Line6 Linux USB driver
Version 0.9.1beta

Markus Grabner

August 23rd, 2010

Contents

1 Introduction

This document describes a Linux USB driver for Line6 hardware such as the POD series and the Variax workbench. Note that the software is really just the driver layer which provides access to the underlying hardware (via the USB interface). However, on top of this driver, applications with a command line or graphical user interface can easily be built.

So far I do not have the USB specifications of these devices, so obviously not all features that are available in the corresponding Windows software distributed by Line6 are supported by this Linux driver.

2 Supported devices

The POD series portion of this driver has been specifically designed for the PODxt Pro product by Line6, which is the only device from the POD series that is available to me. However, users reported that they have been able to use the driver also with other Line6 products. Table 1 shows a list of devices which are recognised by the driver, and the level of support for each device. Note that I can test the driver only with few of these devices myself, support statements for other devices are based on user feedback. If you have problems with one of the devices listed as supported, please contact me.

id	name	support
0x4250	BassPODxt	yes
0x4642	BassPODxt Live	yes
0x4252	BassPODxt Pro	yes
0x4750	GuitarPort	yes
0x5051	Pocket POD	yes
0x4153	POD Studio GX	yes
0x4150	POD Studio UX1	yes
0x4151	POD Studio UX2	yes
0x414a	POD X3	no
0x414b	POD X3 Live	no
0x5044	PODxt	yes
0x4650	PODxt Live*)	yes
0x5050	PODxt Pro*)	yes
0x4147	TonePort GX*)	yes
0x4141	TonePort UX1	yes
0x4142	TonePort UX2	yes
0x534d	Variax workbench*)	partial

Table 1: Devices recognized by the driver, those marked with "*)" are available for me for testing

If your Line6 product is not yet included in this list, you can easily determine its product id by the command

```
lsusb -d 0e41:
```

0e41 is Line6' vendor id, so output is restricted to Line6 devices (this does not yet require the driver to be loaded). The output should be something like

```
Bus 001 Device 004:  ID 0e41:5050 Line6, Inc.  PODxt Pro
```

where 5050 is the product id of the PODxt Pro.

If you are successful in using the driver for a Line6 product different from the ones listed in Table 1 by modifying the arrays `line6_id_table` and `line6_name_table` in the file `driver.c` accordingly, please let me know which device you are using and which product id you found out.

Support for the Variax workbench is limited to the actual controls on the guitar (tone, volume, model) and read-only access to the guitar model parameters.

3 User-space access

Since the driver operates in kernel space, it has to provide an interface to the supported features for user-space programs. The current device settings are available via sysfs, i.e., a directory is automatically created under `/sys/bus/usb/devices` as soon as the Line6 device is connected. The name of this newly created directory (called *sysfs directory* in the following) depends on the actual USB bus number the device is connected to. A small utility is included in the driver package to identify this directory. The command

```
line6_find_device.pl sysdir POD
```

prints the sysfs directory assigned to he first Line6 device on the bus whose name matches "POD". The script `create_links.pl` creates easy to remember symbolic links to the sysfs directories of all detected Line6 devices. The links are created in the current directory and have the form *name*:*interface* (e.g., `PODxtLive:1` for the Variax interface of the PODxt Live). Note, however, that the links will become invalid if you disconnect the device and connect it to a different USB port.

There also exists a raw MIDI interface for binary access and blocking reads (see Section 4.4), which are features typically required by applications that interactively control an external device.

4 POD series driver features

This is a brief list of supported features. See below for details on how to access them.

- Reading/writing individual parameters
- Reading/writing complete channel, effects setup, and amp setup data
- Channel switching
- Tuner access
- Playback/capture/mixer device for any ALSA-compatible PCM audio application
- Signal routing (record clean/processed guitar signal, re-amping)
- Development tools

4.1 POD series parameter setting

The driver supports three types of access to POD's parameter settings. Accessing single parameters (Section 4.1.1) is most useful for shell scripts and on the command line. Batch access (Sections 4.1.2 and 4.1.3) is used to load or save all parameters of a channel, effects setup, or amp setup in a single step. Interactive (GUI) applications should use the MIDI interface instead (Section 4.4), which supports blocking mode and avoids continuous polling of the device.

amp1_engage	mid
amp_model	mod_enable
amp_model_setup	mod_param_1
amp_switch_select	mod_param_1_double_precision
band_1_frequency	mod_param_1_note_value
band_1_gain	mod_param_2
band_2_frequency	mod_param_3
band_2_gain	mod_param_4
band_3_frequency	mod_param_5
band_3_gain	mod_pre_post
band_4_frequency	mod_volume_mix
band_4_gain	modulation_lo_cut
band_5_frequency	modulation_model
band_5_gain	noise_gate_enable
band_6_frequency	pan
band_6_gain	presence
bass	reverb_decay
bypass_volume	reverb_enable
cabinet_model	reverb_mix
chan_vol	reverb_pre_delay
comp_enable	reverb_pre_post
compression_gain	reverb_tone
compression_threshold	reverb_type
delay_enable	roomlevel
delay_model	stomp_enable
delay_param_1	stomp_model
delay_param_1_double_precision	stomp_param_1_note_value
delay_param_1_note_value	stomp_param_2
delay_param_2	stomp_param_3
delay_param_3	stomp_param_4
delay_param_4	stomp_param_5
delay_param_5	stomp_param_6
delay_pre_post	stomp_time
delay_reverb_lo_cut	tap
delay_verb_model	tempo_lsb
delay_volume_mix	tempo_msb
di_delay	treble
di_model	tuner
drive	tweak
effect_setup	tweak_param_select
eq_enable	vol_pedal_position
eq_pre_post	volume_pedal_minimum
fx_loop_on_off	volume_pre_post
gate_decay_time	volume_tweak_pedal_assign
gate_threshold	wah_enable
highmid	wah_model
lowmid	wah_position
mic_selection	

Table 2: POD series control parameters corresponding to Appendix C of the “PODxt (Pro) Pilot’s Handbook” by Line6

name	access	description
bypass_volume	read/write	signal volume when processing unit is bypassed
channel	read/write	the current channel number (write an ASCII number to it to switch to a different channel)
clip	read only	wait for audio clipping (see Section 4.3.5)
device_id	read only	numeric value identifying a member of the POD family
dirty	read only	1 if any channel parameter has been modified since the last channel recall, 0 otherwise
firmware_version	read only	firmware version of the POD device (as displayed on the last page in the SYSTEM menu)
fx_loop_on_off	read/write	turns on (64-127) or off (0-63) the analog effects loop (this parameter is omitted from Appendix C of the Line6 manual)
midi_mask	read/write	MIDI channel mask (see Section 4.4)
midi_postprocess	read/write	MIDI postprocessing flag (see Section 4.4)
name	read only	name of the current channel
serial_number	read only	serial number of device

Table 3: Additional POD series control parameters (including those introduced with firmware version 2.0 or later)

4.1.1 Single parameter access

In the sysfs directory, the driver creates several files corresponding to the "PODxt MIDI Controls" specified in Appendix C of the "PODxt (Pro) Pilot's Handbook" by Line6 and the document "MIDI Continuous Controller Reference"[1] (see also the file `control.h`). Assume for the following that this directory is the current working directory. If you modify the settings at your POD device, the modifications are reflected in the "content" of these virtual files. So the command

```
cat treble
```

will report the current position of the "treble" dial in the range of 0-127 in ASCII representation. Likewise, by using

```
echo 64 > presence
```

you can set the presence dial to the position half way between minimum and maximum (of course the dial is not moved physically, but you can observe the changes in the POD device display if the device is in edit mode). A complete list of parameters that can be manipulated in a similar way is given in Tables 2 and 3.

4.1.2 Batch access (current channel)

You can access all current channel parameters with a single read or write access to the special file `dump`, which is a binary representation of the current channel settings. You can save your favorite sound by using

```
cp dump ~/my_favorite_sound.pod
```

and restore it later to the currently active channel by

```
cp ~/my_favorite_sound.pod dump
```

If you are lucky enough to own both a PODxt Pro and a PODxt Live, you can even copy a channel from one device to the other by the following command (see also Section 3):

```
cp PODxtPro:0/dump PODxtLive:0/dump
```

[1] available from `http://www.line6.com/support/manuals`, authentication required

name	access	range	description
dump_buf	read/write	160 bytes	parameter buffer for access to internal memory
retrieve_channel	write	0-127	number of channel to retrieve into dump_buf
store_channel	write	0-127	number of channel to store contents of dump_buf to
retrieve_effects_setup	write	0-63	number of effects setup to retrieve into dump_buf
store_effects_setup	write	0-63	number of effects setup to store contents of dump_buf to
retrieve_amp_setup	write	0-63	number of amp model to retrieve into dump_buf
store_amp_setup	write	0-63	number of amp model to store contents of dump_buf to
finish	write	–	execute previously buffered "store" operations
name_buf	read	16 bytes	name of the most recently retrieved channel, effects setup, or amp setup

Table 4: Access to PODxt Pro's internal memory

Writing to dump does not store the settings in the POD device internal memory, i.e., your changes will be lost as soon as you switch to a different channel (unless you pressed "SAVE" twice before). A method for permanent storage is presented in the next section.

4.1.3 Batch access (any channel, effects setup, and amp setup)

You can also access data different from the current channel. Writing an ASCII number (in the range 0-127) to the special file retrieve_channel requests transmission of all data for this channel. Once completed, data are stored by the driver in an internal buffer, which can be accessed via dump_buf. Note that reading from this file blocks the caller until the transmission of previously requested data is actually completed.

Storing data to an arbitrary channel follows a similar procedure. First you have to copy your data to the file dump_buf, then the required channel number has to be written to store_channel. This procedure can be repeated for several channels and must be followed by a single write access (with arbitrary data) to finish. Accessing effects and amp setups is similar, however, fewer storage places are available. See Table 4 for a summary of these features.

Storing data to the POD device requires channel numbers (written to store_channel) to strictly increase, otherwise the device will crash after receiving the finish command. The same has been observed for mixing data for channels, effects setups, and amp setups.

The script retrieve_all_data.sh copies all channel, effects setup, and amp setup data into a newly created directory data. Likewise, the script store_all_data.sh restores the POD device to the previously stored state.

4.2 Tuner

There are four special files for tuner access (see Table 5). tuner_note reports the current note as an integer. It is incremented by one with each half tone, see Table 6 for the values for standard guitar tuning. The deviation from the ideal frequency is given in cents (from -50 to 50) and can be accessed via the file tuner_pitch. If note and pitch can't be determined (e.g., when more strings are hit simultaneously), the

driver returns the values -4 and -104 for note and pitch, respectively. The tuner's base frequency (typically 440Hz) and the muting mode can be read from and written to the files `tuner_freq` and `tuner_mute`, respectively. The script `tuner.sh` demonstrates these features.

name	access	description
`tuner_note`	read only	current note
`tuner_pitch`	read only	current pitch
`tuner_freq`	read/write	tuner base frequency in Hertz (typically 440)
`tuner_mute`	read/write	tuner muting mode (0: bypass, 1: mute)

Table 5: Special files for tuner access

E	A	D	G	B	E
17	22	27	32	36	41

Table 6: Guitar strings and their corresponding values reported by the tuner

Transmission of these values is never initiated by the POD device, but must be explicitly requested by the PC. Therefore reading from these files always blocks the calling process for a short time.

Access to the tuner files requires that the tuner is actually switched on. Otherwise a read attempt results in an error (error code `-ENODEV`).

4.3 PCM audio

The driver registers the POD device as an ALSA device that can be used by any ALSA-compatible PCM audio application (e.g., `xmms` and `alsaplayer`). Refer to the documentation of your audio application on how to select a particular device for capture and/or playback. You can verify the registration of the ALSA device by the command

```
aplay -l
```

The output should contain some lines similar to

```
card 2:  PODxtLive [PODxt Live], device 0:  PODxt Live [PODxt Live]
  Subdevices:  1/1
  Subdevice #0:  subdevice #0
```

4.3.1 Audio playback

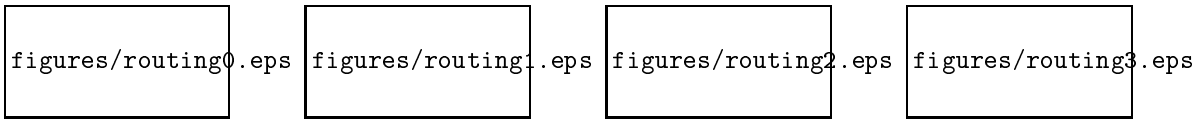
The ALSA device name you should use for playback is `plughw:n` (where `n` is the card number as reported by `aplay -l`). Use the following command to play a WAV-file:

```
aplay -D plughw:n my_audio_file.wav
```

See the file `aplay.sh` for an example. Note that `aplay` doesn't decode MP3, but instead tries to play it as raw audio, which produces nothing but (white) noise! Use the plain `hw` as the mixer, e.g.

```
alsamixergui -D hw:2
```

Turn the master volume ("OUTPUT" dial) all the way down to zero before using the POD device as an ALSA device, especially when using headphones! PCM audio can be much louder than the guitar signal!



(a) mode 0: send processed guitar

(b) mode 1: send clean guitar

(c) mode 2: send clean guitar re-amp playback

(d) mode 3: send re-amp playback

Figure 1: different signal routing options

4.3.2 Audio capture

Audio capture is also implemented via the ALSA interface. The file `arecord.sh` is an example of how to call ALSA's recording utility `arecord` for recording either in POD's native format or with resampling to a more common format.

4.3.3 Synchronized playback and capture

Sometimes it is required that the playback and capture channel are triggered exactly at the same time (e.g., recording with a pilot track, re-amping). This feature is provided by the ALSA user-level library (function `snd_pcm_link`) and is supported by this driver. It is demonstrated in the accompanying application `reamping`, which routes a previously recorded dry guitar signal through the POD device and stores the processed signal (see Section 4.3.4 for details on signal routing). The program is invoked as

```
reamping dry_signal.wav processed_signal.wav
```

The default is to use ALSA device `hw:0`, use the option `-D` to select a different one (in the same way as with the command `aplay`). Currently only WAV files in POD's native sampling format are supported. A compatible dry guitar signal can be recorded by the script `record_dry.sh`. Both programs are not part of the driver distribution, but contained in a separate package.

4.3.4 Signal routing

Normally audio data transmitted to the PC by the POD device have been processed by the amp and effects modeler stages, and audio data sent to the device are directly output without further processing. However, it is also possible to record the clean guitar signal (while still hearing the processed signal, e.g., in the headphones), and to apply the amp and effects processing in a second stage (*re-amping*). Four different configurations exist (see Figure 1), switching between them is accomplished by writing one of the numbers 0 to 3 (as ASCII) to the special file `routing`. The monitoring level of the guitar signal can be read from and written to `monitor_level`, which ranges from 0 to 65535.

Figures 1a to 1d correspond to those displayed in the PODxt system control panel under Windows, with the exception of figure 1d. The Windows driver forces the instrument monitoring level to zero in re-amping mode (which is most likely what you want). The Linux driver leaves this responsibility to the user, giving additional flexibility to create fancy effects.

4.3.5 Audio clipping

Under some conditions (e.g., if channel volume and compressor gain are set to high values), the processed signal exceeds the POD's 24 bit audio range and must be clipped. This is clearly audible, and an application might want to warn the user about it. A read request on the `clip` special file blocks the calling process until an audio clipping event occurs. The return value is empty.

4.3.6 Notes on sample conversion

The POD device uses a rather uncommon sample format, namely 24 bit audio (little endian, stored in 3 bytes) at 39062.5 Hz sampling frequency[2]. ALSA only supports integer sampling frequencies, so I hope you don't mind that I rounded it up to 39063 :-) But more important, since you probably don't have audio files recorded with exactly this frame rate and sample format, you have to use ALSA's built-in conversion utilities by playing to `plughw:n` instead of `hw:n`. The 16→24 bit conversion is supported in ALSA-1.0.4 and higher, without that the player simply refuses to use the device. The sample rate conversion has been significantly improved in ALSA-1.0.6, but is still not perfect. Depending on the frequency distribution of the music you want to listen to, this might be a problem or not (e.g., Gary Moore's guitar on "Still Got The Blues" sounds very strange, while his vocals are ok). If you need better sound quality, you should use an external sample rate conversion utility such as the one included in `sox`[3]. The small script `podxtpro_resample.sh` feeds `sox` with the parameters to do this conversion, such that

```
podxtpro_resample.sh my_song.mp3 my_song_resampled.wav
```

creates `my_song_resampled.wav` (has to be WAV, unfortunately), which is perfectly suitable for playback on the POD device, from `my_song.mp3`, which can be stored at any sample frequency and format that is supported in MP3. Unless you plan to use only the parameter setting features of the driver, you need a recent version of the Linux kernel and ALSA (see `INSTALL`).

A patch to fix the resampling problem of ALSA-1.0.6 is also included in this package. This patch is already incorporated in ALSA-1.0.7, so if you use this version, you neither need to manually resample your audio files nor to patch and recompile the ALSA library. The same is true for SuSE's version of ALSA-1.0.6 distributed with SuSE Linux 9.2, which also includes the patch.

ALSA-1.0.7 has a bug which causes the ALSA driver to crash when recording audio with resampling via the `plughw` device, and ALSA-1.0.8 contains a broken resampler. The resampler of ALSA-1.0.9 has been improved, but has a bug that produces noise when resampling a mono audio stream. This has been fixed in ALSA-1.0.10rc2, which is is therefore highly recommended.

4.3.7 Hard disc recording

The driver has been successfully used to record music with the open source digital audio workstation software `ardour`[4]. This requires the `jack`[5] daemon to be started like

```
jackd -d alsa -d hw:1
```

However, this restricts all recordings to be taken at POD's sample rate of 39063 Hz. Exporting audio to a more common sample rate introduces some "tinny" artifacts (due to up- and subsequent downsampling) that are not present when playing the recorded tracks in `ardour`. A simple demo is available on the driver web site[6].

4.3.8 ALSA controls

The driver registers two volume controls, one for the PCM playback volume, and one for the monitor level (see Section 4.3.4). Any ALSA mixer application can be used to adjust the volume settings, best results are achieved with the KDE4 program `kmix`.

[2] This seems to be derived from the USB clock frequency (10 MHz) by a frequency divider (1/256).
[3] `http://sox.sourceforge.net`
[4] `http://www.ardour.org`
[5] `http://jackit.sourceforge.net`
[6] `http://www.tanzband-scream.at/line6/download/Man_On_The_Mill.mp3`

4.4 MIDI

In addition to the ASCII based special files in the sysfs directory, the driver registers a raw MIDI interface for binary access to the POD device. It can be accessed via ALSA just like any MIDI-capable sound card. However, you don't have to connect a MIDI cable, data are still transmitted and received via the USB cable, and the driver transparently provides the required translation.

Similar to the procedure to identify the PCM device (Section 4.3), you can use the command

```
amidi -l
```

to query existing MIDI interfaces on your system. The output should contain a line similar to

```
hw:1,0 PODxt Pro
```

You will most likely not want to manually communicate with the binary MIDI interface. You can access it in your programs via the ALSA MIDI API or via a device file created by ALSA (which is called `/dev/snd/midiC1D0` for the example above).

The main differences between the MIDI interface and the `raw` special file explained in Section 7 are as follows:

- The MIDI interface is bidirectional.
- The MIDI output interface can be filtered according to a given channel mask. A MIDI command (with the command byte in the range from `0x80` to `0xef`) is transmitted only if the bit corresponding to the MIDI channel is set in the `midi_mask` special file (which is "1" by default, i.e., only channel 0 is allowed for output). Note that the MIDI channel is encoded in the four lower bits of the command byte.
- Data transmitted via the MIDI interface can be interpreted to a certain extent (e.g., requesting a channel dump on channel change). To activate this feature, write "1" to the `midi_postprocess` special file (write "0" to deactivate, which is the default).

4.5 Missing features

- Maintenance functions (such as firmware update)
- More consistency checks to avoid sending commands to the POD device that might cause it to crash
- Probably many more

5 TonePort driver features

5.1 PCM audio

The TonePort and similar devices can be used as full duplex ALSA PCM devices. Note, however, that audio processing (e.g., amp modelling) is done in software on the PC in the original Line6 product. The corresponding parameters of the POD series (Section 4.1) are therefore not available for TonePort devices. Moreover, the TonePort doesn't support hardware monitoring, this feature is therefore emulated by the driver. Note that the driver can directly forward any audio data packet received via the capture channel to the playback channel, which gives a latency of 2ms. In contrast, software such as jack typically maintains larger buffers, which increases latency. Be sure not to enable the driver's monitoring feature and other software monitoring methods at the same time. The same ALSA controls are available for the TonePort as for the POD series (Section 4.3.8). Playback and monitor level should not be set to the maximum value at the same time since this can easily cause overdrive and a noisy signal.

The intensities of the two built-in LEDs (red and green) can be modified by writing values from 0 to 38 to the special files in Table 7. Larger values cause the corresponding LED to blink.

name	description
led_green	intensity of green LED
led_red	intensity of red LED

Table 7: TonePort control parameters

name	description
Microphone	microphone input
Line	line input (stereo)
Instrument	instrument input
Inst & Mic	instrument and microphone input (mixed into a single signal)

Table 8: possible values for “PCM Capture Source” switch of UX1 devices

5.2 Source Select

The UX1 devices have different audio inputs, which can be selected for capturing by means of a mixer control with the name PCM Capture Source. Use an application like alsamixer[7] or kmix to modifiy the value of the source select switch. Possible values are listed in Table 8.

I assume that similar features are available for UX2 devices, but didn’t yet receive any feedback on this, hence the UX2 don’t have the source select switch.

6 Variax driver features

Most of the Variax parameters are read-only in the current version of the driver. Details are given in the following subsections.

6.1 Read/write parameters

The Variax driver is similar to the POD series driver in the way data is accessed via the sysfs directory (see Section 4.1.1) and via MIDI (see Section 4.4). However, only few parameters are available with read/write access in the current version (see table 9). The Variax interface supports explicit activation or deactivation by writing “1” or “0” to the parameter active, respectively. This is useful to avoid “zipper noise” which is audible when the interface is active and one of the guitar’s dials is turned. The interface is initially active on driver startup.

name	description
model	current model (encodes both the model dial and the pickup selector)
tone	current position of tone dial
volume	current position of volume dial
active	current activation status

Table 9: Variax workbench control parameters with read/write access

body	mix6	save_tone
capacitance	pickup1_angle	taper
detune1	pickup1_enable	tone_dump
detune2	pickup1_level	tone_resistance
detune3	pickup1_position	tuning1
detune4	pickup1_type	tuning2
detune5	pickup2_angle	tuning3
detune6	pickup2_enable	tuning4
mix1	pickup2_level	tuning5
mix2	pickup2_position	tuning6
mix3	pickup2_type	tuning_enable
mix4	pickup_phase	volume_dump
mix5	pickup_wiring	volume_resistance

Table 10: Variax model parameters (currently read-only)

name	access	description
bank	read only	name of current bank
dump	read only	binary representation of current model
name	read only	name of current model

Table 11: Additional Variax parameters

6.2 Guitar model parameters

Table 10 lists the parameters that describe the current guitar model. Most of them are self-explanatory and correspond to similarly named controls in the "Variax Workbench" software delivered with the Variax USB interface by Line6. The parameter `pickup_wiring` refers to the "series/parallel" switch, where "1" means "series" and "0" means "parallel". The `tone_dump` and `volume_dump` parameters are stored with the guitar model, in contrast to `tone` and `volume`, which reflect the current position of the corresponding dial on the guitar.

6.3 Other parameters

Table 11 lists a few other parameters which provide access to the current model data and associated meta data. The `dump` special file behaves similar as the one provided by the POD driver (see Section 4.1.2), except the fact that it is read-only in the current version.

The Variax driver can also be configured to provide the `raw` special file (see Section 7).

7 Driver configuration

7.1 Standard kernel setup

The driver supports some optional features, which are mainly used for debugging purposes and therefore disabled by default. You can enable them by invoking

```
make xconfig
```

[7] Note that the GUI variant of the ALSA mixer (`alsamixergui`) does not work for this purpose.

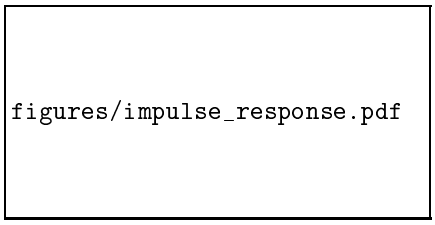


Figure 2: impulse response measurement mode

in the driver source code directory, searching for Line6-related settings (menu Edit→Find), and checking the appropriate boxes. Documentation of each feature is available in the configuration software. You will be asked for the root password since this modifies the configuration header file of the running kernel. Be careful not to modify any settings not related to the Line6 driver. This won't modify your kernel's behaviour (unless you recompile it), but it might confuse other software which looks up the kernel configuration at this place.

7.2 Impulse response measurement

When compiled with the "measure impulse response" option, the driver provides a mode in which a sequence of impulses is sent both to the right output of the device and to the right capture channel of the ALSA subsystem (see Figure 2). The impulse sequence should be returned to the input of the device via an external audio cable, and the returned signal is provided to the left output of the device and the left capture channel. The delay between the impulses on the right and left capture channel is the total A/D and D/A latency of the device. The same measurement could be obtained by connecting the outputs of the device to an oscilloscope, but I never tried to do so.

The impulse response measurement mode is entered by writing a value larger than 0 to the special file `impulse_volume`. This value defines the amplitude of the impulses and ranges from 0 (off) to 127 (maximum). Moreover, the period of the impulse sequence can be modified by writing the desired time in milliseconds between two impulses to the special file `impulse_period` (the default period is 100ms). The driver returns to normal operation by writing 0 to `impulse_volume`.

7.3 Checkpoints

The driver source code can be modified such that a checkpoint (file name and line number) is written to the syslog before (almost) every statement, which is useful for detailed tracing of the driver's operation. This optional feature is documented in the file `checkpoint/README.txt`.

8 Driver test suite

The driver source code directory contains a tool (`testsuite.sh`) to test various features of the Line6 Linux driver. It is started without parameters and offers the user several options to test Line6 devices connected to the system. The tool is meant to be self-explanatory and interactively guides the user through the testing process (the `kdialog` program distributed with KDE is required for user interaction, I am ready to accept patches for other desktop environments). The user is asked whether the tests were successful (e.g., "did you hear what you recorded?"). These test results and the system configuration are written to a log file (`testsuite.log`).

9 Known issues

9.1 POD series

Due to some undocumented timing issues it takes several seconds at driver startup until the current channel data has been received (i.e., the contents of the sysfs directory are valid). Moreover, the Variax driver sometimes fails to properly initialize, requiring to unplug and reconnect the Workbench device from the PC. The current channel number is only valid after it has been changed at least once.

The driver is not really paranoid about being unloaded while in use. While it survives disconnecting the USB cable during operation, there might be situations where it causes the machine to crash.

On the AMD64 platform, repeatedly performing batch writes (Section 4.1.3) can cause the USB subsystem to hang, requiring to reload all involved kernel modules.

Some problems that can be experienced with the audio subsystem of the driver are actually related to old versions of ALSA. See Section 4.3.6 for a discussion.

9.2 Variax

Under some (yet to be examined) conditions the driver blocks the calling process when it tries to access its sysfs directory. If you encounter this problem, disconnect and then reconnect the Variax workbench from the PC (it may remain connected to the Variax).

10 Feedback

Any kind of feedback regarding the driver is highly appreciated. Please send an E-mail to the author (`grabner@icg.tugraz.at`) and include the following data in your report:

- Line6 Linux USB driver version
- Linux kernel version
- ALSA library version
- Linux distribution

11 ChangeLog

11.1 Kernel driver

- version 0.9.1
  - improved support for TonePort/POD Studio GX
  - added support for TonePort/POD Studio UX1 and UX2
  - support for Linux kernel $\geq$ 2.6.35
  - using standard Linux kernel configuration method
  - improved device initialization
  - power management handling (continue audio I/O after suspend/resume)
  - test suite for interactive driver testing

  - source code instrumentation tool (write checkpoints to syslog)
  - optional impulse response measurement
- version 0.9.0 (never released officially)
- version 0.8.1
  - signal monitoring for TonePort GX
  - ALSA control for monitor level
  - reduced latency
- version 0.8.0
  - support for TonePort GX
- version 0.7.3
  - support for Linux kernel $\geq$ 2.6.23
  - MIDI channel mask
  - Debian packaging
- version 0.7.2
  - support for Linux kernel $\geq$ 2.6.21
  - Variax workbench issue fixed
- version 0.7.1
  - fixed support for Linux kernel $<$ 2.6.18
- version 0.7.0
  - support for Linux kernel $\geq$ 2.6.18
  - improved support for PODxt Live
  - improved support for multiple devices
- version 0.6.5
  - ALSA/jack issue fixed
- version 0.6.4
  - consider model and firmware version when creating special files
- version 0.6.3
  - device registration (e.g., for HAL)
- version 0.6.2
  - floating point issues fixed
- version 0.6.1
  - read-only access to Variax guitar model parameters
  - access to POD firmware version and device id
  - some kernel version and platform specific issues fixed
- version 0.6
  - preliminary support for the Variax workbench
  - Line6 device autodetection in demo scripts
- version 0.5.3

  - some issues related to particular kernel versions fixed
- version 0.5.2
  - support for Bass PODxt Live
- version 0.5.1
  - bug fixed (noise after repositioning output stream pointer)
- version 0.5
  - raw MIDI interface
  - “pause” operation implemented
  - support for Linux kernel $\geq$ 2.6.12
  - support for PPC platform
  - query serial number
  - some controls renamed according to updated Line6 manual
- version 0.4
  - improved audio interface (including synchronized playback/capture)
  - report when “SAVE” button pressed twice
  - support for PODxt Pro controls introduced with firmware v2.14
- version 0.3
  - tuner access
  - signal routing
  - store channels, amp setups, and effects setups
  - AMD64 support
- version 0.2.2
  - compatibility with kernel 2.6.8
- version 0.2.1
  - more device ids
- version 0.2
  - audio capture
  - support for PODxt Pro controls introduced with firmware v2.0
- version 0.1
  - audio playback
  - PODxt Pro parameter access

11.2 Re-amping tool

- version 0.1
  - re-amping in PODxt Pro native audio format
  - dry guitar record script

12 Disclaimer

As this is experimental software, I certainly cannot be held responsible if your hardware goes up in flames or has its firmware erased due to the use of this driver, or you suffer hearing damage from not following the advice in Section 4.3.1. Neither of these events, however, occurred during development of the driver, so feel free to use this software, but use it at your own risk!

Or, to say it with the words of the GPL:

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.