```
<title>DVB Audio Device</title>
<para>The DVB audio device controls the MPEG2 audio decoder of the DVB hardware.
It
can be accessed through <emphasis role="tt">/dev/dvb/adapter0/audio0</emphasis>.
Data types and and
ioctl definitions can be accessed by including <emphasis
role="tt">linux/dvb/video.h</emphasis> in your
application.
</para>
<para>Please note that some DVB cards don&#8217;t have their own MPEG decoder,
which results in
the omission of the audio and video device.
</para>

<section id="audio_data_types">
<title>Audio Data Types</title>
<para>This section describes the structures, data types and defines used when
talking to the
audio device.
</para>

<section id="audio_stream_source_t">
<title>audio_stream_source_t</title>
<para>The audio stream source is set through the AUDIO_SELECT_SOURCE call and
can take
the following values, depending on whether we are replaying from an internal
(demux) or
external (user write) source.
</para>
<programlisting>
 typedef enum {
         AUDIO_SOURCE_DEMUX,
         AUDIO_SOURCE_MEMORY
 } audio_stream_source_t;
</programlisting>
<para>AUDIO_SOURCE_DEMUX selects the demultiplexer (fed either by the frontend
or the
DVR device) as the source of the video stream. If AUDIO_SOURCE_MEMORY
is selected the stream comes from the application through the <emphasis
role="tt">write()</emphasis> system
call.
</para>

</section>
<section id="audio_play_state_t">
<title>audio_play_state_t</title>
<para>The following values can be returned by the AUDIO_GET_STATUS call
representing the
state of audio playback.
</para>
<programlisting>
 typedef enum {
         AUDIO_STOPPED,
         AUDIO_PLAYING,
         AUDIO_PAUSED
 } audio_play_state_t;
```

```
</programlisting>

</section>
<section id="audio_channel_select_t">
<title>audio_channel_select_t</title>
<para>The audio channel selected via AUDIO_CHANNEL_SELECT is determined by the
following values.
</para>
<programlisting>
 typedef enum {
         AUDIO_STEREO,
         AUDIO_MONO_LEFT,
         AUDIO_MONO_RIGHT,
 } audio_channel_select_t;
</programlisting>

</section>
<section id="struct_audio_status">
<title>struct audio_status</title>
<para>The AUDIO_GET_STATUS call returns the following structure informing about
various
states of the playback operation.
</para>
<programlisting>
 typedef struct audio_status {
         boolean AV_sync_state;
         boolean mute_state;
         audio_play_state_t play_state;
         audio_stream_source_t stream_source;
         audio_channel_select_t channel_select;
         boolean bypass_mode;
 } audio_status_t;
</programlisting>

</section>
<section id="struct_audio_mixer">
<title>struct audio_mixer</title>
<para>The following structure is used by the AUDIO_SET_MIXER call to set the
audio
volume.
</para>
<programlisting>
 typedef struct audio_mixer {
         unsigned int volume_left;
         unsigned int volume_right;
 } audio_mixer_t;
</programlisting>

</section>
<section id="audio_encodings">
<title>audio encodings</title>
<para>A call to AUDIO_GET_CAPABILITIES returns an unsigned integer with the
following
bits set according to the hardwares capabilities.
</para>
<programlisting>
```

```
 #define AUDIO_CAP_DTS    1
 #define AUDIO_CAP_LPCM   2
 #define AUDIO_CAP_MP1    4
 #define AUDIO_CAP_MP2    8
 #define AUDIO_CAP_MP3   16
 #define AUDIO_CAP_AAC   32
 #define AUDIO_CAP_OGG   64
 #define AUDIO_CAP_SDDS 128
 #define AUDIO_CAP_AC3  256
</programlisting>

</section>
<section id="struct_audio_karaoke">
<title>struct audio_karaoke</title>
<para>The ioctl AUDIO_SET_KARAOKE uses the following format:
</para>
<programlisting>
 typedef
 struct audio_karaoke{
         int vocal1;
         int vocal2;
         int melody;
 } audio_karaoke_t;
</programlisting>
<para>If Vocal1 or Vocal2 are non-zero, they get mixed into left and right t at
70% each. If both,
Vocal1 and Vocal2 are non-zero, Vocal1 gets mixed into the left channel and
Vocal2 into the
right channel at 100% each. Ff Melody is non-zero, the melody channel gets mixed
into left
and right.
</para>

</section>
<section id="audio_attributes">
<title>audio attributes</title>
<para>The following attributes can be set by a call to AUDIO_SET_ATTRIBUTES:
</para>
<programlisting>
 typedef uint16_t audio_attributes_t;
 /&#x22C6;   bits: descr. &#x22C6;/
 /&#x22C6;   15-13 audio coding mode (0=ac3, 2=mpeg1, 3=mpeg2ext, 4=LPCM, 6=DTS,
&#x22C6;/
 /&#x22C6;   12    multichannel extension &#x22C6;/
 /&#x22C6;   11-10 audio type (0=not spec, 1=language included) &#x22C6;/
 /&#x22C6;    9- 8 audio application mode (0=not spec, 1=karaoke, 2=surround)
&#x22C6;/
 /&#x22C6;    7- 6 Quantization / DRC (mpeg audio: 1=DRC exists)(lpcm: 0=16bit,
&#x22C6;/
 /&#x22C6;    5- 4 Sample frequency fs (0=48kHz, 1=96kHz) &#x22C6;/
 /&#x22C6;    2- 0 number of audio channels (n+1 channels) &#x22C6;/
</programlisting>
 </section></section>
<section id="audio_function_calls">
<title>Audio Function Calls</title>
```

```
<section id="audio_fopen">
<title>open()</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This system call opens a named audio device (e.g.
/dev/dvb/adapter0/audio0)
 for subsequent use. When an open() call has succeeded, the device will be ready
 for use. The significance of blocking or non-blocking mode is described in the
 documentation for functions where there is a difference. It does not affect the
 semantics of the open() call itself. A device opened in blocking mode can later
 be put into non-blocking mode (and vice versa) using the F_SETFL command
 of the fcntl system call. This is a standard system call, documented in the
Linux
 manual page for fcntl. Only one user can open the Audio Device in O_RDWR
 mode. All other attempts to open the device in this mode will fail, and an
error
 code will be returned. If the Audio Device is opened in O_RDONLY mode, the
 only ioctl call that can be used is AUDIO_GET_STATUS. All other call will
 return with an error code.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int open(const char &#x22C6;deviceName, int flags);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>const char
 *deviceName</para>
</entry><entry
 align="char">
<para>Name of specific audio device.</para>
</entry>
 </row><row><entry
 align="char">
<para>int flags</para>
</entry><entry
 align="char">
<para>A bit-wise OR of the following flags:</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>O_RDONLY read-only access</para>
</entry>
 </row><row><entry
 align="char">
```

```
</entry><entry
 align="char">
<para>O_RDWR read/write access</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>O_NONBLOCK open in non-blocking mode</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>(blocking mode is the default)</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>ENODEV</para>
</entry><entry
 align="char">
<para>Device driver not loaded/available.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EBUSY</para>
</entry><entry
 align="char">
<para>Device or resource busy.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Invalid argument.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section>
<section id="audio_fclose">
<title>close()</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
```

```
<para>This system call closes a previously opened audio device.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int close(int fd);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section>
<section id="audio_fwrite">
<title>write()</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This system call can only be used if AUDIO_SOURCE_MEMORY is selected
 in the ioctl call AUDIO_SELECT_SOURCE. The data provided shall be in
 PES format. If O_NONBLOCK is not specified the function will block until
 buffer space is available. The amount of data to be transferred is implied by
 count.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>size_t write(int fd, const void &#x22C6;buf, size_t count);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
```

```
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>void *buf</para>
</entry><entry
 align="char">
<para>Pointer to the buffer containing the PES data.</para>
</entry>
 </row><row><entry
 align="char">
<para>size_t count</para>
</entry><entry
 align="char">
<para>Size of buf.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EPERM</para>
</entry><entry
 align="char">
<para>Mode AUDIO_SOURCE_MEMORY not selected.</para>
</entry>
 </row><row><entry
 align="char">
<para>ENOMEM</para>
</entry><entry
 align="char">
<para>Attempted to write more data than the internal buffer can
 hold.</para>
</entry>
 </row><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_STOP</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to stop playing the current
stream.</para>
</entry>
 </row></tbody></tgroup></informaltable>
```

```
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_STOP);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_STOP for this command.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_PLAY</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to start playing an audio stream
from the
 selected source.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
```

```
 align="char">
<para>int ioctl(int fd, int request = AUDIO_PLAY);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_PLAY for this command.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_PAUSE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call suspends the audio stream being played. Decoding and
playing
 are paused. It is then possible to restart again decoding and playing process
of
 the audio stream using AUDIO_CONTINUE command.</para>
</entry>
 </row><row><entry
 align="char">
<para>If AUDIO_SOURCE_MEMORY is selected in the ioctl call
 AUDIO_SELECT_SOURCE, the DVB-subsystem will not decode (consume)
 any more data until the ioctl call AUDIO_CONTINUE or AUDIO_PLAY is
 performed.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_PAUSE);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_PAUSE for this command.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SELECT_SOURCE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call informs the audio device which source shall be used
 for the input data. The possible sources are demux or memory. If
 AUDIO_SOURCE_MEMORY is selected, the data is fed to the Audio Device
 through the write command.</para>
</entry>
```

```
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_SELECT_SOURCE,
 audio_stream_source_t source);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SELECT_SOURCE for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>audio_stream_source_t
 source</para>
</entry><entry
 align="char">
<para>Indicates the source that shall be used for the Audio
 stream.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
```

```
<para>Illegal input parameter.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_MUTE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the audio device to mute the stream that is currently
being
 played.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_SET_MUTE,
 boolean state);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_MUTE for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>boolean state</para>
</entry><entry
 align="char">
<para>Indicates if audio device shall mute or not.</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>TRUE Audio Mute</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
```

```
<para>FALSE Audio Un-mute</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Illegal input parameter.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_AV_SYNC</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to turn ON or OFF A/V
synchronization.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_SET_AV_SYNC,
 boolean state);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_AV_SYNC for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>boolean state</para>
</entry><entry
 align="char">
<para>Tells the DVB subsystem if A/V synchronization shall be
 ON or OFF.</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>TRUE AV-sync ON</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>FALSE AV-sync OFF</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Illegal input parameter.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_BYPASS_MODE</title>
<para>DESCRIPTION
</para>
```

```
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to bypass the Audio decoder and
forward
 the stream without decoding. This mode shall be used if streams that
can&#8217;t be
 handled by the DVB system shall be decoded. Dolby DigitalTM streams are
 automatically forwarded by the DVB subsystem if the hardware can handle
it.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request =
 AUDIO_SET_BYPASS_MODE, boolean mode);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_BYPASS_MODE for this
 command.</para>
</entry>
 </row><row><entry
 align="char">
<para>boolean mode</para>
</entry><entry
 align="char">
<para>Enables or disables the decoding of the current Audio
 stream in the DVB subsystem.</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>TRUE Bypass is disabled</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>FALSE Bypass is enabled</para>
</entry>
```

```
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Illegal input parameter.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_CHANNEL_SELECT</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to select the requested channel if
possible.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request =
 AUDIO_CHANNEL_SELECT, audio_channel_select_t);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
```

```xml
</entry><entry
 align="char">
<para>Equals AUDIO_CHANNEL_SELECT for this
 command.</para>
</entry>
 </row><row><entry
 align="char">
<para>audio_channel_select_t
 ch</para>
</entry><entry
 align="char">
<para>Select the output format of the audio (mono left/right,
 stereo).</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Illegal input parameter ch.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_GET_STATUS</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to return the current state of the
Audio
 Device.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_GET_STATUS,
 struct audio_status &#x22C6;status);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_GET_STATUS for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>struct audio_status
 *status</para>
</entry><entry
 align="char">
<para>Returns the current state of Audio Device.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EFAULT</para>
</entry><entry
 align="char">
<para>status points to invalid address.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_GET_CAPABILITIES</title>
<para>DESCRIPTION
```

```
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to tell us about the decoding
capabilities
 of the audio hardware.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request =
 AUDIO_GET_CAPABILITIES, unsigned int &#x22C6;cap);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_GET_CAPABILITIES for this
 command.</para>
</entry>
 </row><row><entry
 align="char">
<para>unsigned int *cap</para>
</entry><entry
 align="char">
<para>Returns a bit array of supported sound formats.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
```

```
</entry>
 </row><row><entry
 align="char">
<para>EFAULT</para>
</entry><entry
 align="char">
<para>cap points to an invalid address.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_CLEAR_BUFFER</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Audio Device to clear all software and hardware
buffers
 of the audio decoder device.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_CLEAR_BUFFER);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_CLEAR_BUFFER for this command.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
```

```
<para>EINTERNAL</para>
</entry><entry>
 align="char">
<para>Internal error.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_ID</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl selects which sub-stream is to be decoded if a program or
system
 stream is sent to the video device. If no audio stream type is set the id has
to be
 in [0xC0,0xDF] for MPEG sound, in [0x80,0x87] for AC3 and in [0xA0,0xA7]
 for LPCM. More specifications may follow for other stream types. If the stream
 type is set the id just specifies the substream id of the audio stream and only
 the first 5 bits are recognized.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_SET_ID, int
 id);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_ID for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>int id</para>
</entry><entry
 align="char">
<para>audio sub-stream id</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
```

```
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Invalid sub-stream id.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_MIXER</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl lets you adjust the mixer settings of the audio decoder.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(int fd, int request = AUDIO_SET_MIXER,
 audio_mixer_t &#x22C6;mix);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_ID for this command.</para>
```

```
</entry>
 </row><row><entry
 align="char">
<para>audio_mixer_t *mix</para>
</entry><entry
 align="char">
<para>mixer settings.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINTERNAL</para>
</entry><entry
 align="char">
<para>Internal error.</para>
</entry>
 </row><row><entry
 align="char">
<para>EFAULT</para>
</entry><entry
 align="char">
<para>mix points to an invalid address.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_STREAMTYPE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl tells the driver which kind of audio stream to expect. This is 
useful
 if the stream offers several audio sub-streams like LPCM and AC3.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = AUDIO_SET_STREAMTYPE,
 int type);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
```

```
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_STREAMTYPE for this
 command.</para>
</entry>
 </row><row><entry
 align="char">
<para>int type</para>
</entry><entry
 align="char">
<para>stream type</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>type is not a valid or supported stream type.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_EXT_ID</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl can be used to set the extension id for MPEG streams in DVD
 playback. Only the first 3 bits are recognized.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = AUDIO_SET_EXT_ID, int
 id);</para>
```

```
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_EXT_ID for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>int id</para>
</entry><entry
 align="char">
<para>audio sub_stream_id</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>id is not a valid id.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_ATTRIBUTES</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl is intended for DVD playback and allows you to set certain
 information about the audio stream.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
```

```
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = AUDIO_SET_ATTRIBUTES,
 audio_attributes_t attr );</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_ATTRIBUTES for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>audio_attributes_t
 attr</para>
</entry><entry
 align="char">
<para>audio attributes according to section ??</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>attr is not a valid or supported attribute setting.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>AUDIO_SET_KARAOKE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl allows one to set the mixer settings for a karaoke DVD.</para>
```

```
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = AUDIO_SET_STREAMTYPE,
 audio_karaoke_t &#x22C6;karaoke);</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>PARAMETERS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>int fd</para>
</entry><entry
 align="char">
<para>File descriptor returned by a previous call to open().</para>
</entry>
 </row><row><entry
 align="char">
<para>int request</para>
</entry><entry
 align="char">
<para>Equals AUDIO_SET_STREAMTYPE for this
 command.</para>
</entry>
 </row><row><entry
 align="char">
<para>audio_karaoke_t
 *karaoke</para>
</entry><entry
 align="char">
<para>karaoke settings according to section ??.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>ERRORS
</para>
<informaltable><tgroup cols="2"><tbody><row><entry
 align="char">
<para>EBADF</para>
</entry><entry
 align="char">
<para>fd is not a valid open file descriptor</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>karaoke is not a valid or supported karaoke setting.</para>
</entry>
 </row></tbody></tgroup></informaltable>
 </section>
</section>
```