VS1063a / VS1163a / VS8063a MP3 / OGG VORBIS ENCODER AND 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 / 2 AAC-LC(+PNS),
  HE-AAC v2 (Level 3) (SBR + PS);
  Ogg Vorbis; FLAC;
  WMA 4.0/4.1/7/8/9 all profiles (5-384 kbps);
  WAV (PCM, IMA ADPCM, G.711 μ-law/A-law, G.722 ADPCM)
- Full Duplex Codecs with optional AEC:
  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 output 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 upto 12 GPIO pins

Description

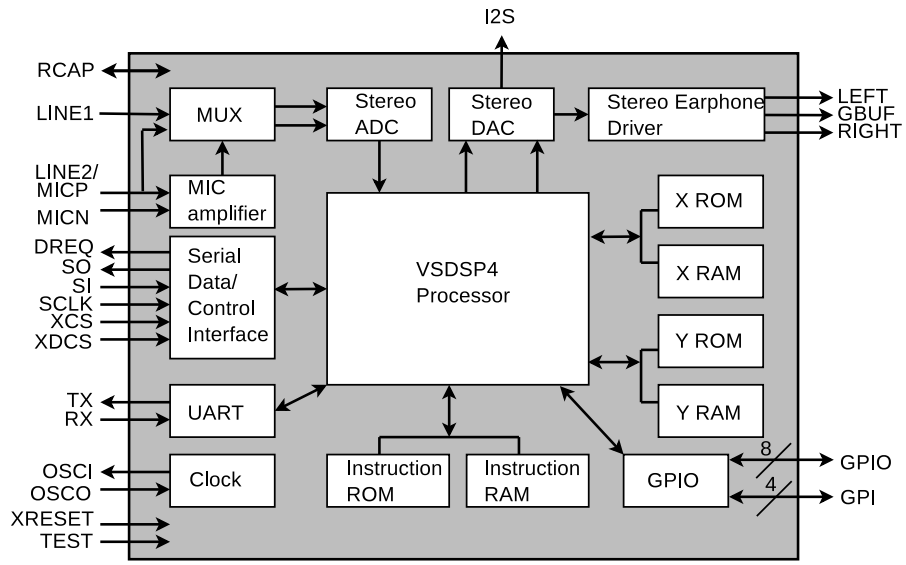
VS1063a is an easy-to-use, versatile encoder, decoder and codec for a multitude of audio formats.

VS1063a contains a high-performance, proprietary low-power DSP core VS_DSP^4, ROM memories, 16 KiB instruction RAM and upto 80 KiB data RAM for user applications running simultaneously with any built-in decoder, serial control and input data interfaces, upto 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 30 Ω 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.

There are three variants of VS1063a: the full-featured VS1063a, VS1163a without an MP3 encoder, and VS8063a without any MP3 functionality.

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

List of Figures

1 Disclaimer

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, semiconductior 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 disabled by using the parametric_x.config1 variable, or by 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 Product Variants

The main encoding and decoding capabilities of VS1063a, VS1163a, and VS8063a are presented in the table below.

Device ID (order code)	MP3		Ogg Vorbis		He-AAC	WMA	FLAC
	Encoder	Decoder	Encoder	Decoder	Dec.	Dec.	Dec.
VS1063A-L	X	X	X	X	X	X	X
VS1163A-L		X	X	X	X	X	X
VS8063A-L			X	X	X	X	X

The only difference between VS1163a and VS1063a is the missing MP3 encoder in VS1163a. With the exception of the parts pertaining to MP3 encoding, all of this datasheet is also applicable to VS1163a.

VS8063a is otherwise similar to VS1163a, except that it also misses MP2/MP3 decoding, so it doesn't have any MP2/MP3 capabilities. With the exception of any parts pertaining to MP2/MP3 functionality, all of this datasheet is also applicable to VS8063a.

5 Characteristics & Specifications

5.1 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Analog Positive Supply	AVDD	-0.3	3.6	V
Digital Positive Supply	CVDD	-0.3	1.85	V
I/O Positive Supply	IOVDD	-0.3	3.6	V
Current at Any Non-Power Pin[1]			±50	mA
Voltage at Any Digital Input		-0.3	IOVDD+0.3[2]	V
Operating Temperature		-30	+85	°C
Storage Temperature		-65	+150	°C

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

5.2 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Ambient Operating Temperature		-30		+85	°C
Analog and Digital Ground [1]	AGND DGND		0.0		V
Positive Analog, REF=1.23V	AVDD	2.5	2.8	3.6	V
Positive Analog, REF=1.65V [2]	AVDD	3.3	3.3	3.6	V
Positive Digital	CVDD	1.7	1.8	1.85	V
I/O Voltage	IOVDD	1.8	2.8	3.6	V
Input Clock Frequency [3]	XTALI	12	12.288	13	MHz
Internal Clock Frequency	CLKI	12	49.152	67.6	MHz
Internal Clock Multiplier [4]		1.0×	4.0×	5.5×	
Master Clock Duty Cycle		40	50	60	%

[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 10.8.2.
[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.

5.3 Analog Characteristics

Unless otherwise noted: AVDD=3.3V, CVDD=1.8V, IOVDD=2.8V, REF=1.65V, TA=-30...+85°C, XTALI=12...13MHz, Internal Clock Multiplier $3.5\times$. DAC tested with 1307.894 Hz full-scale output sinewave, measurement bandwidth 20...20000 Hz, analog output load: LEFT to GBUF 30 Ω, RIGHT to GBUF 30 Ω. Microphone test amplitude 48 mVpp, f_s=1 kHz. Line input test amplitude 2.52 Vpp, f_s=1 kHz.

Parameter	Symbol	Min	Typ	Max	Unit
DAC Resolution			18		bits
Total Harmonic Distortion	THD			0.07	%
Third Harmonic Distortion				0.02	%
Dynamic Range (DAC unmuted, A-weighted)	IDR		100		dB
S/N Ratio (full scale signal)	SNR		94		dB
Interchannel Isolation (Cross Talk), 600Ω + GBUF			80		dB
Interchannel Isolation (Cross Talk), 30Ω + GBUF			53		dB
Interchannel Gain Mismatch		-0.5		0.5	dB
Frequency Response		-0.1		0.1	dB
Full Scale Output Voltage (Peak-to-peak)		1.64	1.85[1]	2.06	Vpp
Deviation from Linear Phase				5	°
Analog Output Load Resistance	AOLR	16	30[2]		Ω
Analog Output Load Capacitance				100	pF
Microphone input amplifier gain	MICG		26		dB
Microphone input amplitude			48	140[3]	mVpp AC
Microphone Total Harmonic Distortion	MTHD		0.03	0.07	%
Microphone S/N Ratio	MSNR	60	70		dB
Microphone input impedances, per pin			45		kΩ
Line input amplitude			2500	2800[3]	mVpp AC
Line input Total Harmonic Distortion	LTHD		0.005	0.014	%
Line input S/N Ratio	LSNR	85	90		dB
Line input impedance			80		kΩ

[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.

5.4 Power Consumption

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

Parameter	Min	Typ	Max	Unit
Power Supply Consumption AVDD, Reset		0.6	5.0	μA
Power Supply Consumption CVDD = 1.8V, Reset		12	20.0	μA
Power Supply Consumption AVDD, sine test, 30 Ω + GBUF	30	36.9	60	mA
Power Supply Consumption CVDD = 1.8V, sine test	8	10	15	mA
Power Supply Consumption AVDD, no load		5		mA
Power Supply Consumption AVDD, output load 30 Ω		11		mA
Power Supply Consumption AVDD, 30 Ω + GBUF		11		mA
Power Supply Consumption CVDD = 1.8V		11		mA

5.5 Digital Characteristics

Parameter	Min	Max	Unit
High-Level Input Voltage (xRESET, XTALI, XTALO)	$0.7\times$IOVDD	IOVDD+0.3[1]	V
High-Level Input Voltage (other input pins)	$0.7\times$CVDD	IOVDD+0.3[1]	V
Low-Level Input Voltage	-0.2	$0.3\times$CVDD	V
High-Level Output Voltage at XTALO = -0.1 mA	$0.7\times$IOVDD		V
Low-Level Output Voltage at XTALO = 0.1 mA		$0.3\times$IOVDD	V
High-Level Output Voltage at I_O = -1.0 mA	$0.7\times$IOVDD		V
Low-Level Output Voltage at I_O = 1.0 mA		$0.3\times$IOVDD	V
Input Leakage Current	-1.0	1.0	μA
SPI Input Clock Frequency [2]		$\frac{CLKI}{7}$	MHz
Rise time of all output pins, load = 50 pF		50	ns

[1] Must not exceed 3.6V
[2] Value for SCI reads. SCI and SDI writes allow $\frac{CLKI}{4}$.

5.6 Switching Characteristics - Boot Initialization

Parameter	Symbol	Min	Max	Unit
XRESET active time		2		XTALI
XRESET inactive to software ready		22000	50000[1]	XTALI
Power on reset, rise time to CVDD		10		V/s

[1] DREQ rises when initialization is complete. Do not send any data or commands before that.

6 Packages and Pin Descriptions

6.1 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*.

6.1.1 LQFP-48

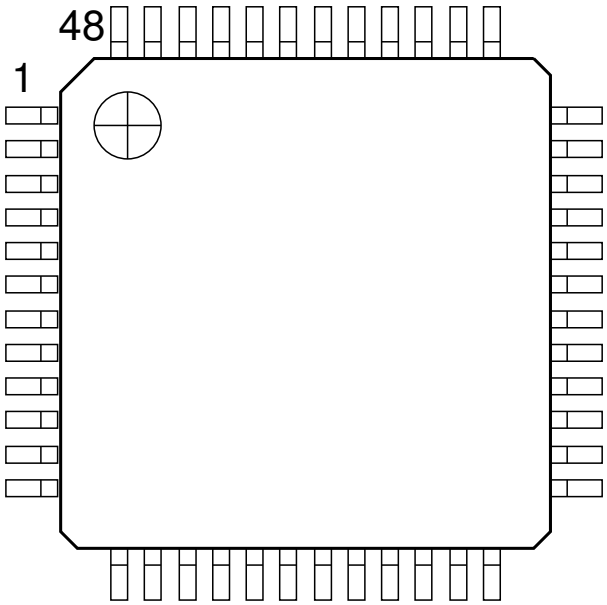


Figure 1: Pin configuration, LQFP-48

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

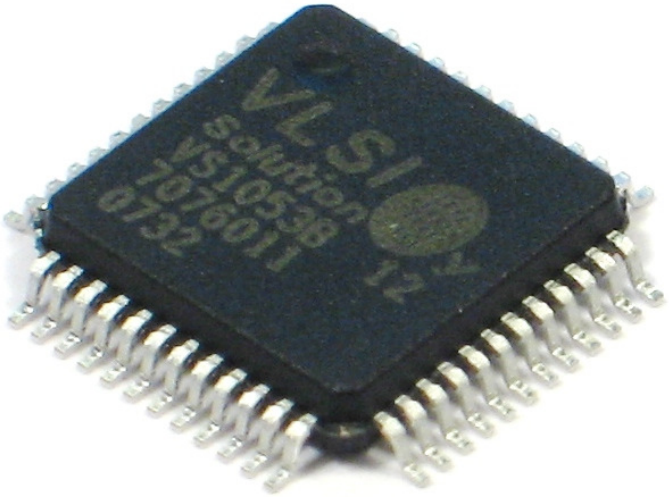


Figure 2: VS1063a in LQFP-48 packaging

Pad Name	LQFP Pin	Pin Type	Function
MICP / LINE1	1	AI	Positive differential mic input, self-biasing / Line-in 1
MICN	2	AI	Negative differential mic input, self-biasing
XRESET	3	DI	Active low asynchronous reset, schmitt-trigger input
DGND0	4	DGND	Core & I/O ground
CVDD0	5	CPWR	Core power supply
IOVDD0	6	IOPWR	I/O power supply
CVDD1	7	CPWR	Core power supply
DREQ	8	DO	Data request, input bus
GPIO2 / DCLK[1]	9	DIO	General purpose IO 2 / serial input data bus clock
GPIO3 / SDATA[1]	10	DIO	General purpose IO 3 / serial data input
GPIO6 / I2S_SCLK[3]	11	DIO	General purpose IO 6 / I2S_SCLK
GPIO7 / I2S_SDATA[3]	12	DIO	General purpose IO 7 / I2S_SDATA
XDCS / BSYNC[1]	13	DI	Data chip select / byte sync
IOVDD1	14	IOPWR	I/O power supply
VCO	15	DO	For testing only (Clock VCO output)
DGND1	16	DGND	Core & I/O ground
XTALO	17	AO	Crystal output
XTALI	18	AI	Crystal input
IOVDD2	19	IOPWR	I/O power supply
DGND2	20	DGND	Core & I/O ground
DGND3	21	DGND	Core & I/O ground
DGND4	22	DGND	Core & I/O ground
XCS	23	DI	Chip select input (active low)
CVDD2	24	CPWR	Core power supply
GPIO5 / I2S_MCLK[3]	25	DIO	General purpose IO 5 / I2S_MCLK
RX	26	DI	UART receive, connect to IOVDD if not used
TX	27	DO	UART transmit
SCLK	28	DI	Clock for serial bus
SI	29	DI	Serial input
SO	30	DO3	Serial output
CVDD3	31	CPWR	Core power supply
XTEST	32	DI	Reserved for test, connect to IOVDD
GPIO0	33	DIO	Gen. purp. IO 0 (SPIBOOT), use 100 kΩ pull-down resistor[2]
GPIO1	34	DIO	General purpose IO 1
GND	35	DGND	I/O Ground
GPIO4 / I2S_LROUT[3]	36	DIO	General purpose IO 4 / I2S_LROUT
AGND0	37	APWR	Analog ground, low-noise reference
AVDD0	38	APWR	Analog power supply
RIGHT	39	AO	Right channel output
AGND1	40	APWR	Analog ground
AGND2	41	APWR	Analog ground
GBUF	42	AO	Common buffer for headphones, do NOT connect to ground!
AVDD1	43	APWR	Analog power supply
RCAP	44	AIO	Filtering capacitance for reference
AVDD2	45	APWR	Analog power supply
LEFT	46	AO	Left channel output
AGND3	47	APWR	Analog ground
LINE2	48	AI	Line-in 2 (right channel)

[1] First pin function is active in New Mode, latter in Compatibility Mode.

[2] Unless pull-down resistor is used, SPI Boot, followed by I2C Boot, is tried. See Chapters 11.8, SPI Boot, and 11.9, I2C Boot, for details.

[3] If I2S_CF_ENA is '0' the pins are used for GPIO. See Chapter *I2S DAC Interface* from *VS1063a Hardware Guide* or Chapter 11.12, *I2S Output* of this document for details.

Pin types:

Type	Description
DI	Digital input, CMOS Input Pad
DO	Digital output, CMOS Input Pad
DIO	Digital input/output
DO3	Digital output, CMOS Tri-stated Output Pad
AI	Analog input

Type	Description
AO	Analog output
AIO	Analog input/output
APWR	Analog power supply pin
DGND	Core or I/O ground pin
CPWR	Core power supply pin
IOPWR	I/O power supply pin

7 Connection Diagram, LQFP-48

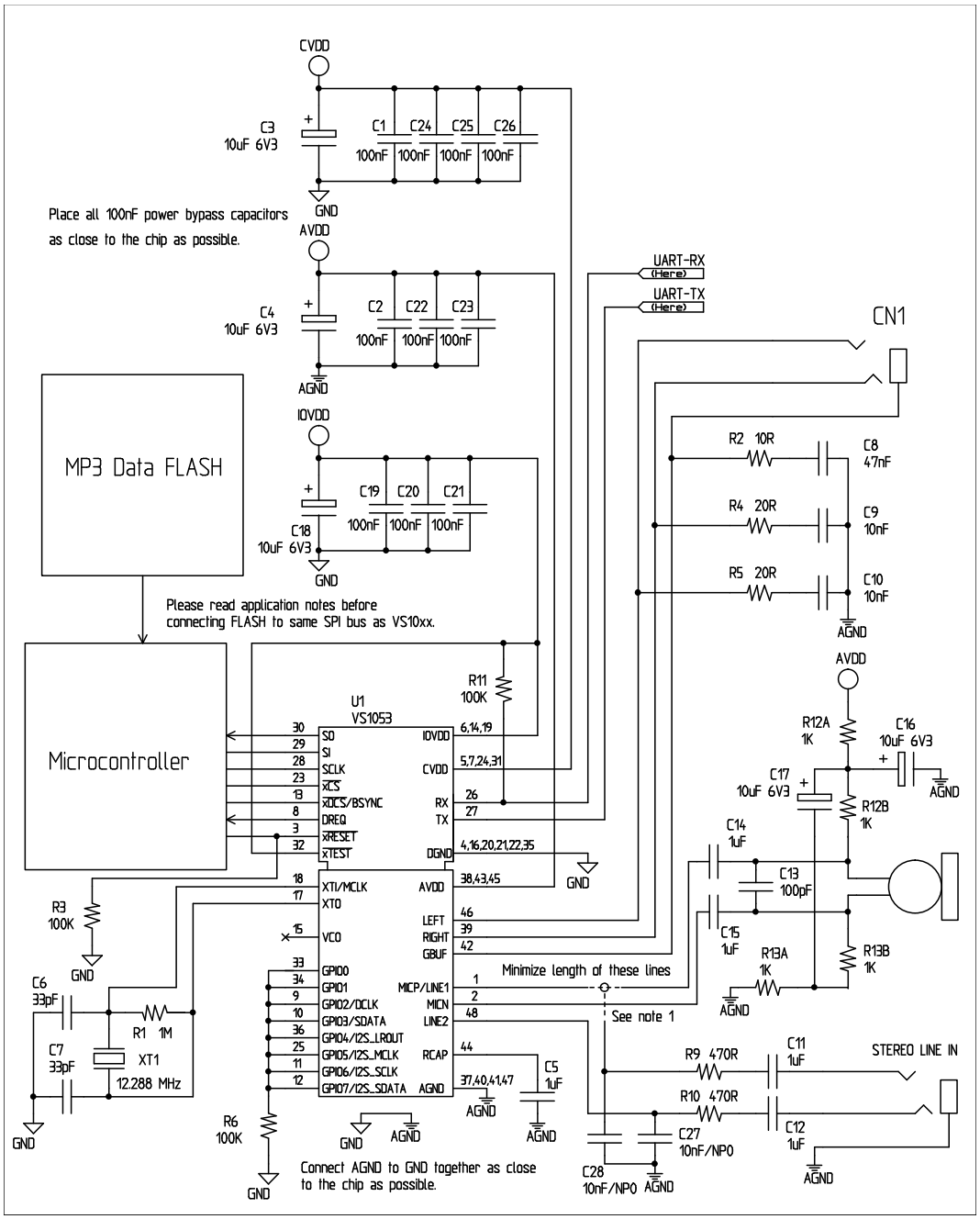


Figure 3: Typical connection diagram using LQFP-48

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 10.8.1). If also SM_SDISHARE is used, xDCS should be tied high (see Chapter 8.1.1).

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.

8 SPI Buses

The SPI Bus - which was originally used in some Motorola devices - has been used for both VS1063a's Serial Data Interface SDI (Chapters 8.3 and 10.6) and Serial Control Interface SCI (Chapters 8.4 and 10.7).

8.1 SPI Bus Pin Descriptions

8.1.1 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).

SDI Pin	SCI Pin	Description
XDCS	XCS	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.
SCK		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.
SI		Serial input. If a chip select is active, SI is sampled on the rising CLK edge.
-	SO	Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.

8.1.2 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.

SDI Pin	SCI Pin	Description
-	XCS	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.
BSYNC	-	SDI data is synchronized with a rising edge of BSYNC.
DCLK	SCK	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.
SDATA	SI	Serial input. SI is sampled on the rising SCK edge, if XCS is low.
-	SO	Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.

8.2 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 an SCI command.

Because of the 32-byte safety area, the sender may send upto 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 can not determine if an SCI command has been executed if SDI is not ready to receive data. In this case you need a long enough delay after every SCI command to make certain none of them are missed. The SCI Registers table in Chapter 10.8 gives the worst-case handling time for each SCI register write.

Note: The status of DREQ can also be read through SCI with the following code. For details on SCI registers, see Chapter 8.4.

```
// This example reads status of DREQ pin through the SPI/SCI register
// interface.
#define SCI_WRAMADDR 7
#define SCI_WRAM 6
while (!endOfFile) {
  int dreq;
  WriteSciReg(SCI_WRAMADDR, 0xC012); // Send address of DREQ register
  dreq = ReadSciReg(SCI_WRAM) & 1;   // Read value of DREQ (in bit 0)
  if (dreq) {
    // DREQ high: send 1-32 bytes audio data
  } else {
    // DREQ low: wait 5 milliseconds (so that VS10xx doesn't get
    // continuous SCI operations)
  }
} /* while (!endOfFile) */
```

8.3 Serial Protocol for Serial Data Interface (SPI / 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 10.8).

VS1063a assumes its data input to be byte-sychronized. SDI bytes may be transmitted either MSb or LSb first, depending of register SCI_MODE bit SM_SDIORD (Chapter 10.8.1).

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

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

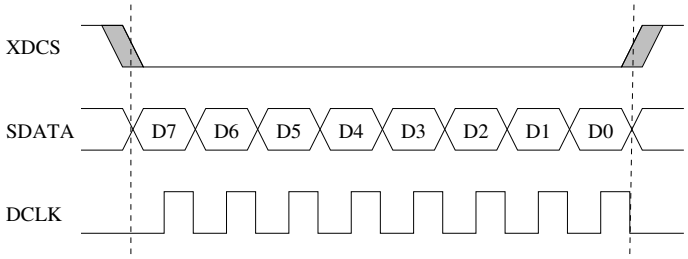


Figure 4: SDI in VS10xx Native Mode, single-byte transfer

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.

Note that when sending data through SDI you have to check the Data Request Pin DREQ at least after every 32 bytes (Chapter 8.2).

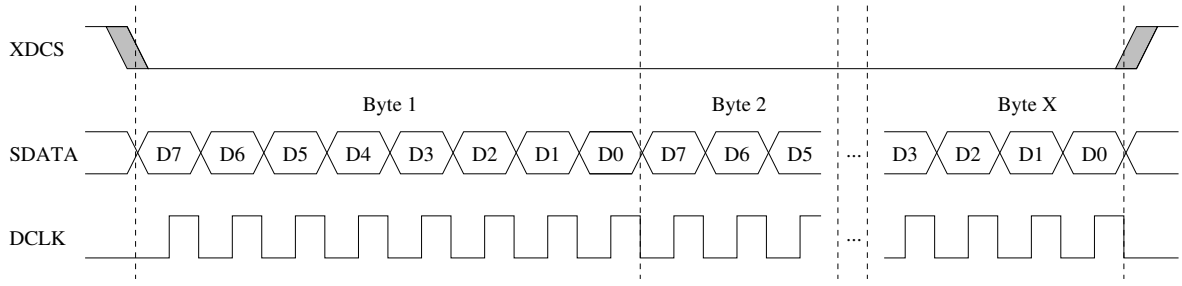


Figure 5: SDI in VS10xx Native Mode, multi-byte transfer, $X \geq 1$

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

8.3.2 SDI Timing Diagram in VS10xx Native Modes (New Mode)

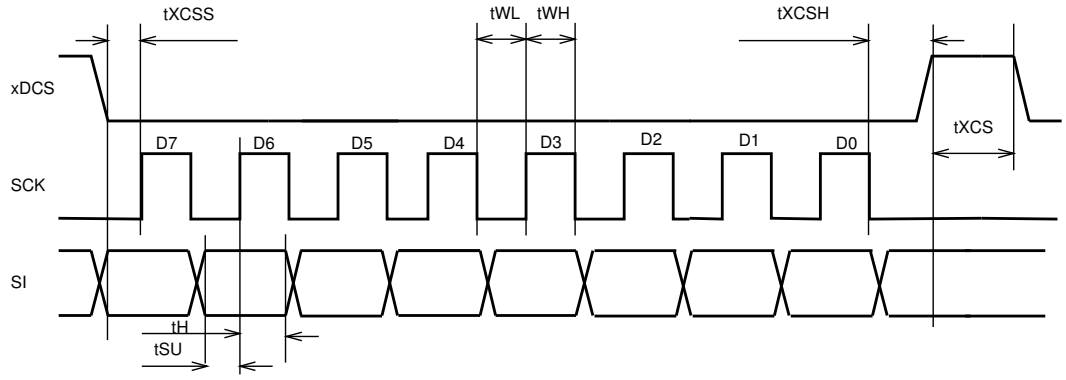


Figure 6: SDI timing diagram

Figure 6 presents SDI bus timing.

Symbol	Min	Max	Unit
tXCSS	5		ns
tSU	0		ns
tH	2		CLKI cycles
tWL	2		CLKI cycles
tWH	2		CLKI cycles
tXCSH	1		CLKI cycles
tXCS	0		CLKI cycles

Note: xDCS is not required to go high between bytes, so tXCS is 0.

Note: Although the timing is derived from the internal clock CLKI, the system always starts up in $1.0\times$ 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.

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

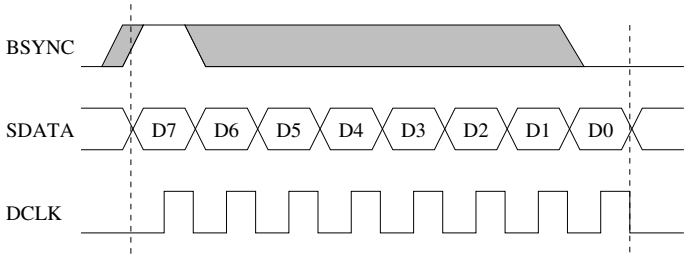


Figure 7: SDI in VS1001 Mode - one byte transfer. Do not use in new designs!

When VS1063a is running in VS1001 compatibility mode, a BSYNC signal must be generated to ensure correct bit-alignment of the input bitstream, as shown in Figures 7 and 8.

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.

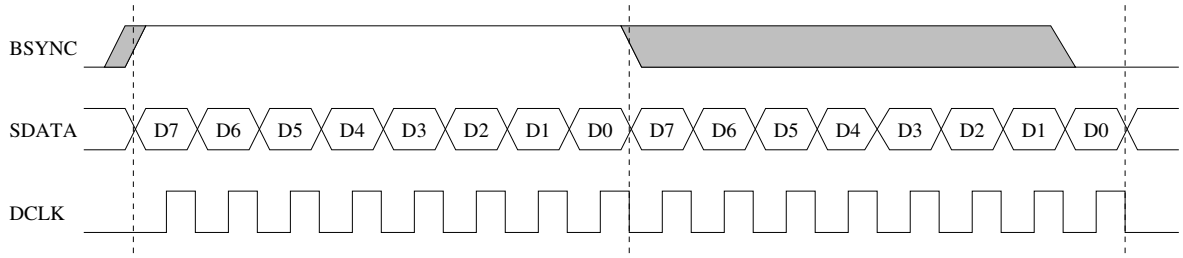


Figure 8: SDI in VS1001 Mode - two byte transfer. Do not use in new designs!

8.3.4 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.

8.4 Serial Protocol for Serial Command Interface (SPI / SCI)

The serial bus protocol for the Serial Command Interface SCI (Chapter 10.7) 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.

Instruction		
Name	Opcode	Operation
READ	0b0000 0011	Read data
WRITE	0b0000 0010	Write data

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.

8.4.1 SCI Read

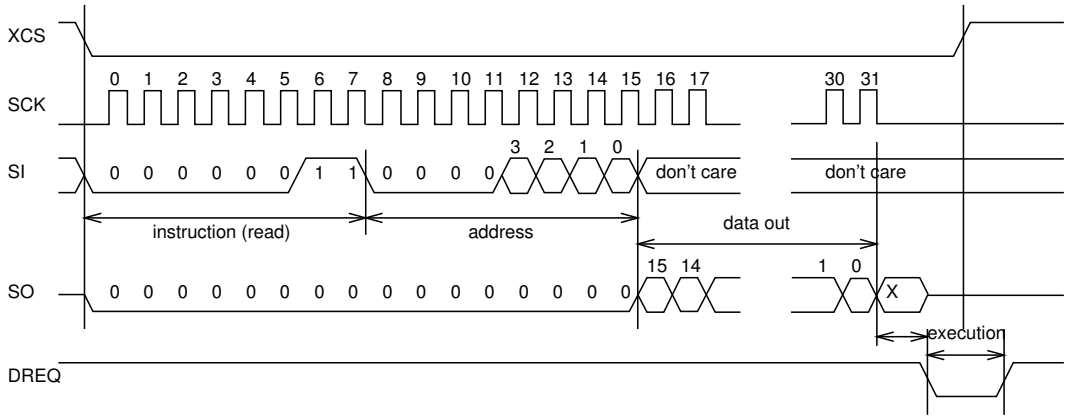


Figure 9: SCI word read

VS1063a registers are read from using the following sequence, as shown in Figure 9. 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.

8.4.2 SCI Write

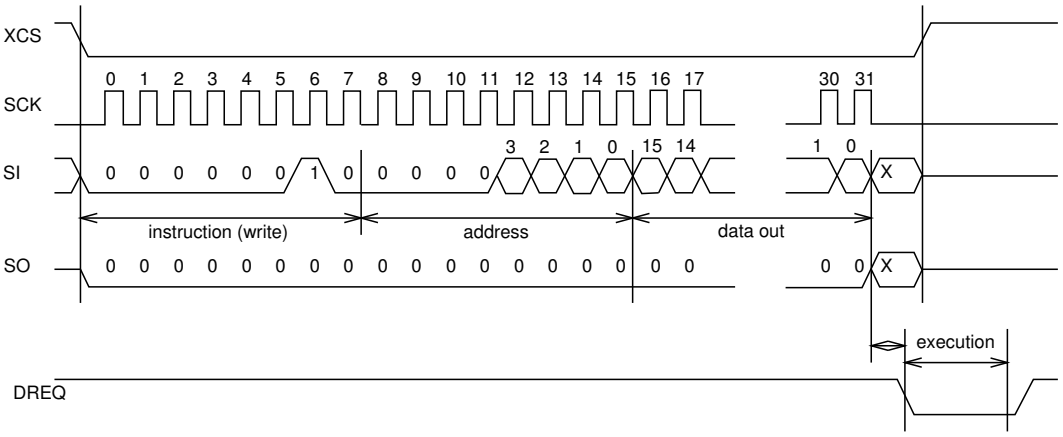


Figure 10: SCI word write

VS1063a registers are written from using the following sequence, as shown in Figure 10. 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 10.8 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.

8.4.3 SCI Multiple Write

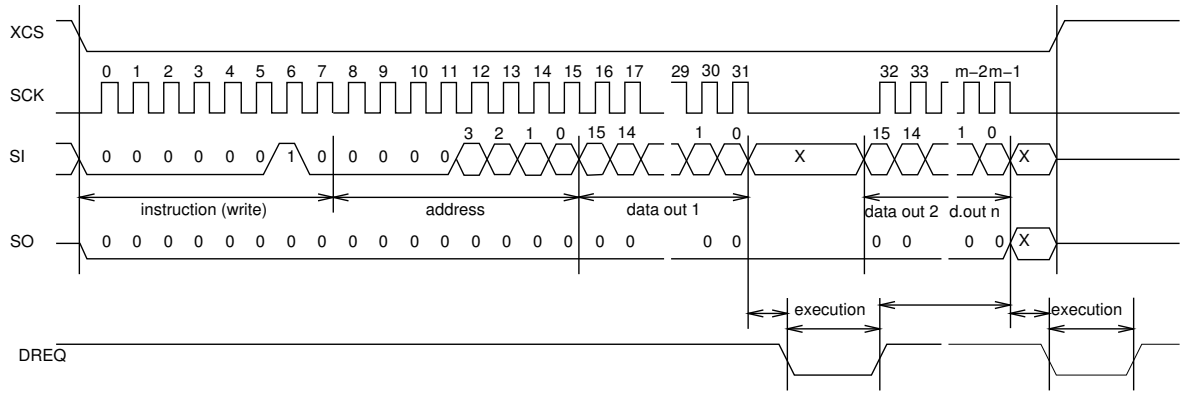


Figure 11: SCI multiple word write

VS1063a allows for the user to send multiple words to the same SCI register, which allows fast SCI uploads, shown in Figure 11. 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 10.8 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.

8.4.4 SCI Timing Diagram

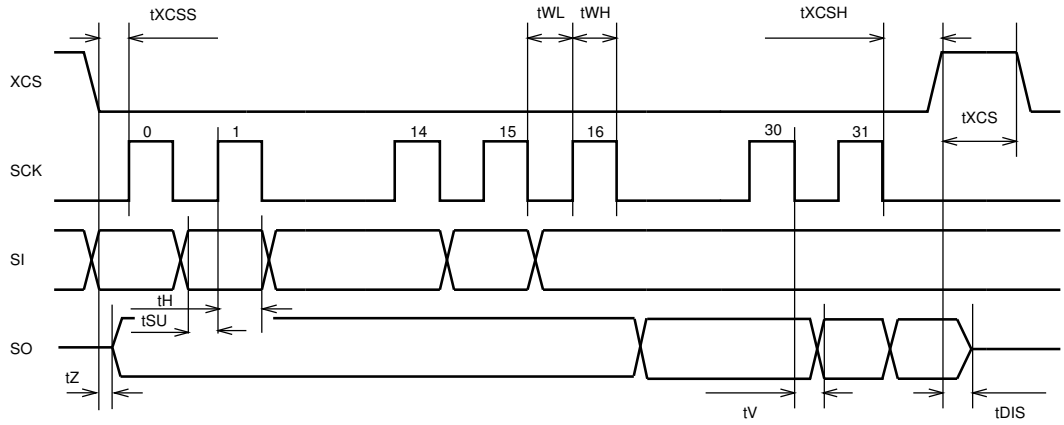


Figure 12: SPI timing diagram

The SCI timing diagram is presented in Figure 12.

Symbol	Min	Max	Unit
tXCSS	5		ns
tSU	0		ns
tH	2		CLKI cycles
tZ	0		ns
tWL	2		CLKI cycles
tWH	2		CLKI cycles
tV		2 (+ 25 ns[1])	CLKI cycles
tXCSH	1		CLKI cycles
tXCS	2		CLKI cycles
tDIS		10	ns

[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 $1.0\times$ 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 $6\times$CLKI + 25 ns, the maximum speed for SCI reads is CLKI/7.

8.5 SPI Examples with SM_SDINEW and SM_SDISHARED set

8.5.1 Two SCI Writes

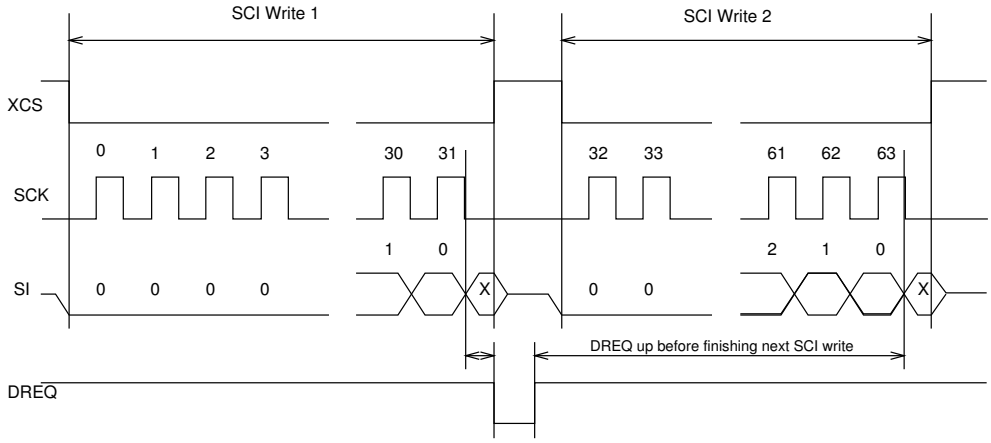


Figure 13: Two SCI operations

Figure 13 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.

8.5.2 Two SDI Bytes

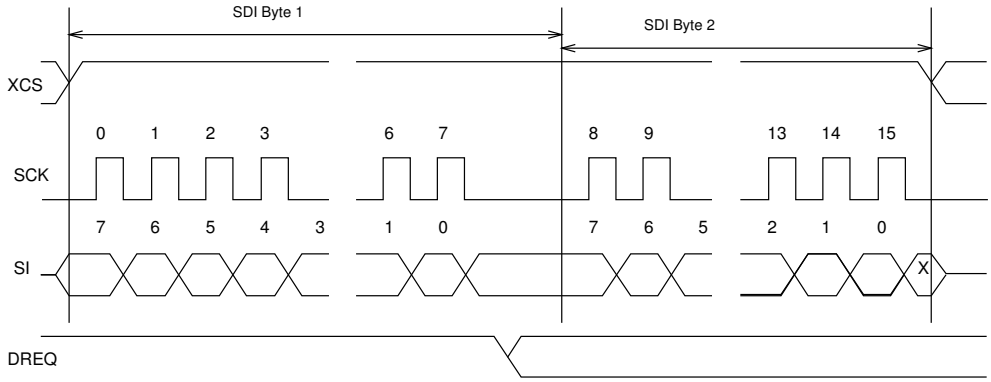


Figure 14: Two SDI bytes

SDI data is synchronized with a raising edge of xCS as shown in Figure 14. However, every byte doesn't need separate synchronization.

8.5.3 SCI Operation in Middle of Two SDI Bytes

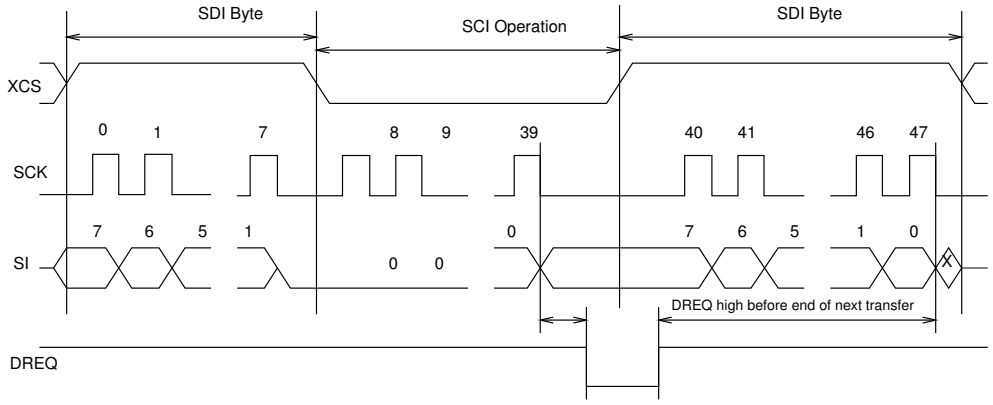


Figure 15: Two SDI bytes separated by an SCI operation

Figure 15 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.

9 Supported Audio Formats

9.1 Supported Audio Formats Overview

VS1063a supports many audio formats. Some are available as encoders, some as decoders, and some are available as full-duplex codecs.

Acoustic Echo Cancellation (AEC) can be applied to the codecs.

Codecs may be mixed. It is for example possible to decode IMA ADPCM, and at the same time encode in PCM.

The table below shows supported audio Decoders, Encoders, and Full-Duplex Codecs:

Audio Format	Decoder	Encoder	Codec
MP3 1.0, 2.0, 2.5	+[1]	+[2]	
Ogg Vorbis	+	+	
MPEG4/2 AAC-LC+PNS	+		
HE-AAC v2 (Level 3), SBR+PS	+		
FLAC	+		
WMA 4.0/4.1/7/8/9	+		
MP2[3]	+[1,3]		
(WAV) PCM	+	+	+
(WAV) G.711 μ-law, A-law	+	+	+
(WAV) G.722 ADPCM	+	+	+
(WAV) IMA ADPCM	+	+	+

[1] Not available in VS8063a
[2] Not available in VS8063a or VS1163a
[3] MP2 is optional, it is not activated by default.

9.2 Supported Audio Decoders

Conventions	
Mark	**Description**
+	Format is supported
-	Format exists but is not supported
	Format doesn't exist

9.2.1 Supported MP3 (MPEG layer III) Decoder Formats (Not VS8063a)

The VS1063 MP3 decoder is full-accuracy compliant.

MP3 MPEG 1.0[1]:

Samplerate / Hz	**Bitrate / kbit/s**													
	32	40	48	56	64	80	96	112	128	160	192	224	256	320
48000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
44100	+	+	+	+	+	+	+	+	+	+	+	+	+	+
32000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

MP3 MPEG 2.0[1]:

Samplerate / Hz	**Bitrate / kbit/s**													
	8	16	24	32	40	48	56	64	80	96	112	128	144	160
24000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
22050	+	+	+	+	+	+	+	+	+	+	+	+	+	+
16000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

MP3 MPEG 2.5[1]:

Samplerate / Hz	**Bitrate / kbit/s**													
	8	16	24	32	40	48	56	64	80	96	112	128	144	160
12000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
11025	+	+	+	+	+	+	+	+	+	+	+	+	+	+
8000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

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

9.2.2 Supported MP2 (MPEG layer II) Decoder Formats (Not VS8063a)

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

MP2 MPEG 1.0:

Samplerate / Hz	**Bitrate / kbit/s**													
	32	48	56	64	80	96	112	128	160	192	224	256	320	384
48000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
44100	+	+	+	+	+	+	+	+	+	+	+	+	+	+
32000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

MP2 MPEG 2.0:

Samplerate / Hz	**Bitrate / kbit/s**													
	8	16	24	32	40	48	56	64	80	96	112	128	144	160
24000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
22050	+	+	+	+	+	+	+	+	+	+	+	+	+	+
16000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

9.2.3 Supported Ogg Vorbis Decoder Formats

Parameter	Min	Max	Unit
Channels	1	2	
Window size	64	4096	samples
Samplerate	1	48000	Hz
Bitrate	0	500	kbit/sec

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.

9.2.4 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 11.10, *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 (.aac) 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 11.5.4).

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`.

AAC[1][2]:

Samplerate / Hz	Maximum Bitrate kbit/s - for 2 channels								
	≤96	132	144	192	264	288	384	529	576
48000	+	+	+	+	+	+	+	+	+
44100	+	+	+	+	+	+	+	+	
32000	+	+	+	+	+	+	+		
24000	+	+	+	+	+	+			
22050	+	+	+	+	+				
16000	+	+	+	+					
12000	+	+	+						
11025	+	+							
8000	+								

[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.

9.2.5 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:

Samplerate	Bitrate / kbit/s																
/ Hz	5	6	8	10	12	16	20	22	32	40	48	64	80	96	128	160	192
8000	+	+	+		+												
11025			+	+													
16000				+	+	+	+										
22050						+	+	+	+								
32000							+	+	+	+	+	+					
44100									+		+	+	+	+	+	+	
48000															+	+	

WMA 7:

Samplerate	Bitrate / kbit/s																
/ Hz	5	6	8	10	12	16	20	22	32	40	48	64	80	96	128	160	192
8000	+	+	+		+												
11025			+	+													
16000				+	+	+	+										
22050						+	+	+	+								
32000							+		+	+	+						
44100									+		+	+	+	+	+	+	+
48000															+	+	

WMA 8:

Samplerate	Bitrate / kbit/s																
/ Hz	5	6	8	10	12	16	20	22	32	40	48	64	80	96	128	160	192
8000	+	+	+		+												
11025			+	+													
16000				+	+	+	+										
22050						+	+	+	+								
32000							+		+	+	+						
44100									+		+	+	+	+	+	+	+
48000															+	+	+

WMA 9:

Samplerate	Bitrate / kbit/s																		
/ Hz	5	6	8	10	12	16	20	22	32	40	48	64	80	96	128	160	192	256	320
8000	+	+	+		+														
11025			+	+															
16000				+	+	+	+												
22050						+	+	+	+										
32000							+		+	+	+								
44100							+		+		+	+	+	+	+	+	+	+	+
48000												+		+	+	+	+		

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.

9.2.6 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).

9.2.7 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).

Format	Name	Comments
0x01	PCM	32, 24, 16 and 8 bits
0x03	IEEE_FLOAT	IEEE floating point data
0x06	ALAW	non-linear-quantized 8-bit samples (G.711 A-law)
0x07	MULAW	non-linear-quantized 8-bit samples (G.711 μ-law)
0x11	IMA_ADPCM	4 bits per sample
0x55	MPEGLAYER3	For supported MP3 modes, see Chapter 9.2.1
0x65	G722_ADPCM	two samples in 8 bits, same as 0x28f
0x28f	ADPCM_G722	two samples in 8 bits, same as 0x65
0xfffe	Extended PCM	32, 24, 16 and 8 bits, default channel configuration supported

9.3 Supported Audio Encoding Formats

9.3.1 Supported MP3 (MPEG layer III) Encoding Formats (VS1063a Only)

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	
Symbol	**Description**
++	Format is supported and recommended for this channel configuration and bitrate.
+	Format is supported.
x	Format is supported but use is strongly discouraged for quality reasons.
v	Format is supported but for best quality lower samplerate with same bitrate is recommended.
<	Format is supported but lower bitrate will give same quality.
-	Format exists but isn't supported.
	Format doesn't exist.

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

Samplerate / Hz	**Bitrate / kbit/s, stereo**													
	32	40	48	56	64	80	96	112	128	160	192	224	256	320
48000	v	v	v	v	v	v	v	+	+	++	++	++	++	++
44100	v	v	v	v	v	v	+	+	+	+	+	+	+	+
32000	v	v	v	v	v	+	+	++	++	+	+	+	+	<

MPEG 2.0 & 2.5 layer III (MP3 low rates), stereo:

Samplerate / Hz	**Bitrate / kbit/s, stereo**													
	8	16	24	32	40	48	56	64	80	96	112	128	144	160
24000	x	v	v	v	v	v	v	+	++	++	+	+	+	<
22050	x	v	v	v	v	v	+	+	+	+	+	+	<	<
16000	x	v	v	v	+	+	+	++	+	+	+	+	<	<
12000	v	v	v	+	++	++	++	+	+	+	+	+	<	<
11025	v	v	v	+	+	+	+	+	+	+	+	+	<	<
8000	++	++	++	++	+	+	+	+	+	+	+	<	<	<

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

Samplerate / Hz	**Bitrate / kbit/s, mono**													
	32	40	48	56	64	80	96	112	128	160	192	224	256	320
48000	v	v	v	+	+	++	++	++	++	++	++	++	++	<
44100	v	v	v	+	+	+	+	+	+	+	+	+	+	<
32000	+	+	+	++	++	+	+	+	+	+	+	<	<	<

MPEG 2.0 & 2.5 layer III (MP3 low rates), mono:

Samplerate / Hz	**Bitrate / kbit/s, mono**													
	8	16	24	32	40	48	56	64	80	96	112	128	144	160
24000	v	v	+	+	+	++	+	+	+	+	+	<	<	<
22050	v	v	+	+	+	+	+	+	+	+	+	<	<	<
16000	v	v	+	++	++	+	+	+	+	<	<	<	<	<
12000	v	v	++	+	+	+	+	+	<	<	<	<	<	<
11025	v	v	+	+	+	+	+	+	<	<	<	<	<	<
8000	++	++	+	+	+	+	<	<	<	<	<	<	<	<

9.3.2 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.

Voice: 8000 Hz mono											
Quality setting	0	1	2	3	4	**5**	6	7	8	9	10
Typical kbit/s	6	7	9	10	12	**13**	16	19	22	25	28

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

Wideband Voice: 16000 Hz mono											
Quality setting	0	1	2	3	4	**5**	6	7	8	9	10
Typical kbit/s	7	11	14	18	21	**25**	31	37	43	49	55

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

Wideband Stereo Voice: 16000 Hz stereo											
Quality setting	0	1	2	3	4	**5**	6	7	8	9	10
Typical kbit/s	10	18	26	34	42	**50**	65	81	96	112	127

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

HiFi Voice, 48000 Hz mono											
Quality setting	0	1	2	3	4	**5**	6	7	8	9	10
Typical kbit/s	37	47	57	68	78	**88**	99	110	122	133	144

The “Music” profiles are intended for HiFi music.

HiFi Voice, 48000 Hz stereo											
Quality setting	0	1	2	3	4	**5**	6	7	8	9	10
Typical kbit/s	53	72	91	110	129	**148**	185	222	259	296	333

9.3.3 Supported RIFF WAV Encoding Formats

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

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

10 Functional Description

10.1 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 samplerates 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.

10.2 Decoder Data Flow of VS1063a

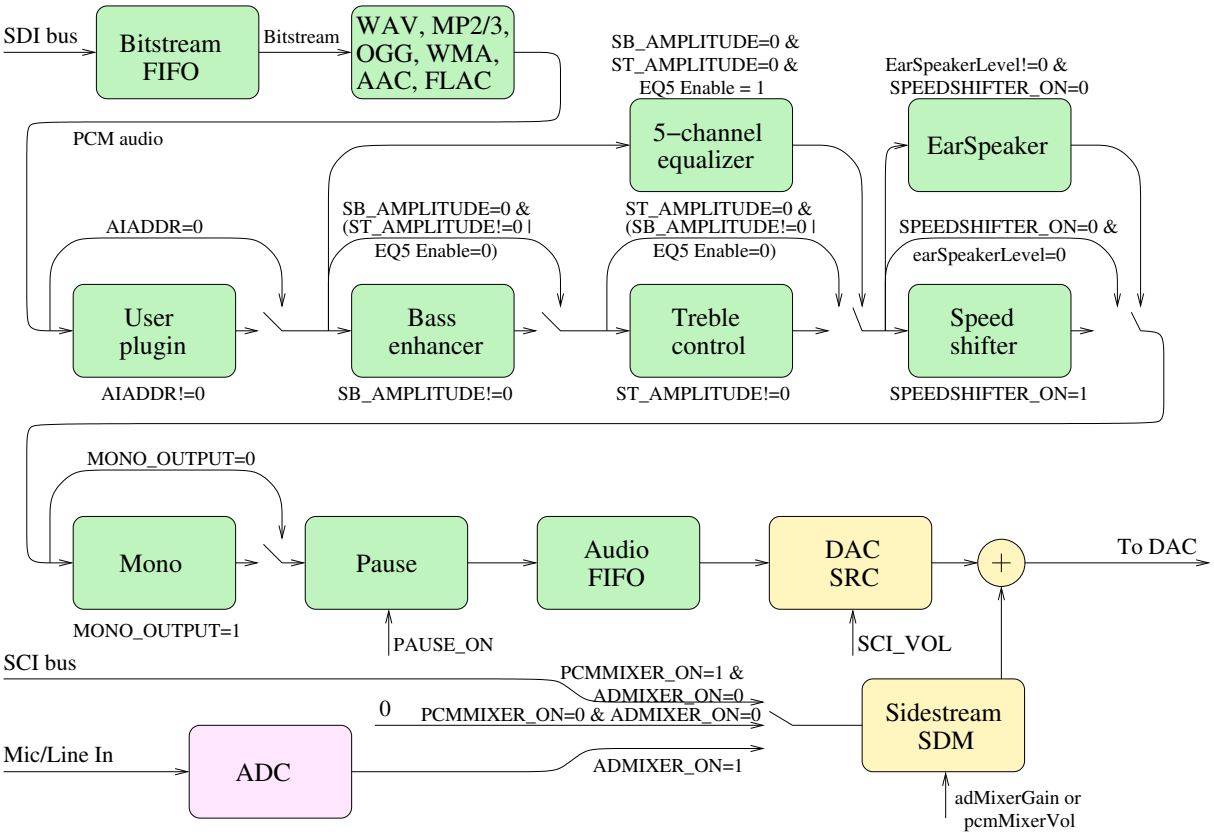


Figure 16: Decoder data flow of VS1063a

Figure 16 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 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 XTALI/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.

10.3 Encoder Data Flow of VS1063a

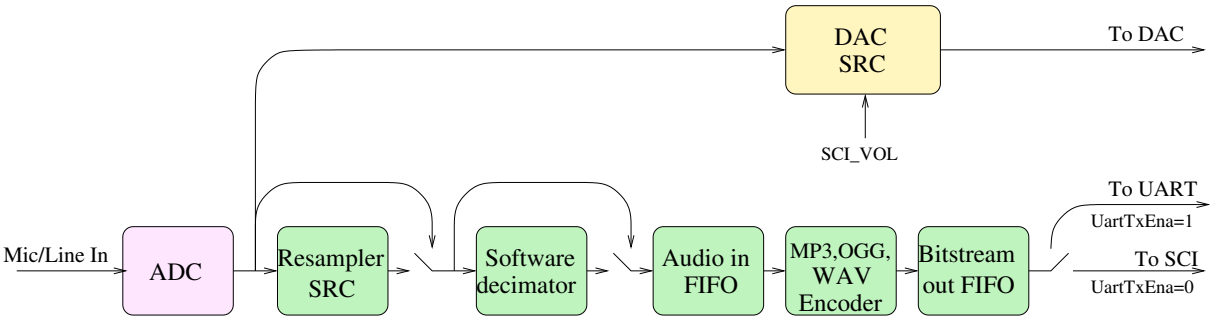


Figure 17: Encoder data flow of VS1063a

Figure 17 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.

10.4 Codec Data Flow of VS1063a

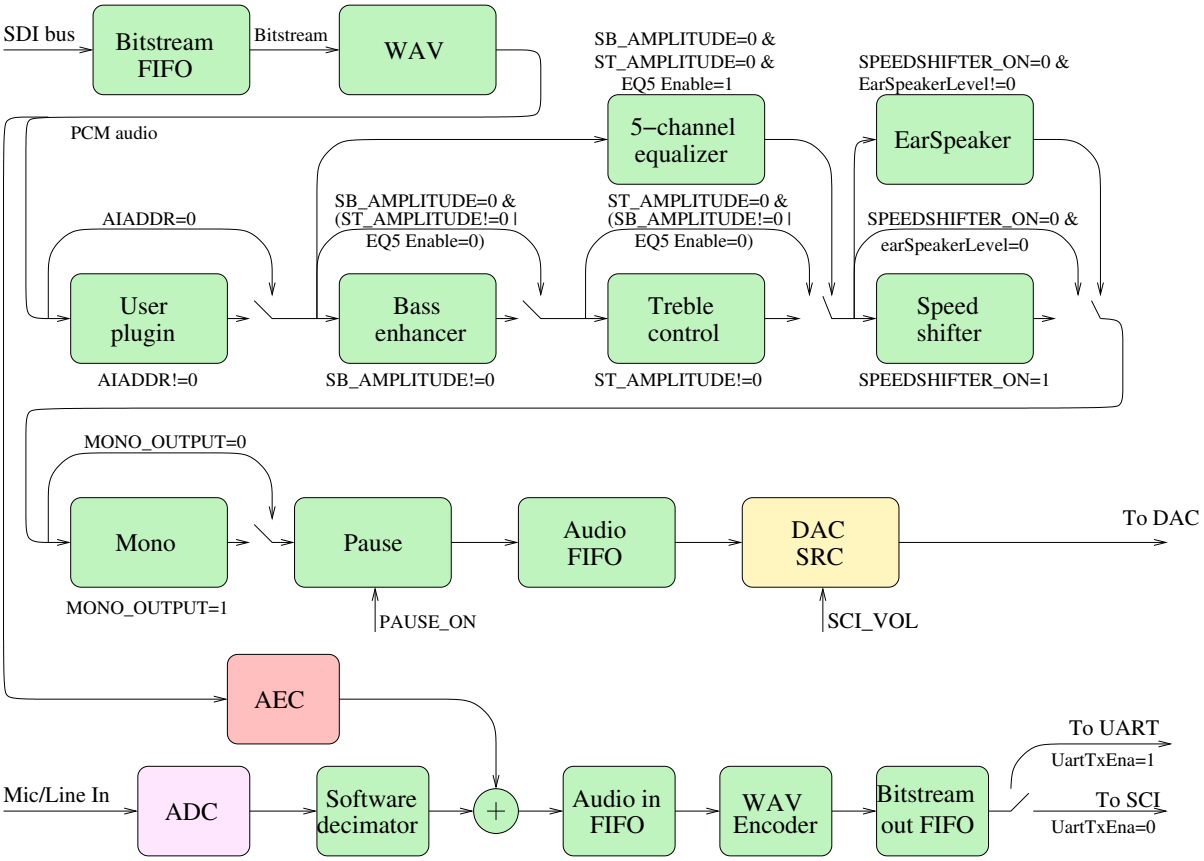


Figure 18: Codec data flow of VS1063a

Figure 18 presents the codec dataflow of VS1063a.

The decoder and encoder paths are almost similar as in the decoder and encoder data flow Chapters 10.2 and 10.3, 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 opened 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.

10.5 EarSpeaker Spatial Processing

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 19). 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.

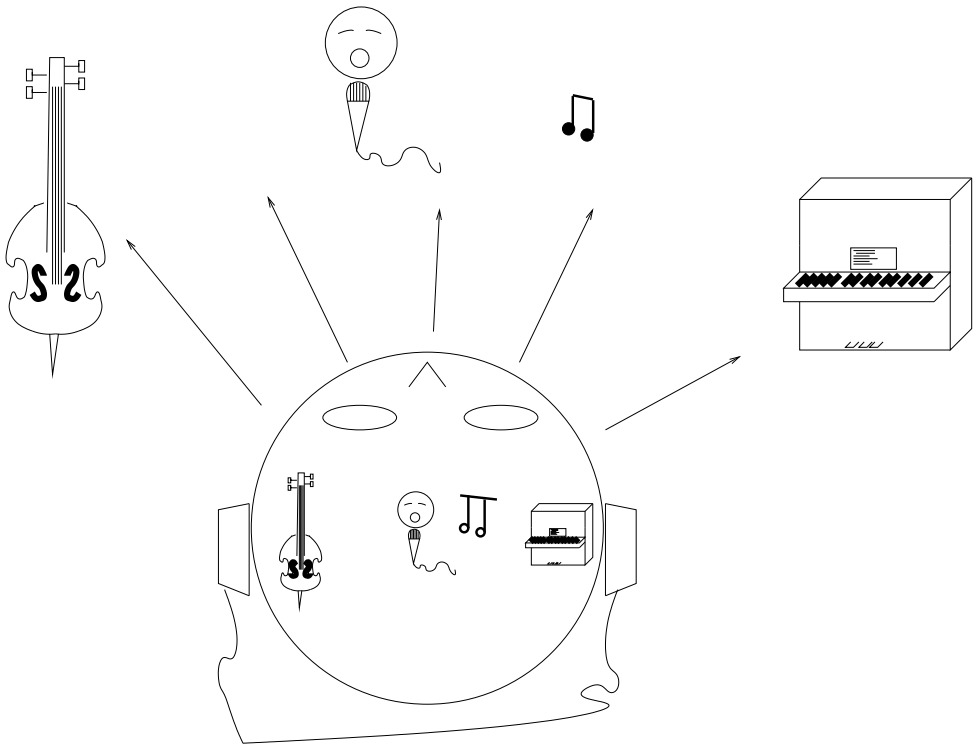


Figure 19: EarSpeaker externalized sound sources vs. normal inside-the-head sound

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 11.10.8.

10.6 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 11.

10.7 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

10.8 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 an SCI operation because the SDI can not accept more data, make certain there is enough time to complete the operation before sending another.

SCI registers					
Reg	**Typ.**	**Reset**	**Write Time**[1]	**Name**	**Description**
0x0	rw	0x4000[6]	80 CLKI[4]	SCI_MODE	Mode control
0x1	rw	0x000C[3]	80 CLKI	SCI_STATUS	Status of VS1063a
0x2	rw	0	80 CLKI	SCI_BASS	Built-in bass/treble control
0x3	rw	0	1200 XTALI[5]	SCI_CLOCKF	Clock freq + multiplier
0x4	rw	0	100 CLKI	SCI_DECODE_TIME	Decode time in seconds
0x5	rw	0	450 CLKI[2]	SCI_AUDATA	Misc. audio data
0x6	rw	0	100 CLKI	SCI_WRAM	RAM write/read
0x7	rw	0	100 CLKI	SCI_WRAMADDR	Set address for RAM write/read
0x8	r	0	80 CLKI	SCI_HDAT0	Stream header data 0
0x9	r	0	80 CLKI	SCI_HDAT1	Stream header data 1
0xA	rw	0	210 CLKI[2]	SCI_AIADDR	Start address of application
0xB	rw	0	80 CLKI	SCI_VOL	Volume control
0xC	rw	0	80 CLKI[2]	SCI_AICTRL0	Application control register 0
0xD	rw	0	80 CLKI[2]	SCI_AICTRL1	Application control register 1
0xE	rw	0	80 CLKI[2]	SCI_AICTRL2	Application control register 2
0xF	rw	0	80 CLKI[2]	SCI_AICTRL3	Application control register 3

[1] 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.

[2] In addition, the cycles spent in the user application/plugin routine must be counted.

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

[4] When mode register write specifies a software reset the worst-case time is 22000 XTALI cycles.

[5] If the clock multiplier is changed, writing to SCI_CLOCKF register may force internal clock to run at $1.0 \times$ XTALI for a while. Thus it is not a good idea to send SCI or SDI bits while this register update is in progress.

[6] 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 *VS1063a Hardware Guide* for details.

10.8.1 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.

SCI_MODE bits					
Name	**Bit**	**Mode**	**Function**	**Value**	**Description**
SM_DIFF	0	SW	Differential	0 1	normal in-phase audio left channel inverted
SM_LAYER12	1	SW	Allow MPEG layer II	0 1	no yes
SM_RESET	2	SW	Soft reset	0 1	no reset reset
SM_CANCEL	3	SW	Cancel decoding current file	0 1	no yes
	4	SW	reserved	0 1	right wrong
SM_TESTS	5	SW	Allow SDI tests	0 1	not allowed allowed
	6	SW	reserved	0 1	right wrong
	7	SW	reserved	0 1	right wrong
SM_DACT	8	HW	DCLK active edge	0 1	rising falling
SM_SDIORD	9	HW	SDI bit order	0 1	MSb first MSb last
SM_SDISHARE	10	HW	Share SPI chip select	0 1	no yes
SM_SDINEW	11	HW	VS10xx native SPI modes	0 1	no yes
SM_ENCODE	12	SW	Activate Encoding	0 1	no yes
-	13	SW	-	0 1	right wrong
SM_LINE1	14	HW	MIC / LINE1 selector	0 1	MICP LINE1
SM_CLK_RANGE	15	HW	Input clock range	0 1	12...13 MHz 24...26 MHz

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 11.5.2 for details.

If SM_TESTS is set, SDI tests are allowed. For more details on SDI tests, look at Chapter 11.11.

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 8.1, if also SM_SDINEW is set.

Setting SM_SDINEW will activate VS10xx native serial modes as described in Chapters 8.1.1 and 8.3.1. 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 Chapters 11.6 and 11.7). 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.

10.8.2 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.

SCI_STATUS bits			
Name	**Bits**	**Mode**	**Description**
SS_DO_NOT_JUMP	15	SW	Header in decode, do not fast forward/rewind
SS_SWING	14:12	HW	Set swing to +0 dB, +0.5 dB, . . . , or +3.5 dB
SS_VCM_OVERLOAD	11	HW	GBUF overload indicator ’1’ = overload
SS_VCM_DISABLE	10	HW	GBUF overload detection ’1’ = disable
	9:8	SW	reserved
SS_VER	7:4	SW	Version
SS_APDOWN2	3	HW	Analog driver powerdown
SS_APDOWN1	2	HW	Analog internal powerdown
SS_AD_CLOCK	1	HW	AD clock select, ’0’ = 6 MHz, ’1’ = 3 MHz
SS_REFERENCE_SEL	0	HW	Reference voltage selection, ’0’ = 1.23 V, ’1’ = 1.65 V

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/VS1163, 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 XTALI 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.

10.8.3 SCI_BASS (RW)

SCI_BASS bits		
Name	**Bits**	**Description**
ST_AMPLITUDE	15:12	Treble Control in 1.5 dB steps (-8...7, 0 = off)
ST_FREQLIMIT	11:8	Lower limit frequency in 1000 Hz steps (1...15)
SB_AMPLITUDE	7:4	Bass Enhancement in 1 dB steps (0...15, 0 = off)
SB_FREQLIMIT	3:0	Lower limit frequency in 10 Hz steps (2...15)

The Bass Enhancer VSBE is a 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 2.5 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 11.10.6) is not run.

10.8.4 SCI_CLOCKF (RW)

The external clock multiplier SCI register SCI_CLOCKF is presented in the table below.

SCI_CLOCKF bits		
Name	**Bits**	**Description**
SC_MULT	15:13	Clock multiplier
SC_ADD	12:11	Allowed multiplier addition
SC_FREQ	10: 0	Clock frequency

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:

SC_MULT	MASK	CLKI
0	0x0000	XTALI
1	0x2000	XTALI$\times 2.0$
2	0x4000	XTALI$\times 2.5$
3	0x6000	XTALI$\times 3.0$
4	0x8000	XTALI$\times 3.5$
5	0xa000	XTALI$\times 4.0$
6	0xc000	XTALI$\times 4.5$
7	0xe000	XTALI$\times 5.0$

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:

SC_ADD	MASK	Max multiplier addition
0	0x0000	No modification is allowed
1	0x0800	XTALI$\times 1.0$
2	0x1000	XTALI$\times 1.5$
3	0x1800	XTALI$\times 2.0$

If SC_FREQ is non-zero, it tells that 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: because maximum samplerate is $\frac{XTALI}{256}$, all samplerates 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 XTALI = $1000 \times 4000 + 8000000$ = 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 11.13, *Clock Speed Requirements*, on page 86,

10.8.5 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 not 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 11.10.

10.8.6 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.

10.8.7 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.

10.8.8 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.

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

Note: Unless otherwise specified, only user areas in X, Y, and instruction memory should be accessed.

10.8.9 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 0x574D (“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 and MP2 files, SCI_HDAT1 is between 0xFFE0 and 0xFFFF. SCI_HDAT1 / 0 contain the following:

SCI_HDAT1 and SCI_HDAT0 bits for MP3/2/1			
Bit	**Function**	**Value**	**Explanation**
HDAT1[15:5]	syncword	2047	stream valid
HDAT1[4:3]	ID	3 2 1 0	ISO 11172-3 MPG 1.0 ISO 13818-3 MPG 2.0 (1/2-rate) MPG 2.5 (1/4-rate) MPG 2.5 (1/4-rate)
HDAT1[2:1]	layer	3 2 1 0	I (MP1) II (MP2) III (MP3) reserved
HDAT1[0]	protect bit	1 0	No CRC CRC protected
HDAT0[15:12]	bitrate		see bitrate table
HDAT0[11:10]	samplerate	3 2 1 0	reserved 32/16/ 8 kHz 48/24/12 kHz 44/22/11 kHz
HDAT0[9]	pad bit	1 0	additional slot normal frame
HDAT0[8]	private bit		not defined
HDAT0[7:6]	mode	3 2 1 0	mono dual channel joint stereo stereo
HDAT0[5:4]	extension		see ISO 11172-3
HDAT0[3]	copyright	1 0	copyrighted free
HDAT0[2]	original	1 0	original copy
HDAT0[1:0]	emphasis	3 2 1 0	CCITT J.17 reserved 50/15 microsec none

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:

SCI_HDAT0 field “samplerate”			
Value	**ID=3**	**ID=2**	**ID=0,1**
3	-	-	-
2	32000	16000	8000
1	48000	24000	12000
0	44100	22050	11025

The “bitrate” field in HDAT0 is read according to the following table. Notice that for variable

bitrate stream the value changes constantly.

SCI_HDAT0 field “bitrate”				
Value	Layer II (MP2)		Layer III (Mp3)	
	ID=3	ID=0,1,2	ID=3	ID=0,1,2
	kbit/s		kbit/s	
15	forbidden	forbidden	forbidden	forbidden
14	384	160	320	160
13	320	144	256	144
12	256	128	224	128
11	224	112	192	112
10	192	96	160	96
9	160	80	128	80
8	128	64	112	64
7	112	56	96	56
6	96	48	80	48
5	80	40	64	40
4	64	32	56	32
3	56	24	48	24
2	48	16	40	16
1	32	8	32	8
0	-	-	-	-

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.

10.8.10 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.

10.8.11 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.

SCI_VOL bits		
Name	**Bits**	**Description**
SVOL_LEFT	15:8	Left channel attenuation from maximum in 1/2 dB steps
SVOL_RIGHT	7:0	Right channel attenuation from maximum in 1/2 dB steps

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 → SCI_VOL = 0x0407.

Example: SCI_VOL = 0x2424 → both left and right volumes are 0x24 * -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.

10.8.12 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 11.6 for details.

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

11 Operation

11.1 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 upto 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 stertup. The system clock is then internally divided by 2 at the clock input and the IC gets a 12. . . 13 MHz input clock.

11.2 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 10.8 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 10.8.4 and 11.13 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 is available at *http://www.vlsi.fi/en/support/software/vs10xxplugins.html* .

11.3 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 10.8.1). Then wait for at least 2 μs, 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* .

11.4 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 $1.0\times$ 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.

11.5 Decode Mode

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.

11.5.1 Playing a Whole File

This is the default playback mode.

1. Send an audio file to VS1063a.
2. Read extra parameter value endFillByte (Chapter 11.10).
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_CANCEL.
5. Send at least 32 bytes of endFillByte[7:0].
6. Read SCI_MODE. If SM_CANCEL is still set, go to 5. If SM_CANCEL 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.

11.5.2 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_CANCEL.
3. Continue sending audio file, but check SM_CANCEL after every 32 bytes of data. If it is still set, goto 3. If SM_CANCEL doesn't clear after 2048 bytes or one second, do a software reset (this should be extremely rare).
4. When SM_CANCEL has cleared, read extra parameter value endFillByte (Chapter 11.10).
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.

11.5.3 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 11.10).
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 11.10). 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.

11.5.4 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 11.10).
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.

11.5.5 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 11.10).
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.

11.5.6 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 (unsigned) linear and 16-bit (signed, 2's complement) linear audio is supported in mono or stereo. A WAV header looks like this:

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

The rules to calculate the four variables are as follows:

- $S =$ samplerate 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 datastreams 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....WAVEfmt |
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....|
```

11.6 Encode Mode

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.

11.6.1 Encoding Control Registers

Register	Bits	Description
SCI_MODE	2, 12, 14	Start ENCODE mode, select MIC/LINE1
SCI_AICTRL0	15:0	Samplerate 8000...48000 Hz (read at encoding startup)
SCI_AICTRL1	15:0	Encoding gain (1024 = 1×) or 0 for automatic gain control
SCI_AICTRL2	15:0	Maximum autogain amplification (1024 = 1×, 65535 = 64×)
SCI_AICTRL3	15 14 13 12 11 10 8:9 7:4 3 2:0	codec mode (both encode and decode) AEC enable UART (8N1) TX enable reserved, set to 0 Pause enable No RIFF WAV header inserted (or expected in codec mode) reserved, set to 0 Encoding format 0...6 Reserved, set to 0 ADC mode 0...4
SCI_WRAMADDR	15...0	Quality / bitrate selection for Ogg Vorbis and MP3

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. Note that recording may fail without the *VS1063a Patches* package.

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). In codec mode, and with certain restrictions, it is also possible to activate bit 14, AEC (Ecoustic Echo Cancellation). For details on how to use AEC, see *VS1063a Application Note: AEC with VS1063a*, available at *http://www.vlsi.fi/en/support/applicationnotes.html* .

If SCI_AICTRL3 bit 13 is set at encode/codec startup, UART transmission of data is initialized with parameters taken from `parametric_x.i.encoding` (see Chapter 11.10.9, *Parametric: Encoding*, for details). If you want to use UART transmission, first initialize the required fields of the `parametric_x.i.encoding` structure, then set the SCI_AICTRL3 UART TX enable bit, and only after that start the encoding/codec mode using SCI_MODE register. When in UART mode, do *not* read data through SCI_HDAT0 and SCI_HDAT1!

SCI_AICTRL3 bits 4 to 7 select the encoding format. 0 = IMA ADPCM, 1 = PCM, 2 = G.711 μ-law, 3 = G.711 A-law, 4 = G.722 ADPCM, 5 = Ogg Vorbis, 6 = MP3.
Note: If MP3 encoding is attempted with VS1163a or VS8063a, Ogg Vorbis will be encoded.

SCI_AICTRL3 bits 0 to 2 select the ADC mode and implicitly the number of channels. 0 = joint stereo (common AGC), 1 = dual channel (separate AGC), 2 = left channel, 3 = right channel, 4 = mono downmix.

SCI_WRAMADDR sets the quality / bit rate selection for Ogg Vorbis and MP3 encoding. For WAV formats this setting is not used. Use value 0xe080 for constant bitrate of 128 kbps. Note that WRAMADDR is read at encoder startup, so modifying it later does not change the settings.

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

The `bitrate base` and `bitrate multipler` define a target bitrate value. For example 2 in multiplier 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.

11.6.2 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. It also correct several other recording issues that might otherwise appear.
2. Initialization; Set samplerate and parameters:
   - SCI_CLOCKF for clock: $4.5\times$ for all encoders except Ogg Vorbis (Chapters 11.6.9 and 11.13).
   - 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:
     - *Highly recommended:* If using the *VS1063a Patches* package start encoding by setting bit SM_ENCODE in SCI_MODE, then write 0x50 to SCI_AIADDR.
     - (If you don't use the *VS1063a Patches* package, start encoding mode by setting SM_ENCODE and SM_RESET 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 11.6.3, 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* (highly recommended):

```
// First command line loads VS1063a Patches. The patch package can be
// loaded from http://www.vlsi.fi/en/support/software/vs10xxplugins.html
// 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 (Ogg Vorbis w/ VS1163a)
WriteVS10xxRegister(SCI_WRAMADDR, 0xE0C0); // Set bitrate to CBR 192 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*. This method is not recommended, because e.g. MP3 recordings may have bit errors, and there may be other issues.

```
// Note that this method doesn't load and use the VS1063a Patches package.
// Without the package, MP3 files cannot be properly recorded. There may
// also be other issues, with e.g. audio channel allocation and setting
// the samplerate. VLSI Solution does not support this recording method.
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 downmix mono IMA ADPCM recording from the left channel using the microphone amplifier, with automatic gain control and maximum amplification of $4\times$ (= +12 dB).

11.6.3 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.

11.6.4 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 μ-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 45 66 6d 74 20  |RIFFT ..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, ..|
```

VS1063a RIFF WAV Header				
File Offset	**Field Name**	**Size**	**Bytes**	**Description**
0	ChunkID	4	"RIFF"	RIFF ident
4	ChunkSize	4	F0 F1 F2 F3	File size - 8
8	Format	4	"WAVE"	WAVE ident
12	SubChunk1ID	4	"fmt "	fmt ident
16	SubChunk1Size	4	0x14 0x0 0x0 0x0	20
20	AudioFormat	2	0x07 0x0	Audio format
22	NumOfChannels	2	0x01 0x00	1 for mono, 2 for stereo
24	SampleRate	4	0x40 0x1f 0x00 0x00	0x1f40 = 8000 Hz
28	ByteRate	4	0x40 0x1f 0x00 0x00	Bytes per second
32	BlockAlign	2	0x01 0x00	1 byte per block
34	BitsPerSample	2	0x08 0x00	8 bits / sample
36	Extra size	2	0x02 0x00	2 extra bytes
38	Samples per block	2	0x01 0x00	1 sample per block
40	SubChunk3ID	4	"data"	Data ident
44	SubChunk3Size	4	D0 D1 D2 D3	Data size (File Size-48)
48	Samples...			data

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 6c 72 1b 00  57 41 56 45 66 6d 74 20  |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..|
```

11.6.5 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.

11.6.6 Encoder Samplerate Considerations

For encoder samplerates to work accurately, it is recommended to load and run the *VS1063a Patches* package. It is 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.

Example encoder samplerates, XTALI = 12.288 MHz			
Requested f_s	**Actual f_s**	**Error**	**Note**
48000 Hz	48000 Hz	0.00 %	
44100 Hz	44100 Hz	0.00 %	All except Ogg Vorbis stereo.
44100 Hz	44201 Hz	+0.23 %	Ogg Vorbis stereo; not recommended for streaming.
32000 Hz	32000 Hz	0.00 %	
24000 Hz	24000 Hz	0.00 %	
22050 Hz	22050 Hz	0.00 %	
16000 Hz	16000 Hz	0.00 %	
12000 Hz	12000 Hz	0.00 %	
11025 Hz	11025 Hz	0.00 %	
8000 Hz	8000 Hz	0.00 %	

11.6.7 Encode Monitoring Volume

In VS1063a writing to the SCI_VOL register during encoding mode will update monitoring volume.

11.6.8 MP3 (format 5) Encoder Specific Considerations (VS1063a Only)

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 9.3.1. 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.

Note: If MP3 encoding is attempted with VS1163a or VS8063a, Ogg Vorbis will be encoded.

11.6.9 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 9.3.2.

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 upto 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 11.10) 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.

11.6.10 Estimated Minimum Encoder/Decoder Delays

This Chapter presents the estimated absolute minimum 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, as well as the playback VS1063a's audio buffer if it is not kept to a minimum by the user.

The following symbols are used:
- f_s = samplerate
- d_m = minimum encoder/decoder delay in milliseconds

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

f_s / Hz	PCM/G.711/ G.722 / ms	IMA / ms		MP3 / ms	Ogg[1] / ms
		normal[2]	codec[3]		
48000	3	14	4	97	124
44100	3	15	4	105	135
32000	3	19	4	143	185
24000	3	25	4	118	125
22050	3	26	4	128	140
16000	3	35	5	175	190
12000	3	46	5	233	250
11025	3	49	5	253	270
8000	3	66	6	347	200

[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 11.6.1, Encoding Control Registers, for details on how to set the "limited frame length" bit.

[2] When IMA ADPCM is decoded in decoder mode. Encoder can be in encoder or codec mode.

[3] When IMA ADPCM is decoded in codec mode. Encoder can be in encoder or codec mode.

11.7 Codec Mode (Full-Duplex)

In codec mode VS1063a encodes and decodes separate signal paths at the same time, acting as a full-duplex device.

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 samplerates 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 $5.0\times$.

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

11.8 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:

Normal Mode	SPI Boot Mode
GPIO0	xCS
GPIO1	CLK
DREQ	MOSI
GPIO2	MISO

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.

11.9 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:

Normal Mode	SPI Boot Mode
GPIO0	high = enable SPI/I2C boot
GPIO4	SDA
GPIO6	SCL

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.

11.10 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 config1;       /*0x1e03 wamf ---C ppss RRRR */
  s_int16 playSpeed;     /*0x1e04 0,1 = normal speed, 2 = twice, etc. */
  u_int16 bitRatePer100; /*0x1e05 average bitrate divided by 100 */
  u_int16 endFillByte;   /*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 eq5Params[10]; /*0x1e12..0x1e1b 5-channel EQ parameters */
  u_int16 eq5Updated;    /*0x1e1c write as non-zero to recalculate filters.*/
  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 */
  u_int32 latestSOF;     /*0x1e25/1e26 latest start of frame V4 */
  u_int32 positionMsec;  /*0x1e27-28 play position if known. V3*/
  s_int16 resync;        /*0x1e29 > 0 for automatic m4a, ADIF, WMA resyncs*/
  /* 42 words */
  union {     /* 22 available -- these are not cleared at software reset! */
    u_int16 generic[22]; /*1e2a*/
    struct {
      s_int16 txUartDiv;          /*1e2a direct set of UART divider*/
      s_int16 txUartByteSpeed;    /*1e2b set UART byte speed (txUartDiv=0)*/
      u_int16 txPauseGpio;        /*1e2c mask: a high level pauses tx*/
      s_int16 aecAdaptMultiplier; /* 2 for default */
      s_int16 reserved[14];
      u_int16 channelMax[2]; /*1e3c,1e3d for record level monitoring*/
      u_int32 serialNumber;  /*1e3e,1e3f for Ogg Vorbis if enabled in WRAMADDR(11)*/
    } encoding;
    struct {
      u_int32 curPacketSize;
      u_int32 packetSize;
    } wma; /* 4*/
    struct {
      u_int16 sceFoundMask; /*1e2a single-channel-el. found since last clr*/
      u_int16 cpeFoundMask; /*1e2b channel-pair-el. found since last clr*/
      u_int16 lfeFoundMask; /*1e2c low-frequency-el. found since last clr*/
      u_int16 playSelect;   /*1e2d 0 = first any, initialized at aac init */
      s_int16 dynCompress;  /*1e2e -8192=1.0, initialized at aac init */
      s_int16 dynBoost;     /*1e2f  8192=1.0, initialized at aac init */
      /* playSelect:   0 = first sce or cpe or lfe
         xxxx0001 first sce     xxxx0010 first cpe
         xxxx0011 first lfe     eeee0101 sce eeee
         eeee0110 cpe eeee      eeee0111 lfe eeee */
      u_int16 sbrAndPsStatus; /*0x1e30 V3 gotSBR/upsampling/gotPS/PSactive*/
      u_int16 sbrPsFlags;     /*0x1e31 V4*/
    } aac; /* 3*/
    struct {
      s_int16 gain; /* 0x1e2a proposed gain offset, default = -12 */
    } vorbis;
  } i;
};
```

11.10.1 Parametric: chipID, version, config1

Parameter	Address	Usage
chipID	0x1e00-01	Fuse-programmed unique ID (cosmetic copy of the fuses)
version	0x1e02	Structure version – 0x0004
config1	0x1e03	Miscellaneous configuration

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 miscellanous 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.

config1	
bits	**Usage**
15	1 = Disable WMA decoding
14	1 = Disable AAC decoding
13	1 = Disable MP3 decoding
12	1 = Disable FLAC decoding
11:9	Reserved, set to 0
8	1 = Disable CRC checking for MP3
7:6	AAC PS configuration
5:4	AAC SBR configuration
3:0	not used in VS1063a

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.

config1(5:4)	Usage
’00’	normal mode, upsample $\leq$24 kHz AAC files
’01’	do not automatically upsample $\leq$24 kHz AAC files, but enable upsampling if SBR is encountered (default)
’10’	never upsample
’11’	disable SBR (also disables PS)

config1(7:6)	Usage
’00’	normal mode, process PS if it is available
’01’	process PS if it is available, but in downsampled mode
’10’	reserved
’11’	disable PS processing

11.10.2 Parametric: Player Configurations

Parameter	Address	Usage
playSpeed	0x1e04	0,1 = normal speed, 2 = double, 3 = three times etc.
bitRatePer100	0x1e05	average bitrate divided by 100
endFillByte	0x1e06	byte to send after file
rateTune	0x1e07:1e08	samplerate finetune in +-1ppm steps
playMode	0x1e09	mono, pause, and extra audio processing selects
sampleCounter	0x1e0a:1e0b	sample counter
sdiFree	0x1e1f	SDI FIFO free space in words
audioFill	0x1e20	Audio buffer fill in stereo samples
latestSOF	0x1e25:1e26	latest start of frame
positionMsec	0x1e27:1e28	File position in milliseconds, if available
resync	0x1e29	Automatic resync selector

`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 1 ppm 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 $(x - 1.0) * 1000000$. For example 5.95% speedup $(1.0595 - 1.0) * 1000000 = 59500$.

`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.

playMode		
bits	**Name**	**Usage**
6	PLAYMODE_SPEEDSHIFTER_ON	Speedshifter enable
5	PLAYMODE_EQ5_ON	EQ5 enable
4	PLAYMODE_PCMMIXER_ON	PCM Mixer enable
3	PLAYMODE_ADMIXER_ON	AD Mixer enable
2	PLAYMODE_VUMETER_ON	VU Meter enable
1	PLAYMODE_PAUSE_ON	Pause enable
0	PLAYMODE_MONO_OUTPUT	Mono output select

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 stucture 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 11.5.1.

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 SM_CANCEL 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_NOMP3          (1<<13)
#define CFG1_NOFLAC         (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_SBROFF         (3<<4) /* no SBR or PS */
#define CFG1_MP3_NOCRC      (1<<8) /* turn off CRC checking*/
#define CFG1_REVERB         (1<<0) /* for MIDI (n/a VS1063) */

#define AAC_SBR_PRESENT 1
#define AAC_UPSAMPLE_ACTIVE  2
#define AAC_PS_PRESENT 4
#define AAC_PS_ACTIVE  8
```

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 halfs 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.

11.10.3 Parametric: VU Meter

Parameter	Address	Usage
playMode	0x1e09	bit 2: VU meter enable
vuMeter	0x1e0c	VU meter result (if VU meter enabled)

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 48 kHz samplerate.

11.10.4 Parametric: AD Mixer

Parameter	Address	Usage
playMode	0x1e09	bit 3: AD Mixer enable
adMixerGain	0x1e0d	AD mixer attenuation in 3dB steps -3. . . -31
adMixerConfig	0x1e0e	AD mixer config

```
#define ADMIXER_RATEMASK    (3<<4)
#define ADMIXER_RATE192     (0<<4) /* 5    MIPS   */
#define ADMIXER_RATE96      (1<<4) /* 2.5  MIPS   */
#define ADMIXER_RATE48      (2<<4) /* 1.25 MIPS   */
#define ADMIXER_RATE24      (3<<4) /* 0.6  MIPS   */
#define ADMIXER_MODEMASK    (3<<6)
#define ADMIXER_MODESTEREO  (0<<6)
#define ADMIXER_MODEMONO    (1<<6)
#define ADMIXER_MODELEFT    (2<<6)
#define ADMIXER_MODERIGHT   (3<<6)
```

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 3 dB 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.

11.10.5 Parametric: PCM Mixer

Parameter	Address	Usage
playMode	0x1e09	bit 4: PCM Mixer enable
pcmMixerRate	0x1e0f	PCM mixer samplerate
pcmMixerFree	0x1e10	PCM mixer FIFO free state
pcmMixerVol	0x1e11	PCM mixer volume 0...191 (-0.5 dB steps)

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 upto 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 0.5dB steps. Note: to prevent sigma-delta modulator overflow, SCI_VOL should be at least 2dB (0x0404), and the sum of SCI_VOL and `pcmMixerVol` attenuations at least 6dB (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 6dB (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.

```
s_int16 samples[32];
s_int16 availSpace;
WriteSciReg(SCI_WRAMADDR, 0x1e10);
availSpace = ReadSciReg(SCI_WRAM);
if (availSpace >= 32) {
  ReadSamples(samples, 32);
  WriteSciRegMultiple(SCI_AICTRL0, samples, 32);
}
```

11.10.6 Parametric: EQ5 5-band Equalizer

Parameter	Address	Usage
playMode	0x1e09	bit 5: EQ5 enable
eq5Params	0x1e12/1b	Frequency/gain pairs
eq5Update	0x1e1c	Indicator that settings have been changed

The 5-band equalizer allows attenuating or enhancing five frequency ranges by upto $\pm$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 can not be active at the same time. Bass and treble controls override EQ5.

eq5Params are as follows:

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

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 eq5LevelX + MAX(SVOL_LEFT, SVOL_RIGHT) $\leq$ 0 dB (see Chapter 10.8.11 on page 51).

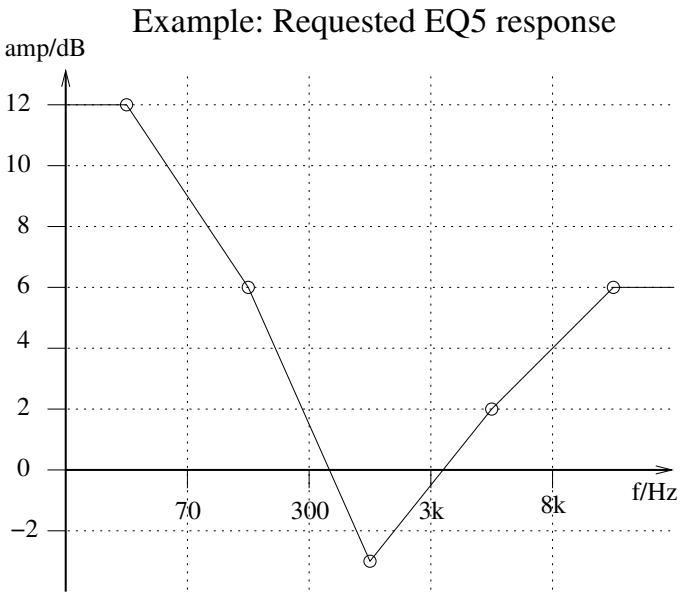


Figure 20: Example EQ5 response request

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 20, and using different volume settings, we get the responses of Figure 21. 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.

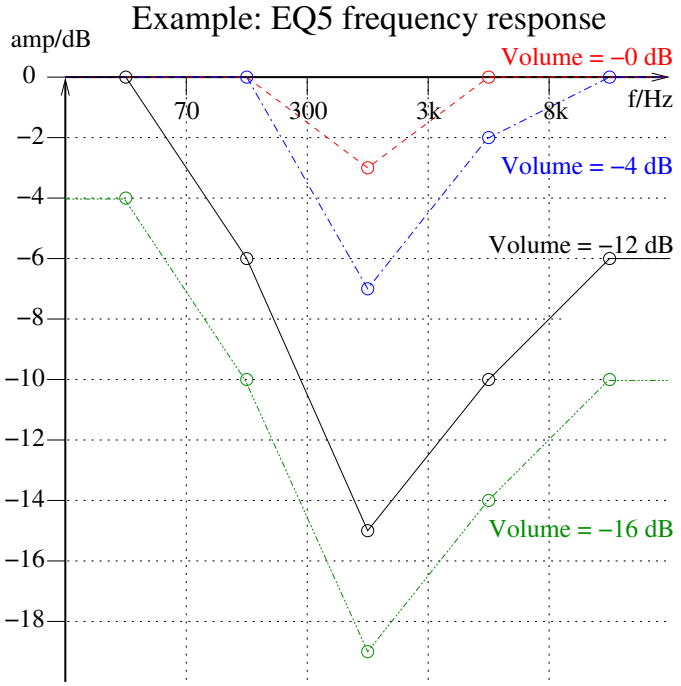


Figure 21: Example EQ5 responses at different volume settings

11.10.7 Parametric: Speed Shifter

Parameter	Address	Usage
playMode	0x1e09	bit 6: SpeedShifter enable
speedShifter	0x1e1d	Speed Shifter speed, 0x4000 = 1.0x

Speed Shifter allows the playback tempo to be changed without changing the playback pitch. The playback tempo is $\frac{speedShifter}{16384}$, 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 $2^{-2/12} = 0.8909$, so set the Speed Shifter to $2^{-2/12} * 16384 = 14596$ and set `rateTune` to $(2^{2/12} - 1) * 1000000 = 122462$.

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

11.10.8 Parametric: EarSpeaker

Parameter	Address	Usage
earSpeakerLevel	0x1e1e	EarSpeaker level, 0 = off

EarSpeaker Spatial Processing is a headphone externalizer algorithm. For information of the algorithm and what it does, see Chapter 10.5, *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 48 kHz samplerate.

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

11.10.9 Parametric: Encoding

These registers are valid when encoding audio.

Parameter	Address	Usage
txUartDiv	0x1e2a	Direct register value to UART_DIV
txUartByteSpeed	0x1e2b	Uart byte speed if txUartDiv = 0
txPauseGpio	0x1e2c	GPIO mask for flow control
aecAdaptMultiplier	0x1e2d	Set to 2 when using AEC
channelMax	0x1e3c/3d	For record level monitoring
serialNumber	0x1e3e/3f	Serial # for Ogg Vorbis here if SCI_WRAMADDR(11) = 1

`txUartDiv` and `txUartByteSpeed` are used to set the UART speed. For low speeds, it is easiert to set the speed through `txUartByteSpeed`, which is the bit speed divided by 10 (e.g. 11520 for 115200 bps). If the speed is high, it is more accurate to use `txUartDiv` which directly controls VS1063a's register UART_DIV (see *VS1063a Hardware Guide* for details). Below are examples for UART values with three different core clock speeds.

Example UART values when encoding XTALI = 12.288 MHz, SCI_CLOCKF = 0xC000 -> CLKI = 55.296 MHz				
txUartDiv	**txUartByteSpeed**	**Nominal/bps**	**Real/bps**	**Error**
0	960	9600	9600	0.00 %
0	11520	115200	115200	0.00 %
0	46080	460800	460800	0.00 %
0	50000	500000	498162	-0.37 %
0x040b	0	1000000	1005382	+0.54 %
0x0025	0	1500000	1494486	-0.37 %
0x0307	0	2000000	1974857	-1.26 %

Example UART values when encoding XTALI = 12.288 MHz, SCI_CLOCKF = 0xB000 -> CLKI = 67.584 MHz				
txUartDiv	**txUartByteSpeed**	**Nominal/bps**	**Real/bps**	**Error**
0	960	9600	9600	0.00 %
0x6106	0	115200	114939	-0.23 %
0x1407	0	460800	459755	-0.23 %
0x0e09	0	500000	500622	+0.12 %
0x0311	0	1000000	993882	-0.61 %
0x0409	0	1500000	1501867	+0.12 %
0x0111	0	2000000	1987765	-0.61 %

Note: UARTs are typically speed error tolerant upto at least ± 2 %.

Note: UART needs 10 bits to transmit one 8-bit byte. So, as an example, the largest bit rate that can be transmitted 115200 bit/s is $115200 \times \frac{8}{10} = 92160$ bit/s.

`txPauseGpio` is a four-bit bitmask (bit 0 for pin GPIO0 through bit 3 for pin GPIO3). If any of the GPIO inputs in the bitmask is high, UART transmission is temporarily paused. Note that pause should not be held up long enough for an encoder buffer overflow to occur.

Example: If `txPauseGpio` is 1, then raising GPIO0 will temporarily pause UART transmission.

If AEC is active, then `aecAdaptMultiplier` is used for configuring it. Unless specifically told to do otherwise, set it to 2.

The highest absolute sample value for the left and right channel can be read from `channelMax[0]` and `channelMax[1]`, respectively. To reset the values, write zeroes to them after reading. Note that these are the maximum values from the input, *not* maximum values after encoding. Lossy encoding like MP3 or Ogg Vorbis may change the largest values in the encoded data.

If you want your Ogg Vorbis recording to have another serial than the default one (highly recommended), write the serial number to `serialNumber` and set SCI_WRAMADDR bit 11 before activating recording.

11.10.10 Parametric: WMA

These registers are valid when decoding WMA audio.

Parameter	Address	Usage
curPacketSize	0x1e2a/2b	The size of the packet being processed
packetSize	0x1e2c/2d	The packet size in ASF header

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 can not be decoded correctly with the current clock. The maximum allowed clock is configured with the SCI_CLOCKF register.

11.10.11 Parametric: AAC

These registers are valid when decoding AAC audio.

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

For an explanation on `config1` AAC bits, see Chapter 11.10.1.

`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), $S * 16 + 5$ - select SCE number S, $P * 16 + 6$ - select CPE number P, $L * 16 + 7$ - 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.

Bit	Usage
0	SBR present
1	upsampling active
2	PS present
3	PS active

`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.

11.10.12 Parametric: Ogg Vorbis

These registers are valid when decoding Ogg Vorbis audio.

Parameter	Address	Usage
gain	0x1e2a	Preferred Replay Gain offset

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 REPLAYGAIN_ALBUM_GAIN tag in the song header, the tag is parsed and the decoded gain setting is written to the `gain` parameter.
If REPLAYGAIN_ALBUM_GAIN is not available, REPLAYGAIN_TRACK_GAIN is used.
If even REPLAYGAIN_TRACK_GAIN is not available, a default of -6 dB (`gain` value -12) is set.

For more information 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.

Gain	Volume	SCI_VOL (Volume-Gain)
-11 (-5.5 dB)	0 (+0.0 dB)	0x0b0b (-5.5 dB)
-11 (-5.5 dB)	3 (-1.5 dB)	0x0e0e (-7.0 dB)
+2 (+1.0 dB)	0 (+0.0 dB)	0x0000 (+0.0 dB)
+2 (+1.0 dB)	1 (-0.5 dB)	0x0000 (+0.0 dB)
+2 (+1.0 dB)	4 (-2.0 dB)	0x0202 (-1.0 dB)

11.11 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.

11.11.1 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:

n bits		
Name	**Bits**	**Description**
F_sIdx	7:5	Samplerate index
S	4:0	Sine skip speed

F_sIdx	F_s	F_sIdx	F_s
0	44100 Hz	4	24000 Hz
1	48000 Hz	5	16000 Hz
2	32000 Hz	6	11025 Hz
3	22050 Hz	7	12000 Hz

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

Example: Sine test is activated with value 126, which is 0b01111110. Breaking n to its components, $F_sIdx = 0b011 = 3$ and thus $F_s = 22050Hz$. $S = 0b11110 = 30$, and thus the final sine frequency $F = 22050Hz \times \frac{30}{128} \approx 5168Hz$.

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.

11.11.2 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.

11.11.3 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, 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.

11.11.4 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. 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:

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

Memory tests overwrite the current contents of the RAM memories.

11.11.5 New Sine and Sweep Tests

A more frequency-accurate sine test can be started and controlled from SCI. SCI_AICTRL0 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:

$$SCI\_AICTRLn = F_{sin} \times 65536/F_s$$

The maximum value for SCI_AICTRLn is 0x8000U. For the best S/N ratio for the generated sine, three LSb's of the SCI_AICTRLn should be zero. The resulting frequencies F_{sin} can be calculated from the DAC samplerate F_s and SCI_AICTRL0 / SCI_AICTRL1 using the following equation.

$$F_{sin} = SCI\_AICTRLn \times F_s/65536$$

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.

11.12 I2S Output

VS1063a can optionally output audio also to 16-bit I2S in addition to the default analog outputs. The pins used for I2S output are I2S_SCLK, I2S_SDATA, I2S_LROUT (called LRCLK in some DAC datasheets), and optionally I2S_MCLK.

To get standard samplerates, XTALI must be either 12.288 MHz or 24.576 MHz. The I2S output is taken after VS1063a's high-quality digital DAC sample rate converter, so its samplerate is independent from the audio samplerate. So, as an example, it is possible to play back MP3 files at 8000 Hz or 44100 Hz and have clean I2S output at 48 kHz. See Chapter *VS1063a Hardware DAC Audio Paths* from *VS1063a Hardware Guide* for details.

Some external I2S digital-to-analog converters require a separate 12.288 MHz clock signal I2S_MCLK.

The table below presents register values for all available I2S samplerates, both with and without I2S_MCLK.

Register values for I2S operation				
GPIO_DDR (addr 0xC017)	**I2S_CONFIG[1] (addr 0xC040)**	**XTALI Divider**	**Samplerate**	**I2S_MCLK**
0x00	0x0	-	I2S off	Off
0xD0	0x4	256[2]	48 kHz[3]	Off
0xD0	0x5	128[2]	96 kHz[3]	Off
0xD0	0x6	64[2]	192 kHz[3]	Off
0x20	0x8	-	I2S off	On
0xF0	0xC	256[2]	48 kHz[3]	On
0xF0	0xD	128[2]	96 kHz[3]	On
0xF0	0xE	64[2]	192 kHz[3]	On

[1] For more detailed information on register I2S_CONFIG, see Chapter *I2S DAC Interface* from *VS1063a Hardware Guide*.
[2] Multiply by 2 if SM_CLK_RANGE is set.
[3] XTALI=12.288 MHz and SM_CLK_RANGE=0, or XTALI=24.576 MHz and SM_CLK_RANGE=1.

Example: If XTALI = 12.288 MHz, and you want I2S output at 48 kHz with the I2S_MCLK signal, do the following:
First write 0xC017 to register SCI_WRAMADDR, then 0x00f0 to SCI_WRAM.
Now write 0xC040 to register SCI_WRAMADDR, and finally 0x000c to SCI_WRAM.

11.13 Clock Speed Requirents

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.

11.13.1 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.

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

11.13.2 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 samplerates are presented. The clock speeds include both the decoder and all audio signal path overheads.

Encoders		
Algorithm	**MIPS**	**Comments**
MP3	61	320 kbit/s, 44.1 kHz stereo
Ogg Vorbis	67[1]	Quality 7 (nominal 183 kbit/s), 32 kHz stereo
Ogg Vorbis	67[1]	Quality 7 (nominal 222 kbit/s), 44.1 kHz stereo
WAV	55	44.1 kHz stereo IMA ADPCM (most CPU intensive WAV format)
G.722	55	16 kHz stereo (standard only requires 16 kHz mono)

[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. 4.0× and clock adder to 1.5×. In this case, if input clock XTALI = 12.288 MHz, then SCI_CLOCKF = 0xB000.

11.13.3 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).

Audio Processing		
Algorithm	**MIPS**	**Comments**
Bass Control	2.5	SCI_BASS bits 7:0
Treble Control	1.3	SCI_BASS bits 15:8
EarSpeaker	11.0	
PCM Mixer	6.5 4.9 6.8 4.0	When PCM audio running at 32 kHz When PCM audio running at 24 kHz When PCM audio running at 16 kHz When PCM audio running at 8 kHz
AD Mixer	5.0 2.5 1.3 0.6	When AD converter is at 192 kHz When AD converter is at 96 kHz When AD converter is at 48 kHz When AD converter is at 24 kHz
VU Meter	0.2	
Mono Output	0.3	

12 VS1063a Version Changes

This chapter describes the lastest and most important changes done to VS1063a

12.1 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 MP3, Ogg Vorbis, μ-law, A-law and G.722 encoding.
- 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$\times$ clock to decode all MP3 bitrates and samplerates.
- 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.
- Bass Enhancer is now full stereo.
- 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.

13 VS1063a Errata

This Chapter describes the firmware and other bugs found in VS1063a, and their status. A current version of this list is available at
http://www.vsdsp-forum.com/phpbb/viewtopic.php?f=10&t=424 .

- 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 11.11, 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.)

  Workaround1: give a hardware reset (or watchdog reset) after each encoding and change of ADMixer rate.
  Workaround2: when you re-activate ADC (DECIM) use a special routine to detect/fix channel swap.
  - 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 192 k).
  - Fixed in vs1063a-patches package v1.2.

- 20111124 FEATURE: codec mode does not support 16kHz/32kHz.
  - Fixed in vs1063a-patches package v1.3
- 20120301 AEC crashes the encoding mode. Reason is still unknown, but routing the AEC calls through additional stub functions seem to prevent the crashes.
  - Workaround: stub functions in vs1063a-patches package v1.31 prevents AEC from crashing
- 20120302 Encoding/codec mode can set the wrong playback rate if AIADDR is read or written (and perhaps on some other case).
  - Fixed in vs1063a-patches package v1.31
- 20120817 ADTS decoding left error variable uncleared. (Can cause no audio to be decoded if a frame has a non-audio element before the first audio element.)
  - Fixed in vs1063a-patches package v1.34
- 20120823 The tentative support for AAC short frames (960/120 samples) has a few problems with syntax and eight_short_frame audio frames. (Note: you need to have a higher XTAL and lie about it to get correct playback speed.)
  - Fixed for ADTS in vs1063a-patches package v1.4, added a short frame selection to parametric_x.config1.
- 20121130 SPI boot does not mask the idle state of MISO correctly for SPI memories with 24/32-bit address. Boot fails if MISO (GPIO2) is pulled up.
  - Workaround: MISO (GPIO2) should have a pull-down resistor for SPI boot to work (like all chips before vs1053b).

14 Latest Document Version Changes

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

Version 1.20, 2016-03-24

- Broke down old Chapter *Other Parameters* to several smaller Parametric chapters.
- Added new Chapter 11.10.9, *Parametric: Encoding*.
- Changed name of the document to better reflect what VS1063a is.
- Other, minor changes.

Version 1.15, 2014-12-19

- Added IMD ADPCM *codec mode* numbers to Chapter 11.6.10, *Minumun Encoder/Decoder Delays*.
- Updated telephone number in Chapter 15, *Contact Information*.

Version 1.14, 2014-11-13

- Added new Chapter 9.1, *Supported Audio Formats Overview*.
- Added mention of RIFF 8-bit and 16-bit data signedness to Chapter 11.5.6, *Feeding PCM Data*.
- Other, minor changes.

Version 1.13, 2014-05-05

- Added VS8063a to front page and Chapter 4, *Product Variants*.
- Estimated MP3 delay times adjusted in Chapter 11.6.10, *Minumum Encoder/Decoder Delays*.
- tV(min) moved to tV(max) in Chapter 8.4.4, *SCI Timing Diagram*.

Version 1.12, 2013-10-09

- Added reference to AEC application note in Chapter 11.6.1, *Encoding Control Registers*.

Version 1.11, 2013-10-01

- Corrected AD Mixer configuration definitions in Chapter 11.10.4, *AD Mixer*.

Version 1.10, 2013-08-14

Added VS1163a, a version of VS1063a without an MP3 encoder, to the datasheet.

- VS1163a added to the front page, as well as Chapter 4, *Product Variants*.
- Clarified how Ogg Vorbis set Replay Gain parameters in Chapter 11.10.12, *Other Parameters / Ogg Vorbis*.

Version 1.02, 2012-12-05

- Added Figure 6, *SDI timing diagram*.
- Minor formatting modifications.

Version 1.01, 2012-11-07

- Added Chapter 11.12, *I2S Output*.

Version 1.00, 2012-10-24

This is a major, production status update to the datasheet.

- Increased maximum clock speed from $5\times$ to $5.5\times$ in Chapter 5.2, *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 10.8.4, SCI_CLOCKF (page 46).
- Added new Chapter 11.13, *Clock Speed Requirements*.
- Added new Chapter 13, *VS1063a Errata.*
- Added Ogg Vorbis encoding clock requirements to Chapter 11.6.9, *Ogg Vorbis (format 6) Encoder Specific Considerations* (page 65).
- Added clock requirement to Chapter 11.7, *Codec Mode*.
- Clarified Chapter 11.7, *Codec Mode*.
- Clarified Chapter 11.6.1, *Encoding Control Registers*.
- Added information to Chapter 11.10.6, *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 11.11, *SDI Tests*. Instead 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 11.6.2, *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 8.3.1, *SDI in VS10xx Native Modes*.
- Removed preliminary status.

15 Contact Information

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

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

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