```
<title>DVB Video Device</title>
<para>The DVB video device controls the MPEG2 video decoder of the DVB hardware.
It
can be accessed through <emphasis role="tt">/dev/dvb/adapter0/video0</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>Note that the DVB video device only controls decoding of the MPEG video
stream, not
its presentation on the TV or computer screen. On PCs this is typically handled
by an
associated video4linux device, e.g. <emphasis role="tt">/dev/video</emphasis>,
which allows scaling and defining output
windows.
</para>
<para>Some DVB cards don&amp;#8217;t have their own MPEG decoder, which results in
the omission of
the audio and video device as well as the video4linux device.
</para>
<para>The ioctls that deal with SPUs (sub picture units) and navigation packets
are only
supported on some MPEG decoders made for DVD playback.
</para>
<section id="video_types">
<title>Video Data Types</title>

<section id="video_format_t">
<title>video_format_t</title>
<para>The <emphasis role="tt">video_format_t</emphasis> data type defined by
</para>
<programlisting>
 typedef enum {
         VIDEO_FORMAT_4_3,
         VIDEO_FORMAT_16_9
 } video_format_t;
</programlisting>
<para>is used in the VIDEO_SET_FORMAT function (??) to tell the driver which
aspect ratio
the output hardware (e.g. TV) has. It is also used in the data structures
video_status
(??) returned by VIDEO_GET_STATUS (??) and video_event (??) returned by
VIDEO_GET_EVENT (??) which report about the display format of the current video
stream.
</para>
</section>

<section id="video_display_format_t">
<title>video_display_format_t</title>
<para>In case the display format of the video stream and of the display hardware
differ the
application has to specify how to handle the cropping of the picture. This can
be done using
the VIDEO_SET_DISPLAY_FORMAT call (??) which accepts
</para>
```

```
<programlisting>
 typedef enum {
         VIDEO_PAN_SCAN,
         VIDEO_LETTER_BOX,
         VIDEO_CENTER_CUT_OUT
 } video_display_format_t;
</programlisting>
<para>as argument.
</para>
</section>

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

<section id="video_play_state">
<title>video play state</title>
<para>The following values can be returned by the VIDEO_GET_STATUS call
representing the
state of video playback.
</para>
<programlisting>
 typedef enum {
         VIDEO_STOPPED,
         VIDEO_PLAYING,
         VIDEO_FREEZED
 } video_play_state_t;
</programlisting>
</section>

<section id="video_event">
<title>struct video_event</title>
<para>The following is the structure of a video event as it is returned by the
VIDEO_GET_EVENT
call.
</para>
<programlisting>
```

```
 struct video_event {
        int32_t type;
        time_t timestamp;
        union {
                video_format_t video_format;
        } u;
 };
</programlisting>
</section>

<section id="video_status">
<title>struct video_status</title>
<para>The VIDEO_GET_STATUS call returns the following structure informing about
various
states of the playback operation.
</para>
<programlisting>
 struct video_status {
        boolean video_blank;
        video_play_state_t play_state;
        video_stream_source_t stream_source;
        video_format_t video_format;
        video_displayformat_t display_format;
 };
</programlisting>
<para>If video_blank is set video will be blanked out if the channel is changed
or if playback is
stopped. Otherwise, the last picture will be displayed. play_state indicates if
the video is
currently frozen, stopped, or being played back. The stream_source corresponds
to the seleted
source for the video stream. It can come either from the demultiplexer or from
memory.
The video_format indicates the aspect ratio (one of 4:3 or 16:9) of the
currently
played video stream. Finally, display_format corresponds to the selected
cropping
mode in case the source video format is not the same as the format of the output
device.
</para>
</section>

<section id="video_still_picture">
<title>struct video_still_picture</title>
<para>An I-frame displayed via the VIDEO_STILLPICTURE call is passed on within
the
following structure.
</para>
<programlisting>
 /&#x22C6; pointer to and size of a single iframe in memory &#x22C6;/
 struct video_still_picture {
        char &#x22C6;iFrame;
        int32_t size;
 };
</programlisting>
</section>
```

```
<section id="video_caps">
<title>video capabilities</title>
<para>A call to VIDEO_GET_CAPABILITIES returns an unsigned integer with the
following
bits set according to the hardwares capabilities.
</para>
<programlisting>
 /&#x22C6; bit definitions for capabilities: &#x22C6;/
 /&#x22C6; can the hardware decode MPEG1 and/or MPEG2? &#x22C6;/
 #define VIDEO_CAP_MPEG1   1
 #define VIDEO_CAP_MPEG2   2
 /&#x22C6; can you send a system and/or program stream to video device?
    (you still have to open the video and the audio device but only
     send the stream to the video device) &#x22C6;/
 #define VIDEO_CAP_SYS     4
 #define VIDEO_CAP_PROG    8
 /&#x22C6; can the driver also handle SPU, NAVI and CSS encoded data?
    (CSS API is not present yet) &#x22C6;/
 #define VIDEO_CAP_SPU    16
 #define VIDEO_CAP_NAVI   32
 #define VIDEO_CAP_CSS    64
</programlisting>
</section>

<section id="video_system">
<title>video system</title>
<para>A call to VIDEO_SET_SYSTEM sets the desired video system for TV output.
The
following system types can be set:
</para>
<programlisting>
 typedef enum {
          VIDEO_SYSTEM_PAL,
          VIDEO_SYSTEM_NTSC,
          VIDEO_SYSTEM_PALN,
          VIDEO_SYSTEM_PALNc,
          VIDEO_SYSTEM_PALM,
          VIDEO_SYSTEM_NTSC60,
          VIDEO_SYSTEM_PAL60,
          VIDEO_SYSTEM_PALM60
 } video_system_t;
</programlisting>
</section>

<section id="video_highlight">
<title>struct video_highlight</title>
<para>Calling the ioctl VIDEO_SET_HIGHLIGHTS posts the SPU highlight
information. The
call expects the following format for that information:
</para>
<programlisting>
 typedef
 struct video_highlight {
         boolean active;      /&#x22C6;    1=show highlight, 0=hide highlight
&#x22C6;/
```

```
        uint8_t contrast1;   /&#x22C6;    7- 4  Pattern pixel contrast
&#x22C6;/
                             /&#x22C6;    3- 0  Background pixel contrast
&#x22C6;/
        uint8_t contrast2;   /&#x22C6;    7- 4  Emphasis pixel-2 contrast
&#x22C6;/
                             /&#x22C6;    3- 0  Emphasis pixel-1 contrast
&#x22C6;/
        uint8_t color1;      /&#x22C6;    7- 4  Pattern pixel color &#x22C6;/
                             /&#x22C6;    3- 0  Background pixel color
&#x22C6;/
        uint8_t color2;      /&#x22C6;    7- 4  Emphasis pixel-2 color
&#x22C6;/
                             /&#x22C6;    3- 0  Emphasis pixel-1 color
&#x22C6;/
        uint32_t ypos;       /&#x22C6;   23-22  auto action mode &#x22C6;/
                             /&#x22C6;   21-12  start y &#x22C6;/
                             /&#x22C6;    9- 0  end y &#x22C6;/
        uint32_t xpos;       /&#x22C6;   23-22  button color number &#x22C6;/
                             /&#x22C6;   21-12  start x &#x22C6;/
                             /&#x22C6;    9- 0  end x &#x22C6;/
 } video_highlight_t;
</programlisting>

</section>
<section id="video_spu">
<title>video SPU</title>
<para>Calling VIDEO_SET_SPU deactivates or activates SPU decoding, according to
the
following format:
</para>
<programlisting>
 typedef
 struct video_spu {
        boolean active;
        int stream_id;
 } video_spu_t;
</programlisting>

</section>
<section id="video_spu_palette">
<title>video SPU palette</title>
<para>The following structure is used to set the SPU palette by calling
VIDEO_SPU_PALETTE:
</para>
<programlisting>
 typedef
 struct video_spu_palette{
        int length;
        uint8_t &#x22C6;palette;
 } video_spu_palette_t;
</programlisting>

</section>
<section id="video_navi_pack">
<title>video NAVI pack</title>
```

<para>In order to get the navigational data the following structure has to be
passed to the ioctl
VIDEO_GET_NAVI:
</para>
<programlisting>
 typedef
 struct video_navi_pack{
         int length;   /⋆ 0 ... 1024 ⋆/
         uint8_t data[1024];
 } video_navi_pack_t;
</programlisting>
</section>

<section id="video_attributes">
<title>video attributes</title>
<para>The following attributes can be set by a call to VIDEO_SET_ATTRIBUTES:
</para>
<programlisting>
 typedef uint16_t video_attributes_t;
 /⋆   bits: descr. ⋆/
 /⋆   15-14 Video compression mode (0=MPEG-1, 1=MPEG-2) ⋆/
 /⋆   13-12 TV system (0=525/60, 1=625/50) ⋆/
 /⋆   11-10 Aspect ratio (0=4:3, 3=16:9) ⋆/
 /⋆   9- 8 permitted display mode on 4:3 monitor (0=both, 1=only pan-sca
⋆/
 /⋆   7   line 21-1 data present in GOP (1=yes, 0=no) ⋆/
 /⋆   6   line 21-2 data present in GOP (1=yes, 0=no) ⋆/
 /⋆   5- 3 source resolution (0=720x480/576, 1=704x480/576, 2=352x480/57
⋆/
 /⋆   2   source letterboxed (1=yes, 0=no) ⋆/
 /⋆   0   film/camera mode (0=camera, 1=film (625/50 only)) ⋆/
</programlisting>
</section></section>

<section id="video_function_calls">
<title>Video Function Calls</title>

<section id="video_fopen">
<title>open()</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This system call opens a named video device (e.g.
/dev/dvb/adapter0/video0)
 for subsequent use.</para>
<para>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 Video 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 Video Device is opened in O_RDONLY
 mode, the only ioctl call that can be used is VIDEO_GET_STATUS. All other
 call will return 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 video 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="video_fclose">
<title>close()</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This system call closes a previously opened video 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="video_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 VIDEO_SOURCE_MEMORY is selected
 in the ioctl call VIDEO_SELECT_SOURCE. The data provided shall be in
 PES format, unless the capability allows other formats. 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 VIDEO_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>VIDEO_STOP</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Video Device to stop playing the current stream.
 Depending on the input parameter, the screen can be blanked out or displaying
 the last decoded frame.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_STOP, 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 VIDEO_STOP for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>Boolean mode</para>
</entry><entry
 align="char">
<para>Indicates how the screen shall be handled.</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>TRUE: Blank screen when stop.</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>FALSE: Show last decoded frame.</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, possibly in the communication with the
 DVB subsystem.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_PLAY</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Video Device to start playing a video 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(fd, int request = VIDEO_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 VIDEO_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, possibly in the communication with the
 DVB subsystem.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_FREEZE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call suspends the live video stream being played. Decoding
 and playing are frozen. It is then possible to restart the decoding
 and playing process of the video stream using the VIDEO_CONTINUE

```
 command. If VIDEO_SOURCE_MEMORY is selected in the ioctl call
 VIDEO_SELECT_SOURCE, the DVB subsystem will not decode any more
 data until the ioctl call VIDEO_CONTINUE or VIDEO_PLAY is performed.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_FREEZE);</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 VIDEO_FREEZE 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, possibly in the communication with the
 DVB subsystem.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_CONTINUE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call restarts decoding and playing processes of the video
stream
```

```
 which was played before a call to VIDEO_FREEZE was made.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_CONTINUE);</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 VIDEO_CONTINUE 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, possibly in the communication with the
 DVB subsystem.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SELECT_SOURCE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call informs the video device which source shall be used for
the input
 data. The possible sources are demux or memory. If memory is selected, the
 data is fed to the video 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(fd, int request = VIDEO_SELECT_SOURCE,
 video_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 VIDEO_SELECT_SOURCE for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_stream_source_t
 source</para>
</entry><entry
 align="char">
<para>Indicates which source shall be used for the Video 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, possibly in the communication with the
 DVB subsystem.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SET_BLANK</title>
```

```
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Video Device to blank out the picture.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_SET_BLANK, 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 VIDEO_SET_BLANK for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>boolean mode</para>
</entry><entry
 align="char">
<para>TRUE: Blank screen when stop.</para>
</entry>
 </row><row><entry
 align="char">
</entry><entry
 align="char">
<para>FALSE: Show last decoded frame.</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, possibly in the communication with the
 DVB subsystem.</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>VIDEO_GET_STATUS</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Video Device to return the current status of the
device.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request = VIDEO_GET_STATUS, struct
 video_status ⋆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 VIDEO_GET_STATUS for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>struct video_status
 *status</para>
</entry><entry
 align="char">
<para>Returns the current status of the Video 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, possibly in the communication with the
 DVB subsystem.</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>VIDEO_GET_EVENT</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call returns an event of type video_event if available. If an
event is
 not available, the behavior depends on whether the device is in blocking or
 non-blocking mode. In the latter case, the call fails immediately with errno
 set to EWOULDBLOCK. In the former case, the call blocks until an event
 becomes available. The standard Linux poll() and/or select() system calls can
 be used with the device file descriptor to watch for new events. For select(),
 the file descriptor should be included in the exceptfds argument, and for
 poll(), POLLPRI should be specified as the wake-up condition. Read-only
 permissions are sufficient for this ioctl call.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request = VIDEO_GET_EVENT, struct
 video_event &#x22C6;ev);</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 VIDEO_GET_EVENT for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>struct video_event
 *ev</para>
</entry><entry
 align="char">
<para>Points to the location where the event, if any, is to be
 stored.</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>EFAULT</para>
</entry><entry
 align="char">
<para>ev points to invalid address</para>
</entry>
 </row><row><entry
 align="char">
<para>EWOULDBLOCK</para>
</entry><entry
 align="char">
<para>There is no event pending, and the device is in
 non-blocking mode.</para>
</entry>
 </row><row><entry
 align="char">
<para>EOVERFLOW</para>
</entry><entry
 align="char">
</entry>
 </row><row><entry
 align="char">

```
</entry><entry
 align="char">
<para>Overflow in event queue - one or more events were lost.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SET_DISPLAY_FORMAT</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Video Device to select the video format to be
applied
 by the MPEG chip on the video.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request =
 VIDEO_SET_DISPLAY_FORMAT, video_display_format_t
 format);</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 VIDEO_SET_DISPLAY_FORMAT for this
 command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_display_format_t
 format</para>
</entry><entry
 align="char">
<para>Selects the video format to be used.</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 parameter format.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_STILLPICTURE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Video Device to display a still picture
(I-frame). The
 input data shall contain an I-frame. If the pointer is NULL, then the current
 displayed still picture is blanked.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_STILLPICTURE,
 struct video_still_picture ⋆sp);</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 VIDEO_STILLPICTURE for this command.</para>

```
</entry>
 </row><row><entry
 align="char">
<para>struct
 video_still_picture
 *sp</para>
</entry><entry
 align="char">
<para>Pointer to a location where an I-frame and size is stored.</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>sp points to an invalid iframe.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_FAST_FORWARD</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the Video Device to skip decoding of N number of
I-frames.
 This call can only be used if VIDEO_SOURCE_MEMORY is selected.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_FAST_FORWARD, int
 nFrames);</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 VIDEO_FAST_FORWARD for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>int nFrames</para>
</entry><entry
 align="char">
<para>The number of frames to skip.</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>EPERM</para>
</entry><entry
 align="char">
<para>Mode VIDEO_SOURCE_MEMORY not selected.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Illegal parameter format.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
```

role="subsection"><title>VIDEO_SLOWMOTION</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the video device to repeat decoding frames N number
of
 times. This call can only be used if VIDEO_SOURCE_MEMORY is selected.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_SLOWMOTION, int
 nFrames);</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 VIDEO_SLOWMOTION for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>int nFrames</para>
</entry><entry
 align="char">
<para>The number of times to repeat each frame.</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>EPERM</para>
</entry><entry
 align="char">
<para>Mode VIDEO_SOURCE_MEMORY not selected.</para>
</entry>
 </row><row><entry
 align="char">
<para>EINVAL</para>
</entry><entry
 align="char">
<para>Illegal parameter format.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_GET_CAPABILITIES</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call asks the video device about its decoding capabilities. On
success
 it returns and integer which has bits set according to the defines in section
??.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_GET_CAPABILITIES,
 unsigned int ⋆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 VIDEO_GET_CAPABILITIES for this
 command.</para>
</entry>
 </row><row><entry

```
 align="char">
<para>unsigned int *cap</para>
</entry><entry
 align="char">
<para>Pointer to a location where to store the capability
 information.</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>EFAULT</para>
</entry><entry
 align="char">
<para>cap points to an invalid iframe.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_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.</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 = VIDEO_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 VIDEO_SET_ID for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>int id</para>
</entry><entry
 align="char">
<para>video 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>VIDEO_CLEAR_BUFFER</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl call clears all video buffers in the driver and in the decoder hardware.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_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 VIDEO_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></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SET_STREAMTYPE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl tells the driver which kind of stream to expect being written
to it. If
 this call is not used the default of video PES is used. Some drivers might not
 support this call and always expect PES.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>int ioctl(fd, int request = VIDEO_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 VIDEO_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>VIDEO_SET_FORMAT</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl sets the screen format (aspect ratio) of the connected output
device
 (TV) so that the output of the decoder can be adjusted accordingly.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request = VIDEO_SET_FORMAT,
 video_format_t format);</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 VIDEO_SET_FORMAT for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_format_t
 format</para>
</entry><entry
 align="char">
<para>video format of TV as defined in 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>format is not a valid video format.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SET_SYSTEM</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl sets the television output format. The format (see section ??)
may
 vary from the color format of the displayed MPEG stream. If the hardware is
 not able to display the requested format the call will return an error.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request = VIDEO_SET_SYSTEM ,

```
 video_system_t system);</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 VIDEO_SET_FORMAT for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_system_t
 system</para>
</entry><entry
 align="char">
<para>video system of TV output.</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>system is not a valid or supported video system.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SET_HIGHLIGHT</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl sets the SPU highlight information for the menu access of a
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 = VIDEO_SET_HIGHLIGHT
 ,video_highlight_t ⋆vhilite)</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 VIDEO_SET_HIGHLIGHT for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_highlight_t
 *vhilite</para>
</entry><entry
 align="char">
<para>SPU Highlight information 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>input is not a valid highlight setting.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SET_SPU</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry

```
 align="char">
<para>This ioctl activates or deactivates SPU decoding in a DVD input stream. It
can
 only be used, if the driver is able to handle a DVD 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 = VIDEO_SET_SPU ,
 video_spu_t &#x22C6;spu)</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 VIDEO_SET_SPU for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_spu_t *spu</para>
</entry><entry
 align="char">
<para>SPU decoding (de)activation and subid setting 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>input is not a valid spu setting or driver cannot handle
 SPU.</para>
</entry>
```

```
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_SET_SPU_PALETTE</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl sets the SPU color palette.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request = VIDEO_SET_SPU_PALETTE
 ,video_spu_palette_t &#x22C6;palette )</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 VIDEO_SET_SPU_PALETTE for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_spu_palette_t
 *palette</para>
</entry><entry
 align="char">
<para>SPU palette 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>input is not a valid palette or driver doesn&#8217;t handle SPU.</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_GET_NAVI</title>
<para>DESCRIPTION
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para>This ioctl returns navigational information from the DVD stream. This is
 especially needed if an encoded stream has to be decoded by the
hardware.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request = VIDEO_GET_NAVI ,
 video_navi_pack_t &#x22C6;navipack)</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 VIDEO_GET_NAVI for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_navi_pack_t
 *navipack</para>
</entry><entry
 align="char">
<para>PCI or DSI pack (private stream 2) 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>EFAULT</para>
</entry><entry
 align="char">
<para>driver is not able to return navigational information</para>
</entry>
 </row></tbody></tgroup></informaltable>

</section><section
role="subsection"><title>VIDEO_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 stream. Some hardware may not need this information,
 but the call also tells the hardware to prepare for DVD playback.</para>
</entry>
 </row></tbody></tgroup></informaltable>
<para>SYNOPSIS
</para>
<informaltable><tgroup cols="1"><tbody><row><entry
 align="char">
<para> int ioctl(fd, int request = VIDEO_SET_ATTRIBUTE
 ,video_attributes_t vattr)</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 VIDEO_SET_ATTRIBUTE for this command.</para>
</entry>
 </row><row><entry
 align="char">
<para>video_attributes_t
 vattr</para>
</entry><entry
 align="char">
<para>video 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>input is not a valid attribute setting.</para>
</entry>
 </row></tbody></tgroup></informaltable>
 </section></section>
```