VS1063a DATASHEET
MP3/OGG/AAC/WMA/FLAC/G.711/G.722 AUDIO CODEC CIRCUIT

Key Features

- Encoders:
  - MP3; Ogg Vorbis; PCM; IMA ADPCM;
  - G.711 (µ-law, A-law); G.722 ADPCM
- Decoders:
  - MP3 (MPEG 1 & 2 audio layer III (CBR +VBR + ABR));
  - MP2 (layer II) (optional);
  - MPEG4 / AAC-LC(+PNS),
  - HE-AAC v2 (Level 3) (SBR + PS);
  - Ogg Vorbis; FLAC;
  - WAV (PCM, IMA ADPCM, G.711 µ-law/A-law, G.722 ADPCM)
- Full Duplex Codecs:
  - PCM; G.711 (µ-law, A-law); G.722 ADPCM;
  - IMA ADPCM
- Streaming support
- Upto 96 KiB RAM for user code and data
- Unique ID for user code protection
- Quiet power-on and power-off
- I2S interface for external DAC
- Serial control and data interfaces
- Can be used either as a slave co-processor or as a standalone processor
- UART for debugging purposes
- New functions may be added with software and up to 12 GPIO pins

Description

VS1063a is an easy-to-use, versatile encoder, decoder and codec for a multitude of audio formats.

VS1063a contains a high-performance, proprietary low-power DSP core VS_DSP4, ROM memories, 16 KiB instruction RAM and up to 80 KiB data RAM for user applications running simultaneously with any built-in decoder, serial control and input data interfaces, up to 12 general purpose I/O pins, a UART, as well as a high-quality variable-sample-rate stereo DAC and a stereo ADC, followed by an earphone amplifier and a common voltage buffer.

VS1063a can act both as an “MP3 decoder IC” or “MP3 encoder IC” slave in a system with a microcontroller, or as a stand-alone circuit that boots from external SPI memory.

Applications

- MP3-recording audio player
- Streaming server and client
- Wireless audio transfer
- Standalone player and recorder
- Internet phones
Additional Features

- EarSpeaker Spatial Processing
- Bass & treble controls
- Alternatively a 5-channel equalizer
- AD Mixer allows monitoring A/D converter input while listening to stream
- PCM Mixer allows inserting a sidestream while listening to main stream
- Adjustable Speed Shifter
- Operates with a single 12...13 MHz or 24...26 MHz clock
- Internal PLL clock multiplier
- Low-power operation
- High-quality on-chip stereo DAC with no phase error between channels
- Zero-cross detection for smooth volume change
- Stereo earphone driver capable of driving a 30Ω load
- Separate voltages for analog, digital, I/O
- Lead-free RoHS-compliant package

Further Description

VS1063a is a pin-compatible alternative for VLSI Solution's VS1053. It has all the functionality of VS1053 (except MP1 and MIDI decoding) and many new features, particularly MP3 and Ogg Vorbis recording.

Also full-duplex codec functions for phone applications have been added to VS1063a.

A factory-programmable unique chip ID provides a basis for digital rights management or unit identification features.

Operating Modes

VS1063a operates in one of two host modes: as a slave co-processor or as a standalone processor.

When used as a slave co-processor VS1063a can operate in three different operation modes: decoder, encoder or codec mode. In decoder mode VS1063a receives its input bitstream through a serial input bus. The input stream is decoded and passed through an 18-bit digital volume control to an oversampling sigma-delta DAC. Decoding is controlled via a serial control bus. In addition to the basic decoding, it is possible to add application specific features, like DSP effects, to the user RAM memory, or even to load user applications. In encoder mode VS1063a can reads audio from its analog inputs, optionally compresses the data, which can then be read by the host processor. In codec mode VS1063a offers a full-duplex audio interface.

When used as a standalone processor the VS1063a can boot either from SPI EEPROM or FLASH memory. Alternatively code and data can be provided by a host controller.

User Code

Users can write their own user interface or signal processing code for the VS1063a using VSIDE (VLSI Solution's Integrated Development Environment).

As a default, there are 16 KiB of free code RAM and about 4 KiB of free data RAM for user plugin applications. Depending on the application, the data RAM can be expanded to the full 80 KiB that is available in VS1063a.
Contents

VS1063 1

Table of Contents 3

List of Figures 6

1 Disclaimer 7

2 Licenses 7

3 Definitions 8

4 Characteristics & Specifications 9

4.1 Absolute Maximum Ratings ........................................... 9

4.2 Recommended Operating Conditions ................................ 9

4.3 Analog Characteristics .................................................. 10

4.4 Power Consumption ..................................................... 11

4.5 Digital Characteristics ................................................ 11

4.6 Switching Characteristics - Boot Initialization .................... 11

5 Packages and Pin Descriptions 12

5.1 Packages ................................................................. 12

5.1.1 LQFP-48 ............................................................ 12

6 Connection Diagram, LQFP-48 15

7 SPI Buses 17

7.1 SPI Bus Pin Descriptions ............................................... 17

7.1.1 VS10xx Native Modes (New Mode, recommended) ............... 17

7.1.2 VS1001 Compatibility Mode (deprecated, do not use in new designs) 17

7.2 Data Request Pin DREQ ................................................ 18

7.3 Serial Protocol for Serial Data Interface (SDI) ..................... 19

7.3.1 SDI in VS10xx Native Modes (New Mode, recommended) ........ 19

7.3.2 SDI in VS1001 Compatibility Mode (deprecated, do not use in new designs) .................................................. 20

7.3.3 Passive SDI Mode (deprecated, do not use in new designs) .... 20

7.4 Serial Protocol for Serial Command Interface (SCI) ............... 21

7.4.1 SCI Read ............................................................. 21

7.4.2 SCI Write ........................................................... 22

7.4.3 SCI Multiple Write .................................................. 22

7.5 SPI Timing Diagram .................................................... 23

7.6 SPI Examples with SM_SDINEW and SM_SDI_SHARED set ........ 24

7.6.1 Two SCI Writes ..................................................... 24

7.6.2 Two SDI Bytes ....................................................... 24

7.6.3 SCI Operation in Middle of Two SDI Bytes ....................... 25

8 Supported Audio Formats 26

8.1 Supported Audio Decoders ............................................. 26

8.1.1 Supported MP3 (MPEG layer III) Decoder Formats ............... 26

8.1.2 Supported MP2 (MPEG layer II) Decoder Formats ............... 26
8.1.3 Supported Ogg Vorbis Decoder Formats .............................................. 27
8.1.4 Supported AAC (ISO/IEC 13818-7 and ISO/IEC 14496-3) Decoder Formats ................................................................. 27
8.1.5 Supported WMA Decoder Formats ...................................................... 29
8.1.6 Supported FLAC Decoder Formats ...................................................... 30
8.1.7 Supported RIFF WAV Decoder Formats ............................................ 30
8.2 Supported Audio Encoding Formats ..................................................... 31
8.2.1 Supported MP3 (MPEG layer III) Encoding Formats ........................ 31
8.2.2 Supported Ogg Vorbis Encoding Formats .......................................... 32
8.2.3 Supported RIFF WAV Encoding Formats .......................................... 32

9 Functional Description ................................................................. 33
9.1 Main Features .................................................................................. 33
9.2 Decoder Data Flow of VS1063a ........................................................ 34
9.3 Encoder Data Flow of VS1063a ........................................................ 35
9.4 Codec Data Flow of VS1063a ........................................................... 36
9.5 EarSpeaker Spatial Processing .......................................................... 37
9.6 Serial Data Interface (SDI) ............................................................... 38
9.7 Serial Control Interface (SCI) ............................................................ 38
9.8 SCI Registers ................................................................................. 39
  9.8.1 SCI_MODE (RW) ........................................................................ 40
  9.8.2 SCI_STATUS (RW) .................................................................... 42
  9.8.3 SCI_BASS (RW) ......................................................................... 43
  9.8.4 SCI_CLOCKF (RW) .................................................................... 44
  9.8.5 SCI_DECODE_TIME (RW) .......................................................... 45
  9.8.6 SCI_AUDATA (RW) ..................................................................... 45
  9.8.7 SCI_WRAM (RW) ....................................................................... 45
  9.8.8 SCI_WRAMADDR (W) ................................................................. 45
  9.8.9 SCI_HDAT0 and SCI_HDAT1 (R) .................................................. 46
  9.8.10 SCI_AIADDR (RW) .................................................................... 49
  9.8.11 SCI_VOL (RW) .......................................................................... 49
  9.8.12 SCI_AICTRL[x] (RW) ................................................................. 50

10 Operation ......................................................................................... 51
10.1 Clocking ......................................................................................... 51
10.2 Hardware Reset ............................................................................... 51
10.3 Software Reset .............................................................................. 52
10.4 Low Power Mode .......................................................................... 52
10.5 Play and Decode ............................................................................ 52
  10.5.1 Playing a Whole File ................................................................ 53
  10.5.2 Cancelling Playback .................................................................. 53
  10.5.3 Fast Play .................................................................................. 53
  10.5.4 Fast Forward and Rewind without Audio .................................... 54
  10.5.5 Maintaining Correct Decode Time ............................................. 54
10.6 Feeding PCM Data ......................................................................... 55
10.7 Audio Encoding ............................................................................. 56
  10.7.1 Encoding Control Registers ....................................................... 56
  10.7.2 The Encoding Procedure .......................................................... 58
  10.7.3 Reading Encoded Data Through SCI ......................................... 59
  10.7.4 File Headers ............................................................................ 60
  10.7.5 Playing Encoded Data ............................................................... 61
10.7.6 Encoder Samplerate Considerations .................................. 61
10.7.7 Encode Monitoring Volume ............................................. 61
10.7.8 MP3 (format 5) Encoder Specific Considerations .................. 62
10.7.9 Ogg Vorbis (format 6) Encoder Specific Considerations ......... 62
10.7.10 Encoder/Decoder Delays .............................................. 63

10.8 Codec Mode ................................................................. 63

10.9 SPI Boot ................................................................. 64
10.10 I2C Boot ................................................................. 64

10.11 Extra Parameters (Parametric Structure) ............................... 65
  10.11.1 chipID, version, config1 ........................................... 66
  10.11.2 Player Configurations .............................................. 67
  10.11.3 VU Meter ............................................................ 69
  10.11.4 AD Mixer ........................................................... 70
  10.11.5 PCM Mixer .......................................................... 71
  10.11.6 EQ5 5-band Equalizer ............................................. 72
  10.11.7 Speed Shifter ....................................................... 74
  10.11.8 EarSpeaker ........................................................ 74
  10.11.9 Other Parameters .................................................. 75

10.12 SDI Tests ................................................................. 78
  10.12.1 Sine Test ............................................................ 78
  10.12.2 Pin Test ............................................................. 78
  10.12.3 SCI Test ............................................................ 78
  10.12.4 Memory Test ....................................................... 79
  10.12.5 New Sine and Sweep Tests ..................................... 79

10.13 Clock Speed Requirements ............................................. 80
  10.13.1 Clock Speed Requirements for Decoders ......................... 80
  10.13.2 Clock Speed Requirements for Encoders ......................... 80
  10.13.3 Clock Speed Requirements for DSP Algorithms .................. 81

11 VS1063a Version Changes .................................................. 82
  11.1 Firmware Changes Between VS1053b and VS1063a, 2011-04-13 .... 82

12 VS1063a Errata .............................................................. 84

13 Latest Document Version Changes ........................................ 85

14 Contact Information ....................................................... 87
List of Figures

1. Pin configuration, LQFP-48 .................................................. 12
2. VS1063a in LQFP-48 packaging ........................................... 12
3. Typical connection diagram using LQFP-48 ............................... 15
4. SDI in VS10xx Native Mode, single-byte transfer ....................... 19
5. SDI in VS10xx Native Mode, multi-byte transfer, \( X \geq 1 \) .................. 19
6. BSYNC signal - one byte transfer. Do not use in new designs! ........ 20
7. BSYNC signal - two byte transfer. Do not use in new designs! ........ 20
8. SCI word read ......................................................................... 21
9. SCI word write ......................................................................... 22
10. SCI multiple word write .......................................................... 22
11. SPI timing diagram ................................................................... 23
12. Two SCI operations ................................................................. 24
13. Two SDI bytes ......................................................................... 24
14. Two SDI bytes separated by an SCI operation .............................. 25
15. Decoder data flow of VS1063a .................................................. 34
16. Encoder data flow of VS1063a .................................................. 35
17. Codec data flow of VS1063a ...................................................... 36
18. EarSpeaker externalized sound sources vs. normal inside-the-head sound ... 37
19. Example EQ5 response request .................................................. 73
20. Example EQ5 responses at different volume settings ..................... 73
1 Disclaimer

This is a preliminary datasheet. All properties and figures are subject to change.

This datasheet assumes that the VS1063a Patches package, available at 
http://www.vlsi.fi/en/support/software/vs10xxplugins.html, has been loaded and activated.

Additional information is provided in two documents called VS1063a Hardware Guide, and VS1063a Programmer’s Guide.

2 Licenses

MPEG Layer-3 audio coding technology licensed from Fraunhofer IIS and Thomson.

Supply of this product does not convey a license nor imply any right to distribute MPEG Layer-3 compliant content created with this product in revenue-generating broadcast systems (terrestrial, satellite, cable and/or other distribution channels), streaming applications (via Internet, intranets and/or other networks), other content distribution systems (pay-audio of audio-on-demand applications and the like) or on physical media (compact discs, digital versatile discs, semiconductor chips, hard drives, memory cards and the like). An independent license for such use is required. For details, please visit http://mp3licensing.com.

Note: If you enable Layer II decoding, you are liable for any patent issues that may arise from using these formats. Joint licensing of MPEG 1.0 / 2.0 Layer III decoding does not cover all patents pertaining to layer II.

VS1063a contains WMA decoding technology from Microsoft. 
This product is protected by certain intellectual property rights of Microsoft and cannot be used or further distributed without a license from Microsoft.

VS1063a contains AAC decoding technology (ISO/IEC 13818-7 and ISO/IEC 14496-3) which cannot be used without a proper license from Via Licensing Corporation or individual patent holders.

VS1063a contains spectral band replication (SBR) and parametric stereo (PS) technologies developed by Coding Technologies. Both are currently part of the MPEG4 AAC licensing, see http://www.vialicensing.com/licensing/aac-overview.aspx for more information.

To the best of VLSI Solution’s knowledge, if the end product does not play a specific format that otherwise would require a customer license: MPEG 1.0/2.0 layer II, WMA, or AAC, the respective license should not be required. Decoding of MPEG layer II is disabled by default, and WMA and AAC formats can be excluded by using the parametric_x.config1 variable, or with a microcontroller based on the contents of register SCI_HDAT1. Also PS and SBR decoding can be separately disabled.
3 Definitions

**ABR** Average BitRate. Bitrate of stream may vary locally, but will stay close to a given number when averaged over a longer time.

**B** Byte, 8 bits.

**b** Bit.

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

**Ki** “Kibi” = $2^{10} = 1024$ (IEC 60027-2).

**Mi** “Mebi” = $2^{20} = 1048576$ (IEC 60027-2).

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

**VS_DSP** VLSI Solution's DSP core.

**VSIDE** VLSI Solution's Integrated Development Environment.

**W** Word. In VS_DSP, instruction words are 32-bits and data words are 16-bits wide.
4 Characteristics & Specifications

4.1 Absolute Maximum Ratings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Positive Supply</td>
<td>AVDD</td>
<td>-0.3</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Digital Positive Supply</td>
<td>CVDD</td>
<td>-0.3</td>
<td>1.85</td>
<td>V</td>
</tr>
<tr>
<td>I/O Positive Supply</td>
<td>IOVDD</td>
<td>-0.3</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Current at Any Non-Power Pin</td>
<td></td>
<td>±50</td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>Voltage at Any Digital Input</td>
<td>IOVDD+0.3</td>
<td>-0.3</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Operating Temperature</td>
<td></td>
<td>-30</td>
<td>+85</td>
<td>°C</td>
</tr>
<tr>
<td>Storage Temperature</td>
<td></td>
<td>-65</td>
<td>+150</td>
<td>°C</td>
</tr>
</tbody>
</table>

1 Higher current can cause latch-up.
2 Must not exceed 3.6 V

4.2 Recommended Operating Conditions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ambient Operating Temperature</td>
<td></td>
<td>-30</td>
<td>+85</td>
<td>°C</td>
<td></td>
</tr>
<tr>
<td>Analog and Digital Ground</td>
<td>AGND DGND</td>
<td>0.0</td>
<td></td>
<td>V</td>
<td></td>
</tr>
<tr>
<td>Positive Analog, REF=1.23V</td>
<td>AVDD</td>
<td>2.5</td>
<td>2.8</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Positive Analog, REF=1.65V</td>
<td>AVDD</td>
<td>3.3</td>
<td>3.3</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Positive Digital</td>
<td>CVDD</td>
<td>1.7</td>
<td>1.8</td>
<td>1.85</td>
<td>V</td>
</tr>
<tr>
<td>I/O Voltage</td>
<td>IOVDD</td>
<td>1.8</td>
<td>2.8</td>
<td>3.6</td>
<td>V</td>
</tr>
<tr>
<td>Input Clock Frequency</td>
<td>XTALI</td>
<td>12</td>
<td>12.288</td>
<td>13 MHz</td>
<td></td>
</tr>
<tr>
<td>Internal Clock Frequency</td>
<td>CLKI</td>
<td>12</td>
<td>36.864</td>
<td>67.6 MHz</td>
<td></td>
</tr>
<tr>
<td>Internal Clock Multiplier</td>
<td></td>
<td>1.0×</td>
<td>4.0×</td>
<td>5.5×</td>
<td></td>
</tr>
<tr>
<td>Master Clock Duty Cycle</td>
<td></td>
<td>40</td>
<td>50</td>
<td>60</td>
<td>%</td>
</tr>
</tbody>
</table>

1 Must be connected together as close the device as possible for latch-up immunity.
2 Reference voltage can be internally selected between 1.23V and 1.65V, see Chapter 9.8.2.
3 The maximum samplerate that can be played with correct speed is XTALI/256 (or XTALI/512 if SM_CLK_RANGE is set). Thus, XTALI must be at least 12.288 MHz (24.576 MHz) to be able to play 48 kHz at correct speed.
4 Reset value is 1.0×. Recommended SC_MULT=4.0×, SC_ADD=1.5× (SCI_CLOCKF=0xB000). Do not exceed maximum specification for CLKI.
### 4.3 Analog Characteristics

Unless otherwise noted: AVDD=3.3V, CVDD=1.8V, IOVDD=2.8V, REF=1.65V, TA=-30...+85°C, XTAL=12...13MHz, Internal Clock Multiplier 3.5×. DAC tested with 1307.894 Hz full-scale output sinewave, measurement bandwidth 20...20000 Hz, analog output load: LEFT to GBUF 30Ω, RIGHT to GBUF 30Ω. Microphone test amplitude 48 mVpp, f_s=1 kHz. Line input test amplitude 1.26 V, f_s=1 kHz.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAC Resolution</td>
<td></td>
<td>18</td>
<td></td>
<td></td>
<td>bits</td>
</tr>
<tr>
<td>Total Harmonic Distortion</td>
<td>THD</td>
<td></td>
<td>0.07</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Third Harmonic Distortion</td>
<td></td>
<td></td>
<td>0.02</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Dynamic Range (DAC unmuted, A-weighted)</td>
<td>IDR</td>
<td>100</td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>S/N Ratio (full scale signal)</td>
<td>SNR</td>
<td>94</td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Interchannel Isolation (Cross Talk), 600Ω + GBUF</td>
<td></td>
<td>80</td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Interchannel Isolation (Cross Talk), 30Ω + GBUF</td>
<td></td>
<td>53</td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Interchannel Gain Mismatch</td>
<td></td>
<td>-0.5</td>
<td>0.5</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Frequency Response</td>
<td></td>
<td>-0.1</td>
<td>0.1</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Full Scale Output Voltage (Peak-to-peak)</td>
<td></td>
<td>1.64</td>
<td>1.85</td>
<td>Vpp</td>
<td>2.06</td>
</tr>
<tr>
<td>Deviation from Linear Phase</td>
<td></td>
<td>5</td>
<td></td>
<td>°</td>
<td></td>
</tr>
<tr>
<td>Analog Output Load Resistance</td>
<td>AOLR</td>
<td>16</td>
<td>30</td>
<td>Ω</td>
<td>2</td>
</tr>
<tr>
<td>Analog Output Load Capacitance</td>
<td></td>
<td>100</td>
<td></td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>Microphone input amplifier gain</td>
<td>MICG</td>
<td>26</td>
<td></td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Microphone input amplitude</td>
<td></td>
<td>48</td>
<td>140</td>
<td>mVpp AC</td>
<td>4</td>
</tr>
<tr>
<td>Microphone Total Harmonic Distortion</td>
<td>MTHD</td>
<td>0.03</td>
<td>0.07</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Microphone S/N Ratio</td>
<td>MSNR</td>
<td>60</td>
<td>70</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Microphone input impedances, per pin</td>
<td></td>
<td>45</td>
<td></td>
<td>kΩ</td>
<td></td>
</tr>
<tr>
<td>Line input amplitude</td>
<td></td>
<td>2500</td>
<td>2800</td>
<td>mVpp AC</td>
<td>3</td>
</tr>
<tr>
<td>Line input Total Harmonic Distortion</td>
<td>LTHD</td>
<td>0.005</td>
<td>0.014</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Line input S/N Ratio</td>
<td>LSNR</td>
<td>85</td>
<td>90</td>
<td>dB</td>
<td></td>
</tr>
<tr>
<td>Line input impedance</td>
<td></td>
<td>80</td>
<td></td>
<td>kΩ</td>
<td></td>
</tr>
</tbody>
</table>

1. 3.0 volts can be achieved with + to + wiring for mono difference sound.
2. AOLR may be much lower, but below Typical distortion performance may be compromised.
3. Above typical amplitude the Harmonic Distortion increases.
4.4 Power Consumption

Tested with an Ogg Vorbis 128 kbps sample and generated sine. Output at full volume. Internal clock multiplier 3.0×. \( TA=+25°C \).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Supply Consumption AVDD, Reset</td>
<td>0.6</td>
<td>5.0</td>
<td>µA</td>
<td></td>
</tr>
<tr>
<td>Power Supply Consumption CVDD = 1.8V, Reset</td>
<td>12</td>
<td>20.0</td>
<td>µA</td>
<td></td>
</tr>
<tr>
<td>Power Supply Consumption AVDD, sine test, 30Ω + GBUF</td>
<td>30</td>
<td>36.9</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Power Supply Consumption CVDD = 1.8V, sine test</td>
<td>8</td>
<td>15</td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Power Supply Consumption AVDD, no load</td>
<td>5</td>
<td></td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Power Supply Consumption AVDD, output load 30Ω</td>
<td>11</td>
<td></td>
<td>mA</td>
<td></td>
</tr>
<tr>
<td>Power Supply Consumption AVDD, 30Ω + GBUF</td>
<td>11</td>
<td></td>
<td>mA</td>
<td></td>
</tr>
</tbody>
</table>

4.5 Digital Characteristics

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>High-Level Input Voltage (xRESET, XTALI, XTALO)</td>
<td>0.7×IOVDD</td>
<td>IOVDD+0.3(^1)</td>
<td>V</td>
</tr>
<tr>
<td>High-Level Input Voltage (other input pins)</td>
<td>0.7×CVDD</td>
<td>IOVDD+0.3(^1)</td>
<td>V</td>
</tr>
<tr>
<td>Low-Level Input Voltage</td>
<td>-0.2</td>
<td>0.3×CVDD</td>
<td>V</td>
</tr>
<tr>
<td>High-Level Output Voltage at XTALO = -0.1 mA</td>
<td>0.7×IOVDD</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Low-Level Output Voltage at XTALO = 0.1 mA</td>
<td>0.3×IOVDD</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>High-Level Output Voltage at ( I_O = -1.0 ) mA</td>
<td>0.7×IOVDD</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Low-Level Output Voltage at ( I_O = 1.0 ) mA</td>
<td>0.3×IOVDD</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>Input Leakage Current</td>
<td>-1.0</td>
<td>1.0</td>
<td>µA</td>
</tr>
<tr>
<td>SPI Input Clock Frequency (^2)</td>
<td>( \frac{CLKI}{4} )</td>
<td>MHz</td>
<td></td>
</tr>
<tr>
<td>Rise time of all output pins, load = 50 pF</td>
<td>50</td>
<td>ns</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) Must not exceed 3.6V
\(^2\) Value for SCI reads. SCI and SDI writes allow \( \frac{CLKI}{4} \).

4.6 Switching Characteristics - Boot Initialization

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRESET active time</td>
<td>2</td>
<td></td>
<td>XTALI</td>
<td></td>
</tr>
<tr>
<td>XRESET inactive to software ready</td>
<td>22000</td>
<td>50000(^1)</td>
<td>XTALI</td>
<td></td>
</tr>
<tr>
<td>Power on reset, rise time to CVDD</td>
<td>10</td>
<td></td>
<td>V/s</td>
<td></td>
</tr>
</tbody>
</table>

\(^1\) DREQ rises when initialization is complete. Do not send any data or commands before that.
5 Packages and Pin Descriptions

5.1 Packages

LPQFP-48 is a lead (Pb) free and also RoHS compliant package. RoHS is a short name of Directive 2002/95/EC on the restriction of the use of certain hazardous substances in electrical and electronic equipment.

5.1.1 LQFP-48

Figure 1: Pin configuration, LQFP-48

LQFP-48 package dimensions are at http://www.vlsi.fi/.

Figure 2: VS1063a in LQFP-48 packaging
<table>
<thead>
<tr>
<th>Pad Name</th>
<th>LQFP Pin</th>
<th>Pin Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>MICP / LINE1</td>
<td>1</td>
<td>AI</td>
<td>Positive differential mic input, self-biasing / Line-in 1</td>
</tr>
<tr>
<td>MICN</td>
<td>2</td>
<td>AI</td>
<td>Negative differential mic input, self-biasing</td>
</tr>
<tr>
<td>XRESET</td>
<td>3</td>
<td>DI</td>
<td>Active low asynchronous reset, schmitt-trigger input</td>
</tr>
<tr>
<td>DGND0</td>
<td>4</td>
<td>DGND</td>
<td>Core &amp; I/O ground</td>
</tr>
<tr>
<td>CVDD0</td>
<td>5</td>
<td>CPWR</td>
<td>Core power supply</td>
</tr>
<tr>
<td>IOVDD0</td>
<td>6</td>
<td>IOPWR</td>
<td>I/O power supply</td>
</tr>
<tr>
<td>CVDD1</td>
<td>7</td>
<td>CPWR</td>
<td>Core power supply</td>
</tr>
<tr>
<td>DREQ</td>
<td>8</td>
<td>DO</td>
<td>Data request, input bus</td>
</tr>
<tr>
<td>GPIO2 / DCLK¹</td>
<td>9</td>
<td>DIO</td>
<td>General purpose IO 2 / serial input data bus clock</td>
</tr>
<tr>
<td>GPIO3 / SDATA¹</td>
<td>10</td>
<td>DIO</td>
<td>General purpose IO 3 / serial data input</td>
</tr>
<tr>
<td>GPIO6 / I2S_SCLK¹</td>
<td>11</td>
<td>DIO</td>
<td>General purpose IO 6 / I2S_SCLK</td>
</tr>
<tr>
<td>GPIO7 / I2S_SDATA²</td>
<td>12</td>
<td>DIO</td>
<td>General purpose IO 7 / I2S_SDATA</td>
</tr>
<tr>
<td>XDCS / BSYNC²</td>
<td>13</td>
<td>DI</td>
<td>Data chip select / byte sync</td>
</tr>
<tr>
<td>IOVDD1</td>
<td>14</td>
<td>IOPWR</td>
<td>I/O power supply</td>
</tr>
<tr>
<td>VCO</td>
<td>15</td>
<td>DO</td>
<td>For testing only (Clock VCO output)</td>
</tr>
<tr>
<td>DGND1</td>
<td>16</td>
<td>DGND</td>
<td>Core &amp; I/O ground</td>
</tr>
<tr>
<td>XTALO</td>
<td>17</td>
<td>AO</td>
<td>Crystal output</td>
</tr>
<tr>
<td>XTAL1</td>
<td>18</td>
<td>AI</td>
<td>Crystal input</td>
</tr>
<tr>
<td>IOVDD2</td>
<td>19</td>
<td>IOPWR</td>
<td>I/O power supply</td>
</tr>
<tr>
<td>DGND2</td>
<td>20</td>
<td>DGND</td>
<td>Core &amp; I/O ground</td>
</tr>
<tr>
<td>DGND3</td>
<td>21</td>
<td>DGND</td>
<td>Core &amp; I/O ground</td>
</tr>
<tr>
<td>DGND4</td>
<td>22</td>
<td>DGND</td>
<td>Core &amp; I/O ground</td>
</tr>
<tr>
<td>XCS</td>
<td>23</td>
<td>DI</td>
<td>Chip select input (active low)</td>
</tr>
<tr>
<td>GPIO5 / I2S_MCLK³</td>
<td>25</td>
<td>DIO</td>
<td>General purpose IO 5 / I2S_MCLK</td>
</tr>
<tr>
<td>RX</td>
<td>26</td>
<td>DI</td>
<td>UART receive, connect to IOVDD if not used</td>
</tr>
<tr>
<td>TX</td>
<td>27</td>
<td>DO</td>
<td>UART transmit</td>
</tr>
<tr>
<td>SCLK</td>
<td>28</td>
<td>DI</td>
<td>Clock for serial bus</td>
</tr>
<tr>
<td>SI</td>
<td>29</td>
<td>DI</td>
<td>Serial input</td>
</tr>
<tr>
<td>SO</td>
<td>30</td>
<td>DO3</td>
<td>Serial output</td>
</tr>
<tr>
<td>CVDD3</td>
<td>31</td>
<td>CPWR</td>
<td>Core power supply</td>
</tr>
<tr>
<td>XTEST</td>
<td>32</td>
<td>DI</td>
<td>Reserved for test, connect to IOVDD</td>
</tr>
<tr>
<td>GPIO0</td>
<td>33</td>
<td>DIO</td>
<td>Gen. purp. IO 0 (SPIBOOT), use 100 kΩ pull-down resistor²</td>
</tr>
<tr>
<td>GPIO1</td>
<td>34</td>
<td>DIO</td>
<td>General purpose IO 1</td>
</tr>
<tr>
<td>GND</td>
<td>35</td>
<td>DGND</td>
<td>I/O Ground</td>
</tr>
<tr>
<td>GPIO4 / I2S_LROUT³</td>
<td>36</td>
<td>DIO</td>
<td>General purpose IO 4 / I2S_LROUT</td>
</tr>
<tr>
<td>AGND0</td>
<td>37</td>
<td>APWR</td>
<td>Analog ground, low-noise reference</td>
</tr>
<tr>
<td>AVDD0</td>
<td>38</td>
<td>APWR</td>
<td>Analog power supply</td>
</tr>
<tr>
<td>RIGHT</td>
<td>39</td>
<td>AO</td>
<td>Right channel output</td>
</tr>
<tr>
<td>AGND1</td>
<td>40</td>
<td>APWR</td>
<td>Analog ground</td>
</tr>
<tr>
<td>AGND2</td>
<td>41</td>
<td>APWR</td>
<td>Analog ground</td>
</tr>
<tr>
<td>GBUF</td>
<td>42</td>
<td>AO</td>
<td>Common buffer for headphones, do NOT connect to ground!</td>
</tr>
<tr>
<td>AVDD1</td>
<td>43</td>
<td>APWR</td>
<td>Analog power supply</td>
</tr>
<tr>
<td>RCAP</td>
<td>44</td>
<td>AO</td>
<td>Filtering capacitance for reference</td>
</tr>
<tr>
<td>AVDD2</td>
<td>45</td>
<td>APWR</td>
<td>Analog power supply</td>
</tr>
<tr>
<td>LEFT</td>
<td>46</td>
<td>AO</td>
<td>Left channel output</td>
</tr>
<tr>
<td>AGND3</td>
<td>47</td>
<td>APWR</td>
<td>Analog ground</td>
</tr>
<tr>
<td>LINE2</td>
<td>48</td>
<td>AI</td>
<td>Line-in 2 (right channel)</td>
</tr>
</tbody>
</table>
First pin function is active in New Mode, latter in Compatibility Mode.

Unless pull-down resistor is used, SPI Boot, followed by I2C Boot, is tried. See Chapters 10.9, SPI Boot, and 10.10, I2C Boot, for details.

If I2S_CF_ENA is ‘0’ the pins are used for GPIO. See VS1063a Hardware Guide’s Chapter I2S DAC Interface for details.

Pin types:

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DI</td>
<td>Digital input, CMOS Input Pad</td>
<td>AO</td>
<td>Analog output</td>
</tr>
<tr>
<td>DO</td>
<td>Digital output, CMOS Input Pad</td>
<td>AIO</td>
<td>Analog input/output</td>
</tr>
<tr>
<td>DIO</td>
<td>Digital input/output</td>
<td>APWR</td>
<td>Analog power supply pin</td>
</tr>
<tr>
<td>DO3</td>
<td>Digital output, CMOS Tri-stated Output Pad</td>
<td>DGND</td>
<td>Core or I/O ground pin</td>
</tr>
<tr>
<td>AI</td>
<td>Analog input</td>
<td>CPWR</td>
<td>Core power supply pin</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IOPWR</td>
<td>I/O power supply pin</td>
</tr>
</tbody>
</table>

Version: 1.00, 2012-10-24
Figure 3: Typical connection diagram using LQFP-48

Figure 3 shows a typical connection diagram for VS1063.

Figure Note 1: Connect either Microphone In or Line In, but not both at the same time.

Note: This connection assumes SM_SDINEW is active (see Chapter 9.8.1). If also SM_SDISHARE is used, xDCS should be tied high (see Chapter 7.1.1).
The common buffer GBUF can be used for common voltage (1.23 V) for earphones. This will eliminate the need for large isolation capacitors on line outputs, and thus the audio output pins from VS1063a may be connected directly to the earphone connector.

GBUF must NOT be connected to ground under any circumstances. If GBUF is not used, LEFT and RIGHT must be provided with coupling capacitors. To keep GBUF stable, you should always have the resistor and capacitor even when GBUF is not used.

Unused GPIO pins should have a pull-down resistor. Unused line and microphone inputs should not be connected.

If UART is not used, RX should be connected to IOVDD and TX be unconnected.

Do not connect any external load to XTALO.
The SPI Bus - that was originally used in some Motorola devices - has been used for both VS1063a's Serial Data Interface SDI (Chapters 7.3 and 9.6) and Serial Control Interface SCI (Chapters 7.4 and 9.7).

7.1 SPI Bus Pin Descriptions

### VS10xx Native Modes (New Mode, recommended)

These modes are active on VS1063a when SM_SDINEW is set to 1 (default at startup). DCLK and SDATA are not used for data transfer and they can be used as general-purpose I/O pins (GPIO2 and GPIO3). BSYNC function changes to data interface chip select (XDCS).

<table>
<thead>
<tr>
<th>SDI Pin</th>
<th>SCI Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>XDCS</td>
<td>XCS</td>
<td>Active low chip select input. A high level forces the serial interface into standby mode, ending the current operation. A high level also forces serial output (SO) to high impedance state. If SM_SDISHARE is 1, pin XDCS is not used, but the signal is generated internally by inverting XCS.</td>
</tr>
<tr>
<td>SCK</td>
<td></td>
<td>Serial clock input. The serial clock is also used internally as the master clock for the register interface. SCK can be gated or continuous. In either case, the first rising clock edge after XCS has gone low marks the first bit to be written.</td>
</tr>
<tr>
<td>SI</td>
<td>SO</td>
<td>Serial input. If a chip select is active, SI is sampled on the rising CLK edge. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.</td>
</tr>
</tbody>
</table>

### VS1001 Compatibility Mode (deprecated, do not use in new designs)

This mode is active when SM_SDINEW is set to 0. In this mode, DCLK, SDATA and BSYNC are active.

<table>
<thead>
<tr>
<th>SDI Pin</th>
<th>SCI Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>XCS</td>
<td>Active low chip select input. A high level forces the serial interface into standby mode, ending the current operation. A high level also forces serial output (SO) to high impedance state.</td>
</tr>
<tr>
<td>BSYNC</td>
<td>-</td>
<td>SDI data is synchronized with a rising edge of BSYNC.</td>
</tr>
<tr>
<td>DCLK</td>
<td>SCK</td>
<td>Serial clock input. The serial clock is also used internally as the master clock for the register interface. SCK can be gated or continuous. In either case, the first rising clock edge after XCS has gone low marks the first bit to be written.</td>
</tr>
<tr>
<td>SDATA</td>
<td>SI</td>
<td>Serial input. SI is sampled on the rising SCK edge, if XCS is low.</td>
</tr>
<tr>
<td>-</td>
<td>SO</td>
<td>Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.</td>
</tr>
</tbody>
</table>
7.2 Data Request Pin DREQ

The DREQ pin/signal is used to signal if VS1063a’s 2048-byte FIFO is capable of receiving data. If DREQ is high, VS1063a can take at least 32 bytes of SDI data or one SCI command. DREQ is turned low when the stream buffer is too full and for the duration of a SCI command.

Because of the 32-byte safety area, the sender may send up to 32 bytes of SDI data at a time without checking the status of DREQ, making controlling VS1063a easier for low-speed microcontrollers.

Note: DREQ may turn low or high at any time, even during a byte transmission. Thus, DREQ should only be used to decide whether to send more bytes. A transmission that has already started doesn’t need to be aborted.

Note: In VS1063a DREQ also goes down while an SCI operation is in progress.

There are cases when you still want to send SCI commands when DREQ is low. Because DREQ is shared between SDI and SCI, you cannot determine if a SCI command has been executed if SDI is not ready to receive. In this case you need a long enough delay after every SCI command to make certain none of them is missed. The SCI Registers table in Chapter 9.8 gives the worst-case handling time for each SCI register write.
7.3 Serial Protocol for Serial Data Interface (SDI)

The serial data interface operates in slave mode so DCLK signal must be generated by an external circuit.

Data (SDATA signal) can be clocked in at either the rising or falling edge of DCLK (Chapter 9.8).

VS1063a assumes its data input to be byte-synchronized. SDI bytes may be transmitted either MSb or LSb first, depending of register SCI_MODE bit SM_SDIORD (Chapter 9.8.1).

The firmware is able to accept the maximum bitrate the SDI supports.

7.3.1 SDI in VS10xx Native Modes (New Mode, recommended)

In VS10xx native modes (SM_NEWMODE is 1), byte synchronization is achieved by XDCS, as shown in Figure 4. The state of XDCS may not change while a data byte transfer is in progress. XDCS does not need to be deactivated and reactivated for every byte transfer, as shown in Figure 5. However, to maintain data synchronization even if there are occasional clock glitches, it is recommended to deactivate and reactivate XDCS every now and then, for example after each 32 bytes of data.

If SM_SDISHARE is 1, the XDCS signal is internally generated by inverting the XCS input.

For new designs, using VS10xx native modes are recommended.
7.3.2 SDI in VS1001 Compatibility Mode (deprecated, do not use in new designs)

When VS1063a is running in VS1001 compatibility mode, a BSYNC signal must be generated to ensure correct bit-alignment of the input bitstream. The first DCLK sampling edge (rising or falling, depending on selected polarity), during which the BSYNC is high, marks the first bit of a byte (LSB, if LSB-first order is used, MSB, if MSB-first order is used). If BSYNC is ‘1’ when the last bit is received, the receiver stays active and next 8 bits are also received.

7.3.3 Passive SDI Mode (deprecated, do not use in new designs)

If SM_NEWMODE is 0 and SM_SDISHARE is 1, the operation is otherwise like the VS1001 compatibility mode, but bits are only received while the BSYNC signal is ‘1’. Rising edge of BSYNC is still used for synchronization.
7.4 Serial Protocol for Serial Command Interface (SCI)

The serial bus protocol for the Serial Command Interface SCI (Chapter 9.7) consists of an instruction byte, address byte and one 16-bit data word. Each read or write operation can read or write a single register. Data bits are read at the rising edge, so the user should update data at the falling edge. Bytes are always send MSb first. XCS should be low for the full duration of the operation, but you can have pauses between bits if needed.

The operation is specified by an 8-bit instruction opcode. The supported instructions are read and write. See table below.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Opcode</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ</td>
<td>0b0000 0011</td>
<td>Read data</td>
</tr>
<tr>
<td>WRITE</td>
<td>0b0000 0010</td>
<td>Write data</td>
</tr>
</tbody>
</table>

Note: VS1063a sets DREQ low after each SCI operation. The duration depends on the operation. It is not allowed to finish a new SCI/SDI operation before DREQ is high again.

7.4.1 SCI Read

VS1063a registers are read from using the following sequence, as shown in Figure 8. First, XCS line is pulled low to select the device. Then the READ opcode (0x3) is transmitted via the SI line followed by an 8-bit word address. After the address has been read in, any further data on SI is ignored by the chip. The 16-bit data corresponding to the received address will be shifted out onto the SO line.

XCS should be driven high after data has been shifted out.

DREQ is driven low for a short while when in a read operation by the chip. This is a very short time and doesn’t require special user attention.
7.4.2 SCI Write

VS1063a registers are written from using the following sequence, as shown in Figure 9. First, XCS line is pulled low to select the device. Then the WRITE opcode (0x2) is transmitted via the SI line followed by an 8-bit word address.

After the word has been shifted in and the last clock has been sent, XCS should be pulled high to end the WRITE sequence.

After the last bit has been sent, DREQ is driven low for the duration of the register update, marked “execution” in the figure. The time varies depending on the register and its contents (see table in Chapter 9.8 for details). If the maximum time is longer than what it takes from the microcontroller to feed the next SCI command or SDI byte, status of DREQ must be checked before finishing the next SCI/SDI operation.

7.4.3 SCI Multiple Write

VS1063a allows for the user to send multiple words to the same SCI register, which allows fast SCI uploads, shown in Figure 10. The main difference to a single write is that instead of
bringing XCS up after sending the last bit of a data word, the next data word is sent immediately. After the last data word, XCS is driven high as with a single word write.

After the last bit of a word has been sent, DREQ is driven low for the duration of the register update, marked “execution” in the figure. The time varies depending on the register and its contents (see table in Chapter 9.8 for details). If the maximum time is longer than what it takes from the microcontroller to feed the next SCI command or SDI byte, status of DREQ must be checked before finishing the next SCI/SDI operation.

### 7.5 SPI Timing Diagram

The SPI timing diagram is presented in Figure 11.

The SPI timing diagram is presented in Figure 11.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>tXCSS</td>
<td>5</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tSU</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tH</td>
<td>2</td>
<td></td>
<td>CLKI cycles</td>
</tr>
<tr>
<td>tZ</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>tWL</td>
<td>2</td>
<td></td>
<td>CLKI cycles</td>
</tr>
<tr>
<td>tWH</td>
<td>2</td>
<td></td>
<td>CLKI cycles</td>
</tr>
<tr>
<td>tV</td>
<td>2 (+ 25 ns$^1$)</td>
<td></td>
<td>CLKI cycles</td>
</tr>
<tr>
<td>tXCSH</td>
<td>1</td>
<td></td>
<td>CLKI cycles</td>
</tr>
<tr>
<td>tXCS</td>
<td>2</td>
<td></td>
<td>CLKI cycles</td>
</tr>
<tr>
<td>tDIS</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

$^1$ 25 ns is when pin loaded with 100 pF capacitance. The time is shorter with lower capacitance.

Note: Although the timing is derived from the internal clock CLKI, the system always starts up in 1.0× mode, thus CLKI = XTALI. After you have configured a higher clock through SCI_CLOCKF and waited for DREQ to rise, you can use a higher SPI speed as well.

Note: Because tWL + tWH + tH is 6×CLKI + 25 ns, the maximum speed for SCI reads is CLKI/7.
7.6 SPI Examples with SM_SDINEW and SM_SDISHARED set

7.6.1 Two SCI Writes

Figure 12: Two SCI operations

Figure 12 shows two consecutive SCI operations. Note that xCS must be raised to inactive state between the writes. Also DREQ must be respected as shown in the figure.

7.6.2 Two SDI Bytes

Figure 13: Two SDI bytes

SDI data is synchronized with a raising edge of xCS as shown in Figure 13. However, every byte doesn’t need separate synchronization.
### 7.6.3 SCI Operation in Middle of Two SDI Bytes

Figure 14 shows how an SCI operation is embedded in between SDI operations. xCS edges are used to synchronize both SDI and SCI. Remember to respect DREQ as shown in the figure.
8 Supported Audio Formats

8.1 Supported Audio Decoders

### Conventions

<table>
<thead>
<tr>
<th>Mark</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Format is supported</td>
</tr>
<tr>
<td>-</td>
<td>Format exists but is not supported</td>
</tr>
<tr>
<td></td>
<td>Format doesn’t exist</td>
</tr>
</tbody>
</table>

#### 8.1.1 Supported MP3 (MPEG layer III) Decoder Formats

The VS1063 MP3 decoder is full-accuracy compliant.

**MPEG 1.0**:  
<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>40 48 56 64 80 96 112 128 160 192 224 256 320</td>
</tr>
<tr>
<td>48000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>44100</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>32000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
</tbody>
</table>

**MPEG 2.0**:  
<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>16 24 32 40 48 56 64 80 96 112 128 144 160</td>
</tr>
<tr>
<td>24000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>22050</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>16000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
</tbody>
</table>

**MPEG 2.5**:  
<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>16 24 32 40 48 56 64 80 96 112 128 144 160</td>
</tr>
<tr>
<td>12000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>11025</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>8000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
</tbody>
</table>

1 Also all variable bitrate (VBR) formats are supported.

#### 8.1.2 Supported MP2 (MPEG layer II) Decoder Formats

Note: Layer II decoding must be specifically enabled from register SCI_MODE.

**MPEG 1.0**:  
<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>48 56 64 80 96 112 128 160 192 224 256 320 384</td>
</tr>
<tr>
<td>48000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>44100</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
<tr>
<td>32000</td>
<td>+ + + + + + + + + + + + + + + + + + + + + + +</td>
</tr>
</tbody>
</table>
8 SUPPORTED AUDIO FORMATS

8.1.3 Supported Ogg Vorbis Decoder Formats

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channels</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>Window size</td>
<td>64</td>
<td>4096</td>
<td>samples</td>
</tr>
<tr>
<td>Samplerate</td>
<td>1</td>
<td>48000</td>
<td>Hz</td>
</tr>
<tr>
<td>Bitrate</td>
<td>0</td>
<td>500</td>
<td>kbit/sec</td>
</tr>
</tbody>
</table>

Of the two Ogg Vorbis floors, only floor 1 is supported. No known encoders since early preliminary releases have ever used floor 0. All one- and two-channel Ogg Vorbis files should be playable with this decoder.

8.1.4 Supported AAC (ISO/IEC 13818-7 and ISO/IEC 14496-3) Decoder Formats

VS1063a decodes MPEG2-AAC-LC-2.0.0.0 and MPEG4-AAC-LC-2.0.0.0 streams, i.e. the low complexity profile with maximum of two channels can be decoded. If a stream contains more than one element and/or element type, you can select which one to decode from the 16 single-channel, 16 channel-pair, and 16 low-frequency elements. The default is to select the first one that appears in the stream.

Dynamic range control (DRC) is supported and can be controlled by the user to limit or enhance the dynamic range of the material that contains DRC information.

Both Sine window and Kaiser-Bessel-derived window are supported.

For MPEG4 pseudo-random noise substitution (PNS) is supported. Short frames (120 and 960 samples) are not fully supported.

Spectral Band Replication (SBR) level 3, and Parametric Stereo (PS) level 3 are supported (HE-AAC v2). Level 3 means that maximum of 2 channels, samplerates upto and including 48 kHz without and with SBR (with or without PS) are supported. Also, both mixing modes ($R_a$ and $R_b$), IPD/OPD synthesis and 34 frequency bands resolution are implemented. The downsampled synthesis mode (core coder samplerates $>24$ kHz and $\leq 48$ kHz with SBR) is implemented.

SBR and PS decoding can also be disabled. Also different operating modes can be selected. See config1 and sbrAndPsStatus in Chapter 10.11, Extra parameters.

If enabled, the internal clock (CLKI) is automatically increased if AAC decoding needs a higher
clock. PS and SBR operation is automatically switched off if the internal clock is too slow for
correct decoding. Generally HE-AAC v2 files need 4.5\times clock to decode both SBR and PS
content. This is why 3.5\times + 2.0\times clock is the recommended default.

For AAC the streaming ADTS format is recommended. This format allows easy rewind and fast
forward because resynchronization is easily possible.

In addition to ADTS (.aac), MPEG2 ADIF (.aad) and MPEG4 AUDIO (.mp4 / .m4a) files are
played, but these formats are less suitable for rewind and fast forward operations. You can still
implement these features by using the jump points table, or using slightly less robust but much
easier automatic resync mechanism (see Chapter 10.5.4).

Because 3GPP (.3gp) and 3GPPv2 (.3g2) files are just MPEG4 files, those that contain only
HE-AAC or HE-AACv2 content are played.

**Important Note:** To be able to play the .3gp, .3g2, .mp4 and .m4a files, the **mdat** atom must
be the last atom in the MP4 file. Because VS1063a receives all data as a stream, all metadata
must be available before the music data is received. Several MP4 file formatters do not satisfy
this requirement and some kind of conversion is required. This is also why the streamable
ADTS format is recommended.

Programs exist that optimize the .mp4 and .m4a into so-called streamable format that has the
mdat atom last in the file, and thus suitable for web servers’ audio streaming. You can use this
kind of tool to process files for VS1063a too. For example `mp4creator -optimize file.mp4`.

### AAC¹²:

<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Maximum Bitrate kbit/s - for 2 channels</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤96</td>
<td>132</td>
</tr>
<tr>
<td>48000</td>
<td>+</td>
</tr>
<tr>
<td>44100</td>
<td>+</td>
</tr>
<tr>
<td>32000</td>
<td>+</td>
</tr>
<tr>
<td>24000</td>
<td>+</td>
</tr>
<tr>
<td>22050</td>
<td>+</td>
</tr>
<tr>
<td>16000</td>
<td>+</td>
</tr>
<tr>
<td>12000</td>
<td>+</td>
</tr>
<tr>
<td>11025</td>
<td>+</td>
</tr>
<tr>
<td>8000</td>
<td>+</td>
</tr>
</tbody>
</table>

¹ 64000 Hz, 88200 Hz, and 96000 Hz AAC files are played at the highest possible samplerate
(48000 Hz with 12.288 MHz XTALI).

² Also all variable bitrate (VBR) formats are supported. Note that the table gives the maximum
bitrate allowed for two channels for a specific samplerate as defined by the AAC specification.
The decoder does not actually have a fixed lower or upper limit.
8.1.5 Supported WMA Decoder Formats

Windows Media Audio codec versions 2, 7, 8, and 9 are supported. All WMA profiles (L1, L2, and L3) are supported. Previously streams were separated into Classes 1, 2a, 2b, and 3. The decoder has passed Microsoft's conformance testing program. Windows Media Audio Professional and Windows Media Audio Voice are different codecs and are not supported.

WMA 4.0 / 4.1:

<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>11025</td>
<td>+ +</td>
</tr>
<tr>
<td>16000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>22050</td>
<td>+ + + +</td>
</tr>
<tr>
<td>32000</td>
<td>+ + + + +</td>
</tr>
<tr>
<td>44100</td>
<td>+ + + + + + +</td>
</tr>
<tr>
<td>48000</td>
<td></td>
</tr>
</tbody>
</table>

WMA 7:

<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>11025</td>
<td>+ +</td>
</tr>
<tr>
<td>16000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>22050</td>
<td>+ + + +</td>
</tr>
<tr>
<td>32000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>44100</td>
<td>+ + + + + + +</td>
</tr>
<tr>
<td>48000</td>
<td></td>
</tr>
</tbody>
</table>

WMA 8:

<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>11025</td>
<td>+ +</td>
</tr>
<tr>
<td>16000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>22050</td>
<td>+ + + +</td>
</tr>
<tr>
<td>32000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>44100</td>
<td>+ + + + + + +</td>
</tr>
<tr>
<td>48000</td>
<td></td>
</tr>
</tbody>
</table>

WMA 9:

<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>8000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>11025</td>
<td>+ +</td>
</tr>
<tr>
<td>16000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>22050</td>
<td>+ + + +</td>
</tr>
<tr>
<td>32000</td>
<td>+ + + +</td>
</tr>
<tr>
<td>44100</td>
<td>+ + + + + + +</td>
</tr>
<tr>
<td>48000</td>
<td></td>
</tr>
</tbody>
</table>

In addition to these expected WMA decoding profiles, all other bitrate and samplerate combinations are supported, including variable bitrate WMA streams. Note that WMA does not consume the bitstream as evenly as MP3, so you need a higher peak transfer capability for clean playback at the same bitrate.
8.1.6  Supported FLAC Decoder Formats

The FLAC decoder provides the highest quality by providing lossless audio decompression.

Upto 48 kHz and 24-bit FLAC files with upto two channels are supported.

Because of the high data rate, the requirements for data transfer are much higher than for lossy codecs. Because of compression, audio buffer being shorter than the default FLAC block size, and some design choices in the FLAC format itself, the peak data transfer rate must be even higher than the sustained data rate required for uncompressed WAV files.

The FLAC decoder lowers the peak data transfer requirement a little by providing a larger stream buffer (12 KiB).

8.1.7  Supported RIFF WAV Decoder Formats

The following RIFF WAV formats are supported, with 1 or 2 audio channels and any samplerate upto $XTALI/256$ (48 kHz with 12.288 MHz clock).

<table>
<thead>
<tr>
<th>Format</th>
<th>Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>PCM</td>
<td>32, 24, 16 and 8 bits</td>
</tr>
<tr>
<td>0x03</td>
<td>IEEE_FLOAT</td>
<td>IEEE floating point data</td>
</tr>
<tr>
<td>0x06</td>
<td>ALAW</td>
<td>non-linear-quantized 8-bit samples (G.711 A-law)</td>
</tr>
<tr>
<td>0x07</td>
<td>MULAW</td>
<td>non-linear-quantized 8-bit samples (G.711 µ-law)</td>
</tr>
<tr>
<td>0x11</td>
<td>IMA_ADPCM</td>
<td>4 bits per sample</td>
</tr>
<tr>
<td>0x55</td>
<td>MPEGLAYER3</td>
<td>For supported MP3 modes, see Chapter 8.1.1</td>
</tr>
<tr>
<td>0x65</td>
<td>G722_ADPCM</td>
<td>two samples in 8 bits, same as 0x28f</td>
</tr>
<tr>
<td>0x28f</td>
<td>ADPCM_G722</td>
<td>two samples in 8 bits, same as 0x65</td>
</tr>
<tr>
<td>0xfffe</td>
<td>Extended PCM</td>
<td>32, 24, 16 and 8 bits, default channel configuration supported</td>
</tr>
</tbody>
</table>
8.2 Supported Audio Encoding Formats

8.2.1 Supported MP3 (MPEG layer III) Encoding Formats

VS1063a supports all MP3 samplerates and bitrates, in stereo and mono, both with constant bit-rate (CBR) or variable bitrate (VBR). The following tables apply to constant bit-rate.

### Conventions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>++</td>
<td>Format is supported and recommended for this channel configuration and bitrate.</td>
</tr>
<tr>
<td>+</td>
<td>Format is supported.</td>
</tr>
<tr>
<td>x</td>
<td>Format is supported but use is strongly discouraged for quality reasons.</td>
</tr>
<tr>
<td>v</td>
<td>Format is supported but for best quality lower samplerate with same bitrate is recommended.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Format is supported but lower bitrate will give same quality.</td>
</tr>
<tr>
<td>-</td>
<td>Format exists but isn’t supported.</td>
</tr>
<tr>
<td></td>
<td>Format doesn’t exist.</td>
</tr>
</tbody>
</table>

**MPEG 1.0 layer III (MP3 full-rates), stereo:**

<table>
<thead>
<tr>
<th>Samplerate / Hz</th>
<th>Bitrate / kbit/s, stereo</th>
</tr>
</thead>
</table>
| 48000           | V  V  V  V  V  V  V  +  +  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  ++  +
8.2.2 Supported Ogg Vorbis Encoding Formats

The Ogg Vorbis Encoder supports encoding in mono and stereo, with any samplerate between 1 and 48000 Hz, and with different quality settings. Ogg Vorbis is always encoded using variable bitrate (VBR).

Some example setting profiles are provided below. Note, however, that the encoder is not limited to these configurations.

The “Voice” profiles are intended for speech applications.

<table>
<thead>
<tr>
<th>Quality setting</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical kbit/s</td>
<td>6</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>12</td>
<td>13</td>
<td>16</td>
<td>19</td>
<td>22</td>
<td>25</td>
<td>28</td>
</tr>
</tbody>
</table>

“Wideband Voice” is intended to be used when high speech quality is required.

<table>
<thead>
<tr>
<th>Quality setting</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical kbit/s</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>18</td>
<td>21</td>
<td>25</td>
<td>31</td>
<td>37</td>
<td>43</td>
<td>49</td>
<td>55</td>
</tr>
</tbody>
</table>

“Wideband Stereo Voice” is intended to be used when high speech quality with directional information is required.

<table>
<thead>
<tr>
<th>Quality setting</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical kbit/s</td>
<td>10</td>
<td>18</td>
<td>26</td>
<td>34</td>
<td>42</td>
<td>50</td>
<td>65</td>
<td>81</td>
<td>96</td>
<td>112</td>
<td>127</td>
</tr>
</tbody>
</table>

When extremely high quality speech is required, use the “HiFi Voice” profiles.

<table>
<thead>
<tr>
<th>Quality setting</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical kbit/s</td>
<td>37</td>
<td>47</td>
<td>57</td>
<td>68</td>
<td>78</td>
<td>88</td>
<td>99</td>
<td>110</td>
<td>122</td>
<td>133</td>
<td>144</td>
</tr>
</tbody>
</table>

The “Music” profiles are intended for HiFi music.

<table>
<thead>
<tr>
<th>Quality setting</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Typical kbit/s</td>
<td>53</td>
<td>72</td>
<td>91</td>
<td>110</td>
<td>129</td>
<td>148</td>
<td>185</td>
<td>222</td>
<td>259</td>
<td>296</td>
<td>333</td>
</tr>
</tbody>
</table>

8.2.3 Supported RIFF WAV Encoding Formats

The following RIFF WAV formats are supported in encoding and codec modes with one or two channels and samplersates up to $XTAL/256$ (48 kHz with 12.288 MHz clock).

<table>
<thead>
<tr>
<th>Format</th>
<th>Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>PCM</td>
<td>16 and 8 bits linear PCM</td>
</tr>
<tr>
<td>0x06</td>
<td>ALAW</td>
<td>A-law, non-linear-quantized 8-bit samples</td>
</tr>
<tr>
<td>0x07</td>
<td>MULAW</td>
<td>µ-law, non-linear-quantized 8-bit samples</td>
</tr>
<tr>
<td>0x11</td>
<td>IMA_ADPCM</td>
<td>IMA ADPCM, 4 bits per sample, 505 samples per block</td>
</tr>
<tr>
<td>0x28f</td>
<td>ADPCM_G722</td>
<td>G722 subband ADPCM, two samples in 8 bits</td>
</tr>
</tbody>
</table>

Version: 1.00, 2012-10-24
9 Functional Description

9.1 Main Features

VS1063a is based on a proprietary digital signal processor, VS_DSP. It contains all the code and data memory needed for Ogg Vorbis, MP3, AAC, WMA, FLAC and WAV PCM + ADPCM audio decoding together with serial interfaces, a multirate stereo audio DAC and analog output amplifiers and filters.

Also MP3, OGG, PCM, ADPCM, µ-law, A-law and G.722 audio encoding is supported using a microphone amplifier and/or line-level inputs and a stereo A/D converter.

For streaming applications there exists a codec mode that supports full-duplex operation using PCM, ADPCM, µ-law, A-law or G.722 formats. The formats and samplersates don’t have to be the same in both directions.

A UART is provided for debugging purposes to connect with VLSI Solution’s Integrated Developments Environment VSIDE.
9.2 Decoder Data Flow of VS1063a

Figure 15: Decoder data flow of VS1063a

Figure 15 presents the decoder dataflow of VS1063a.

First, depending on the audio data, and provided encoding mode is not set (register SCI_MODE but SM_ENCODE is 0), audio bitstream is received from the SDI bus and decoded.

After decoding, if SCI_AIADDR is non-zero, user plugin code is executed from the address pointed to by that register. For more details, see VS1063a Programmer’s Guide.

Then data may be sent to the Bass Enhancer and Treble Control depending on the SCI_BASS register. If SCI_BASS is 0, but EQ5 Enable bit in Extra Parameters register playMode is 1, the 5-channel equalizer is used.

Next, if bit speedShifterEnable of Extra Parameters register playMode is 1, Speed Shifter is called. Otherwise, and if EarSpeakerLevel is not 0, headphone processing is done.

At this stage, and if Extra Parameters register playMode bit monoOutputSelect is 1, audio is converted to mono.

If Extra Parameters register playMode bit pause is 1, audio transmission is stopped.
After that the data is fed to the Audio FIFO. The size of the audio FIFO is 2048 stereo (2 × 16-bit) samples, or 8 KiB.

Now decoded and processed audio is sent to a samplerate converter, where volume control is applied. After this step it is combined with an optional sidestream which may be either PCM samples coming through the SCI bus or analog data from the line/mic input.

The samplerate converter upsamples all different samplerates to XTAL/2, or 128 times the highest usable samplerate with 18-bit precision. Volume control is performed in the upsampled domain. New volume settings are loaded only when the upsampled signal crosses the zero point (or after a timeout). This zero-crossing detection almost completely removes all audible noise that occurs when volume is suddenly changed.

The samplerate conversion to a common samplerate removes the need for complex PLL-based clocking schemes and allows almost unlimited samplerate accuracy with one fixed input clock frequency. With a 12.288 MHz clock, the DA converter operates at 128 × 48 kHz, i.e. 6.144 MHz, and creates a stereo in-phase analog signal. The oversampled output is low-pass filtered by an on-chip analog filter. This signal is then forwarded to the earphone amplifier.

9.3 Encoder Data Flow of VS1063a

Figure 16 presents the encoder dataflow of VS1063a.

Depending on which samplerate the user has requested, data is read from the Analog-to-Digital Converter with one of samplerates or 12, 24, or 48 kHz. A 10 Hz subsonic high-pass filter (not shown in the figure) is applied to the signal.

Here audio is split into two: one path going to monitoring, the other path going to the encoder.

Depending whether the signal needs to be resampled, it may be fed to the Resampler Sample Rate Converter that is used for samplerate fine tuning, and/or to the Software decimator which can decimate the signal by 2 or 3. (E.g. if chosen samplerate is 8 kHz, it will be digitized at 24 kHz, then downsampled by 3 with the Software decimator).

From the decimator stages, audio is fed to audio in FIFO, from which the encoder reads the samples.

The bitstream generated by the encoder is fed to the Bitstream out FIFO. The data is then either read through SCI by the user or output by the VS1063a to the UART.
9.4 Codec Data Flow of VS1063a

Figure 17: Codec data flow of VS1063a

The decoder and encoder paths are almost similar as in the decoder and encoder data flow Chapters 9.2 and 9.3, except that there is no decoder audio side path and the amount of samplerates in the encoder is much more limited because the Resampler SRC is not used.

A new path is when Acoustic Echo Cancellation (AEC) is active. In this case there is a feedback from the output path to the input path.

Note: Do not use Speed Shifter or Pause in Codec mode.

Note: Do not use EarSpeaker if AEC is active.

Note: If AEC is used, encoding and decoding samplerates must be the same. Otherwise they may be different.
While listening to headphones the sound has a tendency to be localized inside the head. The sound field becomes flat and lacking the sensation of dimensions. This is an unnatural, awkward and sometimes even disturbing situation. This phenomenon is often referred in literature as 'lateralization', meaning 'in-the-head' localization. Long-term listening to lateralized sound may lead to listening fatigue.

All real-life sound sources are external, leaving traces to the acoustic wavefront that arrives to the ear drums. From these traces, the auditory system of the brain is able to judge the distance and angle of each sound source. In loudspeaker listening the sound is external and these traces are available. In headphone listening these traces are missing or ambiguous.

EarSpeaker processes sound to make listening via headphones more like listening to the same music from real loudspeakers or live music. Once EarSpeaker processing is activated, the instruments are moved from inside to the outside of the head, making it easier to separate the different instruments (see Figure 18). The listening experience becomes more natural and pleasant, and the stereo image is sharper as the instruments are widely on front of the listener instead of being inside the head.

![Figure 18: EarSpeaker externalized sound sources vs. normal inside-the-head sound](image)

Note that EarSpeaker differs from any common spatial processing effects, such as echo, reverb, or bass boost. EarSpeaker accurately simulates the human auditory model and real listening environment acoustics. Thus is does not change the tonal character of the music by introducing artificial effects.

For how to set EarSpeaker registers, see Chapter 10.11.8.
9.6 Serial Data Interface (SDI)

The serial data interface is meant for transferring compressed data for the different decoders of VS1063a.

If the input of the decoder is invalid or it is not received fast enough, analog outputs are automatically muted.

Also several different tests may be activated through SDI as described in Chapter 10.

9.7 Serial Control Interface (SCI)

The serial control interface is compatible with the SPI bus specification. Data transfers are always 16 bits. VS1063a is controlled by writing and reading the registers of the interface.

The main controls of the serial control interface are:

- control of the operation mode, clock, and builtin effects
- access to status information and header data
- receiving encoded data in recording mode
- uploading and controlling user programs
9.8 SCI Registers

VS1063a sets DREQ low when it detects an SCI operation (this delay is 16 to 40 CLKI cycles depending on whether an interrupt service routine is active) and restores it when it has processed the operation. The duration depends on the operation. If DREQ is low when an SCI operation is performed, it also stays low after SCI operation processing.

If DREQ is high before an SCI operation, do not start a new SCI/SDI operation before DREQ is high again. If DREQ is low before a SCI operation because the SDI can not accept more data, make certain there is enough time to complete the operation before sending another.

<table>
<thead>
<tr>
<th>Reg</th>
<th>Typ.</th>
<th>Reset</th>
<th>Write Time</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>rw</td>
<td>0x4000&lt;sup&gt;1&lt;/sup&gt;</td>
<td>80 CLKI</td>
<td>MODE</td>
<td>Mode control</td>
</tr>
<tr>
<td>0x1</td>
<td>rw</td>
<td>0x000C&lt;sup&gt;1&lt;/sup&gt;</td>
<td>80 CLKI</td>
<td>STATUS</td>
<td>Status of VS1063a</td>
</tr>
<tr>
<td>0x2</td>
<td>rw</td>
<td>0</td>
<td>80 CLKI</td>
<td>BASS</td>
<td>Built-in bass/treble control</td>
</tr>
<tr>
<td>0x3</td>
<td>rw</td>
<td>0</td>
<td>1200 XTALI&lt;sup&gt;1&lt;/sup&gt;</td>
<td>CLOCKF</td>
<td>Clock freq + multiplier</td>
</tr>
<tr>
<td>0x4</td>
<td>rw</td>
<td>0</td>
<td>100 CLKI</td>
<td>DECODE_TIME</td>
<td>Decode time in seconds</td>
</tr>
<tr>
<td>0x5</td>
<td>rw</td>
<td>0</td>
<td>450 CLKI</td>
<td>AUDATA</td>
<td>Misc. audio data</td>
</tr>
<tr>
<td>0x6</td>
<td>rw</td>
<td>0</td>
<td>100 CLKI</td>
<td>WRAM</td>
<td>RAM write/read</td>
</tr>
<tr>
<td>0x7</td>
<td>rw</td>
<td>0</td>
<td>100 CLKI</td>
<td>WRAMADDR</td>
<td>Set address for RAM write/read</td>
</tr>
<tr>
<td>0x8</td>
<td>r</td>
<td>0</td>
<td>80 CLKI</td>
<td>HDAT0</td>
<td>Stream header data 0</td>
</tr>
<tr>
<td>0x9</td>
<td>r</td>
<td>0</td>
<td>80 CLKI</td>
<td>HDAT1</td>
<td>Stream header data 1</td>
</tr>
<tr>
<td>0xA</td>
<td>rw</td>
<td>0</td>
<td>210 CLKI</td>
<td>AIADDR</td>
<td>Start address of application/plugin</td>
</tr>
<tr>
<td>0xB</td>
<td>rw</td>
<td>0</td>
<td>80 CLKI</td>
<td>VOL</td>
<td>Volume control</td>
</tr>
<tr>
<td>0xC</td>
<td>rw</td>
<td>0</td>
<td>80 CLKI</td>
<td>AICTRL0</td>
<td>Application control register 0</td>
</tr>
<tr>
<td>0xD</td>
<td>rw</td>
<td>0</td>
<td>80 CLKI</td>
<td>AICTRL1</td>
<td>Application control register 1</td>
</tr>
<tr>
<td>0xE</td>
<td>rw</td>
<td>0</td>
<td>80 CLKI</td>
<td>AICTRL2</td>
<td>Application control register 2</td>
</tr>
<tr>
<td>0xF</td>
<td>rw</td>
<td>0</td>
<td>80 CLKI</td>
<td>AICTRL3</td>
<td>Application control register 3</td>
</tr>
</tbody>
</table>

<sup>1</sup> This is the worst-case time that DREQ stays low after writing to this register. The user may choose to skip the DREQ check for those register writes that take less than 100 clock cycles to execute and use a fixed delay instead.

<sup>2</sup> In addition, the cycles spent in the user application/plugin routine must be counted.

<sup>3</sup> Firmware changes the value of this register immediately to 0x68 (analog enabled), and after a short while to 0x60 (analog drivers enabled).

<sup>4</sup> When mode register write specifies a software reset the worst-case time is 22000 XTALI cycles.

<sup>5</sup> If the clock multiplier is changed, writing to SCI_CLOCKF register may force internal clock to run at 1.0 × XTALI for a while. Thus it is not a good idea to send SCI or SDI bits while this register update is in progress.

<sup>6</sup> Firmware changes the value of this register immediately to 0x4800.

Reads from all SCI registers complete in under 100 CLKI cycles, except for SCI_AIADDR, which may take 200 cycles. In addition the cycles spent in the user application/plugin routine must be counted to the read time of SCI_AIADDR, SCI_AUDATA, and SCI_AICTRL0...3.

Some bits in SCI_MODE and SCI_STATUS are hardware bits; other registers only control the firmware. See VS1063 Hardware Guide for details.
9.8.1 \textbf{SCI\_MODE (RW)}

SCI\_MODE is used to control the operation of VS1063a and defaults to 0x0800 (SM\_SDINEW set).

Note: “Mode” in the following table tells if that bit is a hardware (HW) or software (SW) control.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit</th>
<th>Mode</th>
<th>Function</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SM_DIFF</td>
<td>0</td>
<td>SW</td>
<td>Differential</td>
<td>0</td>
<td>normal in-phase audio</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>left channel inverted</td>
</tr>
<tr>
<td>SM_LAYER12</td>
<td>1</td>
<td>SW</td>
<td>Allow MPEG layer II</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>SM_RESET</td>
<td>2</td>
<td>SW</td>
<td>Soft reset</td>
<td>0</td>
<td>no reset</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>reset</td>
</tr>
<tr>
<td>SM_CANCEL</td>
<td>3</td>
<td>SW</td>
<td>Cancel decoding current file</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>SM_RESET</td>
<td>4</td>
<td>SW</td>
<td>reserved</td>
<td>0</td>
<td>right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>wrong</td>
</tr>
<tr>
<td>SM_TESTS</td>
<td>5</td>
<td>SW</td>
<td>Allow SDI tests</td>
<td>0</td>
<td>not allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>allowed</td>
</tr>
<tr>
<td>SM_TESTS</td>
<td>6</td>
<td>SW</td>
<td>reserved</td>
<td>0</td>
<td>right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>wrong</td>
</tr>
<tr>
<td>SM_TESTS</td>
<td>7</td>
<td>SW</td>
<td>reserved</td>
<td>0</td>
<td>right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>wrong</td>
</tr>
<tr>
<td>SM_DACT</td>
<td>8</td>
<td>HW</td>
<td>DCLK active edge</td>
<td>0</td>
<td>rising</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>falling</td>
</tr>
<tr>
<td>SM_SDIORD</td>
<td>9</td>
<td>HW</td>
<td>SDI bit order</td>
<td>0</td>
<td>MSb first</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>MSb last</td>
</tr>
<tr>
<td>SM_DSHARE</td>
<td>10</td>
<td>HW</td>
<td>Share SPI chip select</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>SM_SDINEW</td>
<td>11</td>
<td>HW</td>
<td>VS10xx native SPI modes</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>SM_ENC</td>
<td>12</td>
<td>SW</td>
<td>Activate Encoding</td>
<td>0</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>yes</td>
</tr>
<tr>
<td>-</td>
<td>13</td>
<td>SW</td>
<td>-</td>
<td>0</td>
<td>right</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>wrong</td>
</tr>
<tr>
<td>SM_LINE1</td>
<td>14</td>
<td>HW</td>
<td>MIC / LINE1 selector</td>
<td>0</td>
<td>MICP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>LINE1</td>
</tr>
<tr>
<td>SM_CLK_RANGE</td>
<td>15</td>
<td>HW</td>
<td>Input clock range</td>
<td>0</td>
<td>12\ldots13 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>24\ldots26 MHz</td>
</tr>
</tbody>
</table>

When SM\_DIFF is set, the player inverts the left channel output. For a stereo input this creates virtual surround, and for a mono input this creates a differential left/right signal.

SM\_LAYER12 enables MPEG 1.0 and 2.0 layer II decoding in addition to layer III. **If you enable Layer II decoding, you are liable for any patent issues that may arise.** Joint licensing of MPEG 1.0 / 2.0 Layer III does not cover all patents pertaining to layer II.

Software reset is initiated by setting SM\_RESET to 1. This bit is cleared automatically.

If you want to stop decoding in the middle of a stream, set SM\_CANCEL, and continue sending...
data honouring DREQ. When SM\_CANCEL is detected by a codec, it will stop decoding and return to the main loop. The stream buffer content is discarded and the SM\_CANCEL bit cleared. SCI\_HDAT1 will also be cleared. See Chapter 10.5.2 for details.

If SM\_TESTS is set, SDI tests are allowed. For more details on SDI tests, look at Chapter 10.12.

SM\_DACT defines the active edge of data clock for SDI. When ’0’, data is read at the rising edge, when ’1’, data is read at the falling edge.

When SM\_SDIORD is clear, bytes on SDI are sent MSb first. By setting SM\_SDIORD, the user may reverse the bit order for SDI, i.e. bit 0 is received first and bit 7 last. Bytes are, however, still sent in the default order. This register bit has no effect on the SCI bus.

Setting SM\_SDISHARE makes SCI and SDI share the same chip select, as explained in Chapter 7.1, if also SM\_SDINEW is set.

Setting SM\_SDINEW will activate VS10xx native serial modes as described in Chapters 7.1.1 and 7.3.1. Note, that this bit is set as a default when VS1063a is started up.

By activating SM\_ENCODE and SM\_RESET at the same time, the user will activate the encoding or codec mode (see Chapter 10.7 on page 56). However, note that if the recommended VS1063a Patches package is used (http://www.vlsi.fi/en/support/software/vs10xxplugins.html), then audio encoding is started as instructed in the manual of the package.

SM\_LINE\_IN is used to select the left-channel input for analog input. If ’0’, differential microphone input pins MICP and MICN are used; if ’1’, line-level MICP/LINEIN1 pin is used.

SM\_CLK\_RANGE activates a clock divider in the XTAL input. When SM\_CLK\_RANGE is set, the clock is divided by 2 at the input. From the chip’s point of view e.g. 24 MHz becomes 12 MHz. SM\_CLK\_RANGE should be set as soon as possible after a chip reset.
9.8.2 SCI_STATUS (RW)

SCI_STATUS contains information on the current status of VS1063a. It also controls some low-level things that the user does not usually have to care about.

Note: “Mode” in the following table tells if that bit is a hardware (HW) or software (SW) control.

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Mode</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS_DO_NOT_JUMP</td>
<td>15</td>
<td>SW</td>
<td>Header in decode, do not fast forward/rewind</td>
</tr>
<tr>
<td>SS_SWING</td>
<td>14:12</td>
<td>HW</td>
<td>Set swing to +0 dB, +0.5 dB, . . . , or +3.5 dB</td>
</tr>
<tr>
<td>SS_VCM_OVERLOAD</td>
<td>11</td>
<td>HW</td>
<td>GBUF overload indicator ‘1’ = overload</td>
</tr>
<tr>
<td>SS_VCM_DISABLE</td>
<td>10</td>
<td>HW</td>
<td>GBUF overload detection ‘1’ = disable</td>
</tr>
<tr>
<td>SS_VER</td>
<td>9:4</td>
<td>SW</td>
<td>reserved</td>
</tr>
<tr>
<td>SS_VCM_DISABLE</td>
<td>10</td>
<td>HW</td>
<td>GBUF overload detection ‘1’ = disable</td>
</tr>
<tr>
<td>SS_APDOWN2</td>
<td>3</td>
<td>HW</td>
<td>Analog driver powerdown</td>
</tr>
<tr>
<td>SS_APDOWN1</td>
<td>2</td>
<td>HW</td>
<td>Analog internal powerdown</td>
</tr>
<tr>
<td>SS_AD_CLOCK</td>
<td>1</td>
<td>HW</td>
<td>AD clock select, ‘0’ = 6 MHz, ‘1’ = 3 MHz</td>
</tr>
<tr>
<td>SS_REFERENCE_SEL</td>
<td>0</td>
<td>HW</td>
<td>Reference voltage selection, ‘0’ = 1.23 V, ‘1’ = 1.65 V</td>
</tr>
</tbody>
</table>

SS_DO_NOT_JUMP is set when a WAV, Ogg Vorbis, WMA, MP4, or AAC-ADIF header is being decoded and jumping to another location in the file is not allowed. If you use soft reset or cancel, clear this bit yourself or it can be accidentally left set.

SS_SWING allows you to go above the 0 dB volume setting. Value 0 is normal mode, 1 gives +0.5 dB, and 2 gives +1.0 dB. Settings from 3 to 7 cause the DAC modulator to be overdriven and should not be used. You can use SS_SWING with I2S to control the amount of headroom.

VS1063a contains GBUF protection circuit which disconnects the GBUF driver when too much current is drawn, indicating a short-circuit to ground. SS_VCM_OVERLOAD is high while the overload is detected. SS_VCM_DISABLE can be set to disable the protection feature.

SS_VER is 0 for VS1001, 1 for VS1011, 2 for VS1002, 3 for VS1003, 4 for VS1053 and VS8053, 5 for VS1033, 6 for VS1063, and 7 for VS1103.

SS_APDOWN2 controls analog driver powerdown. SS_APDOWN1 controls internal analog powerdown. These bits are meant to be used by the system firmware only.

If the user wants to powerdown VS1063a with a minimum power-off transient, set SCI_VOL to 0xffff, then wait for at least a few milliseconds before activating reset.

SS_AD_CLOCK can be set to divide the AD modulator frequency by 2 if XTAL1 is in the 24 . . . 26 MHz range.

If AVDD is at least 3.3 V, SS_REFERENCE_SEL can be set to select 1.65 V reference voltage to increase the analog output swing.
9.8.3 SCI_BASS (RW)

<table>
<thead>
<tr>
<th>Name</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ST_AMPLITUDE</td>
<td>15:12</td>
<td>Treble Control in 1.5 dB steps (-8...7, 0 = off)</td>
</tr>
<tr>
<td>ST_FREQLIMIT</td>
<td>11:8</td>
<td>Lower limit frequency in 1000 Hz steps (1...15)</td>
</tr>
<tr>
<td>SB_AMPLITUDE</td>
<td>7:4</td>
<td>Bass Enhancement in 1 dB steps (0...15, 0 = off)</td>
</tr>
<tr>
<td>SB_FREQLIMIT</td>
<td>3:0</td>
<td>Lower limit frequency in 10 Hz steps (2...15)</td>
</tr>
</tbody>
</table>

The Bass Enhancer VSBE is a powerful bass boosting DSP algorithm, which tries to take the most out of the users earphones without causing clipping.

VSBE is activated when SB_AMPLITUDE is non-zero. SB_AMPLITUDE should be set to the user’s preferences, and SB_FREQLIMIT to roughly 1.5 times the lowest frequency the user’s audio system can reproduce. For example setting SCI_BASS to 0x00f6 will have 15 dB enhancement below 60 Hz.

Note: Because VSBE tries to avoid clipping, it gives the best bass boost with dynamical music material, or when the playback volume is not set to maximum. It also does not create bass: the source material must have some bass to begin with.

Treble Control VSTC is activated when ST_AMPLITUDE is non-zero. For example setting SCI_BASS to 0x7a00 will have 10.5 dB treble enhancement at and above 10 kHz.

Bass Enhancer uses about 2.1 MIPS and Treble Control 1.2 MIPS at 44100 Hz samplerate. Both can be on simultaneously.

In VS1063a bass and treble initialization and volume change is delayed until the next batch of samples are sent to the audio FIFO. Thus, unlike with earlier VS10XX chips, audio interrupts can no longer be missed when SCI_BASS or SCI_VOL is written to.

When either the Bass Enhancer or Treble Control is active, the 5-band equalizer (Chapter 10.11.6) is not run.
9.8.4 SCI_CLOCKF (RW)

SCI register SCI_CLOCKF is presented in the table below.

<table>
<thead>
<tr>
<th>SCI_CLOCKF bits</th>
<th>Name</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SC_MULT</td>
<td></td>
<td>15:13</td>
<td>Clock multiplier</td>
</tr>
<tr>
<td>SC_ADD</td>
<td></td>
<td>12:11</td>
<td>Allowed multiplier addition</td>
</tr>
<tr>
<td>SC_FREQ</td>
<td></td>
<td>10:0</td>
<td>Clock frequency</td>
</tr>
</tbody>
</table>

SC_MULT activates the built-in clock multiplier. This will multiply XTALI to create a higher CLKI. When the multiplier is changed by more than $0.5 \times$, the chip runs at $1.0 \times$ clock for a few hundred clock cycles. The values are as follows:

<table>
<thead>
<tr>
<th>SC_MULT</th>
<th>MASK</th>
<th>CLKI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0000</td>
<td>XTALI</td>
</tr>
<tr>
<td>1</td>
<td>0x2000</td>
<td>XTALI $\times 2$</td>
</tr>
<tr>
<td>2</td>
<td>0x4000</td>
<td>XTALI $\times 2.5$</td>
</tr>
<tr>
<td>3</td>
<td>0x6000</td>
<td>XTALI $\times 3.0$</td>
</tr>
<tr>
<td>4</td>
<td>0x8000</td>
<td>XTALI $\times 3.5$</td>
</tr>
<tr>
<td>5</td>
<td>0xa000</td>
<td>XTALI $\times 4.0$</td>
</tr>
<tr>
<td>6</td>
<td>0xc000</td>
<td>XTALI $\times 4.5$</td>
</tr>
<tr>
<td>7</td>
<td>0xe000</td>
<td>XTALI $\times 5.0$</td>
</tr>
</tbody>
</table>

SC_ADD tells how much the decoder firmware is allowed to add to the multiplier specified by SC_MULT if more cycles are temporarily needed to decode a WMA or AAC stream. The values are:

<table>
<thead>
<tr>
<th>SC_ADD</th>
<th>MASK</th>
<th>Multiplier addition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x0000</td>
<td>No modification is allowed</td>
</tr>
<tr>
<td>1</td>
<td>0x0800</td>
<td>1.0$\times$</td>
</tr>
<tr>
<td>2</td>
<td>0x1000</td>
<td>1.5$\times$</td>
</tr>
<tr>
<td>3</td>
<td>0x1800</td>
<td>2.0$\times$</td>
</tr>
</tbody>
</table>

SC_FREQ is used to tell if the input clock XTALI is running at something else than 12.288 MHz. XTALI is set in 4 kHz steps. The formula for calculating the correct value for this register is $\frac{XTALI - 8000000}{4000}$ (XTALI is in Hz).

Note: If SC_FREQ is 0, the default XTALI=12.288 MHz is assumed.

Note: because maximum samplerate is $\frac{XTALI}{256}$, all samplermates are not available if XTALI < 12.288 MHz.

Note: Automatic clock change can only happen when decoding WMA and AAC files. Automatic clock change is done one $0.5 \times$ at a time. This does not cause a drop to $1.0 \times$ clock and you can use the same SCI and SDI clock throughout the file.

Example: If SCI_CLOCKF is 0x8BE8, SC_MULT = 4, SC_ADD = 1 and SC_FREQ = 0x3E8 = 1000. This means that $\frac{XTALI}{256} = 1000 \times \frac{4000 + 8000000}{4000} = 12$ MHz. The clock multiplier is set to $3.5 \times$XTALI = 42 MHz, and the maximum allowed multiplier that the firmware may automatically choose to use is $(3.5 + 1.0) \times$XTALI = 54 MHz.

For how high to set SCI_CLOCKF, see Chapter 10.13, Clock Speed Requirements, on page 80,
9.8.5 SCI_DECODE_TIME (RW)

When decoding correct data, current decoded time is shown in this register in full seconds.

The user may change the value of this register. In that case the new value should be written twice to make absolutely certain that the change is not overwritten by the firmware. A write to SCI_DECODE_TIME also resets the bitRatePer100 calculation.

SCI_DECODE_TIME is reset at every hardware and software reset. It is no longer cleared when decoding of a file ends to allow the decode time to proceed automatically with looped files and with seamless playback of multiple files.

With fast playback (see the playSpeed extra parameter) the decode time also counts faster.

Some codecs (WMA and Ogg Vorbis) can also indicate the absolute play position, see the positionMsec extra parameter in Chapter 10.11.

9.8.6 SCI_AUDATA (RW)

When decoding correct data, the current samplerate and number of channels can be found in bits 15:1 and 0 of SCI_AUDATA, respectively. Bits 15:1 contain the samplerate divided by two, and bit 0 is 0 for mono data and 1 for stereo. Writing to SCI_AUDATA will change the samplerate directly.

Example: 44100 Hz stereo data reads as 0xAC45 (44101).
Example: 11025 Hz mono data reads as 0x2B10 (11024).
Example: Writing 0xAC80 sets samplerate to 44160 Hz, stereo mode does not change.

To reduce digital power consumption when idle, you can write a low samplerate to SCI_AUDATA.

9.8.7 SCI_WRAM (RW)

SCI_WRAM is used to upload application programs and data to instruction and data RAMs. The start address must be initialized by writing to SCI_WRAMADDR prior to the first write/read of SCI_WRAM. One 16-bit data word can be transferred with one SCI_WRAM write/read. As the instruction word is 32 bits long, two consecutive writes/reads are needed for each instruction word. The byte order is big-endian (i.e. most significant words first). After each full-word write/read, the internal pointer is autoincremented.

9.8.8 SCI_WRAMADDR (W)

SCI_WRAMADDR is used to set the program address for following SCI_WRAM writes/reads. Use an address offset from the following table to access X, Y, I or peripheral memory.

<table>
<thead>
<tr>
<th>WRAMADDR Start...End</th>
<th>Dest. addr. Start...End</th>
<th>Bits/Word</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000...0x3FFF</td>
<td>0x0000...0x3FFF</td>
<td>16</td>
<td>X data RAM</td>
</tr>
<tr>
<td>0x4000...0x7FFF</td>
<td>0x0000...0x3FFF</td>
<td>16</td>
<td>Y data RAM</td>
</tr>
<tr>
<td>0x8000...0x8FFF</td>
<td>0x0000...0x0FFF</td>
<td>32</td>
<td>Instruction RAM</td>
</tr>
<tr>
<td>0xC000...0x0C0F</td>
<td>0xC000...0xC0BF</td>
<td>16</td>
<td>I/O</td>
</tr>
<tr>
<td>0xC0C0...0xC0FF</td>
<td>0x1E00...0x1E3F</td>
<td>16</td>
<td>parametric_x</td>
</tr>
<tr>
<td>0xE000...0xFFFF</td>
<td>0xE000...0xFFFF</td>
<td>16</td>
<td>Y data RAM</td>
</tr>
</tbody>
</table>

Note: Unless otherwise specified, only user areas in X, Y, and instruction memory should be accessed.
9.8.9 SCI_HDAT0 and SCI_HDAT1 (R)

For WAV files, SCI_HDAT1 contains 0x7665 ("ve"). SCI_HDAT0 contains the data rate measured in bytes per second for all supported RIFF WAVE formats. To get the bitrate of the file, multiply the value by 8.
Note: if bitrate is over 524280 bit/s, SCI_HDAT1 value saturates to 65535.

For AAC ADTS streams, SCI_HDAT1 contains 0x4154 ("AT"). For AAC ADIF files, SCI_HDAT1 contains 0x4144 ("AD"). For AAC .mp4 / .m4a files, SCI_HDAT1 contains 0x4D34 ("M4"). SCI_HDAT0 contains the average data rate in bytes per second. To get the bitrate of the file, multiply the value by 8.

For WMA files, SCI_HDAT1 contains 0x574M ("WM") and SCI_HDAT0 contains the data rate measured in bytes per second. To get the bitrate of the file, multiply the value by 8.

For Ogg Vorbis files, SCI_HDAT1 contains 0x4F67 "Og". SCI_HDAT0 contains the average data rate in bytes per second. To get the bitrate of the file, multiply the value by 8.

When FLAC format is detected, SCI_HDAT1 contains "fL" (0x664c). SCI_HDAT0 contains the average data rate in byte quadruples per second. To get the bitrate of the file, multiply the value by 32.
For MP3 files, SCI_HDAT1 is between 0xFFE0 and 0xFFFF. SCI_HDAT1 / 0 contain the following:

<table>
<thead>
<tr>
<th>Bit</th>
<th>Function</th>
<th>Value</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>HDAT1[15:5]</td>
<td>syncword</td>
<td>2047</td>
<td>stream valid</td>
</tr>
<tr>
<td>HDAT1[4:3]</td>
<td>ID</td>
<td>3</td>
<td>ISO 11172-3 MPG 1.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>ISO 13818-3 MPG 2.0 (1/2-rate)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>MPG 2.5 (1/4-rate)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>MPG 2.5 (1/4-rate)</td>
</tr>
<tr>
<td>HDAT1[2:1]</td>
<td>layer</td>
<td>3</td>
<td>I</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>II</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>III</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>reserved</td>
</tr>
<tr>
<td>HDAT1[0]</td>
<td>protect bit</td>
<td>1</td>
<td>No CRC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>CRC protected</td>
</tr>
<tr>
<td>HDAT0[15:12]</td>
<td>bitrate</td>
<td>see bitrate table</td>
<td></td>
</tr>
<tr>
<td>HDAT0[11:10]</td>
<td>samplerate</td>
<td>3</td>
<td>reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>32/16/8 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>48/24/12 kHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>44/22/11 kHz</td>
</tr>
<tr>
<td>HDAT0[9]</td>
<td>pad bit</td>
<td>1</td>
<td>additional slot</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>normal frame</td>
</tr>
<tr>
<td>HDAT0[8]</td>
<td>private bit</td>
<td>not defined</td>
<td></td>
</tr>
<tr>
<td>HDAT0[7:6]</td>
<td>mode</td>
<td>3</td>
<td>mono</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>dual channel</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>joint stereo</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>stereo</td>
</tr>
<tr>
<td>HDAT0[5:4]</td>
<td>extension</td>
<td>see ISO 11172-3</td>
<td></td>
</tr>
<tr>
<td>HDAT0[3]</td>
<td>copyright</td>
<td>1</td>
<td>copyrighted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>free</td>
</tr>
<tr>
<td>HDAT0[2]</td>
<td>original</td>
<td>1</td>
<td>original</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>copy</td>
</tr>
<tr>
<td>HDAT0[1:0]</td>
<td>emphasis</td>
<td>3</td>
<td>CCITT J.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
<td>reserved</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>50/15 microsec</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>none</td>
</tr>
</tbody>
</table>

When read, SCI_HDAT0 and SCI_HDAT1 contain header information that is extracted from MP3 stream currently being decoded. After reset both registers are cleared, indicating no data has been found yet.

The “samplerate” field in SCI_HDAT0 is interpreted according to the following table:

<table>
<thead>
<tr>
<th>&quot;samplerate&quot;</th>
<th>ID=3</th>
<th>ID=2</th>
<th>ID=0,1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>32000</td>
<td>16000</td>
<td>8000</td>
</tr>
<tr>
<td>1</td>
<td>48000</td>
<td>24000</td>
<td>12000</td>
</tr>
<tr>
<td>0</td>
<td>44100</td>
<td>22050</td>
<td>11025</td>
</tr>
</tbody>
</table>

The “bitrate” field in HDAT0 is read according to the following table. Notice that for variable bitrate stream the value changes constantly.
### 9 FUNCTIONAL DESCRIPTION

#### Layer II
- ID=3
- ID=0,1,2

#### Layer III
- ID=3
- ID=0,1,2

<table>
<thead>
<tr>
<th>“bitrate”</th>
<th>Layer II kbit/s</th>
<th>Layer III kbit/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ID=3</td>
<td>ID=0,1,2</td>
</tr>
<tr>
<td>15</td>
<td>forbidden</td>
<td>forbidden</td>
</tr>
<tr>
<td>14</td>
<td>384</td>
<td>160</td>
</tr>
<tr>
<td>13</td>
<td>320</td>
<td>144</td>
</tr>
<tr>
<td>12</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>11</td>
<td>224</td>
<td>112</td>
</tr>
<tr>
<td>10</td>
<td>192</td>
<td>96</td>
</tr>
<tr>
<td>9</td>
<td>160</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>112</td>
<td>56</td>
</tr>
<tr>
<td>6</td>
<td>96</td>
<td>48</td>
</tr>
<tr>
<td>5</td>
<td>80</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>56</td>
<td>24</td>
</tr>
<tr>
<td>2</td>
<td>48</td>
<td>16</td>
</tr>
<tr>
<td>1</td>
<td>32</td>
<td>8</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

The average data rate in bytes per second can be read from memory, see the `bitRatePer100` extra parameter. This variable contains the byte rate for all codecs. To get the bitrate of the file, multiply the value by 100, and to get the kilobitrate, make a rounded divide by 10.

The bitrate calculation is not automatically reset between songs, but it can also be reset without a software or hardware reset by writing to `SCI_DECODE_TIME`. 
9.8.10 SCI_AIADDR (RW)

SCI_AIADDR defines the start address of the application/plugin code that has been uploaded earlier with SCI_WRAMADDR and SCI_WRAM registers. If no application code is used, this register should not be written to, or it should be written zero.

Note: Reading SCI_AIADDR is not recommended.

For more details on how to write user applications and plugins, see VS1063 Programmer’s Guide.

9.8.11 SCI_VOL (RW)

SCI_VOL is a volume control for the player hardware. The most significant byte of the volume register controls the left channel volume, the low part controls the right channel volume. The channel volume sets the attenuation from the maximum volume level in 0.5 dB steps. Maximum volume is 0x0000 and total silence but with the output drivers on is 0xFEFE. Setting SCI_VOL to 0xFFFF will activate analog powerdown mode.

<table>
<thead>
<tr>
<th>SCI_VOL bits</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVOL_LEFT</td>
<td>15:8</td>
<td>Left channel attenuation from maximum in 1/2 dB steps</td>
</tr>
<tr>
<td>SVOL_RIGHT</td>
<td>7:0</td>
<td>Right channel attenuation from maximum in 1/2 dB steps</td>
</tr>
</tbody>
</table>

Example: for a volume of -2.0 dB for the left channel and -3.5 dB for the right channel: \((2.0/0.5) = 4, 3.5/0.5 = 7 \rightarrow SCI\_VOL = 0x0407.\)

Example: SCI\_VOL = 0x2424 → both left and right volumes are \(0x24 \times -0.5 = -18.0\ dB\)

In VS1063a bass and treble initialization and volume change is delayed until the next batch of samples are sent to the audio FIFO. This delays the volume setting slightly. The hardware volume control has zero-cross detection, which almost completely removes all audible noise that occurs when volume is changed.

Note: After hardware reset the volume is set to full volume. Resetting the software does not reset the volume setting.
9.8.12 SCI_AICTRL[x] (RW)

SCI_AICTRL[x] registers (x=[0...3]) can be used to access the user’s application(plugin program).

The SCI_AICTRL registers are also used as parameter registers when encoding audio. See Chapter 10.7 for details.

For more details on how to write user applications, see VS1063 Programmer’s Guide.
10 Operation

10.1 Clocking

VS1063a operates on a single, nominally 12.288 MHz fundamental frequency master clock. This clock can be generated by external circuitry (connected to pin XTALI) or by the internal clock crystal interface (pins XTALI and XTALO). This clock is used by the analog parts and determines the highest available samplerate. With 12.288 MHz clock all samplerates up to 48000 Hz are available.

VS1063a can also use 24...26 MHz clocks, but in this case SM_CLK_RANGE in the SCI_MODE register has to be set to 1 after startup. The system clock is then internally divided by 2 at the clock input and the IC gets a 12...13 MHz input clock.

10.2 Hardware Reset

When the XRESET signal is driven low, VS1063a is reset and all the control registers and internal states are set to the initial values. XRESET-signal is asynchronous to any external clock. The reset mode doubles as a full-powerdown mode, where both digital and analog parts of VS1063a are in minimum power consumption stage, and where clocks are stopped. Also XTALO is grounded.

When XRESET is asserted, all output pins go to their default states. All input pins will go to high-impedance state (input state), except SO, which is still controlled by XCS.

After a hardware reset (or at power-up) DREQ will stay down for around 22000 clock cycles, which means an approximate 1.8 ms delay if VS1063a is run at 12.288 MHz. After this the user should set such basic software registers as SCI_MODE, SCI_BASS, SCI_CLOCKF, and SCI_VOL before starting decoding. See Chapter 9.8 for details.

If the input clock is 24...26 MHz, SM_CLK_RANGE should be set as soon as possible after a chip reset without waiting for DREQ.

Internal clock can be multiplied with a PLL. Supported multipliers through the SCI_CLOCKF register are $1.0 \times \ldots 5.0 \times$ the input clock. Reset value for Internal Clock Multiplier is $1.0 \times$. Wait until DREQ rises, then write e.g. value 0xB000 to SCI_CLOCKF (register 3). See Chapters 9.8.4 and 10.13 for details on good values for SCI_CLOCKF.

Before VS1063a is used it is recommended to load and run the current VS1063a Patches package. It is available at http://www.vlsi.fi/en/support/software/ vs10xxplugins.html.
10.3 Software Reset

In some cases the decoder software has to be reset. This is done by activating bit SM_RESET in register SCI_MODE (Chapter 9.8.1). Then wait for at least 2 $\mu$s, then look at DREQ. DREQ will stay down for about 22000 clock cycles, which means an approximate 1.8 ms delay if VS1063a is run at 12.288 MHz. When DREQ goes high, you may continue playback as usual.

As opposed to all earlier VS10XX chips, it is not recommended to do a software reset between songs. This way the user may be sure that even files with low samplerates or bitrates are played right to their end.

After each software reset it is recommended to load and run the current VS1063a Patches package. It is available at [http://www.vlsi.fi/en/support/software/vs10xxplugins.html](http://www.vlsi.fi/en/support/software/vs10xxplugins.html).

10.4 Low Power Mode

If you need to keep the system running while not decoding data, but need to lower the power consumption, you can use the following tricks.

- Select the 1.0× clock by writing 0x0000 to SCI_CLOCKF. This disables the PLL and saves some power.
- Write a low non-zero value, such as 0x0010 to SCI_AUDATA. This will reduce the samplerate and the number of audio interrupts required. Between audio interrupts the VSDSP core will just wait for an interrupt, thus saving power.
- Turn off all audio post-processing (tone controls, EarSpeaker, etc).
- If possible for the application, write 0xffff to SCI_VOL to disable the analog drivers.

To return from low-power mode, revert register values in reverse order.

Note: The low power mode consumes significantly more electricity than hardware reset.

10.5 Play and Decode

This is the normal operation mode of VS1063a. SDI data is decoded. Decoded samples are converted to analog domain by the internal DAC. If no decodable data is found, SCI_HDAT0 and SCI_HDAT1 are set to 0.

When there is no input for decoding, VS1063a goes into idle mode (lower power consumption than during decoding) and actively monitors the serial data input for valid data.
10.5.1 Playing a Whole File

This is the default playback mode.

1. Send an audio file to VS1063a.
2. Read extra parameter value endFillByte (Chapter 10.11).
3. Send at least 2052 bytes of endFillByte[7:0]. For FLAC you should send 12288 endFillBytes when ending a file.
4. Set SCI_MODE bit SM_CANCELS.
5. Send at least 32 bytes of endFillByte[7:0].
6. Read SCI_MODE. If SM_CANCELS is still set, go to 5. If SM_CANCELS hasn't cleared after sending 2048 bytes, do a software reset (this should be extremely rare).
7. The song has now been successfully sent. HDAT0 and HDAT1 should now both contain 0 to indicate that no format is being decoded. Return to 1.

10.5.2 Cancelling Playback

Cancelling playback of a song is a normal operation when the user wants to jump to another song while doing playback.

1. Send a portion of an audio file to VS1063a.
2. Set SCI_MODE bit SM_CANCELS.
3. Continue sending audio file, but check SM_CANCELS after every 32 bytes of data. If it is still set, goto 3. If SM_CANCELS doesn't clear after 2048 bytes or one second, do a software reset (this should be extremely rare).
4. When SM_CANCELS has cleared, read extra parameter value endFillByte (Chapter 10.11).
5. Send 2052 bytes of endFillByte[7:0]. For FLAC you should send 12288 endFillBytes.
6. HDAT0 and HDAT1 should now both contain 0 to indicate that no format is being decoded. You can now send the next audio file.

10.5.3 Fast Play

VS1063a allows fast audio playback. If your microcontroller can feed data fast enough to the VS1063a, this is the preferred way to fast forward audio.

1. Start sending an audio file to VS1063a.
2. To set fast play, set extra parameter value playSpeed (Chapter 10.11).
3. Continue sending audio file.
4. To exit fast play mode, write 1 to playSpeed.
To estimate whether or not your microcontroller can feed enough data to VS1063a in fast play mode, see contents of extra parameter value bitRatePer100 (Chapter 10.11). Note that bitRatePer100 contains the data speed of the file played back at nominal speed even when fast play is active.

Note: Play speed is not reset when song is changed.

### 10.5.4 Fast Forward and Rewind without Audio

To do fast forward and rewind you need the capability to do random access to the audio file. Unfortunately fast forward and rewind isn’t available at all times, like when file headers are being read.

1. Send a portion of an audio file to VS1063a.
2. When random access is required, read SCI_STATUS bit SS_DO_NOT_JUMP. If that bit is set, random access cannot be performed, so go back to 1.
3. Read extra parameter value endFillByte (Chapter 10.11).
4. Send at least 2048 bytes of endFillByte[7:0].
5. Jump forwards or backwards in the file.
6. Continue sending the file.

Note: It is recommended that playback volume is decreased by e.g. 10 dB when fast forwarding/rewinding.

Note: Register DECODE_TIME does not take jumps into account.

### 10.5.5 Maintaining Correct Decode Time

When fast forward and rewind operations are performed, there is no way to maintain correct decode time for most files. However, WMA and Ogg Vorbis files offer exact time information in the file. To use accurate time information whenever possible, use the following algorithm:

1. Start sending an audio file to VS1063a.
2. Read extra parameter value pair positionMsec (Chapter 10.11).
3. If positionMsec is -1, show you estimation of decoding time using DECODE_TIME (and your estimate of file position if you have performed fast forward / rewind operations).
4. If positionMsec is not -1, use this time to show the exact position in the file.
10.6 Feeding PCM Data

VS1063a can be used as a PCM decoder by sending a WAV file header, followed by PCM data. If the length sent in the WAV header is 0xFFFFFFFF, VS1063a will stay in PCM mode indefinitely (or until SM_CANCEL has been set). 8-bit linear and 16-bit linear audio is supported in mono or stereo. A WAV header looks like this:

<table>
<thead>
<tr>
<th>File Offset</th>
<th>Field Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ChunkID</td>
<td>4</td>
<td>&quot;RIFF&quot;</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>ChunkSize</td>
<td>4</td>
<td>0xff 0xff 0xff 0xff</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Format</td>
<td>4</td>
<td>&quot;WAVE&quot;</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>SubChunk1ID</td>
<td>4</td>
<td>&quot;fmt &quot;</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>SubChunk1Size</td>
<td>4</td>
<td>0x10 0x0 0x0 0x0</td>
<td>16</td>
</tr>
<tr>
<td>20</td>
<td>AudioFormat</td>
<td>2</td>
<td>0x1 0x0</td>
<td>Linear PCM</td>
</tr>
<tr>
<td>22</td>
<td>NumOfChannels</td>
<td>2</td>
<td>C0 C1</td>
<td>1 for mono, 2 for stereo</td>
</tr>
<tr>
<td>24</td>
<td>SampleRate</td>
<td>4</td>
<td>S0 S1 S2 S3</td>
<td>0x1f40 for 8 kHz</td>
</tr>
<tr>
<td>28</td>
<td>ByteRate</td>
<td>4</td>
<td>R0 R1 R2 R3</td>
<td>0x3e80 for 8 kHz 16-bit mono</td>
</tr>
<tr>
<td>32</td>
<td>BlockAlign</td>
<td>2</td>
<td>A0 A1</td>
<td>0x02 0x00 for mono, 0x04 0x00 for stereo 16-bit</td>
</tr>
<tr>
<td>34</td>
<td>BitsPerSample</td>
<td>2</td>
<td>B0 B1</td>
<td>0x10 0x00 for 16-bit data</td>
</tr>
<tr>
<td>52</td>
<td>SubChunk2ID</td>
<td>4</td>
<td>&quot;data&quot;</td>
<td></td>
</tr>
<tr>
<td>56</td>
<td>SubChunk2Size</td>
<td>4</td>
<td>0xff 0xff 0xff 0xff</td>
<td>Data size</td>
</tr>
</tbody>
</table>

The rules to calculate the four variables are as follows:

- \( S = \) sample rate in Hz, e.g. 44100 for 44.1 kHz.
- For 8-bit data \( B = 8 \), and for 16-bit data \( B = 16 \).
- For mono data \( C = 1 \), for stereo data \( C = 2 \).
- \( A = \frac{C \times B}{8} \).
- \( R = S \times A \).

Note: When playing back PCM, VS1063a ignores \( R \) and \( A \). You may set them to anything if you don’t intend the data streams to be sent to any other devices.

Example: A 44100 Hz 16-bit stereo PCM header would read as follows:

```
0000 52 49 46 46 ff ff ff ff 57 41 56 45 66 6d 74 20 |RIFF...WAV...|
0010 10 00 00 00 01 00 02 00 44 ac 00 00 10 b1 02 00 |......D......|
0020 04 00 10 00 64 61 74 61 ff ff ff ff |...data.....|
```
10.7 Audio Encoding

This chapter explains how to use the encoding and codec modes of VS1063a.

VS1063 has a stereo ADC, thus also two-channel (separate AGC, if AGC enabled) and stereo (common AGC, if AGC enabled) modes are available. Mono encoding can select either left or right channel, or a mono down-mix of the left and right channels. The left channel is either MIC or LINE1 depending on the SCI_MODE register, the right channel is LINE2.

10.7.1 Encoding Control Registers

<table>
<thead>
<tr>
<th>Register</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCI_MODE</td>
<td>2, 12, 14</td>
<td>Start ENCODE mode, select MIC/LINE1</td>
</tr>
<tr>
<td>SCI_AICTRL0</td>
<td>15:0</td>
<td>Samplerate 8000…48000 Hz (read at encoding startup)</td>
</tr>
<tr>
<td>SCI_AICTRL1</td>
<td>15:0</td>
<td>Encoding gain (1024 = 1 ×) or 0 for automatic gain control</td>
</tr>
<tr>
<td>SCI_AICTRL2</td>
<td>15:0</td>
<td>Maximum autogain amplification (1024 = 1 ×, 65535 = 64 ×)</td>
</tr>
<tr>
<td>SCI_AICTRL3</td>
<td>15</td>
<td>codec mode (both encode and decode)</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>AEC enable</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>UART TX enable</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>reserved, set to 0</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>Pause enable</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>No RIFF WAV header inserted (or expected in codec mode)</td>
</tr>
<tr>
<td></td>
<td>8:9</td>
<td>reserved, set to 0</td>
</tr>
<tr>
<td></td>
<td>7:4</td>
<td>Encoding format 0…6</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>Reserved, set to 0</td>
</tr>
<tr>
<td></td>
<td>2:0</td>
<td>ADC mode 0…4</td>
</tr>
<tr>
<td>SCI_WRAMADDR</td>
<td>15…0</td>
<td>Quality / bitrate selection for Ogg Vorbis and MP3</td>
</tr>
</tbody>
</table>

If you use the VS1063a Patches package (highly recommended), activate encoding mode by first setting the bit SM_ENCODE in register SCI_MODE, then writing 0x50 to SCI_AIADDR. Otherwise, activate encoding by setting bits SM_RESET and SM_ENCODE in SCI_MODE.

Line input 1 is used instead of differential mic input if SM_LINE1 is set. Before activating encoding, user must write the right values to SCI_AICTRL0, SCI_AICTRL3, and SCI_WRAMADDR. These values are only read at encoding startup. SCI_AICTRL1 and SCI_AICTRL2 can be altered anytime, but it is preferable to write good init values before activation.

SCI_AICTRL1 controls linear encoding gain. The gain is \( \frac{AICTRL1}{1024} \), so 1024 is equal to digital gain 1.0, 2000 is 1.95, 512 is 0.5 and so on. If the user wants to use automatic gain control (AGC), SCI_AICTRL1 should be set to 0. Typical speech applications usually are better off using AGC, as this takes care of relatively uniform speech loudness in encodings.

SCI_AICTRL2 controls the maximum AGC gain. This can be used to limit the amplification of noise when there is no signal. If SCI_AICTRL2 is zero, the maximum gain is initialized to 65535 (64 ×), i.e. whole range is used.
If SCI_AICTRL3 bit 15 is set at startup, codec mode is initialized. If MP3 and Ogg Vorbis formats are specified, the configuration bit is ignored and codec mode is not available. In codec mode the encoded data is provided through HDAT0 and HDAT1, and data to be decoded is expected through the serial data interface (SDI).

If SCI_AICTRL3 bit 13 is set at encode/codec startup, UART transmission of data is initialized with parameters taken from parametric_x. If you want to use UART transmission, you must first write the tx configuration values, then set the AICTRL3 UART TX enable bit, and only after that start the encoding/codec mode using SCI_MODE register.

UART is configured from parametric_x.i.encoding, that part of of the parametric structure is no longer cleared at software reset.

<table>
<thead>
<tr>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:14</td>
<td>Bitrate mode, 0 = Quality Mode, 1 = VBR, 2 = ABR, 3 = CBR</td>
</tr>
<tr>
<td>13:12</td>
<td>Bitrate multiplier, 0 = 10, 1 = 100, 2 = 1000, 3 = 10000</td>
</tr>
<tr>
<td>11</td>
<td>Encoder-specific, Ogg Vorbis: 1=use parametric_x.i.encoding.serialNumber</td>
</tr>
<tr>
<td>10</td>
<td>Encoder-specific, Ogg Vorbis: 1=limited frame length</td>
</tr>
<tr>
<td></td>
<td>mp3: 1 = do not use bit-reservoir</td>
</tr>
<tr>
<td>9</td>
<td>Used internally, set to 0.</td>
</tr>
<tr>
<td>8:0</td>
<td>Bitrate base 0 to 511 (or quality 0 to 9 if Quality Mode selected).</td>
</tr>
</tbody>
</table>

The bitrate base and bitrate multiplier define a target bitrate value. For example 2 in multiplexer and 128 in base means 128 kbit/s. The bitrate mode selects how the bitrate and bitrate multiplier fields are interpreted. In variable bitrate (VBR) mode the bitrate and bitrate multiplier fields sets a very relaxed average bitrate. Currently the average bitrate mode (ABR) equals VBR mode in both encoders. In Quality Mode bitrate multiplier is ignored and the bitrate base field value sets encoding quality from 0 to 9.

SCI_WRAMADDR bit 10 is encoder-specific. When set with the MP3 encoder, the bit reservoir will not be used. When set with Ogg Vorbis encoder, the bit requests a smaller output delay.

SCI_WRAMADDR bit 11 is encoder-specific. When set with Ogg Vorbis encoder, the stream serial number is fetched from parametric_x.i.encoding.serialNumber.

Note: When recording, parametric_x.i.encoding.channelMax contains the maximum absolute value encountered in the corresponding channel since the last clear of the variable. In mono modes only channelMax[0] is updated.
10.7.2 The Encoding Procedure

The encoding procedure from start to finish goes as follows:

1. Pre-initialization; Setup system:
   - Load the VS1063a Patches package, available at http://www.vlsi.fi/en/support/software/vs10xxplugins.html. Note that the package is required for conforming MP3 bitstreams: without it recording quality will be significantly lower and the bitstream may have errors.

2. Initialization; Set samplerate and parameters:
   - SCI_CLOCKF for clock: $4.5 \times$ for all encoders except Ogg Vorbis (Chapters 10.7.9 and 10.13).
   - SCI_AICTRL0 for samplerate (SCI_WRAMADDR for bitrate/quality setting)
   - SCI_AICTRL1 for gain/AGC
   - SCI_AICTRL2 for AGC max gain
   - SCI_AICTRL3 for channel selection, encoding format and options
   - If used, fill in UART configuration.
   - If used, fill in Ogg Vorbis serial number.
   - SCI_WRAMADDR to set bitrate/quality (mp3 and Ogg Vorbis only)
   - Activate encoding with one of the two ways:
     - Recommended: If you use the VS1063a Patches package, start encoding as follows: set bit SM_ENCODE in SCI_MODE, then write 0x50 to SCI_AIADDR.
     - If you do not use the VS1063a Patches package, start encoding mode by setting SM_ENCODE and SM_SOFTRESET in register SCI_MODE.

3. Recording:
   - Depending on whether you selected SCI or UART data transfers with bit 13 of SCI_AICTRL3, read data through SCI_HDAT0/SCI_HDAT1 as described in Chapter 10.7.3, or through the UART.

4. Finalizing recording:
   - When you want to finish encoding a file, set bit SM_CANCEL in SCI_MODE.
   - After a while (typically less than 100 ms), SM_CANCEL will be cleared by VS1063a.
   - If using SCI for data transfers, read all remaining words using SCI_HDAT1/SCI_HDAT0. Then read parametric_x.endFillByte. If the most significant bit (bit 15) is set to 1, then the file is of an odd length and bits 7:0 contain the last byte that still should be written to the output file. Now write 0 to endFillByte.
   - When all samples have been transmitted, SM_ENCODE bit of SCI_MODE will be cleared by VS1063a, and SCI_HDAT1 and SCI_HDAT0 are cleared.

5. Now you can give a software reset to enter player mode or start encoding again.
Example of Encoding initialization including loading *VS1063a Patches* (recommended):

```c
// First command line loads VS1063a Patches. The patch package can be
// This package is required for best MP3 quality and correct monitoring.
LoadUserCode(vs1063apatch);
WriteVS10xxRegister(SCI_AICTRL0, 48000U); // 48 kHz
WriteVS10xxRegister(SCI_AICTRL1, 1024U); // Manual gain at 1.0x
WriteVS10xxRegister(SCI_AICTRL3, 0x60); // Stereo MP3
WriteVS10xxRegister(SCI_WRAMADDR, 0xE0A0); // Set bitrate to CBR 160 kbit/s
WriteVS10xxRegister(SCI_MODE, ReadVS10xxRegister(SCI_MODE) |
                      SM_ENCODE | SM_LINE1); // Set record parameters
WriteVS10xxRegister(SCI_AIADDR, 0x50); // Activate recording
```

The previous code sets 48 kHz stereo MP3 recording with manual gain control set at $1 \times$ (0 dB).

Example of initialization without loading *VS1063a Patches* (not recommended):

```c
WriteVS10xxRegister(SCI_AICTRL0, 16000U); // 16 kHz
WriteVS10xxRegister(SCI_AICTRL1, 0); // Manual gain 0 = AGC on
WriteVS10xxRegister(SCI_AICTRL2, 4096U); // AGC max gain 4.0x
WriteVS10xxRegister(SCI_AICTRL3, 2); // Mono IMA ADPCM
WriteVS10xxRegister(SCI_MODE, ReadVS10xxRegister(SCI_MODE) | SM_RESET | SM_ENCODE) &
                   ~SM_LINE1); // Microphone input, activate
```

The previous code sets 16 kHz mono IMA ADPCM recording from the left channel using the microphone amplifier, with automatic gain control and maximum amplification of $4 \times$ (= +12 dB).

### 10.7.3 Reading Encoded Data Through SCI

After encoding mode has been activated, registers SCI_HDAT0 and SCI_HDAT1 have new functions.

The encoding data buffer is 3712 16-bit words. The fill status of the buffer can be read from SCI_HDAT1. If SCI_HDAT1 is greater than 0, you can read that many 16-bit words from SCI_HDAT0. If the data is not read fast enough, the buffer overflows and returns to empty state.

The encoded data is read from SCI_HDAT0 and written into file as follows. The high 8 bits of SCI_HDAT0 should be written as the first byte to a file, then the low 8 bits. Note that this is contrary to the default operation of some 16-bit microcontrollers, and you may have to take extra care to do this right.
10.7.4 File Headers

VS1063 automatically creates a suitable header for the selected encoding mode. If you have selected MP3 or Ogg Vorbis, the headers will be in those formats, otherwise you get a RIFF WAV header with the correct samplerate, number of channels, and other information. (If you have set bit 10 of SCI_AICTRL3, the RIFF WAV header is not generated.) When you finish encoding you have to fix the RIFF size and data size fields.

The following shows a header for a 8 kHz mono $\mu$-law WAV file. Note that 2- and 4-byte values are little-endian (least significant byte first).

```
00000000 52 49 46 46 ff ff ff ff 57 41 56 66 6d 74 20 |RIFF ..WAVEfmt |
00000010 14 00 00 00 07 00 01 00 40 1f 00 00 40 1f 00 00 |........@...@...|
00000020 01 00 08 00 02 00 01 00 64 61 74 61 ff ff ff ff |........data, ..|
```

The following shows a header for a 8 kHz mono $\mu$-law WAV file. Note that 2- and 4-byte values are little-endian (least significant byte first).

```
00000000 52 49 46 46 ff ff ff ff 57 41 56 66 6d 74 20 |RIFF ..WAVEfmt |
00000010 14 00 00 00 07 00 01 00 40 1f 00 00 40 1f 00 00 |........@...@...|
00000020 01 00 08 00 02 00 01 00 64 61 74 61 ff ff ff ff |........data, ..|
```

The following shows a header for a 8 kHz mono $\mu$-law WAV file.

<table>
<thead>
<tr>
<th>File Offset</th>
<th>Field Name</th>
<th>Size</th>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>ChunkID</td>
<td>4</td>
<td>&quot;RIFF&quot;</td>
<td>RIFF ident</td>
</tr>
<tr>
<td>4</td>
<td>ChunkSize</td>
<td>4</td>
<td>F0 F1 F2 F3</td>
<td>File size - 8</td>
</tr>
<tr>
<td>8</td>
<td>Format</td>
<td>4</td>
<td>&quot;WAVE&quot;</td>
<td>WAVE ident</td>
</tr>
<tr>
<td>12</td>
<td>SubChunk1ID</td>
<td>4</td>
<td>&quot;fmt &quot;</td>
<td>fmt ident</td>
</tr>
<tr>
<td>16</td>
<td>SubChunk1Size</td>
<td>4</td>
<td>0x14 0x0 0x0 0x0</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>AudioFormat</td>
<td>2</td>
<td>0x07 0x0</td>
<td>Audio format</td>
</tr>
<tr>
<td>22</td>
<td>NumOfChannels</td>
<td>2</td>
<td>0x01 0x00</td>
<td>1 for mono, 2 for stereo</td>
</tr>
<tr>
<td>24</td>
<td>SampleRate</td>
<td>4</td>
<td>0x40 0x1f 0x00 0x00</td>
<td>0x1f40 = 8000 Hz</td>
</tr>
<tr>
<td>28</td>
<td>ByteRate</td>
<td>4</td>
<td>0x40 0x1f 0x00 0x00</td>
<td>Bytes per second</td>
</tr>
<tr>
<td>32</td>
<td>BlockAlign</td>
<td>2</td>
<td>0x01 0x00</td>
<td>1 byte per block</td>
</tr>
<tr>
<td>34</td>
<td>BitsPerSample</td>
<td>2</td>
<td>0x08 0x00</td>
<td>8 bits / sample</td>
</tr>
<tr>
<td>36</td>
<td>Extra size</td>
<td>2</td>
<td>0x02 0x00</td>
<td>2 extra bytes</td>
</tr>
<tr>
<td>38</td>
<td>Samples per block</td>
<td>2</td>
<td>0x01 0x00</td>
<td>1 sample per block</td>
</tr>
<tr>
<td>40</td>
<td>SubChunk3ID</td>
<td>4</td>
<td>&quot;data&quot;</td>
<td>Data ident</td>
</tr>
<tr>
<td>44</td>
<td>SubChunk3Size</td>
<td>4</td>
<td>D0 D1 D2 D3</td>
<td>Data size (File Size-48)</td>
</tr>
<tr>
<td>48</td>
<td>Samples...</td>
<td></td>
<td></td>
<td>data</td>
</tr>
</tbody>
</table>

Because VS1063a cannot know in advance how long the recording will be, it will set both RIFF ChunkSize and Data SubChunk3Size fields $F$ and $D$ to 0xFFFFFFFF. You have to fill in correct values for $F$ and $D$ after finishing encoding.

Below is an example of a valid header for a 44.1 kHz mono PCM file that has a final length of 1798772 (0x1B7274) bytes:

```
0000 52 49 46 46 46 46 6c 72 1b 00 |RIFFlr ..WAVEfmt |
0010 14 00 00 00 01 00 01 00 80 bb 00 00 00 77 01 00 |...........w...|
0020 02 00 10 00 02 00 01 00 64 61 74 61 44 72 1b 00 |........dataDr..|
```
10.7.5 Playing Encoded Data

In order to play back your encoding, all you need to do is to provide the file through SDI as you would with any audio file.

10.7.6 Encoder Samplerate Considerations

For encoder samplerates to work accurately, it is recommended to load and run the VS1063a Patches package. It is available at http://www.vlsi.fi/en/support/software/vs10xxplugins.html.

When the VS1063a Patches package, v1.2 or higher, is installed, then almost all recording samplerates for almost all encoders can be represented accurately. The only exception is Stereo Ogg Vorbis recording at over 32 kHz, in which case recording speed may not be accurate.

Below is a encoding samplerate accuracy table for all standard MP3 samplerates, with nominal crystal speed XTALI = 12.288 MHz.

<table>
<thead>
<tr>
<th>Requested $f_s$</th>
<th>Actual $f_s$</th>
<th>Error</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>48000 Hz</td>
<td>48000 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
<tr>
<td>44100 Hz</td>
<td>44100 Hz</td>
<td>0.00 %</td>
<td>All except Ogg Vorbis stereo.</td>
</tr>
<tr>
<td>44100 Hz</td>
<td>44201 Hz</td>
<td>+0.23 %</td>
<td>Ogg Vorbis stereo; not recommended for streaming.</td>
</tr>
<tr>
<td>32000 Hz</td>
<td>32000 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
<tr>
<td>24000 Hz</td>
<td>24000 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
<tr>
<td>22050 Hz</td>
<td>22050 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
<tr>
<td>16000 Hz</td>
<td>16000 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
<tr>
<td>12000 Hz</td>
<td>12000 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
<tr>
<td>11025 Hz</td>
<td>11025 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
<tr>
<td>8000 Hz</td>
<td>8000 Hz</td>
<td>0.00 %</td>
<td></td>
</tr>
</tbody>
</table>

10.7.7 Encode Monitoring Volume

In VS1063a writing to the SCI_VOL register during encoding mode will update monitoring volume.
10.7.8 MP3 (format 5) Encoder Specific Considerations

The MP3 encoder supports all bitrates and samplerates of the MP3 format, both in mono and stereo. For details of supported and recommended modes, see Chapter 8.2.1. Notice particularly that only the MP3 official samplerates are supported (8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100 and 48000 Hz). If you try to start MP3 encoding with any other samplerate, the encoder will silently fail.

Quality mode, VBR, CBR are the main modes supported by the encoder. If ABR is selected, VBR mode is used instead. When Quality mode is selected, 5 is designed to be "near PCM quality" for the given samplerate.

The so-called MP3 bit reservoir offers a way to more efficiently encode MP3 files. To make streaming more resilient to transmission errors, encoder only makes bit reservoir references one frame back.

For some streaming applications it may be beneficial to turn the bit reservoir off by setting bit 10 of register SCI_WRAMADDR before activating encoding. This will make frames more self-contained. When using ABR/VBR/Quality encoding, turning bit reservoir off will increase the bitrate approximately 4...16 kbit/s. Turning bit reservoir off in CBR mode is strongly discouraged as it will have a huge impact in quality and coding efficiency.

10.7.9 Ogg Vorbis (format 6) Encoder Specific Considerations

The Ogg Vorbis encoder supports a wide range of bitrates and all samplerates at 8...48 kHz, in mono and stereo. For some examples of supported modes, see Chapter 8.2.2.

Quality mode is the main mode supported by the encoder. If VBR is selected, the value is internally converted to a quality value between 0...9, and this value is used. If ABR or CBR is selected, VBR mode is used instead. When Quality mode is selected, 5 is designed to be "near PCM quality" for the given samplerate.

When silence is detected, the bitstream width may be reduced by up to 90%. Because the encoder attempts to make Ogg frames as long as possible (upto 4 KiB), this means that in such a case the frame delay may grow dramatically, which may cause problems for streaming systems. To avoid this, the user may set register SCI_WRAMADDR bit 10 before activating encoding. This will instruct the encoder to create a frame always after at least 1024 but not more than 2048 samples have been generated in an Ogg frame.

The Ogg stream default serial number is 0xfecaadab. If the user wants to change it, he should, before activating encoding, write to parametric.x.i.encoding.serialNumber (Chapter 10.11) and set bit 11 of register SCI_WRAMADDR.

If you run the Ogg Vorbis encoder in stereo, and with a samplerate of 40 kHz or higher, set the clock multiplier register SCI_CLOCKF to at least $5.0 \times$ (e.g. SCI_CLOCKF = 0xe000). Otherwise SCI_CLOCKF = $4.5 \times$ is enough.
10.7.10 Encoder/Decoder Delays

This Chapter presents the absolute minimum estimated encoder/decoder total delays between two VS1063a ICs. In addition to these numbers come all data transfer times from the transmitting to the receiving unit.

The following symbols are used:
- $f_s$ = samplerate
- $d_{m}$ = minimum encoder/decoder delay in milliseconds

Note! Delays have been calculated for standard MP3 samplrates. Other encoders can also encode non-standard samplersates upto 48 kHz.

<table>
<thead>
<tr>
<th>$f_s$ / Hz</th>
<th>PCM/G.711/G.722 / ms</th>
<th>IMA / ms</th>
<th>MP3 / ms</th>
<th>Ogg / ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>48000</td>
<td>3</td>
<td>14</td>
<td>36</td>
<td>124</td>
</tr>
<tr>
<td>44100</td>
<td>3</td>
<td>15</td>
<td>40</td>
<td>135</td>
</tr>
<tr>
<td>32000</td>
<td>3</td>
<td>19</td>
<td>54</td>
<td>185</td>
</tr>
<tr>
<td>24000</td>
<td>3</td>
<td>25</td>
<td>48</td>
<td>125</td>
</tr>
<tr>
<td>22050</td>
<td>3</td>
<td>26</td>
<td>52</td>
<td>140</td>
</tr>
<tr>
<td>16000</td>
<td>3</td>
<td>35</td>
<td>72</td>
<td>190</td>
</tr>
<tr>
<td>12000</td>
<td>3</td>
<td>46</td>
<td>96</td>
<td>250</td>
</tr>
<tr>
<td>11025</td>
<td>3</td>
<td>49</td>
<td>105</td>
<td>270</td>
</tr>
<tr>
<td>8000</td>
<td>3</td>
<td>66</td>
<td>144</td>
<td>200</td>
</tr>
</tbody>
</table>

1 Numbers apply if “limited frame length” (bit 10 of register SCI_WRAMADDR) is set. If the bit is not set, encoder/decoder delay can be upto several seconds. See Chapter 10.7.1, Encoding Control Registers, for details on how to set the “limited frame length” bit.

10.8 Codec Mode

In codec mode VS1063a can encode and decode separate signal paths at the same time.

However, there are some restrictions in codec mode. The samplerate should be XTALI/256, XTALI/512, XTALI/1024 or XTALI/1536 (with XTALI = 12.288 MHz, supported samplrates are 48000, 24000, 12000 or 8000 Hz). MP3 and Ogg Vorbis formats are not available.

A RIFF WAV header is automatically generated in the encoded data. Encoded data is read through SCI_HDAT1 and SCI_HDAT0 like in encoding mode (except if UART mode is used).

The data to be decoded is sent to SDI like in decoding mode. The format, number of channels and samplerate are determined from a RIFF WAV header. If you have set bit 10 of SCI_AICTRL3, the RIFF WAV header is not expected and the format, number of channels and rate are set to the ones used in encoding.

If you use Codec Mode at its largest data transfer rates, i.e. 48 kHz 16-bit PCM written through SDI and read through SCI, you have to set the clock multiplier SCI_CLOCKF to 5.0×.

Note: the RIFF WAV parser used in the codec mode decoder is a simplified one.
10.9 SPI Boot

If GPIO0 is set with a pull-up resistor to 1 at boot time, VS1063a tries to boot from external SPI memory.

SPI boot redefines the following pins:

<table>
<thead>
<tr>
<th>Normal Mode</th>
<th>SPI Boot Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO0</td>
<td>xCS</td>
</tr>
<tr>
<td>GPIO1</td>
<td>CLK</td>
</tr>
<tr>
<td>DREQ</td>
<td>MOSI</td>
</tr>
<tr>
<td>GPIO2</td>
<td>MISO</td>
</tr>
</tbody>
</table>

The memory has to be an SPI Bus Serial EEPROM with 16-bit or 24-bit addresses. The serial speed used by VS1063a is 245 kHz with the nominal 12.288 MHz clock. The first three bytes in the memory have to be 0x50, 0x26, 0x48.

The exact record format is explained in the VS1063a Programmer’s Guide.

10.10 I2C Boot

VS1063 also supports boot from I2C EEPROM. I2C boot is only tried if GPIO0 is pulled high, but the required boot ident is not found from SPI EEPROM. When GPIO0 is low, boot is not tried and normal decoding mode is entered.

I2C boot redefines the following pins:

<table>
<thead>
<tr>
<th>Normal Mode</th>
<th>SPI Boot Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO0</td>
<td>high = enable SPI/I2C boot</td>
</tr>
<tr>
<td>GPIO4</td>
<td>SDA</td>
</tr>
<tr>
<td>GPIO6</td>
<td>SCL</td>
</tr>
</tbody>
</table>

Both SDA and SCL has to have an external pull-up.

The memory has to be an I2C EEPROM with 8-bit or 16-bit address. The serial speed used by VS1063a is <100 kHz with the nominal 12.288 MHz clock. The boot record format is the same as for SPI boot.
10.11 Extra Parameters (Parametric Structure)

The following parametric structure is in X memory at address 0x1e00 and can be used to set extra parameters or get useful information. SCI_WRAMADDR addresses 0xc0c0 to 0xc0ff are translated automatically to parametric structure addresses 0x1e00…0x1e3f. Also, when an address from 0xc0c0 to 0xc0ff is written, sdiFree and audioFill are updated.

#define PARAMETRIC_VERSION 0x0004
struct parametric { /* configs are not cleared between files */
    u_int32 chipID; /*0x1e00/01 Initialized at reset for your convenience*/
    u_int16 version; /*0x1e02 - structure version */
    u_int16 configI; /*0x1e03 wanf --C pps RRR */
    s_int16 playSpeed; /*0x1e04 0.1 = normal speed, 2 = twice, etc. */
    u_int16 bitRatePer100; /*0x1e05 average bitrate divided by 100 */
    u_int16 endFill1Byte; /*0x1e06 which byte value to send after file */
    s_int32 rateTune; /*0x1e07..8 samplerate tune in +-1ppm steps. V4*/
    u_int16 playMode; /*0x1e09 play and processing enables V4 */
    s_int32 sampleCounter; /*0x1e0a..b sample counter. V4*/
    u_int16 vuMeter; /*0x1e0c VU meter result V4*/
    u_int16 adMixerGain; /*0x1e0d AD mixer attenuation in 3dB steps -3..-31*/
    u_int16 adMixerConfig; /*0x1e0e AD mixer config, bits 5-4=rate, 7-6=mode */
    u_int16 pcmMixerRate; /*0x1e0f PCM mixer samplerate (read when enabled)*/
    u_int16 pcmMixerFree; /*0x1e10 PCM mixer FIFO free state */
    u_int16 pcmMixerVol; /*0x1e11 PCM mixer volume 0..191 (-0.5dB steps) */
    u_int16 eq5Updated; /*0x1e12..0x1e1b 5-channel EQ parameters */
    u_int16 speedShifter; /*0x1e1d Speed Shifter speed 0x4000 == 1.0x V4 */
    u_int16 earSpeakerLevel; /*0x1e1e EarSpeaker level, 0 = off. V4*/
    u_int16 sdiFree; /*0x1e1f SDI FIFO free in words. V4*/
    u_int16 audioFill; /*0x1e20 Audio buffer fill in stereo samples. V4*/
    u_int16 reserved[4]; /*0x1e21..24 */
    s_int16 resync; /*0x1e29 > 0 for automatic m4a, ADIF, WMA resyncs*/
} i;

union { /* 22 available -- these are not cleared at software reset! */
    struct {
        s_int16 txUartDiv; /*0x1e2a direct set of UART divider*/
        s_int16 txUartByteSpeed; /*0x1e2b set UART byte speed (txUartDiv<9)*/
        u_int16 txPauseGpio; /*0x1e2c mask: a high level pauses tx*/
        s_int16 aecAdaptMultiplier; /*0x1e2d 2 for default */
        s_int16 reserved[4];
        u_int16 channelMax[2]; /*0x1e3c,1e3d for record level monitoring*/
        u_int32 serialNumber; /*0x1e3e,1e3f for Ogg Vorbis if enabled in WRAMADDR(11)*/
    } encoding;
    struct {
        u_int32 curPacketSize;
        u_int32 packetSize;
    } wma; /* 4*/
} generic[22]; /*0x1e2a*/

union {
    struct {
        u_int16 sceFoundMask; /*0x1e2a single-channel-el. found since last clr*/
        u_int16 cpeFoundMask; /*0x1e2b channel-pair-el. found since last clr*/
        u_int16 lfeFoundMask; /*0x1e2c low-frequency-el. found since last clr*/
        u_int16 playSelect; /*0x1e2d 0 = first any, initialized at aac init */
        s_int16 dynCompress; /*0x1e2e -8192=1.0, initialized at aac init */
        s_int16 dynBoost; /*0x1e2f 8192=1.0, initialized at aac init */
    } playSelect;
    struct {
        s_int16 gain; /*0x1e2a proposed gain offset, default = -12 */
    } vorbis;
} i;
10.11.1 chipID, version, config1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>chipID</td>
<td>0x1e00-01</td>
<td>Fuse-programmed unique ID (cosmetic copy of the fuses)</td>
</tr>
<tr>
<td>version</td>
<td>0x1e02</td>
<td>Structure version – 0x0004</td>
</tr>
<tr>
<td>config1</td>
<td>0x1e03</td>
<td>Miscellaneous configuration</td>
</tr>
</tbody>
</table>

The fuse-programmed ID is read at startup and copied into the chipID field. If not available, the value will be all zeros.

The version field can be used to determine the layout of the rest of the structure. The version number is changed when the structure is changed. For VS1063a the structure version is 4.

config1 sets miscellaneous settings. Bits 12 to 15 can be used by the user to easily disable certain decoders. Disabling FLAC may be useful in standalone applications to increase the data memory available for the application. The default reset value for config1 is 0x0010.

<table>
<thead>
<tr>
<th>bits</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>1 = Disable WMA decoding</td>
</tr>
<tr>
<td>14</td>
<td>1 = Disable AAC decoding</td>
</tr>
<tr>
<td>13</td>
<td>1 = Disable MP3 decoding</td>
</tr>
<tr>
<td>12</td>
<td>1 = Disable FLAC decoding</td>
</tr>
<tr>
<td>11:9</td>
<td>Reserved, set to 0</td>
</tr>
<tr>
<td>8</td>
<td>1 = Disable CRC checking for MP3</td>
</tr>
<tr>
<td>7:6</td>
<td>AAC PS configuration</td>
</tr>
<tr>
<td>5:4</td>
<td>AAC SBR configuration</td>
</tr>
<tr>
<td>3:0</td>
<td>not used in VS1063a</td>
</tr>
</tbody>
</table>

Bits 7 to 4 in config1 can be used to control the SBR (Spectral Band Replication) and PS (Parametric Stereo) decoding. Bits 5 and 4 select SBR mode and bits 7 and 6 select PS mode. These configuration bits are useful if your AAC license does not cover SBR and/or PS.

<table>
<thead>
<tr>
<th>config1(5:4)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00'</td>
<td>normal mode, upsample ≤24 kHz AAC files</td>
</tr>
<tr>
<td>'01'</td>
<td>do not automatically upsample ≤24 kHz AAC files, but enable upsampling if SBR is encountered (default)</td>
</tr>
<tr>
<td>'10'</td>
<td>never upsample</td>
</tr>
<tr>
<td>'11'</td>
<td>disable SBR (also disables PS)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>config1(7:6)</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>'00'</td>
<td>normal mode, process PS if it is available</td>
</tr>
<tr>
<td>'01'</td>
<td>process PS if it is available, but in downsampled mode</td>
</tr>
<tr>
<td>'10'</td>
<td>reserved</td>
</tr>
<tr>
<td>'11'</td>
<td>disable PS processing</td>
</tr>
</tbody>
</table>
10.11.2 Player Configurations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>playSpeed</td>
<td>0xe04</td>
<td>0, 1 = normal speed, 2 = double, 3 = three times etc.</td>
</tr>
<tr>
<td>bitRatePer100</td>
<td>0xe05</td>
<td>average bitrate divided by 100</td>
</tr>
<tr>
<td>endFillByte</td>
<td>0xe06</td>
<td>byte to send after file</td>
</tr>
<tr>
<td>rateTune</td>
<td>0xe07:1e08</td>
<td>samplerate finetune in +/-1ppm steps</td>
</tr>
<tr>
<td>playMode</td>
<td>0xe09</td>
<td>mono, pause, and extra audio processing selects</td>
</tr>
<tr>
<td>sampleCounter</td>
<td>0xe0a:1e0b</td>
<td>sample counter</td>
</tr>
<tr>
<td>sdiFree</td>
<td>0xe1f</td>
<td>SDI FIFO free space in words</td>
</tr>
<tr>
<td>audioFill</td>
<td>0xe20</td>
<td>Audio buffer fill in stereo samples</td>
</tr>
<tr>
<td>latestSOF</td>
<td>0xe25:1e26</td>
<td>latest start of frame</td>
</tr>
<tr>
<td>positionMsec</td>
<td>0xe27:1e28</td>
<td>File position in milliseconds, if available</td>
</tr>
<tr>
<td>resync</td>
<td>0xe29</td>
<td>Automatic resync selector</td>
</tr>
</tbody>
</table>

**playSpeed** makes it possible to fast forward songs. Decoding of the bitstream is performed, but only each **playSpeed** frames are played. For example by writing 4 to **playSpeed** will play the song four times as fast as normal, if you are able to feed the data with that speed. Write 0 or 1 to return to normal speed. **SCI_DECODE_TIME** will also count faster. All current codecs support the **playSpeed** configuration.

**bitRatePer100** contains the average bitrate divided by 100. The value is updated once per second and it can be used to calculate an estimate of the remaining playtime. This value is also available in **SCI_HDAT0** for all codecs except MP3, MP2, and MP1.

**endFillByte** indicates what byte value to send after file is sent before SM_CANCEL.

**rateTune** finetunes the samplerate in 1ppm steps. This is useful in streaming applications where long-term buffer fullness is used to adjust the samplerate very accurately. Zero is normal speed, positive values speed up, negative values slow down. To calculate **rateTune** for a speed, use \((x - 1.0) \times 1000000\). For example 5.95% speedup \((1.0595 - 1.0) \times 1000000 = 59500\).

**playMode** provides mono and pause select bits. It also contains some extra processing block enables. Setting the pause bit will immediately stop audio sample output. Samples already in the audio buffer will be played, but stream buffer is not read until pause bit is cleared. The mono select averages left and right channel so LEFT and RIGHT outputs will be the same. Other bits are explained separately.

<table>
<thead>
<tr>
<th>bits</th>
<th>Name</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>PLAYMODE_SPEEDSHIFTER_ON</td>
<td>Speedshifter enable</td>
</tr>
<tr>
<td>5</td>
<td>PLAYMODE_EQ5_ON</td>
<td>EQ5 enable</td>
</tr>
<tr>
<td>4</td>
<td>PLAYMODE_PCMMIXER_ON</td>
<td>PCM Mixer enable</td>
</tr>
<tr>
<td>3</td>
<td>PLAYMODE_ADMIXER_ON</td>
<td>AD Mixer enable</td>
</tr>
<tr>
<td>2</td>
<td>PLAYMODE_VUMETER_ON</td>
<td>VU Meter enable</td>
</tr>
<tr>
<td>1</td>
<td>PLAYMODE_PAUSE_ON</td>
<td>Pause enable</td>
</tr>
<tr>
<td>0</td>
<td>PLAYMODE_MONO_OUTPUT</td>
<td>Mono output select</td>
</tr>
</tbody>
</table>
sampleCounter advances for each played sample and is initialized by Ogg Vorbis decoding.

sdiFree and audioFill can be used to monitor and control the playback delay in special applications. sdiFree and audioFill are updated when WRAMADDR is written with values from 0xc0c0 to 0xc0ff. These translate to parametric structure addresses 0x1e00...0x1e3f automatically. So, write 0xc0df to WRAMADDR, and then read WRAM twice to get both sdiFree and audioFill.

latestSOF returns the position of the current (AAC) or next (WMA) beginning of a frame. You can use this information to implement glitch-free A-B loop or rewind.

positionMsec is a field that gives the current play position in a file in milliseconds, regardless of rewind and fast forward operations. The value is only available in codecs that can determine the play position from the stream itself. Currently WMA and Ogg Vorbis provide this information. If the position is unknown, this field contains -1.

resync field is used to force a resynchronization to the stream for WMA and AAC (ADIF, .mp4 / .m4a) instead of ending the decode at first error. This field can be used to implement almost perfect fast forward and rewind for WMA and AAC (ADIF, .mp4 / .m4a). The user should set this field before performing data seeks if they are not in packet or data block boundaries. The field value tells how many tries are allowed before giving up. The value 32767 gives infinite tries.

The resync field is set to 32767 after a reset to make resynchronization the default action, but it can be cleared after reset to restore the old action. When resync is set, every file decode should always end as described in Chapter 10.5.1.

When resync is required, WMA and AAC codecs now enter broadcast/stream mode where file size information is ignored. Also, the file size and data size information of WAV files are ignored when resync is non-zero. The user must use SMCANCEL or software reset to end decoding.

Note: WAV, WMA, ADIF, and .mp4 /.m4a files begin with a metadata or header section, which must be fully processed before any fast forward or rewind operation. SS_DO_NOT_JUMP (in SCI_STATUS) is clear when the header information has been processed and jumps are allowed.

#define CFG1_NOWMA (1<<15)
#define CFG1_NOAAC (1<<14)
#define CFG1_NOP3 (1<<13)
#define CFG1_NOPFLAC (1<<12) /* To allow more memory for the user */
#define CFG1_PSNORMAL (0<<6)
#define CFG1_PSDOWNSAMPLED (1<<6) /* PS in downsampled mode */
#define CFG1_PSOFF (3<<6) /* no PS */
#define CFG1_SBRNORMAL (0<<4)
#define CFG1_SBRNOIMPLICIT (1<<4) /* default! */
#define CFG1_SBRDOWNSAMPLED (2<<4) /* never upsample */
#define CFG1_SBR0FF (3<<4) /* no SBR or PS */
#define AAC_SBR_PRESENT 1
#define AAC_UPSAMPLE_ACTIVE 2
#define AAC_PS_PRESENT 4
#define AAC_PS_ACTIVE 8

Version: 1.00, 2012-10-24
Notice that reading two-word variables through the SCI_WRAMADDR and SCI_WRAM interface is only partly atomic. In VS1063 a write to SCI_WRAMADDR reads ahead two words that it provides to SCI_WRAM, so the two halves of a long variable are sampled together. But as the write to the variable may not be protected from interrupts, the SCI interrupt may occur between the update of the low and high parts of the variable.

It is quite improbable though. If you want to make certain the value is correct, read it twice and compare the results.

### 10.11.3 VU Meter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>playMode</td>
<td>0x1e09</td>
<td>bit 2: VU meter enable</td>
</tr>
<tr>
<td>vuMeter</td>
<td>0x1e0c</td>
<td>VU meter result (if VU meter enabled)</td>
</tr>
</tbody>
</table>

VU Meter takes the absolute maximum of the output samples and reports it in 3dB steps from 0 to 32, separately for left and right channel. Bits 15...8 of `parametric_x.vuMeter` contain the left channel result, bits 7...0 contain the right channel result.

VU Meter uses about 0.2 MIPS of processing power at 48kHz samplerate.
### 10.11.4 AD Mixer

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>playMode</td>
<td>0x1e09</td>
<td>bit 3: AD Mixer enable</td>
</tr>
<tr>
<td>adMixerGain</td>
<td>0x1e0d</td>
<td>AD mixer attenuation in 3dB steps -3...-31</td>
</tr>
<tr>
<td>adMixerConfig</td>
<td>0x1e0e</td>
<td>AD mixer config</td>
</tr>
</tbody>
</table>

```c
#define ADMIXER_RATEMASK  (3<<0)
#define ADMIXER_RATE192    (0<<0) /* 5 MIPS */
#define ADMIXER_RATE96     (1<<0) /* 2.5 MIPS */
#define ADMIXER_RATE48     (2<<0) /* 1.25 MIPS */
#define ADMIXER_RATE24     (3<<0) /* 0.6 MIPS */
#define ADMIXER_MODEMASK   (3<<2)
#define ADMIXER_MODESTEREO (0<<2)
#define ADMIXER_MODEMONO   (1<<2)
#define ADMIXER_MODELEFT   (2<<2)
#define ADMIXER_MODERIGHT  (3<<2)
```

AD Mixer allows to mix MIC or LINE inputs with any decoded audio format. Four modes are provided: stereo, mono down-mix of left and right channels, left channel, right channel.

The mix gain can be set in 3dB steps using `adMixerGain`. The mixing samplerate can be 24 kHz, 48 kHz, 96 kHz, or 192 kHz. The higher the rate, the better the quality, but also the more processing power is required.

In practice 48 kHz is good enough quality for all applications (takes 1.25 MIPS), using 96 kHz and 192 kHz are only recommended if you use I2S with those rates.

The AD Mixer configuration `adMixerConfig` must be set before AD Mixer enable bit is set in `playMode`. The gain control can be adjusted at any time.

AD Mixer and PCM Mixer can not be on simultaneously. AD Mixer overrides PCM Mixer.
10.11.5 PCM Mixer

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>playMode</td>
<td>0x1e09</td>
<td>bit 4: PCM Mixer enable</td>
</tr>
<tr>
<td>pcmMixerRate</td>
<td>0x1e0f</td>
<td>PCM mixer samplerate</td>
</tr>
<tr>
<td>pcmMixerFree</td>
<td>0x1e10</td>
<td>PCM mixer FIFO free state</td>
</tr>
<tr>
<td>pcmMixerVol</td>
<td>0x1e11</td>
<td>PCM mixer volume 0...191 (-0.5 dB steps)</td>
</tr>
</tbody>
</table>

The PCM Mixer allows a mono 16-bit linear PCM stream be played back during any audio format playback. Because the SDM audio side path does not have any interpolation, the PCM audio is automatically upsampled to at least 22000 Hz to keep good audio quality.

The PCM samplerate is configured from `pcmMixerRate`, and it must be written before PCM Mixer is enabled from the `playMode` variable. With the nominal 12.288 MHz clock the samplerates 8000 Hz, 12000 Hz, 16000 Hz, 24000 Hz, 32000 Hz, 48000 Hz are exact. You can use other rates as well, but they are not exact (for example 11025 Hz, 22050 Hz and 44100 Hz play 0.23% too fast).

The PCM data is to be written to SCI_AICTRL0 register, and `pcmMixerFree` tells how much space is in the PCM FIFO (you can send up to this many words). Note that SCI multiple write can be used to write multiple words with minimal overhead.

`pcmMixerVol` controls volume independently of the normal playback volume. Values from 0 to 182 control PCM volume in 0.5 dB steps. Note: to prevent sigma-delta modulator overflow, SCI_VOL should be at least 2 dB (0x0404), and the sum of SCI_VOL and `pcmMixerVol` attenuations at least 6 dB (12). If you have not set large enough attenuations, the PCM Mixer adjusts the registers automatically to have at least these values. To have absolutely safe scaling, have 6 dB (0x0c0c) or more in both SCI_VOL and `pcmMixerVol`.

The processing power needed depends on the samplerate, e.g. 8 kHz = 4.0 MIPS, 16 kHz = 6.8 MIPS, 24 kHz = 4.9 MIPS, 32 kHz = 6.5 MIPS.

Processing will be automatically disabled after a 0.125-second timeout when samples are not being written to SCI_AICTRL0. The processing is resumed when there are at least 128 samples in the PCM FIFO (1/4 full).

AD Mixer and PCM Mixer can not be on simultaneously. AD Mixer overrides PCM Mixer.

```c
s_int16 samples[32];
s_int16 availSpace;
Mp3WriteReg(SCI_WRAMADDR, 0x1e10);
availSpace = Mp3ReadReg(SCI_WRAM);
if (availSpace >= 32) {
    ReadSamples(samples, 32);
    Mp3WriteRegMultiple(SCI_AICTRL0, samples, 32);
}
```
10.11.6 EQ5 5-band Equalizer

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>playMode</td>
<td>0x1e09</td>
<td>bit 5: EQ5 enable</td>
</tr>
<tr>
<td>eq5Params</td>
<td>0x1e12/1b</td>
<td>Frequency/gain pairs</td>
</tr>
<tr>
<td>eq5Update</td>
<td>0x1e1c</td>
<td>Indicator that settings have been changed</td>
</tr>
</tbody>
</table>

The 5-band equalizer allows attenuating or enhancing five frequency ranges by up to ±16 dB.

The 5-band equalizer takes its parameters from eq5Params array, which needs to be written before the EQ5 is enabled from bit 5 of playMode. If the settings are changed while EQ5 is active, new settings can be forced to be taken into use by writing a non-zero value to eq5Update.

Currently EQ5 and Bass/Treble control cannot be active at the same time. Bass and treble controls override EQ5.

Eq5Params are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Low</th>
<th>High</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>eq5Dummy</td>
<td>0x1e12</td>
<td>0</td>
<td>0</td>
<td>Not used</td>
</tr>
<tr>
<td>eq5Level1</td>
<td>0x1e13</td>
<td>-32</td>
<td>32</td>
<td>Bass level in 1/2 dB steps</td>
</tr>
<tr>
<td>eq5Freq1</td>
<td>0x1e14</td>
<td>20</td>
<td>150</td>
<td>Bass/Mid-Bass cutoff in Hz</td>
</tr>
<tr>
<td>eq5Level2</td>
<td>0x1e15</td>
<td>-32</td>
<td>32</td>
<td>Mid-Bass level in 1/2 dB steps</td>
</tr>
<tr>
<td>eq5Freq2</td>
<td>0x1e16</td>
<td>50</td>
<td>1000</td>
<td>Mid-Bass/Mid cutoff in Hz</td>
</tr>
<tr>
<td>eq5Level3</td>
<td>0x1e17</td>
<td>-32</td>
<td>32</td>
<td>Mid level in 1/2 dB steps</td>
</tr>
<tr>
<td>eq5Freq3</td>
<td>0x1e18</td>
<td>1000</td>
<td>15000</td>
<td>Mid/Mid-High cutoff in Hz</td>
</tr>
<tr>
<td>eq5Level4</td>
<td>0x1e19</td>
<td>-32</td>
<td>32</td>
<td>Mid-High level in 1/2 dB steps</td>
</tr>
<tr>
<td>eq5Freq4</td>
<td>0x1e1a</td>
<td>2000</td>
<td>15000</td>
<td>Mid-High/treble cutoff in Hz</td>
</tr>
<tr>
<td>eq5Level5</td>
<td>0x1e1b</td>
<td>-32</td>
<td>32</td>
<td>Treble level in 1/2 dB steps</td>
</tr>
</tbody>
</table>

Freq values must be strictly ascending; e.g. eq5Freq2 must be higher than eq5Freq1, so e.g. combination eq5Freq1=80, eq5Freq2=50 is not allowed.

Example: Vector 0, 24, 70, 12, 300, -6, 3000, 4, 8000, 12 emphasizes bass and treble a lot. However, see below.
To avoid distortion caused by audio clipping, the equalizer internally limits maximum gain for each band so that \( \text{eq5LevelX} + \max(\text{SVOL_LEFT}, \text{SVOL_RIGHT}) \leq 0 \text{ dB} \) (see Chapter 9.8.11 on page 49).

Example: Requested EQ5 response

![Figure 19: Example EQ5 response request](image)

Example: Using the previous example, with a requested frequency response of (+12 dB, +6 dB, -3 dB, +2 dB, +6 dB) as presented in Figure 19, and using different volume settings, we get the responses of Figure 20. Note that because the maximum requested enhancement is +12 dB, the request can only be fulfilled accurately if volume is set to -12 dB or lower.

Example: EQ5 frequency response

![Figure 20: Example EQ5 responses at different volume settings](image)
### 10.11.7 Speed Shifter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>playMode</td>
<td>0x1e09</td>
<td>bit 6: SpeedShifter enable</td>
</tr>
<tr>
<td>speedShifter</td>
<td>0x1e1d</td>
<td>Speed Shifter speed, 0x4000 = 1.0x</td>
</tr>
</tbody>
</table>

Speed Shifter allows the playback tempo to be changed without changing the playback pitch. The playback tempo is \( \frac{\text{speedShifter}}{16384} \), i.e. 16384 is the normal speed. The minimum speed is 0.68x (11141) and maximum speed 1.64x (26869).

If you want to change pitch without changing tempo, adjust the speed and compensate by also adjusting the samplerate. For example two semitones is \( 2^{-2/12} \approx 0.8909 \), so set the Speed Shifter to \( 2^{-2/12} \times 16384 = 14596 \) and set rateTune to \( (2^{2/12} - 1) \times 100000 = 122462 \).

Speed Shifter and EarSpeaker can not be used at the same time. Speed Shifter overrides EarSpeaker.

### 10.11.8 EarSpeaker

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>earSpeakerLevel</td>
<td>0x1e1e</td>
<td>EarSpeaker level, 0 = off</td>
</tr>
</tbody>
</table>

EarSpeaker Spatial Processing is a headphone externalizer algorithm. For information of the algorithm and what it does, see Chapter 9.5, *EarSpeaker Spatial Processing*.

EarSpeaker processing can be adjusted using earSpeakerLevel. Different levels simulate a little different type of acoustical situation, suiting different personal preferences and types of recording.

- 0: Best option when listening through loudspeakers or if the audio to be played contains binaural preprocessing.
- 12000: Suited for listening to normal musical scores with headphones, very subtle.
- 38000: Suited for listening to normal musical scores with headphones, moves sound source further away than minimal.
- 50000: Suited for old or ‘dry’ recordings.

EarSpeaker uses approximately 11 MIPS at 48kHz samplerate.

Speed Shifter and EarSpeaker can not be used at the same time. Speed Shifter overrides EarSpeaker.
10.11.9 Other Parameters

WMA

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>curPacketSize</td>
<td>0x1e2a/2b</td>
<td>The size of the packet being processed</td>
</tr>
<tr>
<td>packetSize</td>
<td>0x1e2c/2d</td>
<td>The packet size in ASF header</td>
</tr>
</tbody>
</table>

The ASF header packet size is available in packetSize. With this information and a packet start offset from latestSOF you can parse the packet headers and skip packets in ASF files.

WMA decoder can also increase the internal clock automatically when it detects that a file cannot be decoded correctly with the current clock. The maximum allowed clock is configured with the SCI_CLOCKF register.

AAC

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Address</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>config1</td>
<td>0x1e03(7:4)</td>
<td>SBR and PS select</td>
</tr>
<tr>
<td>sceFoundMask</td>
<td>0x1e2a</td>
<td>Single channel elements found</td>
</tr>
<tr>
<td>cpeFoundMask</td>
<td>0x1e2b</td>
<td>Channel pair elements found</td>
</tr>
<tr>
<td>lfeFoundMask</td>
<td>0x1e2c</td>
<td>Low frequency elements found</td>
</tr>
<tr>
<td>playSelect</td>
<td>0x1e2d</td>
<td>Play element selection</td>
</tr>
<tr>
<td>dynCompress</td>
<td>0x1e2e</td>
<td>Compress coefficient for DRC, -8192=1.0</td>
</tr>
<tr>
<td>dynBoost</td>
<td>0x1e2f</td>
<td>Boost coefficient for DRC, 8192=1.0</td>
</tr>
<tr>
<td>sbrAndPsStatus</td>
<td>0x1e30</td>
<td>SBR and PS available flags</td>
</tr>
<tr>
<td>sbrAndPsFlags</td>
<td>0x1e31</td>
<td>SBR and PS mode</td>
</tr>
</tbody>
</table>

For an explanation on config1 AAC bits, see Chapter 10.11.1.

playSelect determines which element to decode if a stream has multiple elements. The value is set to 0 each time AAC decoding starts, which causes the first element that appears in the stream to be selected for decoding. Other values are: 0x01 - select first single channel element (SCE), 0x02 - select first channel pair element (CPE), 0x03 - select first low frequency element (LFE), $S \times 16 + 5$ - select SCE number $S$, $P \times 16 + 6$ - select CPE number $P$, $L \times 16 + 7$ - select LFE number $L$. When automatic selection has been performed, playSelect reflects the selected element.

sceFoundMask, cpeFoundMask, and lfeFoundMask indicate which elements have been found in an AAC stream since the variables have last been cleared. The values can be used to present an element selection menu with only the available elements.

dynCompress and dynBoost change the behavior of the dynamic range control (DRC) that is present in some AAC streams. These are also initialized when AAC decoding starts.
sbrAndPsStatus indicates spectral band replication (SBR) and parametric stereo (PS) status.

<table>
<thead>
<tr>
<th>Bit</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SBR present</td>
</tr>
<tr>
<td>1</td>
<td>upsampling active</td>
</tr>
<tr>
<td>2</td>
<td>PS present</td>
</tr>
<tr>
<td>3</td>
<td>PS active</td>
</tr>
</tbody>
</table>

sbrAndPsFlags indicates the current spectral band replication (SBR) and parametric stereo (PS) mode in bits 7:4 in the same order as in config1 SBR and PS bits 7:4.

AAC decoder can also increase the internal clock automatically when it detects that a file can not be decoded correctly with the current clock. The maximum allowed clock is configured with the SCI_CLOCKF register.

If even the highest allowed clock is too slow to decode an AAC file with SBR and PS components, the advanced decoding features are automatically dropped one by one until the file can be played. First the parametric stereo processing is dropped (the playback becomes mono). If that is not enough, the spectral band replication is turned into downsampled mode (reduced bandwidth). As the last resort the spectral band replication is fully disabled. Dropped features are restored at each song change.
Ogg Vorbis decoding supports Replay Gain technology. The Replay Gain technology is used to automatically give all songs a matching volume so that the user does not need to adjust the volume setting between songs. If the Ogg Vorbis decoder finds a Replay Gain tag in the song header, the tag is parsed and the decoded gain setting can be found from the gain parameter. For a song without any Replay Gain tag, a default of -6 dB (gain value -12) is used. For more details about Replay Gain, see http://en.wikipedia.org/wiki/Replay_Gain and http://www.replaygain.org/.

The player software can use the gain value to adjust the volume level. Negative values mean that the volume should be decreased, positive values mean that the volume should be increased.

For example, gain = -11 means that volume should be decreased by 5.5 dB \((-11/2 = -5.5\)), and left and right attenuation should be increased by 11. When gain = 2 volume should be increased by 1 dB \((2/2 = 1.0)\), and left and right attenuation should be decreased by 2. Because volume setting can not go above +0 dB, the value should be saturated.

<table>
<thead>
<tr>
<th>Gain</th>
<th>Volume</th>
<th>SCI_VOL (Volume-Gain)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-11 (-5.5 dB)</td>
<td>0 (+0.0 dB)</td>
<td>0x0b0b (-5.5 dB)</td>
</tr>
<tr>
<td>-11 (-5.5 dB)</td>
<td>3 (-1.5 dB)</td>
<td>0xe0e (-7.0 dB)</td>
</tr>
<tr>
<td>+2 (+1.0 dB)</td>
<td>0 (+0.0 dB)</td>
<td>0x0000 (+0.0 dB)</td>
</tr>
<tr>
<td>+2 (+1.0 dB)</td>
<td>1 (-0.5 dB)</td>
<td>0x0000 (+0.0 dB)</td>
</tr>
<tr>
<td>+2 (+1.0 dB)</td>
<td>4 (-2.0 dB)</td>
<td>0x0202 (-1.0 dB)</td>
</tr>
</tbody>
</table>
10.12 SDI Tests

There are several test modes in VS1063a, which allow the user to perform memory tests, SCI bus tests, and several different sine wave tests.

All tests are started in a similar way: VS1063a is hardware reset, SM_TESTS is set, and then a test command is sent to the SDI bus. Each test is started by sending a 4-byte special command sequence, followed by 12 zeros. The sequences are described below.

10.12.1 Sine Test

Sine test is initialized with the 16-byte sequence 0x53 0xEF 0x6E n 0 0 0 0 0 0 0 0 0 0 0 0, where n defines the sine test to use. n is defined as follows:

<table>
<thead>
<tr>
<th>n bits</th>
<th>Name</th>
<th>Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>7:5</td>
<td>FsIdx</td>
<td>7:5</td>
<td>Samplerate index</td>
</tr>
<tr>
<td>4:0</td>
<td>S</td>
<td>4:0</td>
<td>Sine skip speed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FsIdx</th>
<th>Fs</th>
<th>FsIdx</th>
<th>Fs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>44100 Hz</td>
<td>4</td>
<td>24000 Hz</td>
</tr>
<tr>
<td>1</td>
<td>48000 Hz</td>
<td>5</td>
<td>16000 Hz</td>
</tr>
<tr>
<td>2</td>
<td>32000 Hz</td>
<td>6</td>
<td>11025 Hz</td>
</tr>
<tr>
<td>3</td>
<td>22050 Hz</td>
<td>7</td>
<td>12000 Hz</td>
</tr>
</tbody>
</table>

The frequency of the sine to be output can now be calculated from $F_s = F_s \times \frac{S}{128}$.

Example: Sine test is activated with value 126, which is 0b01111110. Breaking n to its components, FsIdx = 0b0101 = 3 and thus $F_s = 22050$ Hz. $S = 0b11110 = 30$, and thus the final sine frequency $F = 22050$ Hz $\times \frac{30}{128} \approx 5168$ Hz.

To exit the sine test, send the 16-byte sequence 0x45 0x78 0x69 0x74 0 0 0 0 0 0 0 0 0 0 0 0.

Note: Sine test signals go through the digital volume control, so it is possible to test channels separately.

10.12.2 Pin Test

Pin test is activated with the 16-byte sequence 0x50 0xED 0x6E 0x54 0 0 0 0 0 0 0 0 0 0 0 0.

This test is meant for chip production testing only.

10.12.3 SCI Test

Sci test is initialized with the 16-byte sequence 0x53 0x70 0xEE n 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0, where n is the register number to test. The content of the given register is read and copied to SCI_HDAT0. If the register to be tested is HDAT0, the result is copied to SCI_HDAT1.

Example: if n is 0, contents of SCI register 0 (SCI_MODE) is copied to SCI_HDAT0.
10.12.4 Memory Test

Memory test mode is initialized with the 16-byte sequence \(0x4D\ 0xEA\ 0x6D\ 0x54\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\). After this sequence, wait for 1100000 clock cycles. The result can be read from the SCI register SCI_HDAT0, and ‘one’ bits are interpreted as follows:

<table>
<thead>
<tr>
<th>Bit(s)</th>
<th>Mask</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>0x8000</td>
<td>Test finished</td>
</tr>
<tr>
<td>14:10</td>
<td>Unused</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0x0200</td>
<td>Mux test succeeded</td>
</tr>
<tr>
<td>8</td>
<td>0x0100</td>
<td>Good MAC RAM</td>
</tr>
<tr>
<td>7</td>
<td>0x0080</td>
<td>Good I RAM</td>
</tr>
<tr>
<td>6</td>
<td>0x0040</td>
<td>Good Y RAM</td>
</tr>
<tr>
<td>5</td>
<td>0x0020</td>
<td>Good X RAM</td>
</tr>
<tr>
<td>4</td>
<td>0x0010</td>
<td>Good I ROM 1</td>
</tr>
<tr>
<td>3</td>
<td>0x0008</td>
<td>Good I ROM 2</td>
</tr>
<tr>
<td>2</td>
<td>0x0004</td>
<td>Good Y ROM</td>
</tr>
<tr>
<td>1</td>
<td>0x0002</td>
<td>Good X ROM 1</td>
</tr>
<tr>
<td>0</td>
<td>0x0001</td>
<td>Good X ROM 2</td>
</tr>
<tr>
<td></td>
<td>0x83ff</td>
<td>All ok</td>
</tr>
</tbody>
</table>

Memory tests overwrite the current contents of the RAM memories.

10.12.5 New Sine and Sweep Tests

A more frequency-accurate sine test can be started and controlled from SCI. SCI_AICTRLo and SCI_AICTRL1 set the sine frequencies for left and right channel, respectively. These registers, volume (SCI_VOL), and samplerate (SCI_AUDATA) can be set before or during the test. Write 0x4020 to SCI_AIADDR to start the test.

SCI_AICTRLn can be calculated from the desired frequency and DAC samplerate by:

\[
SCI_{-}AICTRLn = F_{\text{sin}} \times 65536/F_s
\]

The maximum value for SCI_AICTRLn is 0x8000U. For the best S/N ratio for the generated sine, three LSBs of the SCI_AICTRLn should be zero. The resulting frequencies \(F_{\text{sin}}\) can be calculated from the DAC samplerate \(F_s\) and SCI_AICTRLo / SCI_AICTRL1 using the following equation.

\[
F_{\text{sin}} = SCI_{-}AICTRLn \times F_s/65536
\]

Sine sweep test can be started by writing 0x4022 to SCI_AIADDR.

Both these tests use the normal audio path, thus also SCI_BASS, differential output mode, and EarSpeaker settings have an effect.
10.13 Clock Speed Requirements

When given the highest allowed clock speed VS1063a is capable of decoding or encoding all audio formats with all audio processing. However, to save power it might be useful to run VS1063a at a lower clock speed when all options are not used. This Chapter presents the processing power requirements for the worst case of each VS1063a processing algorithm.

Note: All requirements in this chapter are simulated estimates.

10.13.1 Clock Speed Requirements for Decoders

The table below presents worst-case clock speed requirements for VS1063a decoders. The clock speeds include both the decoder and all audio signal path overheads.

<table>
<thead>
<tr>
<th>Decoders</th>
<th>Algorithm</th>
<th>MIPS</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MP2</td>
<td>18</td>
<td>384 kbit/s CBR, 48 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>MP3</td>
<td>30</td>
<td>320 kbit/s CBR, 48 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>AAC-LC</td>
<td>30</td>
<td>300 kbit/s VBR, 48 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>AAC-HE</td>
<td>55</td>
<td>80 kbit/s CBR, 48 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>AAC-HE2</td>
<td>55</td>
<td>39 kbit/s CBR, 48 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>Ogg Vorbis</td>
<td>36</td>
<td>300 kbit/s VBR, 48 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>WMA</td>
<td>55</td>
<td>22 kbit/s, 32 kHz stereo, Class 4 (Level 2) LPC</td>
</tr>
<tr>
<td></td>
<td>WAV</td>
<td>12</td>
<td>389 kbit/s, 48 kHz IMA ADPCM stereo</td>
</tr>
<tr>
<td></td>
<td>WAV</td>
<td>12</td>
<td>1536 kbit/s, 48 kHz 16-bit PCM stereo</td>
</tr>
</tbody>
</table>

10.13.2 Clock Speed Requirements for Encoders

The following tables are worst cases for the encoders, with XTALI = 12.288 MHz. For Ogg Vorbis values for several sample rates are presented. The clock speeds include both the decoder and all audio signal path overheads.

<table>
<thead>
<tr>
<th>Encoders</th>
<th>Algorithm</th>
<th>MIPS</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MP3</td>
<td>61</td>
<td>320 kbit/s, 44.1 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>Ogg Vorbis</td>
<td>67¹</td>
<td>Quality 7 (nominal 183 kbit/s), 32 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>Ogg Vorbis</td>
<td>67¹</td>
<td>Quality 7 (nominal 222 kbit/s), 44.1 kHz stereo</td>
</tr>
<tr>
<td></td>
<td>WAV</td>
<td>55</td>
<td>44.1 kHz stereo IMA ADPCM (most CPU intensive WAV format)</td>
</tr>
<tr>
<td></td>
<td>G.722</td>
<td>55</td>
<td>16 kHz stereo (standard only requires 16 kHz mono)</td>
</tr>
</tbody>
</table>

¹ Requires 5.5× clock multiplier. There are no direct settings to do this, but you can set the base clock multiplier to e.g. 4.5× and clock adder to 1.0×. In this case, if input clock XTALI = 12.288 MHz, then SCI_CLOCKF = 0xC800.
10.13.3 Clock Speed Requirements for DSP Algorithms

The following table shows the required processing power for different algorithm options of VS1063a.

Unless otherwise stated, the processing power numbers in the table below are calculated for a worst-case 48 kHz stereo stream. If audio is running at a lower samplerate, processing power requirements will proportionally be lower (e.g. at 24 kHz the processing power requirement will be halved).

<table>
<thead>
<tr>
<th>Audio Processing</th>
<th>Algorithm</th>
<th>MIPS</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bass Control</td>
<td>2.3</td>
<td>SCI_BASS bits 7:0</td>
<td></td>
</tr>
<tr>
<td>Treble Control</td>
<td>1.3</td>
<td>SCI_BASS bits 15:8</td>
<td></td>
</tr>
<tr>
<td>EarSpeaker</td>
<td>11.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCM Mixer</td>
<td>6.5</td>
<td>When PCM audio running at 32 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.9</td>
<td>When PCM audio running at 24 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6.8</td>
<td>When PCM audio running at 16 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4.0</td>
<td>When PCM audio running at  8 kHz</td>
<td></td>
</tr>
<tr>
<td>AD Mixer</td>
<td>5.0</td>
<td>When AD converter is at 192 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2.5</td>
<td>When AD converter is at  96 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.3</td>
<td>When AD converter is at  48 kHz</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.6</td>
<td>When AD converter is at  24 kHz</td>
<td></td>
</tr>
<tr>
<td>VU Meter</td>
<td>0.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mono Output</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11 VS1063a Version Changes

This chapter describes the latest and most important changes done to VS1063a.

11.1 Firmware Changes Between VS1053b and VS1063a, 2011-04-13

VS1063a is a pin-compatible firmware upgrade to the VS1053b.

Completely new or major changes:

- Added codec mode.
- Removed MIDI and MPEG layer I (MP1) decoders.
- Layers II and III: new, more robust and accurate decoding. MP3 is now full accuracy compliant. Use at least 2.5× clock to decode all MP3 bitrates and samplers.
- CRC checking added for layer III files that contain CRC. CRC checking can be disabled.
- Keeps track of the valid data in bit reservoir, which allows noiseless start of decoding in the middle of an mp3 file.
- Samplerate finetuning in parametric_x.rateTune.
- Added hooks for detecting and decoding user audio formats.
- WRAMADDR 0xc0c0...0xc0ff is mapped to parametric_x structure.
- Support reading u_int32's (almost) atomically through WRAM.
- Reading of stream and audio buffer fill states possible.
- Proportional and fixed-width font in data ROM for standalone applications.
- WAV decoding supports 24-bit and 32-bit and floating-point formats.
- RIFF-WAV header is generated automatically in WAV encoding (and codec) modes. The user needs to fix the RIFF size and data size fields to make them valid WAV files.
- Sample-exact samplerate and volume change.
- Added mono mode and pause mode for player (parametric_x.playMode)
- Added FLAC decoding upto 2 channels.
- Added VU meter.
- Added AD mixer.
- Added PCM mixer.
- Added Speed shifter.
- AAC, WMA, MP3 and FLAC decoding can be individually disabled using bits in parametric_x.config1.
Minor changes and bug fixes:

- IROM4 switched off and DO_NOT_JUMP cleared in software reset also.
- Handles SM_CANCEL also for mp3 (clears stream buffer).
- Fixed a problem when the first 'OggS' in Ogg Vorbis file was spanning the end and beginning of stream buffer.
- AAC feature drop works in non-implicit upsample mode, and was also otherwise improved.
- Default AAC decoding mode is non-implicit upsample, i.e. upsample only when SBR/PS is detected. This allows to save power with $\leq 24$ kHz files that do not have SBR.
- Ogg Vorbis sets rate only when it changes, allowing the user to override the rate.
- Output volume is now updated in encoding/codec modes.
- Bitrate calculation uses 32-bit second counter.
- Does not clear GPIO_DDR if SPI boot is not tried, so I2S will remain active if you need to use a soft reset. If boot is tried (GPIO0 is high at startup) but fails, restores old GPIO_DDR value.
- Subsonic filter always run for ADC inputs.
- EarSpeaker control is now in parametric_x and gives finer control.
- New adcMode 4 gives mono-downmix of left and right channels.
- WMA fix: sflength must be non-zero.
- MP4 fix: first audio block does not need to start the beginning of the mdat atom.
- AAC fix: works now correctly even if PS header is not available at the first SBR block.
- AAC fix: PNS information was overwritten in transition frames.
- 1.65V reference voltage select (SCI_STATUS(0)) and 3 MHz ADC mode (SCI_STATUS(1)) now work.
- Extra parameter byteRate replaced with bitRatePer100. The new field works consistently with all codecs.
This Chapter describes the firmware and other bugs found in VS1063a, and their status.

- **Sine test** (and other tests) started through SDI requires additional 7-8 zero bytes.
  - Workaround: send 12 zero bytes after sending an SDI test command.
  - Starting from datasheet v1.00 Chapter 10.12, SDI Tests, includes this information.

- **Encoding** (especially) with 48kHz rate can leave the monitoring volume at zero (off) when SCI_VOL is 0.
  - Fixed in vs1063a-patches package v1.03.

- **MP3 encoding** uses a wrong huffman code in one of the encoding tables, causing high-frequency noise. Seems to be highly dependent on the audio material.
  - Fixed in vs1063a-patches package v1.03.

- **20110930:** When using codec mode without headers, the playback rate is set to adcrate (48/24/12 kHz) instead of the requested rate.
  - Fixed in vs1063a-patches package v1.10.

- **20111027:** Disabling, then enabling ADC (DECIM) can cause channels to be swapped.
  - If encoding twice without hardware reset in-between, channels can be swapped.
  - If encoding after using ADMixer (without hw reset in-between), channels can be swapped.
  - If using ADMixer after encoding (without hw reset in-between), channels can be swapped.
  - If using ADMixer with two different rates (without hw reset in-between), channels can be swapped.
  - Switching ADMixer on/off with the same rate is ok. (If no software reset in-between.)
  - Workaround: give a hardware reset (or watchdog reset) after each encoding and change of ADMixer rate.
  - Encoding/codec mode fixed in vs1063a-patches package v1.10
  - ADMixer fixed in vs1063a-patches package v1.2

- **20111031:** MP3 encoding overwrites quantizer selectors for the highest used band (13-16kHz).
  - Fixed in vs1063a-patches package v1.10.

- **20111103:** When using codec mode without headers with PCM mode there is no sound output because the sample size defaults to 0 bits.
  - Fixed in vs1063a-patches package v1.11.

- **20111114:** MP4 decoding is lacking one StreamDiscard()
  - Fixed in vs1063a-patches package v1.2.

- **20111116:** FLAC decoding does not set DO_NOT_JUMP bit during header decode.
  - Fixed in vs1063a-patches package v1.2.

- **20111122:** ADMixer reads configuration from wrong variable (thus always stereo 192k).
  - Fixed in vs1063a-patches package v1.2.
13 Latest Document Version Changes

This chapter describes the most important changes to this document.

Version 1.00, 2012-10-24

This is a major update to the datasheet. As the VS1063a is in production, the datasheet “preliminary” status has been dropped. Other changes and additions are listed below.

- Increased maximum clock speed from $5 \times$ to $5.5 \times$ in Chapter 4.2, *Recommended Operation Conditions* (page 9). Related to this, recommended internal clock has been changed from $3.5 \times + 1.0 \times$ to $4.0 \times + 1.5 \times$ in Chapter 9.8.4, SCI_CLOCKF (page 44).
- Added new Chapter 10.13, *Clock Speed Requirements*.
- Added new Chapter 12, *VS1063a Errata*.
- Added Ogg Vorbis encoding clock requirements to Chapter 10.7.9, *Ogg Vorbis (format 6) Encoder Specific Considerations* (page 62).
- Added clock requirement to Chapter 10.8, *Codec Mode*.
- Clarified Chapter 10.8, *Codec Mode*.
- Clarified Chapter 10.7.1, *Encoding Control Registers*.
- Added information to Chapter 10.11.6, *EQ5 5-band Equalizer*, and corrected scaling error (numbers are in 1/2 dB steps, not 1 dB steps).
- Changed the way SDI tests are activated in Chapter 10.12, *SDI Tests*. Instead of of 8-byte sequences, 16-byte sequences are needed.
- Added IMA ADPCM to the front page list of Full Duplex Codecs.
- Corrected error in first example, called *Example of Encoding initialization including loading VS1063a Patches*, in Chapter 10.7.2, *The Encoding Procedure*.
- Added Figure 4, *SDI in VS10xx Native Mode, single-byte transfer*, and Figure 5, *SDI in VS10xx Native Mode, multi-byte transfer, $X \geq 1$*, to Chapter 7.3.1, *SDI in VS10xx Native Modes*.

Version 0.42, 2011-11-24

- Updated Chapter 10.7.6, *Samplerate Considerations* (page 61), to reflect enhancements caused by the latest *VS1063a Patches v1.2* package.
- Added text to Chapter 1, *Disclaimer*, to inform that this document is only valid when the latest *VS1063a Patches* package has been loaded and activated.
Version 0.41, 2011-09-29

- New VS1063a Patches package is mentioned in Chapters 10.2, Hardware Reset, 10.3, Software Reset, 10.7.2, Encoding Procedure, and 10.7.8, Encoder-Specific Considerations / MP3.
- Corrected Chapter 10.7.4, File Headers.
- Added byte order information to Chapter 10.7.3, Reading Encoded Data Through SCI.

Version 0.40, 2011-09-02

- xRESET, XTALI and XTALO high-level are referenced from IOVDD in Chapter 4.5, Digital Characteristics.
- SCI_STATUS default values fixed.
- MP3 license text updated in Chapter 2.
- Added Chapter 10.7.10, Encoder/Decoder Delays.
- Added new Chapter 10.7.6 under Audio Encoding, Sample Rate Considerations.
- Added example samplerates to Chapter 10.8, Codec Mode.
- Other, minor corrections.

Version 0.30, 2011-05-05

- Typo corrections, synced with VS1063a Hardware Guide.

Version 0.21, 2011-04-15

- Minor bug and typo corrections.

Version 0.20, 2011-04-14

- First public release of the complete datasheet.
- Parts of datasheet split into VS1063a Hardware Guide and VS1063a Programmer’s Guide. These guides will be published in May 2011.

Version 0.10, 2011-01-17

- First two pages released on VLSI Solution’s web pages.
14 Contact Information

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

王丽青
13267231725
292335090
zhiliyingbill@163.com