

USB Audio/Video Codec Model 2253

Linux Software Manual

Ver.1.0.3 | August 2010

SENSORAY | embedded electronics



Designed and manufactured in the U.S.A

SENSORAY | p. 503.684.8005 | email: info@SENSORAY.com | 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 Driver.....	5
GPIO Driver.....	5
Troubleshooting.....	6
Demo Program.....	6
GUI Demo Program.....	7
Playing Record Files.....	8
SDK REFERENCE.....	9
General Notes.....	9
Driver Ioctl Reference.....	9
VIDIOC_G_FMT, VIDIOC_S_FMT, VIDIOC_TRY_FMT.....	9
VIDIOC_G_JPEGCOMP, VIDIOC_S_JPEGCOMP.....	10
S2253_VIDIOC_OSD.....	10
S2253_VIDIOC_CLOCK.....	11
VIDIOC_ENUM_INPUT, VIDIOC_G_INPUT, VIDIOC_S_INPUT.....	11
VIDIOC_QUERYCTRL, VIDIOC_G_CTRL, VIDIOC_S_CTRL.....	11
VIDIOC_TRY_EXT_CTRL, VIDIOC_G_EXT_CTRL, VIDIOC_S_EXT_CTRL.....	11
REVISION HISTORY.....	13

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.

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 xvfz sdk-2253-linux_vXYZ.tgz,
```

where XYZ is the version of the SDK.

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

```
cd sdk-2253-linux_vXYZ
```

- 3) Build the files:

```
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, Step 4 may need to be performed again to reinstall the modules.

Basic Operation

When the driver is loaded, there are two 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 GPIO char device will be named "s2253_gpioX" where X depends on the number of 2253 devices present in the system.

Video Capture 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 will not be supported for kernels below 2.6.25.

The two 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. Both capture devices record video from a single source, and each capture device can be started, stopped, and configured independently. Some options cannot be configured independently, such as field selection, 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.

GPIO Driver

The GPIO device node is a very basic input/output device. Writing character values 0 or 1 to the device will set the GPO appropriately. Reading from the device will block until the GPI changes state, and return a single character having the value 0 or 1, equal to the state of GPI. The device file descriptor may also be set to non-blocking and the current state of the GPI will be returned immediately. However, the 2253 itself polls the GPI at 1ms intervals, and the non-blocking mode cannot be used to increase the polling frequency. A simple loopback test can be created by piping the s2253_gpio device to itself, and changes on the GPI should be reflected on the GPO.

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`. 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.

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.mp4`” to save a MPEG4 video at 2 Mbps in the file `output.mp4`.

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=yvy: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 capture devices simultaneously.

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 AVI file. The left side controls are used only when starting the stream, and the right side 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/>

Due to the options required, use a terminal to enter the commands.

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
```

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
```

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
```

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. Two `/dev/videoX` nodes will be created for each 2253 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.

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.

Below the `ioctls` 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 768 in steps of 16.

`fmt->pix.height`

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

`fmt->pix.pixelformat`

Supported FOURCC codes:

```
V4L2_PIX_FMT_NV12
V4L2_PIX_FMT_YUYV
V4L2_PIX_FMT_UYVY
V4L2_PIX_FMT_GREY
V4L2_PIX_FMT_MJPEG
V4L2_PIX_FMT_JPEG
```

```
V4L2_PIX_FMT_MP4V defined as v4l2_fourcc('M', 'P', '4', 'V')
V4L2_PIX_FMT_H264 defined as v4l2_fourcc('H', '2', '6', '4')
```

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.

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.

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,          //year value in 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 the top or 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. Changing the clock while streaming will cause embedded timestamps to shift, and it is not advised to do so. The clock on the device does not make adjustments for timezones or daylight-saving.

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".

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.

`V4L2_CID_CONTRAST`: 0 to 255, default 128.

`V4L2_CID_SATURATION`: 0 to 255, default 128.

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

`S2253_CID_INTERPOLATE`:

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

`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_DATE_FORMAT:

MM-DD-YYYY: 0, DD-MM-YYYY: 1, MM-DD-YY: 2, DD-MM-YY: 3, Default 0.

S2253_CID_OSD_DATE_FORMAT:

HH:MM:SS: 0, HH:MM:SS.0: 1, HH:MM:SS.00: 2, Default 0.

S2253_CID_OSD_TIME_ZONE:

Menu containing hour offsets from UTC. Default UTC + 0.

S2253_CID_OSD_MESSAGE:

String.

S2253_CID_OSD_DATE_FORMAT:

Integer range 0 to format width, default 20.

S2253_CID_OSD_DATE_FORMAT:

Integer range 0 to format height, default 20.

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

V4L2_CID_MPEG_VIDEO_BITRATE:

100,000 to 20,000,000 bps in steps of 100,000 bps, default 2,000,000 bps.

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.