National Semiconductor

August 2001
Revision 1.0

Geode™ SC1200/SC1210 IAOC Devices: Video4Linux2 Programmer's Reference

Introduction

This document describes the configuration and use of the National Semiconductor® Video4Linux2 software for set-top box reference platforms based on National's Information Appliance on a Chip (IAOC) devices: Geode™ SC1200 Set-Top Box on a Chip and Geode SC1210 Multimedia System on a Chip.

General Description

Linux has grown from a simple Internet community project to a widely used operating system for servers, desktop, and more recently embedded applications. Linux is very similar to the standard UNIX distributions and has been primarily used in server applications, thus leaving little support for high-end graphics, audio, or other multimedia applications.

The Video4Linux2 software enables simultaneous viewing of analog video display on a VGA monitor and TV. It combines features of video overlay, alpha blending, video encoding, closed captioning (CC), VBI data capture, and more, all on a small footprint.

The V4L2 driver for Geode SC1200 or SC1210 based systems is based on the V4L2 sample driver. It is dependent on the V4L2 subsystem. RedHat Linux 6.2 uses the original Video4Linux API for its distribution.

Physical Data Flow

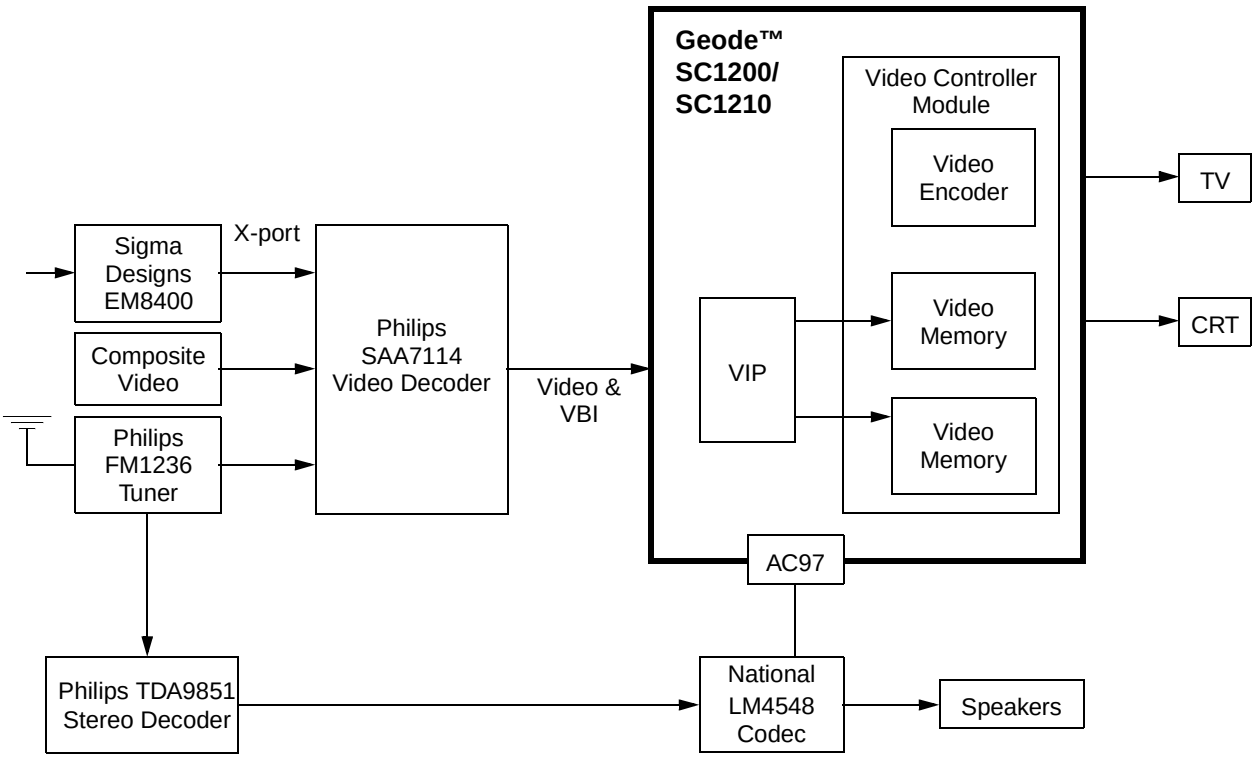

Table of Contents

1.0 Configuring The Software For The Platform 3

1.1 CONFIGURING DRIVER FOR XPRESSROM AND CYGNUS 3

1.1.1 Using the I^2C Interface 3

1.1.2 Excluding the Tuner and/or the Decoder 3

1.1.3 Load Time Options for the Module 3

2.0 Using the Driver 4

2.1 OPENING AND CLOSING THE DEVICE 4

2.1.1 Multiple Opens 4

2.2 QUERYING CAPABILITIES 4

2.3 VIDEO FORMAT SETUP 5

2.4 VIDEO WINDOW SIZE AND POSITION SETUP 5

2.5 VBI 6

2.5.1 VBI Format Setup 6

2.5.2 Reading VBI Data 8

2.6 SIGNALING FEATURE SETUP 8

2.7 VIDEO OVERLAY SETUP 8

2.8 ALPHA BLENDING SURFACES SETUP 9

2.9 FINER CONTROL OF OVERLAY AND ALPHA-BLEND FEATURES 9

2.9.1 Alpha Window 9

2.9.2 Alpha Operation 9

2.9.3 Video Outside Alpha 9

2.9.4 Color Key 10

2.9.5 Set Graphics on Video 10

2.10 VIDEO INPUT SETUP 10

2.11 CHANGING TV CHANNELS: TUNING THE TUNER 10

2.12 ADJUSTING THE VIDEO DECODER PROPERTIES 10

2.12.1 Brightness 10

2.12.2 Contrast 10

2.12.3 Saturation 10

2.12.4 Hue 11

2.12.5 Sharpness 11

3.0 How geode_v.o Differs from Other Video4Linux2 Drivers 12

4.0 Unit Testing 12

4.1 VBI TESTING 12

4.2 VIDEO TESTING 12

4.3 OVERLAY AND ALPHA BLENDING TESTING 12

5.0 Supporting Documentation 13

1.0 Configuring The Software For The Platform

The geode_v.o driver source may be configured in several ways. Modify the Makefile with compile-time flags to achieve this.

1.1 CONFIGURING DRIVER FOR XPRESSROM AND CYGNUS

The Video4Linux2 software runs on SC1200/SC1210-based platforms. Any Geode SP1SC10 reference platform-specific code is flagged with CYGNUS. Modify the Makefile to build for a particular platform.

To compile the driver for the SP1SC10 platform using XpressROM use the compile flags -DXPRESSROM and -DCYGNUS.

Cygnus NTSC systems use an Alps TMDH2 TV tuner, and PAL (Phase Alternating Line) systems use an Alps TMDZ2 tuner. These tuners must be programmed differently. The correct tuner is supported for the SP1SC10 by defining the -DCYGNUS flag in the Makefile. One significant difference between the firmware implementations for the SP1SC10 is the ACCESS.bus (ACB) base addresses. To enhance portability, these base addresses may be determined dynamically.

Another important issue regarding Phillips I^2C access is the fact that the SP1SC10 uses two I^2C buses. To abstract the I^2C access from the core logic of the driver, use the modularized I^2C Linux subsystem described below.

1.1.1 Using the I^2C Interface

Linux now includes a standard technique for accessing the I^2C buses on the system. This support is included as a component of the distribution for kernels 2.4 and later. Earlier kernels may use add-on packages i2c-2.5.2.tar.gz and lm_sensors-2_5_2.tar.gz. The Geode-specific I^2C code is integrated into these so that they can work with the ACCESS.bus. See the National Semiconductor specification for Geode-specific I^2C code and release notes for more information.

To build the geode_v.o driver with this standard I^2C support, use the -DUSE_SYSTEM_I2C flag in the Makefile.

1.1.2 Excluding the Tuner and/or the Decoder

To remove tuner support from the driver, use the -DEXCLUDE_TUNER flag. Some systems may not have the Video Decoder. To exclude the Video Decoder, use the -DNO_DECODER compile flag.

1.1.3 Load Time Options for the Module

The driver module can be loaded with different options specified below.

- IRQ - IRQ number to be used by timer (VideoLib). (8)
- VBI_IRQ - IRQ number to be used by VBI interrupt. (11)
- VIN_STD - Input standard, 1 = NTSC, 2= PAL. (1)
- VOUT_STD - Output standard 1 = NTSC, 2 = PAL. (1)
  - Set VOUT_STD to any other value other than 1 or 2 in order for the video settings to remain unchanged.
- USE_CRT - Show output on CRT (YUV mixing). (0)
- NO_DVIP - Dont try to go into DVIP even if possible. (1)
- DEINTERLACE - Deinterlacing mode.
  - 1 = NONE
  - 2 = INTERLEAVE
  - 3 = BOB
- VOUT_MINOR - The minor device number for the Video output device. (16)
- VIDEO_MINOR - The minor device number for the Video capture device. (0)
- VBI_MINOR - The minor device number for the VBI device. (224)
- MAX_VBI_STREAM_BUFFERS - The number of stream buffers to allocate in the frame buffer.
- MAX_VBI_OPEN_BUFFERS - The number of per-open buffers.

2.0 Using the Driver

The programming examples below assume that the device handle has been successfully opened before use. Applications using these API calls must also include the Video4Linux2 header, followed by the ioctl_geode_v.h header.

#include <linux/videodev.h> /* V4L2 include */

#include <ioctl_geode_v.h> /* IOCTLs and structure dfns for the geode driver*/

Sample code implementing this is available in the Vtest4Linux, vbitest, and captest applications.

2.1 OPENING AND CLOSING THE DEVICE

The device may be opened and closed as follows:

```
int   vid = 0;
int   vbi = 0;
int   vout = 0;

vid = open("/dev/video",O_RDONLY);
if( vid == -1 )

// error

close( vid );

vbi = open( "/dev/vbi", O_RDONLY );
if( vbi == -1 )

// error

close( vid );

vout = open( "/dev/vout", O_RDONLY );
if( vout == -1 )

// error

close( vout );
```

2.1.1 Multiple Opens

The V4L2 driver supports multiple opens. Each device can have only one capturing open. All subsequent opens on each device must be non-capturing opens. This is in compliance with the V4L2 specifications. A non-capturing open is carried out as follows.

```
vbi = open( "/dev/vbi", O_RDONLY | O_NONCAP );
if( vbi < 0 )

// error
```

Multiple opens are specially useful for writing control-panel type applications in a different process. Multiple opens are also used to capture multi-standard, sliced VBI from multiple lines as explained later.

2.2 QUERYING CAPABILITIES

A process can query a device about its capabilities.

```
struct v4l2_capability caps;
memset( &caps, 0, sizeof(caps) );

if( ioctl( vid, VIDIOC_QUERYCAP, &caps ) < 0 )

// error
```

2.3 VIDEO FORMAT SETUP

This IOCTL is supported by the video capture device (/dev/video). It is used to setup the scaling parameters of the TV decoder. The video capture device deals only with the decoder. The video window position and size must be set by the VIDIOC_S_WIN IOCTL on /dev/vout.

```
struct v4l2_format fmt;
memset( &fmt, 0, sizeof(fmt) );

fmt.fmt.pix.width  = 320;
fmt.fmt.pix.height = 240;

if( ioctl( vid, VIDIOC_S_FMT, &fmt ) < 0 )

// error
```

2.4 VIDEO WINDOW SIZE AND POSITION SETUP

The VIDIOC_S_WIN IOCTL is used to set the video window position and size for the video output device. The width and height in the IOCTL should be the same as the values passed to the VIDIOC_S_FMT IOCTL on the video capture device.

```
struct v4l2_window window;
memset( &window, 0, sizeof(window) );
window.x = 0;
window.y = 0;
window.width = 320;
window.height = 240
if( ioctl( vout, VIDIOC_S_WIN, &window ) < 0 )

// error
```

2.5 VBI

The VBI module follows the V4L2 recommended standards. The figures show the supported line numbers for the NTSC and PAL/SECAM standards.

2.5.1 VBI Format Setup

The SAA7114 can be programmed to pre-process several different types of VBI data. The Geode VIP VBI capture region can store multiple lines of VBI data. The VBI module of the Geode V4L2 driver can provide multiple lines of data within the VBI region. Each open of the VBI device can be configured to use a different decoder slicer format for the user specified set of lines. The capture regions of different opens cannot overlap.

The VBI regions are, lines 10 to 21 and lines 273 to 284 for NTSC, and lines 6 to 22 and lines 319 to 335 for PAL, as shown in Figure 2-1 and Figure 2-2. Any region outside of these regions will not be accepted by the driver.

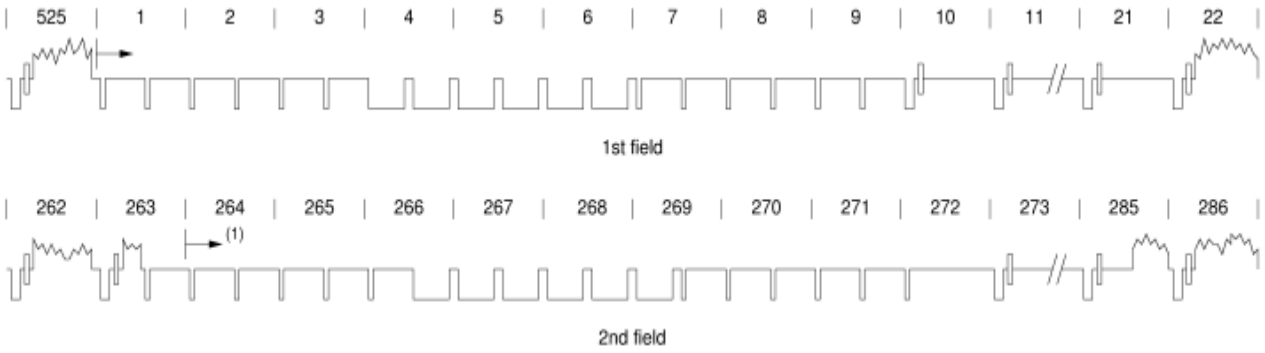

Figure 2-1. NTSC (525) Line Numbering

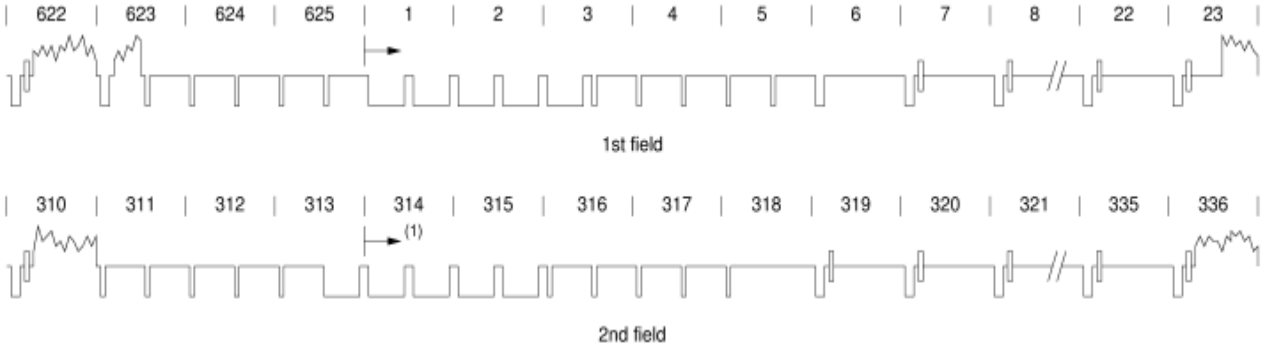

Figure 2-2. PAL/SECAM (625) Line Numbering

Table 2-1 shows the recognized data types supported by the driver.

Table 2-1. Supported Data Types

Sample Format	Description
GEODE_V4L2_VBI_SF_CC_US	US closed caption
GEODE_V4L2_VBI_SF_CC_EURO	European closed caption
GEODE_V4L2_VBI_SF_VPS	Video Programming service
GEODE_V4L2_VBI_SF_WSS	Wide-screen signalling bits
GEODE_V4L2_VBI_SF_WST	US Teletext (WST)
GEODE_V4L2_VBI_SF_RAW	RAW, Y-only data
GEODE_V4L2_VBI_SF_TELETEXT	Teletext
GEODE_V4L2_VBI_SF_NABTS	US NABTS
GEODE_V4L2_VBI_SF_MOJI	MOJI (Japanese)
GEODE_V4L2_VBI_SF_JFS	Japanese format switch
GEODE_V4L2_VBI_SF_VIDEO	YUV data
GEODE_V4L2_VBI_SF_TS_US	VITC/SMPTE time codes (US)
GEODE_V4L2_VBI_SF_TS_EURO	VITC/EBU time codes (Europe)

The following shows how to program the driver to capture only one line of VBI from each frame and to interpret the data as US closed caption.

```
struct v4l2_format fmt;

memset( &fmt, 0, sizeof(fmt) );

fmt.fmt.vbi.sampling_rate = 60;
fmt.fmt.vbi.offset = 9;
fmt.fmt.vbi.samples_per_line = 2;
fmt.fmt.vbi.sample_format = GEODE_V4L2_VBI_SF_CC_US;
fmt.fmt.vbi.start[0] = 21;
fmt.fmt.vbi.start[1] = 284;
fmt.fmt.vbi.count[0] = 1; /* One line from this field */
fmt.fmt.vbi.count[1] = 1; /* One line from this field */
fmt.fmt.vbi.flags = 0;

if( ioctl(vbi,VIDIOC_S_FMT,&fmt) < 0 )

// error
```

If there are no opens already capturing the same region, the VIDIOC_S_FMT will succeed.

2.5.2 Reading VBI Data

The number of bytes per line to be captured is specified by the user in the samples_per_line field of the VBI format. The minimum amount of data transferred per read is count[0]+count[1]*samples_per_line*sample size in bytes. The user buffer for a read should have this minimum size. The read will fail if the size is less than the minimum. The actual number of bytes transferred per line is the minimum of the number of bytes specified by the user and the number of valid bytes sliced by the decoder. The two field images will be transmitted sequentially, that is, all lines of the first field followed by all lines of the second field.

The application must use the VIDIOC_STREAMON call to tell the driver to start queueing VBI data.

```
// How to request sliced US CC data.
__u8  vbi_data[ 4 ]; /* 2 samples per line, one line per
             * field,
             * two fields
             */

int len;
int streamon_val = 0;

memset( &vbi_data, 0, sizeof(vbi_data) );

if( ioctl( vbi, VIDIOC_STREAMON, &streamon_val ) < 0 )

       // error

len = read( vbi, vbi_data, 4 );

if( len < 0 )

       // error

// Parse the CC and display it.

...
```

If there is not enough data to satisfy the read, the process will be put to sleep and will be woken up when there is enough data.

2.6 SIGNALING FEATURE SETUP

The driver can be setup to signal the process with either SIGUSR1 or SIGUSR2 when there is enough data to be read. The level at which the signal should be sent is also configurable. This feature is setup as follows:

```
void signal_handler(int sig)

       printf("Got signal\n");
       // ... read the data and process it here

       struct  signal_info_t   sig_info;
       sig_info.signal_number = SIGUSR1; /* Send SIGUSR1 */
       sig_info.signal_on_length = 2; /*Signal when two buffers are full*/
       signal(SIGUSR1,signal_handler); /* register the signal handler*/
       if( ioctl( vbi, GEODE_IOCTL_S_SIGNAL, &sig_info ) < 0 )

              printf("failed to set signal\n");

       if( ioctl( vbi, VIDIOC_STREAMON, &stream_on ) < 0 )

              printf("failed to stream on\n");
```

2.7 VIDEO OVERLAY SETUP

```
overlay_info_t overlay_info;
memset( &overlay_info, 0, sizeof(overlay_info) );

overlay_info.ColorKey = 0x00F000F0;
overlay_info.ColorMask = 0x00F0F0F0;
overlay_info.Graphics = 1;
overlay_info.Enable = 1;

if( ioctl( vid, GEODE_IOCTL_SET_OVERLAY_INFO, &overlay_info ) < 0 )

// error
```

2.8 ALPHA BLENDING SURFACES SETUP

```
alpha_blend_info_t alpha_blend_info;
memset( &alpha_blend_info, 0, sizeof(alpha_blend_info) );

alpha_blend_info.WindowNumber = 2;
alpha_blend_info.Priority = 1;
alpha_blend_info.StartX = 50;
alpha_blend_info.StartY = 20;
alpha_blend_info.Width = 100;
alpha_blend_info.Height = 80;
alpha_blend_info.Alpha = 0x80
alpha_blend_info.Delta = 1;
alpha_blend_info.Enable = 1;

if( ioctl( vout, GEODE_IOCTL_SET_ALPHA_BLEND_INFO, &alpha_blend_info ) < 0 )

// error
```

2.9 FINER CONTROL OF OVERLAY AND ALPHA-BLEND FEATURES

2.9.1 Alpha Window

Sets the alpha window position and size.

```
geode_general_param_t param;

param.p_1 = 0;   /* window number */
param.p_2 = 20;  /* x */
param.p_3 = 20;  /* y */
param.p_4 = 200; /* width */
param.p_5 = 300; /* height */

if (ioctl(vout, GEODE_IOCTL_S_ALPHA_WINDOW, &param) < 0)

//error
```

2.9.2 Alpha Operation

Sets the relevant values for the alpha window operation desired.

geode_general_param_t param;

```
param.p_1 = 0;     /* window number */
param.p_2 = 150;        /* alpha value */
param.p_3 = 0x00F000F0; /* color key */
param.p_4 = 1;   /* enable color */
param.p_5 = 1;   /* priority */
param.p_6 = 0;   /* fade */
param.p_7 = 1;   /* enable */

if ( ioctl(vout, GEODE_IOCTL_S_ALPHA_OPER, &param) < 0)

//error
```

2.9.3 Video Outside Alpha

Enable/disable video display outside the alpha window.

```
geode_general_param_t param;

param.p_1 = 1; /* enable */
if (ioctl( vout, GEODE_IOCTL_VIDEO_OUTSIDE_ALPHA, &param) < 0)

//error
```

2.9.4 Color Key

Sets the color/chroma key and mask values.

```
geode_general_param_t param;

param.p_1 = 0x00F000F0; //color key
param.p_2 = 0x00F0F0F0; //mask

if (ioctl(vout, GEODE_IOCTL_S_COLOR_KEY, &param) < 0)

//error
```

2.9.5 Set Graphics on Video

Enable/disable graphics on video.

```
geode_general_param_t param;

param.p_1 = 1    /* enable */
param.p_2 = TRUE /* graphics stream compared to color key */

if (ioctl(vout, GEODE_IOCTL_S_GRAPHICS_ON_VID, &param) < 0)

//error
```

2.10 VIDEO INPUT SETUP

This driver by default assumes the configuration of the SP1SC10 platform and supports four inputs: tuner, composite video, s-video input, and digital input from the MPEG decoder. They are indexed 0 to 3 starting from the tuner.

```
int input = 0;
if( ioctl( video, VIDIOC_S_INPUT, &input ) < 0 )

// error
```

2.11 CHANGING TV CHANNELS: TUNING THE TUNER

```
int channel = 14;
if( ioctl( vid, VIDIOC_S_FREQ, &channel ) < 0 )

// error
```

2.12 ADJUSTING THE VIDEO DECODER PROPERTIES

2.12.1 Brightness

Brightness can take values 0 to 255.

```
struct v4l2_control ctrl;
ctrl.id = V4L2_CID_BRIGHTNESS;
ctrl.value = 100;
if( ioctl( video, VIDIOC_S_CTRL, &ctrl ) < 0 )

// error
```

2.12.2 Contrast

Contrast can take values 0 to 255.

```
struct v4l2_control ctrl;
ctrl.id = V4L2_CID_CONTRAST;
ctrl.value = 100;
if( ioctl( video, VIDIOC_S_CTRL, &ctrl ) < 0 )

// error
```

2.12.3 Saturation

Saturation can take values 0 to 255.

```
struct v4l2_control ctrl;
ctrl.id = V4L2_CID_SATURATION;
ctrl.value = 100;
if( ioctl( video, VIDIOC_S_CTRL, &ctrl ) < 0 )

// error
```

2.12.4 Hue

Hue can take values 0 to 255.

```
struct v4l2_control ctrl;
ctrl.id = V4L2_CID_HUE;
ctrl.value = 100;
if( ioctl( video, VIDIOC_S_CTRL, &ctrl ) < 0 )

// error
```

2.12.5 Sharpness

Sharpness can take values 0 to 0x0f.

```
struct v4l2_control ctrl;
ctrl.id = V4L2_CID_WHITENESS;/*V4L2 does not have sharpness*/
ctrl.value = 100;
if( ioctl( video, VIDIOC_S_CTRL, &ctrl ) < 0 )

// error
```

3.0 How geode_v.o Differs from Other Video4Linux2 Drivers

- Currently, tuner channel changes are made by passing the requested channel and not the requested frequency. For example, the following code will tune to channel 14.

```
unsigned long channel = 14;
if( ioctl( vid, VIDIOC\_S\_FREQ, &channel ) < 0 )
// error
```

- The VIDIOC_STREAMON call must be used to tell the driver to start queueing VBI data.
- The device /dev/vout allows the overlay surface to be configured. See the sample test application for demonstration code.
- Currently, the geode_v.o driver does not support any video capture.

4.0 Unit Testing

4.1 VBI TESTING

Use the vbitest application to verify that VBI data is handled correctly. Invoke vbitest with the command-line option to tune to the local PBS TV station. When closed captioning (CC) is broadcast, parsed data should appear in the console window. Use a regular television set configured to display CC to confirm that CC is being broadcast.

4.2 VIDEO TESTING

Use the testapp test application to verify that video is handled correctly.

4.3 OVERLAY AND ALPHA BLENDING TESTING

Use the testapp test application to verify that overlay and alpha blending is done correctly.

5.0 Supporting Documentation

Table 5-1. Video4Linux2 IOCTLs

IOCTL	/dev/video	/dev/vout	/dev/vbi
VIDIOC_QUERYCAP	Yes	No	No
VIDIOC_RESERVED	No	No	No
VIDIOC_ENUM_PIXFMT	No	No	No
VIDIOC_ENUM_FBUFFMT	No	No	No
VIDIOC_G_FMT	Yes	Yes	Yes
VIDIOC_S_FMT	Yes	No	Yes
VIDIOC_G_COMP	No	No	No
VIDIOC_S_COMP	No	No	No
VIDIOC_REQBUFS	No	No	No
VIDIOC_QUERYBUF	No	No	No
VIDIOC_G_FBUF	No	No	No
VIDIOC_S_FBUF	No	No	No
VIDIOC_G_WIN	No	No	No
VIDIOC_S_WIN	No	Yes	No
VIDIOC_PREVIEW	No	Yes	No
VIDIOC_QBUF	No	No	No
VIDIOC_DQBUF	No	No	No
VIDIOC_STREAMON	Yes	No	Yes
VIDIOC_STREAMOFF	Yes	No	Yes
VIDIOC_G_PERF	No	No	No
VIDIOC_G_PARM	No	No	No
VIDIOC_S_PARM	No	No	No
VIDIOC_G_STD	Yes	No	No
VIDIOC_S_STD	Yes	Yes	No
VIDIOC_ENUMSTD	Yes	No	No
VIDIOC_ENUMINPUT	Yes	No	No
VIDIOC_G_CTRL	Yes	No	No
VIDIOC_S_CTRL	Yes	No	No
VIDIOC_G_TUNER	Yes	No	No
VIDIOC_S_TUNER	Yes	No	No
VIDIOC_G_FREQ	Yes	No	No
VIDIOC_S_FREQ	Yes	No	No
VIDIOC_G_AUDIO	No	No	No
VIDIOC_S_AUDIO	No	No	No
VIDIOC_QUERYCTRL	Yes	No	No
VIDIOC_QUERYMENU	No	No	No
VIDIOC_G_INPUT	Yes	No	No
VIDIOC_S_INPUT	Yes	No	No
VIDIOC_G_OUTPUT	No	No	No

Table 5-1. Video4Linux2 IOCTLs

IOCTL	/dev/video	/dev/vout	/dev/vbi
VIDIOC_S_OUTPUT	No	No	No
VIDIOC_ENUMOUTPUT	No	No	No
VIDIOC_ENUMFX	No	No	No
VIDIOC_G_EFFECT	No	No	No
VIDIOC_S_EFFECT	No	No	No

Table 5-2. Device Driver Interfaces

Driver Entry	/dev/video	/dev/vout	/dev/vbi
open()	Yes	Yes	Yes
close()	Yes	Yes	Yes
read()	No	No	Yes
write()	No	No	No
ioctl()	Yes	Yes	Yes
mmap()	No	No	No
poll()	No	No	Yes
initialize()	No	No	No

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Email:
new.feedback@nsc.com

National Semiconductor
Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 87 90

National Semiconductor
Asia Pacific Customer
Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor
Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507
Email: nsj.crc@jksmtp.nsc.com

www.national.com