f rawvideo -pix\_fmt gray -s 640x480 -r 29.97 -i output grey  
-vcodec copy output grey.avi

MPEG4: ffmpeg -f m4v -r 29.97 -i output.m4v -vcodec copy output\_mpeg4.avi

h264: ffmpeg -f h264 -r 29.97 -i output.h264 -vcodec copy output\_h264.avi

### **The raw files can also be converted to AVI format using mencoder.**

MJPEG: mencoder output.mjpg -fps 29.97 -ovc copy -o output\_mjpg.avi

UYVY: mencoder -rawvideo format=uyvy:w=640:h=480 -demuxer rawvideo output uyvy  
-fps 29.97 -ovc copy -o output uyvy.avi

YUYV: mencoder -rawvideo format=yuy2:w=640:h=480 -demuxer rawvideo output yuyv  
-fps 29.97 -ovc copy -o output yuyv.avi

NV12: mencoder -rawvideo format=nv12:w=640:h=480 -demuxer rawvideo output nv12  
-fps 29.97 -ovc copy -o output nv12.avi

grey: mencoder -rawvideo format=y800:w=640:h=480 -demuxer rawvideo output grey  
-fps 29.97 -ovc copy -o output grey.avi

MPEG4: mencoder output.m4v -ovc copy -o output\_mpeg4.avi

h264: mencoder output.h264 -fps 29.97 -ovc copy -o output\_h264.avi

## **Using ALSA for audio capture and playback**

The driver provides an ALSA card interface for raw PCM, g.711 u-law and A-law audio capture. The ALSA device operates independently of the video capture device.

To find out the ALSA name of the 2253 board, use the program “arecord -l”:

```
**** List of CAPTURE Hardware Devices ****
card 1: s2253 [Sensoray 2253], device 0: s2253 [Sensoray 2253]
  Subdevices: 1/1
    Subdevice #0: subdevice #0
```

Here we see card 1 device 0 is the 2253, so it can be referred to as “hw:1,0”.

To record audio, use the arecord program and specify the device, format, sample rate, and number of channels. For example:

```
PCM 16-bit stereo: arecord -D hw:1,0 -f S16_LE -r 48000 -c 2 output.pcm
g.711 A-law mono:   arecord -D hw:1,0 -f A_LAW -r 8000 -c 1 output.g711
g.711 mu-law mono: arecord -D hw:1,0 -f MU_LAW -r 8000 -c 1 output.g711
```

Use <ctrl>C to end recording, or add the -d option to limit the duration. For more detailed options, see the man page: “man arecord”.

To play back recorded audio on the host, use the aplay program and specify the same settings used to record. For example:

```
PCM 16-bit stereo: aplay -f S16_LE -r 48000 -c 2 output.pcm
g.711 A-law mono:   aplay -f A_LAW -r 8000 -c 1 output.g711
g.711 mu-law mono: aplay -f MU_LAW -r 8000 -c 1 output.g711
```

To adjust the volume and audio input, use the alsamixer command. For example, if the 2253 ALSA card number is 1, use the -c option “alsamixer -c 1”. Use the tab key to select [All] interface, and arrow keys to select the setting to change. There are two volume controls, for Playback and Capture. The Input AGC setting is Automatic Gain Control for left and right channels - it is important to note that the volume controls have no effect when AGC is enabled. The Input Source setting allows you to select either the Mic or Line In input source. Press ESC to exit alsamixer.

In addition to the command-line utilities, any ALSA-compliant application should be able to record and change the settings of the 2253 device.

The volume and input source settings are per-board settings and will also affect the AAC audio of any MP4 or MPEG-TS streams being captured.

## ***Using the Video Output Device***

The easiest way to play a file on the video output device is to pipe a recorded file to it using the “write” method:

```
cat output.m4v > /dev/video2
```

The driver will attempt to detect the file format and set the decoder format automatically. The video output device can only play back MPEG-4 and H.264 elementary streams, MP4 fragmented streams, and MPEG-TS.

The “playback” demo program includes an example on how to convert a non-fragmented MP4 stream to a fragmented stream playable on the device. The device will determine the dimensions of the compressed stream, and there is no need to provide width and height parameters.

The “playback” demo may also be used to play uncompressed streams, but require the format and dimensions be provided as parameters.

The video standard must be set manually for playback. Failure to do so may result in a distorted image or incorrect frame rate. If the stream dimensions are smaller than the display size (720x480 NTSC or 720x576 PAL) the image will be centered in the display with a black border.

## ***Using Loopback on Video Output Device***

Use this command to test loopback of mp4 fragmented stream on video output:

```
./capture -z -s 704x480 -g 1 -f 0 -o - | ./playback -
```

The capture demo flushes the output file descriptor after each frame and the playback demo uses non-buffered io to keep latency at a minimum while copying the stream over the host. This ensures that the complete packet of an encoded frame is not delayed at any point, so that it can be decoded as soon as possible. Similar measures should be taken by a streaming application developer.

Using cat (for example: cat /dev/video0 > /dev/video2) to capture from the device will introduce unnecessary 4k buffering of the stream. This buffering will transfer partial frames, causing the decoder to wait until the complete frame arrives, increasing the latency.

The playback demo uses a low-latency mode feature when playing from a non-file source, allowing the device to automatically reduce latency when the device determines that it has enough information to smoothly drop audio and video frames. See also the ioctl control S2253\_CID\_LOW\_LATENCY in the API reference.

## ***Using the Video Overlay***

The device can render graphic images overlaid the video output display. The hardware overlay features 16-bits per-pixel color information and 3-bits per-pixel transparency information. The device supports overlay images in the PNG format and 24 or 32-bit BMP format. The device can draw multiple lines of text and also supports line-drawing functions.

The SDK `overlay` directory contains an `overlay` program to load overlay images and a shell script `overlay-demo.sh` to load the sample images provided. The shell script `overlay-lines.sh` will draw lines on the display. The shell script `overlay-text.sh` will draw text on the display. The sample code `overlay.c` shows how to format the drawing commands for the device.

See also `S2253_VIDIIOC_OVERLAY` ioctl below describing data formats for line drawing and text.

## ***GPIO Device***

The GPIO device is exposed as a Linux sysfs GPIO class. The directories containing the device are under `/sys/class/gpio/`. Each board appears as a `gpiochipNNN` subdirectory, containing a label file describing the board and serial no: “S2253 SN123456”, and the base and `ngpio` files describing the GPIO range used by the board. The base describes the first `gpioNNN` subdirectory and the `ngpio` should be 2. The `gpioNNN` subdirectory contains a direction file (indicating “in” or “out”) and the value file. An “in” GPIO can be monitored by reading the value file, and an “out” GPIO can be modified by writing a “1” or “0” to the value file.

A simpler method is also provided as a `v4l2` control for the GPI and another for GPO. See the API reference for `S2253_CID_GPI` and `S2253_CID_GPO`.

## **Known Limitations**

The device cannot play back MJPEG recorded video on composite output.

Due to device resources, only two streams should be used at a time: two capture, or one capture and one output. This table lists valid combination pairs:

<b>Capture 1</b>	<b>Capture 2</b>	<b>Output</b>
Ok	Ok	
Ok		Ok
	Ok	Ok

MPEG-4 capture/output cannot be used simultaneously with H.264 capture. This table lists valid combination pairs:

<b>Capture H.264</b>	<b>Capture MPEG-4</b>	<b>Output H.264</b>	<b>Output MPEG-4</b>
Ok		Ok	
	Ok		Ok
	Ok	Ok	

Low-latency preview mode cannot be used simultaneously with AAC encode/decode (when using MP4 or MPEG-TS mux, for example.) The low-latency preview will be switched automatically to normal-latency preview when AAC is started, and revert back when AAC is stopped. If mux is required without AAC audio, use `V4L2_MPEG_AUDIO_ENCODING_NONE` option with the `V4L2_CID_MPEG_AUDIO_ENCODING` control.

Preview mode can consume a large amount of USB bandwidth. If multiple devices are connected on the same USB root hub, previewing on each may cause unwanted effects (jerky or stuttering video on capture or output.)

To reduce these effects:

- move each device to a different USB root hub
- reduce the width and height of the preview video
- reduce the frame rate

## • SDK Reference

### **General Notes**

Model 2253 device is accessed via `/dev/videoX` where X is the number assigned to the device when it was plugged in. Three `/dev/videoX` nodes will be created for each 2253 device. The first two video device nodes are for video capture and the third is a video output. Which device the node references can be determined by using the `VIDIOC_QUERYCAP` ioctl to examine the card field for the serial no. The devices support multiple open, allowing some parameters to be changed on-the-fly while another program is accessing the device.

### **Driver ioctl Reference**

This reference is provided as a convenience to detail the 2253-specific capabilities, without requiring the user to query the driver using the V4L2 API or reading the driver source code. For a complete Video4Linux2 API reference, please consult the LinuxTV.org web site. (<http://www.linuxtv.org/downloads/v4l-dvb-apis/>)

All calls to `ioctl` function follow the same general approach.

The first parameter (`fd`) is a file descriptor returned by `open()`, which specifies the addressed device.

The second parameter (`request`) is the request code, which specifies the type of request.

The third parameter is a pointer to context-specific data.

All calls return 0 in case of success, or a negative value in case of an error.

The `ioctls` below are listed by the request parameter.

---

## **VIDIOC\_G\_FMT, VIDIOC\_S\_FMT, VIDIOC\_TRY\_FMT**

```
int ioctl(int fd, int request, struct v4l2_format *fmt);
```

```
fmt->pix.width
```

Integer range between 128 to 720 in steps of 16.

```
fmt->pix.height
```

Integer range 96 to 480 (NTSC) or 576 (PAL) in steps of 16.

Note: for output, width and height are ignored for compressed formats (JPEG/MP2V/MP4V/H264/MP42) since dimensions are encoded in the stream.

```
fmt->pix.pixelformat
```

Supported FOURCC codes:

V4L2\_PIX\_FMT\_NV12 - raw YUV 420 in semi-planar format

V4L2\_PIX\_FMT\_YUYV - raw YUV 422 interleaved luminance first

V4L2\_PIX\_FMT\_UYVY - raw YUV 422 interleaved chroma first

V4L2\_PIX\_FMT\_GREY - raw Y 8-bit samples only

V4L2\_PIX\_FMT\_MJPEG - compressed motion JPEG

V4L2\_PIX\_FMT\_JPEG - same as V4L2\_PIX\_FMT\_MJPEG

V4L2\_PIX\_FMT\_MP4V - MPEG-4 video elementary stream

V4L2\_PIX\_FMT\_H264 - H.264 video elementary stream

V4L2\_PIX\_FMT\_MP42 - MP4 fragmented stream

V4L2\_PIX\_FMT\_MPEG - MPEG transport stream

Note: Codes not defined in videodev2.h are in s2253ioctl.h.

```
fmt->pix.field
```

Supported modes:

The V4L2\_FIELD\_NONE option is used to select a single field and interpolates it to produce the missing lines. This method removes the jagged edges cause by motion between fields, producing a smooth progressive frame. This is a shared option affecting both video output streams on the 2253.

The V4L2\_FIELD\_INTERLACE option provides both interlaced fields in the captured frame. Blurriness or combing artifacts may appear when the capture height does not match the native height for the video standard. This is a shared option affecting both video output streams on the 2253.

The V4L2\_FIELD\_ANY option does not change the current mode, leaving it unchanged on the device.

The field mode can also be controlled by the S2253\_CID\_INTERPOLATE extended control, see below.

---

## **VIDIOC\_G\_JPEGCOMP, VIDIOC\_S\_JPEGCOMP**

```
int ioctl(int fd, int request, v4l2_jpegcompression *jc);
```

```
jc->quality
```

Integer range 10 to 90. The default is 75.

V4L2\_PIX\_FMT\_MJPEG must be selected.

---

## **VIDIOC\_ENUM\_INPUT, VIDIOC\_G\_INPUT, VIDIOC\_S\_INPUT**

```
int ioctl(int fd, int request, struct v4l2_input *inp);
```

There is only one input at index 0, named “Composite”.

The `inp->status` field may have the bit `V4L2_IN_ST_NO_H_LOCK` set if the device was unable to lock the input video signal.

---

## **VIDIOC\_ENUM\_OUTPUT, VIDIOC\_G\_OUTPUT, VIDIOC\_S\_OUTPUT**

```
int ioctl(int fd, int request, struct v4l2_output *inp);
```

There is only one output at index 0, named “Composite”.

---

## **VIDIOC\_QUERYCTRL, VIDIOC\_G\_CTRL, VIDIOC\_S\_CTRL, VIDIOC\_TRY\_EXT\_CTRL, VIDIOC\_G\_EXT\_CTRL, VIDIOC\_S\_EXT\_CTRL**

```
int ioctl(int fd, int request, struct v4l2_ext_controls *ctrls);
```

When using `VIDIOC_G_CTRL` or `VIDIOC_S_CTRL`,

or `ctrls->ctrl_class == V4L2_CTRL_CLASS_USER`,

the supported `ctrls->controls` id identifiers and values are:

`V4L2_CID_BRIGHTNESS`: 0 to 255, default 128. Capture only.

`V4L2_CID_CONTRAST`: 0 to 255, default 128. Capture only.

`V4L2_CID_SATURATION`: 0 to 255, default 128. Capture only.

`V4L2_CID_HUE`: -128 to 128, default 0. Capture only.

`V4L2_CID_AUDIO_VOLUME`: 0 to 119 for capture, 0 to 117 for output.

`V4L2_CID_AUDIO_BALANCE`: -100 to 100, default 0.

`V4L2_CID_HFLIP`: boolean horizontal flip, default 0. Capture only.

---

V4L2\_CID\_VFLIP: boolean vertical flip, default 0. Capture only.

S2253\_CID\_DFLIP:

Boolean diagonal flip, default 0. Capture only. Not available for YUYV, UYVY or JPEG formats.

To rotate the video image, use the following flip combinations:

90 degrees: HFLIP=1 VFLIP=0 DFLIP=1

180 degrees: HFLIP=1 VFLIP=1 DFLIP=0

270 degrees: HFLIP=0 VFLIP=1 DFLIP=1

S2253\_CID\_INTERPOLATE:

Boolean, duplicates the functionality of the pix.field option in the format struct, provided as a convenience to the GUI demo. Capture only.

S2253\_CID\_OUTPUT\_MODE:

Integer range 0 to 3, with the following meanings:

0: Idle - display black screen on output.

1: Video Capture - display the input video on output.

2: Color Bars - display a set of color bars.

3: Flash - display a single white frame and play a sound blip. Useful for measuring audio-video sync.

Output device only.

S2253\_CID\_LOW\_LATENCY:

Boolean, determines whether low-latency playback mode is on or off.

0: Off - use when playing from a file or recorded stream.

1: On - use when playing a live stream. The playback device will attempt to drop audio and video frames to reduce playback latency based on the amount of stream data available. Frames are dropped gradually to avoid noticeable glitches in the audio or video - however it may take tens of seconds to settle at lowest possible latency.

Output device only.

S2253\_CID\_GPI:

Boolean, read-only control whose value indicates the state of the general-purpose input.

S2253\_CID\_GPO:

Boolean, determines state of the general-purpose output.

S2253\_CID\_XLED\_MODE:

Integer range 0 to 3, with the following meanings:

0: HVLock Blink - XLED will blink when video lock is absent.

1: HVLock Solid - XLED will turn off when video lock is absent.

2: On - XLED is always on (power indicator).

3: Off - XLED is always off.

S2253\_CID\_OSD\_ENABLE:

Boolean, determines whether the on-screen-display text is visible, Default 0.

S2253\_CID\_OSD\_TRANSPARENT:  
**Boolean, determines whether the on-screen-display text background is transparent or black, Default 1.**

S2253\_CID\_OSD\_POSITION:  
**Bottom: 0, Top: 1, Default 1.**

S2253\_CID\_OSD\_DATEFORMAT:  
**MM-DD-YYYY: 0, DD-MM-YYYY: 1, MM-DD-YY: 2, DD-MM-YY: 3, Default 0.**

S2253\_CID\_OSD\_TIMEFORMAT:  
**HH:MM:SS: 0, HH:MM:SS.0: 1, HH:MM:SS.00: 2, Default 0.**

S2253\_CID\_OSD\_TIMEZONE:  
**Menu containing hour offsets from UTC. Default UTC + 0.  
 Note: this is only used to set the time on the device relative to the UTC time stored in the host kernel.**

S2253\_CID\_OSD\_MESSAGE:  
**String. May contain “^d” and “^t” to insert date and time, respectively.**

S2253\_CID\_OSD\_XOFF:  
**Integer range 0 to image format width, default 10.**

S2253\_CID\_OSD\_YOFF:  
**Integer range 0 to image format height, default 10.**

**When `ctrls->ctrl_class == V4L2_CTRL_CLASS_MPEG`,  
 supported `ctrls->controls` id identifiers and values:**

V4L2\_CID\_MPEG\_AUDIO\_ENCODING:  
**Use enum `V4L2_MPEG_AUDIO_ENCODING_AAC` or `V4L2_MPEG_AUDIO_ENCODING_NONE`. Only with MP4 or MPEG-TS. Capture only.**

V4L2\_CID\_MPEG\_AUDIO\_MODE:  
**Use enum `V4L2_MPEG_AUDIO_MODE_STEREO` or `V4L2_MPEG_AUDIO_MODE_MONO`. Only with MP4 or MPEG-TS. Capture only.**

V4L2\_CID\_MPEG\_AUDIO\_AAC\_BITRATE:  
**32,000 to 512,000 bps in steps of 16,000 bps, default 192,000 bps.  
 Only with MP4 or MPEG-TS. Capture only.**

V4L2\_CID\_MPEG\_VIDEO\_ENCODING:  
**Use enum `V4L2_MPEG_VIDEO_ENCODING_MPEG_4_AVC` (h.264) or `V4L2_MPEG_VIDEO_ENCODING_MPEG_4` (MPEG-4)  
 Only with MP4 or MPEG-TS. Capture only.**

V4L2\_CID\_MPEG\_VIDEO\_ASPECT:  
**Integer range 0 to 3 to select aspect ratio of video input signal. Capture only.**  
**0: 1x1  
 1: 4x3  
 2: 16x9  
 3: 2.21x1**  
**Note: option 0 is Pixel Aspect Ratio, others are Display Aspect Ratio.**

V4L2\_CID\_MPEG\_VIDEO\_GOP\_SIZE:

Integer range 0 to 30. The default setting of 0 means to use the codec-default GOP size. Capture only.

V4L2\_CID\_MPEG\_VIDEO\_BITRATE:

100,000 to 10,000,000 bps in steps of 100,000 bps, default 2,000,000 bps. Capture only.

V4L2\_CID\_MPEG\_VIDEO\_H264\_I\_PERIOD:

Integer range 0 to 100. Only for H.264 encoding. Default setting of 0 will encode first frame as IDR only, otherwise encode IDR at first frame of every Nth GOP.

---

## **VIDIOC\_ENUMAUDIO, VIDIOC\_G\_AUDIO, VIDIOC\_S\_AUDIO**

```
int ioctl(int fd, int request, struct v4l2_audio *audio);
```

There are two audio inputs:

Index 0: "Mic" (mono)

Index 1: "Line In" (stereo)

Both inputs support audio->mode V4L2\_AUDMODE\_AVL (automatic volume level, also known as AGC, automatic gain control.) The audio input selection and AGC is per-board: both streams record from the same input regardless of which stream selects it.

---

## **VIDIOC\_ENUMAUDOUT, VIDIOC\_G\_AUDOUT, VIDIOC\_S\_AUDOUT**

```
int ioctl(int fd, int request, struct v4l2_audioout *audio);
```

There is one audio output:

Index 0: "Line out" (stereo)

No properties can be changed.

---

## **S2253\_VIDIOC\_OSD**

```
int ioctl(int fd, int request, struct s2253_osd *osd);
```

osd

Pointer to OSD struct

```
struct s2253_osd {
    int  osdOn,           //OSD on if != 0
    int  osdChan,        //OSD channel (filled in by driver)
    int  osdBmp,         //OSD on bitmap if !=0
    int  transparent,    //text background transparent if !=0
    int  positionTop,    //OSD string on top of the screen if !=0
    int  ddmm,           //date format is dd-mm if != 0
```

```

    int    year2,           //date is truncated to 2 digits if !=0
    int    fraction,       //digits showing fraction of a second: 0, 1, 2
    char   line[80],       //caption text
    int    xoff, yoff      //x and y offsets from top/bottom left corner
};

```

---

## S2253\_VIDIOC\_CLOCK

```
int ioctl(int fd, int request, struct s2253_clock *clk);
```

clk

Pointer to

```

struct s2253_clock {
    __u64  sec,           //Seconds
    __u32  usec          //Microseconds
}

```

This structure is similar to the `struct timeval` used with the `gettimeofday()` function. If this struct is initialized using `gettimeofday()`, it should also be adjusted to local time. The clock on the device does not make adjustments for timezones or daylight-saving.

---

## S2253\_VIDIOC\_OVERLAY

```
int ioctl(int fd, int request, struct s2253_overlay_image *ovl);
```

```

struct s2253_overlay_image {
    unsigned char id; // unique id representing this overlay
    unsigned char transparent; // 0=transparent .. 7=opaque
                                // 8=use PNG or BMP transparency info
    unsigned char update; // bit 0: update the display,
                            // bit 1: update the x&y offset,
                            // bit 2: update the transparency
                            // (see defines below)
    unsigned char reserved;
    __u32 xOffset; // x offset from left side
    __u32 yOffset; // y offset from top
    __u32 length; // length of image data (or 0 to use existing
                  // data, and update transparent,
                  // xOffset and yOffset only)
    unsigned char data[0]; // data must contain headers that
                            // describe image dimensions
};

```

```

#define OVERLAY_UPDATE_DISPLAY 1
#define OVERLAY_UPDATE_POSITION 2

```

---

```
#define OVERLAY_UPDATE_TRANSPARENT 4
```

Upload an image to be overlaid on the video output frames. The device remembers the id where each image was placed, allowing it to be moved later. To move an existing overlay image, set the id, set length to zero, set update to OVERLAY\_UPDATE\_POSITION, and set the x and y offsets to the new position. Due to hardware limitations, the x offset may be limited to a multiple of 2. The transparent setting allows the overlay to have a transparency effect, allowing the underlying video to be mixed with the image. When transparent is 8, the alpha channel in a 32-bit PNG or BMP will be used. To hide an existing image, update the id with transparent set to zero and update set to OVERLAY\_UPDATE\_TRANSPARENT. When update bit 0 is not set, the images are rendered to a working buffer, and don't become visible until an operation is called with update bit 0 set to OVERLAY\_UPDATE\_DISPLAY.

Image overlays are destructive; moving or overlapping images will cause the previous image to be overwritten by the rectangular region of the updated image, regardless of the transparency setting.

Lines may be drawn using overlay image data in the following format:

4 bytes: "line"

```
1 or more struct s2253_line {
    int x1;
    int y1;
    int x2;
    int y2;
    int width;
    int argb;
};
```

(struct s2253\_overlay\_image members id, transparent, xOffset, yOffset are not used for line drawing)

The line width parameter is useful for drawing squares, by drawing a vertical line with a large line width. For example, to clear the whole screen send

data:

```
"line"
x1=0
y1=0
x2=0
y2=575
width=720
argb=0
```

Text is drawn using overlay image data in the following format:

4 bytes: "text"

4 bytes: argb color

remaining bytes: text

---

## **S2253\_VIDIOC\_SET\_USER\_DATA**

```
int ioctl(int fd, int request, struct s2253_user_data *data);
struct s2253_user_data {
    char *data;      /* data to be inserted in the stream */
    int len;         /* length of the data */
    int interval;    /* 0=insert once per ioctl,
                    otherwise every Nth frame */
};
```

User data is inserted in the MPEG-4 or H.264 stream, once or at a specified frame interval. Up to 1024 bytes may be inserted per frame. User data may NOT contain MPEG start codes, which means there may not be more than 23 consecutive zero bits in the data. This function will not check for start codes. It is recommended, therefore to separate the data; for instance, if you want to store a 32 bit (4 byte) latitude position in the stream, you could put the high order bytes in the first 2 bytes of user data, followed by a “marker byte” of 0xff, and put the other 2 bytes in the next 2 user data bytes. Eg. For a latitude with hex value 0x70000111, the user data would be data[0] = 0x70, data[1] = 0x00, data[2]=0xff(marker), data[3]=0x01, data[4]=0x11. Similarly, if other user data needs inserted, simply add a marker byte to separate the data into sections.

# Revision history

Version	Notes
1.0.0, June 2010	Initial release.
1.0.1, July 2010	Mplayer examples added.
1.0.2, July 2010	GUI Demo and OSD controls
1.0.3, August 2010	Playback section added.
1.0.4, November 2010	MP4 format and audio controls added.
1.0.5, March 2011	Video output device added.
1.0.6, June 2011	User data ioctl added.
1.2.1, September 2011	Add known limitations
1.2.2, January 2012	Add MPEG-TS and new controls.