VS1005 VSOS AUDIO SUBSYSTEM

VS1005g

All information in this document is provided as-is without warranty. Features are subject to change without notice.

Revision History			
Rev.	**Date**	**Author**	**Description**
1.00	2015-09-04	HH	Initial release.

Contents

List of Figures

1 Introduction

The VS1005 VSOS offers many, versatile audio drivers.

This document explains how to use the numerous drivers to your best advantage.

After the disclaimer and definitions in Chapters 2 and 3, an overview of the Audio subsystem is given in Chapter 4, *Overview*, followed by requirements in Chapter 5, *Requirements*.

The VSOS audio subsystem is presented in Chapter 6.

The currently existing audio drivers are presented in Chapter 7, *Audio Drivers*, followed by a presentation of the currently existing filters in Chapter 8, *Audio Driver Filters*.

Some examples on how to start audio drivers are shown in Chapter 9, *Configuration Examples*.

Chapter 10 shows how to control some aspects on audio using UiMessages, even if the program that is currently running doesn't have any audio controls.

The document ends with Chapter 11, *Latest Document Version Changes*, and Chapter 12, *Contact Information*.

2 Disclaimer

VLSI Solution makes everything it can to make this documentation as accurate as possible. However, no warranties or guarantees are given for the correctness of this documentation.

3 Definitions

DSP Digital Signal Processor.

I-mem Instruction Memory.

LSW Least Significant (16-bit) Word.

MSW Most Significant (16-bit) Word.

RISC Reduced Instruction Set Computer.

VS_DSP[4] VLSI Solution's DSP core.

VSIDE VLSI Solution's Integrated Development Environment.

VSOS VLSI Solution's Operating System.

X-mem X Data Memory.

Y-mem Y Data Memory.

4 Overview

The VSOS Audio Subsystem provides numerous drivers to handle the many audio Input/Output options of VS1005. The audio drivers can be controlled either with ioctl() calls from the C language, or from VSOS Shell control program.

While instructions for how to use each audio driver are provided in the README.TXT or documentation .PDF files of the drivers, this document will provide an overview of the capabilities of the drivers. However, for details, refer to documentation of the audio drivers themselves.

5 Requirements

To test the audio drivers in this document, you need to have the following building blocks:

- VS1005g Developer Board. The VS1005g BreakOut Board should also work, but these instructions have been tested with the DevBoard.
- Latest version of VSOS installed (at least v3.23, released 2015-09-04).
- USB cable between DevBoard and PC for uploading new software.
- If you want to use the VSOS Shell environment, you will also need:
  - UART or USB->UART cable connected between DevBoard and PC for using the UART interface. Data speed is 115200 bps, format is 8N1.
  - Your favorite UART Terminal Emulation program installed on the PC. Read the “VS1005 VSOS Shell” for further details.

When all of this is in order, you are ready to test the VSOS Audio Subystem.

6 The VS1005 VSOS Audio Subsystem

As a default, VSOS offers a simple audio output driver that lets the user output 16-bit mono or stereo audio to the VS1005 analog output pins LEFT and RIGHT, and control the sample rate.

VSOS Audio makes it very easy to produce sound with its standard-C-like standard audio interface (Chapter 6.1, *Standard Audio*). Instead of being forced to use audio-specific I/O routines, audio looks just like files.

More complex audio operations and redirections can be done using the audio drivers, described Chapter 7, *Audio Drivers*.

6.1 Standard Audio

VSOS offers the user a standard audio source and destination, although the audio source is only activated if an appropriate audio input driver is loaded (Chapter 7). Called *stdaudioin* and *stdaudioout*, standard audio file handles are to sound much like *stdin* and *stdout* are to standard input and output in standard C. It is not allowed for the user to close standard audio input or output files, but the user may modify their parameters.

By default, *stdaudioout* is connected to analog output pins LEFT and RIGHT, although this can be changed with appropriate audio drivers.

Both standard audio input and output open in stereo, 16-bit, 48 kHz mode. These parameters can be changed by the user, with driver and hardware dependent limitations.

The user may use all standard read and write operations to read from and write to standard audio. It is, however, required that fread() / fwrite() functions are used instead of character-based operations like fgetc() and fprintf(). It is also recommended to handle larger chunks of samples, like 32, at a time.

Stereo samples are stored in an interleaved fashion. In 32-bit mode, the least significant word is stored first. This is the same as the native VSDSP 32-bit word order.

Audio sample buffer 16-bit word order				
Audio format	**Word 0**	**Word 1**	**Word 2**	**Word 3**
16-bit stereo	Left 0	Right 0	Left 1	Right 1
32-bit stereo	Left 0 LSW	Left 0 MSW	Right 0 LSW	Right 0 MSW

6.2 VSOS Audio Example Program

The following audio program example creates a low-intensity sine wave to the left channel, then outputs the samples.

```
#include <vo_stdio.h>
#include <stdlib.h>
#include <math.h>
#include <saturate.h>
#include <apploader.h>

#define SIN_TAB_SIZE 96
#define SIN_AMPLITUDE 1000 /* Max 32767 */

static const s_int16 __y sinTab[SIN_TAB_SIZE];

int main(void) {
  // Remember to never allocate buffers from stack space. So, if you
  // allocate the space inside your function, never forget "static"!
  static s_int16 myBuf[2*SIN_TAB_SIZE];
  int i;

  /* Build sine table */
  for (i=0; i<SIN_TAB_SIZE; i++) {
    sinTab[i] = (s_int16)(sin(i*2.0*M_PI/SIN_TAB_SIZE)*SIN_AMPLITUDE);
  }

  while (1) {
    // Clear buffer
    memset(myBuf, 0, sizeof(myBuf));

    // Create sine wave to the left channel.
    for (i=0; i<SIN_TAB_SIZE; i++) {
      myBuf[i*2] = sinTab[i];
    }

    // Write result
    fwrite(myBuf, sizeof(s_int16), 2*SIN_TAB_SIZE, stdaudioout);
  }

  // Not really needed because there was a while(1) before
  return EXIT_SUCCESS;
}
```

6.3 VSOS Audio Controls

VSOS Audio has ioctl() controls declared in <aucommon.h>, that look like the the following example.

6.3.1 ioctl() Control Example: Setting Output Sample Rate

As an example, to set the sample rate, do the following. Notice that sample rate doesn't necessarily fit into 16 bits, so it needs to be passed as a pointer.

```
#include <vo_stdio.h>
#include <audiofs.h>
#include <devAudio.h>
FILE *fp;
s_int32 sampleRate = 48000;
ioresult ior;
ior = ioctl(fp, IOCTL_AUDIO_SET_ORATE, (void *)(&sampleRate));
```

There are much more definitions in the #include file <aucommon.h>. Refer to the documentation of the specific drivers you use for exact details on what functions they support.

7 Audio Drivers

VS1005g has multiple audio paths. This Chapter will explain which driver you will need to attach each audio driver to your software.

7.1 General

Audio drivers are named using the following format:
`AUdyyyyy.DL3`
where

Symbol	Description
d	Driver direction: I = input, O = output, X = Input+Output
yyyyy	Driver name, max. 5 characters

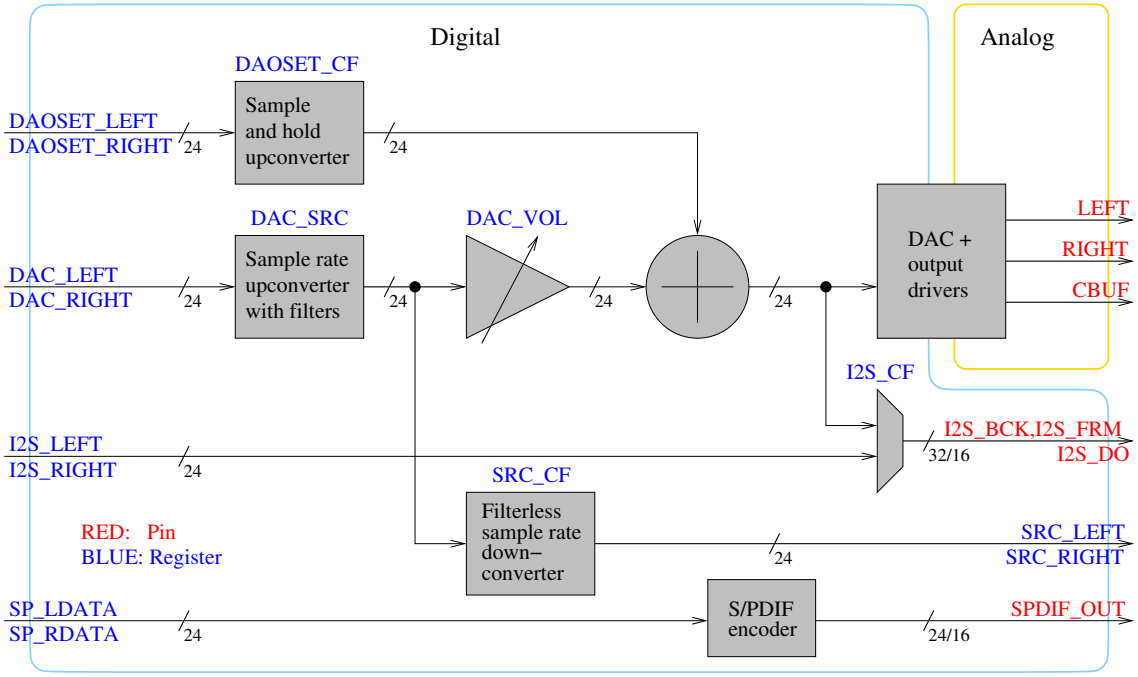

Figure 1: VS1005g playback (DA) audio paths

Figure 1 shows the VS1005 hardware output audio paths. Most of these have a driver controlling them.

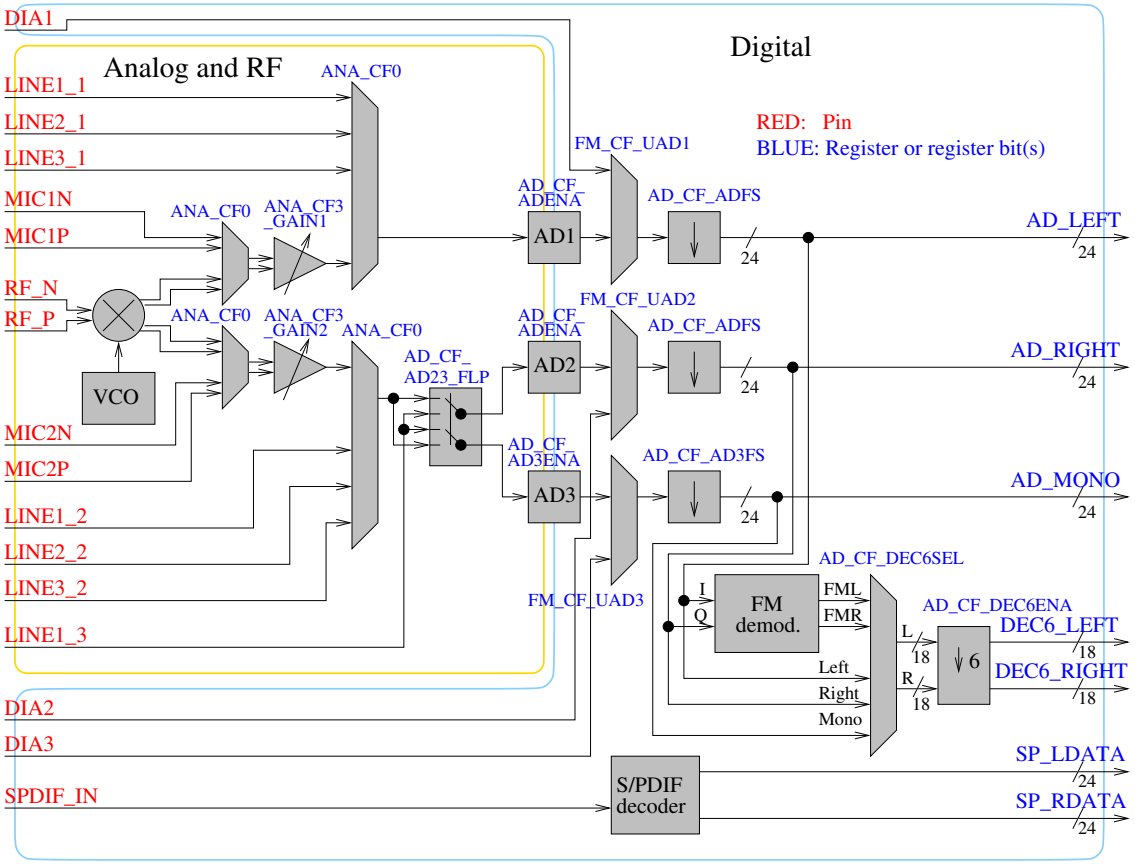

Figure 2: VS1005g recording (AD, FM, etc) signal paths

Figure 2 shows the VS1005 hardware input audio paths. Many of these have a driver controlling them.

7.2 Analog Output DAC Audio Drivers

7.2.1 Driver AUODAC.DL3

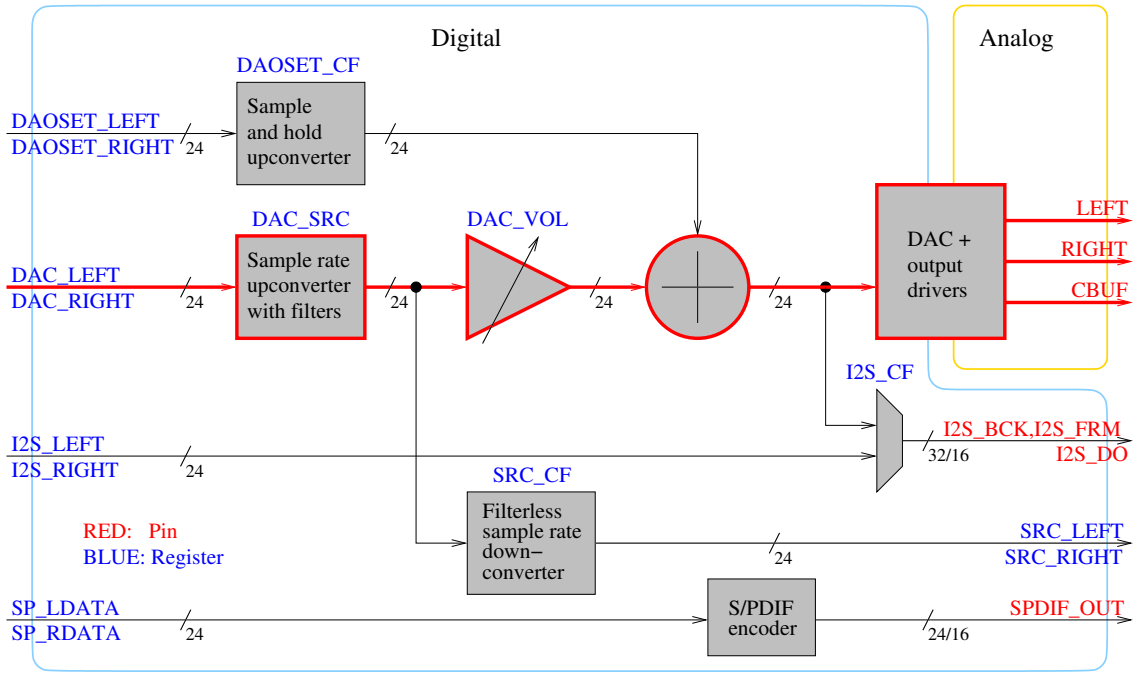

Figure 3: AUODAC.DL3 signal paths shown in **bold red**

Figure 3 shows the VS1005 high-quality, fully filtered analog output main audio path.

AUODAC.DL3 is the basic DAC output driver. It takes over the VSOS default driver and offers a lot of cuntionality over it, like 16-bit and 32-bit data transfers. It takes over stdaudioout, so all software that writes to standard output will send audio to this driver.

The driver offers setting the sample rate with 1 Hz accuracy upto 96000 Hz. Audio is upconverted to an extremely high rate of 6.144 MHz by a high-quality Sample rate up-converter.

Playback volume can be set with 0.5 dB accuracy between full level volume (-0 dB) and -127 dB.

7.3 Analog Side Path Audio Drivers

7.3.1 Driver AUOOSET.DL3

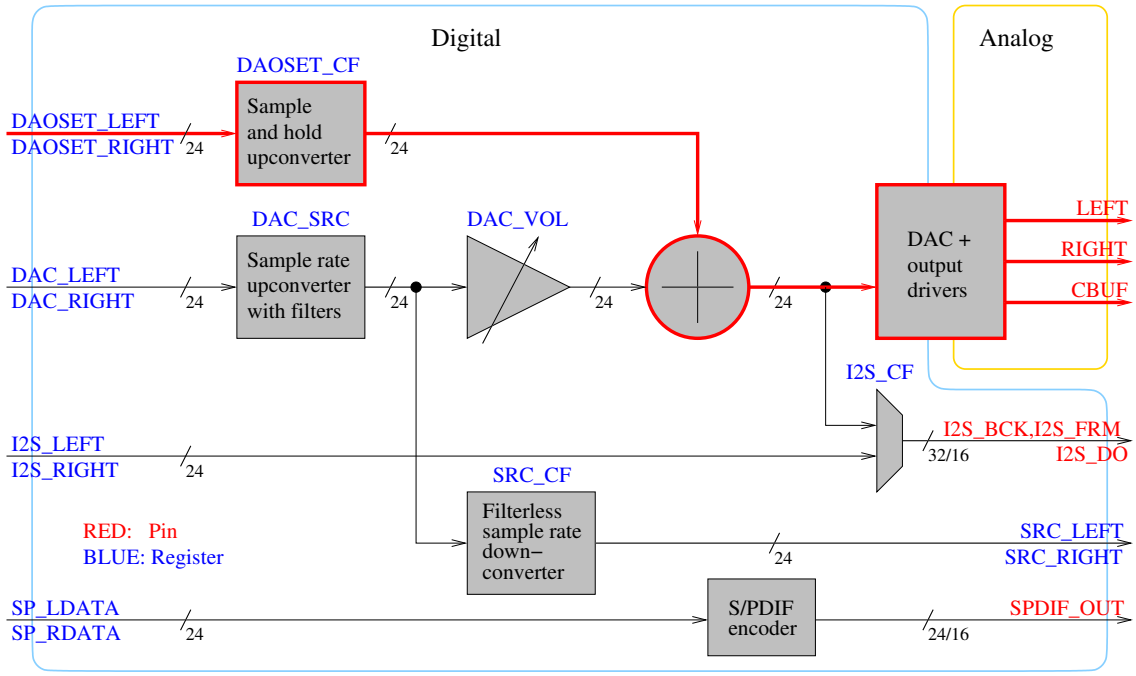

Figure 4: AUOOSET.DL3 signal paths shown in **bold red**

Figure 4 shows the VS1005 analog output audio side path. This audio path is not filtered; it is only put through a Sample and hold upconverter. As such, audible aliasing distortion may be heard if low sample rates are used. This audio path is best suitable for different kinds of alarm and effects sounds that may easily be overlayed on top of the audio of the main audio path (see Chapter 7.2.1).

While the sample rate of the side audio path may be set to upto 192 kHz, it is not always accurate. While certain sample rates like 24, 48, and 96 kHz can be represented accurately, some others, like 44.1 kHz, may have an upto 150 Hz error. While not a problem for effects sounds, this may make accurate timing difficult.

7.4 Analog Input ADC Audio Drivers

7.4.1 Driver AUIADC.DL3

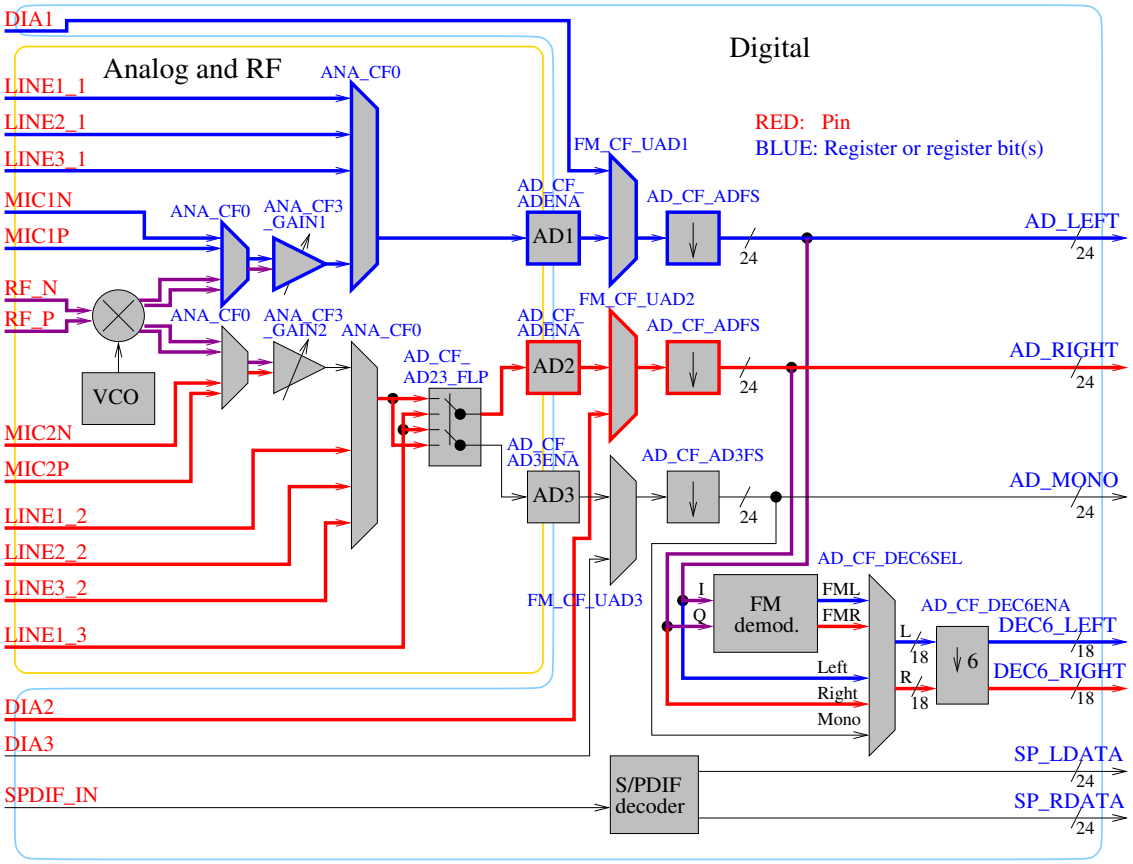

Figure 5: AUIADC.DL3 selectable input signal paths shown in **bold blue** for the left channel, and **bold red** for the right channel

The AUIADC.DL3 driver lets the user select a stereo input from a multitude of analog (and even some digital) sources. The sources used may be chosen at startup time, or changed dynamically while the driver is running. Any **red** source in Figure 5 may be combined with any **blue** source to form a stereo signal. However, if the **magenta-coloured** RF input is selected, it needs to take over the whole stereo audio path.

Supported sample rates are 192, 96, 48, and 24 kHz. However, it is also possible to use a high-quality down-by-6 decimator to create such sample rates as 32, 16, and 8 kHz. When the decimator is selected, the driver automatically reads it samples from the DEC6_LEFT/DEC6_RIGHT registers instead of the default AD_LEFT/AD_RIGHT ones.

Note that even if RF is selected for FM radio input, all of the FM hardware is not started by the driver. So you will still need a dedicated FM Receiver program to e.g. tune the FM radio. The only supported sample rate for the FM receiver is 32 kHz (using 192 kHz main sample rate and putting the signal through the I/Q - FM demodulation - down-by-6

decimator hardware).

Optinally, digital microphone inputs DIA1 and DIA2 may be used instead of the analog inputs. The 1-bit signals in the megahertz domain from the microphones are fed to the high-quality digital low-pass filtering path of VS1005.

On the VS1005 DevBoard, LINE1_1 and LINE1_3 are used as the default analog inputs. On the VS1005 BreakOut Board, LINE1_1 and LINE1_2 are used.

7.4.2 Control Program AUINPUT.DL3

```
Usage: AuInput [rate|chan|-h]
rate    sampleRate (Hz)
chan    Audio channel configuration (see below)
-h      Show this help

chan needs either one stereo element, or one left and one center/right element.
Stereo elements:
- fm
Left elements:
- line1_1, line2_1, line3_1, mic1, dia1
Right elements:
- line1_2, line2_2, line3_2, line1_3, mic2, dia2

Example:
auinput line1_1 line1_3
```

This programs lets the user control the inputs and sample rate of the analog audio input driver.

7.5 I2S Audio Drivers

I2S Audio drivers allows for I2S operation in both master and slave mode. Whenever possible, it is recommended to use master mode, because that way VS1005 has exact control over the sample rate.

The sample rate and number of bits (16/32) may be controlled with ioctl() commands IOCTL_AUDIO_SET_RATE_AND_BITS (recommended), IOCTL_AUDIO_SET_IRATE, and IOCTL_AUDIO_SET_ORATE. In master mode, sample rates 24, 48, 96, and 192 kHz are supported.

In slave mode, the other end selects the sample rate, which is the same for both I2S input and output. In this case VS1005 currently offers no support to synchronize the audio with e.g. its own DAC.

With the exception of AUOI2SMA.DL3, all drivers connect to stdaudioin and/or stdaudiout if started with parameter “s”. Otherwise, the drivers need to be opened and accessed manually.

7.5.1 Driver AUOI2SMA.DL3

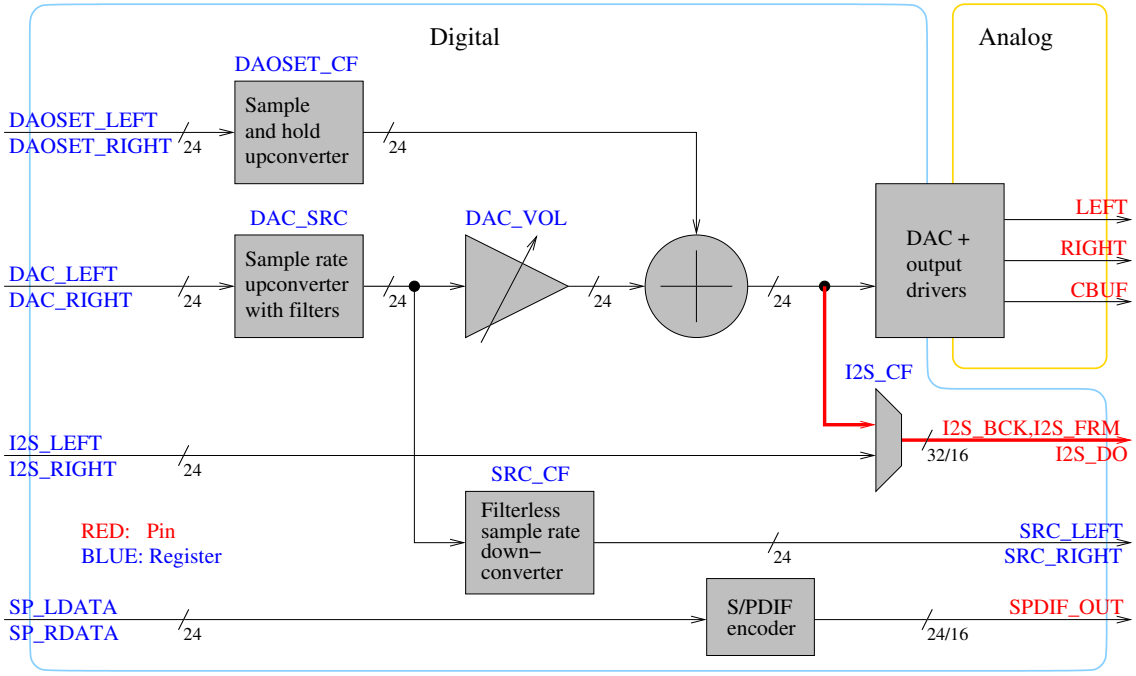

Figure 6: AUOI2SMA.DL3 audio path shown in **bold red**

Figure 6 shows the automatic audio path activated by the driver. The driver copies the sum of the DAC and DAOSET drivers, with volume applied to the DAC contents, and sends them to I2S. To function, it needs a DAC (e.g. AUODAC.DL3) and/or DAOSET (e.g. AUOOSET.DL3) driver to be installed.

The sample rate is set to a default of 96000 Hz / 32 bits. Anything played back through VS1005's analog audio path is converted to the target sample rate by VS1005 hardware.

If you need to send independent audio to I2S, using AUOI2SM.DL3 is required instead (Chapter 7.5.2). Note, however, that that driver can only support the basic master mode sample rates (e.g. not 44100 Hz).

7.5.2 Driver AUOI2SM.DL3

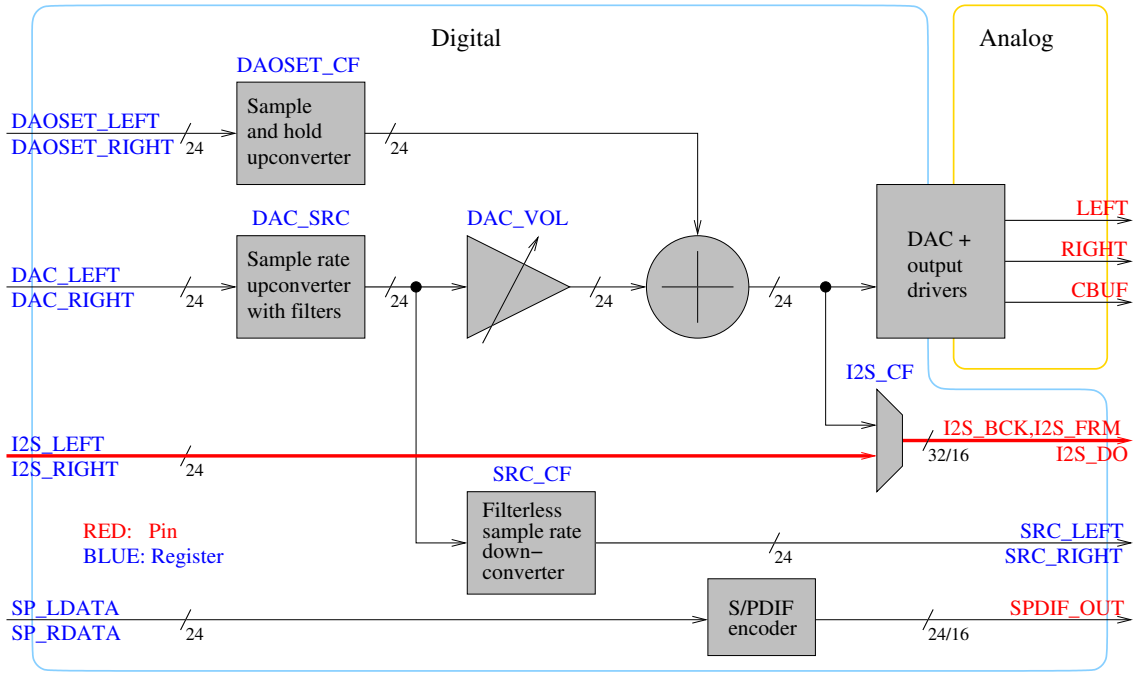

Figure 7: AUOI2SM.DL3 audio path shown in **bold red**

Figure 7 shows the manual audio path activated by the AUOI2SM.DL3 driver.

7.5.3 Driver AUOI2SS.DL3

The AUOI2SS.DL3 is otherwise similar to AUOI2SM.DL3 (Chapter 7.5.2), except that the driver operates in slave mode.

In slave mode the user has no control over sample rate, so the audio cannot be fed anywhere else except the I2S output without difficult resynchronization, which is not supported yet on VS1005.

7.5.4 Driver AUII2SM.DL3

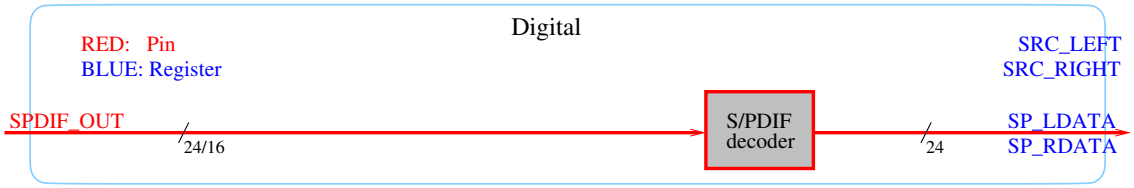

Figure 8: AUII2SM.DL3 audio path shown in **bold red**

Figure 8 shows the audio path activated by the AUII2SM.DL3 driver, which is a master mode input driver.

7.5.5 Driver AUII2SS.DL3

The AUII2SS.DL3 is otherwise similar to AUII2SM.DL3 (Chapter 7.5.4), except that the driver operates in slave mode.

In slave mode the user has no control over sample rate, so the audio cannot be fed anywhere else except the I2S output without difficult resynchronization, which is not supported yet on VS1005.

7.5.6 Driver AUXI2SM.DL3

The AUXi2SM.DL3 audio driver handles both I2S input and output in master mode, as shown in Figures 7 and 8.

The I2S input and output is always kept in sync, so software using both the input and output doesn't need to do synchronization. Also, because the exact I2S sample rates 24 and 48 kHz are directly supported by VS1005's analog audio output path, as well as the analon audio input path, once in sync they will stay in sync.

7.5.7 Driver AUXI2SS.DL3

The AUXI2SS.DL3 is otherwise similar to AUXI2SM.DL3 (Chapter 7.5.6), except that the driver operates in slave mode.

In slave mode the user has no control over sample rate, so the audio that has been input cannot directly be fed anywhere else in realtime, except to to the I2S output, which is always in sync with the input.

7.6 S/PDIF Audio Drivers

7.6.1 Driver AUOSPDA.DL3

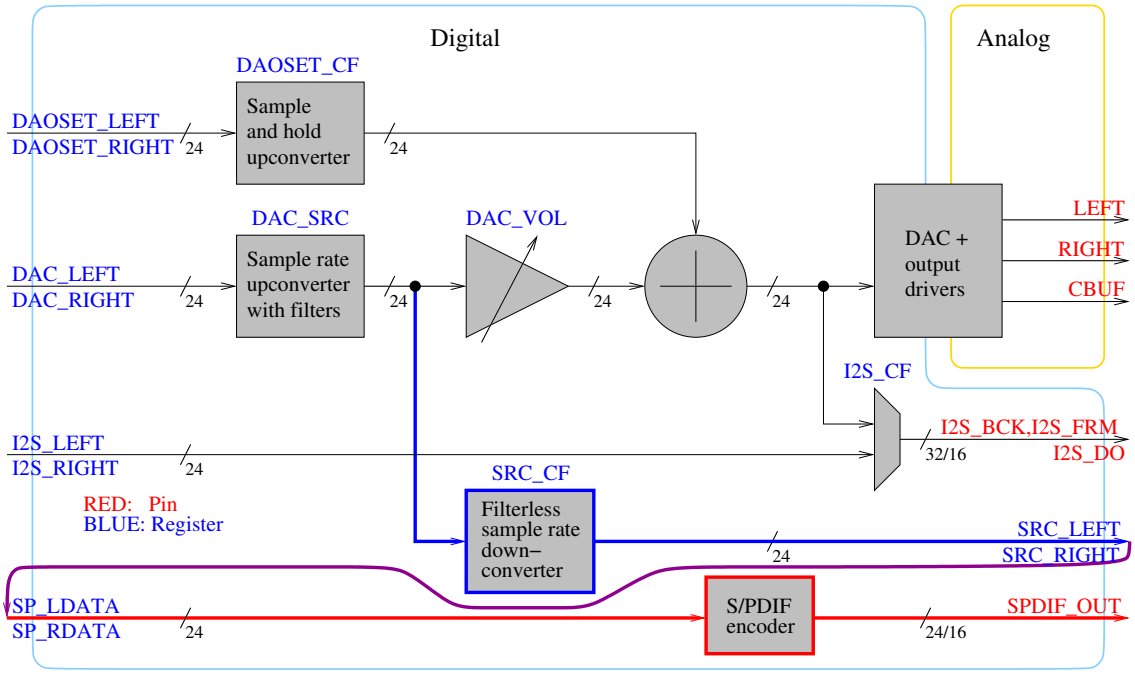

Figure 9: AUOSPDA.DL3 audio path shown in **bold red**. Software driver connecting to the filterless sample rate converter (**bold blue**) shown in **bold magenta**

Figure 9 shows the automatic audio path built by the driver. The driver copies the sum of the DAC drivers, and sends it to the S/PDIF output. To function, it needs a DAC (e.g. AUODAC.DL3) driver to be installed.

The sample rate is set by default to 96000 Hz, and anything played back is converted to the target sample rate by high-quality VS1005 Filterless sample rate downconverter, then copied by software to the SP_LDATA/SP_RDATA registers. As long as the audio that is being played back has a sample rate that is not higher than the S/PDIF output sample rate, no aliasing will occur, and sound quality will remain good.

The driver can be opened manually, in which case it has a separate volume control that can be used. As a default, full output volume is used. However, if the driver is started with parameter “v”, it will automatically copy any volume setting sent to stdaudioout.

8 Audio Filter Drivers

Audio filter drivers connect to an audio source or sink, and offer additional functionality, like filtering.

Audio filter drivers are named using the following format:
`FTdyyyyy.DL3`
where

Symbol	Description
d	Driver direction: I = input, O = output, X = Input/Output
yyyyy	Driver name, max 5 characters

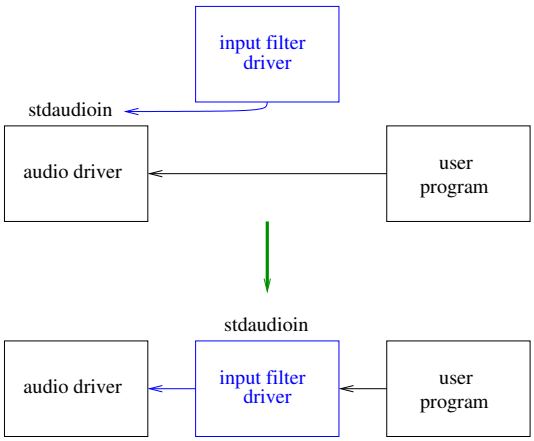

Figure 10: A filter input driver connects to the stdaudioin chain

All filter input drivers connect directly between the current stdaudioin program chain and the user program, as shown in Figure 10. It is important to first start the base driver responsible for stdaudioin before starting the filter driver!

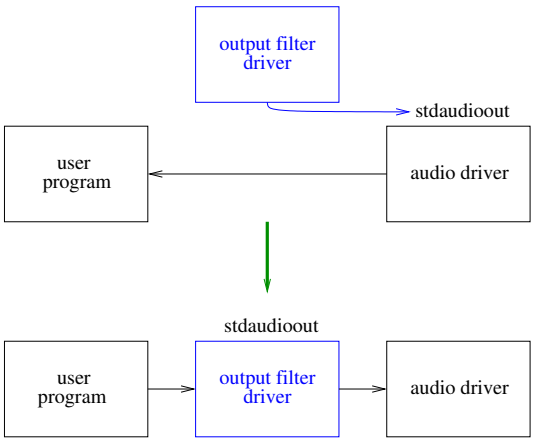

Figure 11: A filter output driver connects to the stdaudioout chain

All filter output drivers connect directly between the user program and the current stdaudioout program chain, as shown in Figure 11. It is important to first start the base driver

responsible for stdaudioout before starting the filter driver!

If the user wishes to remove an audio driver or some filters, they always have to removed in reverse order as they were allocated. E.g. if AUODAC.DL3 and FTOEQU.DL3 have been loaded, they must be released in order FTOEQU.DL3, AUODAC.DL3.

8.1 Equalizer Audio Drivers

The Equalizer audio drivers allow for multiband equalization to be implemented to the VS1005's output audio path.

The package itself contains detailed PDF documentation; please read that for details.

8.1.1 Driver FTOEQU.DL3

FTOEQU.DL3 connects the equalizer driver to stdaudioout. Read the PDF documentation for FTEQU for more details.

8.1.2 Control Program SETEQU.DL3

```
Usage: SetEqu [-i|-o] [n [flags centerF gain qFactor]] [-h]
-i      Set stdaudioin
-o      Set stdaudioout (default)
n       Use filter number n (1 ... MAX_FILTERS)
flags   1=left, 2=right
centerF Center frequency in Hz
gain    Gain in dB (-12.0 ... 12.0)
qFactor Q Factor (0.1 ... 4.0)
-h      Show this help

Examples
setequ 1 3 400 -6.0 0.5 # Set filter 1
setequ 1 0              # Clear filter 1, L+R, 400 Hz, -6 dB, Q 0.5
setequ 1                # Show filter 1
setequ                  # Show all filters
```

SETEQU.DL3 is a program to set and/or display the equalizer parameters.

Note that the equalizer is a relatively expensive function, so more than a bass/treble controller should only be used with care.

The full documentation for the software is in the PDF file for the FtEqu package.

8.2 DC Offset/AGC Audio Drivers

When audio is digitized, two technical issues are DC Offset and Large Dynamic Range.

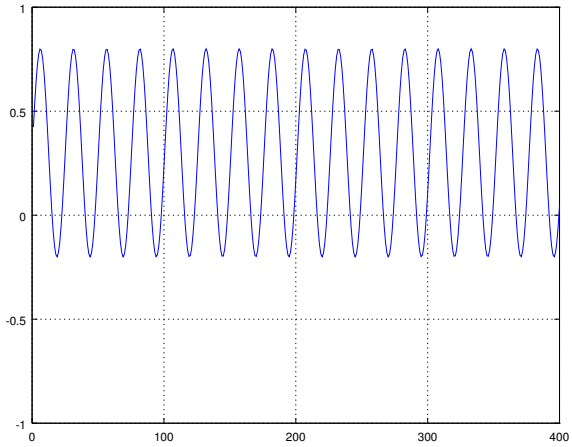

Figure 12: Audio with exaggerated DC offset

In an ideal world DC Offset wouldn't happen. However, in the real world, signals almost always have a slight DC offset. Note, how the sine wave in Figure 12) does not move evenly around the center point, but has an offset of about +0.35. While the figure has been greatly exaggerated, this is a real phenomenon caused by a myriad of different reasons.

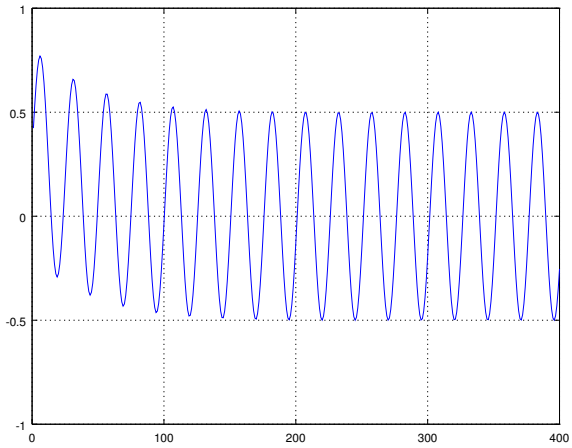

Figure 13: Audio with DC blocking

DC offset may cause many issues, including increased power consumption, audible cracks and pops, waring down of speaker elements, and non-ideal audio compression. Because of this, it is best to remove the audio offset with a DC Blocker algorithm, as shown in Figure 13. Notice how the offset disappear after a little while (in this case, it

has vanished practically completely by sample 150).

Another issue in audio is excessive dynamic range. This is not a problem when recording well-mixed, pre-recorded music, but it may be a big issue when recording speech from the microphone. To compensate for the audio level differences of close and faw away speakers, and Automatic Gain Control (AGC) unit may often be useful. Note, however, that AGC should not be used for HiFi recording applications!

8.2.1 Driver FTIDCBL.DL3

The DC Block driver FTIDCBL.DL3 connects to stdaudioin as shown in Figure 10.

The driver may be controlled either through C ioctl() function calls as described in the README.TXT file for the driver itself, or from the VSOS Shell using the SETAGC.DL3 command.

8.2.2 Driver FTIAGC.DL3

In addition to the DC Block driver described in 8.2.1, the AGC driver offers an Automatic Gain Control function.

The driver may be controlled either through C ioctl() function calls as described in the README.TXT file for the driver itself, or from the VSOS Shell using the SETAGC.DL3 command.

8.2.3 Control Program SETAGC.DL3

```
Usage: SetAgc [-i|-o] [-a x|-d x|-t x|-max x|-min x|-d x] [-h]
-i      Set stdaudioin (default)
-o      Set stdaudioout
-a x    Set attack (ms)
-d x    Set decay (ms)
-t x    Set target level (dB)
-max x  Set maximum gain (dB)
-min x  Set minimum gain (dB)
-b x    Set DC Block Filter (0x4000 = HiFi, 0x8000 = Speech,
                             0x0    = Auto, 0xC00x = Set to x)
-h      Show this help

With no parameters SetAgc will show current values
```

Sets/Prints AGC and/or DC Block Filter parameters. For the DC Block Filters only the -b setting option is available.

9 Configuration Examples

Here are some configuration examples for starting different audio drivers.

For full options for each of these programs, have a look at the README.TXT / PDF file for each of the drivers.

9.1 Minimal config.sys for Playback

```
# New 2015 audio DAC out driver
AUODAC s
```

9.2 config.sys for Playback with Bass/Treble Controls and I2S + S/PDIF Outputs

```
# New 2015 audio DAC out driver
AUODAC s
# I2S automatic out; automatic is hardware, so doesn't require CPU
AUOI2SMA
# S/PDIF automatic out, parameter can be either 48000 or 96000 (default)
# If "v" is defined, stdaudioout volume control is copied to S/PDIF,
# otherwise it needs to be controlled manually (otherwise it stays at
# maximum level)
AUOSPDA 96000 v
# Equalizer, set 100 and 10000 Hz to +6 dB with Q Factor 0.7
FTOEQU
run setequ 1 3 100 +6 0.7
run setequ 2 3 10000 -6 0.7
```

9.3 Basic config.sys for Recording

```
# New 2015 audio DAC out driver
AUODAC s
# New 2015 audio ADC in driver
AUIADC s 48000 line1_1 line1_3
# DC Block; use at least this with analog input even if not using AGC
FTIDCBL
```

9.4 Versatile config.sys for Recording with AGC and I2S + S/PDIF Outputs

```
# New 2015 audio DAC out driver
AUODAC s
# New 2015 audio ADC in driver
AUIADC s 48000 line1_1 line1_3
# I2S automatic out; automatic is hardware, so doesn't require CPU
AUOI2SMA
# S/PDIF automatic out, parameter can be either 48000 or 96000 (default)
# If "v" is defined, stdaudioout volume control is copied to S/PDIF,
# otherwise it needs to be controlled manually (otherwise it stays at
# maximum level)
AUOSPDA 96000 v
# DC Block + AGC unit to stdaudioin
FTIAGC
```

10 Controlling Audio from VSOS Shell with UiMessages

When using the VSOS Shell, some audio functions may be controlled even if running a VSOS program that doesn't take audio controls. If the TTY is not in RAW mode, the following escape sequences defined in <uimessages.h> may be sent to the shell.

10.1 Setting Volume anywhere from VSOS Shell

Note that <A> here means sending ASCII code 1, invoked in most terminal emulation programs by pushing Ctrl-A.

Volume up by 1/2 dB:
<A>111ms

Volume down by 1/2 dB:
<A>111ms

Set attenuation to -HH/2 dB, where HH is a hexadecimal number:
<A>206mHHs

Example:
To set volume to -20 dB, you need to send 40 = 0x28:
<A>206m28s

10.2 Sending equalizer controls from VSOS Shell

The filters are accessed with UiMessages that have the following format, where X is the filter number (0..f), and YY is the 16-bit signed value presented as an unsigned 16-bit hexadecimal number. <A>21XmYYs

Example:
Let's assume we have the following configuration lines in config.txt:

```
run setequ 1 3 100 0 0.7
run setequ 2 3 10000 0 0.7
```

Now, to set bass (filter channel 1) to +6 dB (6), send the following command:
<A>210m6s

To set treble (filter channel 2) to -12 dB (0xfff4), send the following command:
<A>211mfff4s

Upto 16 channels may be accesses with messages ranging from 0x210 to 0x21f.

11 Latest Document Version Changes

This chapter describes the most important changes to this document.

Version 1.00, 2015-09-04

First release.

12 Contact Information

VLSI Solution Oy
Entrance G, 2nd floor
Hermiankatu 8
FI-33720 Tampere
FINLAND

URL: http://www.vlsi.fi/
Phone: +358-50-462-3200
Commercial e-mail: sales@vlsi.fi

For technical support or suggestions regarding this document, please participate at
http://www.vsdsp-forum.com/
For confidential technical discussions, contact
support@vlsi.fi