NAME

snd_hda - Intel High Definition Audio bridge device driver

SYNOPSIS

To compile this driver into the kernel, place the following lines in your kernel configuration file:

```
device sound
device snd_hda
```

Alternatively, to load the driver as a module at boot time, place the following line in loader.conf(5):

```
snd_hda_load="YES"
```

DESCRIPTION

The High Definition (HD) Audio specification was developed by Intel as the logical successor of the old AC’97 specification and has several advantages, such as higher bandwidth which allows more channels and more detailed formats, support for several logical audio devices, and general purpose DMA channels.

The **snd_hda** driver includes HDA bus controller driver (hdac), HDA codec driver (hdacc) and HDA codecs audio functions bridge driver (hdaa) that allows the generic audio driver, sound(4), to be used with this hardware. Only audio functions are supported by **snd_hda**. Modem and other possible functions are not implemented.

The **snd_hda** driver supports hardware that conforms with revision 1.0 of the Intel High Definition Audio specification and tries to behave much like the Microsoft Universal Audio Architecture (UAA) draft (revision 0.7b) for handling audio devices.

According to HDA and UAA specifications, depending on the number of HDA buses and codecs present in system, their audio capabilities and BIOS provided configuration, the **snd_hda** driver often provides several PCM audio devices. For example, one device for main rear 7.1 output and inputs, one device for independent headset connectors at front and one device for SPDIF or HDMI audio input/output. The assignment of audio inputs and outputs may be tuned with device.hints(5) or sysctl(8). The driver’s verbose boot messages provide a lot of information about the operation of the driver and present audio setup.

The default audio device may be tuned by setting the *hw.snd.default_unit* sysctl, as described in sound(4), or explicitly specified in application settings.

Boot-time Configuration

The following variables are available at boot-time through the device.hints(5) file:

hint.hdac.%d.config Configures a range of possible controller options. Possible values are: "64bit", "dmapos", "msi". An option prefixed with "no", such as "nomsi", will do the opposite and takes precedence. Options can be separated by whitespace and commas.

hint.hdac.%d.msi Controls MSI (Message Signaled Interrupts) support.

hint.hdac.%d.cad%d.nid%d.config
Same as *hint.hdaa.%d.nid%d.config*

hint.hdaa.%d.config Configures a range of possible audio function options. Possible values are: "eapdinv", "ivref", "ivref50", "ivref80", "ivref100", "fixedrate", "forcestereo", "ovref", "ovref50", "ovref80", "ovref100", "senseinv", "softpcmvol", and "vref". An option prefixed with "no", such as "nofixedrate", will do the opposite and takes precedence. Options can be separated by whitespace and commas.

The "eapdinv" option inverts External Amplifier Power Down signal. The "fixedrate" denies all sampling rates except 48KHz. The "forcestereo" denies mono playback/recording. The "senseinv" option inverts jack sensing logic. The "ivref*X*" and "ovref*X*" options control the voltage used to power external microphones.

dev.hdaa.%d.init_clear
Zero out the pin widget config setup by the system. Some systems seem to have unusable audio devices if the pin widget configuration is cleared. Set this value to 0 to accept the default configuration values setup by the BIOS.

hint.hdaa.%d.gpio_config
Overrides audio function GPIO pins configuration set by BIOS. May be specified as a set of space-separated "*num*=*value*" pairs, where *num* is GPIO line number, and *value* is one of: "keep", "set", "clear", "disable" and "input".

"GPIOs" are a codec’s General Purpose I/O pins which system integrators sometimes use to control external muters, amplifiers and so on. If you have no sound, or sound volume is not adequate, you may have to experiment a bit with the GPIO setup to find the optimal setup for your system.

hint.hdaa.%d.nid%d.config

Overrides audio function pin configuration set by BIOS. May be specified as a 32-bit hexadecimal value with a leading "0x", or as a set of space-separated "*option=value*" pairs.

hint.pcm.%d.rec.autosrc

Controls automatic recording source feature:
0 disabled,
1 once on attach,
2 enabled.
When enabled, driver will automatically set recording source of the mixer to connected input using jack presence detection statuses.

Pin configuration is the UAA driver's main source of information about codec usage. This information is usually provided by the codec manufacturer and tuned by system integrators for specific system requirements. The **snd_hda** driver allows users to override it to fix integrator mistakes or to use the available codec in alternative ways (for example to get stereo output and 2 inputs instead of a single 5.1 output).

The following options are supported:

as Association number. Associations are used to group individual pins to form a complex multi-pin device. For example, to group 4 connectors for 7.1 input/output, or to treat several input connectors as sources for the same input device. Association numbers can be specified as numeric values from 0 to 15. A value of 0 means disabled pin. A value of 15 is a set of independent unassociated pins. Each association includes only pins of the same direction (in/out) and is detected atomically (all pins or none). A separate PCM audio device is created for every pair of input and output associations.

seq Sequence number. A unique, per-association number used to order pins inside the particular association. Sequence numbers can be specified as numeric values from 0 to 15.

The sequence number 15 has a special meaning for output associations. Output pins with this number and device type "*Headphones*" will duplicate (with automatic mute if jack detection is supported) the first pin in that association.

The sequence numbers 14 and 15 has a special meaning for input associations. Their presence in association defines it as multiplexed or mixed respectively. If none of them are present and there are more than one pin in association, the association will provide

multichannel input.

For multichannel input/output associations sequence numbers encode channel pairs positions: 0 - Front, 1 - Center/LFE, 2 - Back, 3 - Front Wide Center, 4 - Side. Standard combinations are: (0) - Stereo; (0, 2), (0, 4) - Quadro; (0, 1, 2), (0, 1, 4) - 5.1; (0, 1, 2, 4) - 7.1.

device Device type. Can be specified as a number from 0 to 15 or as a name: "Line-out", "Speaker", "Headphones," "CD", "SPDIF-out", "Digital-out", "Modem-line", "Modem-handset", "Line-in", "AUX", "Mic", "Telephony", "SPDIF-in", "Digital-in", "Res.E", or "Other". The device type also describes the pin direction (in/out). For example, "CD" always means an input pin, while "Headphones" always means an output.

conn Connection type. Can be specified as a number from 0 to 3. The connection type can also be specified as one of the special names "Jack", "None", "Fixed", or "Both". Pins with a connection type of "None" are disabled.

ctype Connector physical type. Can be specified as a number from 0 to 15. This is a reference only value. It is ignored by the **snd_hda** driver.

color Connector color. Can be specified as a number from 0 to 15 or as one of the names "Unknown", "Black", "Grey", "Blue", "Green", "Red", "Orange", "Yellow", "Purple", "Pink", "Res.A", "Res.B", "Res.C", "Res.D", "White", or "Other". This is a reference only value. It is ignored by the **snd_hda** driver.

loc Connector physical location. Can be specified as a number from 0 to 63. This is a reference only value. It is ignored by the **snd_hda** driver.

misc Misc bits. Can be specified as a number from 0 to 15. Bit 0 has a special meaning. When set it means that jack detection is not implemented in hardware.

Runtime Configuration

The following sysctl(8) variables are available in addition to those available to all sound(4) devices:

dev.hdac.%d.pindump Setting this to a non-zero value dumps the current pin configuration, main capabilities and jack sense status of all audio functions on the controller to console and syslog.

dev.hdac.%d.polling Enables polling mode. In this mode the driver operates by querying the device state on timer ticks using callout(9) instead of interrupts.

Polling is disabled by default. Do not enable it unless you are facing weird interrupt problems or if the device cannot generate interrupts at all.

dev.hdaa.%d.config Run-time equivalent of the *hint.hdaa.%d.config* tunable.

dev.hdaa.%d.gpi_state Current state of GPI lines.

dev.hdaa.%d.gpio_state Current state of GPIO lines.

dev.hdaa.%d.gpio_config Run-time equivalent of the *hint.hdaa.%d.gpio.config* tunable.

dev.hdaa.%d.gpo_state Current state of GPO lines.

dev.hdaa.%d.nid%d_config Run-time equivalent of the *hint.hdaa.%d.nid%d.config* tunable.

dev.hdaa.%d.nid%d_original Original pin configuration written by BIOS.

dev.hdaa.%d.reconfig Setting this to a non-zero value makes driver to destroy existing pcm devices and process new pins configuration set via *dev.hdaa.%d.nid%d_config*.

dev.pcm.%d.play.32bit, dev.pcm.%d.rec.32bit
HDA controller uses 32bit representation for all samples of more then 16 bits. These variables allow to specify how many bits of these 32 should be used by CODEC. Depending on codec capabilities, possible values are 20, 24 and 32 bit. The default value is 24.

dev.pcm.%d.rec.autosrc Run-time equivalent of the *hint.pcm.%d.rec.autosrc* tunable.

EXAMPLES

Taking HP Compaq DX2300 with Realtek ALC888 HDA codec for example. This system has two audio connectors on a front side, three audio connectors on a rear side and one internal speaker. According to verbose driver output and the codec datasheet, this codec has five stereo DACs and two stereo ADCs, all of them are routable to any codec pin (external connector). All codec pins are reversible (could be configured either as input or output).

So high codec uniformity and flexibility allow driver to configure it in many different ways, depending on requested pins usage described by pins configuration. The driver reports such default pin

configuration when verbose messages enabled:

```
hdaa0: nid   0x    as seq device       conn  jack    loc        color   misc
hdaa0: 20 01014020 2  0  Line-out      Jack  1/8     Rear       Green   0
hdaa0: 21 99130110 1  0  Speaker       Fixed ATAPI   Onboard    Unknown 1
hdaa0: 22 411111f0 15 0  Speaker       None  1/8     Rear       Black   1 DISA
hdaa0: 23 411111f0 15 0  Speaker       None  1/8     Rear       Black   1 DISA
hdaa0: 24 01a19830 3  0  Mic           Jack  1/8     Rear       Pink    8
hdaa0: 25 02a1983f 3  15 Mic           Jack  1/8     Front      Pink    8
hdaa0: 26 01813031 3  1  Line-in       Jack  1/8     Rear       Blue    0
hdaa0: 27 0221401f 1  15 Headphones    Jack  1/8     Front      Green   0
hdaa0: 28 411111f0 15 0  Speaker       None  1/8     Rear       Black   1 DISA
hdaa0: 30 411111f0 15 0  Speaker       None  1/8     Rear       Black   1 DISA
hdaa0: 31 411111f0 15 0  Speaker       None  1/8     Rear       Black   1 DISA
```

Here we can see, that the nodes with ID (nid) 25 and 27 are front panel connectors (Jack, Front), nids 20, 24 and 26 are rear panel connectors (Jack, Rear) and nid 21 is a built-in speaker (Fixed, Onboard). Pins with nids 22, 23, 28, 30 and 31 will be disabled by driver due to "None" connectivity. So the pin count and description matches to connectors that we have.

Using association (as) and sequence (seq) fields values pins are grouped into 3 associations:

```
hdaa0: Association 0 (1) out:
hdaa0:   Pin nid=21 seq=0
hdaa0:   Pin nid=27 seq=15
hdaa0: Association 1 (2) out:
hdaa0:   Pin nid=20 seq=0
hdaa0: Association 2 (3) in:
hdaa0:   Pin nid=24 seq=0
hdaa0:   Pin nid=26 seq=1
hdaa0:   Pin nid=25 seq=15
```

Each pcm(4) device uses two associations: one for playback and one for recording. Associations processed and assigned to pcm(4) devices in increasing numerical order. In this case association #0 (1) will become pcm0 device playback, using the internal speakers and *Headphones* jack with speaker automute on the headphones jack connection. Association #1 (2) will become pcm1 playback, using the *Line-out* jack. Association #2 (3) will become pcm0 recording, using the external microphones and the *Line-in* jack.

The **snd_hda** driver provides extensive verbose messages to diagnose its operation logic and describe its

current codec configuration.

Using device.hints(5) it is possible to modify the configuration of the existing pins, allowing a broad range of different audio setups. Here are a few examples of some setups possible for this particular hardware:

Example 1

Setting the device.hints(5) options

```
hint.hdac.0.cad0.nid20.config="as=1"
hint.hdac.0.cad0.nid21.config="as=2"
```

will swap line-out and speaker functions. So the pcm0 device will play to the line-out and headphones jacks. Line-out will be muted on the headphones jack connection. Recording on pcm0 will go from two external microphones and line-in jacks. pcm1 playback will go to the internal speaker.

Example 2

Setting the device.hints(5) options

```
hint.hdac.0.cad0.nid20.config="as=1 seq=15 device=Headphones"
hint.hdac.0.cad0.nid27.config="as=2 seq=0"
hint.hdac.0.cad0.nid25.config="as=4 seq=0"
```

will split the headphones and one of the microphones to a separate device. The pcm0 device will play to the internal speaker and to the line-out jack, with speaker automute on the line-out jack connection. Recording on pcm0 will use input from one external microphone and the line-in jacks. The pcm1 device will be completely dedicated to a headset (headphones and mic) connected to the front connectors.

Example 3

Setting the device.hints(5) options

```
hint.hdac.0.cad0.nid20.config="as=1 seq=0"
hint.hdac.0.cad0.nid26.config="as=2 seq=0"
hint.hdac.0.cad0.nid27.config="as=3 seq=0"
hint.hdac.0.cad0.nid25.config="as=4 seq=0"
hint.hdac.0.cad0.nid24.config="as=5 seq=0 device=Line-out"
hint.hdac.0.cad0.nid21.config="as=6 seq=0"
```

will give 4 independent devices: pcm0 (line-out and line-in), pcm1 (headphones and mic), pcm2 (additional line-out via retasked rear mic jack), and pcm3 (internal speaker).

Example 4

Setting the device.hints(5) options

```
hint.hdac.0.cad0.nid20.config="as=1 seq=0"
hint.hdac.0.cad0.nid24.config="as=1 seq=1 device=Line-out"
hint.hdac.0.cad0.nid26.config="as=1 seq=2 device=Line-out"
hint.hdac.0.cad0.nid21.config="as=2 seq=0"
```

will give 2 devices: pcm0 for 5.1 playback via 3 rear connectors (line-out and retasked mic and line-in) and headset (headphones and mic) at front connectors. pcm1 for internal speaker playback. On headphones connection rear connectors will be muted.

MIXER CONTROLS

Depending on codec configuration, these controls and signal sources could be reported to sound(4):

vol overall output level (volume)

rec overall recording level

igain input-to-output monitoring loopback level

ogain external amplifier control

pcm PCM playback

mix input mix

mic first external or second internal microphone input

monitor
first internal or second external microphone input

line, line1, line2, line3
analog (line) inputs

dig1, dig2, dig3
digital (S/PDIF, HDMI or DisplayPort) inputs

cd CD input

speaker PC speaker input

phin, phout, radio, video
other random inputs

Controls have different precision. Some could be just an on/off triggers. Most of controls use logarithmic scale.

HARDWARE

The **snd_hda** driver supports controllers having PCI class 4 (multimedia) and subclass 3 (HDA), compatible with Intel HDA specification.

The **snd_hda** driver supports more than two hundred different controllers and CODECs. There is no sense to list all of them here, as in most cases specific CODEC configuration and wiring are more important then type of the CODEC itself.

SEE ALSO

snd_ich(4), sound(4), device.hints(5), loader.conf(5), sysctl(8)

HISTORY

The **snd_hda** device driver first appeared in FreeBSD 6.3.

AUTHORS

The **snd_hda** driver was written by Stephane E. Potvin <*sepotvin@videotron.ca*>, Ariff Abdullah <*ariff@FreeBSD.org*> and Alexander Motin <*mav@FreeBSD.org*>. This manual page was written by Joel Dahl <*joel@FreeBSD.org*>, Alexander Motin <*mav@FreeBSD.org*> and Giorgos Keramidas <*keramida@FreeBSD.org*>.

BUGS

Some Hardware/OEM vendors tend to screw up BIOS settings or use custom unusual CODEC wiring that create problems to the driver. This may result in missing pcm devices, or a state where the **snd_hda** driver seems to attach and work, but no sound is played. Some cases can be solved by tuning *loader.conf* variables. But before trying to fix problem that way, make sure that there really is a problem and that the PCM audio device in use really corresponds to the expected audio connector.

Some vendors use non-standardized General Purpose I/O (GPIO) pins of the codec to control external amplifiers. In some cases setting a combination of GPIO bits may be needed to make sound work on a specific device.

HDMI and DisplayPort audio may also require support from video driver.