Via 686 Audio Driver for Linux

Jeff Garzik

Via 686 Audio Driver for Linux

by Jeff Garzik

Table of Contents

1. Introduction **1**
2. Driver Installation **1**
3. Submitting a bug report **2**
3.1. Description of problem 2
3.2. Diagnostic output 2
3.3. Driver debug output 2
3.4. Bigger kernel message buffer 2
4. Known Bugs And Assumptions **4**
5. Thanks **5**
6. Random Notes **6**
7. Driver ChangeLog **7**
7.1. Version 1.9.1 7
7.2. Version 1.1.15 7
7.3. Version 1.1.14 7
7.4. Version 1.1.12 8
7.5. Version 1.1.11 8
7.6. Version 1.1.10 8
7.7. Version 1.1.9 8
7.8. Version 1.1.8 9
7.9. Version 1.1.7 9
7.10. Version 1.1.6 10
7.11. Version 1.1.5 10
7.12. Version 1.1.4 10
8. Internal Functions **11**
via_chan_stop 11
via_chan_status_clear 12
sg_begin 13
via_syscall_down 13
via_stop_everything 14
via_set_rate 15

via_chan_init_defaults 17
via_chan_init 18
via_chan_buffer_init 19
via_chan_free 20
via_chan_pcm_fmt 21
via_chan_clear 22
via_chan_set_speed 23
via_chan_set_fmt 24
via_chan_set_stereo 26
via_chan_dump_bufs 27
via_chan_flush_frag 28
via_chan_maybe_start 29
via_ac97_wait_idle 30
via_ac97_read_reg 31
via_ac97_write_reg 32
via_intr_channel 33
via_interrupt_init 35
via_dsp_drain_playback 35
via_dsp_ioctl_space 37
via_dsp_ioctl_ptr 38

Chapter 1. Introduction

The Via VT82C686A "super southbridge" chips contain AC97-compatible audio logic which features dual 16-bit stereo PCM sound channels (full duplex), plus a third PCM channel intended for use in hardware-assisted FM synthesis.

The current Linux kernel audio driver for this family of chips supports audio playback and recording, but hardware-assisted FM features, and hardware buffer direct-access (mmap) support are not yet available.

This driver supports any Linux kernel version after 2.4.10.

Please send bug reports to the mailing list <`linux-via@gtf.org`>. To subscribe, e-mail <`majordomo@gtf.org`> with

```
subscribe linux-via
```

in the body of the message.

Chapter 2. Driver Installation

To use this audio driver, select the CONFIG_SOUND_VIA82CXXX option in the section Sound during kernel configuration. Follow the usual kernel procedures for rebuilding the kernel, or building and installing driver modules.

To make this driver the default audio driver, you can add the following to your /etc/conf.modules file:

```
alias sound via82cxxx_audio
```

Note that soundcore and ac97_codec support modules are also required for working audio, in addition to the via82cxxx_audio module itself.

Chapter 3. Submitting a bug report

3.1. Description of problem

Describe the application you were using to play/record sound, and how to reproduce the problem.

3.2. Diagnostic output

Obtain the via-audio-diag diagnostics program from http://sf.net/projects/gkernel/ and provide a dump of the audio chip's registers while the problem is occurring. Sample command line:

```
./via-audio-diag -aps > diag-output.txt
```

3.3. Driver debug output

Define `VIA_DEBUG` at the beginning of the driver, then capture and email the kernel log output. This can be viewed in the system kernel log (if enabled), or via the dmesg program. Sample command line:

```
dmesg > /tmp/dmesg-output.txt
```

3.4. Bigger kernel message buffer

If you wish to increase the size of the buffer displayed by dmesg, then change the `LOG_BUF_LEN` macro at the top of linux/kernel/printk.c, recompile your kernel, and pass the `LOG_BUF_LEN` value to dmesg. Sample command line with `LOG_BUF_LEN` == 32768:

```
dmesg -s 32768 > /tmp/dmesg-output.txt
```

Chapter 4. Known Bugs And Assumptions

Low volume

Volume too low on many systems. Workaround: use mixer program such as xmixer to increase volume.

Chapter 5. Thanks

Via for providing e-mail support, specs, and NDA’d source code.

MandrakeSoft for providing hacking time.

AC97 mixer interface fixes and debugging by Ron Cemer <roncemer@gte.net>.

Rui Sousa <rui.sousa@conexant.com>, for bugfixing MMAP support, and several other notable fixes that resulted from his hard work and testing.

Adrian Cox <adrian@humboldt.co.uk>, for bugfixing MMAP support, and several other notable fixes that resulted from his hard work and testing.

Thomas Sailer for further bugfixes.

Chapter 6. Random Notes

Two /proc pseudo-files provide diagnostic information. This is generally not useful to most users. Power users can disable CONFIG_SOUND_VIA82CXXX_PROCFS, and remove the /proc support code. Once version 2.0.0 is released, the /proc support code will be disabled by default. Available /proc pseudo-files:

```
/proc/driver/via/0/info
/proc/driver/via/0/ac97
```

This driver by default supports all PCI audio devices which report a vendor id of 0x1106, and a device id of 0x3058. Subsystem vendor and device ids are not examined.

GNU indent formatting options:

```
-kr -i8 -ts8 -br -ce -bap -sob -l80 -pcs -cs -ss -bs -di1 -nbc -lp -
psl
```

Via has graciously donated e-mail support and source code to help further the development of this driver. Their assistance has been invaluable in the design and coding of the next major version of this driver.

The Via audio chip apparently provides a second PCM scatter-gather DMA channel just for FM data, but does not have a full hardware MIDI processor. I haven’t put much thought towards a solution here, but it might involve using SoftOSS midi wave table, or simply disabling MIDI support altogether and using the FM PCM channel as a second (input? output?)

Chapter 7. Driver ChangeLog

7.1. Version 1.9.1

- DSP read/write bugfixes from Thomas Sailer.
- Add new PCI id for single-channel use of Via 8233.
- Other bug fixes, tweaks, new ioctls.

7.2. Version 1.1.15

- Support for variable fragment size and variable fragment number (Rui Sousa)
- Fixes for the SPEED, STEREO, CHANNELS, FMT ioctls when in read & write mode (Rui Sousa)
- Mmaped sound is now fully functional. (Rui Sousa)
- Make sure to enable PCI device before reading any of its PCI config information. (fixes potential hotplug problems)
- Clean up code a bit and add more internal function documentation.
- AC97 codec access fixes (Adrian Cox)
- Big endian fixes (Adrian Cox)
- MIDI support (Adrian Cox)
- Detect and report locked-rate AC97 codecs. If your hardware only supports 48Khz (locked rate), then your recording/playback software must upsample or downsample accordingly. The hardware cannot do it.
- Use new pci_request_regions and pci_disable_device functions in kernel 2.4.6.

7.3. Version 1.1.14

- Use VM_RESERVE when available, to eliminate unnecessary page faults.

7.4. Version 1.1.12

- mmap bug fixes from Linus.

7.5. Version 1.1.11

- Many more bug fixes. mmap enabled by default, but may still be buggy.
- Uses new and spiffy method of mmap'ing the DMA buffer, based on a suggestion from Linus.

7.6. Version 1.1.10

- Many bug fixes. mmap enabled by default, but may still be buggy.

7.7. Version 1.1.9

- Redesign and rewrite audio playback implementation. (faster and smaller, hopefully)

- Implement recording and full duplex (DSP_CAP_DUPLEX) support.
- Make procfs support optional.
- Quick interrupt status check, to lessen overhead in interrupt sharing situations.
- Add mmap(2) support. Disabled for now, it is still buggy and experimental.
- Surround all syscalls with a semaphore for cheap and easy SMP protection.
- Fix bug in channel shutdown (hardware channel reset) code.
- Remove unnecessary spinlocks (better performance).
- Eliminate "unknown AFMT" message by using a different method of selecting the best AFMT_xxx sound sample format for use.
- Support for realtime hardware pointer position reporting (DSP_CAP_REALTIME, SNDCTL_DSP_GETxPTR ioctls)
- Support for capture/playback triggering (DSP_CAP_TRIGGER, SNDCTL_DSP_SETTRIGGER ioctls)
- SNDCTL_DSP_SETDUPLEX and SNDCTL_DSP_POST ioctls now handled.
- Rewrite open(2) and close(2) logic to allow only one user at a time. All other open(2) attempts will sleep until they succeed. FIXME: open(O_RDONLY) and open(O_WRONLY) should be allowed to succeed.
- Reviewed code to ensure that SMP and multiple audio devices are fully supported.

7.8. Version 1.1.8

- Clean up interrupt handler output. Fixes the following kernel error message:

  ```
  unhandled interrupt ...
  ```

- Convert documentation to DocBook, so that PDF, HTML and PostScript (.ps) output is readily available.

7.9. Version 1.1.7

- Fix module unload bug where mixer device left registered after driver exit

7.10. Version 1.1.6

- Rewrite via_set_rate to mimic ALSA basic AC97 rate setting
- Remove much dead code
- Complete spin_lock_irqsave -> spin_lock_irq conversion in via_dsp_ioctl
- Fix build problem in via_dsp_ioctl
- Optimize included headers to eliminate headers found in linux/drivers/sound

7.11. Version 1.1.5

- Disable some overly-verbose debugging code
- Remove unnecessary sound locks
- Fix some ioctls for better time resolution
- Begin spin_lock_irqsave -> spin_lock_irq conversion in via_dsp_ioctl

7.12. Version 1.1.4

- Completed rewrite of driver. Eliminated SoundBlaster compatibility completely, and now uses the much-faster scatter-gather DMA engine.

Chapter 8. Internal Functions

via_chan_stop

Name

`via_chan_stop` — Terminate DMA on specified PCM channel

Synopsis

```
void via_chan_stop (long iobase);
```

Arguments

`iobase`

PCI base address for SGD channel registers

Description

Terminate scatter-gather DMA operation for given channel (derived from *`iobase`*), if DMA is active.

Note that *`iobase`* is not the PCI base address, but the PCI base address plus an offset to one of three PCM channels supported by the chip.

via_chan_status_clear

Name

`via_chan_status_clear` — Clear status flags on specified DMA channel

Synopsis

```
void via_chan_status_clear (long iobase);
```

Arguments

`iobase`

PCI base address for SGD channel registers

Description

Clear any pending status flags for the given DMA channel (derived from *`iobase`*), if any flags are asserted.

Note that *`iobase`* is not the PCI base address, but the PCI base address plus an offset to one of three PCM channels supported by the chip.

sg_begin

Name

`sg_begin` — Begin recording or playback on a PCM channel

Synopsis

```
void sg_begin (struct via_channel * chan);
```

Arguments

chan

Channel for which DMA operation shall begin

Description

Start scatter-gather DMA for the given channel.

via_syscall_down

Name

`via_syscall_down` — down the card-specific syscell semaphore

Synopsis

```
int via_syscall_down (struct via_info * card, int nonblock);
```

Arguments

`card`

Private info for specified board

`nonblock`

boolean, non-zero if O_NONBLOCK is set

Description

Encapsulates standard method of acquiring the syscall sem.

Returns negative errno on error, or zero for success.

via_stop_everything

Name

`via_stop_everything` — Stop all audio operations

Synopsis

```
void via_stop_everything (struct via_info * card);
```

Arguments

`card`

Private info for specified board

Description

Stops all DMA operations and interrupts, and clear any pending status bits resulting from those operations.

via_set_rate

Name

`via_set_rate` — Set PCM rate for given channel

Synopsis

```
int via_set_rate (struct ac97_codec * ac97, struct via_channel *
chan, unsigned rate);
```

Arguments

`ac97`

Pointer to generic codec info struct

`chan`

Private info for specified channel

`rate`

Desired PCM sample rate, in Khz

Description

Sets the PCM sample rate for a channel.

Values for *rate* are clamped to a range of 4000 Khz through 48000 Khz, due to hardware constraints.

via_chan_init_defaults

Name

via_chan_init_defaults — Initialize a struct via_channel

Synopsis

```
void via_chan_init_defaults (struct via_info * card, struct
via_channel * chan);
```

Arguments

card

Private audio chip info

chan

Channel to be initialized

Description

Zero *chan*, and then set all static defaults for the structure.

via_chan_init

Name

via_chan_init — Initialize PCM channel

Synopsis

```
void via_chan_init (struct via_info * card, struct via_channel *
chan);
```

Arguments

card

Private audio chip info

chan

Channel to be initialized

Description

Performs some of the preparations necessary to begin using a PCM channel.

Currently the preparations consist in setting the PCM channel to a known state.

via_chan_buffer_init

Name

`via_chan_buffer_init` — Initialize PCM channel buffer

Synopsis

```
int via_chan_buffer_init (struct via_info * card, struct
via_channel * chan);
```

Arguments

`card`

Private audio chip info

`chan`

Channel to be initialized

Description

Performs some of the preparations necessary to begin using a PCM channel.

Currently the preparations include allocating the scatter-gather DMA table and buffers, and passing the address of the DMA table to the hardware.

Note that special care is taken when passing the DMA table address to hardware, because it was found during driver development that the hardware did not always "take" the address.

via_chan_free

Name

`via_chan_free` — Release a PCM channel

Synopsis

```
void via_chan_free (struct via_info * card, struct via_channel *
chan);
```

Arguments

card

Private audio chip info

chan

Channel to be released

Description

Performs all the functions necessary to clean up an initialized channel.

Currently these functions include disabled any active DMA operations, setting the PCM channel back to a known state, and releasing any allocated sound buffers.

via_chan_pcm_fmt

Name

via_chan_pcm_fmt — Update PCM channel settings

Synopsis

```
void via_chan_pcm_fmt (struct via_channel * chan, int reset);
```

Arguments

chan

Channel to be updated

reset

Boolean. If non-zero, channel will be reset to 8-bit mono mode.

Description

Stores the settings of the current PCM format, 8-bit or 16-bit, and mono/stereo, into the hardware settings for the specified channel. If *reset* is non-zero, the channel is reset to 8-bit mono mode. Otherwise, the channel is set to the values stored in the channel information struct *chan*.

via_chan_clear

Name

via_chan_clear — Stop DMA channel operation, and reset pointers

Synopsis

```
void via_chan_clear (struct via_info * card, struct via_channel
* chan);
```

Arguments

card

the chip to accessed

chan

Channel to be cleared

Description

Call via_chan_stop to halt DMA operations, and then resets all software pointers which track DMA operation.

via_chan_set_speed

Name

via_chan_set_speed — Set PCM sample rate for given channel

Synopsis

```
int via_chan_set_speed (struct via_info * card, struct
```

```
via_channel * chan, int val);
```

Arguments

card

Private info for specified board

chan

Channel whose sample rate will be adjusted

val

New sample rate, in Khz

Description

Helper function for the SNDCTL_DSP_SPEED ioctl. OSS semantics demand that all audio operations halt (if they are not already halted) when the SNDCTL_DSP_SPEED is given.

This function halts all audio operations for the given channel *chan*, and then calls via_set_rate to set the audio hardware to the new rate.

via_chan_set_fmt

Name

`via_chan_set_fmt` — Set PCM sample size for given channel

Synopsis

```
int via_chan_set_fmt (struct via_info * card, struct via_channel
* chan, int val);
```

Arguments

`card`

Private info for specified board

`chan`

Channel whose sample size will be adjusted

`val`

New sample size, use the `AFMT_xxx` constants

Description

Helper function for the `SNDCTL_DSP_SETFMT` ioctl. OSS semantics demand that all audio operations halt (if they are not already halted) when the `SNDCTL_DSP_SETFMT` is

given.

This function halts all audio operations for the given channel *chan*, and then calls via_chan_pcm_fmt to set the audio hardware to the new sample size, either 8-bit or 16-bit.

via_chan_set_stereo

Name

via_chan_set_stereo — Enable or disable stereo for a DMA channel

Synopsis

```
int via_chan_set_stereo (struct via_info * card, struct
via_channel * chan, int val);
```

Arguments

card

Private info for specified board

chan

Channel whose stereo setting will be adjusted

val

New sample size, use the AFMT_xxx constants

Description

Helper function for the SNDCTL_DSP_CHANNELS and SNDCTL_DSP_STEREO ioctls. OSS semantics demand that all audio operations halt (if they are not already halted) when SNDCTL_DSP_CHANNELS or SNDCTL_DSP_STEREO is given.

This function halts all audio operations for the given channel *chan*, and then calls via_chan_pcm_fmt to set the audio hardware to enable or disable stereo.

via_chan_dump_bufs

Name

via_chan_dump_bufs — Display DMA table contents

Synopsis

```
void via_chan_dump_bufs (struct via_channel * chan);
```

Arguments

chan

Channel whose DMA table will be displayed

Description

Debugging function which displays the contents of the scatter-gather DMA table for the given channel *chan*.

via_chan_flush_frag

Name

via_chan_flush_frag — Flush partially-full playback buffer to hardware

Synopsis

```
void via_chan_flush_frag (struct via_channel * chan);
```

Arguments

chan

Channel whose DMA table will be displayed

Description

Flushes partially-full playback buffer to hardware.

via_chan_maybe_start

Name

`via_chan_maybe_start` — Initiate audio hardware DMA operation

Synopsis

```
void via_chan_maybe_start (struct via_channel * chan);
```

Arguments

chan

Channel whose DMA is to be started

Description

Initiate DMA operation, if the DMA engine for the given channel *chan* is not already active.

via_ac97_wait_idle

Name

via_ac97_wait_idle — Wait until AC97 codec is not busy

Synopsis

```
u8 via_ac97_wait_idle (struct via_info * card);
```

Arguments

card

Private info for specified board

Description

Sleep until the AC97 codec is no longer busy. Returns the final value read from the SGD register being polled.

via_ac97_read_reg

Name

via_ac97_read_reg — Read AC97 standard register

Synopsis

```
u16 via_ac97_read_reg (struct ac97_codec * codec, u8 reg);
```

Arguments

codec

Pointer to generic AC97 codec info

reg

Index of AC97 register to be read

Description

Read the value of a single AC97 codec register, as defined by the Intel AC97 specification.

Defines the standard AC97 read-register operation required by the kernel's ac97_codec interface.

Returns the 16-bit value stored in the specified register.

via_ac97_write_reg

Name

via_ac97_write_reg — Write AC97 standard register

Synopsis

```
void via_ac97_write_reg (struct ac97_codec * codec, u8 reg, u16
value);
```

Arguments

codec

Pointer to generic AC97 codec info

reg

Index of AC97 register to be written

value

Value to be written to AC97 register

Description

Write the value of a single AC97 codec register, as defined by the Intel AC97 specification.

Defines the standard AC97 write-register operation required by the kernel’s ac97_codec interface.

via_intr_channel

Name

`via_intr_channel` — handle an interrupt for a single channel

Synopsis

```
void via_intr_channel (struct via_channel * chan);
```

Arguments

chan

handle interrupt for this channel

Description

This is the “meat” of the interrupt handler, containing the actions taken each time an interrupt occurs. All communication and coordination with userspace takes place here.

Locking

inside card->lock

via_interrupt_init

Name

`via_interrupt_init` — Initialize interrupt handling

Synopsis

```
int via_interrupt_init (struct via_info * card);
```

Arguments

`card`

Private info for specified board

Description

Obtain and reserve IRQ for using in handling audio events. Also, disable any IRQ-generating resources, to make sure we don’t get interrupts before we want them.

via_dsp_drain_playback

Name

`via_dsp_drain_playback` — sleep until all playback samples are flushed

Synopsis

```
int via_dsp_drain_playback (struct via_info * card, struct
via_channel * chan, int nonblock);
```

Arguments

`card`

Private info for specified board

`chan`

Channel to drain

`nonblock`

boolean, non-zero if O_NONBLOCK is set

Description

Sleeps until all playback has been flushed to the audio hardware.

Locking

inside card->syscall_sem

via_dsp_ioctl_space

Name

`via_dsp_ioctl_space` — get information about channel buffering

Synopsis

```
int via_dsp_ioctl_space (struct via_info * card, struct
via_channel * chan, void * arg);
```

Arguments

`card`

Private info for specified board

`chan`

pointer to channel-specific info

arg

user buffer for returned information

Description

Handles SNDCTL_DSP_GETISPACE and SNDCTL_DSP_GETOSPACE.

Locking

inside card->syscall_sem

via_dsp_ioctl_ptr

Name

via_dsp_ioctl_ptr — get information about hardware buffer ptr

Synopsis

```
int via_dsp_ioctl_ptr (struct via_info * card, struct
via_channel * chan, void * arg);
```

Arguments

card

Private info for specified board

chan

pointer to channel-specific info

arg

user buffer for returned information

Description

Handles SNDCTL_DSP_GETIPTR and SNDCTL_DSP_GETOPTR.

Locking

inside card->syscall_sem