VS1005 VSOS SHELL

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**
3.54	2018-02-08	HH	Removed obsolete installation information.
3.53	2018-01-31	HH	Updates for VSOS 3.53.
3.52	2018-01-22	HH	Updates for VSOS 3.52.
3.42	2017-05-18	HH	Added Format and FatInfo, other updates.
3.40	2016-11-03	HH	Updates for VSOS 3.40.
3.30	2016-06-22	HH	Updates for VSOS 3.30.
1.50	2016-03-18	HH	Updates for VSOS 3.27.
1.40	2016-02-15	HH	Added new programs.
1.30	2015-10-02	HH	Added many new programs.
1.20	2015-07-17	HH	Added Rec - an audio recorder.
1.11	2015-06-01	HH	PlayDir developed further.
1.10	2015-05-29	HH	Initial release of documentation.

Contents

List of Figures

1 Introduction

The VS1005 VSOS Shell is a powerful tool that allows controlling VS1005 from an external interface, e.g. the UART.

This document will explain how to install and use the VSOS Shell Environment, as well as the UART Controlled Audio Player that is included in this package.

After the disclaimer and definitions in Chapters 2 and 3, an overview of the Shell is given in Chapter 4, *Overview*. It is followed by requirements in Chapter 5, *Requirements*, and shell installation instructions in Chapter 6, *Installing the VSOS Shell Environment*.

Instruction on using the shell is given in Chapter 7, *Using the Shell Environment*.

The UART Player PlayDir is explained in Chapter 8, *Using the UART Controlled Player PlayDir*, followed by the UART Recorder Rec in Chapter 9, *Using the UART Controlled Recorder Rec*.

The document ends with Chapter 10, *Latest Document Version Changes*, and Chapter 11, *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

CBR Constant BitRate. Bitrate of stream will be the same for each compression block.

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.

VBR Variable BitRate. Bitrate will vary depending on the complexity of the source material.

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 VS1005 VSOS Shell is a new, powerful tool that allows controlling VS1005 from an external interface, e.g. the UART.

Using the VSOS Shell interface it is possible to create an audio player and recorder system (recorder not yet available) that doesn't require any specific VS1005 programming skills.

Nevertheless, for those who are familiar with programming VS1005, the VSOS Shell Environment makes it possible to create convenient shell commands / programs. By combining these building blocks it is possible to create more complex systems, controlled either by an external microcontroller or VS1005 itself.

5 Requirements

Before using the VSOS Shell Environment, you need to have the following building blocks:

- VS1005g Developer Board. The VS1005g BreakOut Board should work, too, but these instructions have been tested with the DevBoard.
- Latest version of VSOS installed (VSOS 3.54 or higher recommended).
- USB cable between DevBoard and PC for uploading new software.
- 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. Note that the current VSOS Shell uses only the line feed character (0x0a) for line feed, and no carriage return (0x0d), so the terminal emulation program needs to be able to handle that (see examples in Chapter 5.1, *Terminal Program Settings*). For Microsoft Windows computers, PuTTY and TeraTerm have been tested and found working.

When all of this is in order, you are ready to install the VSOS Shell Environment.

5.1 Terminal Program Settings

This chapter shows recommended settings for two example terminal programs, tested by VLSI. Other terminal programs can also be used, provided that the crucial parameters are set in a similar way.

5.1.1 Settings for PuTTY

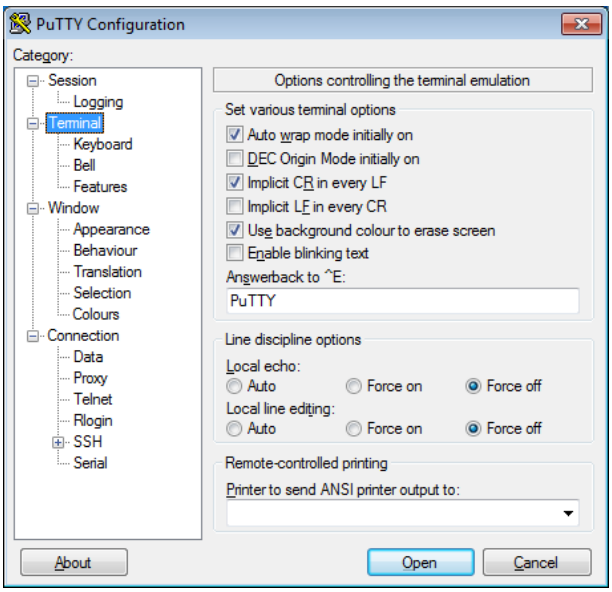

Figure 1: PuTTY Configuration: Terminal

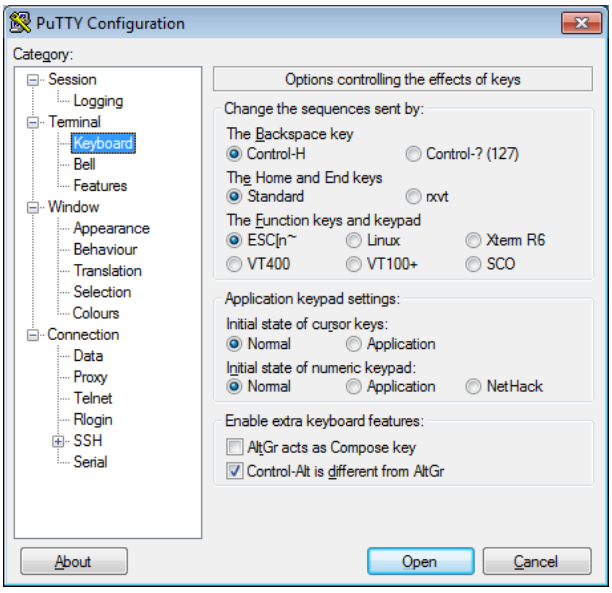

Figure 2: PuTTY Configuration: Keyboard

Figure 1 shows the terminal emulation settings for PuTTY. Make sure you check the “Implicit CR in every LF” box. In the keyboard settings in Figure 2, you may set Backspace key to either Control-H or Control-?.

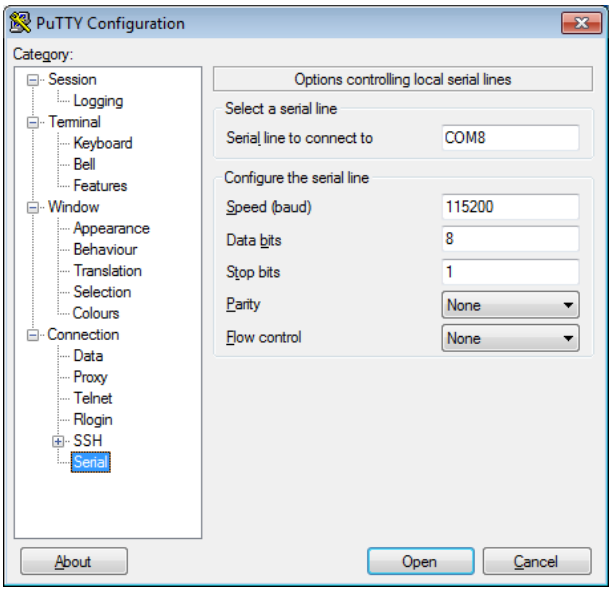

Figure 3: PuTTY Configuration: Serial

Figure 3 shows the serial communication parameters for 115200 bps, 8N1. For binary file transfers to work, it is important to disable flow control.

5.1.2 Settings for Tera Term

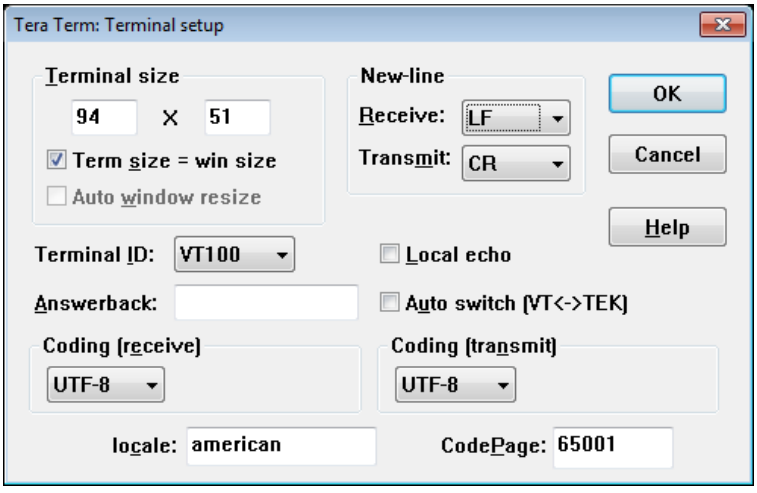

Figure 4: Tera Term: Terminal Setup

Figure 4 shows how to set line feeds and basic terminal emulation mode (VT100) in Tera Term.

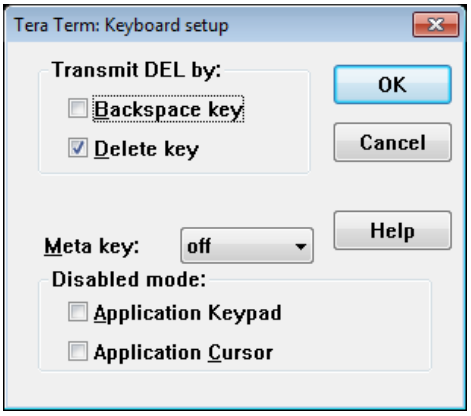

Figure 5: Tera Term: Keyboard Setup

Figure 5 shows a working keyboard setup for Tera Term.

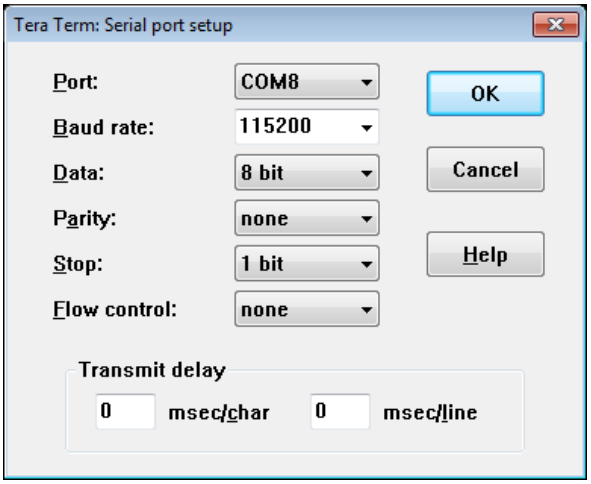

Figure 6: Tera Term: Serial Port Setup

Figure 6 shows how to set serial parameters 115200 bps, 8N1, without flow control.

6 Installing the VSOS Shell Environment

The VSOS Shell Environment comes preinstalled in all VLSI Solution's VS1005 Developer Boards since 2016.

If you have an older version of VSOS (3.22 or older) that didn't come with the VSOS Shell installed, update the VSOS Kernel and system root files using information from this thread on VLSI Solution's VSDSP Forum:
http://www.vsdsp-forum.com/phpbb/viewtopic.php?f=13&t=680

When you start a Developer Board into the VSOS Shell environment, you should see output like the following on your PC's terminal screen:

```
Hello.
VSOS 3.22 build Mar 10 2015 15:51:56
VLSI Solution Oy 2012-2015 - www.vlsi.fi

Starting the kernel..
Starting Devices...
External SPI Flash

Installed system devices:
S: SPI Flash c814, handled by FAT.
Load drivers, config 0...
Driver: RUN... FC00,D Y:0xfc00: 0x2000-13 -> 0x0
Driver: RUN... FC00,C Y:0xfc00: 0x0-12 -> 0x0
Driver: SDSD... D: SD card in SD mode

Driver: UARTIN...
Driver: S:SHELL.AP3...
VSOS SHELL
S:>
```

If this is the case, you have successfully started the VSOS Shell Environment.

On some boards and software configurations, you may have to push S2 (connect VS1005 pin GPIO0_2 to IOVDD using a 10k resistor) while booting.

7 Using the Shell Environment

The shell environment have the following features:

- Very small (roughly 1/4 kilowords code size).
- Executes commands from the S:SYS/ folder.
- Can also execute .dl3 and .ap3 programs from other locations if full path is provided.
- Interactive command line editing.
- Command history:
  - Up/down arrow (vt100 terminal).
  - “history” shows command history.
  - “!!” repeats last command.
  - “!-n” repeats n’th last command. E.g. “!-1” is equivalent to “!!”.
  - “!xx” repeats last command that started with xx.
- Concept of current directory.
- Buffered, interrupt-handled UART stdio driver.
- Ctrl-C to notify programs that they should close.
- Ctrl-C three times to hard reset VS1005.
- Interactive (echo +e) or machine-controlled silent (echo -e) mode.

If the shell is in interactive mode, it will present a command prompt.

If in non-interactive mode, the shell will display a line containing the hash character “#” when expecting input from the user. After receiving the full command line, it will output a line containing “:” before executing the command.

This package contains several programs for the shell, many of which take command line arguments.

The shell commands can also be accessed from the C environment through the function RunProgram() in <apploader.h>.

Example:

```
ioresult errorCode = RunProgram("dir", "-st -r")
```

is equivalent to typing

```
S:>dir -st -r
```

in the VSOS Shell command line. Usually programs return S_OK for a successful run, and some other value if there were problems.

The shell environment also allows dynamic loading and unloading of system drivers with the command DRIVER (Chapter 7.1.20). This is very useful in a memory-limited system with lots of functionality.

7.1 Shell Command Programs and VSOS Libraries

This section presents some of the shell programs and drivers included in this package. They are located in the VSOS system disk's directory S:SYS/ .

The program are in rapid development, so this documentation will necessarily be somewhat out of date most of the time.

To get more information on how to run these programs / use these libraries:

- If the file is a shell command and not a driver, try running it with the "-h" command line option.
- Some shell commands give help if run without any parameters.
- Source code for most of the shell commands / libraries is available in the VSOS 3.26 Root And Libraries Source Code package (or its current version), in the directory *solutions*. Many of the solutions contain contain README.TXT files that contain very useful documentation beyond the scope of this document.

Many programs offer a "-h" command line option to show what options they can handle. Also running a program without any parameters may give useful information.

7.1.1 audio

Old and obsoleted analog input/output audio driver. Do not use.

For information on the current audio drivers that you should use, read the separate document *VSOS Audio Subsystem*.

7.1.2 audiodec

Audio decoder main library. For details on how to use this library, see for instance the source code for *PlayDir*, *MP3 Model*, or the *Simple MP3 Player*.

See also:
audiodecdebug (Chapter 7.1.3), audiodecsmall (Chapter 7.1.4), dec* (Chapter 7.1.15), enc* (Chapter 7.1.24), PlayDir (Chapter 7.1.53).

7.1.3 audiodecdebug

Replacement driver for audiodec, but includes debug outputs approximately once a second if there are underflows in the output buffer, i.e. if playback is too slow.

Example output:

```
Underflows at 28.806s = 1347652, grown 24647
```

```
Underflows at 29.812s = 1372452, grown 24800
Underflows at 30.819s = 1397269, grown 24817
Underflows at 31.826s = 1422082, grown 24813
```

See also:
audiodec (Chapter 7.1.2), audiodecsmall (Chapter 7.1.4).

7.1.4 audiodecsmall

Replacement driver for audiodec, but missing some features and squeezed to take as little memory as possible.

See also:
audiodec (Chapter 7.1.2), audiodecdebug (Chapter 7.1.3).

7.1.5 aui*, auo*, aux*

Audio drivers. For details, read the separate document *VSOS Audio Subsystem*.

7.1.6 AuInfo

```
Usage: AuInfo [-f|+f|-v|+v|-p x] [-h] [files]
-f|+f   Fast mode on/off (don't determine MP2/MP3 file playback time)
-v|+v   Verbose mode on/off
-p x    Copy audioInfo data to x
-h      Show this help
```

AuInfo provides basic information of an audio file, like in the following example:

```
D:>auinfo island.flac
island.flac:
     size: 42159.9 KiB
   format: FLAC
     conf: 2 channels at 44100 Hz
     time: 6:47.1 seconds
  bitrate: 848.4 kbit/s
  analyze: 0.00 seconds
```

7.1.7 AuInput

Controls stdaudioin or other audio inputs. For details, read the separate document *VSOS Audio Subsystem*.

7.1.8 AuOutput

Controls stdaudioout or other audio outputs. For details, read the separate document *VSOS Audio Subsystem*.

7.1.9 cd

```
Usage: cd [-v|+v] [-h] dir
-v|+v   Turn verbose on/off
-h      Show this help
Note: cd without parameters shows the current directory
      cd :: lists all available devices
```

Change or print current directory or list all devices.

7.1.10 Cmp

```
Usage: Cmp [-h] [f1 f2]
f1      File 1
f2      File 2
-h      Show this help
```

Compares two files. If the files are similar, cmp prints nothing. If the files are different, cmp can print one of the two following messages:

cmp: "ro" and "ro2" differ: byte 25083
Files ro and ro2 was similar for the first 25082 bytes, but the next byte differs.

cmp: EOF on "ro2": byte 25085
The two files to compare were similar for the first 25085 bytes, but then ro2 ended.

7.1.11 CopyF

```
Usage: CopyF [-h] [src dst]
src     File to be copied
dst     Destination file
-h      Show this help
```

Copy a file. Long file names are lost when copying.

7.1.12 CopyR

```
Usage: CopyR [-v|-h] [src dst]
src     Directory to be copied
dst     Destination directory
-v      Be verbose
-h      Show this help
```

Make a recursive copy of a directory. Long file names are lost when copying.

7.1.13 Crc32

```
Usage: Crc32 [-v|+v|-i|+i|-h] file1 [...]
-v|+v   Verbose on|off
-i|+i   Ignore CRC32 calculation on|off
-h      Show this help
```

Print size and CRC32 checksum for a file.

Can be used as a read speed test with the “-i” option.

7.1.14 Date

```
Usage: Date [formatString] [-h]
-t      Print uptime timer
-q      Quiet mode
-h      Show this help

FormatString:
%a wdayName   %A weekdayName   %b monName    %b monthName
%c date&time  %d dayOfMonth    %H hour-24    %h hour-12
%j dayOfYear  %m month         %M minute     %p AM/PM
%S second     %w weekDay,S=0   %x date       %X time
%y yr         %Y year          %Z UTC        %% %
```

Displays the current date, either in default format, or with a user-given format string.

An LR44 battery or similar power source must be connected to the RTC.

See also:
SetDate (Chapter 7.1.71), RtcRead (Chapter 7.1.63).

7.1.15 dec*

Decoder libraries for different compressed audio formats used by the audiodec library.

See also:
audiodec (Chapter 7.1.2), enc* (Chapter 7.1.24).

7.1.16 del

```
Usage: Del [-v|+v|-h] file|dir
-v|+v   Verbose on/off
-h      Show this help
```

Deletes a file or recursively a directory.

7.1.17 delay

```
Usage: Delay ms
```

Delay waits for a given amount of milliseconds, then exits.

7.1.18 Dir

```
Usage: dir [-s|-sn|-st|-ss|+s|-r|+r|-d|+d|-a|+a|-f|+f|-h] [path]
-s|-sn  Sort files by name (default)
-st     Sort files by date
-ss     Sort files by size
+s      Don't sort files (faster with large directories)
-r      Reverse sort (if sort selected)
+r      Forwards sort (if sort selected)
-d      Show file date
+d      Don't show file date
-a      Only audio files
+a      All files
-f      Fast listing (don't show play time for MP3 files with -a)
+f      Slower listing (show play time also for MP3 files with -a)
-v      Verbose on
+v      Verbose off
-h      Show this help page
```

dir lists the contents of a directory. It can optionally examine each file and list only those ones that are audio media files.

If “-sn” is defined but the directory is too large to be sorted in memory, dir lists the files in file system order.

The output of dir is as e.g. as follows:

```
D:ManMachine/>dir +s
-    3. 03_MET~1.WAV   63755708 2015-07-17 11:38:16 03_Metropolis.wav
-    4. 04_THE~1.WAV   39508940 2015-07-17 11:38:26 04_TheModel.wav
-    5. 05_NEO~1.WAV   94268204 2015-07-17 11:38:36 05_NeonLights.wav
-    6. 06_THE~1.WAV   58616588 2015-07-17 11:38:44 06_TheManMachine.wav
-    7. 01_THE~1.WAV   65656124 2015-07-17 11:37:58 01_TheRobots.wav
-    8. 02_SPA~1.WAV   62633804 2015-07-17 11:38:08 02_Spaceleb.wav
```

The fields are as follows:

1. Entry type. ’D’ is a directory, ’-’ is a normal file, and ’L’ is a volume label.
2. The file system order of the entry
3. Short name
4. File size in bytes
5. File date in YYYY-MM-DD format
6. File time in hh:mm:ss format
7. Long name

When listing audio files, the output format is different:

```
D:ManMachine/>dir -a
-    7. 01_THE~1.WAV    6:12.2  44100 2 2015-07-17 11:37:58 01_TheRobots.wav
-    8. 02_SPA~1.WAV    5:55.1  44100 2 2015-07-17 11:38:08 02_Spaceleb.wav
-    3. 03_MET~1.WAV    6:01.4  44100 2 2015-07-17 11:38:16 03_Metropolis.wav
-    4. 04_THE~1.WAV    3:44.0  44100 2 2015-07-17 11:38:26 04_TheModel.wav
-    5. 05_NEO~1.WAV    8:54.4  44100 2 2015-07-17 11:38:36 05_NeonLights.wav
-    6. 06_THE~1.WAV    5:32.3  44100 2 2015-07-17 11:38:44 06_TheManMachine.wav
```

Now the fields are as follows:

1. Entry type. ’D’ is a directory, ’-’ is a normal file, and ’L’ is a volume label.
2. The file system order of the entry
3. Short name
4. File playback time in m:ss.t format
5. Sample rate
6. Number of audio channels
7. File date in YYYY-MM-DD format
8. File time in hh:mm:ss format
9. Long name

Determining playback times for MP3 files requires scanning the whole file, and may be slow. Because of this, unless +f is defined with -a, MP3 file playback times are shown as 0:00.0.

With the -v option, Dir prints more information about each file, including its cluster chains. It also checks if the cluster chain length is consistent with file length. An example of a very fragmented file below:

```
S:>dir s:sys
[...]
-   37. HIRESREC.DL3      67620 2018-01-31 09:23:06 HiResRec.dl3
        First block 0x150,  Cluster list: 0x29, 0x34, 0x68, 0x6a, 0x8a-0x96
        Total cluster chain 0x11, file size 0x11, OK
[...]
```

7.1.19 DiskFree

```
Usage: DiskFree [-p x|-m x|-v|-h] [D:]
D:      Print results for drive D (:: means all drives)
        There may be multiple drive parameters.
        Drive parameters must be after -p/-m options.
-p x    Store last result in KiB to X-memory 32-bit pointer pointed to by x
        (Usable for calling with RunLibraryFunction(); makes operation silent)
-m x    Look only until x MiB (faster operation)
-v      Verbose (giving twice makes even more verbose)
-h      Show this help
```

Displays the amount of free space on a drive, or, with the verbose option, also disk size and its fill state. By giving the verbose option twice, also print a map of allocated clusters of a disk.

Because of the way FAT allocation tables work, running DiskFree on a larger drive may take significant amounts of time; e.g. analyzing a 4 GiB SD card can take up to a second. To make DiskFree run faster, option -m may be used.

Example:

```
S:>diskfree -v -v d:
+.......+############################################################
############################################################################
############################+.................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
.............................................................
Drv  Used/KiB   Free/KiB  Total/KiB  Use%  Name
D:   36527776  217062752  253590528   14%  SD + VS23S040
```

See also:

FatInfo (Chapter 7.1.25).

7.1.20 driver

```
Usage: DRIVER +libname|-libname|?libname
```

Can be used to load, unload, and list libraries in memory.

Examples:

- `driver +auiadc 48000 line1_1 line1_3` loads the AUIADC.DL3 driver to memory (if not already there), and gives it parameters.
- `driver -auiadc` unloads the AUIADC.DL3 driver from memory if no other programs are accessing it. Only the beginning of the driver name needs to be written.
- `driver ?auiadc` checks whether the driver is currently in memory.

See also:
liblist (Chapter 7.1.41), liblist2 (Chapter 7.1.43).

7.1.21 DumpFlash

DumpFlash is a metasolution that allows the user to dump the contents of either the 1 MiB internal or 2 MiB external flash memory for later duplication.

For details, open the solution and read the README.TXT.

See also:
ProgramFlash (Chapter 7.1.58).

7.1.22 echo

```
Usage: echo [-n|+n|-e|+e|-|-h] string
-n      No newline
+n      Output newline
-e      Turn shell interactive echo mode off
+e      Turn shell interactive echo mode on
-       End of parameters
-h      Show this help
Note: String may contain escape codes such as \" and \n
```

Echoes its input on the screen.

Options “-e” and “+e” switch the global interactive mode off and on, respectively. The non-interactive mode may be easier to handle when the shell is used with a microcontroller.

7.1.23 edit

```
Usage: Edit [-r rows] [-c columns] [-h] fileName
-h      Show this help
```

Edit is a very simple, VT100-compatible full-screen text editor. It is mainly intended to edit short configuration files. It will *not* preserve CR/LF combinations or any binary codes, so never try to modify a binary file with it.

For a full listing of Edit's capabilities, start it, then push Ctrl-P.

When Edit starts, it reads the configuration file S:SYS/EDIT.INI. By changing the definitions there, you may change the way backspace and delete keys work.

7.1.24 enc*

Audio encoder drivers for various compressed formats. For details, read the separate document *VSOS Audio Subsystem*.

On how to use these libraries, see for instance the source code for *Rec*.

See also:
audiodec (Chapter 7.1.2), dec* (Chapter 7.1.15), Rec (Chapter 7.1.62).

7.1.25 FatInfo

```
Usage: FatInfo x: [blkNum] [-v|+v|-h]
x:      Drive name
blkNum  Print contents of a block
-v|+v   Verbose on/off
-h      Show this help
```

Provides information of a FAT32 file system.

```
S:>fatinfo d:
FAT FOUND IN ROOT BLOCK
  FAT32 SECTOR 0:
    Jump Code + NOP:  eb 58 90
    OEM Name:  76 6c 73 69 64 69 73 6b "vlsidisk"
    Bytes Per Sector 512
    Sectors Per Cluster 64
    Reserved Sectors 3624
    Number Of FATs 1
    16-bit Number of Sectors per FAT 0
    Media Descriptor (0xf8 for HDs) 0xf8
    16-bit FAT size 0
```

```
  Sectors Per Track 61
  Number of Heads 31
  Number of Hidden Sectors in Partition 0
  32-bit Number of Sectors in Partition 507246592
  32-bit Number of Sectors per FAT 61912
  Flags 0x0
  Version of FAT32 Drive 0x0
  Cluster Number of the Start of the Root Directory 2
  Sector Number of the FileSystem Information Sector 1
  Sector Number of the Backup Boot Sector 6
  Logical Drive Number of Partition 128
  Extended Signature (0x29) 0x29
  Serial Number of Partition 0x96895f6b
  Volume name of Partition: 56 53 4f 53 20 20 20 20 20 20 20 "VSOS       "
  Boot Record Signature (0x55 0xaa) 0x55 0xaa
FAT32 SECTOR 1:
  First Signature (0x52 0x52 0x61 0x41) 0x52 0x52 0x61 0x41
  Signature of FSInfo Sector (0x72 0x72 0x41 0x61) 0x72 0x72 0x41 0x61
  Number of Free Clusters 0xffffffff
  Most Recently Allocated Cluster 0x00000002
  Boot Record Signature (0x55 0xaa) 0x55 0xaa
fatStart 3624, fatSize 61912, rootStart 65536
FAT ROOT SECTOR 0x10000 (65536)
```

See also:
DiskFree (Chapter 7.1.19).

7.1.26 FILEBUF

Driver that creates a RAM buffer for writing files. Required by the Rec application to be able to smoothly record to SD cards.

See also:
Rec (Chapter 7.1.62).

7.1.27 Format

```
Usage: Format x: [-v|+v|-llabel|-sx|-cx|-fx|-ix|-p|-px|-F|-y|+y|-h]
x:      Drive name
-v|+v   Verbose on/off
-llabel Set disk label
-sx     Force size to x MiB
-cx     Force cluster size to x 512-byte sectors
-fx     Set number of FATs (1 or 2)
-ix     Set 32-bit serial number volume ID to x
-p|+p   Make/don't make partition table
-px     Reserve x MiB for partition table
```

```
-F|+F   Force / don't force making file system even if illegal
-y|+y   Don't ask / Ask for confirmation
-n|+n   Dry run (don't actually write) on/off
-a|+a   Erase (very slow) / Don't erase all data
-h      Show this help
```

Formats a FAT32 file system in a way that it is optimized for VSOS.

This program is very useful for e.g. SD cards of 64 GB and larger, which often are by default formatted to the proprietary exFAT file system that VSOS does not support. By formatting such cards to FAT32, they work both under VSOS and with computers.

If a serial number is not entered, a pseudo-random number, generated from a combination of the system clock and uptime counter, is used.

See also:
DiskFree (Chapter 7.1.19), FatInfo (Chapter 7.1.25).

7.1.28 Frags / MEMTRACK

```
Usage: Frags [-v|+v|-h]
-v|+v   Verbose on|off
-h      Show this help
```

List all libraries, and show the amount of static Instruction, X-Data and Y-Data memory they consume as ASCII graphics. With the help of the MEMTRACK.DL3 driver Frags can also show dynamically allocated memory.

You can run Frags from the VSOS Shell command line. However, more detailed output can be shown if you add the following line as the first line of config.sys:

```
MEMTRACK
```

You may also load the memory tracker from the VSOS Shell as follows:

```
S:>driver +memtrack
```

However, in this case the memory tracker cannot give as detailed information as if it was started earlier in the boot process.

The output of Frags looks as follows:

```
S:>frags
Libs: Memtrack Sdsd Auodac auIi2ss auXsyncs auxPlay Run Uartin sHell Frags
I 0000: ################ ################ ################ ################
I 1000: ################ ################ ################ ################
I 2000: ################ ################ ################ ################
I 3000: MMMMMMMMMSSSSSSS SSSSSSSSSSAAAAA AAAAAAAIIIIIIIIX XXXXXXXPPPPPRUU
I 4000: UUUUUUHHHFFFFFFF FFFFFFF......... ................ ................
```

```
I 5000: ................ ................ ................ ................
I 6000: ................ ................ ................ ................
I 7000: ................ ................ ................ ...............#

X 0000: ################ ################ ################ ################
X 1000: ################ ################ ################ ################SS
X 2000: ##IXPP#PPPP#HHHH HHHHHFFFFFF#.... ................ ................
X 3000: ................ ................ ................ ................
X 4000: ................ ................ ................ ................
X 5000: ................ ................ ................ ................
X 6000: ................ ................ ................ ................
X 7000: ................ ................ ................ ................

Y 0000: ################ ################ ################ MMMMMMMFIIIIIIII
Y 1000: UUUUUU.......... ................ ................ .####...........
Y 2000: .......PPPP..... ................ ................ ................
Y 3000: ................ ................ ................ ................
Y 4000: ................ ................ ................ ................
Y 5000: ................ ................ ................ ................
Y 6000: ................ ................ ................ ................
Y 7000: ################ ################ ################ ################

Free: I: 14864 words (45%), X: 22788 words (69%), Y: 23648 words (72%)
```

The first line shows the libraries that reside in memory. Below that, memory maps for Instruction, X Data, and Y Data memories are shown.

Free areas are marked with '.', allocated unknown areas (usually reserved by VSOS) with '#', and areas allocated by libraries with capital letters (e.g. 'M' of Memtrack). Finally, the total free amount of each memory type is shown.

See also:
liblist (Chapter 7.1.41), liblist2 (Chapter 7.1.43), driver (Chapter 7.1.20).

7.1.29 Ft?Agc

DC Blocker driver. For details, read the separate document *VSOS Audio Subsystem*.

See also:
SetAgc (Chapter 7.1.69), Ft?Equ (Chapter 7.1.31).

7.1.30 Ft?DcBl

DC Blocker driver. For details, read the separate document *VSOS Audio Subsystem*.

See also:
SetAgc (Chapter 7.1.69), Ft?Agc (Chapter 7.1.31).

7.1.31 Ft?Equ

DC Blocker and Equalizer driver. For details, read the separate document *VS1005 VSOS3 Equalizer FtEqu*.

See also:
SetEqu (Chapter 7.1.72).

7.1.32 Ft?Pitch

Pitch / Speed shifter. For details, read the separate document *VSOS Audio Subsystem*.

See also:
SetPitch (Chapter 7.1.73).

7.1.33 getcmd

```
Usage: getcmd
```

Library to reads a command line for the shell.

To see how to call this library, see the source code for The Shell.

7.1.34 GpioDemo

Demonstrates how to use GPIO interrupts in C language. Uses GPIO0_0 through GPIO0_3 (buttons S1-S4).

7.1.35 HiResRec

HiRes Recorder that can record stereo PCM up to 96 kHz 24-bit to an SD card. More documentation and latest version of the software is available at:
http://www.vsdsp-forum.com/phpbb/viewtopic.php?t=2210

The HiRes Recorder requires the SDSD23 SD Card driver to work.

See also:
SDSD23 (Chapter 7.1.67).

7.1.36 IntDemo

Demonstrates how to take control of interrupts in C language. Read the accompanying README.TXT for more details.

7.1.37 IntTrace

Interrupt-based tracing program. This program will connect itself to the INT_REGU interrupt (Power Button). Whenever the Power Button is pushed, lost of debug information is given on screen. This is especially useful for debugging systems that have got stuck.

Example:

```
S:>driver +inttrace
S:>
[... now push POWER ...]
SOF count: 0
Stop at 37160(0x9128): IROMNoSym
Previous PC samples:
0x9128=IROMNoSym
0x9128=IROMNoSym
[... etc ...]
SOF count: 0

Task 0x0021, priority 1, in RUNNING, name "MainTask"
  State: 4 (TS_WAIT)
  Stack: Start 0x0030, size 0x200, in use 0x54, max used 0x17a (0x86 free)
  Stack Trace: current PC 0x5214, Tasks::main[0xc9]
    Next: PC 0x1ecc @ stack 0x0096, Tasks::main[0xc9]
    Next: PC 0x4a06 @ stack 0x0090, IntTrace::IntReguC[0x50]
[... etc ...]

Task 0x0021, priority 1, in waitQueue, name "MainTask"
[... etc ...]

Timer queue 0x1c1f for task 0x1c01 ("cyclic")
  Tick count: 0x003d

Timer queue 0x23e9 for task 0x23cd ("SDSD23")
  Tick count: 0x0004

Timer queue 0x006a for task 0x0021 ("MainTask")
  Tick count: 0x0001

Interrupts:
  INT  0 INT_DAC     , pri 2, vector 0x4327= auodac::DacInterrupt
  INT  6 INT_MAC0    , pri 2, vector 0x4567= auiadc::AdcInterrupt
```

```
  INT 12 INT_UART_TX , pri 2, vector 0x409e= uartin::IntUartTxAsm
  INT 13 INT_UART_RX , pri 3, vector 0x408a= uartin::IntUartRxAsm
  INT 15 INT_TIMER1  , pri 2, vector 0x4b13= IntTrace::IntTimer1Asm
  INT 16 INT_TIMER2  , pri 1, vector 0x29f4= IntTrace::IntTimer1Asm
  INT 25 INT_REGU    , pri 1, vector 0x4b27= IntTrace::IntReguAsm

Hardware locks:
  BUFFER  4  locked:YES  in_queue:no  name:HLB_4
  IO     10  locked:YES  in_queue:no  name:HLIO_SD
  PERIP   5  locked:YES  in_queue:no  name:HLP_SD

10 libs loaded:

Lib 2101 has entry points 350d and 3 sections:
  I:3200..35e7 (1000 words)
  X:1f18..20b1 (410 words)
  X:20b4..20ff (76 words)
LCD177    1000i +  486x +    0y

Lib 211d has entry points 35e8 and 1 sections:
  I:35e8..360d (38 words)
run          38i +    0x +    0y

[... etc ...]

Free memory:
 I:48fa..49b5
 I:5124..7fbf
I: 12120 words (48480 bytes)
 X:2d0c..2e0f
 X:2f18..7fcf
X: 20924 words (41848 bytes)
 Y:0e04..0eff
 Y:11d8..1c0f
 Y:1d10..23db
 Y:25dc..2fff
 Y:5000..6fdf
 Y:7000..7fff
Y: 19460 words (38920 bytes)
```

See also:
Tasks (Chapter 7.1.75), liblist2 (Chapter 7.1.43).

7.1.38 lcd177

1.77-inch LCD driver.

See also:
lcdcon (Chapter 7.1.40).

7.1.39 lcd288, lcd288v

2.88-inch LCD drivers (horizontal, vertical).

See also:
lcdcon (Chapter 7.1.40), touch288 (Chapter 7.1.78).

7.1.40 lcdcon

LCD console driver.

See also:
lcd288 (Chapter 7.1.39), touch288 (Chapter 7.1.78).

7.1.41 liblist

```
Usage: liblist
```

List all libraries, their start instruction address, and the amount of static Instruction, X-Data and Y-Data memory they consume.

Below is an example output from liblist:

```
S:>liblist

10 libs loaded:
3000: memtrack   522i +   14x +  386y
320a: sdsd      1144i +  166x +   34y
3682: auodac     774i +   40x +    4y
3988: auii2ss    550i +   40x +    4y
3bae: auxsyncs   562i +   60x +    8y
3de0: auxplay    336i +  132x +    2y
3f30: run         38i +    0x +    0y
8000: uartin     496i +   16x +  388y
4142: shell      252i +  544x +    0y
423e: liblist    990i +  182x +    2y
Free memory after loading drivers:
 I: 14756 words (59024 bytes)
 X: 23236 words (46472 bytes)
 Y: 23708 words (47416 bytes)
```

See also:
liblist2 (Chapter 7.1.43), Frags (Chapter 7.1.28), driver (Chapter 7.1.20).

7.1.42 LCDMessage

```
Usage: LCDMessage Text follows here
```

Displays text on the LCD.

7.1.43 liblist2

```
Usage: liblist
```

List all libraries, their start instruction address, and the static Instruction, X-Data and Y-Data memory sections they consume.

See also:
liblist2 (Chapter 7.1.43), Frags (Chapter 7.1.28), driver (Chapter 7.1.20).

7.1.44 libtrace

Library that traces which library has caused e.g. a zero pointer violation.

7.1.45 loopback

```
Usage: loopback
```

Copies stdaudioin to stdaudioout in a busy loop.

7.1.46 ls

```
Usage: ls [path]
```

Show a short listing of a directory in file system file order.

See also:
dir (Chapter 7.1.18).

7.1.47 MEMTRACK

See Frags (Chapter 7.1.28).

7.1.48 metadata

```
Usage: metadata filename
```

Shows and/or returns metadata for an audio file. The output is in UTF-8 format, and for the fields it can find, its output looks as follows (but not necessarily in this order):

```
~0503'Song name
~0504'Album
~0505'Artist
~0506'Year
~050d'Track number
```

metadata can be used either from the command line, or from another application. For how to use metadata from C code, see the source code for PlayDir.

7.1.49 MkDir

```
Usage: MkDir [fulPath|-h]
fulPath Full path to the directory to create
-h      Show this help
```

Creats a new directory.

7.1.50 more

```
Usage: More [-r rows] [-c columns] [-x|+x] [-h] fileName
-x/+x   Hex mode on / off
-h      Show this help
```

Present an ASCII or binary file one page at a time.

See also:
Type (Chapter 7.1.80).

7.1.51 mp3model

MP3 decoder model library.

7.1.52 paramspl

```
Usage: paramspl parameters
```

Library that splits a parameter string to null-terminated strings, taking into account parenthesis and escape characters, then returns the number of parameters it created by splitting.

To see how to use this library, see the source code for Echo.

See also:
Echo (Chapter 7.1.22).

7.1.53 PlayDir

```
Usage: PlayDir [-v|+v|-p|+p|-s|+s|-h]
-v      Verbose
+v      Not verbose
-p      Start in pause mode
+p      Start in play mode
-s      Shuffle mode on
+s      Shuffle mode off
-h      Show this help page
```

Plays all audio files in the current directory.

PlayDir is described in detail in Chapter 8, *Using the UART Controlled Player PlayDir*.

7.1.54 PlayFile

```
Usage: PlayFile file
```

Plays one file in the file system. Can output some of the metadata that PlayDir does.

See also:
PlayFileLoop (Chapter 7.1.55).

7.1.55 PlayFileLoop

```
Usage: PlayFileLoop file
```

Demonstration for playing a file with a loop. Read the *VSOS Audio Subsystem* document for information on decoders that support looping.

See also:
PlayFile (Chapter 7.1.54).

7.1.56 PlayFiles

```
Usage: PlayFiles pattern
```

Plays files in the file system in the order they appear in the file system. Can output some of the metadata that PlayDir does.

Example:

```
PlayFiles *
```

7.1.57 preg

```
Usage: preg [-v|+v|-b|+b|-f|+f|-sSym|-h] [reg]|[addr=val|addr|=val|
             addr&=val|addr^=val]
reg     number, number-number, name
val     value, may be preceded with ~ for bitwise not
        For memory outside of Y space, precede with X: or I:
-l      List all registers (decrease verbosity to not get all bits)
-v/+v   Increase / decrease verbosity
-b/+b   Bit mode on/off
-f/+f   Fast mode on/off (removes disassembly symbols)
-sSym   Find public symbol Sym and print its address ('-' show all)
-h      Show this help

Examples:
  preg i:0x20-0x3f
  preg ana_cf
  preg -b 2g
  preg +v -l
  preg y:0xfec0=0x1010
  preg 0xfca1&=~0x0200
  preg -svo_printf
```

Debug program that prints / sets / modifies contents of registers or memory. When printing, symbolic register names as defined in vs1005g.h may be used.

In addition to setting registers / memory locations with operator ('='), preg can modify them with bitwise or ('|='), bitwise and ('&='), or bitwise exclusive or ('^=') operations. If any of these operations are used, the contents of the memory location is read from before written to.

Can also print addresses of symbolic dependencies with the -s option.

7.1.58 ProgramFlash

ProgramFlash is a metasolution that allows the user to program the contents of either the 1 MiB internal or 2 MiB external flash memory with an image earlier created by DumpFlash.

For details, open the solution and read the README.TXT.

See also:
DumpFlash (Chapter 7.1.21).

7.1.59 Purity

Experimental MLC NAND Flash driver.

See the solution's Readme.txt file for details.

7.1.60 RdsRadio

This is an experimental RDS radio receiver. Usage is as follows.

Example 1:

```
RdsRadio 93700
```

Start RDS Radio in interactive mode at 93.7 MHz.

Example 2:

```
RdsRadio s
```

Scan full frequency range, then exit.

In interactive mode there are the following keys available:

- n - Search for next channel
- p - Search for previous channel
- ? - Show current frequency and signal level.
- CTRL-C - Exit application.
- . - Adjust current frequency by +10 kHz (debug)
- , - Adjust current frequency by -10 kHz (debug)
- : - Adjust current frequency by +100 kHz (debug)
- ; - Adjust current frequency by -100 kHz (debug)
- T - Print current time count (debug)
- t - Perform automatic channel tuning (debug)
- d - Toggle RDS Debug mode on / off (debug)

To get radio working in background, the following commands may be used:

```
RdsRadio 93700
[... use the application until you want to put it into background ...]
[... push CTRL-C ...]
driver +auxsyncs
driver +auxplay
```

Now you have radio audio still playing, although without controls or RDS.

7.1.61 ReadCyc

Example of how to read a variable from the Cyclic demo.

See also:
RunCyc (Chapter 7.1.65).

7.1.62 Rec

```
Usage: Rec [-fm|-fo|-ff|-cs|-cm|-cl|-cr|-q{x}|-b{x}|-r{x}|-f|+f|-h] outFile
-fm|-fo|-ff     Format MP3|Ogg Vorbis|FLAC (alternative: file suffix)
-cs|-cm|-cl|-cr Stereo|Mono|Left|Right
-qx     Set quality to x (0-10, higher is better)
-bx     Set bitrate to x kbit/s (1-511)
-rx     Set sample rate to x Hz, or x kHz if x<1000
-f/+f   Use / Don't Use FILEBUF library for audio data buffering
-h      Show this help
outFile The output file name (e.g. D:REC.MP3)
```

Records audio to a file in MP3, Ogg Vorbis or FLAC format.

Rec is described in detail in Chapter 9.

7.1.63 rtcread

```
Usage: rtcread
```

Reads the current contents of the RTC, and locate results into the currentTime structure (see <vsos.h> for details on the structure).

An LR44 battery or similar power source must be connected to the RTC. If no battery can be found, or if the time is not set, 1999-12-31 12:00:00 is written to currentTime.

See also:
Date (Chapter 7.1.14), SetDate (Chapter 7.1.71).

7.1.64 run

```
Usage: run cmd param
```

Runs command *cmd* with parameters *param*, then remove *cmd* from memory.

Not for command line use, but useful in the startup script config.txt.

7.1.65 RunCyc

Example of how to start and run a Cyclic node.

Example:

```
S:>driver +runcyc
S:>readcyc
Tenths counter 26
```

See also:
ReadCyc (Chapter 7.1.61).

7.1.66 SDSD, SDSDR, SDSDX, SDSDMONO

```
Usage in config file: SDSD D
Usage on command line: driver +SDSD D
```

Efficient SD card SD mode driver. There are three versions of the driver:

- SDSD: Read and Write capability. Will call driver SDSDX for initialization. After that the SDSDX driver is removed from memory.
- SDSDR: Like SDSD, but with read-only capability.
- SDSDMONO: Monolithic version of the SD driver that includes the functionality of both SDSD and SDSDX.

See also:
SDSPI (Chapter 7.1.68), SDSD23 (Chapter 7.1.67).

7.1.67 SDSD23, SDSDX23, SDSDMN23

```
Usage in config file: SDSD D
Usage on command line: driver +SDSD D
```

Buffering SD card SD mode driver. The driver uses from one to four 1 Mbit VS23S010 buffer ICs or one 4 Mbit VS23S020 SRAM ICs for buffering. There are two versions of the driver:

- SDSD23: Read and Write capability. Will call driver SDSDX23 for initialization. After that the SDSDX23 driver is removed from memory.
- SDSD23MN: Monolithic version of the SD driver that includes the functionality of both SDSD23 and SDSDX23.

See also:
HiResRec (Chapter 7.1.35), SDSD (Chapter 7.1.66).

7.1.68 SDSPI

Old SD card driver that operates in SPI mode.

See also:
SDSD (Chapter 7.1.66).

7.1.69 SetAgc

Controls the Automatic Gain Control driver. For details, read the separate document *VSOS Audio Subsystem*.

See also:
Ft?Agc (Chapter 7.1.29), Ft?DcBl (Chapter 7.1.30).

7.1.70 SetClock

```
Usage: SetClock [f|usb|rtc|-fx|-lx|-cv|-iv|-av|-uf|-sf|-pd|-nf|-df|-rd|-wx|
                 -P|+P|-v|+v|-h]
x       Set clock to x MHz, auto-adjust(1)
usb     Set clock to exactly 60 MHz, auto-adjust(1)
rtc     Set to 32.768 kHz RTC clock, auto-adjust(1)
-fx     Set clock to x MHz (USE WITH CAUTION!)
        x may also be "usb" or "rtc"
-lx     Limit top speed to x MHz (USE WITH CAUTION!)
-cv     Set CVDD  to v volts (USE WITH CAUTION!)
-iv     Set IOVDD to v volts (USE WITH CAUTION!)
-av     Set AVDD  to v volts (USE WITH CAUTION!)
-uf     Set UART speed to f bps
-sf     Set SPI0 speed to f MHz
-pf     Set SPI1 speed to f MHz
```

```
-nf     Set NAND FLASH speed to f MHz
-df     Set SD speed to f MHz
-rd     Set RAM delay line to d (range 0(slow)-3(fast))
-wx     Wait x milliseconds
-P|+P   Disable/Enable Power-On-Reset (USE WITH CAUTION!)
-v/+v   Verbose on/off
-t      Test execution speed
-h      Show this help

(1) Auto-adjustment affects CVDD, RAM Delay, and POR.
```

Set and displays VS1005g clock speed, regulator voltages, and other speed-related parameters.

If the clock speed is given in auto-adjustment mode then core voltage, RAM delay and power-on-reset are automatically adjusted to match. If parameters are set individually, it is up to the user to do so in a safe order, and with safe parameter values.

Example output is shown below:

```
S:>setclock
SetClock: CLKI 86.016 MHz, limit 100.000 MHz, src PLL, RAM Delay 1, POR on
          CVDD 1.800V(21), IOVDD 3.48V(28), AVDD 3.48V(25)
          UART nominal 115200 bps, real 114841 bps (0.3% error), reg 0x006b
          SPI0 6.1 Mbit/s, SPI1 43.0 Mbit/s, NAND 43.0 Mbit/s, SD 21.5 Mbit/s
```

Warning:
Do *not* exceed maximum voltage limits. Doing so may cause permanent damage to VS1005 and/or external components.

7.1.71 SetDate

```
Usage: SetDate [YYYY-MM-DD|YY-MM-DD|HH:MM:SS] [-h]
YYYY-MM-DD      Set date, e.g. 2015-09-18
YY-MM-DD        Set date, e.g. 15-09-18
HH:MM:SS        Set time, e.g. 12:34:56
HH:MM           Set time, e.g. 12:34
-h              Show this help
```

Sets the time of the VS1005's RTC. Both the date and time may be set at the same time.

An LR44 battery or similar power source must be connected to the RTC.

See also:
Date (Chapter 7.1.14), RtcRead (Chapter 7.1.63).

7.1.72 SetEqu

Control the Equalizer driver. For details, read the separate document *VS1005 VSOS3 Equalizer FtEqu*.

See also:
Ft?Equ (Chapter 7.1.31).

7.1.73 SetPitch

```
Usage: SetPitch [-i|-o] [-sx] [-px] [-h]
-i      Set stdaudioin
-o      Set stdaudioout (default)
-sx     Set speed to x times normal (0.68 - 1.64 if pitch=1.0)
-px     Set pitch to x times normal (0.61 - 1.47 if speed=1.0)
-h      Show this help

Note:
Correct playback requires that 0.68 <= speed/pitch <= 1.644.
```

Control the Pitch Shifter driver. For details, read the separate document *VSOS Audio Subsystem*.

See also:
Ft?Pitch (Chapter 7.1.32).

7.1.74 Sine

```
Usage: Sine freq1 dbl1 dbr1 [freq2 dbl2 dbr2 [...]] [-rrate|-h]
freq dbl dbr    Set frequency to freq Hz, left volume to dbl dB,
                right volume to dbr dB. If dbl or dbr is greater than zero,
                volume is muted for that channel.
-rrate  Set sample rate to rate Hz
-h      Show this help
```

Generate one or several sine waves.

Examples:

```
S:>Sine 1001.2371 0 0
S:>Sine -r24000 1000 -6 1 2000 1 -6 3000 -6.1 -6.1
```

7.1.75 Tasks

```
Usage: Tasks [-v|+v] [-h]
-v|+v   Verbose on/off
-h      Show this help
```

Show running tasks, their timer queues and interrupts. With the verbose option, also show registers, stacks traces and hardware queues.

An example output for tasks is shown below:

```
S:>Tasks

Task 0x0021, priority 1, in RUNNING, name "MainTask"
 State: 3 (TS_READY)
  Stack: Start 0x0030, size 0x200, in use 0xd9, max used 0x172 (0x8e free)
  Stack Trace: current PC 0x4369, Tasks::main[185]
    Next: PC 0x1e42 @ stack 0x008c, KERNEL
    Next: PC 0x41d0 @ stack 0x0086, shell::main[43]
    Next: PC 0x04c1 @ stack 0x007c, KERNEL
    Next: PC 0x0569 @ stack 0x003e, KERNEL
    Next: PC 0x0087 @ stack 0x0032, KERNEL

Task 0x1c01, priority 10, in waitQueue, name "cyclic"
  State: 4 (TS_WAIT)
  Stack: Start 0x1c10, size 0x100, in use 0x29, max used 0x3a (0xc6 free)
  Stack Trace: current PC 0x918f, IROM
    Next: PC 0x2b79 @ stack 0x1c1e, KERNEL
    Next: PC 0x90b5 @ stack 0x1c11, IROM::exit
  Registers:
    i0:0x1c1f i1:0x0012 i2:0x1beb i3:0x1c08
    i4:0x1c1e i5:0x0000 i6:0x1c2b i7:0xfc08
    a2:0x0000 a1:0x0004 a0:0x8000 b2:0x0000 b1:0x0000 b0:0x0000
    c2:0x0000 c1:0x0000 c0:0x0064 d2:0x0000 d1:0x0005 d0:0x0004
    p1:0x0000 p0:0x0000 ls:0xebab le:0xffff lc:0x0000 mr0:0x0210 lr0:0x9184

Task 0x21a1, priority 2, in waitQueue, name "AUXPLAY"
  State: 4 (TS_WAIT)
  Stack: Start 0x21b0, size 0x100, in use 0x40, max used 0x57 (0xa9 free)
  Stack Trace: current PC 0x918f, IROM
    Next: PC 0x3b82 @ stack 0x21d5, auii2ss::AudioRead[68]
    Next: PC 0x091d @ stack 0x21cc, KERNEL
    Next: PC 0x3dfb @ stack 0x21c4, auxplay::AudioTask[27]
    Next: PC 0x90b5 @ stack 0x21b1, IROM::exit
  Registers:
    i0:0x21d6 i1:0x0012 i2:0x0e30 i3:0x21a8
    i4:0x21d5 i5:0x0000 i6:0x21e2 i7:0xfc08
    a2:0x0000 a1:0x0004 a0:0x8000 b2:0x0000 b1:0x0000 b0:0x0000
    c2:0x0000 c1:0x2150 c0:0x0080 d2:0x0000 d1:0x0001 d0:0x0030
    p1:0x0000 p0:0x0040 ls:0xebab le:0xffff lc:0x0000 mr0:0x0210 lr0:0x9184

Timer queue 0x1c1f for task 0x1c01 ("cyclic")
  Tick count: 0x0064
```

```
Timer queue 0x21d6 for task 0x21a1 ("AUXPLAY")
  Tick count: 0x0001

Interrupts:
  INT  0 INT_DAC     , pri 2, vector 0x37e5= auodac::DacInterrupt
  INT  5 INT_MAC1    , pri 2, vector 0x3a7c= auiadc::Dec6Interrupt
  INT 12 INT_UART_TX , pri 1, vector 0x3d70= uartin::UartTransmitInterrupt
  INT 13 INT_UART_RX , pri 1, vector 0x3da0= uartin::UartReceiveInterrupt
  INT 15 INT_TIMER1  , pri 2, vector 0x2a09= KERNEL
  INT 16 INT_TIMER2  , pri 2, vector 0x2a09= KERNEL

Hardware locks:
  BUFFER  4  locked:YES  in_queue:no  name:HLB_4
  IO     10  locked:YES  in_queue:no  name:HLIO_SD
  PERIP   5  locked:YES  in_queue:no  name:HLP_SD
```

There are three tasks running, the VSOS MainTask, the VSOS cyclic task, and a task for AUXPLAY driver. The cyclic task is running at the highest priority (10).

There are also two timer queues where cyclic and AUXPLAY are currently waiting. cyclic still has 0x64 ticks = 100/1000 seconds before it will next be wokrn up, but AUXPLAY will be woken up at the next 1/1000 s.

Stack status for each task is also shown. If the stack for a task ever gets full, memory will be trashed and the system may crash.

With the `-v` option also stack traces are shown for each task. For instance, you can see that current execution of AUXPLAY is in IROM address 0x918f (actually the Delay() function), called by AudioRead() from library auii2ss, which was called by a VSOS Kernel function (read() in this case), which again was called by the AudioTask() function of library AUXPLAY.

Then, all active interrupts are shown. With the `-v` option the output is as detailed as in the example. There you seen the interrupt numbers, their symbolic names, their priorities (higher is better), jump vectors, and the jump vector's symbolic name if available.

Finally, hardware locks and their queues are shown. If in_queue for any locks is 1, there may be an unrecoverable hardware lock race condition.

7.1.76 TextXY

```
Usage: TextXY [-c|-h] -xX -yY text
-c      Clear display
-xX     Set x coordinate to X
-yY     Set y coordinate to Y
```

Displays text on the current LCD screen.

7.1.77 Time

```
Usage: Time [program [parameters]|-h]
-h      Show this help
```

Measures and displays the time it took to execute a program.

An example output is shown below:

```
S:>time delay 1000
errCode 0, time:  1.010s
```

7.1.78 touch288

LCD touch driver. The same driver works for both the horizontal and vertical LCD driver.

See also:
lcd288 (Chapter 7.1.39), lcdcon (Chapter 7.1.40).

7.1.79 trace

```
Usage: trace startAddr [endAddr]   # trace instruction address [or range]
Usage: trace i:startAddr [endAddr] # trace instruction address [or range]
Usage: trace x:startAddr [endAddr] # trace X data mem  address [or range]
Usage: trace y:startAddr [endAddr] # trace Y data mem  address [or range]
```

Trace (a) memory address(es).

Example:

```
S:>trace i:0x4000
uartin::UartPutChar[44]
```

7.1.80 type

```
Usage: type [-c|+c|-h] [file1 [file2 [...]]]
-c      Print file information line
+c      Don't print file information
-h      Show this help
```

Type a file to screen.

See also:
more (Chapter 7.1.50).

7.1.81 uartin

Interrupt-based UART stdin/stdout driver.

7.1.82 usbhost

USB Host driver.

7.1.83 usbmsc

```
Usage: usbmsc d
```

USB Mass Storage driver. Publishes *d* as a USB mass storage drive.

Example:
`usbmsc s`
publishes the system drive 'S' as a USB Mass Storage driver (similar to botting the VS1005g Developer Board with button S1 pushed).

7.1.84 vs3emuc

Make the system wait for a connection with VSIDE / VS3EMU's UART interface. Makes it possible to e.g. load a new kernel to a system that is on.

Example:
```
S:>vs3emuc
VS1005g ready for vs3emu connection. Bye!
```
Now you may connect to the board with VSIDE / VS3EMU.

7.1.85 WatchdogDemo

Demonstrates how to use the VS1005 hardware watchdog. See source code for details.

7.1.86 ybitclr

```
Usage: ybitclr xxxx,y
```

Clears bit *y* in Y memory register *xxxx*. Both *xxxx* and *y* must bee hexadecimal numbers.

See also:
ybitset (Chapter 7.1.87).

7.1.87 ybitset

```
Usage: ybitset xxxx,y
```

Sets bit *y* in Y memory register *xxxx*. Both *xxxx* and *y* must bee hexadecimal numbers.

See also:
ybitclr (Chapter 7.1.86).

8 Using the UART Controlled Player PlayDir

To test the UART Controlled Player PlayDir, you need to do the following steps:

1. Copy some audio files to an SD card.
2. Insert SD card to VS1005 DevBoard.
3. Boot DevBoard to the VSOS Shell Environment.
4. Use `cd` to get to your audio content directory, and potentially `dir -a` or `dir -a +f` to list it.
5. Type `playdir` to run PlayDir.
6. Use PlayDir keyboard shortcuts and control commands described in Chapter 8.

8.1 Starting PlayDir

```
Usage: PlayDir [-v|+v|-p|+p|-s|+s|-h]
-v      Verbose
+v      Not verbose
-p      Start in pause mode
+p      Start in play mode
-s      Shuffle mode on
+s      Shuffle mode off
-h      Show this help
```

PlayDir plays the files in sorted order in the current directory. If there is not enough memory to sort the files, they are played in file system order.

First it takes a listing of the files in the current directory using *dir*.

Then, it outputs the following line:

```
~0205=Number of audio media files encountered
```

If PlayDir was started with the “-v” command line option, it will list all audio files with potential metadata.

After the optional list, PlayDir will open the first file, see if it can find metadata, and outputs information for the file. When file playback has finished, PlayDir will play the next file. When it has played all files, it exits.

NOTE!
For systems with a microcontroller, it is recommended that machine-controlled silent mode (echo -e) is switched on before starting PlayDir. For interactive use the default interactive echo mode (echo +e) is recommended.

8.2 PlayDir Output

For each file, and depending on whether metadata for the file was found, the following fields may be output (not necessarily in this order):

```
~0308=PlayDir's playlist track number, always first field, always printed
~0205=PlayDir's song number, always second field, always printed
~050b'File name, always third field, always printed
~0503'Song name
~0504'Album
~0505'Artist
~0506'Year
~050d'Track number
```

If not in shuffle mode, the Playlist Track Number and Song Number are the same. If in shuffle mode, the Playlist Track number is increasing, while the Song Number is shuffled.

When the song is playing, the following messages are being sent once a second:

```
~030a'Playback time in seconds
~030b'File position in per cent (0..100), only sent if changed
```

When master volume changes, the following message is shown:

```
~0206=volume
```

where *volume* is the attenuation from maximum volume in 0.5 dB steps. Setting for maximum volume is 0 (-0 dB), minimum volume is 254 (-127 dB).

When pause mode is toggled, the following message is shown:

```
~0104=pauseMode
```

where *pauseMode* is 1 if in pause mode, or 0 if in playback mode.

If the last song of a playlist is playing, and next song is selected, a line containing only the character '>' is printed, and PlayDir exits. If the first song is playing and previous song is selected, a line containing only '<' is printed, and PlayDir exits.

When PlayDir closes, it outputs the following line:

```
~0205=0
```

8.3 PlayDir Control Keys

PlayDir has the following controls keys:

n Next song. If last song is playing, '>' is printed and player exits
p Previous song. If first song is playing, '<' is printed and player exits
whitespace Toggle pause mode
< Lower volume by 0.5 dB
> Increase volume by 0.5 dB
. Fast forward 10 seconds
: Fast forward 60 seconds
, Rewind 10 seconds
; Rewind 60 seconds
- Lower playback pitch by 0.5 % (requires pitch shifter driver)
+ Heighten playback pitch by 0.5 % (requires pitch shifter driver)
= Return to base pitch (requires pitch shifter driver)
f Make playback 0.5 % faster (requires speed shifter driver)
s Make playback 0.5 % slower (requires speed shifter driver)
b Back to base playback speed (requires speed shifter driver)
q or Ctrl-C Exit player

8.4 PlayDir Control Commands

In addition to one-character controls, PlayDir also accepts control commands that are of the following format

`~xxxx=y`

where `xxxx` is an exactly 4-character UI Control message in hexadecimal, and `y` is the value as a C formatted number (decimal, hexadecimal, or octal).

An example that sets playback volume to -10 dB of maximum:

```
~0206=20
```

If the shell is in interactive echo mode (echo +e), then output of file progress messages are suppressed after the command-initiating '~' character to let the user more easily to see what he is typing. In machine-controlled silent mode (echo -e) text output continues as normal.

PlayDir recognizes the following control commands:

Control commands recognized by PlayDir			
Msg	**Parameter**		**Description**
	Min	**Max**	
0104	0	1	Set pause mode (0=off, 1=on).
0183	0	1	Set shuffle mode (0=off, 1=on).
0308	1	32767	Select track from current playlist. If out of bounds (0 or too large), PlayDir exits. Shuffle mode is automatically deactivated.
0206	0	255	Set volume to `-val`/2 dB of maximum. Maximum value (255) shuts down analog drivers if output is to driver AUODAC.DL3.
030a	0	65534	Go to `val` seconds in audio file.

If shuffle mode is activated with the set shuffle mode command, shuffle mode starts when the current song ends. When shuffle mode is deactivated, playback is continued from the current track.

9 Using the UART Controlled Recorder Rec

```
Usage: Rec [-fm|-fo|-ff|-cs|-cm|-cl|-cr|-q{x}|-b{x}|-r{x}|-f|+f|-h] outFile
-fm|-fo|-ff     Format MP3|Ogg Vorbis|FLAC (alternative: file suffix)
-cs|-cm|-cl|-cr Stereo|Mono|Left|Right
-qx     Set quality to x (0-10, higher is better)
-bx     Set bitrate to x kbit/s (1-511)
-rx     Set sample rate to x Hz, or x kHz if x<1000
-f/+f   Use / Don't Use FILEBUF library for audio data buffering
-h      Show this help
outFile The output file name (e.g. D:REC.MP3)
```

Rec records MP3, Ogg Vorbis, or FLAC audio to a given file. The user may define the channel modes, sample rate, and for MP3 or Ogg Vorbis either set a quality or a suggested bitrate.

If quality is set, the encoder used Variable BitRate (VBR) encoding, which gives the best quality / file size ratio. If the bitrate is set, the MP3 encoder uses Constant BitRate (CBR).

Sample rate may be set to anything the current audio driver for stdaudioin supports. If not set, the current sample rate for the audio driver is used.

If using an analog input audio driver, it is strongly recommended to start the FtIDcBl (Chapter 7.1.30) or FtIAgc (Chapter 7.1.29) driver before recording.

NOTE!
If the file is output to a device that may have latency issues, like an SD card, some kind of buffering must be activated. If the output driver does not support automatic buffering, it is advisable to activate the FILEBUF RAM buffer library with the “+f” option. Otherwise there will be gaps in the recording. Because of the much larger bitrates of FLAC files, FILEBUF’s buffering capabilities may not be sufficient to avoid gaps when recording to SD cards.

NOTE!
For systems with a microcontroller, it is recommended that machine-controlled silent mode (echo -e) is switched on before starting Rec. For interactive use the default interactive echo mode (echo +e) is recommended.

NOTE!
As a safety feature, Rec stops recording whenever the recording file grows to at least one billion bytes (1 GB).

9.1 Rec Output

When startingFor each file, and depending on whether metadata for the file was found, the following fields may be output (not necessarily in this order):

```
~050a'fileName
Output buffer size xx bytes
```

When the song is playing, the following message is sent once a second:

```
~030a'Playback time in seconds
```

When pause mode is toggled, the following message is shown:

```
~0104=pauseMode
```

where *pauseMode* is 1 if in pause mode, or 0 if in playback mode.

When Rec closes, it outputs the following line:

```
~010b=1
```

9.2 Rec Control Keys

Rec has the following controls keys:

n Next song. (Currently closes the program.)

whitespace Toggle pause mode

q or Ctrl-C Exit recorder

9.3 Rec Control Commands

In addition to one-character controls, Rec also accepts control commands that are of the following format

```
~xxxx=y
```

where `xxxx` is an exactly 4-character UI Control message in hexadecimal, and `y` is the value as a C formatted number (decimal, hexadecimal, or octal).

Rec also accepts report requests that are of the following format

```
~xxxx=y
```

where `xxxx` is an exactly 4-character UI Control message in hexadecimal.

An example that sets pause mode on:

```
~0104=1
```

If the shell is in interactive echo mode (echo +e), then output of file progress messages are suppressed after the command-initiating '~' character to let the user more easily to see what he is typing. In machine-controlled silent mode (echo -e) text output continues as normal.

Rec recognizes the following control commands:

Control commands recognized by Rec			
Msg	**Val**		**Description**
	Min	**Max**	
0104	0	1	Set pause mode (0=off, 1=on).

Rec recognizes the following report requests:

Report requests recognized by Rec	
Msg	**Description**
030d	Return file position in bytes.
030c	Return average bitrate for file in bps.
0209	Return maximum signal level in dB since last time read.

Request 0209 can be used to build a maximum level VU meter. Such a meter should warn the user (with e.g. yellow vs green colour) whenever the input level exceeds -5 dB, and show a clear overload error (with e.g. red colour or an “OVERLOAD” text) whenever maximum absolute level 0 dB is reached.

10 Latest Document Version Changes

This chapter describes the latest and most important changes to this document.

Version 3.54, 2018-02-08

- Removed information meant for users of very old versions of VSIDE (2.34 or older).

Version 3.53, 2018-01-31

- Some programs in Chapter 7.1, *Shell Command Programs and VSOS Libraries*, got new options, e.g.
  - Dir, Chapter 7.1.18
  - FatInfo, Chapter 7.1.25
- SD Card drivers now handle deinserting and inserting SD cards more gracefully.

Version 3.52, 2018-01-22

- Added several new programs to Chapter 7.1, *Shell Command Programs and VSOS Libraries*, e.g.
  - audiodecsmall, Chapter 7.1.4
  - HiResRec, Chapter 7.1.35
  - IntTrace, Chapter 7.1.37
  - LCD177, Chapter 7.1.38
  - Ft?Pitch, Chapter 7.1.32
  - TextXY, Chapter 7.1.76

Version 3.42, 2017-05-18

- Added several new programs to Chapter 7.1, *Shell Command Programs and VSOS Libraries*, and updated documentation for existing ones. New programs of particular interest are:
  - Format, Chapter 7.1.27
  - FatInfo, Chapter 7.1.25
- PlayDir was called PlayFile several times in Chapter 8, *Using the UART Controlled Player PlayDir*. Fixed.

Version 3.40, 2016-11-03

- Added several new programs to Chapter 7.1, *Shell Command Programs and VSOS Libraries*, and updated documentation for existing ones.

Version 3.30, 2016-06-22

- Added several new programs to Chapter 7.1, *Shell Command Programs and VSOS Libraries*, and updated documentation for existing ones.
- Changed document version numbering to reflect on the VSOS release this document has been written for.

Version 1.50, 2016-03-18

- Documented new features like command history for the VSOS Shell in Chapter 7, *Using the Shell Environment*.
- Added some new programs, most notably AuInfo (Chapter 7.1.6) and Edit (Chapter 7.1.23).

Version 1.40, 2016-02-15

This is an update for VSOS 3.26.

- Added many new programs and libraries to Chapter 7.1, *Shell Command Programs and VSOS Libraries*, and updated documentation for existing ones.
- Updated documetation in Chapter 7.1, *Rec*, to include Ogg Vorbis, and added recommendations on other libraries to use.

Version 1.30, 2015-10-02

- Added many new programs to Chapter 7.1, *Shell Programs*.

Version 1.20, 2015-07-17

- Added an audio recorder program, Rec, Chapter 7.1.62.

Version 1.11, 2015-06-01

- Added new functionality to PlayDir, Chapter 7.1.53, and slightly changed its output.
- Added terminal emulation program screen caps to Chapter 5, *Requirements*.

Version 1.10, 2015-05-29

First release with documentation. Added PlayDir.

Version 1.00, 2015-05-20

First release.

11 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