VS1103b Datasheet

VS1103b - MIDI/ADPCM AUDIO CODEC

Features

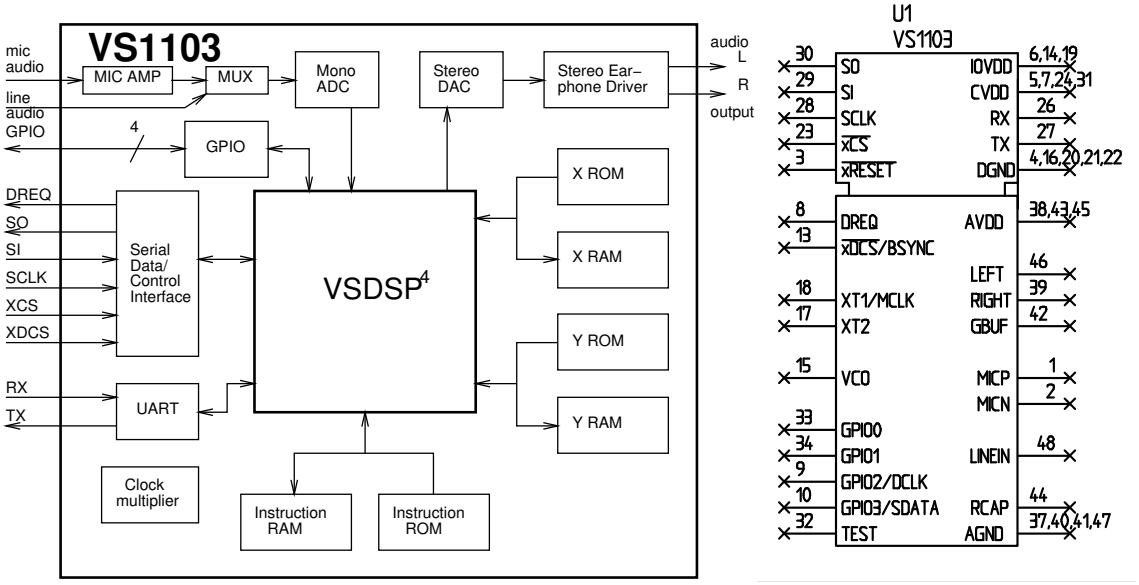
- Mixes three audio sources
  - General MIDI 1+ / SP-MIDI
  - WAV (PCM + IMA ADPCM)
  - Microphone or line input
- Encodes IMA ADPCM from microphone, line input or mixed output
- Input streams can use different sample rates
- EarSpeaker Spatial Processing
- Bass and treble controls
- Operates with a single 12. . . 13 MHz clock
- Internal PLL clock multiplier
- Low-power operation
- High-quality on-chip stereo DAC with no phase error between channels
- Stereo earphone driver capable of driving a 30 Ω load
- Separate operating voltages for analog, digital and I/O
- 5.5 KiB on-chip RAM for user code / data
- Serial control and data interfaces
- Can be used as a slave co-processor
- SPI flash boot for special applications
- UART for debugging purposes
- New functions may be added with software and 4 GPIO pins

Description

VS1103b is a single-chip MIDI/ADPCM/WAV audio decoder and ADPCM encoder that can handle up to three simultaneous audio streams. It can also act as a Midi synthesizer.

VS1103b contains VS_DSP^4, a high-performance, proprietary low-power DSP processor core, working data memory, 5 KiB instruction RAM and 0.5 KiB data RAM for user applications, serial control and input data interfaces, 4 general purpose I/O pins, a UART, as well as a high-quality variable-sample-rate mono ADC and stereo DAC, followed by an earphone amplifier and a common buffer.

VS1103b receives its input bitstreams through serial input buses, which it listens to as a system slave. The input streams are decoded and passed through digital volume controls to an 18-bit oversampling, multi-bit, sigma-delta DAC. Decoding is controlled via a serial control bus. In addition to basic decoding, it is possible to add application specific features, like DSP effects, to user RAM memory.



Contents

List of Figures

1 Disclaimer

All properties and figures are subject to change.

2 Definitions

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

B Byte, 8 bits.

b Bit.

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

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 VS1103 control.

SDI Serial Data Interface, an SPI bus for VS1103 bitstream data.

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

VS_DSP VLSI Solution's DSP core.

VSIDE VLSI Solution's Integrated Development Environment.

W Word. In VS_DSP, instruction words are 32 bits and data words are 16 bits wide.

3 Characteristics & Specifications

3.1 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Analog Positive Supply	AVDD	-0.3	2.85	V
Digital Positive Supply	CVDD	-0.3	2.7	V
I/O Positive Supply	IOVDD	-0.3	3.6	V
Current at Any Digital Output			$\pm$50	mA
Voltage at Any Digital Input		-0.3	IOVDD+0.3[1]	V
Operating Temperature		-40	+85	°C
Storage Temperature		-65	+150	°C

[1] Must not exceed 3.6 V

3.2 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	AVDD	2.6	2.8	2.85	V
Positive Digital	CVDD	2.4	2.5	2.7	V
I/O Voltage	IOVDD	CVDD-0.6V	2.8	3.6	V
Input Clock Frequency[2]	XTALI	12	12.288	13	MHz
Internal Clock Frequency	CLKI	12	36.864	52.0[4]	MHz
Internal Clock Multiplier[3]		$1.0\times$	$3.0\times$	$4.5\times$[4]	
Master Clock Duty Cycle		40	50	60	%

[1] Must be connected together as close the device as possible for latch-up immunity.
[2] The maximum sample rate that can be played with correct speed is XTALI/256.
Thus, XTALI must be at least 12.288 MHz to be able to play 48 kHz at correct speed.
For other implications rising from not using a 12.288 MHz clock, see Chapter 7.4.5.
[3] Reset value is $1.0\times$. Recommended SC_MULT=$4.0\times$.
Performance may be poor if SC_MULT< 3.5.
[4] 52.0 MHz is the maximum clock for the full CVDD range.
(4.0×12.288 MHz=49.152 MHz or 4.0×13.0 MHz=52.0 MHz)

3.3 Analog Characteristics

Unless otherwise noted:
AVDD = 2.6. . . 2.85 V, CVDD = 2.4. . . 2.7 V, IOVDD = CVDD - 0.6 V. . . 3.6 V, TA = -25. . . +70 °C, XTALI = 12. . . 13 MHz, 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 50 mVpp, f_s=1 kHz, Line input test amplitude 1.1 V, f_s=1 kHz.

Parameter	Symbol	Min	Typ	Max	Unit
DAC Resolution			18		bits
Total Harmonic Distortion	THD		0.1	0.3	%
Dynamic Range (DAC unmuted, A-weighted)	IDR		90		dB
S/N Ratio (full scale signal)	SNR	70			dB
Interchannel Isolation (Cross Talk)		50	75		dB
Interchannel Isolation (Cross Talk), with GBUF			40		dB
Interchannel Gain Mismatch		-0.5		0.5	dB
Frequency Response		-0.1		0.1	dB
Full Scale Output Voltage (Peak-to-peak)		1.3	1.5[1]	1.7	Vpp
Deviation from Linear Phase				5	°
Analog Output Load Resistance	AOLR	16	30[2]		Ω
Analog Output Load Capacitance				100	pF
Microphone input amplifier gain	MICG		26		dB
Microphone input amplitude			50	140[3]	mVpp AC
Microphone Total Harmonic Distortion	MTHD		0.02	0.10	%
Microphone S/N Ratio	MSNR	50	62		dB
Line input amplitude			2200	2800[3]	mVpp AC
Line input Total Harmonic Distortion	LTHD		0.06	0.10	%
Line input S/N Ratio	LSNR	60	68		dB
Line and Microphone input impedances			100		kΩ

[1] 3.0 volts can be achieved with +-to-+ wiring for mono difference sound.
[2] AOLR may be much lower, but below *Typical* distortion performance may be compromised.
[3] Harmonic Distortion increases above typical amplitude.

3.4 Power Consumption

TBD

3.5 Digital Characteristics

Parameter	Symbol	Min	Typ	Max	Unit
High-Level Input Voltage		$0.7\times$IOVDD		IOVDD+0.3[1]	V
Low-Level Input Voltage		-0.2		$0.3\times$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 $\frac{CLKI}{4}$.

3.6 Switching Characteristics - Boot Initialization

Parameter	Symbol	Min	Max	Unit
XRESET active time		2		XTALI
XRESET inactive to software ready		16600	50000[1]	XTALI
Power on reset, rise time to CVDD		10		V/s

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

4 Packages and Pin Descriptions

4.1 Packages

Both LPQFP-48 and BGA-49 are lead (Pb) free and also RoHS compliant packages. 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*.

4.1.1 LQFP-48

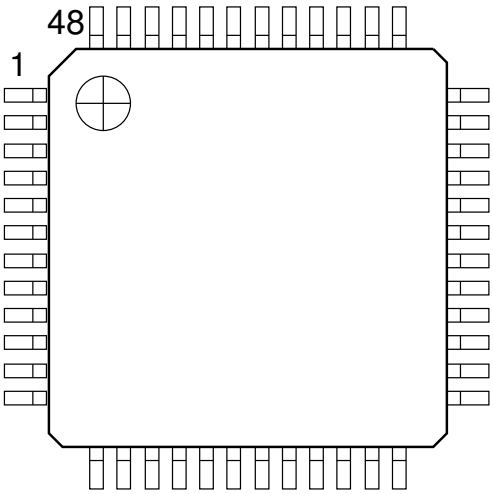


Figure 1: Pin Configuration, LQFP-48.

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

4.1.2 BGA-49

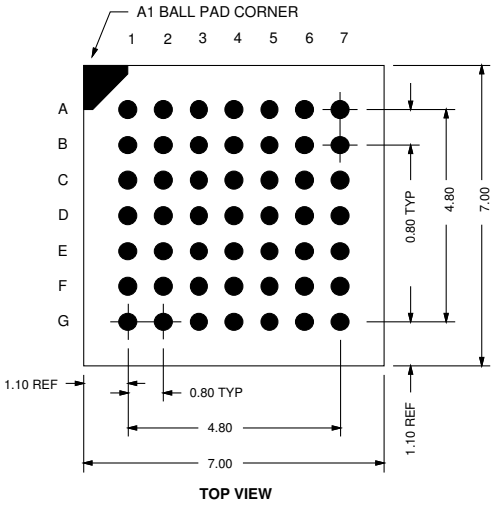


Figure 2: Pin Configuration, BGA-49.

BGA-49 package dimensions are at *http://www.vlsi.fi/* .

4.2 LQFP-48 and BGA-49 Pin Descriptions

Pin Name	LQFP-48 Pin	BGA49 Ball	Pin Type	Function
MICP	1	C3	AI	Positive differential microphone input, self-biasing
MICN	2	C2	AI	Negative differential microphone input, self-biasing
XRESET	3	B1	DI	Active low asynchronous reset
DGND0	4	D2	DGND	Core & I/O ground
CVDD0	5	C1	CPWR	Core power supply
IOVDD0	6	D3	IOPWR	I/O power supply
CVDD1	7	D1	CPWR	Core power supply
DREQ	8	E2	DO	Data request, input bus
GPIO2 / DCLK[1]	9	E1	DIO	General purpose IO 2 / serial input data bus clock
GPIO3 / SDATA[1]	10	F2	DIO	General purpose IO 3 / serial data input
XDCS / BSYNC[1]	13	E3	DI	Data chip select / byte sync
IOVDD1	14	F3	IOPWR	I/O power supply
VCO	15	G2	DO	For testing only (Clock VCO output)
DGND1	16	F4	DGND	Core & I/O ground
XTALO	17	G3	AO	Crystal output
XTALI	18	E4	AI	Crystal input
IOVDD2	19	G4	IOPWR	I/O power supply
IOVDD3		F5	IOPWR	I/O power supply
DGND2	20		DGND	Core & I/O ground
DGND3	21	G5	DGND	Core & I/O ground
DGND4	22	F6	DGND	Core & I/O ground
XCS	23	G6	DI	Chip select input (active low)
CVDD2	24	G7	CPWR	Core power supply
RX	26	E6	DI	UART receive, connect to IOVDD if not used
TX	27	F7	DO	UART transmit
SCLK	28	D6	DI	Clock for serial bus
SI	29	E7	DI	Serial input
SO	30	D5	DO3	Serial output
CVDD3	31	D7	CPWR	Core power supply
TEST	32	C6	DI	Reserved for test, connect to IOVDD
GPIO0 / SPIBOOT	33	C7	DIO	General purpose IO 0 / SPIBOOT, use 100 kΩ pull-down resistor[2]
GPIO1	34	B6	DIO	General purpose IO 1
AGND0	37	C5	APWR	Analog ground, low-noise reference
AVDD0	38	B5	APWR	Analog power supply
RIGHT	39	A6	AO	Right channel output
AGND1	40	B4	APWR	Analog ground
AGND2	41	A5	APWR	Analog ground
GBUF	42	C4	AO	Common buffer for headphones
AVDD1	43	A4	APWR	Analog power supply
RCAP	44	B3	AIO	Filtering capacitance for reference
AVDD2	45	A3	APWR	Analog power supply
LEFT	46	B2	AO	Left channel output
AGND3	47	A2	APWR	Analog ground
LINEIN	48	A1	AI	Line input

[1] First pin function is active in New Mode, latter in Compatibility Mode.
[2] Unless pull-down resistor is used, SPI Boot is tried. See Chapter 7.5 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

In BGA-49, no-connect balls are A7, B7, D4, E5, F1, G1.
In LQFP-48, no-connect pins are 11, 12, 25, 35, 36.

5 SPI Buses

5.1 General

The SPI Bus - that was originally used in some Motorola devices - has been used for both VS1103b's Serial Data Interface SDI (Chapters 5.4 and 6.4) and Serial Control Interface SCI (Chapters 5.5 and 6.5).

5.2 SPI Bus Pin Descriptions

5.2.1 VS10xx Native Modes (New Mode, recommended)

These modes are active on VS1103b when SM_SDINEW is set to 1 (default at startup). DCLK, SDATA and BSYNC are replaced with GPIO2, GPIO3 and XDCS, respectively.

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

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

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

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

5.3 Data Request Pin DREQ

The DREQ pin/signal is used to signal if VS1103b's SDI FIFO is capable of receiving data. If DREQ is high, VS1103b can take at least 32 bytes of SDI data or one SCI command.

Because of a 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 VS1103b easier for low-speed microcontrollers. If SARC_DREQ512 is set, the safety area is 512 bytes (see Chapter 6.6.13).

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. It should not abort a transmission that has already started.

Note: In VS10XX products up to VS1002, DREQ was only used for SDI. In VS1103b DREQ is also used to tell the status of SCI.

5.4 Serial Protocol for Serial Data Interface (SDI)

5.4.1 General

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

Data (SDATA signal) can be clocked in at either the rising or falling edge of DCLK (Chapter 6.6).

VS1103b assumes its data input to be byte-sychronized. SDI bytes may be transmitted either MSb or LSb first, depending of SCI_MODE (Chapter 6.6.1).

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

In VS10xx native modes (SM_NEWMODE is 1), byte synchronization is achieved by XDCS. The state of XDCS may not change while a data byte transfer is in progress. To always maintain data synchronization even if there may be glitches in the boards using VS1103b, it is recommended to turn XDCS every now and then, for instance once after every flash data block or a few kilobytes, just to keep sure the host and VS1103b are in sync.

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

For new designs, using VS10xx native modes are recommended.

5.4.3 SDI in VS1001 Compatibility Mode

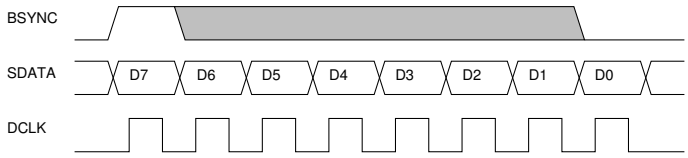


Figure 3: BSYNC Signal - one byte transfer.

When VS1103b is running in VS1001 compatibility mode, a BSYNC signal must be generated to ensure correct bit-alignment of the input bitstream. The first DCLK sampling edge (rising or falling, depending on selected polarity), during which the BSYNC is high, marks the first bit of a byte (LSB, if LSB-first order is used, MSB, if MSB-first order is used). If BSYNC is '1' when the last bit is received, the receiver stays active and next 8 bits are also received.

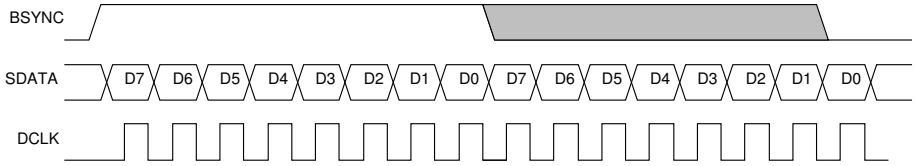


Figure 4: BSYNC Signal - two byte transfer.

5.4.4 Passive SDI Mode

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.

5.5 Serial Protocol for Serial Command Interface (SCI)

5.5.1 General

The serial bus protocol for the Serial Command Interface SCI (Chapter 6.5) consists of an instruction byte, address byte and one 16-bit data word. Each read or write operation can read or write a single register. Data bits are read at the rising edge, so the user should update data at the falling edge. Bytes are always sent MSb first. XCS should be low for the full duration of the operation, but you can have pauses between bits if needed.

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

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

Note: VS1103b 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.

5.5.2 SCI Read

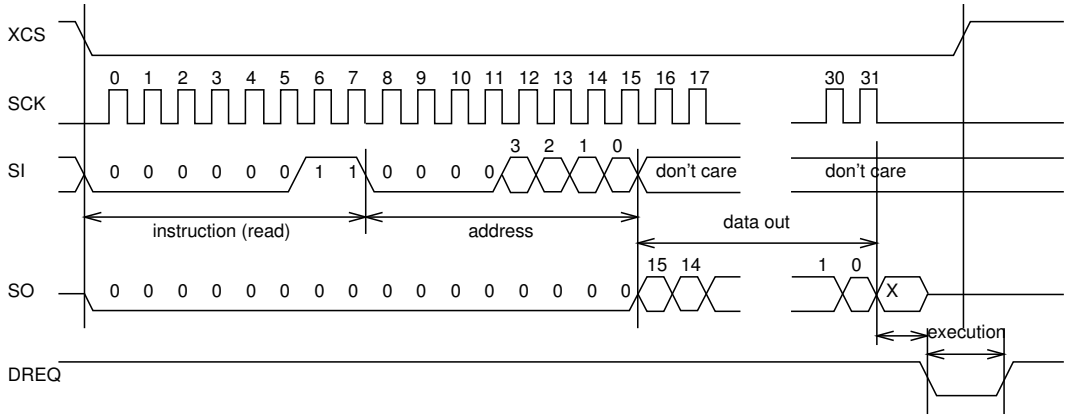


Figure 5: SCI Word Read

VS1103b registers are read from using the following sequence, as shown in Figure 5. First, XCS line is pulled low to select the device. Then the READ opcode (0x3) is transmitted via the SI line followed by an 8-bit word address. After the address has been read in, any further data on SI is ignored by the chip. The 16-bit data corresponding to the received address will be shifted out onto the SO line.

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

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

5.5.3 SCI Write

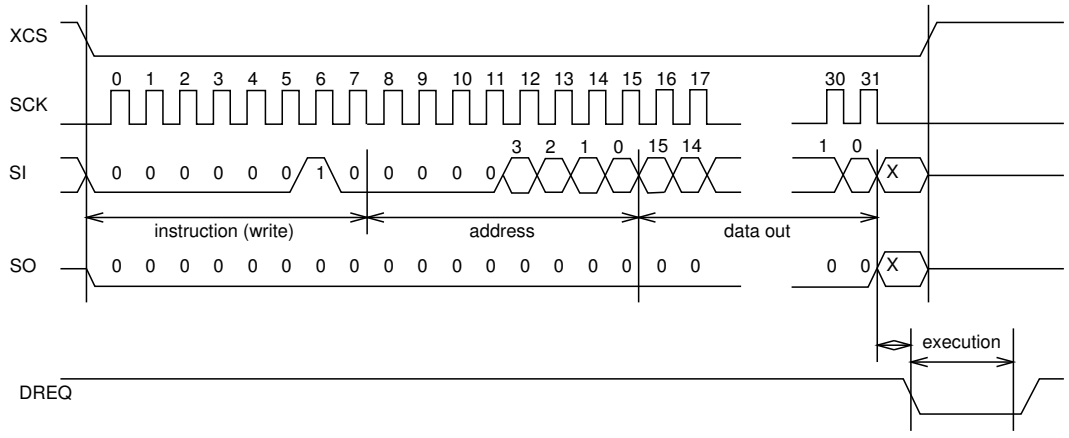


Figure 6: SCI Word Write

VS1103b registers are written from using the following sequence, as shown in Figure 6. 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 6.6 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.

5.5.4 SCI Multiple Write

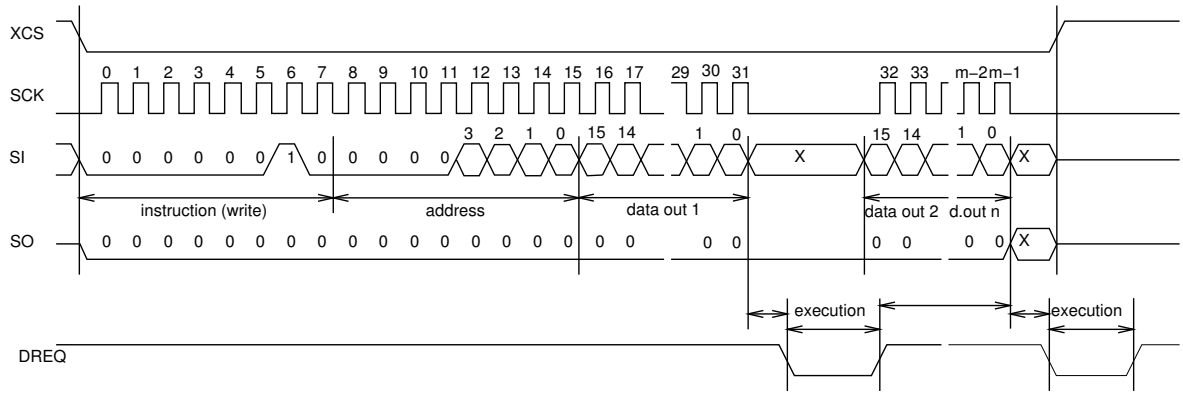


Figure 7: SCI Multiple Word Write

VS1103b allows for the user to send multiple words to the same SCI register, which allows fast SCI uploads, shown in Figure 7. The main difference with 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 6.6 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.

5.6 SPI Timing Diagram

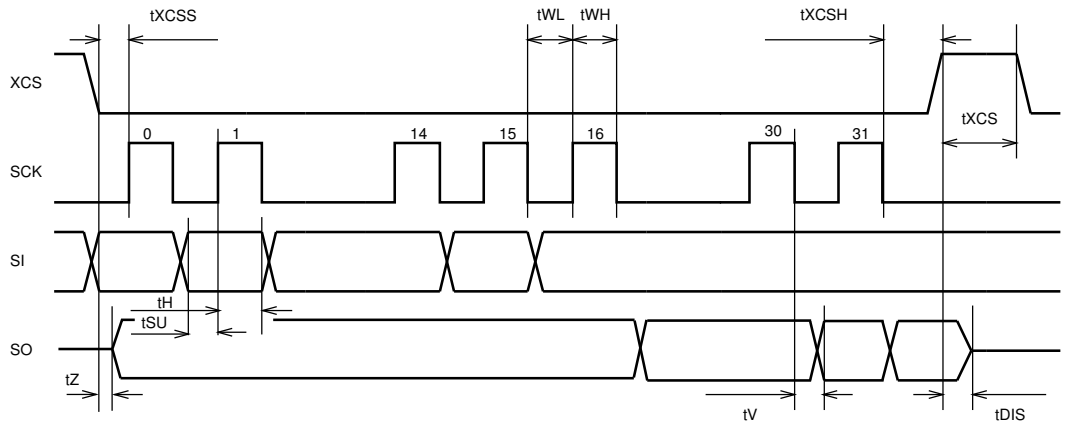


Figure 8: SPI Timing Diagram.

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 (+ 25ns[1])		CLKI cycles
tXCSH	1		CLKI cycles
tXCS	2		CLKI cycles
tDIS		10	ns

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

Note: As tWL and tWH, as well as tH require at least 2 clock cycles, the maximum speed for the SPI bus that can easily be used with asynchronous clocks is 1/7 of VS1103b's internal clock speed CLKI.

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

Note: Negative numbers mean that the signal can change in different order from what is shown in the diagram.

5.7 SPI Examples with SM_SDINEW and SM_SDISHARED set

5.7.1 Two SCI Writes

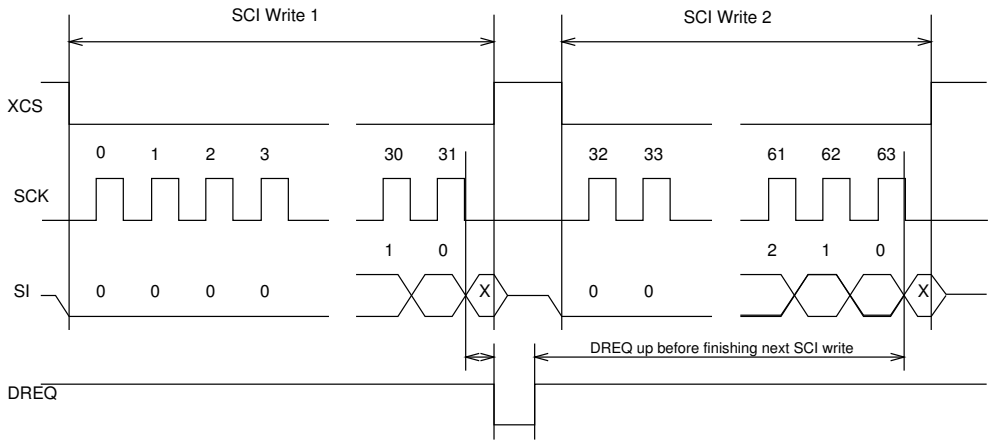


Figure 9: Two SCI Operations.

Figure 9 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.

5.7.2 Two SDI Bytes

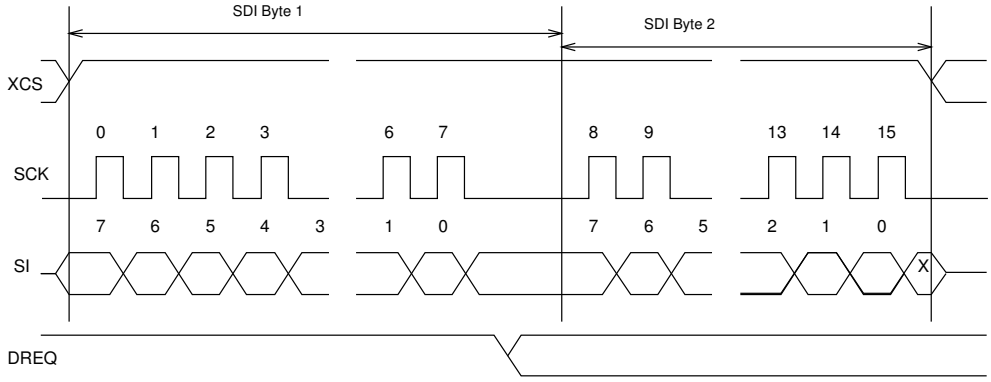


Figure 10: Two SDI Bytes.

SDI data is synchronized with a raising edge of xCS as shown in Figure 10. However, every byte doesn't need separate synchronization.

5.7.3 SCI Operation in Middle of Two SDI Bytes

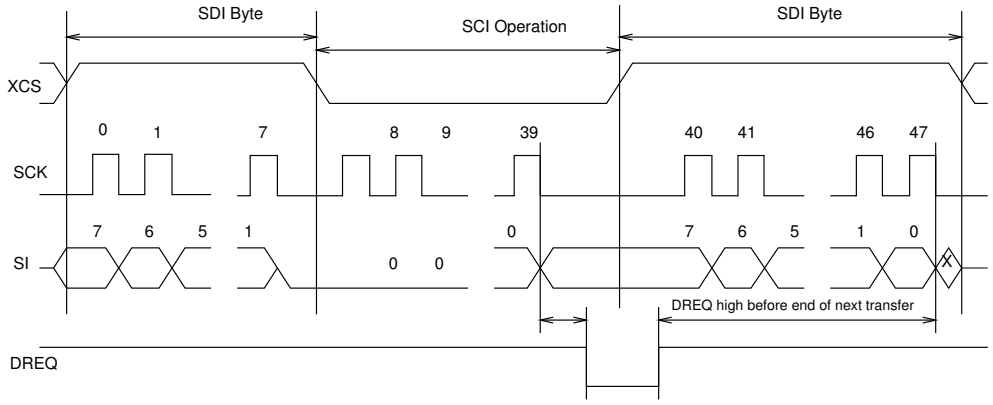


Figure 11: Two SDI Bytes Separated By an SCI Operation.

Figure 11 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.

6 Functional Description

6.1 Main Features

VS1103b is based on a proprietary digital signal processor, VS_DSP. It contains all the code and data memory needed for WAV PCM + ADPCM audio decoding, MIDI synthesizer, together with serial interfaces, a multirate stereo audio DAC and analog output amplifiers and filters. Also ADPCM audio encoding is supported using a microphone amplifier and A/D converter. A UART is provided for debugging purposes.

6.2 Supported Audio Codecs

6.2.1 Supported RIFF WAV Formats

The most common RIFF WAV subformats are supported.

Format	Name	Supported	Comments
0x01	PCM	+	16 and 8 bits, any sample rate ≤ 48kHz
0x02	ADPCM	-	
0x03	IEEE_FLOAT	-	
0x06	ALAW	-	
0x07	MULAW	-	
0x10	OKI_ADPCM	-	
0x11	IMA_ADPCM	+	Any sample rate ≤ 48kHz
0x15	DIGISTD	-	
0x16	DIGIFIX	-	
0x30	DOLBY_AC2	-	
0x31	GSM610	-	
0x3b	ROCKWELL_ADPCM	-	
0x3c	ROCKWELL_DIGITALK	-	
0x40	G721_ADPCM	-	
0x41	G728_CELP	-	
0x50	MPEG	-	
0x55	MPEGLAYER3	-	
0x64	G726_ADPCM	-	
0x65	G722_ADPCM	-	

6.2.2 Supported MIDI Formats

General MIDI and SP-MIDI format 0 files are played. Format 1 and 2 files must be converted to format 0 by the user. The maximum simultaneous polyphony is 40 (peak polyphony 64). Actual polyphony depends on the internal clock rate (which is user-selectable), the instruments used, and the possible postprocessing effects enabled, such as bass and treble enhancers. The polyphony restriction algorithm makes use of the SP-MIDI MIP table, if present. MIDI implements click-avoiding smooth note removal.

When run in Real Time RT-Midi mode without other signal paths, 36.86 MHz ($3.0\times$ input clock) achieves 16-26 simultaneous sustained notes. The instantaneous amount of notes can be larger. 36 MHz is a fair compromise between power consumption and quality, but higher clocks can be used to increase polyphony. They are also needed if multiple signal paths are used.

Reverb effect can be controlled by the user. In addition to reverb automatic and reverb off modes, 14 different decay times can be selected. These roughly correspond to different room sizes. Also, each midi song decides how much effect each instrument gets. Because the reverb effect uses about 4 MHz of processing power the automatic control enables reverb only when the internal clock is at least $3.0\times$.

When EarSpeaker spatial processing is active, MIDI reverb is not used.

VS1103b supports unique instruments in the whole GM1 instrument set and one bank of GM2 percussions.

VS1103b Melodic Intruments (GM1)			
1 Acoustic Grand Piano	33 Acoustic Bass	65 Soprano Sax	97 Rain (FX 1)
2 Bright Acoustic Piano	34 Electric Bass (finger)	66 Alto Sax	98 Sound Track (FX 2)
3 Electric Grand Piano	35 Electric Bass (pick)	67 Tenor Sax	99 Crystal (FX 3)
4 Honky-tonk Piano	36 Fretless Bass	68 Baritone Sax	100 Atmosphere (FX 4)
5 Electric Piano 1	37 Slap Bass 1	69 Oboe	101 Brightness (FX 5)
6 Electric Piano 2	38 Slap Bass 2	70 English Horn	102 Goblins (FX 6)
7 Harpsichord	39 Synth Bass 1	71 Bassoon	103 Echoes (FX 7)
8 Clavi	40 Synth Bass 2	72 Clarinet	104 Sci-fi (FX 8)
9 Celesta	41 Violin	73 Piccolo	105 Sitar
10 Glockenspiel	42 Viola	74 Flute	106 Banjo
11 Music Box	43 Cello	75 Recorder	107 Shamisen
12 Vibraphone	44 Contrabass	76 Pan Flute	108 Koto
13 Marimba	45 Tremolo Strings	77 Blown Bottle	109 Kalimba
14 Xylophone	46 Pizzicato Strings	78 Shakuhachi	110 Bag Pipe
15 Tubular Bells	47 Orchestral Harp	79 Whistle	111 Fiddle
16 Dulcimer	48 Timpani	80 Ocarina	112 Shanai
17 Drawbar Organ	49 String Ensembles 1	81 Square Lead (Lead 1)	113 Tinkle Bell
18 Percussive Organ	50 String Ensembles 2	82 Saw Lead (Lead)	114 Agogo
19 Rock Organ	51 Synth Strings 1	83 Calliope Lead (Lead 3)	115 Pitched Percussion
20 Church Organ	52 Synth Strings 2	84 Chiff Lead (Lead 4)	116 Woodblock
21 Reed Organ	53 Choir Aahs	85 Charang Lead (Lead 5)	117 Taiko Drum
22 Accordion	54 Voice Oohs	86 Voice Lead (Lead 6)	118 Melodic Tom
23 Harmonica	55 Synth Voice	87 Fifths Lead (Lead 7)	119 Synth Drum
24 Tango Accordion	56 Orchestra Hit	88 Bass + Lead (Lead 8)	120 Reverse Cymbal
25 Acoustic Guitar (nylon)	57 Trumpet	89 New Age (Pad 1)	121 Guitar Fret Noise
26 Acoustic Guitar (steel)	58 Trombone	90 Warm Pad (Pad 2)	122 Breath Noise
27 Electric Guitar (jazz)	59 Tuba	91 Polysynth (Pad 3)	123 Seashore
28 Electric Guitar (clean)	60 Muted Trumpet	92 Choir (Pad 4)	124 Bird Tweet
29 Electric Guitar (muted)	61 French Horn	93 Bowed (Pad 5)	125 Telephone Ring
30 Overdriven Guitar	62 Brass Section	94 Metallic (Pad 6)	126 Helicopter
31 Distortion Guitar	63 Synth Brass 1	95 Halo (Pad 7)	127 Applause
32 Guitar Harmonics	64 Synth Brass 2	96 Sweep (Pad 8)	128 Gunshot

VS1103b Percussion Intruments (GM1+GM2)			
27 High Q	43 High Floor Tom	59 Ride Cymbal 2	75 Claves
28 Slap	44 Pedal Hi-hat [EXC 1]	60 High Bongo	76 Hi Wood Block
29 Scratch Push [EXC 7]	45 Low Tom	61 Low Bongo	77 Low Wood Block
30 Scratch Pull [EXC 7]	46 Open Hi-hat [EXC 1]	62 Mute Hi Conga	78 Mute Cuica [EXC 4]
31 Sticks	47 Low-Mid Tom	63 Open Hi Conga	79 Open Cuica [EXC 4]
32 Square Click	48 High Mid Tom	64 Low Conga	80 Mute Triangle [EXC 5]
33 Metronome Click	49 Crash Cymbal 1	65 High Timbale	81 Open Triangle [EXC 5]
34 Metronome Bell	50 High Tom	66 Low Timbale	82 Shaker
35 Acoustic Bass Drum	51 Ride Cymbal 1	67 High Agogo	83 Jingle bell
36 Bass Drum 1	52 Chinese Cymbal	68 Low Agogo	84 Bell tree
37 Side Stick	53 Ride Bell	69 Cabasa	85 Castanets
38 Acoustic Snare	54 Tambourine	70 Maracas	86 Mute Surdo [EXC 6]
39 Hand Clap	55 Splash Cymbal	71 Short Whistle [EXC 2]	87 Open Surdo [EXC 6]
40 Electric Snare	56 Cowbell	72 Long Whistle [EXC 2]	
41 Low Floor Tom	57 Crash Cymbal 2	73 Short Guiro [EXC 3]	
42 Closed Hi-hat [EXC 1]	58 Vibra-slap	74 Long Guiro [EXC 3]	

6.3 Data Flow of VS1103b

6.3.1 Normal Data Flow

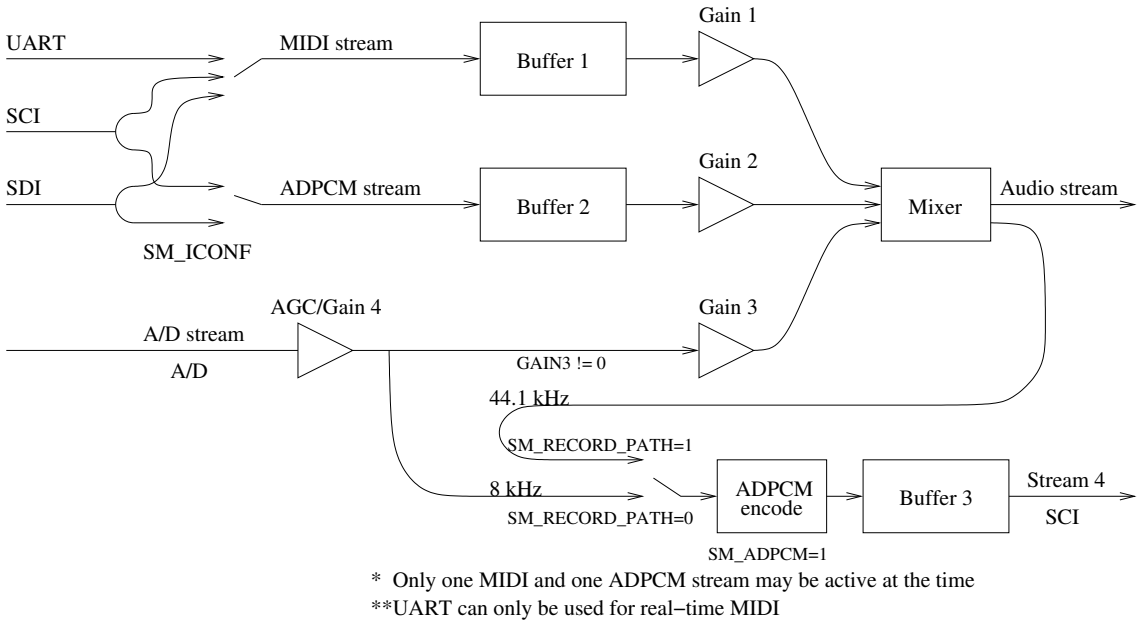


Figure 12: Normal Data Flow of VS1103b, Part 1.

Generation of the Audio stream and recording A/D stream is presented in Figure 12.

Stream 1, which is the MIDI stream, may be fed either through SDI, SCI or UART. If it is fed through UART, real-time MIDI, or RT-MIDI is assumed. The buffer size is 1024 bytes.

Stream 2, which is the ADPCM stream, may be fed either through SDI or SCI. The buffer size is 1024 bytes.

Stream 3, which is the A/D stream, running always at 8 kHz, is active if register SM_ADPCM is set.

The outputs of the three streams are forwarded to the Mixer, which resamples all data to 44.1 kHz, and forwards the data.

Either one of the A/D stream and the output of the Mixer can be fed to ADPCM encoding. If the data is read from the A/D stream, it will be encoded as 8 kHz mono and if it is read from the Mixer, it will encode as 44.1 kHz mono. The ADPCM compressed data may be read from SCI registers SCI_IN0 and SCI_IN1. The buffer size is 1024 bytes.

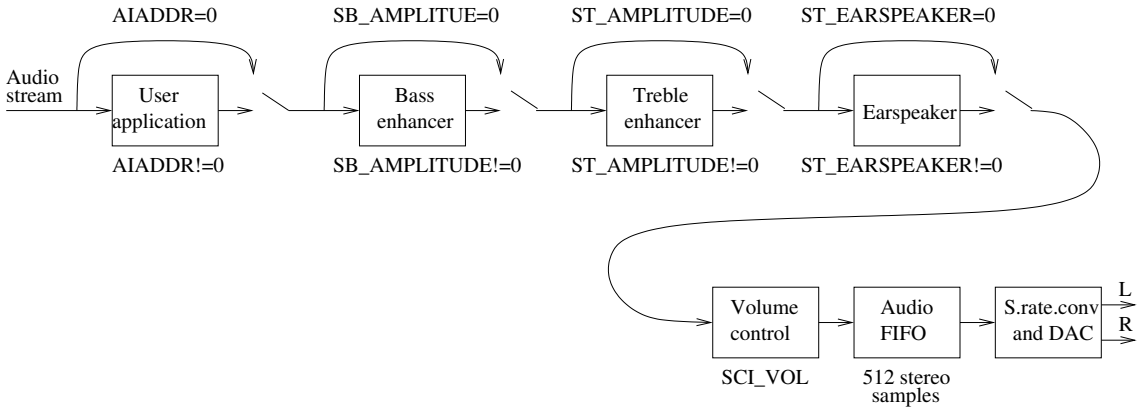


Figure 13: Normal Data Flow of VS1103b, Part 2.

Figure 13 presents the data flow of the Audio stream generated in Figure 12.

If SCI_AIADDR is non-zero, application code is executed from the address pointed to by that register. For more details, see Application Notes for VS10XX.

Then data may be sent to the Bass and Treble Enhancer depending on the SCI_BASS register, followed by Earspeaker Spatial Processing, depending on ST_EARSPEAKER.

After that the signal is fed to the volume control unit, which also copies the data to the Audio FIFO.

The Audio FIFO holds the data, which is read by the Audio interrupt (Chapter 8.13.1) and fed to the sample rate converter and DACs. The size of the audio FIFO is 1024 stereo (2×16-bit) samples, or 4 KiB.

The sample rate converter converts all different sample rates to XTALI/2, or 128 times the highest usable sample rate. This removes the need for complex PLL-based clocking schemes and allows almost unlimited sample rate accuracy with one fixed input clock frequency. With a 12.288 MHz clock, the DA converter operates at 128×48 kHz, i.e. 6.144 MHz, and creates a stereo in-phase analog signal. The oversampled output is low-pass filtered by an on-chip analog filter. This signal is then forwarded to the earphone amplifier.

6.3.2 Real-Time RT-Midi Mode

If GPIO1 is 1 and GPIO0 is 0 at startup, RT-Midi Mode is activated. In this mode RT-Midi data is read through the UART at the default MIDI speed 31250 bit/s. The generated audio is sent to the audio path as shown in Figure 13.

When RT-MIDI mode is activated, GPIO2 and GPIO3 are read and their contents are copied to register bits SCIMB_EARSPEAKER0 and SCIMB_EARSPEAKER1, respectively. This way it is possible to activate EarSpeaker in this mode without writing to any SCI registers. Also, if SCI_CLOCKF has not been set to a non-zero value, the clock multiplier is automatically set to 3.5X.

This mode is intended for connecting a MIDI keyboard or sequencer to VS1103b.

6.4 Serial Data Interface (SDI)

The serial data interface is meant for transferring ADPCM or MIDI data.

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 7.

6.5 Serial Control Interface (SCI)

The serial control interface is compatible with the SPI bus specification. Data transfers are always 16 bits. VS1103b is controlled by writing and reading the registers of the interface.

The main controls of the control interface are:

- control of the operation mode, clock, and builtin effects
- access to status information and header data
- access to encoded digital data
- uploading user programs

6.6 SCI Registers

SCI registers, prefix SCI_					
Reg	**Type**	**Reset**	**Time**[1]	**Abbrev[bits]**	**Description**
0x0	rw	0x800	200 CLKI[4]	MODE	Mode control
0x1	rw	0x3C[3]	40 CLKI	STATUS	Status of VS1103b
0x2	rw	0	2100 CLKI	BASS	Built-in bass/treble enhancer
0x3	rw	0	11000 XTALI[5]	CLOCKF	Clock freq + multiplier
0x4	rw	0	40 CLKI	DECODE_TIME	Stream 0 decode time
0x5	rw	0	3200 CLKI	AUDATA	Misc. audio data
0x6	rw	0	80 CLKI	WRAM	RAM write/read
0x7	rw	0	80 CLKI	WRAMADDR	Base address for RAM write/read
0x8	rw	0	90 CLKI	IN0	Input 0
0x9	rw	0	90 CLKI	IN1	Input 1
0xA	rw	0	3200 CLKI[2]	AIADDR	Start address of application
0xB	rw	0	2100 CLKI	VOL	Volume control
0xC	rw	0	70 CLKI[2]	MIXERVOL	Mixer volume
0xD	rw	0	50 CLKI[2]	ADPCMRECCTL	IMA ADPCM record control
0xE	rw	0	50 CLKI[2]	AICTRL2	Application control register 2
0xF	rw	0	50 CLKI[2]	AICTRL3	Application control register 3

[1] This is the worst-case time that DREQ stays low after writing to / reading from this register. The user may choose to skip the DREQ check for those register writes that take less than 100 clock cycles to execute.
[2] In addition, the cycles spent in the user application routine must be counted.
[3] Firmware changes the value of this register immediately after reset to 0x38, and in less than 100 ms to 0x30.

[4] When mode register write specifies a software reset the worst-case time is 16600 XTALI cycles.
[5] Writing to this register may force internal clock to run at $1.0 \times$ XTALI for a while. Thus it is not a good idea to send SCI or SDI bits while this register update is in progress.

Note: it is not allowed to do an SCI operation while DREQ is low. If this is done, however, DREQ stays low even after the SCI operation has been processed.

6.6.1 SCI_MODE (RW)

SCI_MODE is used to control the operation of VS1103b and defaults to 0x0800 (SM_SDINEW set).

SCI_MODE bits				
Name	**Bit**	**Function**	**Value**	**Description**
SM_DIFF	0	Differential	0 1	normal in-phase audio left channel inverted
SM_RECORD_PATH	1	Choose ADPCM recording path	0 1	A/D stream Mixer output
SM_RESET	2	Soft reset	0 1	no reset reset
SM_OUTOFMIDI	3	Cancel MIDI decoding	0 1	no yes
SM_PDOWN	4	Powerdown	0 1	power on powerdown
SM_TESTS	5	Allow SDI tests	0 1	not allowed allowed
SM_ICONF	7:6	Input configuration	0 1 2 3	SDI MIDI, SCI ADPCM SCI MIDI, SDI ADPCM UART RT-MIDI, SCI ADPCM UART RT-MIDI, SDI ADPCM
SM_DACT	8	DCLK active edge	0 1	rising falling
SM_SDIORD	9	SDI bit order	0 1	MSb first MSb last
SM_SDISHARE	10	Share SPI chip select	0 1	no yes
SM_SDINEW	11	VS10xx native SPI modes	0 1	no yes
SM_EARSPEAKER	13:12	Earspeaker setting	0 1 2 3	off low mid high
SM_LINE_IN	14	A/D stream input selector	0 1	microphone line in
SM_ADPCM	15	ADPCM recording active	0 1	no yes

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

If SM_RECORD_PATH is set, ADPCM recording is performed from the A/D stream at 8 kHz, otherwise the Mixer output is recorded at 44.1 kHz. This bit is only valid if SM_ADPCM is set.

Software reset is initiated by setting SM_RESET to 1. This bit is cleared automatically.

To stop decoding a MIDI file set SM_OUTOFMIDI, and send data until SM_OUTOFMIDI has cleared. If SM_OUTOFMIDI is set while MIDI decoding has not been going on, the register bit will not be cleared before the few first words of the next MIDI file (or zeros) have been sent to the decoder.

Bit SM_PDOWN sets VS1103b into software powerdown mode where the only operational software part is the control bus handler. Note: software powerdown is not nearly as power efficient as hardware powerdown activated with the XRESET pin.

If SM_TESTS is set, SDI tests are allowed. For more details on SDI tests, look at Chapter 7.8.

SM_ICONF specifies the configuration of the data input streams. The following table shows its bits.

SM_ICONF	S1 Port	Stream1	S2 Port	Stream2
0	SDI	MIDI	SCI	ADPCM
1	SCI	MIDI	SDI	ADPCM
2	UART/SDI	RT-MIDI/RT-SDI	SCI	ADPCM
3	UART	RT-MIDI	SDI	ADPCM

When SM_ICONF is set to 2, Real Time MIDI messages can be sent either through the UART or SDI. If sent through UART, the standard MIDI protocol and date speed (31250 bit/s) is used. If send through SDI, the protocol is otherwise the same, but every byte must either be preceded or followed by a zero byte (but only one of these two alternative zero byte orders may be used at a time). So, a message that would be sent as 0x92 0x37 0x73 through normal MIDI, would become 0x92 0x00 0x37 0x00 0x73 0x00 if sent through SDI.
NOTE! If you change SM_ICONF, a software reset is performed as if you had also set SM_RESET!

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 as a default 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 5.2, if also SM_SDINEW is set.

Setting SM_SDINEW will activate VS10xx native serial modes as described in Chapters 5.2.1 and 5.4.2. Note, that this bit is set as a default when VS1103b is started up.

Bits in SM_EARSPEAKER control EarSpeaker spatial processing. They are used as follows:

SM_EARSPEAKER	Setting
0	Off
1	Minimal
2	Normal
3	Extreme

EarSpeaker uses approximately 6 MIPS at 44.1 kHz sample rate.

SM_LINE_IN is used to select the input for ADPCM recording. If ’0’, microphone input pins MICP and MICN are used; if ’1’, LINEIN is used.

When SM_ADPCM is turned on, ADPCM encoding is activated (see Image 12 at Page 26).

6.6.2 SCI_STATUS (RW)

SCI_STATUS contains information on the current status of VS1103b and lets the user shutdown the chip without audio glitches.

Name	Bits	Description
SS_VER	7:4	Version
SS_APDOWN2	3	Analog driver powerdown
SS_APDOWN1	2	Analog internal powerdown
SS_AVOL	1:0	Analog volume control

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

SS_APDOWN2 controls analog driver powerdown. Normally this bit is controlled by the system firmware. However, if the user wants to powerdown VS1103b with a minimum power-off transient, turn this bit to 1, then wait for at least a few milliseconds before activating reset.

SS_APDOWN1 controls internal analog powerdown. This bit is meant to be used by the system firmware only.

SS_AVOL is the analog volume control: 0 = -0 dB, 1 = -6 dB, 3 = -12 dB. This register is meant to be used automatically by the system firmware only.

6.6.3 SCI_BASS (RW)

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 (0..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 powerful bass boosting DSP algorithm, which tries to take the most out of the users earphones without causing clipping.

VSBE is activated when SB_AMPLITUDE is non-zero. SB_AMPLITUDE should be set to the user's preferences, and SB_FREQLIMIT to roughly 1.5 times the lowest frequency the user's audio system can reproduce. For example setting SCI_BASS to 0x00a6 will give 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 from nothing: the source material must have some bass to begin with.

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

Bass Enhancer uses about 3.0 MIPS and Treble Control 1.2 MIPS at 44100 Hz sample rate. Both can be on simultaneously.

6.6.4 SCI_CLOCKF (RW)

SCI_CLOCKF is used to control the internal clock of VS1103b.

SCI_CLOCKF bits		
Name	**Bits**	**Description**
SC_MULT	15:13	Clock multiplier
SC_ZERO	12:11	Set to zero
SC_FREQ	10: 0	Clock frequency

SC_MULT activates the built-in clock multiplier. This will multiply XTALI to create a higher CLKI. The values are as follows:

SC_MULT	MASK	CLKI
0	0x0000	XTALI
1	0x2000	XTALI$\times 1.5$
2	0x4000	XTALI$\times 2.0$
3	0x6000	XTALI$\times 2.5$
4	0x8000	XTALI$\times 3.0$
5	0xa000	XTALI$\times 3.5$
6	0xc000	XTALI$\times 4.0$
7	0xe000	XTALI$\times 4.5$

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

Note: As opposed to some other VS10XX chips, a software reset must be performed after SCI_CLOCKF has been set. It is recommended that SCI_CLOCKF is set only after each hardware reset / startup.

Note: The default value 0 is assumed to mean XTALI=12.288 MHz.

Note: Because maximum sample rate is $\frac{XTALI}{256}$, all sample rates are not available if XTALI < 12.288 MHz.

Example: If SCI_CLOCKF is 0xC3E8, SC_MULT = 6 and SC_FREQ = 0x3E8 = 1000. This means that XTALI = $1000 \times 4000 + 8000000$ = 12 MHz. The clock multiplier is set to $4.0\times$XTALI $=$ 48 MHz.

6.6.5 SCI_DECODE_TIME (RW)

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

The user may change the value of this register. In that case the new value should be written twice.

SCI_DECODE_TIME is reset at every software reset and also when MIDI decoding starts or ends.

6.6.6 SCI_AUDATA (RW)

The current sample rate and number of channels can be found 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. Writing to SCI_AUDATA will change the sample rate directly (not recommended for VS1103b!).

As VS1103b always runs in stereo mode at 44100 Hz, contents of this register is always 0xAC45 (44101).

6.6.7 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. As 16 bits of data can be transferred with one SCI_WRAM write/read, and 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 byte first). After each full-word write/read, the internal pointer is autoincremented.

6.6.8 SCI_WRAMADDR (W)

SCI_WRAMADDR is used to set the program address for following SCI_WRAM writes/reads. Address offset of 0 is used for X, 0x4000 for Y, and 0x8000 for instruction memory. Peripheral registers can also be accessed.

SM_WRAMADDR Start...End	Dest. addr. Start...End	Bits/ Word	Description
0x1800...0x187F	0x1800...0x187F	16	X data RAM
0x5800...0x587F	0x1800...0x187F	16	Y data RAM
0x8030...0x84FF	0x0030...0x04FF	32	Instruction RAM
0xC000...0xFFFF	0xC000...0xFFFF	16	I/O

Only user areas in X, Y, and instruction memory are listed above. Other areas can be accessed, but should not be written to unless otherwise specified.

6.6.9 SCI_IN0 and SCI_IN1 (R)

SCI_IN0 and SCI_IN1 are used for offering SCI stream data and for reading encoded ADPCM Stream 4 data.

The bits in the registers are as follows:

Register	R/W	Bits/	Description
SCI_IN0	Read	15:0	Read one word from Stream 4
SCI_IN0	Write	15:0	Write one word to SCI sourced stream
SCI_IN1	Read	15:8	Number of words $\times 8$ that can be read from SCI_IN0 (Stream 4)
SCI_IN1	Read	7:0	Number of words $\times 8$ that can be written to SCI_IN0 (SCI sourced stream)

Note: Data word length is 16 bits.

Example: If reading SCI_IN1 returns 0x0312, then 0x03$\times$8 words = 24 words = 48 bytes can be read from SCI_IN0 and 0x12$\times$8 words = 144 words = 288 bytes can be written to SCI_IN0.

6.6.10 SCI_AIADDR (RW)

SCI_AIADDR indicates the start address of the application code written earlier with SCI_WRAMADDR and SCI_WRAM registers. If no application code is used, this register should not be initialized, or it should be initialized to zero. For more details, see Application Notes for VS10XX.

6.6.11 SCI_VOL (RW)

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

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

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

Example: SCI_VOL = 0x2424 $\rightarrow$ both left and right volumes are 0x24 * -0.5 = -18.0 dB.

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

6.6.12 SCI_MIXERVOL (RW)

Control mixer volume. The contents of this register is as follows:

SCI_MIXERVOL bits		
Name	**Bits**	**Description**
SMV_ACTIVE	15	Control active
SMV_GAIN3	14:10	Gain 3
SMV_GAIN2	9:5	Gain 2
SMV_GAIN1	4:0	Gain 1

Gain values are defined in 1 dB steps so that 25 corresponds to 0 dB (signal is passed on as is) and 31 is +6 dB (signal is doubled). 0 means the channel is disabled.

If SMV_ACTIVE is 0, then Gain 1 is set to 25 (0 dB), and both Gain 2 and Gain 3 are set to 0 (mute).

See Figure 12 on page 26 for more details on where gains are applied.

Note: The polarity of the gains are opposite to registr SCI_VOL: higher means a higher gain, not higher attenuation.

6.6.13 SCI_ADPCMRECCTL (RW)

SCI_ADPCMRECCTL bits		
Name	**Bits**	**Description**
SARC_DREQ512	8	If set, DREQ needs 512 byte space to turn on.
SARC_OUTOFADPCM	7	If set, current ADPCM playback is canceled.
SARC_MANUALGAIN	6	If set, automatic gain control (AGC) is not used.
SARC_GAIN4	5:0	If SARC_MANUALGAIN is 1, this is Gain 4; otherwise it is maximum gain of AGC

SARC_DREQ512 affects how the DREQ pin works. If not set, when DREQ is active there is at least 32 bytes space to write to. If set, DREQ is set only when there is at least 512 bytes of free space in the SDI input buffer.

SARC_OUTOFADPCM does same to ADPCM playback as SCI_MODE register bit SM_OUTOFMIDI does to MIDI playback. Thus, if you want to stop decoding an ADPCM file, set SARC_OUTOFADPCM, and send data until SARC_OUTOFADPCM is cleared.

SARC_MANUALGAIN controls whether Gain 4 is manual or automatic.

If SARC_MANUALGAIN is set to 1, SARC_GAIN4 sets Gain 4. Otherwise SARC_GAIN4 sets the maximum gain allowed for the automatic gain control. The value is set at 1 dB steps and value 25 means 0 dB gain (signal is passed on without change). 31 is equal to to +6 dB gain, etc. 0 disables the signal path completely.

6.6.14 SCI_AICTRL[x] (RW)

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

Note: VS1103b reservs AICTRL0 as SCI_MIXERVOL and AICTRL1 as SCI_ADPCMRECCTL. They can, however, also be used for user applications if the applications don't conflict with the originally intended register contents.

7 Operation

7.1 Clocking

VS1103b 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).

7.2 Hardware Reset

When the XRESET -signal is driven low, VS1103b 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 VS1103b are in minimum power consumption stage, and where clocks are stopped. Also XTALO is grounded.

After a hardware reset (or at power-up) DREQ will stay down for at least 16600 clock cycles, which means an approximate 1.35 ms delay if VS1103b is run at 12.288 MHz. After this the user should set SCI_CLOCKF, perform a software reset, and then set other basic software registers as e.g. SCI_MODE, SCI_BASS, and SCI_VOL before starting decoding. See section 6.6 for details.

The internal clock can be multiplied with a PLL. Supported multipliers through the SCI_CLOCKF register are $1.0 \times \ldots 4.5\times$ the input clock. Reset value for Internal Clock Multiplier is $1.0\times$. If typical values are wanted, the Internal Clock Multiplier needs to be set to $4.0\times$ after reset. Wait until DREQ rises, then write a proper value to SCI_CLOCKF, followed by a software reset. See section 6.6.4 for details.

After XRESET is released, a software reset operation is also performed.

7.3 Software Reset

In some cases the decoder software has to be reset. This is done by activating bit 2 in SCI_MODE register (Chapter 6.6.1). Then wait for at least 2 μs, then look at DREQ. DREQ will stay down for at least 16600 clock cycles, which means an approximate 1.35 ms delay if VS1103b is run at 12.288 MHz. After DREQ is up, you may continue playback as usual.

If GPIO0 is set to 1, Spi Boot is performed (Chapter 7.5). If GPIO0 is set to 0 and GPIO1 to 1, RT-MIDI Mode is activated (Chapter 6.3.2).

As opposed to some earlier VS10XX products, VS1103b has been designed so that using software resets during normal operation shouldn't be necessary.

7.4 ADPCM Recording

This chapter explains how to create RIFF/WAV file with IMA ADPCM format. This is a widely supported ADPCM format and many PC audio playback programs can play it.

IMA ADPCM recording gives roughly a compression ratio of 4:1 compared to linear, 16-bit audio. This makes it possible to record approx. 8 kHz audio at approx. 32.44 kbit/s or 44.1 kHz audio at 178.85 kbit/s.

7.4.1 Activating ADPCM Recording

IMA ADPCM recording mode is activated by setting bit SM_ADPCM in SCI_MODE. Before activating ADPCM recording, user **must** see to it that SCI_ADPCMRECCTL has been properly set.

7.4.2 Reading IMA ADPCM Data

After IMA ADPCM recording has been activated, results can be read through registers SCI_IN0 and SCI_IN1.

The IMA ADPCM sample buffer size is 512 16-bit words, or 1 KiB. If the data is not read fast enough, the buffer overflows and returns to empty state.

Each IMA ADPCM block consists of 128 words, i.e. 256 bytes (or 505 mono audio samples). If you wish to interrupt reading data and possibly continue later, please stop at a 128-word boundary. This way whole blocks are skipped and the encoded stream stays valid.

Note: if SCI_IN1[15:8] ≥ 60 (i.e. there are more than $60 \times 8 = 480$ words waiting), wait for the buffer to overflow and clear before reading samples to avoid buffer aliasing.

7.4.3 Adding a RIFF Header

To make your IMA ADPCM file a RIFF / WAV file, you have to add a header before the actual data. Note that 2- and 4-byte values are little-endian (lowest byte first) in this format:

File Offset	Field Name	Size	Bytes	Description
0	ChunkID	4	"RIFF"	
4	ChunkSize	4	F0 F1 F2 F3	File size - 8
8	Format	4	"WAVE"	
12	SubChunk1ID	4	"fmt "	
16	SubChunk1Size	4	0x14 0x0 0x0 0x0	20
20	AudioFormat	2	0x11 0x0	0x11 for IMA ADPCM
22	NumOfChannels	2	0x1 0x0	Mono sound
24	SampleRate	4	R0 R1 R2 R3	0x1f40 for 8 kHz
28	ByteRate	4	B0 B1 B2 B3	0xfd7 for 8 kHz
32	BlockAlign	2	0x0 0x1	0x100
34	BitsPerSample	2	0x4 0x0	4-bit ADPCM
36	ByteExtraData	2	0x2 0x0	2
38	ExtraData	2	0xf9 0x1	Samples per block (505)
40	SubChunk2ID	4	"fact"	
44	SubChunk2Size	4	0x4 0x0 0x0 0x0	4
48	NumOfSamples	4	S0 S1 S2 S3	
52	SubChunk3ID	4	"data"	
56	SubChunk3Size	4	D0 D1 D2 D3	Data size (File Size-60)
60	Block1	256		First ADPCM block
316	...			More ADPCM data blocks

If we have n audio blocks, the values in the table are as follows:
$F = n \times 256 + 52$
$R = F_s$ (see Chapter 7.4.1 to see how to calculate F_s)
$B = \frac{F_s \times 256}{505}$
$S = n \times 505$. $D = n \times 256$

If you know beforehand how much you are going to record, you may fill in the complete header before any actual data. However, if you don't know how much you are going to record, you have to fill in the header size datas F, S and D after finishing recording.

The 128 words (256 bytes) of an ADPCM block are read from SCI_IN0 and written into file as follows. The high 8 bits of SCI_IN0 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.

A way to see if you have written the file in the right way is to check bytes 2 and 3 (the first byte counts as byte 0) of each 256-byte block. Byte 2 should always be less than 90, and byte 3 should always be zero.

7.4.4 Playing ADPCM Data

In order to play back your IMA ADPCM recordings, you have to have a file with a header as described in Chapter 7.4.3. If this is the case, all you need to do is to provide the ADPCM file to VS1103 as an ADPCM stream.

7.4.5 Sample Rate Considerations

VS10xx chips that support IMA ADPCM playback are capable of playing back ADPCM files with any sample rate. However, some other programs may expect IMA ADPCM files to have some exact sample rates, like 8000 or 11025 Hz. Also, some programs or systems do not support sample rates below 8000 Hz.

If SM_RECORD_PATH is set, recording is performed from the mixer output at exactly 44.1 kHz from the mixer output. If SM_RECORD_PATH is not set, recording is performed at $8kHz \times \frac{XTALI}{12.288MHz}$ from the microphone or line input. From the formula it can be seen that the nominal 8 kHz sample rate can only be obtained if XTALI = 12.288_MHz. Example: If you have a 12 MHz clock, you will get a sample rate of 7812.5 Hz, which should be recorded to the file.

7.4.6 Example Code

The following code initializes IMA ADPCM encoding on VS1103 and shows how to read the data.

```
const unsigned char header[] = {
  0x52, 0x49, 0x46, 0x46, 0x1c, 0x10, 0x00, 0x00,
  0x57, 0x41, 0x56, 0x45, 0x66, 0x6d, 0x74, 0x20, /*|RIFF....WAVEfmt |*/
  0x14, 0x00, 0x00, 0x00, 0x11, 0x00, 0x01, 0x00,
  0x40, 0x1f, 0x00, 0x00, 0x75, 0x12, 0x00, 0x00, /*|........@...u...|*/
  0x00, 0x01, 0x04, 0x00, 0x02, 0x00, 0xf9, 0x01,
  0x66, 0x61, 0x63, 0x74, 0x04, 0x00, 0x00, 0x00, /*|......ù.fact....|*/
  0x5c, 0x1f, 0x00, 0x00, 0x64, 0x61, 0x74, 0x61,
  0xe8, 0x0f, 0x00, 0x00
};

unsigned char db[512]; /* data buffer for saving to disk */
```

```
void RecordAdpcm1103(void) {  /* VS1003b/VS1023 */
  u_int16 w = 0, idx = 0, n = 0;
  s_int32 adpcmBlocks = -1;

  ...    /* Check and locate free space on disk */

  WriteMp3SpiReg(SCI_CLOCKF, 0xC3E8); /* 4.0x 12.288MHz */
  WaitForDreq();                      /* Wait for DREQ to go up again */
  WriteMp3SpiReg(SCI_MODE,   0x0804); /* Normal SW reset + other bits */
  WaitForDreq();                      /* Wait for DREQ to go up again */
  WriteMp3SpiReg(SCI_VOL,    0x1414); /* Recording monitor volume */
  WriteMp3SpiReg(SCI_BASS,        0); /* Bass/treble disabled */
  WriteMp3SpiReg(SCI_MIXERVOL, 0x8000U | (23<<10) | (23<<5) | (23));
  WriteMp3SpiReg(SCI_ADPCMRECCTL,25+20); /* Auto gain to max +20 dB */
  WriteMp3SpiReg(SCI_MODE,   0x08c8); /* Line in, SDI MIDI, SCI ADPCM, 8 kHz */

  for (idx=0; idx < sizeof(header); idx++)   /* Save header first */
    db[idx] = header[idx];
  db[24] = sampleRate;     /* Set sample rate */
  db[25] = samplRate>>8;

  /* Synchronize */
  do {
    n = 8 * ((ReadMp3SpiReg(SCI_IN1) >> 8) & 0xFF);
    Yield(1);              /* Give control to other processes for 1 ms */
  } while (n >= 480);      /* whole buffer size = 512 words */

  /* Record loop */
  while (recording_on) {
    while (idx < 512) {
      do {
        n = 8 * ((ReadMp3SpiReg(SCI_IN1) >> 8) & 0xFF);
        Yield(1);        /* Give control to other processes for 1 ms */
      } while (n < 16);  /* Only load data if >= 16 words available */

      while (n--) {
        w = ReadMp3SpiReg(SCI_IN0);
        db[idx++] = w>>8;
        db[idx++] = w&0xFF;
      }
    }
    idx = 0;
    write_block(datasector++, db); /* Write one disk block */
    adpcmBlocks+=2;                /* Disk block contains 2 adpcm blocks */
  }
  if (adpcmBlocks >= 0) {
    /* The previous algorithm will always write an unfinished ADPCM block.
       It doesn't matter as we consistently only tell of the data until
       the last completely written block. */
    dataSizeD     = adpcmBlocks*256;
    chunkSizeF    = dataSizeD+52:
    numOfSamplesS = adpcmBlocks*505;
    ...    /* Fix WAV header information  */
  }
}
```

7.5 SPI Boot

If GPIO0 is set with a pull-up resistor to 1 at boot time, VS1103b 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 addresses (i.e. at least 1 KiB). The serial speed used by VS1103b 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 Application Notes for VS10XX.

7.6 Play/Decode

This is the normal operation mode of VS1103b. MIDI and ADPCM are decoded, mixed and converted to analog domain by the internal DAC.

When there is no input for decoding, VS1103b goes into idle mode (lower power consumption than during decoding) and actively monitors the serial data input for valid data.

All different formats can be played back-to-back without software resets in-between. Send at least 4 zeros after each stream.

7.7 Feeding PCM data

VS1103b can be used as a PCM decoder by sending to it a WAV file header. If the length sent in the WAV file is 0xFFFFFFFF, VS1103b will stay in PCM mode for a long time (or until SARC_OUTOFADPCM has been set). 8-bit linear and 16-bit linear audio is supported in mono or stereo.

7.8 SDI Tests

There are several test modes in VS1103b, 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: VS1103b 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 4 zeros. The sequences are described below.

7.8.1 Sine Test

Sine test is initialized with the 8-byte sequence 0x53 0xEF 0x6E n 0 0 0 0, where n defines the sine test to use. n is defined as follows:

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

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

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

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

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

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

7.8.2 Pin Test

Pin test is activated with the 8-byte sequence 0x50 0xED 0x6E 0x54 0 0 0 0. This test is meant for chip production testing only.

7.8.3 Memory Test

Memory test mode is initialized with the 8-byte sequence 0x4D 0xEA 0x6D 0x54 0 0 0 0. After this sequence, wait for 500000 clock cycles. The result can be read from the SCI register SCI_IN0, and 'one' bits are interpreted as follows:

Bit(s)	Mask	Meaning
15	0x8000	Test finished
14:7		Unused
6	0x0040	Mux test succeeded
5	0x0020	Good I RAM
4	0x0010	Good Y RAM
3	0x0008	Good X RAM
2	0x0004	Good I ROM
1	0x0002	Good Y ROM
0	0x0001	Good X ROM
	0x807f	All ok

Memory tests overwrite the current contents of the RAM memories.

7.8.4 SCI Test

Sci test is initialized with the 8-byte sequence 0x53 0x70 0xEE n 0 0 0 0, where $n - 48$ is the register number to test. The content of the given register is read and copied to SCI_IN0. If the register to be tested is SCI_IN0, the result is copied to SCI_IN1.

Example: if n is 48, contents of SCI register 0 (SCI_MODE) is copied to SCI_IN0.

8 VS1103b Registers

8.1 Who Needs to Read This Chapter

User software is required when a user wishes to add some own functionality like DSP effects to VS1103b.

However, most users of VS1103b don't need to worry about writing their own code, or about this chapter, including those who only download software plugins from VLSI Solution's Web site.

8.2 The Processor Core

VS_DSP is a 16/32-bit DSP processor core that also had extensive all-purpose processor features. VLSI Solution's free VSKIT Software Package contains all the tools and documentation needed to write, simulate and debug Assembly Language or Extended ANSI C programs for the VS_DSP processor core. VLSI Solution also offers a full Integrated Development Environment VSIDE for full debug capabilities.

8.3 VS1103b Memory Map

VS1103b's Memory Map is shown in Figure 14.

8.4 SCI Registers

SCI registers described in Chapter 6.6 can be found here between 0xC000..0xC00F. In addition to these registers, there is one in address 0xC010, called SCI_CHANGE.

SCI registers, prefix SCI_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC010	r	0	CHANGE[5:0]	Last SCI access address.

SCI_CHANGE bits		
Name	**Bits**	**Description**
SCI_CH_WRITE	4	1 if last access was a write cycle.
SCI_CH_ADDR	3:0	SPI address of last access.

8.5 Serial Data Registers

SDI registers, prefix SER_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC011	r	0	DATA	Last received 2 bytes, big-endian.
0xC012	w	0	DREQ[0]	DREQ pin control.

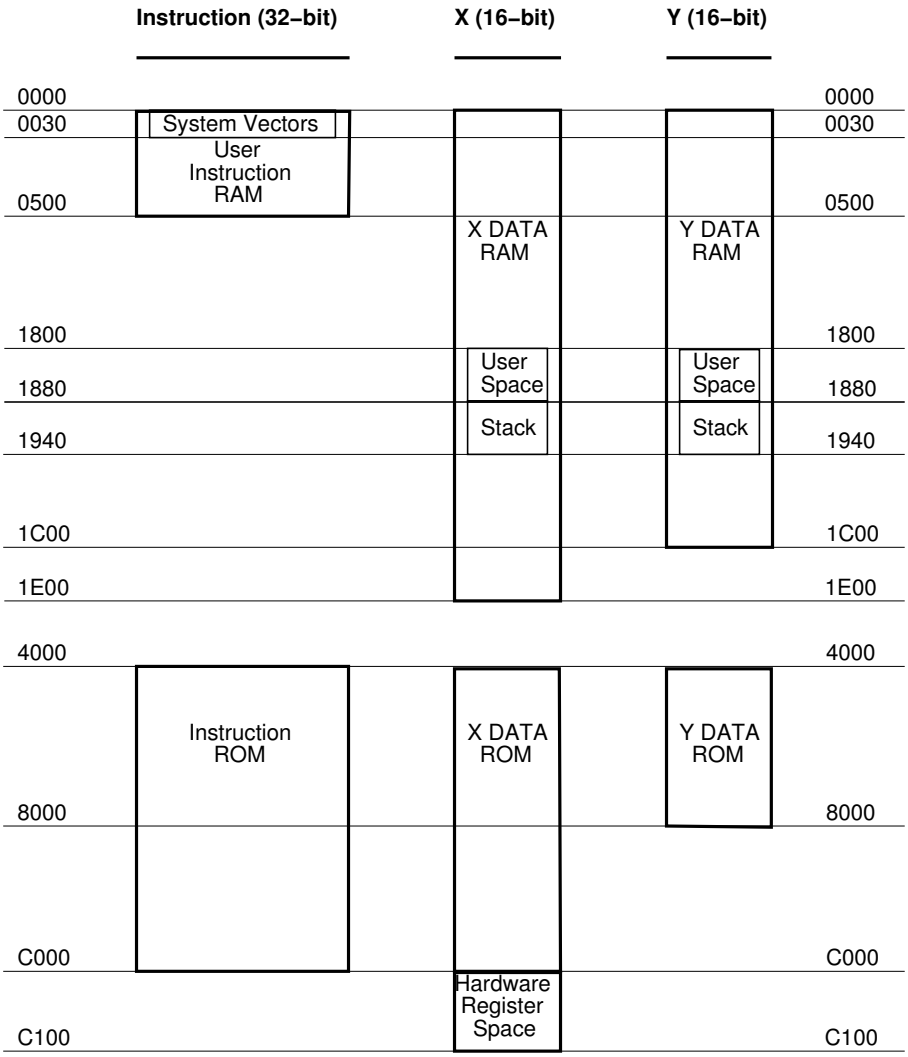


Figure 14: User's Memory Map.

8.6 DAC Registers

DAC registers, prefix DAC_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC013	rw	0	FCTLL	DAC frequency control, 16 LSbs.
0xC014	rw	0	FCTLH	DAC frequency control 4MSbs, PLL control.
0xC015	rw	0	LEFT	DAC left channel PCM value.
0xC016	rw	0	RIGHT	DAC right channel PCM value.

Every fourth clock cycle, an internal 26-bit counter is added to by (DAC_FCTLH & 15) $\times$ 65536 + DAC_FCTLL. Whenever this counter overflows, values from DAC_LEFT and DAC_RIGHT are read and a DAC interrupt is generated.

8.7 GPIO Registers

GPIO registers, prefix GPIO_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC017	rw	0	DDR[3:0]	Direction.
0xC018	r	0	IDATA[3:0]	Values read from the pins.
0xC019	rw	0	ODATA[3:0]	Values set to the pins.

GPIO_DIR is used to set the direction of the GPIO pins. 1 means output. GPIO_ODATA remembers its values even if a GPIO_DIR bit is set to input.

GPIO registers don't generate interrupts.

Note that in VS1103b the VSDSP registers can be read and written through the SCI_WRAMADDR and SCI_WRAM registers. You can thus use the GPIO pins quite conveniently.

8.8 Interrupt Registers

Interrupt registers, prefix INT_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC01A	rw	0	ENABLE[7:0]	Interrupt enable.
0xC01B	w	0	GLOB_DIS[-]	Write to add to interrupt counter.
0xC01C	w	0	GLOB_ENA[-]	Write to subtract from interript counter.
0xC01D	rw	0	COUNTER[4:0]	Interrupt counter.

INT_ENABLE controls the interrupts. The control bits are as follows:

INT_ENABLE bits		
Name	**Bits**	**Description**
INT_EN_TIM1	7	Enable Timer 1 interrupt.
INT_EN_TIM0	6	Enable Timer 0 interrupt.
INT_EN_RX	5	Enable UART RX interrupt.
INT_EN_TX	4	Enable UART TX interrupt.
INT_EN_MODU	3	Enable AD modulator interrupt.
INT_EN_SDI	2	Enable Data interrupt.
INT_EN_SCI	1	Enable SCI interrupt.
INT_EN_DAC	0	Enable DAC interrupt.

Note: It may take up to 6 clock cycles before changing INT_ENABLE has any effect.

Writing any value to INT_GLOB_DIS adds one to the interrupt counter INT_COUNTER and effectively disables all interrupts. It may take up to 6 clock cycles before writing to this register has any effect.

Writing any value to INT_GLOB_ENA subtracts one from the interrupt counter (unless INT_COUNTER already was 0). If the interrupt counter becomes zero, interrupts selected with INT_ENABLE are restored. An interrupt routine should always write to this register as the last thing it does, because interrupts automatically add one to the interrupt counter, but subtracting it back to its initial value is the responsibility of the user. It may take up to 6 clock cycles before writing this register has any effect.

By reading INT_COUNTER the user may check if the interrupt counter is correct or not. If the register is not 0, interrupts are disabled.

8.9 A/D Modulator Registers

Interrupt registers, prefix AD_				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0xC01E	rw	0	DIV	A/D Modulator divider.
0xC01F	rw	0	DATA	A/D Modulator data.

AD_DIV bits		
Name	**Bits**	**Description**
ADM_POWERDOWN	15	1 in powerdown.
ADM_DIVIDER	14:0	Divider.

ADM_DIVIDER controls the AD converter's sampling frequency. To gather one sample, $128 \times n$ clock cycles are used (n is value of AD_DIV). The lowest usable value is 4, which gives a 48 kHz sample rate when CLKI is 24.576 MHz. When ADM_POWERDOWN is 1, the A/D converter is turned off.

AD_DATA contains the latest decoded A/D value.

8.10 Watchdog

The watchdog consist of a watchdog counter and some logic. After reset, the watchdog is inactive. The counter reload value can be set by writing to WDOG_CONFIG. The watchdog is activated by writing 0x4ea9 to register WDOG_RESET. Every time this is done, the watchdog counter is reset. Every 65536'th clock cycle the counter is decremented by one. If the counter underflows, it will activate vsdsp's internal reset sequence.

Thus, after the first 0x4ea9 write to WDOG_RESET, subsequent writes to the same register with the same value must be made no less than every $65536 \times$WDOG_CONFIG clock cycles.

Once started, the watchdog cannot be turned off. Also, a write to WDOG_CONFIG doesn't change the counter reload value.

After watchdog has been activated, any read/write operation from/to WDOG_CONFIG or WDOG_DUMMY will invalidate the next write operation to WDOG_RESET. This will prevent runaway loops from resetting the counter, even if they do happen to write the correct number. Writing a wrong value to WDOG_RESET will also invalidate the next write to WDOG_RESET.

Reads from watchdog registers return undefined values.

8.10.1 Registers

Watchdog, prefix WDOG_				
Reg	**Type**	**Reset**	**Abbrev**	**Description**
0xC020	w	0	CONFIG	Configuration
0xC021	w	0	RESET	Clock configuration
0xC022	w	0	DUMMY[-]	Dummy register

8.11 UART (Universal Asynchronous Receiver / Transmitter)

The RS232 UART implements a serial interface using RS232 standard 8N1 (8 data bits, no parity, 1 stop bit).

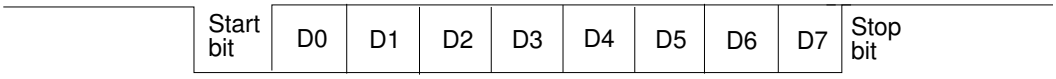


Figure 15: RS232 Serial Interface Protocol

When the line is idling, it stays in logic high state. When a byte is transmitted, the transmission begins with a start bit (logic zero) and continues with data bits (LSB first) and ends up with a stop bit (logic high). 10 bits are sent for each 8-bit byte frame.

8.11.1 Registers

UART registers, prefix UARTx_				
Reg	**Type**	**Reset**	**Abbrev**	**Description**
0xC028	r	0	STATUS[3:0]	Status
0xC029	r/w	0	DATA[7:0]	Data
0xC02A	r/w	0	DATAH[15:8]	Data High
0xC02B	r/w	0	DIV	Divider

8.11.2 Status UARTx_STATUS

A read from the status register returns the transmitter and receiver states.

UARTx_STATUS Bits		
Name	**Bits**	**Description**
UART_ST_RXORUN	3	Receiver overrun
UART_ST_RXFULL	2	Receiver data register full
UART_ST_TXFULL	1	Transmitter data register full
UART_ST_TXRUNNING	0	Transmitter running

UART_ST_RXORUN is set if a received byte overwrites unread data when it is transferred from the receiver shift register to the data register, otherwise it is cleared.

UART_ST_RXFULL is set if there is unread data in the data register.

UART_ST_TXFULL is set if a write to the data register is not allowed (data register full).

UART_ST_TXRUNNING is set if the transmitter shift register is in operation.

8.11.3 Data UARTx_DATA

A read from UARTx_DATA returns the received byte in bits 7:0, bits 15:8 are returned as '0'. If there is no more data to be read, the receiver data register full indicator will be cleared.

A receive interrupt will be generated when a byte is moved from the receiver shift register to the receiver data register.

A write to UARTx_DATA sets a byte for transmission. The data is taken from bits 7:0, other bits in the written value are ignored. If the transmitter is idle, the byte is immediately moved to the transmitter shift register, a transmit interrupt request is generated, and transmission is started. If the transmitter is busy, the UART_ST_TXFULL will be set and the byte remains in the transmitter data register until the previous byte has been sent and transmission can proceed.

8.11.4 Data High UARTx_DATAH

The same as UARTx_DATA, except that bits 15:8 are used.

8.11.5 Divider UARTx_DIV

UARTx_DIV Bits		
Name	**Bits**	**Description**
UART_DIV_D1	15:8	Divider 1 (0..255)
UART_DIV_D2	7:0	Divider 2 (6..255)

The divider is set to 0x0000 in reset. The ROM boot code must initialize it correctly depending on the master clock frequency to get the correct bit speed. The second divider (D_2) must be from 6 to 255.

The communication speed $f = \frac{f_m}{(D_1+1)\times(D_2)}$, where f_m is the master clock frequency, and f is the TX/RX speed in bps.

Divider values for common communication speeds at 26 MHz master clock:

Example UART Speeds, $f_m = 26MHz$		
Comm. Speed [bps]	UART_DIV_D1	UART_DIV_D2
4800	85	63
9600	42	63
14400	42	42
19200	51	26
28800	42	21
38400	25	26
57600	1	226
115200	0	226

8.11.6 Interrupts and Operation

Transmitter operates as follows: After an 8-bit word is written to the transmit data register it will be transmitted instantly if the transmitter is not busy transmitting the previous byte. When the transmission begins a TX_INTR interrupt will be sent. Status bit [1] informs the transmitter data register empty (or full state) and bit [0] informs the transmitter (shift register) empty state. A new word must not be written to transmitter data register if it is not empty (bit [1] = '0'). The

transmitter data register will be empty as soon as it is shifted to transmitter and the transmission is begun. It is safe to write a new word to transmitter data register every time a transmit interrupt is generated.

Receiver operates as follows: It samples the RX signal line and if it detects a high to low transition, a start bit is found. After this it samples each 8 bit at the middle of the bit time (using a constant timer), and fills the receiver (shift register) LSB first. Finally if a stop bit (logic high) is detected the data in the receiver is moved to the reveive data register and the RX_INTR interrupt is sent and a status bit[2] (receive data register full) is set, and status bit[2] old state is copied to bit[3] (receive data overrun). After that the receiver returns to idle state to wait for a new start bit. Status bit[2] is zeroed when the receiver data register is read.

RS232 communication speed is set using two clock dividers. The base clock is the processor master clock. Bits 15-8 in these registers are for first divider and bits 7-0 for second divider. RX sample frequency is the clock frequency that is input for the second divider.

8.12 Timers

There are two 32-bit timers that can be initialized and enabled independently of each other. If enabled, a timer initializes to its start value, written by a processor, and starts decrementing every clock cycle. When the value goes past zero, an interrupt is sent, and the timer initializes to the value in its start value register, and continues downcounting. A timer stays in that loop as long as it is enabled.

A timer has a 32-bit timer register for down counting and a 32-bit TIMER1_LH register for holding the timer start value written by the processor. Timers have also a 2-bit TIMER_ENA register. Each timer is enabled (1) or disabled (0) by a corresponding bit of the enable register.

8.12.1 Registers

Timer registers, prefix TIMER_				
Reg	**Type**	**Reset**	**Abbrev**	**Description**
0xC030	r/w	0	CONFIG[7:0]	Timer configuration
0xC031	r/w	0	ENABLE[1:0]	Timer enable
0xC034	r/w	0	T0L	Timer0 startvalue - LSBs
0xC035	r/w	0	T0H	Timer0 startvalue - MSBs
0xC036	r/w	0	T0CNTL	Timer0 counter - LSBs
0xC037	r/w	0	T0CNTH	Timer0 counter - MSBs
0xC038	r/w	0	T1L	Timer1 startvalue - LSBs
0xC039	r/w	0	T1H	Timer1 startvalue - MSBs
0xC03A	r/w	0	T1CNTL	Timer1 counter - LSBs
0xC03B	r/w	0	T1CNTH	Timer1 counter - MSBs

8.12.2 Configuration TIMER_CONFIG

TIMER_CONFIG Bits		
Name	**Bits**	**Description**
TIMER_CF_CLKDIV	7:0	Master clock divider

TIMER_CF_CLKDIV is the master clock divider for all timer clocks. The generated internal clock frequency $f_i = \frac{f_m}{c+1}$, where f_m is the master clock frequency and c is TIMER_CF_CLKDIV. Example: With a 12 MHz master clock, TIMER_CF_DIV=3 divides the master clock by 4, and the output/sampling clock would thus be $f_i = \frac{12MHz}{3+1} = 3MHz$.

8.12.3 Configuration TIMER_ENABLE

TIMER_ENABLE Bits		
Name	**Bits**	**Description**
TIMER_EN_T1	1	Enable timer 1
TIMER_EN_T0	0	Enable timer 0

8.12.4 Timer X Startvalue TIMER_Tx[L/H]

The 32-bit start value TIMER_Tx[L/H] sets the initial counter value when the timer is reset. The timer interrupt frequency $f_t = \frac{f_i}{c+1}$ where f_i is the master clock obtained with the clock divider (see Chapter 8.12.2 and c is TIMER_Tx[L/H].

Example: With a 12 MHz master clock and with TIMER_CF_CLKDIV=3, the master clock $f_i = 3MHz$. If TIMER_TH=0, TIMER_TL=99, then the timer interrupt frequency $f_t = \frac{3MHz}{99+1} = 30kHz$.

8.12.5 Timer X Counter TIMER_TxCNT[L/H]

TIMER_TxCNT[L/H] contains the current counter values. By reading this register pair, the user may get knowledge of how long it will take before the next timer interrupt. Also, by writing to this register, a one-shot different length timer interrupt delay may be realized.

8.12.6 Interrupts

Each timer has its own interrupt, which is asserted when the timer counter underflows.

8.13 System Vector Tags

The System Vector Tags are tags that may be replaced by the user to take control over several decoder functions.

8.13.1 AudioInt, 0x20

Normally contains the following VS_DSP assembly code:

```
jmpi DAC_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the first instruction with a $jmpi$ command to gain control over the audio interrupt.

8.13.2 SciInt, 0x21

Normally contains the following VS_DSP assembly code:

```
jmpi SCI_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the instruction with a $jmpi$ command to gain control over the SCI interrupt.

8.13.3 DataInt, 0x22

Normally contains the following VS_DSP assembly code:

```
jmpi SDI_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the instruction with a $jmpi$ command to gain control over the SDI interrupt.

8.13.4 ModuInt, 0x23

Normally contains the following VS_DSP assembly code:

```
jmpi MODU_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the instruction with a $jmpi$ command to gain control over the AD Modulator interrupt.

8.13.5 TxInt, 0x24

Normally contains the following VS_DSP assembly code:

```
jmpi EMPTY_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the instruction with a $jmpi$ command to gain control over the UART TX interrupt.

8.13.6 RxInt, 0x25

Normally contains the following VS_DSP assembly code:

```
jmpi RX_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the first instruction with a $jmpi$ command to gain control over the UART RX interrupt.

8.13.7 Timer0Int, 0x26

Normally contains the following VS_DSP assembly code:

```
jmpi EMPTY_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the first instruction with a $jmpi$ command to gain control over the Timer 0 interrupt.

8.13.8 Timer1Int, 0x27

Normally contains the following VS_DSP assembly code:

```
jmpi EMPTY_INT_ADDRESS,(i6)+1
```

The user may, at will, replace the first instruction with a $jmpi$ command to gain control over the Timer 1 interrupt.

8.13.9 UserCodec, 0x0

Normally contains the following VS_DSP assembly code:

```
jr
nop
```

If the user wants to take control away from the standard decoder, the first instruction should be replaced with an appropriate j command to user's own code.

The system usually activates the user program in less than 1 ms. After this, the user should steal interrupt vectors from the system, and insert user programs.

8.14 System Vector Functions

The System Vector Functions are pointers to some functions that the user may call to help implementing his own applications.

8.14.1 WriteIRam(), 0x2

VS_DSP C prototype:

void WriteIRam(register __i0 u_int16 *addr, register __a1 u_int16 msW, register __a0 u_int16 lsW);

This is the preferred way to write to the User Instruction RAM.

8.14.2 ReadIRam(), 0x4

VS_DSP C prototype:

u_int32 ReadIRam(register __i0 u_int16 *addr);

This is the preferred way to read from the User Instruction RAM.

A1 contains the MSBs and a0 the LSBs of the result.

8.14.3 DataBytes(), 0x6

VS_DSP C prototype:

u_int16 DataBytes(void);

If the user has taken over the normal operation of the system by switching the pointer in UserCodec to point to his own code, he may read data from the Data Interface through this and the following two functions.

This function returns the number of data bytes that can be read.

8.14.4 GetDataByte(), 0x8

VS_DSP C prototype:

u_int16 GetDataByte(void);

Reads and returns one data byte from the Data Interface. This function will wait until there is enough data in the input buffer.

8.14.5 GetDataWords(), 0xa

VS_DSP C prototype:

void GetDataWords(register __i0 __y u_int16 *d, register __a0 u_int16 n);

Read n data byte pairs and copy them in big-endian format (first byte to MSBs) to d. This function will wait until there is enough data in the input buffer.

8.14.6 Reboot(), 0xc

VS_DSP C prototype:

void Reboot(void);

Causes a software reboot, i.e. jump to the standard firmware without reinitializing the IRAM vectors.

This is NOT the same as the software reset function, which causes complete initialization.

9 Document Version Changes

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

Version 1.04, 2018-03-16

- Added mention of 8N1 format to Chapter 8.11, *UART (Universal Asynchronous Receiver / Transmitter)*.
- Updated text in Chapter 6.6.11, *SCI_VOL (RW)*.
- Updated Chapter 2, *Definitions*.
- Minor typo corrections.

Version 1.03, 2014-12-19

- Updated telephone number in Chapter 10, *Contact Information*.

Version 1.02 for VS1103b, 2014-08-14

- AVDD minimum fixed in Chapter 3.3, *Analog Characteristics*.
- Maximum SCI speed updated in Chapter 3.5, *Digital Characteristics*.
- Updated formatting.
- Other minor fixes.

Version 1.01 for VS1103b, 2007-09-03

- Corrected recording examples, Chapters 7.4.2 and 7.4.6.

Version 1.00 for VS1103b, 2007-08-22

- Removed preliminary status.

Version 0.4 for VS1103b, 2007-07-06

- Added SCI Multiple Write mode, Chapter 5.5.4.
- Corrected documentation for SCI_IN1 in Chapter 6.6.9.
- Added note to always run a software reset after setting SCI_CLOCKF to Chapter 6.6.4.

10 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