VS1073a: MP3, Ogg Vorbis, FLAC, WAV ENCODER and AUDIO CODEC

Key Features

- **Encoders:**
  MP3; Ogg Vorbis; FLAC; PCM; IMA ADPCM; G.711 μ-law/A-law; G.722 ADPCM
- **Decoders:**
  AAC-LC / HE-AAC; AC-3 (ATSC52); AIFF/AIFC (PCM, Float, G.711 μ-law / A-law, G.722 ADPCM); ALAC;
  Monkey's Audio (APE); DSD; FLAC;
  MP1, MP2, MP3 (MPEG layers I, II, III);
  Ogg Opus (1-2 channels); Ogg Vorbis;
  WAV (PCM, Float, IMA ADPCM,
  G.711 μ-law/A-law, G.722 ADPCM);
  WMA 4.0/4.1/7/8/9
- **Codecs (Full Duplex):**
  PCM; G.711 μ-law/A-law;
  G.722 ADPCM; IMA ADPCM
- Streaming support - sample rate tune
- Stereo earphone driver capable of driving a 30 Ω load
- 16/32-bit I2S output 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 and alternative encoder output
- New functions may be added with software and up to 12 GPIO pins

Description

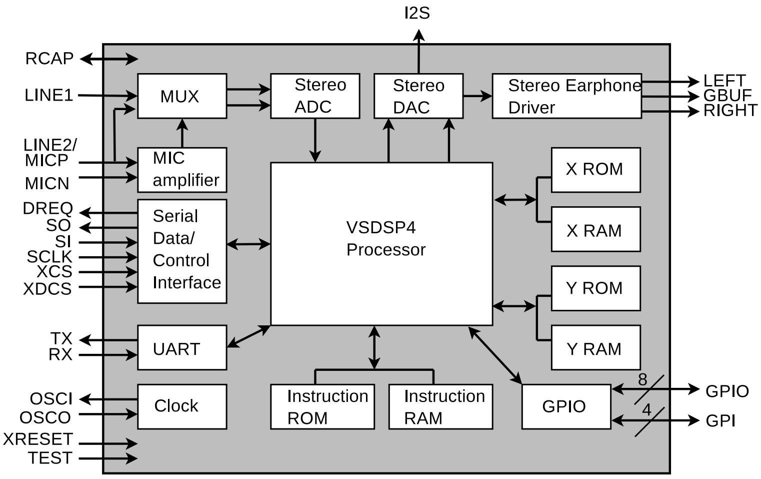
VS1073a is an easy-to-use, versatile encoder, decoder and codec for a multitude of audio formats.

VS1073a contains a high-performance, proprietary low-power DSP core VS_DSP^4, ROM memories, 32 KiB instruction RAM and up to 80 KiB data RAM for user applications, serial control and input data interfaces, up to 12 general purpose I/O pins, a UART, as well as a high-quality stereo ADC, and a variable-sample-rate stereo DAC, followed by an earphone amplifier and a common voltage buffer.

VS1073a can act both as an “Audio decoder IC” or “Audio encoder IC” slave in a system with a microcontroller, or as a stand-alone circuit that boots from an external SPI memory.

Applications

- Audio co-processor for microcontroller
- Recording audio player
- Streaming audio decoder
- Wireless audio transfer
- Audio decoder for DAB and DAB+
- Standalone player and recorder



Additional Features

- EarSpeaker Spatial Processing
- Bass & treble controls
  Alternatively a 5-channel equalizer
- AD Mixer allows monitoring the analog 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, 12.288 MHz required to output standard I2S frequencies
- Internal PLL clock multiplier
- Low-power operation
- High-quality on-chip stereo DAC and ADC with no phase error between channels
- Zero-cross detection for smooth volume change
- Separate voltages for analog, digital, I/O
- Lead-free RoHS-compliant package

Further Description

VS1073a is a pin-compatible alternative for VLSI Solution's VS1053 and VS1063 with a change of core voltage regulator. It has all the functionality of VS1063 (except AEC) and many new features, particularly many new decoders and FLAC encoding.

Operating Modes

VS1073a operates in one of two modes: as a slave co-processor or as a standalone processor.

When used as a slave co-processor VS1073a can operate in three different operation modes: *decoder*, *encoder* or *codec* mode.

In the *decoder mode* VS1073a receives its input bitstream through a serial data interface (SDI). The input stream is decoded and passed through an 18-bit digital volume control to an oversampling sigma-delta DAC. Configuration is performed via a serial control interface (SCI). In addition to the basic decoding and built-in processing functions, it is possible to add application-specific features like DSP effects to the user RAM memory, or to load user applications to replace the ROM functionality.

In the *encoder mode* VS1073a reads audio from its analog inputs, encodes it to the selected format and stores into a FIFO memory. The data in the FIFO is then read by the host processor.

In the *codec mode* VS1073a offers a full-duplex audio interface for simultaneous encoding and decoding for selected WAV formats.

When used as a standalone processor the VS1073a loads its code from an external SPI EEPROM or SPI FLASH. Alternatively code and data can be provided by a host controller through the serial control interface, or in some cases through the UART.

User Code

Users can write their own user interface or signal processing code for the VS1073a using VSIDE (VLSI Solution's Integrated Development Environment).

As a default, there are 32 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 VS1073a.

Contents

List of Figures

1 Disclaimer

All properties and figures are subject to change.

2 Licenses

VS1073a 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.

VS1073a 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.

VS1073a 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: WMA, or AAC, the respective license should not be required. 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 Parametric Stereo (PS) and Spectral Band Replication (SBR) decoding can be separately disabled.

3 Definitions

ABR	Average bitrate. Bitrate of stream may vary locally, but stays close to a given number when averaged over a longer time.
B	Byte, 8 bits.
b	Bit.
CBR	Constant bitrate. Bitrate of the stream is the same for each frame.
CBUF	Headphone Common Buffer. Outputs DC voltage.
GBUF	Same as CBUF.
Ki	“Kibi” = 2^{10} = 1024 (IEC 60027-2).
Mi	“Mebi” = 2^{20} = 1048576 (IEC 60027-2).
SCI	Serial Control Interface, an SPI bus for VS1073a control.
SDI	Serial Data Interface, an SPI bus for VS1073a bitstream data.
VBR	Variable bitrate. Bitrate varies depending on the complexity of the source material.
VS_DSP	VLSI Solution's DSP core.
VSIDE	VLSI Solution's Integrated Development Environment.
W	Word. In VS_DSP, instruction words are 32 bits and data words are 16 bits wide.

4 Characteristics & Specifications

4.1 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Analog Positive Supply	AVDD	-0.3	3.6	V
Digital Positive Supply	CVDD	-0.3	1.35	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		-40	+85	°C
Storage Temperature		-65	+150	°C

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

4.2 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Ambient Operating Temperature		-40		+85	°C
Analog and Digital Ground [1]	AGND DGND		0.0		V
Positive Analog, REF=1.23V [2]	AVDD12	2.6	2.8	3.6	V
Positive Analog, REF=1.65V [2]	AVDD16	3.3	3.3	3.6	V
Positive Digital	CVDD	1.2	1.25	1.35	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	67.6	98.3	MHz
Internal Clock Multiplier [4]	CLKM	$1.0\times$	$5.5\times$	$8.0\times$	
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 9.8.2.
[3] The maximum sample rate that can be natively played by the DAC is XTALI/256 (or XTALI/512 if SM_CLK_RANGE is set). XTALI must be at least 12.288 MHz (24.576 MHz) to be able to play 48 kHz. Audio with higher rates is downsampled automatically.
[4] Reset value is $1.0\times$. Recommended SC_MULT=$5.5\times$ (SCI_CLOCKF=0x8000). Do not exceed maximum specification for CLKI.

4.3 Analog Characteristics

Unless otherwise noted: AVDD=3.3V, CVDD=1.25V, IOVDD=2.8V, VREF=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.

DAC Characteristics					
Parameter	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
DAC Resolution			18		bits
Total Harmonic Distortion, -3 dB of full-scale	THD			0.04	%
Third Harmonic Distortion, -3 dB of full-scale				0.01	%
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, VREF=1.65V	LEVEL16		2750[1]		mVpp
Full Scale Output Voltage, VREF=1.2V	LEVEL12		2050[1]		mVpp
Deviation from Linear Phase				5	°
Analog Output Load Resistance	AOLR	16	30[2]		Ω
Analog Output Load Capacitance				100	pF
DC level (CBUF, LEFT, RIGHT), VREF=1.65V	VREF16		1.65		V
DC level (CBUF, LEFT, RIGHT), VREF=1.2V	VREF12		1.23		V

[1] double can be achieved with +-to-+ wiring for mono difference sound.

[2] AOLR may be much lower, but below *Typical* distortion performance may be compromised.

ADC Characteristics					
Parameter	**Symbol**	**Min**	**Typ**	**Max**	**Unit**
Microphone input amplifier gain	MGAIN		26		dB
Microphone input amplitude (differential)	MLEV16		64	180[1]	mVpp AC
Microphone input amplitude (diff.), VREF = 1.2 V	MLEV12		48	140[1]	mVpp AC
Microphone Total Harmonic Distortion	MTHD		0.03	0.07	%
Microphone S/N Ratio	MSNR	60	72		dB
Microphone input impedances, per pin	MIMP		45		kΩ
Line input amplitude	LLEV16		2500	2800[1]	mVpp AC
Line input amplitude, VREF = 1.2 V	LLEV12		1900	2100[1]	mVpp AC
Line input Total Harmonic Distortion	LTHD		0.005	0.014	%
Line input S/N Ratio	LSNR	85	90		dB
Line input impedance	LIMP		80		kΩ

[1] Harmonic Distortion increases above typical amplitude.

4.4 Power Consumption

Internal clock multiplier $3.0\times$. TA=+25°C. IOVDD =2.8 V, AVDD = 3.3 V, CVDD = 1.25V̇.

XRESET active				
Parameter	**Min**	**Typ**	**Max**	**Unit**
Current AVDD		0		μA
Current CVDD		100		μA
Current IOVDD		0		μA

Full-scale sine in sine test mode – 1.2V / 1.6V reference				
Parameter	**Min**	**Typ**	**Max**	**Unit**
Current AVDD, no load		9 / 13		mA
Current AVDD, output load 33 Ω + GBUF		32 / 55		mA
Current CVDD		4		mA
Current IOVDD		0.2		mA

110 kbit/s Ogg Vorbis audio playback, full volume – 1.2V/1.6V ref				
Parameter	**Min**	**Typ**	**Max**	**Unit**
Current AVDD, no load		9 / 13		mA
Current AVDD, output load 33 Ω + GBUF		16 / 21		mA
Current CVDD		5		mA
Current IOVDD		0.2		mA

4.5 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 $\frac{CLKI}{4}$.

4.6 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] Doesn't include oscillator wakeup time. DREQ rises when the initialization is complete. Do not send any data or commands before that.

5 Packages and Pin Descriptions

5.1 Packages

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

5.1.1 LQFP-48

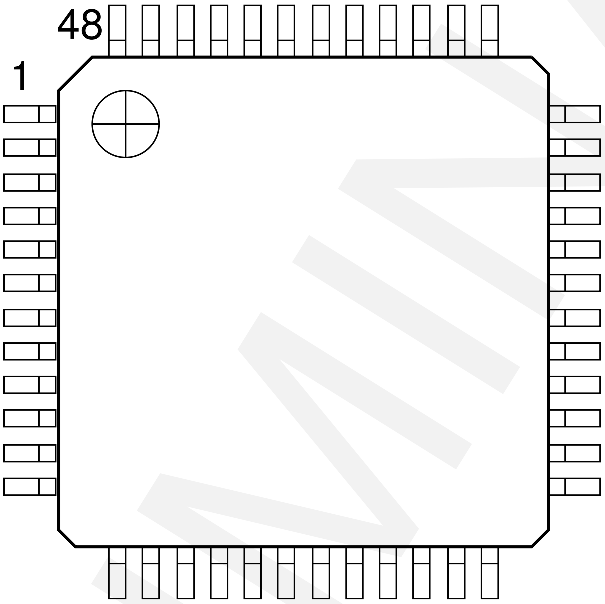


Figure 1: Pin configuration, LQFP-48

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

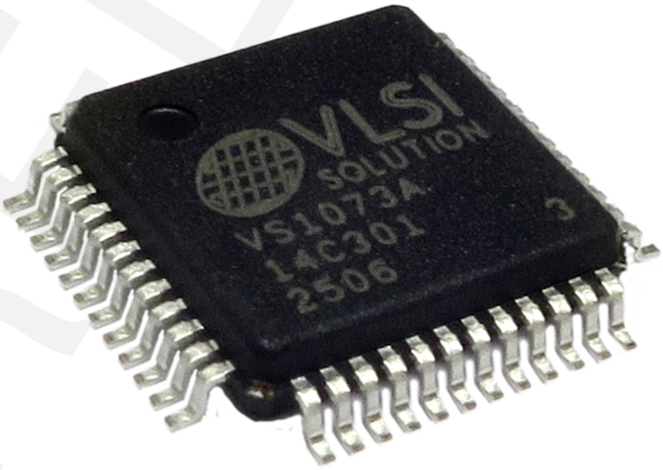


Figure 2: VS1073a in LQFP-48 packaging

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 10.8, SPI Boot, and 10.9, I2C Boot, for details.

[3] If I2S_CF_ENA is '0' the pins are used for GPIO. See Chapter *I2S DAC Interface* from *VS1073a Hardware Guide* or Chapter 10.13, *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

6 Connection Diagram, LQFP-48

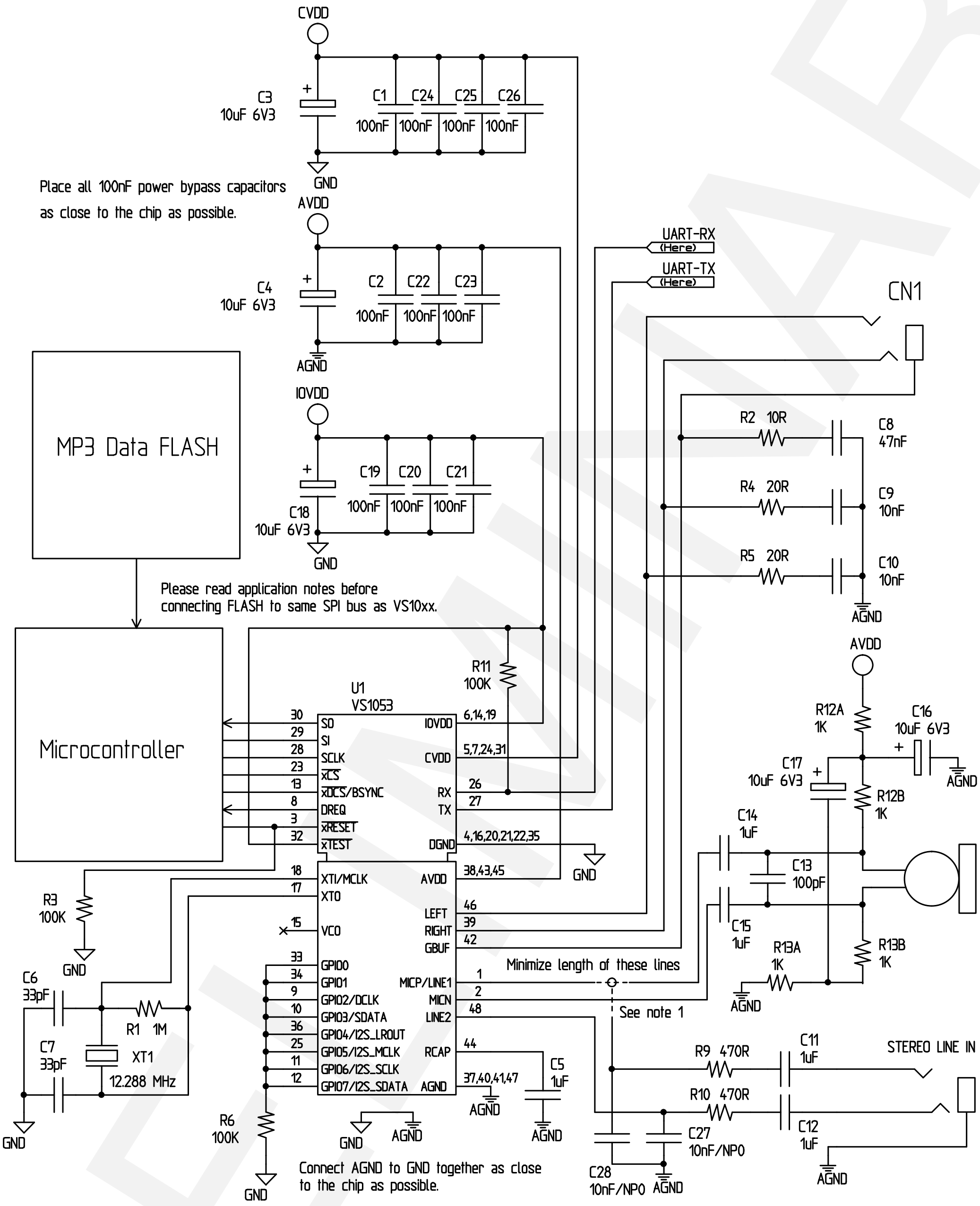


Figure 3: Typical connection diagram using LQFP-48

Figure 3 shows a typical connection diagram for VS1073. Also see the VSDSP Forum thread "Line out, Line in and Headphone connections". *http://www.vsdsp-forum.com/phpbb/viewtopic.php?f=9&t=69* .

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

Note: This connection assumes SM_SDINEW is active (see Chapter 9.8.1). If also SM_SDISHARE is used, xDCS should be tied high (see Chapter 7.1.1).

The common buffer GBUF can be used for common voltage (1.23 V or 1.65 V) for earphones. This eliminates the need for large isolation capacitors on line outputs, and thus the audio output pins from VS1073a 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 can be connected to IOVDD and TX be unconnected.

Do not connect any external load to XTALO.

7 SPI Buses

The SPI bus - originally used in some Motorola devices - is used for both Serial Data Interface SDI (Chapters 7.3 and 9.6) and Serial Control Interface SCI (Chapters 7.4 and 9.7).

7.1 SPI Bus Pin Descriptions

7.1.1 VS10xx Native Modes (New Mode, recommended)

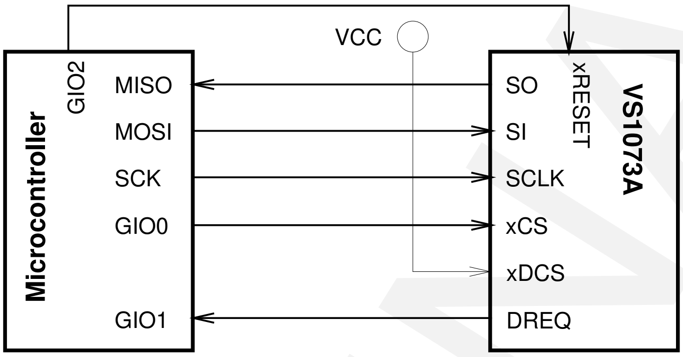


Figure 4: Native mode connection

These modes are active on VS1073a 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. 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. 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 SCK edge.
-	SO	Serial output. In reads, data is shifted out on the falling SCK edge. In writes SO is at a high impedance state.

7.1.2 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. 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.

7.2 Data Request Pin DREQ

The DREQ pin/signal is used to signal if VS1073a's 2048-byte FIFO is capable of receiving data. If DREQ is high, VS1073a 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 up to 32 bytes of SDI data at a time without checking the status of DREQ, making controlling VS1073a 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 should not be aborted.

Note: In VS1073a 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 9.8 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 7.4.

```
// 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) */
```

7.3 Serial Protocol for Serial Data Interface (SDI)

The serial data interface operates in slave mode and is clocked by the pin SCK / DCLK.

The data (SI / SDATA) can be clocked in at either the rising or falling edge of the clock (Chapter 9.8).

VS1073a 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 9.8.1). The default is most significant bit first.

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

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

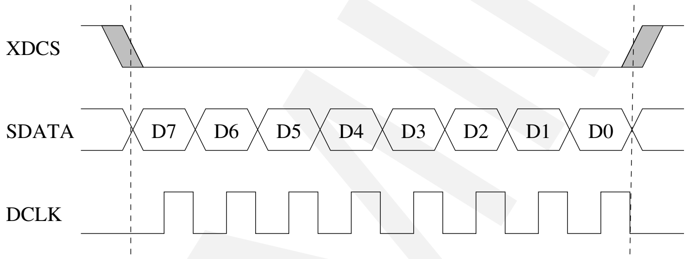


Figure 5: SDI in VS10xx Native Mode, single-byte transfer

In VS10xx native modes (SM_NEWMODE is 1), byte synchronization is achieved by XDCS, as shown in Figure 5. 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 6. 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 7.2).

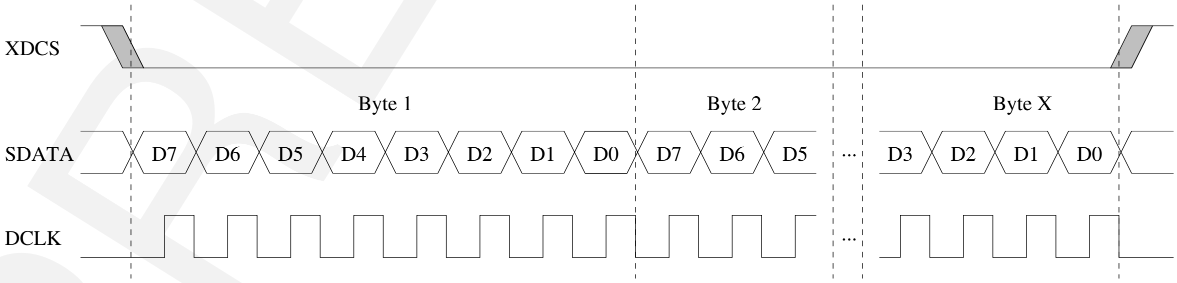


Figure 6: SDI in VS10xx Native Mode, multi-byte transfer, $X \geq 1$

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

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

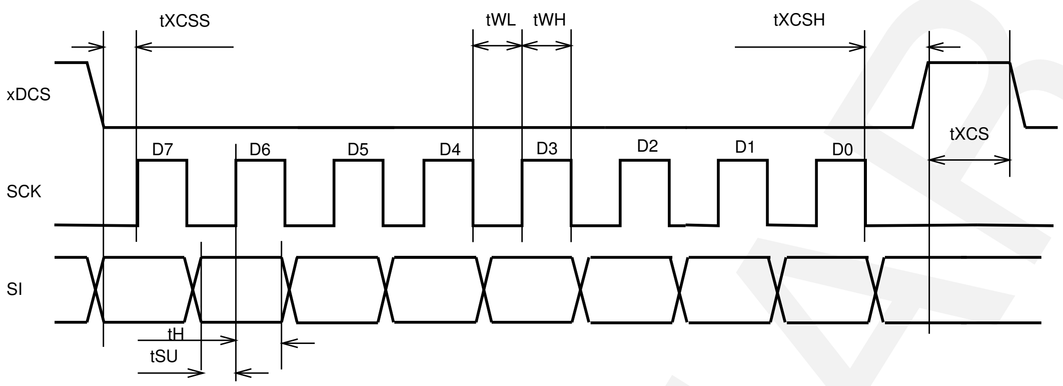


Figure 7: SDI timing diagram

Figure 7 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.

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

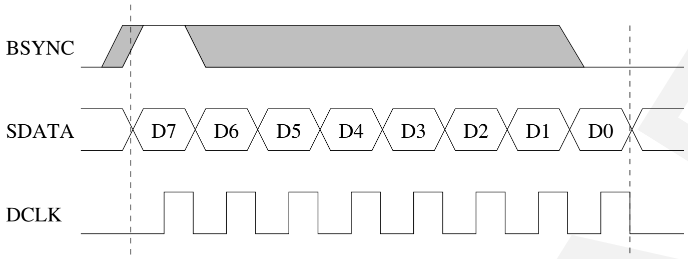


Figure 8: SDI in VS1001 Mode - one byte transfer. Do not use in new designs!

When VS1073a 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 8 and 9.

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.

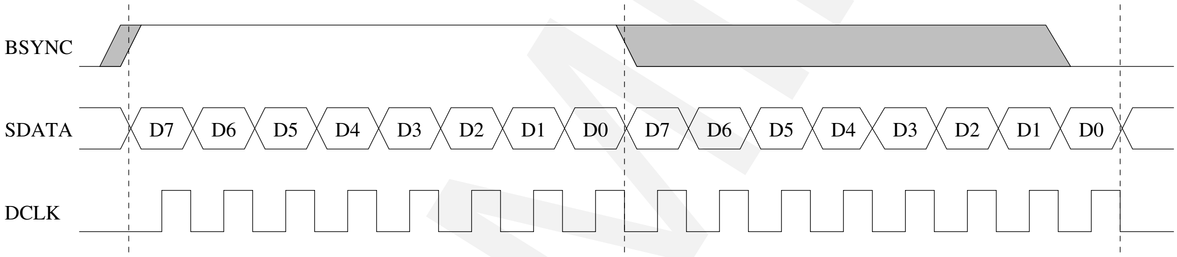


Figure 9: SDI in VS1001 Mode - two byte transfer. Do not use in new designs!

7.3.4 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.

7.4 Serial Protocol for Serial Command Interface (SCI)

The serial bus protocol for the Serial Command Interface (SCI) (see Chapter 9.7) consists of a command 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 and the master should update data at the falling edge. Bytes are always send most significant bit 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 command. The supported operations are read and write. See the table below.

Instruction		
Name	Opcode	Operation
READ	0b0000 0011 = 0x03	Read data
WRITE	0b0000 0010 = 0x02	Write data

Note: VS1073a sets DREQ low after each SCI operation. The duration depends on the operation. It is not allowed to finish a new SCI/SDI operation before DREQ is high again.

7.4.1 SCI Read

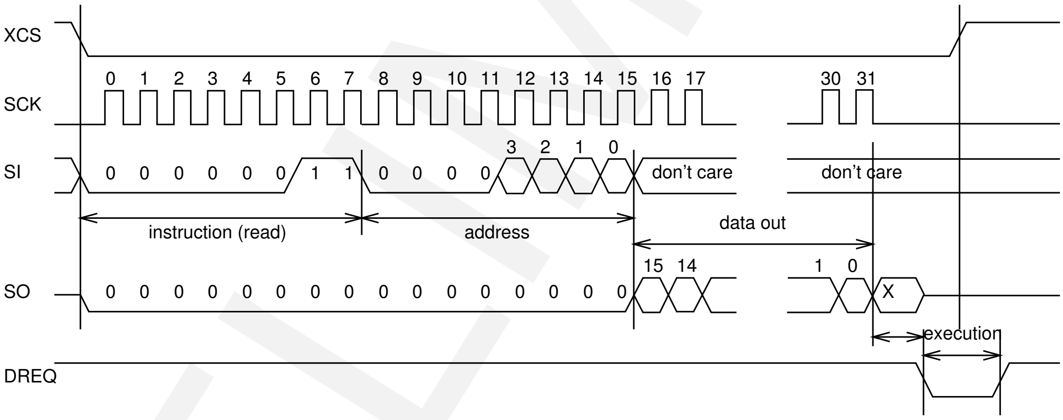


Figure 10: SCI word read

VS1073a registers are read from using the following sequence, as shown in Figure 10. 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 is shifted out onto the SO line.

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

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

7.4.2 SCI Write

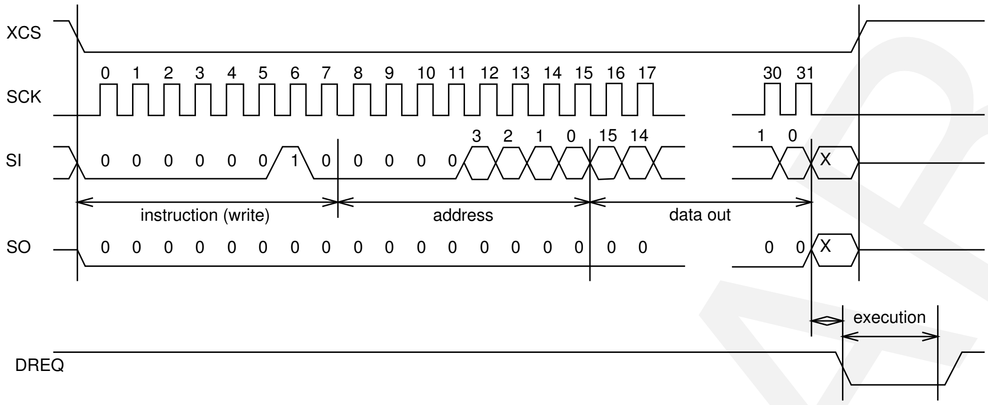


Figure 11: SCI word write

VS1073a registers are written from using the following sequence, as shown in Figure 11. First, XCS line is pulled low to select the device. Then the WRITE opcode (0x2) is transmitted via the SI line followed by an 8-bit word address. After the word has been shifted in and the last clock has been sent, XCS should be pulled high to end the WRITE sequence.

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

7.4.3 SCI Multiple Write

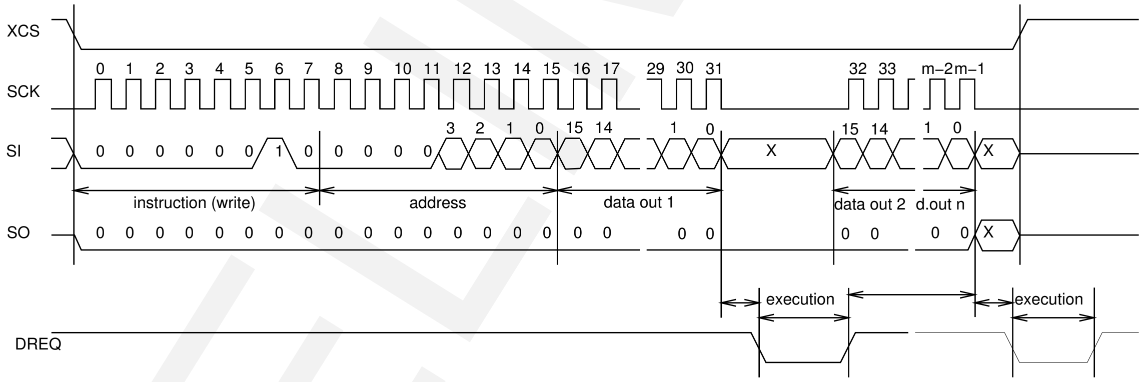


Figure 12: SCI multiple word write

VS1073a allows for the user to send multiple words to the same SCI register, which allows fast SCI uploads, shown in Figure 12. The main difference to a single write is that instead of bringing XCS up after sending the last bit of a data word, the next data word is sent immediately. After the last data word, XCS is driven high as with a single word write.

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

7.4.4 SCI Multiple Read

In VS1073a setting bit 13 (SM_SCIMULTIREAD) of SCI_MODE allows to read multiple words from the same SCI register in the same SCI transaction. Instead of bringing XCS up after reading the last bit of a data word, more data words can be read in succession. After the last data word, drive XCS high like with a single-word SCI read.

Only full 16-bit words should be read in SCI Multiple Read mode. After the **first bit** of each word has been read, DREQ is driven low for the duration of the register update. The time varies depending on the register and its contents (see table in Chapter 9.8 for details). If the maximum time is longer than what it takes from the microcontroller to read the remaining 15 bits of the SCI word, the status of DREQ must be checked before reading the next SCI word.

7.4.5 SCI Timing Diagram

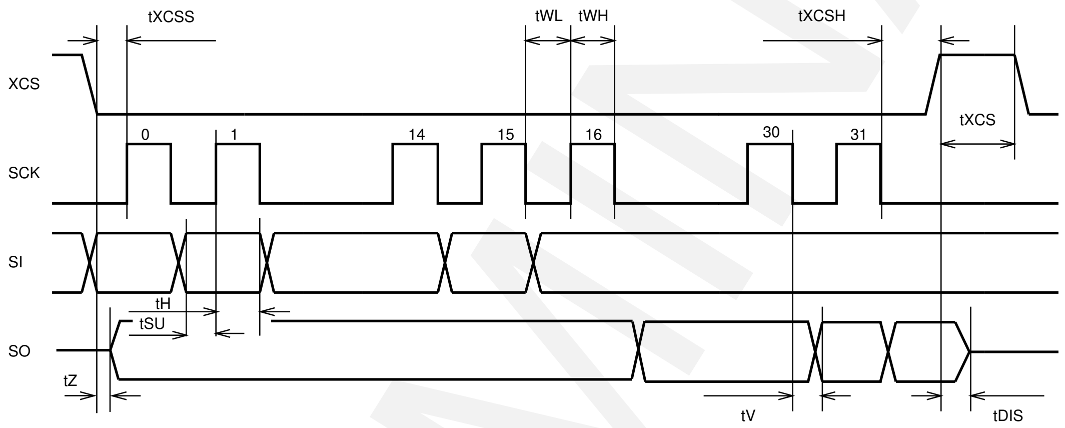


Figure 13: SPI timing diagram

The SCI timing diagram is presented in Figure 13.

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×CLKI + 25 ns, the maximum speed for SCI reads is CLKI/7.

7.5 SPI Examples with SM_SDINEW and SM_SDISHARED set

7.5.1 Two SCI Writes

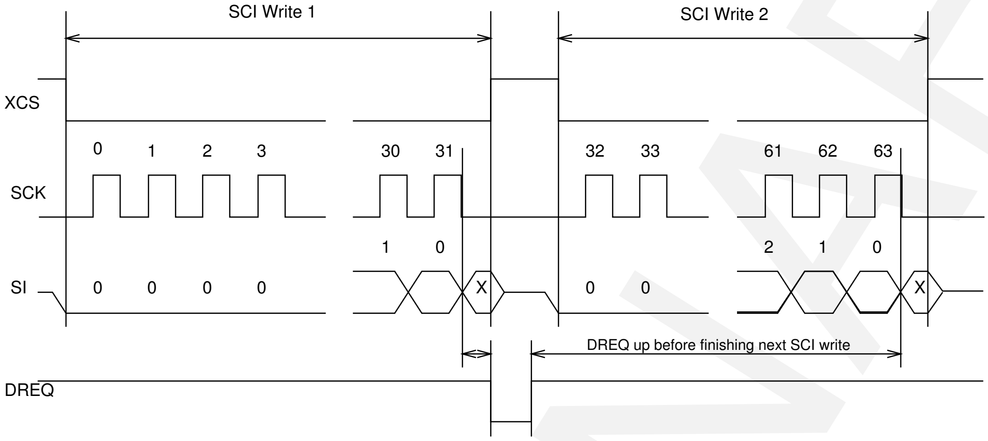


Figure 14: Two SCI operations

Figure 14 shows two consecutive SCI operations. Note that xCS *must* be raised to inactive state between the writes. Also DREQ must be respected as shown in the figure.

7.5.2 Two SDI Bytes

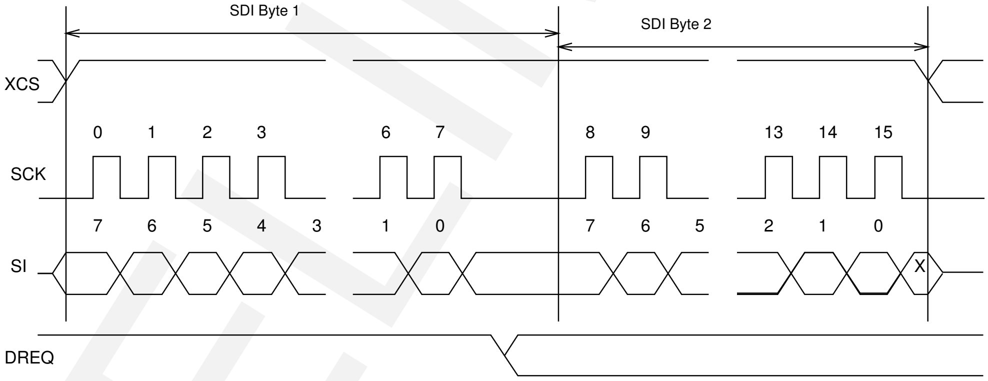


Figure 15: Two SDI bytes

SDI data is synchronized with a raising edge of xCS as shown in Figure 15. However, every byte doesn't need separate synchronization.

7.5.3 SCI Operation in Middle of Two SDI Bytes

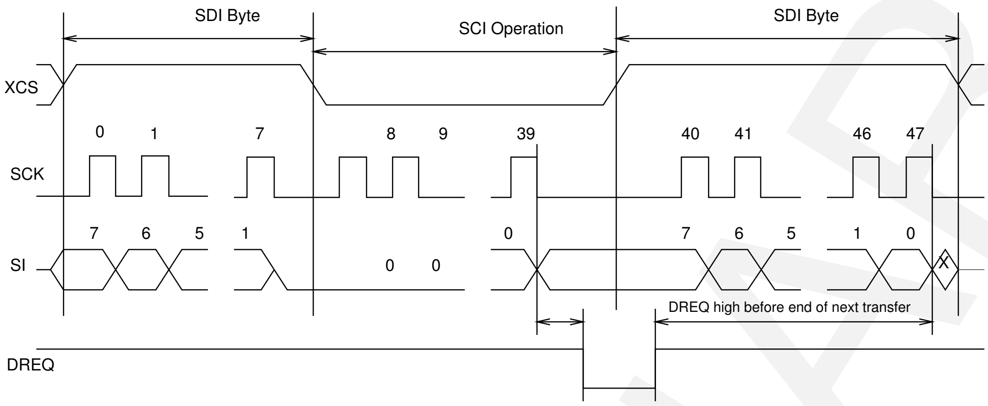


Figure 16: Two SDI bytes separated by an SCI operation

Figure 16 shows how an SCI operation is embedded in between SDI operations. xCS edges are used to synchronize both SDI and SCI. Remember to respect DREQ as shown in the figure.

8 Supported Audio Formats

8.1 Supported Audio Formats Overview

VS1073a supports many audio formats. Some are available as decoders, some as encoders, and some are available as full-duplex codecs.

In the codec mode the audio format of the decoding may be different than the format used for encoding. It is for example possible to decode IMA ADPCM, and at the same time encode in PCM.

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

Audio Format	Decoder	Encoder	Codec
AAC-LC (PNS)	+		
HE-AACv2 (SBR+PS)	+		
AC-3 (ATSC-52)	+		
ALAC	+		
APE - Monkey's Audio	+ [3]		
DSD (.dff .dsf)	+		
FLAC	+	+	
MP1, MP2	+ [1]		
MP3 MPEG1.0, 2.0, 2.5	+	+	
Ogg Opus	+		
Ogg Vorbis	+	+	
WAV/AIFF PCM	+	+ [2]	+ [2]
WAV/AIFF FLOAT	+ [2]		
WAV/AIFC G.711 μ-law, A-law	+ [2]	+ [2]	+ [2]
WAV/AIFC G.722 ADPCM	+	+ [2]	+ [2]
WAV IMA ADPCM	+	+	+
WMA 4.0/4.1/7/8/9	+		

[1] MP1 and MP2 are optional, but they are activated by default.
[2] AIFF/AIFC is only supported in the decoding mode. The encoding mode and the codec mode only support WAV. The codec mode supports FLOAT32 for playback, but not for encoding.
[3] Fast, Normal, and High compression levels are supported for APE. Extra High and Insane compression levels are not supported.

In the decoding and codec modes, playback rates up to $XTALI/256$ (48 kHz with 12.288 MHz XTALI) are played natively, higher rates are automatically downsampled.

8.2 Supported Audio Decoders

Conventions					
Mark	**Description**	**Mark**	**Description**	**Mark**	**Description**
+	Supported	-	Exists, not supported		Format doesn't exist

8.2.1 Supported MP3 (MPEG layer III) Decoder Formats

The VS1073 MP3 decoder is full-accuracy compliant. Also all variable bitrate (VBR) formats are supported.

	Sample rate / Hz	**Bitrate / kbit/s**													
		32	40	48	56	64	80	96	112	128	160	192	224	256	320
MPEG 1.0	48000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	44100	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	32000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

	Sample rate / Hz	**Bitrate / kbit/s**													
		8	16	24	32	40	48	56	64	80	96	112	128	144	160
MPEG 2.0	24000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	22050	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	16000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

	Sample rate / Hz	**Bitrate / kbit/s**													
		8	16	24	32	40	48	56	64	80	96	112	128	144	160
MPEG 2.5	12000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	11025	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	8000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

8.2.2 Supported MP2 (MPEG layer II) Decoder Formats

Note: Layer II decoding can be disabled by clearing the SM_LAYER12 bit from register SCI_MODE.

	Sample rate / Hz	**Bitrate / kbit/s**													
		32	48	56	64	80	96	112	128	160	192	224	256	320	384
MPEG 1.0	48000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	44100	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	32000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

	Sample rate / Hz	**Bitrate / kbit/s**													
		8	16	24	32	40	48	56	64	80	96	112	128	144	160
MPEG 2.0	24000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	22050	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	16000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

8.2.3 Supported MP1 (MPEG layer I) Formats

Note: Layer I decoding can be disabled by clearing the SM_LAYER12 bit from register SCI_MODE.

	Samplerate / Hz	**Bitrate / kbit/s**													
		32	64	96	128	160	192	224	256	288	320	352	384	416	448
MPEG 1.0	48000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	44100	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	32000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

	Samplerate / Hz	**Bitrate / kbit/s**													
		32	48	56	64	80	96	112	128	144	160	176	192	224	256
MPEG 2.0	24000	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	22050	+	+	+	+	+	+	+	+	+	+	+	+	+	+
	16000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

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

VS1073a decodes MPEG2 AAC ADTS streams and MPEG4 files (.mp4), the low complexity and high-efficiency profiles with maximum of two channels. 3GPP (.3gp) and 3GPPv2 (.3g2) files are MPEG4 files, those that contain only HE-AAC or HE-AACv2 audio content are played. AAC is also decoded from the LATM/LOAS container.

- AAC-LC and HE-AACv2
- Mono and stereo, selectable if more channel elements in the stream.
- Sample rates up to 96000 Hz with any bitrate
- MPEG2 AAC in ADTS frames (.aac)
- MPEG4 AAC in a MP4 container (.mp4, .3gp, .3g2)
- Spectral Band Replication (SBR) level 3
- Parametric Stereo (PS) level 3 with R_a and R_b mixing modes, IPD/OPD synthesis and 34 frequency bands resolution, and downsampled synthesis mode
- AAC in LATM/LOAS container
- Dynamic Range Control (DRC)
- Both Sine and Kaiser-Bessel-derived windows
- Pseudo-Random Noise Substitution (PNS)
- Short frames (120 and 960 samples) automatic in MP4, user-indicated in ADTS
- SBR and PS decoding can be disabled or set to different operating modes

HE-AAC v2 files up to 48000 Hz are decoded with 4.5× clock.

Important Note: To be able to play .3gp, .3g2, .mp4 and .m4a files, the **mdat** atom must be the last atom in the MP4 file. VS1073a receives all data as a stream, so all metadata must be available before the audio data is received. Several MP4 file formatters do not satisfy this requirement.

You can find a program that converts the files to a so-called *streamable* format that has the **mdat** atom last in the file. You can use this kind of tool to process files for VS1073a. E.g.:

```
mp4creator -optimize file.mp4
ffmpeg -i input.m4a -movflags faststart optimized.m4a
```

8.2.5 Supported AC-3 (ATSC-52) Decoder Formats

The AC-3 decoder produces a 2-channel surround-encoded down-mix of up to 5.1-channel AC-3 bitstreams. Dual language and dynamic range controls are available.

8.2.6 Supported ALAC Decoder Formats

ALAC can be decoded from a mp4 container or from a CAFF container. Mono and stereo files up to 48 kHz require about 35 MHz (3.0× clock). Mono and stereo files up to 96 kHz are decoded with 74 MHz (6.5× clock).

You should send 12288 endFillBytes instead of just 2050 when ending a file.

8.2.7 Supported Monkey's Audio (APE) Formats

Mono and stereo Monkey's Audio files are decoded. Fast, Normal, and High compression levels are supported. Extra High and Insane compression levels are not supported.

Seek in the file / resync is not supported for APE.

8.2.8 Supported DSD Decoder Formats

DSD64, DSD128, and DSD256 are supported in .dff and .dsf formats, but note that data transfer requirements may make especially DSD256 not practical. See section 8.2.15. DSD64 and DSD128 take about 50 MHz to decode, DSD256 takes 100 MHz.

DXD can be decoded by the WAV decoder, but note the high data transfer requirements. DXD with IEEE32 float takes about 90 MHz to decode.

8.2.9 Supported FLAC Decoder Formats

The FLAC decoder provides very high quality by providing lossless audio decompression. Up to 96 kHz/24-bit or 192 kHz/16-bit FLAC files with up to two channels are decoded with 8.0× clock.

Because of the high data rate, the requirements for data transfer are much higher than for lossy codecs. Because of compression and 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 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) and a larger audio buffer than the other decoders.

Both header CRC-8 and data CRC-16 checking are implemented in this version.

You should send 12288 endFillBytes instead of just 2050 when ending a file.

8.2.10 Supported Ogg Opus Decoder Formats

Only mono and stereo Ogg Opus files are decoded. Downmix from more than 2 channels is not supported.

Ogg Opus uses a subset of features of OPUS, and only this subset is supported. The output is always two channnels and 48 kHz (regardless of the silk and celt core decoder options).

8.2.11 Supported Ogg Vorbis Decoder Formats

Parameter	Min	Max	Unit
Channels	1	2	
Window size	64	4096	samples
Sample rate	100	192000[1]	Hz
Bitrate	0	700	kbit/sec

[1] stereo 48 kHz 192 kbit/sec with 2.0× clock, 96 kHz with 3.5× clock, 192 kHz with 6.0× clock. Higher bitrates may need higher clock.

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

8.2.12 Supported WMA Decoder Formats

Windows Media Audio codec versions WMA 4.0 and 4.1, WMA 7, WMA 8, and WMA 9 are supported. 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.

8.2.13 Supported RIFF WAV Decoder Formats

The following RIFF WAV formats are supported with 1 or 2 audio channels and sample rates up to 384kHz. Note that data transfer requirements (and the CPU requirement for g.722) may make certain combinations unplayable. See section 8.2.15.

Format	Name	Comments
0x01	PCM	32, 24, 16 and 8 bits linear PCM
0x03	IEEE_FLOAT	IEEE 32 / 64 -bit floating point
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	MP3 in WAV container, see Chapter 8.2.1 for formats
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

PCM and IEEE Float formats of up to 11-channels are automatically downmixed to two output channels.

8.2.14 Supported AIFF/AIFC Decoder Formats

The AIFF/AIFC decoder supports 1 and 2 audio channels, sample rates up to 384kHz, with formats: linear PCM 8…32 bits (little or big-endian as appropriate), IEEE32 / IEEE64 float, g.711 A-law, g.711 μ-law, g.722 ADPCM. Note that data transfer requirements (and the CPU requirement for g.722) may make certain combinations unplayable. See section 8.2.15.

8.2.15 Data Transfer Requirements

Some of the newly added audio formats, especially the lossless formats when used at higher sample rates require a lot of bandwidth to transfer at real time.

Provided low clock jitter and good board design, the serial control interface (SDI) of vs1073 can receive data with SPI clock up to CLKI/4. With 8.0× internal clock (CLKI), this produces the maximum receive data rate of 24.576 Mbit/sec. (With 9.0× clock 27.648 Mbit/sec.)

If the controller can read data with high enough sustained speed and send it simultaneously to VS1073, this rate allows the following:

Required Sustained Transfer		
Format	**Sample Rate**	**Data Rate**
DSD256		22.5792 Mbit/sec
DXD (Stereo IEEE32 FLOAT)	352kHz	22.528 Mbit/sec
Stereo 32-bit LPCM	384kHz	24.576 Mbit/sec
Stereo IEEE32 FLOAT	384kHz	24.576 Mbit/sec

8.3 Supported Audio Encoding Formats

8.3.1 Supported MP3 (MPEG layer III) Encoding Formats

VS1073a supports all MP3 sample rates 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 sample rate with same bitrate is recommended.
<	Format is supported but lower bitrate gives the same quality.
-	Format exists but isn't supported.
	Format doesn't exist.

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

Sample rate / 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:

Sample rate / 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:

Sample rate / 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:

Sample rate / 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	++	++	+	+	+	+	<	<	<	<	<	<	<	<

8.3.2 Supported Ogg Vorbis Encoding Formats

The Ogg Vorbis Encoder supports encoding in mono and stereo, with any sample rate from 8 kHz to 48. . . 192 kHz, 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

8.3.3 Supported RIFF WAV Encoding Formats

The following RIFF WAV formats are supported in encoding and codec modes with one or two channels and sample rates up to $XTALI/64$ (192 kHz with 12.288 MHz clock), but note the data rate considerations. The maximum encoding rate of g.722 is limited by CPU. Due to the high data rate generated, see SCI Multiple Read in section 7.4.4, and the UART output mode.

WAV 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 each 8 bits

8.3.4 Supported FLAC Encoding Formats

The FLAC encoder supports encoding one or two channels and sample rates up to $XTALI/64$ (192 kHz with 12.288 MHz clock), but rates above 48 kHz are limited by available CPU. See section 10.6.9 for encoder-specific considerations. Due to the high data rate generated, see SCI Multiple Read in section 7.4.4, and the UART output mode.

9 Functional Description

9.1 Main Features

VS1073a is based on a proprietary digital signal processor, VS_DSP. It contains all the code and data memory needed for decoding its supported audio formats, serial interfaces, a multirate stereo audio DAC and analog output amplifiers and filters.

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

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

The encoded data can be read through either the serial control interface (SCI) or UART (with user-defined speed) in both the encoder mode and the codec mode.

The UART also provides connection to VLSI Solution's Integrated Development Environment VSIDE for development of custom user code if required.

9.2 Decoder Data Flow of VS1073a

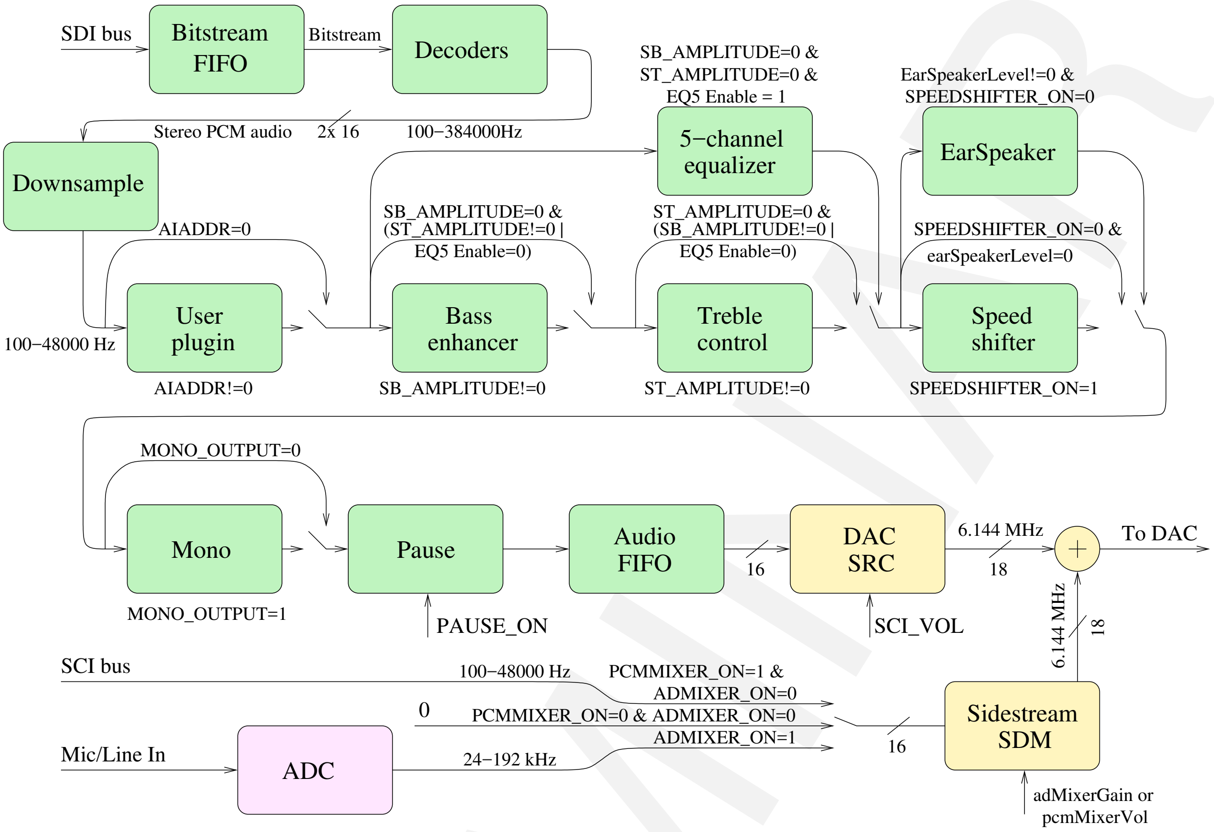


Figure 17: Decoder data flow of VS1073a

Figure 17 presents the decoder dataflow of VS1073a.

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.

If the decoded audio rate is over the maximum supported by the DAC, the audio is downsampled with the appropriate ratio. The choice of ratio takes into account whether the user wants to use the sample rate tuning (the **RESAMPLE_FOR_FINETUNE** bit is set in `playMode`), and the AAC short frame flag. The decoded audio rate is indicated in SCI_AUDATA, not the DAC rate.

Downsampling Examples, XTAL=12.288 MHz		
Audio Rate (Hz)	**Ratio**	**DAC Rate** (Hz)
384000	1:8	48000
96000	1:2	48000
88200	1:2	44100
64000	5:7	45714
Resample for Finetune		
96000	15:32	45000
88200	1:2	44100
48000	15:16	45000
44100	-	44100

Then, if SCI_AIADDR is non-zero, the user plugin code in that address is executed. For more details, see VS1073a Programmer's Guide.

Then data may be processed by the Bass Enhancer and/or Treble Control depending on the contents of the SCI_BASS register. If SCI_BASS is 0 (no Bass Enhance or Treble Control), but the EQ5 Enable bit in the Extra Parameters register `playMode` is 1, the the 5-channel equalizer is used.

Next, if bit `speedShifterEnable` of the 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 the Extra Parameters register `playMode` bit `PLAYMODE_MONO_OUTPUT` is 1, audio is converted to mono.

If the Extra Parameters register `playMode` bit `pause` is 1, audio transmission to audio FIFO is stopped.

After that the data is fed to the Audio FIFO. The default size of the audio FIFO is 2048 stereo (2×16-bit) samples, or 8 KiB, but can be smaller (in the encoding mode) or larger (for AAC and ALAC decoding).

Now the decoded and processed audio is upconverted by a sample rate converter. At the same stage volume control is applied. After this an optional sidestream may be added. This can be either PCM samples sent through the SCI (PCM Mixer) or analog data from the line/mic input (AD Mixer).

The sample rate converter upsamples all different sample rates to XTALI/2 (typically 6.144 MHz) 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 sample rate conversion to a common sample rate removes the need for complex PLL-based clocking schemes and allows a sample rate accuracy of approximately 0.09 Hz with one fixed input clock frequency. Because there is no PLL, the converter is natively jitter free. The output from the converter is 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 and to headphones.

If using an external amplifier or line output connection, an additional analog low-pass filter is required.

9.3 Encoder Data Flow of VS1073a

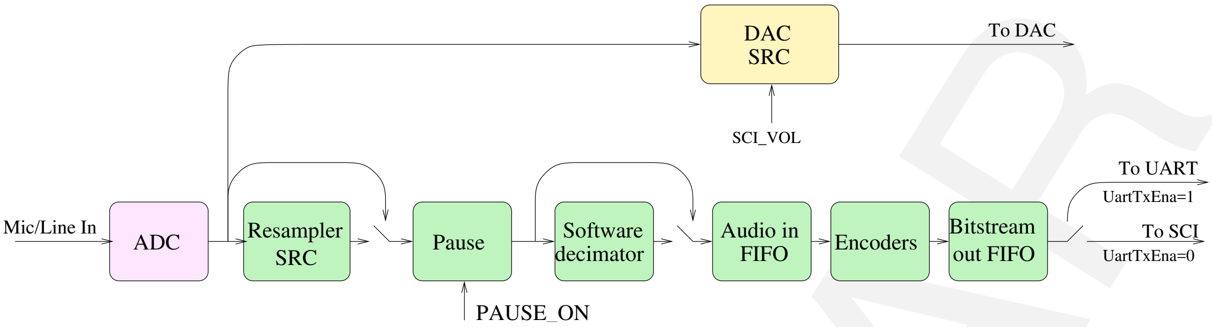


Figure 18: Encoder data flow of VS1073a

Figure 18 presents the encoder dataflow of VS1073a.

Depending on which sample rate the user has requested, data is read from the Analog-to-Digital Converter with one of sample rates of 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.

If resampling is needed, it is performed with the combination of the resampler Sample Rate Converter and software decimators (decimate by 2 or 3). E.g. if the chosen rate is 8 kHz, the analog to digital conversion is performed at 24 kHz, then downsampled by 3 in software.

From the decimator stages, audio is fed to the Audio in FIFO, from which the encoder reads the samples. During Pause mode (when PAUSE_ON bit set in AICTRL3) the samples get discarded instead and this stops the encoding.

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 VS1073a to the UART.

Which sample rates are supported depends on each encoder.

The encoders advance the value of SCI_DECODE_TIME.

9.4 Codec Data Flow of VS1073a

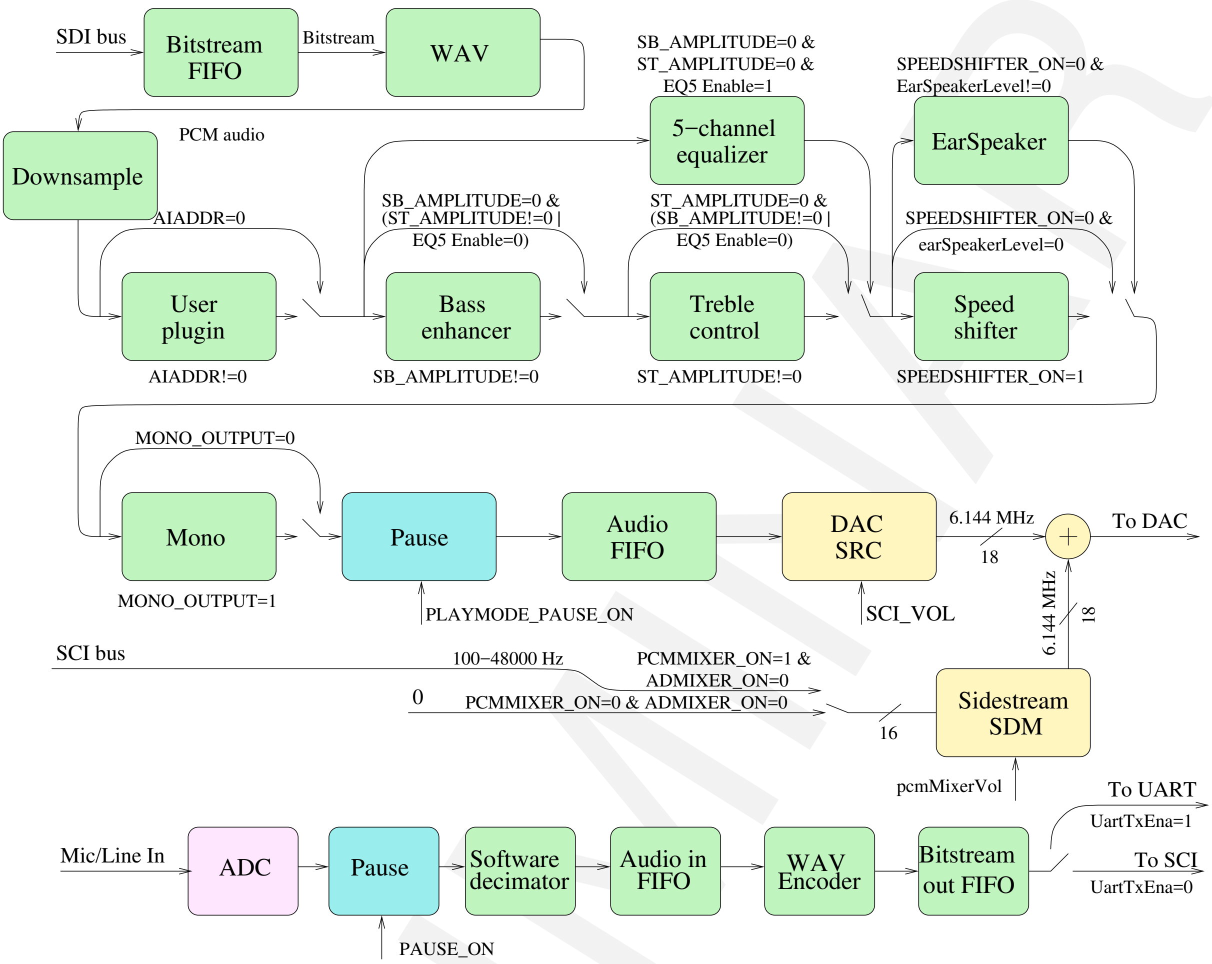


Figure 19: Codec data flow of VS1073a

Figure 19 presents the codec dataflow of VS1073a.

The decoder and encoder paths are almost the same as in the decoder and encoder data flows in Chapters 9.2 and 9.3, except that AD Mixer is not available and the choice of sample rates is more limited in the codec encoder because the Resampler SRC is not available.

Playback rates over 48 kHz are now supported in the codec mode. Also, samplerate fine tuning is supported for the playback. Because playMode is cleared due when encoding mode is started with a software reset, first initialize the encoding mode, then wait until DREQ raises, then write RESAMPLE_FOR_FINETUNE bit in playMode, and only then start sending the file to play.

Note: Encoder pause mode (the PAUSE_ON bit in AICTRL3) stops input to the encoder. The codec decoder can be paused with PLAYMODE_PAUSE_ON bit in parametric_x.playMode, or just by stopping sending data.

See also section 10.7.

9.5 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 20). 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.

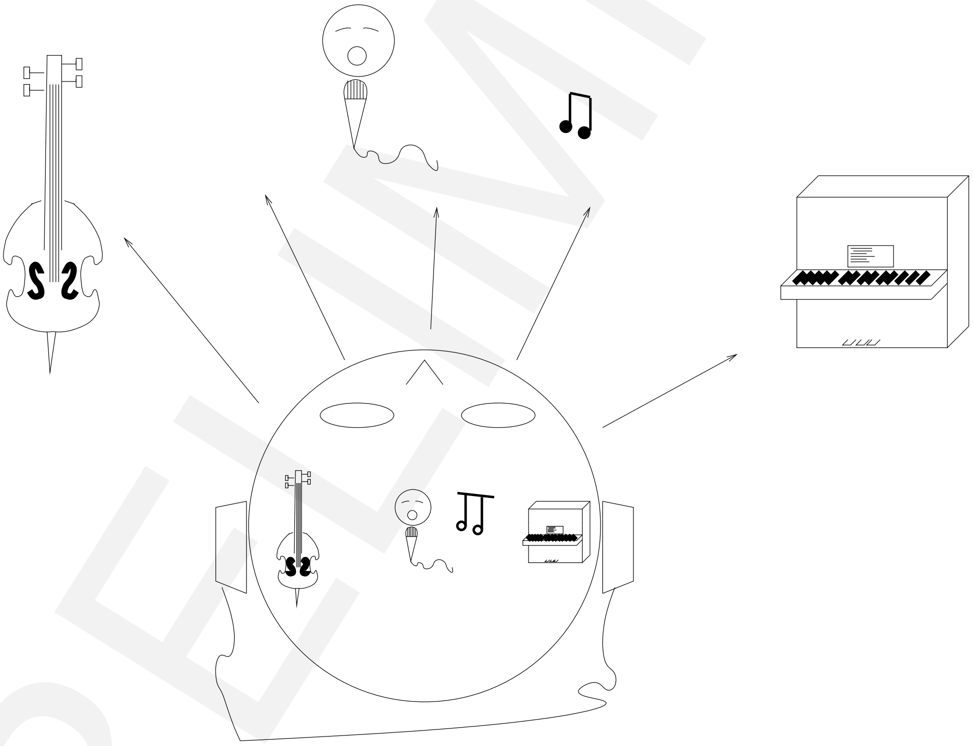


Figure 20: EarSpeaker externalized sound sources vs. normal inside-the-head sound

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 10.10.8.

9.6 Serial Data Interface (SDI)

The serial data interface is meant for transferring compressed data for the different decoders of VS1073a.

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

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

9.7 Serial Control Interface (SCI)

The serial control interface is compatible with the SPI bus specification. Data transfers are always 16 bits. VS1073a 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 the encoding or codec mode
- uploading and controlling user programs

9.8 SCI Registers

VS1073a 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 VS1073a
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	Bitrate in 0.1 kbit/s
0x9	r	0	80 CLKI	SCI_HDAT1	Detected audio format
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 0x88 (analog power enabled). After a short while the user should write 0x80 (analog drivers enabled) to SCI_STATUS.

[4] When write to the mode register causes a software reset the 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 (or UART bytes) while this register update is in progress.

[6] Firmware changes the value of this register immediately to 0x4802.

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 *VS1073a Hardware Guide* for details.

9.8.1 SCI_MODE (RW)

SCI_MODE is used to control the operation of VS1073a and defaults to 0x4802 (SM_LAYER12, SM_SDINEW, and SM_LINE1 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
SM_SCIMULTIREAD	13	HW	SCI Multiple Read	0 1	disabled enabled
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 decode and codec mode players invert the left channel output. For a stereo input this creates virtual surround, and for a mono input this creates a differential left/right signal, both suitable for a differential connection of a mono amplifier.

SM_LAYER12 enables MPEG1.0 layer I and MPEG 1.0 and 2.0 layer II decoding in addition to layer III. The default is enabled.

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 stops decoding and return to the main loop. The stream buffer content is discarded and the SM_CANCEL bit cleared. SCI_HDAT1 is also cleared. See Chapter 10.5.2 for details.

If SM_TESTS is set, SDI tests are allowed. For more details on SDI tests, look at Chapter 10.11.

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 7.1, if also SM_SDINEW is set.

Setting SM_SDINEW activates VS10xx native serial modes as described in Chapters 7.1.1 and 7.3.1. Note, that this bit is set as a default when VS1073a is started up.

By activating SM_ENCODE and SM_RESET at the same time, the user activates the encoding or codec mode (see Chapters 10.6 and 10.7).

SM_SCIMULTIREAD enables the SCI Multiple Read feature, where the same SCI register can be read repeatedly in the same SCI transaction. The default value is disabled.

SM_LINE_IN selects the mode of the left-channel analog input. If '0', the differential microphone amplifier with MICP and MICN inputs is used; if '1', the line-level MICP/LINEIN1 pin is used. The default is line input.

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, so use the divided value for SCI_CLOCKF configuration. SM_CLK_RANGE should be set as soon as possible after a chip reset.

9.8.2 SCI_STATUS (RW)

SCI_STATUS contains information on the current status of VS1073a. 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	SW	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 an audio format header is being decoded and jumping to another location in the file is not allowed.

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 when at full volume and thus should not be used. You can use SS_SWING with I2S to control the amount of headroom, i.e. if you don’t use the analog output, you can set SS_SWING to 2 or more for a higher I2S output level.

VS1073a 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, 7 for VS1103, and 8 for VS1073.

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 power down VS1073a 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. The encoding and codec modes force the divider on when 12000 Hz or lower rates are to be encoded.

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

9.8.3 SCI_BASS (RW)

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 has 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 has 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 sample rate. Both can be on simultaneously.

In VS1073a bass and treble initialization and volume change is delayed until the next batch of samples are sent to the audio FIFO. Thus, unlike with earlier VS10XX chips, audio interrupts can no longer be missed when SCI_BASS or SCI_VOL is written to.

When either the Bass Enhancer or Treble Control is active, the 5-band equalizer (Chapter 10.10.6) is not run.

9.8.4 SCI_CLOCKF (RW)

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

SCI_CLOCKF bits		
Name	**Bits**	**Description**
SC_CLK	15:11	Clock multiplier
SC_FREQ	10: 0	Clock frequency

SC_CLK activates the built-in clock multiplier. This multiplies 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_CLK	CLKI	SC_CLK	CLKI
0x0000	$1.0\times$	0x7800	$5.0\times +1.0\times$
0x1000	$2.0\times$	0x8000	$5.5\times$
0x1800	$2.0\times +1.0\times$	0x8800	$5.5\times +1.0\times$
0x2000	$2.5\times$	0x9000	$6.0\times$
0x2800	$2.5\times +1.0\times$	0x9800	$6.0\times +1.0\times$
0x3000	$3.0\times$	0xa000	$6.5\times$
0x3800	$3.0\times +1.0\times$	0xb000	$7.0\times$
0x4000	$3.5\times$	0xc000	$7.5\times$
0x4800	$3.5\times +1.0\times$	0xd000	$8.0\times$
0x5000	$4.0\times$	0xd800	$9.0\times$
0x5800	$4.0\times +1.0\times$	0xe000	$10.0\times$
0x6000	$4.5\times$	0xe800	$11.0\times$
0x6800	$4.5\times +1.0\times$	0xf000	$12.0\times$
0x7000	$5.0\times$	0xf800	$13.0\times$

Settings with $+1.0\times$ indicate that the decoder is allowed to add to the base multiplier if more cycles are temporarily needed to decode audio. Currently not used by decoders.

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: the maximum sample rate is $\frac{XTALI}{256}$, sample rates above this are played back downsampled.

Example: If SCI_CLOCKF is 0x8BE8, the base PLL multiplier is $5.5\times$, $+1.0\times$ is allowed and SC_FREQ = 0x3E8 = 1000. This means that XTALI = $1000 \times 4000 + 8000000$ = 12 MHz. The clock multiplier is set to $5.5\times$XTALI $= 66$ MHz, and the maximum allowed multiplier that the firmware may automatically choose to use is $(5.5 + 1.0)\times$XTALI $= 78$ MHz.

For how high to set SCI_CLOCKF, see Chapter 10.14, *Clock Speed Requirements*, on page 83.

9.8.5 SCI_DECODE_TIME (RW)

When decoding valid audio data, the current decoded time is shown in this register in full seconds. In the encoding and codec modes SCI_DECODE_TIME indicates the encoding time.

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 `kbitRate` 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 10.10.

9.8.6 SCI_AUDATA (RW)

When decoding correct data, the current sample rate and number of channels can be read in bits 15:1 and 0 of SCI_AUDATA, respectively. Bits 15:1 contain the sample rate divided by two, and bit 0 is 0 for mono data and 1 for stereo. The currently running decoder provides the number of channels indication.

VS1073 supports playing audio content with sample rate over 48000 Hz. If SCI_AUDATA is over 50000, the rate of the original audio is $rate = ((audata\&65534) - 50000) * 100 + 50000$. This gives reading back the rates above 50000 Hz a resolution of 200 Hz and rates below 50000 Hz a resolution of 2 Hz.

Writing to SCI_AUDATA changes the sample rate directly. All 16 bits of the written value are used. To set rates over 50000 Hz, $value = (rate - 50000)/100 + 50000$. This gives setting a rate above 50000 Hz a resolution of 100Hz.

Example: 44100 Hz stereo data reads as 0xAC45 (44101).
Example: 11025 Hz mono data reads as 0x2B10 (11024).
Example: Writing 0xAC80 (44160) sets sample rate to 44160 Hz, stereo indication does not change.
Example: 50461 (0xc51d) read from SCI_AUDATA indicates stereo 96000 Hz audio.

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

9.8.7 SCI_WRAM (RW)

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

9.8.8 SCI_WRAMADDR (W)

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

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.

9.8.9 SCI_HDAT0 and SCI_HDAT1 (R)

SCI_HDAT0 contains the average data rate measured in bits per second divided by 100. To get the bitrate of the file in kilobits per second, divide the value by 10.

If the bitrate is over 6553.5 kbit/s, SCI_HDAT1 value is saturated to 65535.

If HDAT1 is 0xffff, the average bitrate in kbit/s can be read from `parametric_x.kbitRate`.

SCI_HDAT1 indicates the detected format of the file.

HDAT0	HDAT0	Format
0		Nothing detected, no decoder is running
0x6146	"aF"	AIFF/AIFC
0x616C	"al"	MP4 with ALAC audio ("M4" during header decode)
0x614C	"aL"	ALAC in CAFF container
0x4130	"A3"	AC-3 (ATSC-52)
0x4150	"AP"	APE - Monkey's Audio
0x4144	"AD"	AAC in ADIF container
0x4154	"AT"	AAC in ADTS container
0x4453	"DS"	DSD (either .dff or .dsf)
0x664c	"fL"	FLAC
0x4944	"ID"	ID3V2 (shown while skipping the tag)
0x4C41	"LA"	AAC in LATM/LOAS container
0x4D31	"M1"	MPEG1 Layer 1
0x4D32	"M2"	MPEG1 Layer 2
0x4D33	"M3"	MPEG1/2 Layer 3
0x4D34	"M4"	MP4 with AAC audio
0x4F67	"Og"	Ogg Vorbis
0x4F50	"OP"	Ogg Opus ('Og' shown very shortly before detecting Opus)
0x7665	"ve"	RIFF WAV (including .dxd)
0x574D	"WM"	WMA V7. . . 9

9.8.10 SCI_AIADDR (RW)

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

Note: Reading SCI_AIADDR is not recommended.

For more details on how to write user applications and plugins, see VS1073 Programmer's Guide.

9.8.11 SCI_VOL (RW)

SCI_VOL is a volume control for the player hardware. The most significant byte of the volume register controls the left channel volume, the low part controls the right channel volume. The channel volume sets the attenuation from the maximum volume level in 0.5 dB steps. Maximum volume is 0x0000 and total silence but with the output drivers on is 0xFEFE. Setting SCI_VOL to 0xFFFF activates 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 VS1073a bass and treble initialization and volume change is delayed until the next batch of samples are sent to the audio FIFO. This delays the volume setting slightly. The hardware volume control has zero-cross detection, which almost completely removes all audible noise that occurs when volume is changed.

Note: After hardware reset the volume is set to full volume. Resetting the software does not reset the volume setting.

9.8.12 SCI_AICTRL[x] (RW)

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

The SCI_AICTRL registers are also used as parameter registers when encoding audio. See Chapter 10.6 for details.

For more details on how to write user applications, see VS1073 Programmer's Guide.

10 Operation

10.1 Clocking

VS1073a 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 sample rate. With 12.288 MHz clock all sample rates up to 48000 Hz are available.

VS1073a 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.

10.2 Hardware Reset

When the XRESET signal is driven low, VS1073a 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 VS1073a 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 go to high-impedance state (input state), except SO, which is still controlled by XCS.

After a hardware reset (or at power-up) DREQ stays down for around 22000 clock cycles, which means an approximate 1.8 ms delay if VS1073a is run at 12.288 MHz. After this the user should set such basic software registers as SCI_MODE, SCI_BASS, SCI_CLOCKF, and SCI_VOL before starting decoding. See Chapter 9.8 for details.

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

Internal clock can be multiplied with a PLL. Supported multipliers through the SCI_CLOCKF register are $1.0\times \ldots 13.0\times$ the input clock, but observe the CLKI maximum frequency limit. The reset value for the Internal Clock Multiplier is $1.0\times$. Wait until DREQ rises, then write e.g. value 0x8000 to SCI_CLOCKF (register 3). See Chapters 9.8.4 and 10.14 for details on good values for SCI_CLOCKF.

10.3 Software Reset

In some cases the decoder software has to be reset. This is done by activating bit SM_RESET in register SCI_MODE (Chapter 9.8.1). Then wait for at least 2 μs, then look at DREQ. DREQ stays down for about 22000 clock cycles, which means an approximate 1.8 ms delay if VS1073a 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 sample rates or bitrates are played right to their end.

10.4 Low Power Mode

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

- Select the $1.0\times$ 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 reduces the sample rate and the number of audio interrupts required. Between audio interrupts the VSDSP core just waits for an interrupt, thus saving power.
- Turn off all audio post-processing (tone controls, EarSpeaker, etc).
- If possible for the application, write 0xffff to SCI_VOL to disable the analog drivers.

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

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

10.5 Decode Mode

This is the normal operation mode of VS1073a. 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, VS1073a goes into idle mode (lower power consumption than during decoding) and actively monitors the serial data input for valid data.

10.5.1 Playing a Whole File

This is the default playback mode.

1. Send an audio file to VS1073a.
2. Read extra parameter value endFillByte (Chapter 10.10).
3. Send at least 2052 bytes of endFillByte[7:0]. For FLAC and ALAC you should send 12288 endFillBytes when ending a file, and 8192 for DSD.
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.

10.5.2 Cancelling Playback

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

1. Send a portion of an audio file to VS1073a.
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 10.10).
5. Send 2052 bytes of endFillByte[7:0]. For FLAC and ALAC you should send 12288 endFillBytes, and 8192 for DSD.
6. HDAT0 and HDAT1 should now both contain 0 to indicate that no format is being decoded. You can now send the next audio file.

10.5.3 Fast Play

VS1073a allows fast audio playback (except the DSD decoder). If your microcontroller can feed data fast enough to the VS1073a, this is the preferred way to fast forward audio.

1. Start sending an audio file to VS1073a.
2. To set fast play, set extra parameter value `playSpeed` (Chapter 10.10).
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 VS1073a in fast play mode, see contents of extra parameter value `kbitRate` (Chapter 10.10). Note that `kbitRate` contains the data speed of the file played back at nominal speed even when fast play is active.

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

10.5.4 Fast Forward and Rewind without Audio

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

1. Send a portion of an audio file to VS1073a.
2. When random access is required, read SCI_STATUS bit SS_DO_NOT_JUMP. If that bit is set, random access cannot be performed, so go back to 1.
3. Read extra parameter value endFillByte (Chapter 10.10).
4. Send at least 2048 bytes of endFillByte[7:0].
5. Jump forwards or backwards in the file.
6. Continue sending the file.

For some decoders (DSD, WAV) you need to make sure you jump an even number of samples or frame sizes. In this case you would not send endFillBytes. The required jump also depends on the container format.

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

Register DECODE_TIME does not take jumps into account. You need to manage it yourself.

10.5.5 Maintaining Correct Decode Time

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

1. Start sending an audio file to VS1073a.
2. Read extra parameter value pair positionMsec (Chapter 10.10).
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.

10.5.6 Feeding PCM Data

VS1073a 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, VS1073a stays in PCM mode indefinitely (or until SM_CANCEL has been set). Also other formats are supported, but 8-bit (unsigned) linear and 16-bit (signed, 2's complement) linear audio in mono or stereo are the most useful for this mode. 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 = sample rate in Hz, e.g. 44100 for 44.1 kHz.
- For 8-bit data $B = 8$, and for 16-bit data $B = 16$.
- For mono data $C = 1$, for stereo data $C = 2$.
- $A = \frac{C \times B}{8}$.
- $R = S \times A$.

Note: When playing back PCM, VS1073a ignores ByteRate R and BlockAlign A. You may set them to any value if you don't intend the stream 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....|
```

10.6 Encode Mode

This chapter explains how to use the encoding and codec modes of VS1073a.

VS1073 has a stereo ADC, so both mono and stereo can be encoded. Automatic Gain Control (AGC) can be enabled in both mono and stereo (common AGC) or two-channel (separate AGC) encoding modes.

Mono encoding can select either of the channels, or a mono down-mix of the channels. Stereo/two-channel encoding can use any combination of the ADC channels. MIC or LINE1 depending on the SCI_MODE register, the other is LINE2.

10.6.1 Encoding Control Registers

Register	Bits	Description
SCI_MODE	2, 12, 14	Start ENCODE mode, select MIC/LINE1
SCI_AICTRL0	15:0	Sample Rate 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	codec mode (both encode and decode)
	14	reserved, set to 0
	13	UART (8N1) TX enable
	12	reserved, set to 0
	11	Pause enable
	10	No RIFF WAV header inserted (or expected in codec mode)
	9	Disables exact encoding rate to save some CPU
	8	reserved, set to 0
	7:4	Encoding format 0. . . 7
	3	Reserved, set to 0
	2:0	ADC mode 0. . . 4
SCI_WRAMADDR	15. . . 0	Quality / bitrate selection for Ogg Vorbis and MP3

Activate encoding by setting bits SM_RESET and SM_ENCODE in SCI_MODE.

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

SCI_AICTRL1 controls linear encoding gain. The gain is $\frac{AICTRL1}{1024}$, so 1024 is equal to digital gain 1.0, 2000 is 1.95, 512 is 0.5 and so on. If the user wants to use automatic gain control (AGC), SCI_AICTRL1 should be set to 0. Typical speech applications usually want to use 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, Ogg Vorbis, or FLAC format is specified, the configuration bit is ignored and codec mode is not available. In codec mode the encoded data is provided through HDAT0 and HDAT1, and data to be decoded is expected through the serial data interface (SDI).

If SCI_AICTRL3 bit 13 is set at encode/codec startup, UART transmission of data is initialized with parameters taken from `parametric_x.i.encoding` (see Chapter 10.10.9, *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, 7 = FLAC.

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, 5 = stereo with swapped channels, 6 = left channel stereo, 7 = right channel stereo.

SCI_WRAMADDR selects the quality / bit rate for Ogg Vorbis, MP3, and FLAC 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 is not 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 encoding, `parametric_x.i.encoding.channelMax` contains the maximum absolute value encountered in the corresponding channel since the last clear of the variable. In mono modes only `channelMax[0]` is updated.

10.6.2 The Encoding Procedure

The encoding procedure from start to finish goes as follows:

1. Initialization; Set sample rate and parameters:
   - SCI_CLOCKF for clock: $5.5\times$ (see also encoder-specific considerations).
   - SCI_AICTRL0 for sample rate.
   - SCI_AICTRL1 for gain/AGC.
   - SCI_AICTRL2 for AGC max gain.
   - SCI_AICTRL3 for channel selection, encoding format and options.
   - If used, set UART configuration in `parametric_x`.
   - If used, set Ogg Vorbis serial number in `parametric_x`.
   - SCI_WRAMADDR to set bitrate/quality for FLAC, MP3, and Ogg Vorbis.
   - Start encoding mode by setting SM_ENCODE and SM_RESET in register SCI_MODE.)
2. Recording:
   - Depending on whether you selected SCI or UART data transfers with bit 13 of SCI_AICTRL3, read data through SCI_HDAT0/SCI_HDAT1 as described in Chapter 10.6.3, or receive the encoded data through UART.
3. 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 is cleared by VS1073a.
   - 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 size is odd and bits 7:0 contain the last byte that should be written to the output file. Now write 0 to endFillByte.
   - When all samples have been transmitted, SM_ENCODE bit of SCI_MODE is cleared by VS1073a, and SCI_HDAT1 and SCI_HDAT0 are cleared.
4. Now you can give a software reset to enter player mode or start encoding again.

Example of Encoding initialization:

```
  WriteVS10xxRegister(SCI_AICTRL0, 48000U); // 48 kHz
  WriteVS10xxRegister(SCI_AICTRL1, 1024U);  // Manual gain at 1.0x
  WriteVS10xxRegister(SCI_AICTRL3, 0x60);   // Stereo MP3
  WriteVS10xxRegister(SCI_WRAMADDR, 0xE0C0); // Set bitrate to CBR 192 kbit/s
  WriteVS10xxRegister(SCI_MODE,
                     (ReadVS10xxRegister(SCI_MODE)|SM_RESET|SM_ENCODE |SM_LINE));
// Activate recording
```

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

10.6.3 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 in the order they were received from SPI. 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.

Some encoding format create a very high data rate. To support these, or to reduce the number of SCI Read transactions in general, see SCI Multiple Read in section 7.4.4, or use the UART output mode.

10.6.4 File Headers

VS1073 automatically creates a suitable header for the selected encoding mode. If you have selected MP3, Ogg Vorbis, or FLAC, the headers are in those formats, otherwise you get a RIFF WAV header with the correct sample rate, 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, ..|
```

VS1073a 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 VS1073a cannot know in advance how long the recording will be, it sets 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 to make it a valid RIFF WAV file.

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..|
```

10.6.5 Encoder Sample Rate Considerations

With the default 12.288 MHz XTALI, almost all encoding sample rates can be represented accurately with the native ADC rates or integer multiples. Resampling is performed for other rates or when XTAL is something else.

If you don't need high sample rate accuracy, you can turn off this higher-accuracy resampling by setting bit 9 of SCI_AICTRL3 before starting the encoding. This saves some CPU.

10.6.6 Encode Monitoring Volume

In VS1073a writing to the SCI_VOL register during encoding mode updates the monitoring volume.

10.6.7 MP3 (format 5) Encoder Specific Considerations

The MP3 encoder supports all bitrates and sample rates of the MP3 format, both in mono and stereo. For details of supported and recommended modes, see Chapter 8.3.1. Notice particularly that only the MP3 official sample rates are supported (8000, 11025, 12000, 16000, 22050, 24000, 32000, 44100 and 48000 Hz). If you try to start MP3 encoding with any other sample rate, the encoder silently fails.

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 sample rate.

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 makes frames more self-contained. When using ABR/VBR/Quality encoding, turning bit reservoir off increases the bitrate approximately 4. . . 16 kbit/s. Turning bit reservoir off in CBR mode is strongly discouraged as it has a huge impact in quality and coding efficiency.

10.6.8 Ogg Vorbis (format 6) Encoder Specific Considerations

The Ogg Vorbis encoder supports a wide range of bitrates and all sample rates at 8. . . 192 kHz, in mono and stereo. For some examples of supported modes, see Chapter 8.3.2.

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 sample rate.

When silence is detected, the bitstream width may be reduced by up to 90 %. Because the encoder attempts to make Ogg frames as long as possible (up to 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 instructs the encoder to create a frame always after at least 1024 but not more than 2048 samples have been generated in an Ogg frame.

The Ogg stream default serial number is 0xfecaadab. If the user wants to change it, he should, before activating encoding, write to `parametric_x.i.encoding.serialNumber` (Chapter 10.10) and set bit 11 of register SCI_WRAMADDR.

If you run the Ogg Vorbis encoder in stereo, and with a sample rate of 40 kHz or higher, set the clock multiplier register SCI_CLOCKF to $5.5\times$ (e.g. SCI_CLOCKF = 0x8000). Otherwise SCI_CLOCKF = $4.5\times$ is enough.

10.6.9 FLAC (format 7) Encoder Specific Considerations

The FLAC encoder supports sample rates 8. . . 48 kHz in mono and stereo. The encoder uses a fixed predictor and a block size of 1024.

There are three different compression settings. A higher compression setting uses more CPU, but might reduce the resulting file size. The compression setting is only relevant for stereo encoding.

Quality	Description
0..1	always encode with mid/side mode (fastest).
2..3	choose between mid/side and discrete channels mode.
4..	choose between all available channel encoding modes.

45 MHz, 54 MHz, and 66 MHz for 48 kHz stereo, respectively, 32 MHz for 48 kHz mono

10.6.10 Estimated Minimum Encoder/Decoder Delays

This Chapter presents the estimated absolute minimum encoder/decoder total delays between two VS1073a ICs. In addition to these numbers come all data transfer times from the transmitting to the receiving unit, as well as the playback VS1073a's audio buffer if it is not kept to a minimum by the user.

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

Note! Delays have been calculated for standard MP3 sample rates. Other encoders can also encode non-standard sample rates up to 48 kHz.

f_s / Hz	PCM/G.711/	IMA / ms		MP3 / ms	Ogg[1] / ms	FLAC / ms
	G.722 / ms	normal[2]	codec[3]			
48000	3	14	4	97	124	64 ?
44100	3	15	4	105	135	70 ?
32000	3	19	4	143	185	96 ?
24000	3	25	4	118	125	128 ?
22050	3	26	4	128	140	140 ?
16000	3	35	5	175	190	192 ?
12000	3	46	5	233	250	256 ?
11025	3	49	5	253	270	280 ?
8000	3	66	6	347	200	384 ?

[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 up to several seconds. See Chapter 10.6.1, 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.

10.7 Codec Mode (Full-Duplex)

In codec mode VS1073a 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 encoding sample rate should be specific integer fractions of XTALI.
  With XTALI = 12.288 MHz the supported sample rates are 192000, 96000, 64000, 48000, 32000, 24000, 16000, 12000, 8000, and 4000 Hz. If you choose an unsupported rate, the encoding uses the best available rate.
- Only the WAV format is available. If you choose MP3, Ogg Vorbis, or FLAC encoder, the codec mode switches to the encoding mode and ignores the SDI input.
- The PCM encoding only produces 16-bit samples (in mono or stereo), regardless of the playback bit width or encoding type.

With XTALI=12.288 MHz	
Requested Rate (Hz)	Actual (Hz)
1..4000	4000
4001..6000	6000
8000..9999	8000
10000.. 12000	12000
12001.. 19999	16000
20000.. 24000	24000
24001.. 39999	32000
40000.. 48001	48000
48002.. 79999	64000
80000.. 96004	96000
96005..	192000

A RIFF WAV header is automatically generated in the encoded data, unless disabled by bit 10 of SCI_AICTRL3. Like in the encoding mode, encoded data is returned through SCI_HDAT1 and SCI_HDAT0, or through UART when the UART mode is selected.

The data to be decoded is sent to SDI like in decoding mode. The format, number of channels and sample rate of the playback 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 the same as in encoding.

If you use Codec Mode at 48 kHz 16-bit stereo PCM written through SDI and read through SCI, you have to set the clock multiplier SCI_CLOCKF to at least $5.0\times$. Higher rates require higher clock to support the data transfer.

Note: The RIFF WAV parser used in the codec mode decoder is a simplified one. IMA ADPCM, g.722, and g.711 μ-law and A-law are supported, but only 8, 16, and 32-bit linear PCM and 32-bit float formats are supported. This excludes 24-bit PCM and 64-bit float formats.

10.8 SPI Boot

If GPIO0 is set with a pull-up resistor to 1 at boot time, VS1073a 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 VS1073a 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 VS1073a Programmer's Guide.

10.9 I2C Boot

VS1073 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 VS1073a is <100 kHz with the nominal 12.288 MHz clock. The boot record format is the same as for SPI boot.

10.10 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 information. SCI_WRAMADDR addresses 0xc0c0...0xc0ff are mapped to parametric structure addresses 0x1e00...0x1e3f. Also, when an address between 0xc0c0 to 0xc0ff is written to SCI_WRAMADDR, `sdiFree` and `audioFill` are updated.

```
#define PARAMETRIC_VERSION 0x0005
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 --FC ppss RRRR */
  s_int16 playSpeed;     /*0x1e04 0,1 = normal speed, 2 = twice, etc. */
  u_int16 kbitRate;      /*0x1e05 average bitrate in kbit/sec */
  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;
    struct {
      s_int16 dummy; /* 0x1e2a location shared with vorbis decoding */
    } dsd;
    struct {
      s_int16 dummy; /* 0x1e2a location shared with vorbis decoding */
      s_int16 outChannels; /*1e2b mono (1) / stereo (default:2) output */
      s_int16 dualChMode;  /*1e2c both (default:16), language 1 (17), or language 2 (18)*/
      s_int16 dynRangeMode;/*1e2d normal (32) / medium (default:33) / high (34) */
    } a52; /* ATSC52 (AC-3) */
  } i;
};


#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_FEATUREDROP    (1<<10) /* Enable feature drop for AAC/WMA */
#define CFG1_AACSHORTFRAME   (1<<9) /* specify short-frame for ADTS AAC */
#define CFG1_MP3_NOCRC       (1<<8) /* turn off CRC checking*/
#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_REVERB          (1<<0) /* for MIDI (n/a VS1073) */
#define AAC_SBR_PRESENT 1
#define AAC_UPSAMPLE_ACTIVE  2
#define AAC_PS_PRESENT 4
#define AAC_PS_ACTIVE  8
```

10.10.1 Parametric: chipID, version, config1

Parameter	Address	Usage
chipID	0x1e00-01	Chip version (cosmetic copy)
version	0x1e02	Structure version – 0x0005
config1	0x1e03	Miscellaneous configuration

The chip version is read at startup and copied into the `chipID` field. If not available, the value is 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 VS1073a the structure version is 5.

`config1` sets miscellanous settings. Bits 12 to 15 can be used by the user to easily disable certain decoders. 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	Reserved, set to 0
10	1 = Enable feature drop, 0 = default
9	AAC 1 = short-frame file, 0 = normal
8	1 = Disable CRC checking for MP3
7:6	AAC PS configuration
5:4	AAC SBR configuration
3:0	not used in VS1073a

When bit 10 is set in `config` the decoders that support it are allowed to drop decoding features, if the audio buffer underflows due to clock being too low.

Bit 9 sets the AAC short frame (120 / 960 -sample transform) flags for ADTS-format files.

10.10.2 Parametric: Player Configurations

Parameter	Address	Usage
playSpeed	0x1e04	0,1 = normal speed, 2 = double, 3 = three times etc.
kbitRate	0x1e05	average bitrate in kbit/sec
endFillByte	0x1e06	byte to send after file
rateTune	0x1e07:1e08	sample rate 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 plays 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 also counts faster. All decoders except DSD support playSpeed.

kbitRate contains the average bitrate in kbit/sec. The value is updated once per second and it can be used to calculate an estimate of the remaining playtime. A more accurate value (in 0.1kbit resolution) is available in SCI_HDAT0 for all codecs.

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

rateTune finetunes the sample rate in 1 ppm steps. This is useful in streaming applications where long-term buffer fullness is used to adjust the sample rate 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 immediately stops audio sample output. Samples already in the audio buffer are 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 are the same. Other bits are explained separately.

playMode		
bits	**Name**	**Usage**
7	PLAYMODE_RESAMPLE_FOR_FINETUNE	Leave adjustment range for finetune.
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

If you use rate tuning and have audio that plays at the highest DAC rate (48kHz at 12.288MHz XTALI), set `PLAYMODE_RESAMPLE_FOR_FINETUNE` in the `playMode` register. When the playback rate is at the maximum frequency, a software 15:16 resampler is activated to give some room to play the audio faster.

`sampleCounter` advances for each played sample and is initialized by Ogg Vorbis and WMA 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.

The file size and sample size information of WAV and AIFF/AIFC files are ignored when `resync` is non-zero. The user must use SM_CANCEL or software reset to end decoding.

Notice that reading two-word variables through the SCI_WRAMADDR and SCI_WRAM interface is only partly atomic. In VS1073 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.

10.10.3 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 sample rate.

10.10.4 Parametric: AD Mixer

AD Mixer is not available in the encoding / codec mode.

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 sample rate can be 24 kHz, 48 kHz, 96 kHz, or 192 kHz. The higher the rate, the better the quality, but also the more processing power is required.

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

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

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

10.10.5 Parametric: PCM Mixer

Parameter	Address	Usage
playMode	0x1e09	bit 4: PCM Mixer enable
pcmMixerRate	0x1e0f	PCM mixer sample rate
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 sample rate 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 sample rates 8000 Hz, 12000 Hz, 16000 Hz, 24000 Hz, 32000 Hz, 48000 Hz are exact. You can use other rates as well, but they are not exact (for example 11025 Hz, 22050 Hz and 44100 Hz play 0.23% too fast).

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

`pcmMixerVol` controls volume independently of the normal playback volume. Values from 0 to 182 control PCM volume in 0.5dB 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 or more in both SCI_VOL (0x0c0c) and `pcmMixerVol` (12).

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

Processing is 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. PCM Mixer is available in both decoding and encoding / codec modes.

```
int16_t samples[32];
int16_t availSpace;
WriteSciReg(SCI_WRAMADDR, 0x1e10); /* Check available space. */
availSpace = ReadSciReg(SCI_WRAM);
if (availSpace >= 32) {
  ReadSamples(samples, 32); /* Read samples from storage. */
  WriteSciRegMultiple(SCI_AICTRL0, samples, 32); /* Send to VS10xx. */
}
```

10.10.6 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 up to ±16 dB.

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

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

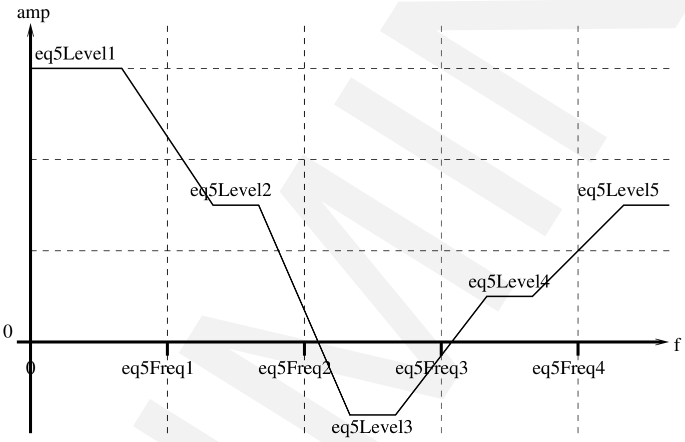


Figure 21: What eq5Params affect

eq5Params (see Figure 21 for what they affect) 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) ≤ 0 dB (see Chapter 9.8.11 on page 49).

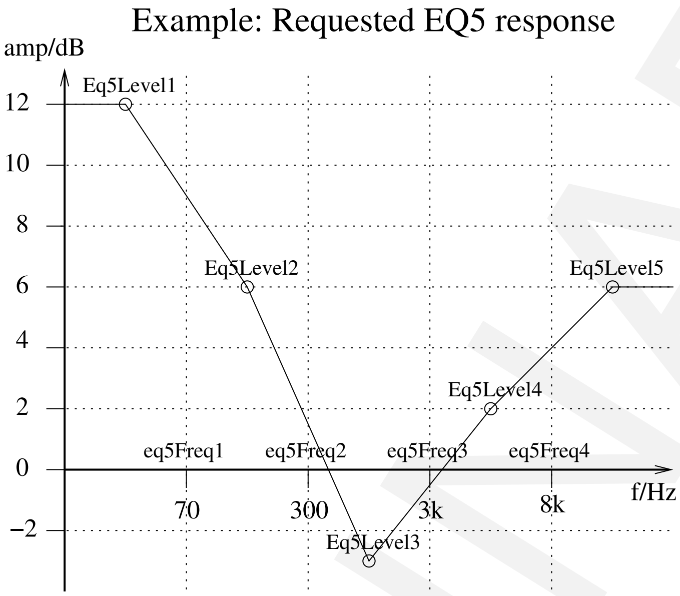


Figure 22: Example EQ5 response request

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 22, and using different volume settings, we get the responses of Figure 23. 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.

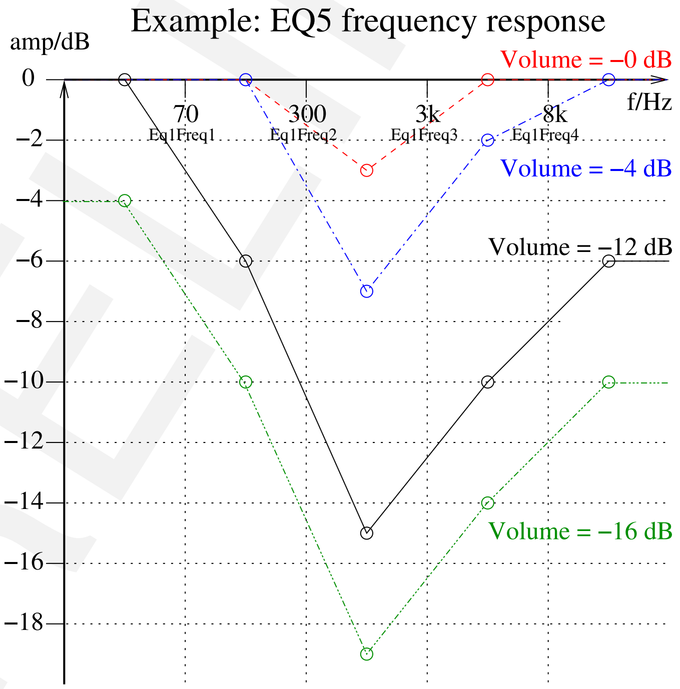


Figure 23: Example EQ5 responses at different volume settings

10.10.7 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 sample rate. 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 $(2^{2/12} - 1) * 1000000 = 122462$.

If you use rate tuning and have audio that plays at the highest DAC rate (48kHz at 12.288MHz XTALI), set `PLAYMODE_RESAMPLE_FOR_FINETUNE` in the `playMode` register before you start playing.

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

10.10.8 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 9.5, *EarSpeaker Spatial Processing*.

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

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

EarSpeaker uses approximately 11 MIPS at 48 kHz sample rate.

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

10.10.9 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
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 VS1073a's register UART_DIV (see *VS1063a Hardware Guide* for details). Below are examples for UART values with two different core clock speeds.

Example UART values when encoding				
XTALI = 12.288 MHz, SCI_CLOCKF = 0x6000 -> 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 = 0x8000 -> 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 up to 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 with 115200 bit/s is $115200 \times \frac{8}{10} = 92160$ bit/s.

`txPauseGpio` is a eight-bit bitmask (bit 0 for pin GPIO0 through bit 7 for pin GPIO7). If any of the GPIO inputs defined in the bitmask is high, UART transmission is temporarily paused. Note that pause should not be active long enough for the encoder buffer to overflow.

Example: If `txPauseGpio` is 1, then raising GPIO0 pauses UART transmission.
Example: If `txPauseGpio` is 48 (0x30), then raising either GPIO4 or GPIO5 pauses UART transmission.

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 number than the default one (highly recommended), write the serial number to `serialNumber` and set SCI_WRAMADDR bit 11 before activating recording.

10.10.10 Parametric: WMA

These registers are valid when decoding WMA audio.

Parameter	Address	Usage
config1	0x1e03	Feature drop

Bit 10 of `config1` enables feature drop. When set, if the WMA audio cannot be decoded correctly with the current clock, and the file contains the LPC feature, the decoder switches to a faster (and less accurate) version of LPC decoding.

10.10.11 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. For more information about Replay Gain, see *http://en.wikipedia.org/wiki/Replay_Gain* and *http://www.replaygain.org/* .

If the Ogg Vorbis decoder finds a REPLAYGAIN_ALBUM_GAIN tag in the song header, the its gain setting is written to the `gain` parameter. If the tag is not found, REPLAYGAIN_TRACK_GAIN is used. If both are unavailable, a default of -6 dB (`gain` value -12) is set.

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

For example `gain` = -11 means that volume should be decreased by 5.5 dB ($-11/2 = -5.5$), i.e. left and right attenuation should be increased by 11. When `gain` = 2 volume should be increased by 1 dB ($2/2 = 1.0$), i.e. left and right attenuation should be decreased by 2. Because the 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)

10.10.12 Parametric: AAC

These registers are valid when decoding AAC audio.

Parameter	Address	Usage
config1	0x1e03	Feature drop, short-frame, 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
sbrPsFlags	0x1e31	SBR and PS mode

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(7:6)	Parametric Stereo 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
config1(5:4)	**Spectral Band Replication Usage**
'00'	normal mode, upsample ≤24 kHz AAC files
'01'	do not automatically upsample ≤24 kHz AAC files, but enable upsampling if SBR is encountered (default)
'10'	never upsample
'11'	disable SBR (also disables PS)

These configuration bits are only read during AAC decoder initialization and then used internally. `parametric_x.i.aac.sbrAndPsStatus` and `parametric_x.i.aac.sbrPsFlags` reflect the current status of the options detected in the bitstream and the currently active features in the decoding.

`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

`sbrPsFlags` 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.

Bit 9 in `config1` specifies the ADTS-format AAC file uses short-frames (960/120 transform sizes) instead of the normal frame length (1024). This information exists in the bitstream itself for mp4 and LATM/LOAS containers, and the indication in the bitstream is used instead of the bit in `config1`.

Bit 10 of `config1` enables feature drop. When set, if the AAC audio cannot be decoded correctly with the current clock, the advanced decoding features are dropped one by one until the audio 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 AAC decoder initialization.

`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.

10.10.13 Parametric: AC-3 (ATSC-52)

The AC-3 decoder produces a 2-channel down-mix of up to 5.1-channel AC-3 bitstreams, taking about 22 MHz to decode. Using `parametric_x.i.ac3.outChannels` it is also possible to decode all 5.1 channels (creates an up-mix of 5.1 channels if the bitstream has fewer channels) and then pick any two channels to be played back without down-mixing.

parametric_x.i.ac3.outChannels	
1	Mono down-mix
2	Stereo down-mix
B*256+A*16+15 (0xBAf)	ch 'A' to left channel, ch 'B' to right channel 0=LFE, 1=Left, 2=Center, 3=Right, 4=Left Surround, 5=Right Surround, 6..15 undefined
0x1000	Stereo down-mix with surround encoding
Examples (channels as-is without down-mix)	
0x31f 0x20f 0x54f	Pick Left → left, Right → right Pick LFE → left, Center → right Pick Left Surround → left, Right Surround → right

For dual-language streams you can choose whether to listen to Language 1 (17), Language 2

(18), or both (16, default) using (`parametric_x.i.ac3.dualChMode`). The default setting is to play both languages.

Dynamic range control is implemented. The setting can be changed between normal (32), medium (33, default), and high (34) using `parametric_x.i.ac3.dynRangeMode`. The default is medium application of dynamic range control.

The `parametric_x.i.ac3` settings are cleared on every AC-3 decoder startup. So, to change the defaults you need to wait until the decoder has started. You could start the decoder by feeding a quiet frame, then adjust the settings and continue with the actual AC-3 stream.

10.11 SDI Tests

There are several test modes in VS1073a, 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: VS1073a is hardware reset, SM_TESTS is set, and then a test command is sent to the SDI bus. Each test is started by sending a 4-byte special command sequence, followed by 12 zeros. The sequences are described below.

10.11.1 Pin Test

Pin test is activated with the 16-byte sequence 0x50 0xED 0x6E 0x54 0 0 0 0 0 0 0 0 0 0 0 0. This test is meant for chip production testing only.

10.11.2 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.

10.11.3 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 2100000 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
14:13		Unused
12	0x1000	RAM persistence ('1')
11	0x0800	Mux test succeeded
10	0x0400	Good MAC RAM
9	0x0200	Good I RAM (8 kW)
8	0x0100	Good Y RAM (24 kW)
7	0x0080	Good X RAM (16 kW)
6	0x0040	Good I ROM 0..3 (56 kW)
5	0x0020	Good I ROM 4 (16 kW)
4	0x0010	Good I ROM 5 (16 kW)
3	0x0008	Good I ROM 6 (16 kW)
2	0x0004	Good Y ROM (40 kW)
1	0x0002	Good X ROM (32 kW)
0	0x0001	Good X ROM 2 (15.75 kW)
	0x9fff	All ok

The memory test can also be started by writing 0x4024 to SCI_AIADDR. The memory test overwrites the current contents of the RAM memories. A software or hardware reset is required afterwards.

10.12 Sine and Sweep Tests

A sine test starts by writing 0x4020 to SCI_AIADDR. SCI_AICTRL0 and SCI_AICTRL1 set the sine frequencies for left and right channel, respectively. These two registers, volume (SCI_VOL), and sample rate (SCI_AUDATA) can be set before or during the test.

SCI_AICTRLn can be calculated from the desired frequency and DAC sample rate 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 sample rate 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.

10.13 I2S Output

VS1073a can optionally output audio also to 16-bit and 32-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. The used pins must be configured as outputs from the GPIO_DDR register.

To get standard sample rates, XTALI must be either 12.288 MHz or 24.576 MHz (the latter can be divided by 2 internally). The I2S output is taken after VS1073a's high-quality digital DAC sample rate converter, so its sample rate is independent from the audio sample rate. So, as an example, it is possible to play back MP3 files at 8000 Hz or 44100 Hz and have I2S output at 48 kHz. See *VS1073a Hardware DAC Audio Paths* from the *VS1073a Hardware Guide* for details.

I2S_CONFIG (addr 0xc040)	
Bits	**Function**
6	0: 16-bit mode, 1: 32-bit mode (not valid with 192 kHz)
5	0: normal, 1: invert LR
4	0: I2S mode (LR change at LSb), 1: DSP mode (LR change at MSb)
3	0: disable MCLK, 1: enable MCLK
2	0: disable I2S, 1: enable I2S
1-0	0: 48 kHz, 1: 96 kHz, 2: 192 kHz, 3: undefined

For a more detailed information on register I2S_CONFIG, see Chapter *I2S DAC Interface* from the *VS1073a Hardware Guide*.

Some external I2S digital-to-analog converters require a separate 12.288 MHz clock signal I2S_MCLK. The table below presents register values for the available I2S sample rates in 16-bit mode, both with and without I2S_MCLK.

Register values for I2S operation				
GPIO_DDR (addr 0xC017)	**I2S_CONFIG MCLK off**	**I2S_CONFIG MCLK on**	**XTALI Divider**	**Sample Rate**
0x00	0x0	0x8	-	I2S off
0xD0	0x4	0xC	256[1]	48 kHz[2]
0xD0	0x5	0xD	128[1]	96 kHz[2]
0xD0	0x6	0xE	64[1]	192 kHz[2]

[1] Multiply by 2 if SM_CLK_RANGE is set.
[2] 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 32-bit 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 0x004c to SCI_WRAM.

10.14 Clock Speed Requirements

When given the highest allowed clock speed VS1073a is capable of decoding or encoding all audio formats with all audio processing. However, to save power it might be useful to run VS1073a at a lower clock speed when all options are not used. This Chapter presents the processing power requirements for the worst case of each VS1073a processing algorithm.

Note: All requirements in this chapter are simulated estimates.

10.14.1 Clock Speed Requirements for Decoders

The table below presents worst-case clock speed requirements for VS1073a decoders. The clock speeds include both the decoder and all audio signal path overheads.

Decoders		
Algorithm	**MIPS**	**Comments**
AAC-LC	30	300 kbit/s VBR, 48 kHz stereo
HE-AACv2	54	39. . . 80 kbit/s CBR, 48 kHz stereo
AC-3	22	448 kbit/s CBR, 48 kHz stereo
ALAC	34	1 Mbit/s 16-bit 44.1 kHz stereo
ALAC	74	2.5 Mbit/s 24-bit 96 kHz stereo
Monkey's Audio (APE)	44 / 54 / 66	Compression Fast / Normal / High 44.1 kHz stereo
DSD	48 / 50	D64 5.7 Mbit/s / D128 11.5 Mbit/s 44.1 kHz stereo
FLAC	20 / 57	1 Mbit/s 16-bit 44.1 kHz / 3 Mbit/s 24-bit 96 kHz stereo
MP1, MP2	18	384 kbit/s CBR, 48 kHz stereo
MP3	30	320 kbit/s CBR, 48 kHz stereo
Ogg Opus	9 / 68	6 kbit/s / 108 kbit/s 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	128 kbit/s, 16 kHz g.722 stereo
WAV	12	389. . . 1536 kbit/s, 48 kHz IMA ADPCM / PCM stereo
WAV	86	23 Mbit/s, 352 kHz, IEEE32 Float stereo (DXD)

10.14.2 Clock Speed Requirements for Encoders

The following gives the approximate worst cases for the encoders, with XTALI = 12.288 MHz. The clock requirements include the encoder and all audio and data transfer overheads.

Encoders		
Algorithm	**MIPS**	**Comments**
Exact Rate	$+18$	exact 44.1 kHz rate (can be disabled)
MP3	$43 + 18 = 61$	320 kbit/s, 44.1 kHz stereo
Ogg Vorbis	67	Quality 7 (nominal 183 kbit/s), 32 kHz stereo
Ogg Vorbis	$48 + 18 = 66$	Quality 7 (nominal 222 kbit/s), 44.1 kHz stereo
WAV	$36 + 18 = 54$	44.1 kHz stereo IMA ADPCM
G.722	$36/52$	16 kHz mono / stereo (standard only requires 16 kHz mono)
FLAC	32	48 kHz mono
FLAC	$46/55/67$	Compression 1 / 2 / 4, 48 kHz stereo

10.14.3 Clock Speed Requirements for DSP Algorithms

The following table shows the required processing power for different algorithm options of VS1073a.

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 sample rate, processing power requirements are proportionally lower (e.g. at 24 kHz the processing power requirement are 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	

11 VS1073a Version Changes

11.1 Hardware Changes Between VS1063a and VS1073

VS1073 is otherwise pin-compatible with VS1053b and VS1063a, but uses a lower CVDD.

Changes:

- Nominal core voltage CVDD=1.25 V
- Serial Control Interface (SCI) supports multiple read mode to double the read speed of encoded data, see section 7.4.4.
- I2S supports 32-bit mode, inverting L/R, and DSP mode. See section 10.13.
- MPEG Layer I and II decoding is enabled by default. (SCI_MODE=0x4802)
- The version field SS_VER in SCI_STATUS is 8 for vs1073.
- The default UART speed (for vs3emu monitor connection) is 115200bps.

11.2 Firmware Changes Between VS1063a and VS1073

Completely new or major changes:

- SCI_CLOCKF clock multipliers redefined, but is fairly compatible with the old system.
- HDAT0 and HDAT1 no longer have special meaning in MPEG layer 1-3 audio. They always indicate the bitrate (saturated to 6553.5 Mbit/s) and the currently active decoder.
- Backwards-compatible change to SCI_AUDATA to support playback rates over 48000 Hz. Higher rates are automatically downsampled for DAC. Decoders can now output higher rates and they play at the correct speed. You can also request adjustment margin for sample rate tuning for streaming applications.
- Added FLAC encoding. But note the high clock and data transfer requirements.
- Support for encoding rates over 48000 Hz for FLAC, Ogg Vorbis, and most WAV subformats. The practical maximum is limited by data rate and processing power requirements.
- Added AC-3 (ATSC-52) decoding.
- Added Monkey's Audio (APE) decoder (Fast, Normal, High compression levels). But note clock and data transfer requirements.
- Added DSD (.dff, .dsf) decoder. But note clock and data transfer requirements.
- Added ALAC decoder (in mp4 and caff containers). 96kHz is the practical maximum rate for ALAC.
- Added AIFF/AIFC decoder. Added more FLOAT formats to AIFF and WAV decoders. WAV decoder now supports DXD. AIFF and WAV decoders create a 2-channel downmix from up to 11 channels for PCM and IEEE Float formats. But note clock and data transfer requirements for high samplerates, large sample sizes, and large number of channels.
- Included MPEG layer I (MP1) decoder.
- Added LATM/LOAS and rudimentary DASH container support for AAC decoding.
- Added ID3V2 parser to skip it correctly.
- Short-frame support for AAC (e.g. DAB+). It is automatic for MPEG4, short-frame can be configured for ADTS from `parametric_x.config1`.

- Added Ogg Opus decoding. Only implements the Opus features that are allowed for Ogg Opus and only supports 1- and 2-channel files.
- AAC uses a slightly larger audio FIFO to reduce the peak CPU usage. HE-AAC thus only requires 4.5× clock.
- AAC and WMA do not perform automatic clock adjustment. Feature drop is now an option you can enable from *parametric_x.config1*, but you can just use high enough clock instead.
- More versatile codec mode instead of AEC support. (AEC is not available in vs1073a.)

Minor changes and bug fixes:

- `parametric_x.positionMsec` conversion to `sampleCounter` for WMA now works correctly.
- `parametric_x.bitRatePer100` changed to `kbitRate` to report the highest bitrates. SCI_HDAT0 still reports the bit rate divided by 100 (saturated to 0xffff).
- Zero samples are no longer automatically inserted when the decoder needs to wait for data. Thus the DAC underflows counter can be used to detect too slow decoding, and we depend on the DAC underflow detection to fade the DC out of the DAC data. The application hook can still request zero samples to be inserted by returning non-zero for APPL_OUT_OF_DATA, or insert them itself.
  As a side effect analog drivers are not automatically enabled at startup. The user can write 0x80 to SCI_STATUS (or 0x81 for higher reference voltage) during power-up instead.
- FLAC decoder performs CRC checking also for data.
- A bugfix for the unused entries in the last encoded band for Ogg Vorbis encoding.
- MPEG Layer II possible overflow in bass intermediate values fixed.
- UART default speed for monitor connection is 115200bps.

12 VS1073a Errata

This Chapter describes bugs/errors of the firmware found in VS1073a.

- Because zero audio is not fed automatically, analog drivers are not enabled until a decoder first produces output samples. Some of the generated audio samples may be lost or the analog output can pick up inteference before the drivers get enabled.
  - Workaround: write 0x80 to SCI_STATUS after reset before sending the first file. (Use 0x81 for the higher reference voltage.)
- Sine test started through SDI does not work. Description removed from datasheet.

13 Latest Document Version Changes

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

Version 0.6, 2025-04-04

- First public version.

14 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