VS1002d – MP3 AUDIO CODEC

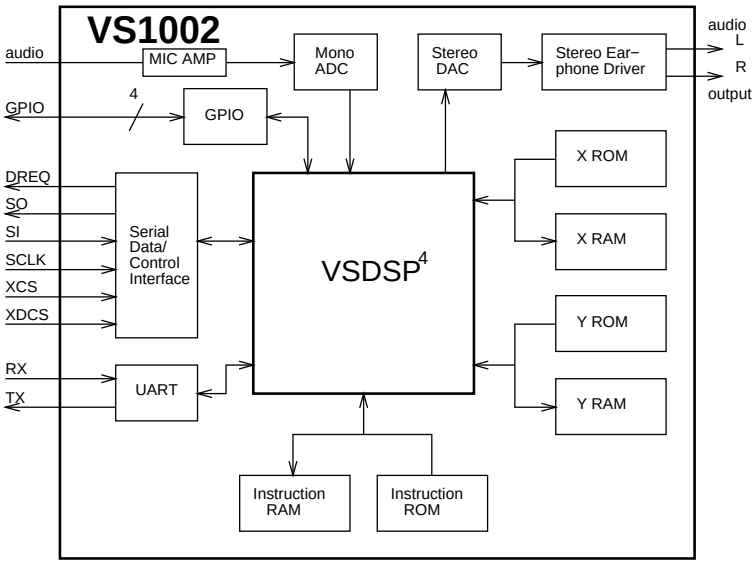
Features

- Decodes MPEG 1 & 2 audio layer 3 (ISO 11172-3), MP3+V, WAV and PCM files
- Support ADPCM recording and decoding
- Supports VBR (variable bitrate) for MP3
- Stream support
- Can be used as a slave co-processor
- Operates with single clock 12.288..14 MHz or 24.576..28 MHz.
- Low-power operation
- High-quality stereo DAC with no phase error between channels
- 16-bit adjustable On-chip A/D converter
- Stereo earphone driver capable of driving a 30Ω load
- Separate 2.5 .. 3.6V operating voltages for analog and digital
- 7.5 KiB On-chip RAM for user code / data
- Serial control and data interfaces
- SPI flash boot for special applications
- UART for debugging purposes
- New functions may be added with software and 4 GPIO pins

Description

VS1002d is a single-chip MP3 audio decoder. It contains a high-performance, low-power DSP processor core VS_DSP^4, working data memory, 5 KiB instruction RAM and 2.5 KiB data RAM for user applications, serial control and input data interfaces, 4 general purpose I/O pins, an UART, as well as a high-quality variable-sample-rate mono ADC and stereo DAC, followed by an earphone amplifier and a ground buffer.

VS1002d receives its input bitstream through a serial input bus, which it listens to as a system slave. The input stream is decoded and passed through a digital volume control to an 18-bit oversampling, multi-bit, sigma-delta DAC. The decoding is controlled via a serial control bus. In addition to the basic decoding, it is possible to add application specific features, like DSP effects, to the user RAM memory.



Contents

1 License **9**

2 Disclaimer **9**

3 Definitions **9**

4 Characteristics & Specifications **10**

4.1 Analog Characteristics 10

4.2 Power Consumption 10

4.3 DAC Interpolation Filter Characteristics 11

4.4 DAC Interpolation Filter Characteristics 11

4.5 Absolute Maximum Ratings 11

4.6 Recommended Operating Conditions 12

4.7 Digital Characteristics 12

4.8 Switching Characteristics - Clocks 13

4.9 Switching Characteristics - DREQ Signal 13

4.10 Switching Characteristics - SPI Interface Output 13

4.11 Switching Characteristics - Boot Initialization 13

5 Packages and Pin Descriptions **14**

5.1 Packages 14

5.1.1 LQFP-48 14

5.1.2 BGA-49 14

5.2 LQFP-48 and BGA-49 Pin Descriptions 15

6 Connection Diagram, LQFP-48 **16**

7 SPI Buses **17**

7.1 General . . . 17

7.2 SPI Bus Pin Descriptions . . . 17

7.2.1 VS1002 Native Modes (New Mode) . . . 17

7.2.2 VS1001 Compatibility Mode . . . 17

7.3 Serial Protocol for Serial Data Interface (SDI) . . . 18

7.3.1 General . . . 18

7.3.2 SDI in VS1002 Native Modes (New Mode) . . . 18

7.3.3 SDI in VS1001 Compatibility Mode . . . 18

7.3.4 SDI and DREQ . . . 18

7.4 Serial Protocol for Serial Command Interface (SCI) . . . 19

7.4.1 General . . . 19

7.4.2 SCI Read . . . 19

7.4.3 SCI Write . . . 19

7.5 SPI Timing Diagram . . . 20

7.6 SPI Examples with SM_SDINEW and SM_SDISHARED set . . . 22

7.6.1 Two SCI Writes . . . 22

7.6.2 Two SDI Bytes . . . 22

7.6.3 SCI Operation in Middle of Two SDI Bytes . . . 22

8 Functional Description **23**

8.1 Main Features . . . 23

8.2 Supported Audio Codecs . . . 23

8.2.1 Supported MP3 (MPEG layer 3) Formats . . . 23

8.2.2 Supported RIFF WAV Formats . . . 24

8.3 Data Flow of VS1002d 24
8.4 Serial Data Interface (SDI) 25
8.5 Serial Control Interface (SCI) 25
8.6 SCI Registers 26
8.6.1 SCI_MODE (RW) 27
8.6.2 SCI_STATUS (RW) 29
8.6.3 SCI_BASS (RW) 29
8.6.4 SCI_CLOCKF (RW) 29
8.6.5 SCI_DECODE_TIME (RW) 30
8.6.6 SCI_AUDATA (RW) 30
8.6.7 SCI_WRAM (RW) 30
8.6.8 SCI_WRAMADDR (RW) 31
8.6.9 SCI_HDAT0 and SCI_HDAT1 (R) 31
8.6.10 SCI_AIADDR (RW) 33
8.6.11 SCI_VOL (RW) 33
8.6.12 SCI_AICTRL[x] (RW) 33
8.7 Stereo Audio DAC 34

9 Operation 35

9.1 Clocking 35
9.2 Hardware Reset 35
9.3 Software Reset 35
9.4 SPI Boot 36
9.5 Play/Decode 36
9.6 Feeding PCM data 36

9.7 SDI Tests . . . 37
9.7.1 Sine Test . . . 37
9.7.2 Pin Test . . . 37
9.7.3 Memory Test . . . 38
9.7.4 SCI Test . . . 38

10 VS1002d Registers 39

10.1 Who Needs to Read This Chapter . . . 39
10.2 The Processor Core . . . 39
10.3 VS1002d Memory Map . . . 39
10.4 SCI Registers . . . 39
10.5 Serial Data Registers . . . 40
10.6 DAC Registers . . . 41
10.7 GPIO Registers . . . 41
10.8 Interrupt Registers . . . 42
10.9 A/D Modulator Registers . . . 43
10.10 Watchdog v1.0 2002-08-26 . . . 44
10.10.1 Registers . . . 44
10.11 UART v1.0 2002-04-23 . . . 45
10.11.1 Registers . . . 45
10.11.2 Status UARTx_STATUS . . . 45
10.11.3 Data UARTx_DATA . . . 46
10.11.4 Data High UARTx_DATAH . . . 46
10.11.5 Divider UARTx_DIV . . . 46
10.11.6 Interrupts and Operation . . . 47

10.12 Timers v1.0 2002-04-23 48
10.12.1 Registers 48
10.12.2 Configuration TIMER_CONFIG 48
10.12.3 Configuration TIMER_ENABLE 49
10.12.4 Timer X Startvalue TIMER_Tx[L/H] 49
10.12.5 Timer X Counter TIMER_TxCNT[L/H] 49
10.12.6 Interrupts 49
10.13 System Vector Tags 50
10.13.1 AudioInt, 0x20 50
10.13.2 SciInt, 0x21 50
10.13.3 DataInt, 0x22 50
10.13.4 ModuInt, 0x23 50
10.13.5 TxInt, 0x24 51
10.13.6 RxInt, 0x25 51
10.13.7 Timer0Int, 0x26 51
10.13.8 Timer1Int, 0x27 51
10.13.9 UserCodec, 0x0 52
10.14 System Vector Functions 52
10.14.1 WriteIRam(), 0x2 52
10.14.2 ReadIRam(), 0x4 52
10.14.3 DataBytes(), 0x6 53
10.14.4 GetDataByte(), 0x8 53
10.14.5 GetDataWords(), 0xa 53
10.14.6 Reboot(), 0xc 53

11 VS1002 Version Changes **54**

11.1 Changes Between VS1002c and VS1002d, 2004-05-13 54

12 Document Version Changes **55**

12.1 Version 0.71 for VS1002d, 2004-07-20 55

12.2 Version 0.70 for VS1002d, 2004-05-13 55

12.3 Version 0.62 for VS1002c, 2004-03-24 55

12.4 Version 0.61 for VS1002c, 2004-03-11 55

12.5 Version 0.6 for VS1002c, 2004-02-13 55

12.6 Version 0.5 for VS1002c, 2004-01-15 55

12.7 Version 0.4 for VS1002c, 2003-12-23 55

13 Contact Information **56**

List of Figures

1 Pin Configuration, LQFP-48. 14
2 Pin Configuration, BGA-49. 14
3 Typical Connection Diagram Using LQFP-48. 16
4 BSYNC Signal. 18
5 SCI Word Read 20
6 SCI Word Write 20
7 SPI Timing Diagram. 20
8 Two SCI Operations. 22
9 Two SDI Bytes. 22
10 Two SDI Bytes Separated By an SCI Operation. 22
11 Data Flow of VS1002d. 24
12 ADPCM Frequency Responses. 28
13 User’s Memory Map. 40
14 RS232 Serial Interface Protocol 45

1 License

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

2 Disclaimer

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

3 Definitions

ASIC Application Specific Integrated Circuit.

B Byte, 8 bits.

b Bit.

IC Integrated Circuit.

Ki "Kibi" = 2^{10} = 1024 (IEC 60027-2).

Mi "Mebi" = 2^{20} = 1048576 (IEC 60027-2).

VS_DSP VLSI Solution's DSP core.

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

4 Characteristics & Specifications

Unless otherwise noted: AVDD=2.9..3.6V, DVDD=2.5..3.6V, TA=-30..+85°C, XTALI=24.576MHz, Full-Scale Output Sinewave at 1.526 kHz, measurement bandwidth 20..20000 Hz, analog output load 30Ω (no ground buffer) or 100Ω (with ground buffer), bitstream 128 kbit/s, local components as shown in Figure 3.

Note, that some analog values are in practice better than in these tables if chips are used within a limited temperature range and not too close to lower voltage limits.

4.1 Analog Characteristics

Parameter	Symbol	Min	Typ	Max	Unit
DAC Resolution			16		bits
Total Harmonic Distortion	THD		0.1	0.2	%
Dynamic Range (DAC unmuted, A-weighted)	IDR		90		dB
S/N Ratio (full scale signal)	SNR	70	87		dB
Interchannel Isolation		50	75		dB
Interchannel Gain Mismatch		-0.5		0.5	dB
Frequency Response		-0.1		0.1	dB
Frequency Response, AVDD = 2.8V		-0.3		0.3	dB
Full Scale Output Voltage (Peak-to-peak)		1.4	1.8[1]	2.0	Vpp
Deviation from Linear Phase				5	°
Out of Band Energy			-60		dB
Out of Band Energy with Analog Filter			-90		dB
Analog Output Load Resistance, no ground buffer	AOLR1	16	30[2]		Ω
Analog Output Load Resistance, ground buffer	AOLR2	16	100[2]		Ω
Analog Output Load Capacitance				1000	pF
Microphone input amplifier gain	MICG		20		dB

[1] 3.6 volts can be achieved with +-to-+ wiring for mono difference sound.

[2] AOLR1/2 may be much lower, but below *Typical* distortion performance may be compromised.

4.2 Power Consumption

Parameter	Symbol	Min	Typ	Max	Unit
Power Supply Rejection			40		dB
Power Supply Consumption AVDD, Reset			0.6	5.0	μA
Power Supply Consumption AVDD, no load		3.0	4.5	6.0	mA
Power Supply Consumption AVDD, output loaded at 30Ω		4.0	5.5	40.0	mA
Power Supply Consumption AVDD, o. @ 30Ω + GND-buf.		6.0	7.5	40.0	mA
Power Supply Consumption DVDD, Reset			3.7	10.0	μA
Power Supply Consumption DVDD			15.0		mA

4.3 DAC Interpolation Filter Characteristics

Parameter	Symbol	Min	Typ	Max	Unit
Passband (to -3dB corner)		0		0.459Fs	Hz
Passband (Ripple Spec)		0		0.420Fs	Hz
Passband Ripple				±0.056	dB
Transition Band		0.420Fs		0.580Fs	Hz
Stop Band		0.580Fs			Hz
Stop Band Rejection		90			dB
Group Delay			15/Fs		s

Fs is conversion frequency

4.4 DAC Interpolation Filter Characteristics

Parameter	Symbol	Min	Typ	Max	Unit
-3 dB bandwidth		300			kHz
Passband Response at 20 kHz		-0.05			dB

4.5 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Analog Positive Supply	AVDD	-0.3	3.6	V
Digital Positive Supply	DVDD	-0.3	3.6	V
Current at Any Digital Output			±50	mA
Voltage at Any Digital Input		DGND-1.0	DVDD+1.0	V
Operating Temperature		-30	+85	°C
Functional Operating Temperature		-40	+95	°C
Storage Temperature		-65	+150	°C

4.6 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Analog and Digital Ground	AGND DGND		0.0		V
Positive Analog	AVDD	2.5[1]	3.0	3.6	V
Ambient Operating Temperature		-30		+85	°C

[1] If AVDD is below 2.8 V, distortion performance may be compromised.

The following values are to be used when the clock doubler is active:

Parameter	Symbol	Min	Typ	Max	Unit
Positive Digital	DVDD	2.5	2.7	3.6	V
Input Clock Frequency	XTALI		12.288	13	MHz
Internal Clock Frequency[1]	CLKI		24.576	26	MHz

[1] The maximum sample rate that may be played with correct speed is CLKI/512.

The following values are to be used when the clock doubler is not active:

Parameter	Symbol	Min	Typ	Max	Unit
Positive Digital	DVDD	2.5	2.7	3.6	V
Input Clock Frequency	XTALI		24.576	26	MHz
Internal Clock Frequency[1]	CLKI		24.576	26	MHz

[1] The maximum sample rate that may be decoded with correct speed is CLKI/512.

Note: With higher than typical voltages, VS1002d may operate with CLKI upto 30..32 MHz. However, the chips are not qualified for this kind of usage. If necessary, VLSI Solution Oy can qualify chips for higher clock rates for quantity orders.

4.7 Digital Characteristics

Parameter	Symbol	Min	Typ	Max	Unit
High-Level Input Voltage		0.7DVDD			V
Low-Level Input Voltage				0.3DVDD	V
High-Level Output Voltage at I_O = -2.0 mA		0.7DVDD			V
Low-Level Output Voltage at I_O = 2.0 mA				0.3DVDD	V
Input Leakage Current				1.0	μA

4.8 Switching Characteristics - Clocks

Parameter	Symbol	Min	Typ	Max	Unit
Master Clock Frequency [1]	XTALI		12.288		MHz
Master Clock Frequency [2]	XTALI		24.576		MHz
Master Clock Duty Cycle		40	50	60	%
Clock Output	XTALO		XTALI		MHz

[1] Clock doubler active.
[2] Clock doubler not active.

4.9 Switching Characteristics - DREQ Signal

Parameter	Symbol	Min	Typ	Max	Unit
Data Request Signal	DREQ			200	ns

4.10 Switching Characteristics - SPI Interface Output

Parameter	Symbol	Min	Typ	Max	Unit
SPI Input Clock Frequency				$\frac{CLKI}{6}$	MHz
Rise time for SO				25	ns

Note: Maximum load for SO is 100 pF.

4.11 Switching Characteristics - Boot Initialization

Parameter	Symbol	Min	Max	Unit
_RESET active time		2		XTALI
_RESET inactive to software ready			50000	XTALI

5 Packages and Pin Descriptions

5.1 Packages

5.1.1 LQFP-48

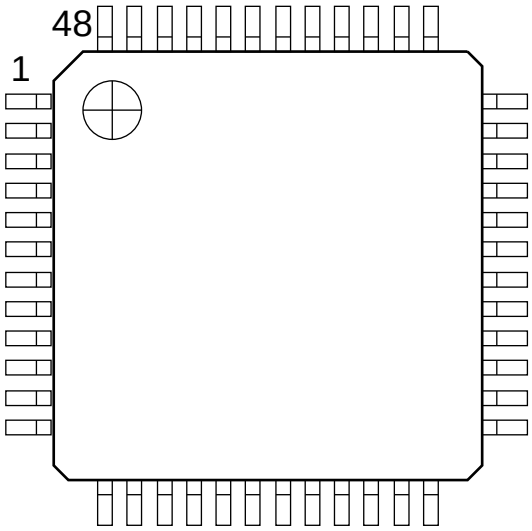


Figure 1: Pin Configuration, LQFP-48.

LQFP-48 package dimensions are at *http://www.vlsi.fi/vs1001/lqfp48.pdf* .

5.1.2 BGA-49

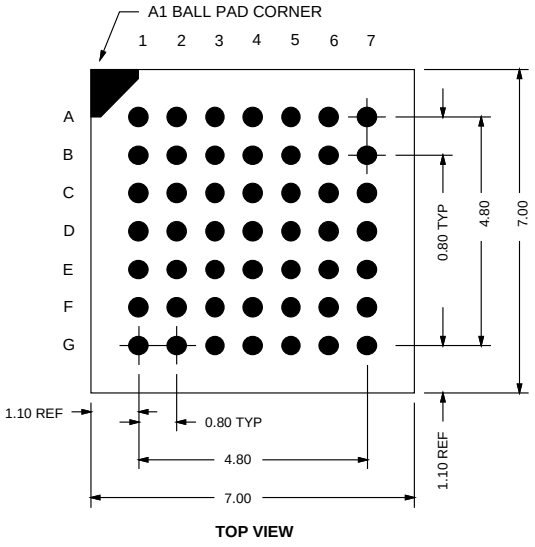


Figure 2: Pin Configuration, BGA-49.

BGA-49 package dimensions are at *http://www.vlsi.fi/vs1001/bga49.pdf* .

5.2 LQFP-48 and BGA-49 Pin Descriptions

Pin Name	LQFP-48 Pin	BGA49 Ball	Pin Type	Function
MICP	1	C3	AI	microphone input
MICN	2	C2	AI	microphone input
XRESET	3	B1	DI	active low asynchronous reset
DGND0	4	D2	PWR	digital ground
DVDD0	6	D3	PWR	digital power supply
DREQ	8	E2	DO	data request, input bus
GPIO2[2] / DCLK[1]	9	E1	DI	general purpose IO 2 / serial input data bus clock
GPIO3[2] / SDATA[1]	10	F2	DI	general purpose IO 3 / serial data input
XDCS / BSYNC[1]	13	E3	DI	data chip select / byte sync, connect to DVDD if not used
DVDD1	14	F3	PWR	digital power supply
DGND1	16	F4	PWR	digital ground
XTALO	17	G3	AO	crystal output
XTALI	18	E4	AI	crystal input
DVDD2	19	F5	PWR	digital power supply
DGND2	20	F6	PWR	digital ground (in BGA-49, DGND2, 3, 4 conn. together)
DGND3	21	F6	PWR	digital ground
DGND4	22	F6	PWR	digital ground
XCS	23	G6	DI	chip select input (active low)
RX	26	E6	DI	UART receive
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
TEST	32	C6	DI	reserved for test, connect to DVDD
GPIO0[3]	33	C7	DIO	general purpose IO 0, use 100 kΩ pull-down resistor
GPIO1[2]	34	B6	DIO	general purpose IO 1
AGND0	37	C5	PWR	analog ground, low-noise reference
AVDD0	38	B5	PWR	analog power supply
RIGHT	39	A6	AO	right channel output
AGND1	40	B4	PWR	analog ground
AGND2	41	A5	PWR	analog ground
GBUF	42	C4	AO	ground buffer
AVDD1	43	A4	PWR	analog power supply
RCAP	44	B3	AIO	filtering capacitance for reference
AVDD2	45	A3	PWR	analog power supply
LEFT	46	B2	AO	left channel output
AGND3	47	A2	PWR	analog ground

[1] First pin function is active in New Mode, latter in Compatibility Mode.
[2] If not used, use 100 kΩ pull-down resistor.
[3] Unless pull-down resistor is used, SPI Boot is tried. See Chapter 9.4 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

Type	Description
AI	Analog input
AO	Analog output
AIO	Analog input/output
PWR	Power supply pin

In BGA-49, no-connect balls are C1, D1, F1, G1, G2, G4, G5, G7, E5, D7, B7, A7, A1.
In LQFP-48, no-connect pins are 5, 7, 11, 12, 15, 24, 25, 31, 35, 36, 48.

6 Connection Diagram, LQFP-48

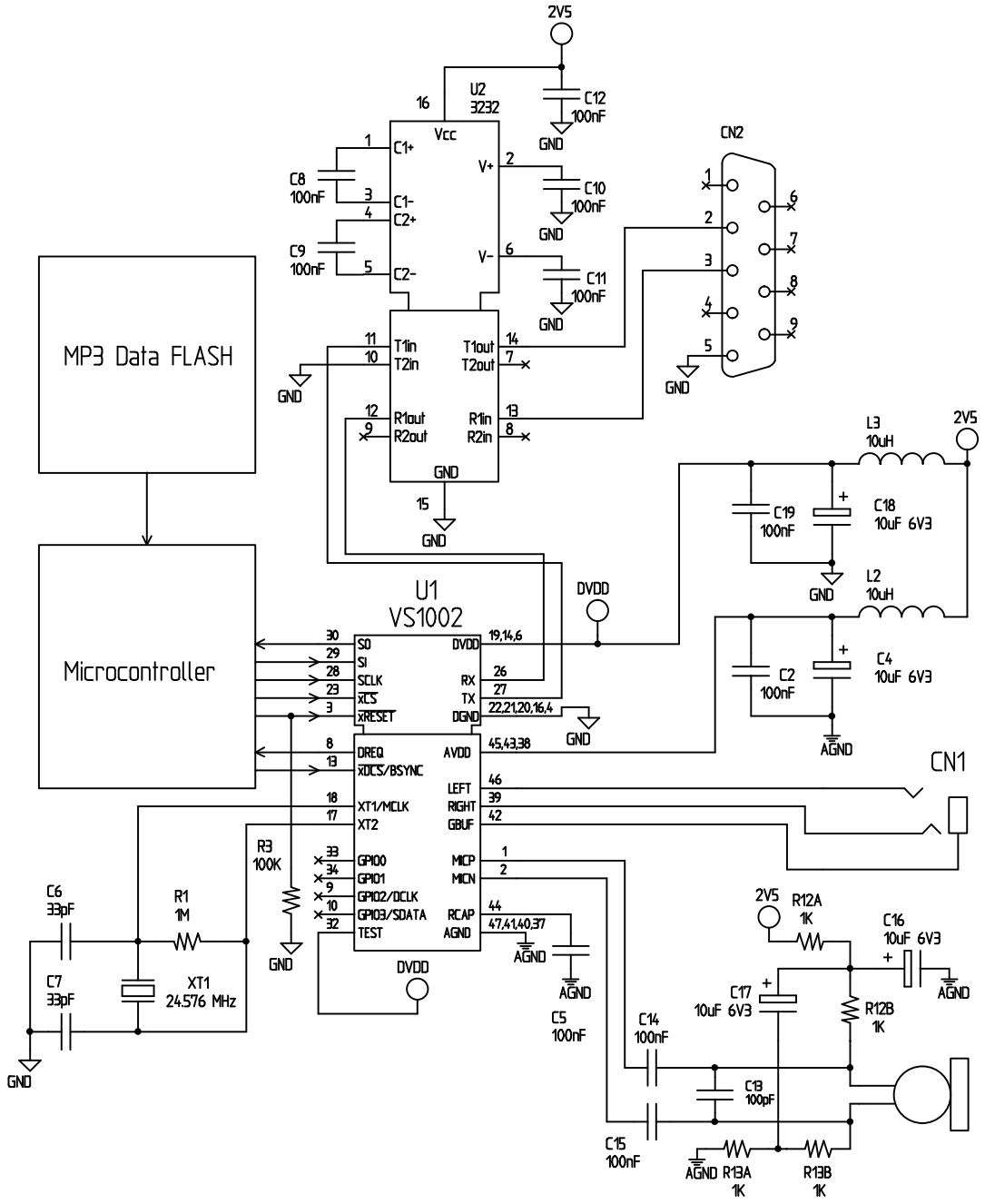


Figure 3: Typical Connection Diagram Using LQFP-48.

The ground buffer GBUF can be used for common voltage (1.37 V) for earphones. This will eliminate the need for large isolation capacitors on line outputs, and thus the audio output pins from VS1002d may be connected directly to the earphone connector.

If GBUF is not used, LEFT and RIGHT must be provided with 100 μF capacitors.

If UART is not used, RX should connect to ground and TX be unconnected. U2 and CN2 are not needed.

Note: This connection assumes SM_SDINEW is active (see Chapter 8.6.1). If also SM_SDISHARE is used, xDCS doesn't need to be connected (see Chapter 7.2.1).

7 SPI Buses

7.1 General

The SPI Bus - that was originally used in some Motorola devices - has been used for both VS1002d's Serial Data Interface SDI (Chapters 7.3 and 8.4) and Serial Control Interface SCI (Chapters 7.4 and 8.5).

7.2 SPI Bus Pin Descriptions

7.2.1 VS1002 Native Modes (New Mode)

These modes are active 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.

7.2.2 VS1001 Compatibility Mode

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. There is no chip select for SDI, which is always active.
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.

7.3 Serial Protocol for Serial Data Interface (SDI)

7.3.1 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 8.6).

VS1002d assumes its data input to be byte-sychronized. SDI bytes may be transmitted either MSb or LSb first, depending of contents of SCI_MODE (Chapter 8.6.1).

7.3.2 SDI in VS1002 Native Modes (New Mode)

In VS1002 native modes, byte synchronization is achieved by XDCS (or XCS if SM_SDISHARE is 1). The state of XDCS (or XCS) 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 VS1002d, it is recommended to turn XDCS (or XCS) every now and then, for instance once after every flash data block or a few kilobytes, just to keep sure the host and VS1002d are in sync.

For new designs, using VS1002 native modes are recommended, as they are easier to implement than BSYNC generation.

7.3.3 SDI in VS1001 Compatibility Mode

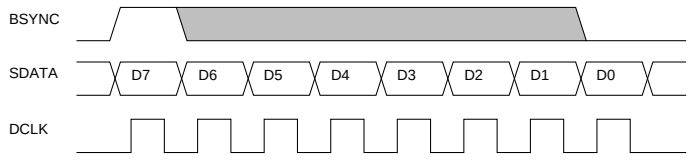


Figure 4: BSYNC Signal.

When VS1002 is running in VS1001 compatibility mode, a BSYNC signal must be generated to ensure correct byte-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).

7.3.4 SDI and DREQ

The DREQ signal of the data interface is used in slave mode to signal if VS1002d's FIFO is capable of receiving more input data. If DREQ is high, VS1002d can take at least 32 bytes of data. When there is less than 32 bytes of free space, DREQ is turned low, and the sender should stop transferring new data.

Because of the 32-byte safety area, the sender may send upto 32 bytes of data at a time without checking the status of DREQ, making controlling VS1002d 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. It should not abort a byte transmission that has already started.

7.4 Serial Protocol for Serial Command Interface (SCI)

7.4.1 General

The serial bus protocol for the Serial Command Interface SCI (Chapter 8.5) 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 send MSb firrst.

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: After sending an SCI command, it is not allowed to send SCI or SDI data for 5 microseconds.

7.4.2 SCI Read

VS1002d registers are read by 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. 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.

7.4.3 SCI Write

VS1002d registers are written to 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.

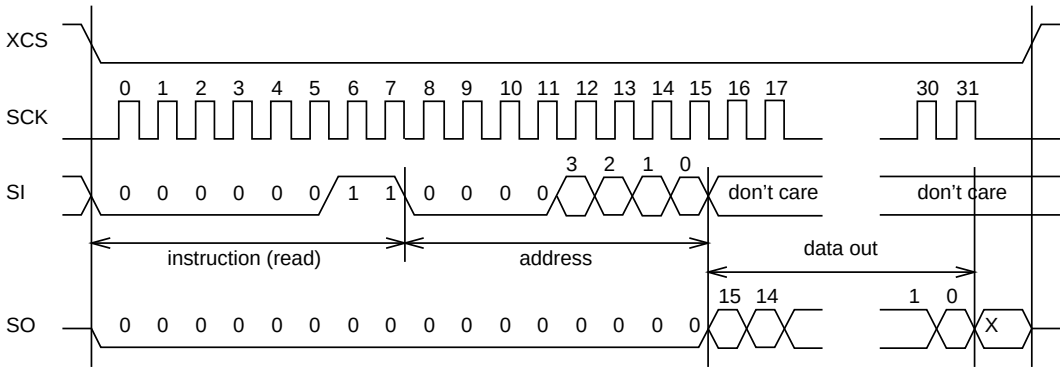


Figure 5: SCI Word Read

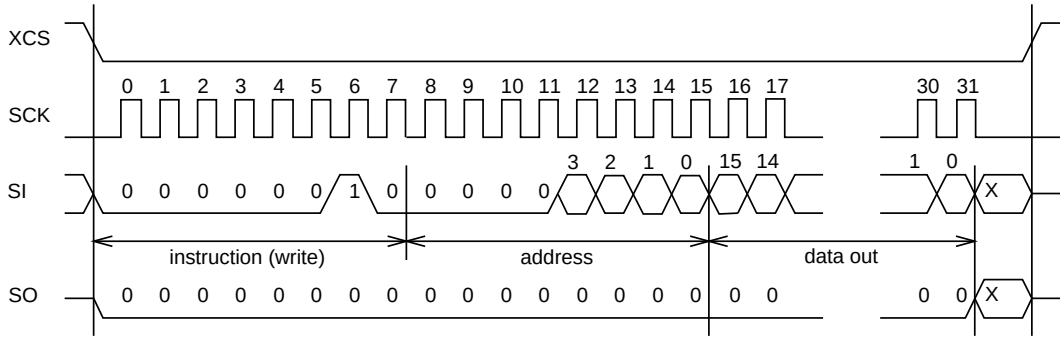


Figure 6: SCI Word Write

7.5 SPI Timing Diagram

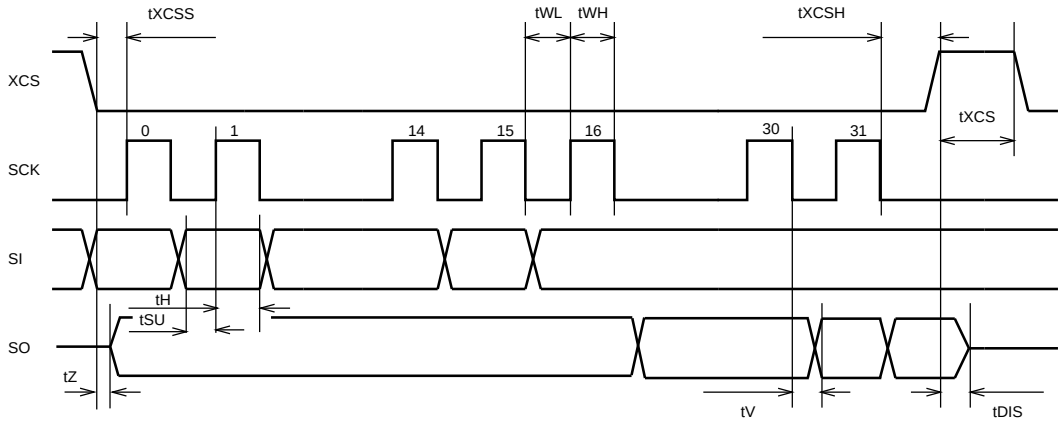


Figure 7: SPI Timing Diagram.

Symbol	Min	Max	Unit
tXCSS	5		ns
tSU	-26		ns
tH	2		XTALI cycles
tZ	0		ns
tWL	2		XTALI cycles
tWH	2		XTALI cycles
tV		2 (+ 25ns[1])	XTALI cycles
tXCSH	-26		ns
tXCS	2		XTALI 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 is 1/6 of VS1011's external clock speed XTALI. Slightly higher speed can be achieved with very careful timing tuning. For details, see Application Notes for VS10XX.

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

7.6 SPI Examples with SM_SDINEW and SM_SDISHARED set

7.6.1 Two SCI Writes

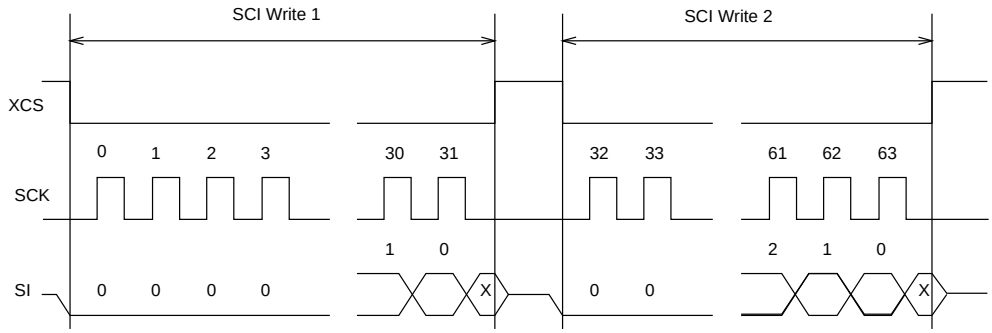


Figure 8: Two SCI Operations.

Figure 8 shows two consecutive SCI operations. Note that xCS *must* be raised to inactive state between the writes.

7.6.2 Two SDI Bytes

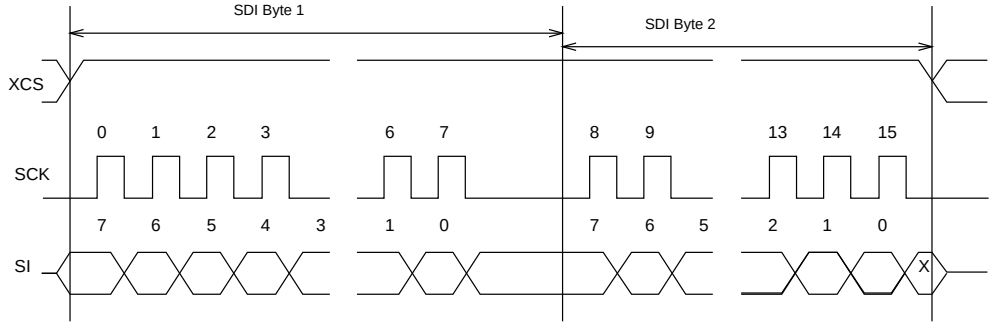


Figure 9: Two SDI Bytes.

SDI data is synchronized with a raising edge of xCS as shown in Figure 9. However, every byte doesn't need separate synchronization.

7.6.3 SCI Operation in Middle of Two SDI Bytes

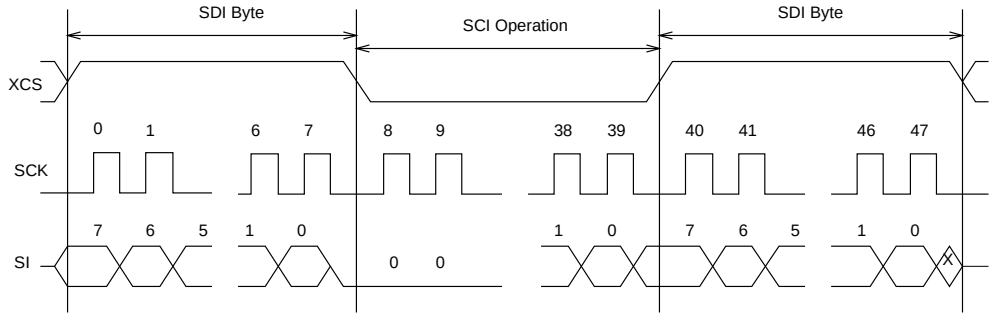


Figure 10: Two SDI Bytes Separated By an SCI Operation.

Figure 10 shows how an SCI operation is embedded in between SDI operations. The changes in xCS are used to synchronize both SDI and SCI.

8 Functional Description

8.1 Main Features

VS1002d is based on a proprietary digital signal processor, VS_DSP. It contains all the code and data memory needed for MP3 and WAV PCM + ADPCM audio decoding, 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.

VS1002d can play all MPEG 1 and 2 layer III files, with all sample rates and bitrates, including variable bitrate (VBR).

8.2 Supported Audio Codecs

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

8.2.1 Supported MP3 (MPEG layer 3) Formats

MPEG 1.0[1]:

Samplerate / Hz	Bitrate / kbit/s													
	32	40	48	56	64	80	96	112	128	160	192	224	256	320
48000	+	+	+	+	+	+	+	+	+	+	+	+	+[3]	+[3]
44100	+	+	+	+	+	+	+	+	+	+	+	+	+	+[3]
32000	+	+	+	+	+	+	+	+	+	+	+	+	+	+

MPEG 2.0[1]:

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

MPEG 2.5[1] [2]:

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

[1] Also all variable bitrate (VBR) formats are supported.
[2] Incompatibilities may occur because MPEG 2.5 is not a standard format.
[3] Nominal CLKI=24.576 MHz may be too little for glitchless playback.

8.2.2 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 $\leq$ 48kHz
0x02	ADPCM	-	
0x03	IEEE_FLOAT	-	
0x06	ALAW	-	
0x07	MULAW	-	
0x10	OKI_ADPCM	-	
0x11	IMA_ADPCM	+	Any sample rate $\leq$ 48kHz, mono only
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	+	For supported MP3 modes, see Chapter 8.2.1
0x64	G726_ADPCM	-	
0x65	G722_ADPCM	-	

8.3 Data Flow of VS1002d

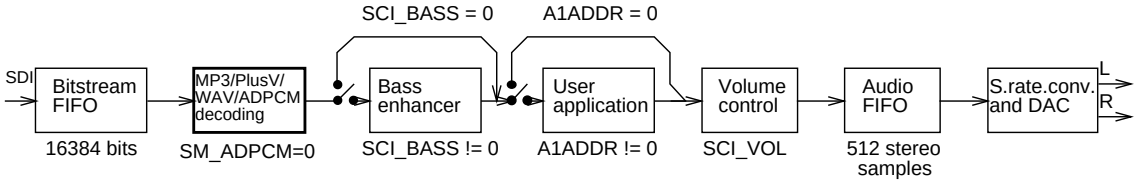


Figure 11: Data Flow of VS1002d.

First, depending on the audio data, and provided ADPCM encoding mode is not set, MP3, PCM WAV or mono IMA ADPCM WAV data is received and decoded from the SDI bus.

After decoding, data may be sent to the Bass Enhancer depending on SCI_BASS.

Then, 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.

After the optional user application, 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 10.13.1) and fed to the sample rate converter and DACs. The size of the audio FIFO is 512 stereo (2$\times$16-bit) samples.

The sample rate converter converts all different sample rates to CLKI/512 and feeds the data to the DAC,

which in order creates a stereo in-phase analog signal. This signal is then forwarded to the earphone amplifier.

8.4 Serial Data Interface (SDI)

The serial data interface is meant for transferring compressed MP3 audio data as well as WAV PCM and ADPCM data.

Also several different tests may be activated through SDI as described in Chapter 9.

8.5 Serial Control Interface (SCI)

The serial control interface is compatible with the SPI bus specification. Data transfers are always 16 bits. VS1002d is controlled by writing and reading the registers of the interface.

The main controls of the control interface are:

- control of the operation mode
- uploading user programs
- access to header data
- status information
- access to encoded digital data
- feeding input data

8.6 SCI Registers

SCI registers, prefix SCI_, offset 0xC000				
Reg	**Type**	**Reset**	**Abbrev[bits]**	**Description**
0x0	rw	0x800	MODE	Mode control.
0x1	rw	0x2C[1]	STATUS	Status of VS1002d.
0x2	rw	0	BASS	Built-in bass enhancer.
0x3	rw	0	CLOCKF	Clock freq + doubler.
0x4	r	0	DECODE_TIME	Decode time in seconds.
0x5	rw	0	AUDATA	Misc. audio data.
0x6	rw	0	WRAM	RAM write.
0x7	rw	0	WRAMADDR	Base address for RAM write.
0x8	r	0	HDAT0	Stream header data 0.
0x9	r	0	HDAT1	Stream header data 1.
0xA	rw	0	AIADDR	Start address of application.
0xB	rw	0	VOL	Volume control.
0xC	rw	0	AICTRL0	Application control register 0.
0xD	rw	0	AICTRL1	Application control register 1.
0xE	rw	0	AICTRL2	Application control register 2.
0xF	rw	0	AICTRL3	Application control register 3.

[1] Firmware changes the value of this register immediately to 0x28, and in less than 100 ms to 0x20.

8.6.1 SCI_MODE (RW)

SCI_MODE is used to control operation of VS1002d. Note that this register is not reset to 0, but to 0x0800 (i.e. SM_SDINEW is set).

Bit	Name	Function	Value	Description
0	SM_DIFF	Differential	0 1	normal in-phase audio left channel inverted
1	SM_SETTOZERO	Set to zero	0 1	right wrong
2	SM_RESET	Soft reset	0 1	no reset reset
3	SM_OUTOFWAV	Jump out of WAV decoding	0 1	no yes
4	SM_PDOWN	Powerdown	0 1	power on powerdown
5	SM_TESTS	Allow SDI tests	0 1	not allowed allowed
6	SM_STREAM	Stream mode	0 1	no yes
7	SM_PLUSV	MP3+V active	0 1	no yes
8	SM_DACT	DCLK active edge	0 1	rising falling
9	SM_SDIORD	SDI bit order	0 1	MSb first MSb last
10	SM_SDISHARE	Share SPI chip select	0 1	no yes
11	SM_SDINEW	VS1002 native SPI modes	0 1	no yes
12	SM_ADPCM	ADPCM recording active	0 1	no yes
13	SM_ADPCM_HP	ADPCM high-pass filter active	0 1	no yes

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

By setting SM_RESET to 1, the player is software reset. This bit clears automatically.

When the user decoding a WAV file wants to get out of the file without playing it to the end, set SM_OUTOFWAV, and send zeros to VS1002d until SM_OUTOFWAV is again zero. If the user doesn't want to check SM_OUTOFWAV, send 128 zeros.

Bit SM_PDOWN sets VS1002d into software powerdown mode. Note that 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 9.7.

SM_STREAM activates VS1002d's stream mode. In this mode, data should be sent with as even intervals as possible (and preferable with data blocks of less than 512 bytes), and VS1002d makes every attempt to keep its input buffer half full by changing its playback speed upto 5%. For best quality sound, the average speed error should be within 0.5%, the bitrate should not exceed 160 kbit/s and VBR should not be used. For details, see Application Notes for VS10XX.

SM_PLUSV activates MP3+V decoding. Without this bit set, only MP3 decoding is performed even for files with additional PlusV data.

SM_DACT defines the active edge of data clock for SDI. If clear data is read at the rising edge, and if set 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 7.2, if also SM_SDINEW is set.

Setting SM_SDINEW will activate VS1002 native serial modes as described in Chapters 7.2.1 and 7.3.2. Note, that this bit is set as a default when VS1002d is started up.

By activating SM_ADPCM and SM_RESET at the same time, the user will activate IMA ADPCM recording mode. More information is available in document Application Notes for VS10XX.

If SM_ADPCM_HP is set at the same time as SM_ADPCM and SM_RESET, ADPCM mode will start with a high-pass filter. This may help intelligibility of speech when there is lots of background noise. The difference created to the ADPCM encoder frequency response is as shown in Figure 12.

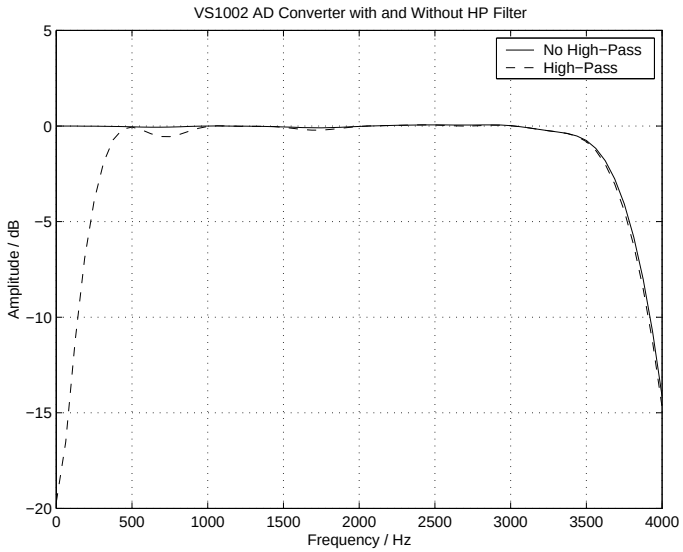


Figure 12: ADPCM Frequency Responses.

8.6.2 SCI_STATUS (RW)

SCI_STATUS contains information on the current status of VS1002d and lets the user shutdown the chip without audio glitches.

Name	Bits	Description
SS_VER	6..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 and 3 for vs1003.

SS_APDOWN2 controls analog driver powerdown. Normally this bit is controlled by the system firmware. However, if the user wants to powerdown VS1002d 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.

8.6.3 SCI_BASS (RW)

Name	Bits	Description
SB_AMPLITUDE	7..4	Enhancement in 1 dB steps (0..15)
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 set to 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.

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.

8.6.4 SCI_CLOCKF (RW)

SCI_CLOCKF is used to tell if the input clock XTALI is running at something else than 24.576 MHz. XTALI is set in 2 kHz steps. Thus, the formula for calculating the correct value for this register is $\frac{XTALI}{2000}$ (XTALI is in Hz). Values may be between 0..32767, although hardware limits the highest allowed speed.

Also, with speeds lower than 24.576 MHz all sample rates and bitstream widths are no longer available.

Setting the MSB of SCI_CLOCKF to 1 activates internal clock-doubling. A clock of upto 15 MHz may be doubled depending on the voltage provided to the chip.

Note: SCI_CLOCKF must be set before beginning decoding audio data; otherwise the sample rate will not be set correctly.

Note: Unlike with VS1011, SCI_CLOCKF only needs to be written to after a hardware reset.

Example 1: For a 26 MHz clock the value would be $\frac{26000000}{2000} = 13000$.

Example 2: For a 13 MHz external clock and using internal clock-doubling for a 26 MHz internal frequency, the value would be $0x8000 + \frac{13000000}{2000} = 39268$.

Example 3: For a 24.576 MHz clock the value would be either $\frac{24576000}{2000} = 12288$, or just the default value 0. For this clock frequency, SCI_CLOCKF doesn't need to be set.

8.6.5 SCI_DECODE_TIME (RW)

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

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

SCI_DECODE_TIME is reset at every software reset.

8.6.6 SCI_AUDATA (RW)

When decoding correct data, 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 this register will change the sample rate on the run to the number given.

Example: 44100 Hz stereo data reads as 0xAC45 (44101).

8.6.7 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 call of SCI_WRAM. As 16 bits of data can be transferred with one SCI_WRAM write, and the instruction word is 32 bits long, two consecutive writes are needed for each instruction word. The byte order is big-endian (i.e. MSBs first). After each full-word write, the internal pointer is autoincremented.

SM_WRAMADDR Start...End	Dest. addr. Start...End	Bits/ Word	Description
0x1380...0x13FF	0x1380...0x13FF	16	X data RAM
0x4780...0x47FF	0x0780...0x07FF	16	Y data RAM
0x8030...0x84FF	0x0030...0x04FF	32	Instruction RAM

8.6.8 SCI_WRAMADDR (RW)

SCI_WRAMADDR is used to set the program address for following SCI_WRAM writes.

8.6.9 SCI_HDAT0 and SCI_HDAT1 (R)

For WAV files, SPI_HDAT0 and SPI_HDAT1 read as 0x7761, and 0x7665, respectively.

For MP3 files, SCI_HDAT[0...1] have the following content:

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

When read, SCI_HDAT0 and SCI_HDAT1 contain header information that is extracted from MP3 stream being currently being decoded. Right after resetting VS1002d, 0 is automatically written to both registers, indicating no data has been found yet.

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

“sample rate”	ID=3 / Hz	ID=2 / Hz	ID=0,1 / Hz
3	-	-	-
2	32000	16000	8000
1	48000	24000	12000
0	44100	22050	11025

The “bitrate” field in HDAT0 is read according to the following table:

"bitrate"	ID=3 / kbit/s	ID=0,1,2 / kbit/s
15	forbidden	forbidden
14	320	160
13	256	144
12	224	128
11	192	112
10	160	96
9	128	80
8	112	64
7	96	56
6	80	48
5	64	40
4	56	32
3	48	24
2	40	16
1	32	8
0	-	-

8.6.10 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.

8.6.11 SCI_VOL (RW)

SCI_VOL is a volume control for the player hardware. For each channel, a value in the range of 0 .. 255 may be defined to set its attenuation from the maximum volume level (in 0.5 dB steps). The left channel value is then multiplied by 256 and the values are added. Thus, maximum volume is 0 and total silence if 0xFFFF. Example: for a volume of -2.0 dB for the left channel and -3.5 dB for the right channel: (4*256) + 7 = 0x407. Note, that at startup volume is set to full volume. Resetting the software does not reset the volume setting.

Note: Setting the volume to total silence (255 for both left and right channels) will turn analog power off.

8.6.12 SCI_AICTRL[x] (RW)

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

8.7 Stereo Audio DAC

Decoded digital data is transformed into analog format by an 18/20-bit oversampling multi-bit sigma-delta DA converter. The oversampled output is low-pass filtered by an on-chip analog filter. The output rate of the DA converter is always 1/4 of the clock rate, or 128 times the highest usable sample rate. For instance for a 24.576 MHz clock, the DA converter operates at 128x48 kHz, which is 6.144 MHz. If the input sample rate is other than 48 kHz, it is internally converted to 48 kHz by the DAC. This removes the need for complex PLL-based clocking schemes and allows almost unlimited sample rate accuracy with one fixed master clock frequency.

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

9 Operation

9.1 Clocking

VS1002d operates on a single, nominally 24.576 MHz fundamental frequency master clock. This clock can be generated by external circuitry (connected to pin XTALI) or by the internal clock chrystal interface (pins XTALI and XTALO). This clock is sufficient to support a high quality audio output for almost all standard sample rates and bit-rates (see Application Notes for VS10XX).

9.2 Hardware Reset

When the XRESET -signal is driven low, VS1002d 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 VS1002d are in minimum power consumption stage, and where clocks are stopped. Also XTALO and XTALI are grounded.

After a hardware reset (or at power-up), the user should set such basic software registers as SCI_VOL for volume (and SCI_CLOCKF if the input clock is anything else than 24.576 MHz) before starting decoding.

9.3 Software Reset

In some cases the decoder software has to be reset. This is done by activating bit 2 in SCI_MODE register (Chapter 8.6.1). Then wait for at least 2 μs, then look at DREQ. DREQ will stay down for at least 6000 clock cycles, which means an approximate 250 μs delay if VS1002d is run at 24.576 MHz. After DREQ is up, you may continue playback as usual.

If you want to make sure VS1002d doesn't cut the ending of low-bitrate data streams and you want to do a software reset, it is recommended to feed 2048 zeros to the SDI bus after the file and before the reset.

9.4 SPI Boot

If GPIO0 is set with a pull-up resistor to 1 at boot time, VS1002d 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 VS1002d is 490 kHz with the nominal 24.576 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.

9.5 Play/Decode

This is the normal operation mode of VS1002d. SDI data is decoded. Decoded samples are converted to analog domain by the internal DAC. If there bad problems in the decoding process, the error flags of SCI_HDAT0 and SCI_HDAT1 are set to 0 and analog outputs are muted.

When there is no input for decoding, VS1002d goes into idle mode (lower power consumption than during decoding) and actively monitors the serial data input for valid data.

9.6 Feeding PCM data

VS1002d can be used as a PCM decoder by sending to it a WAV file header. If the length sent in the WAV file is 0 or 0xFFFFFFF, VS1002d will stay in PCM mode indefinitely. 8-bit linear and 16-bit linear audio is supported in mono or stereo.

9.7 SDI Tests

There are several test modes in VS1002d, 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: VS1002d 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.

9.7.1 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, $F_sIdx = 0b011 = 1$ 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.

9.7.2 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.

9.7.3 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 200000 clock cycles. The result can be read from the SCI register SCI_HDAT0, and 'one' bits are interpreted as follows:

Bit(s)	Meaning
15	Test finished
14..7	Unused
6	Mux test succeeded
5	Good I RAM
4	Good Y RAM
3	Good X RAM
2	Good I ROM
1	Good Y ROM
0	Good X ROM

Memory tests overwrite the current contents of the RAM memories.

9.7.4 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_HDAT0. If the register to be tested is HDAT0, the result is copied to SCI_HDAT1.

Example: if n is 48, contents of SCI register 0 (SCI_MODE) is copied to SCI_HDAT0.

10 VS1002d Registers

10.1 Who Needs to Read This Chapter

User software is required when a user wishes to add some own functionality like DSP effects or tone controls to VS1002d.

However, most users of VS1002d don't need to worry about writing their own code, or about this chapter, including those who only download software plug-ins from VLSI Solution's Web site.

10.2 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.

10.3 VS1002d Memory Map

VS1002d's Memory Map is shown in Figure 13.

10.4 SCI Registers

SCI registers described in Chapter 8.6 can be found here between 0xC000..0xC00F. In addition to these registers, there is one in address 0xC010, called SPI_CHANGE.

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

SPI_CHANGE bits		
Name	**Bits**	**Description**
SPI_CH_WRITE	4	1 if last access was a write cycle.
SPI_CH_ADDR	3:0	SPI address of last access.

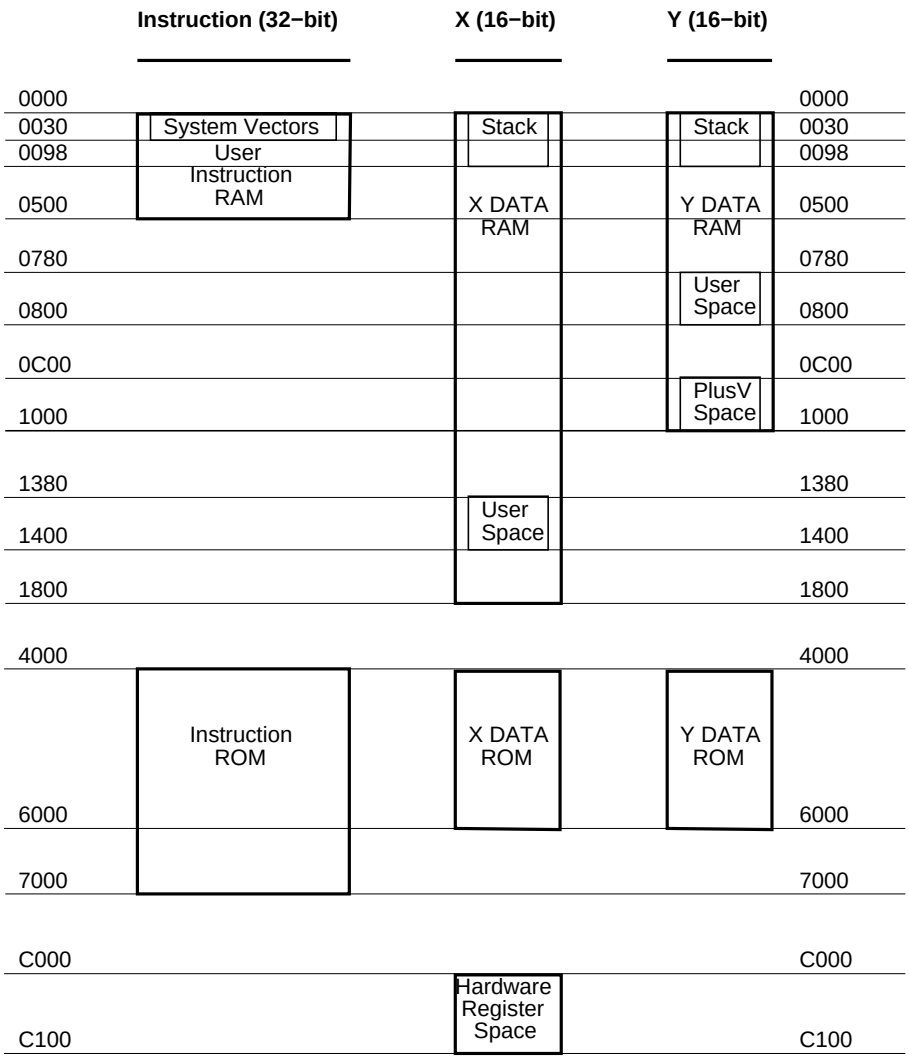


Figure 13: User's Memory Map.

10.5 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.

10.6 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[4:0]	Clock doubler + DAC frequency control MSbs.
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[3:0] $\times$ 65536 + DAC_FCTLL. Whenever this counter overflows, values from DAC_LEFT and DAC_RIGHT are read and a DAC interrupt is generated.

If DAC_FCTL[4] is 1, the internal clock doubler is activated.

10.7 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: Bits 2 and 3 of GPIO_DDR and GPIO_ODATA are switched in prototypes VS1002b and VS1002c. Thus, for example, writing 8 to both registers will set pin GPIO2 to 1 instead of GPIO3.

10.8 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 upto 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 upto 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 upto 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.

10.9 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.

10.10 Watchdog v1.0 2002-08-26

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.

10.10.1 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

10.11 UART v1.0 2002-04-23

RS232 UART implements a serial interface using rs232 standard.

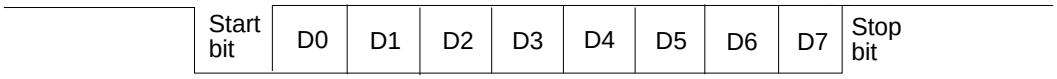


Figure 14: RS232 Serial Interface Protocol

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.

10.11.1 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

10.11.2 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.

10.11.3 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.

10.11.4 Data High UARTx_DATAH

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

10.11.5 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

10.11.6 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.

10.12 Timers v1.0 2002-04-23

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.

10.12.1 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

10.12.2 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 $f_i = \frac{12MHz}{3+1} = 3MHz$.

10.12.3 Configuration TIMER_ENABLE

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

10.12.4 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 10.12.2 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 $f_t = \frac{3MHz}{99+1} = 30kHz$.

10.12.5 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.

10.12.6 Interrupts

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

10.13 System Vector Tags

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

10.13.1 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.

10.13.2 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.

10.13.3 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.

10.13.4 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.

10.13.5 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.

10.13.6 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.

10.13.7 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.

10.13.8 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.

10.13.9 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.

Unless the user is feeding MP3 data at the same time, the system activates the user program in less than 1 ms. After this, the user should steal interrupt vectors from the system, and insert user programs.

10.14 System Vector Functions

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

10.14.1 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 only supported way to write to the User Instruction RAM. This is because Instruction RAM cannot be written when program control is in RAM. Thus, the actual implementation of this function is in ROM, and here is simply a tag to that routine.

10.14.2 ReadIRam(), 0x4

VS_DSP C prototype:

u_int32 ReadIRam(register __i0 u_int16 *addr);

This is the only supported way to read from the User Instruction RAM. This is because Instruction RAM cannot be read when program control is in RAM. Thus, the actual implementation of this function is in ROM, and here is simply a tag to that routine.

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

10.14.3 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.

10.14.4 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.

10.14.5 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.

10.14.6 Reboot(), 0xc

VS_DSP C prototype:

void Reboot(void);

Causes a software reboot.

11 VS1002 Version Changes

This chapter describes changes between different generations of VS1002.

11.1 Changes Between VS1002c and VS1002d, 2004-05-13

- ADPCM recording now works without software patches.

12 Document Version Changes

This chapter describes the most important changes to this document.

12.1 Version 0.71 for VS1002d, 2004-07-20

- Added instructions to add 100 kΩ pull-down resistor to unused GPIOs to Chapter 5.2.

12.2 Version 0.70 for VS1002d, 2004-05-13

- Updated document for VS1002d.
- Removed SM_JUMP.

12.3 Version 0.62 for VS1002c, 2004-03-24

- Redrew Figure 3 to include new microphone connection and serial port.
- Rewrote and clarified Chapter 8.2, Supported Audio Codecs.

12.4 Version 0.61 for VS1002c, 2004-03-11

- Added samplerate and bitrate tables to Chapter 8.6.9.

12.5 Version 0.6 for VS1002c, 2004-02-13

- A/D Modulator powerdown bit explained in (Chapter 10.9).
- Added BGA-49 to Packages and Pin Descriptions (Chapter 5).
- Added new Chapter 8.2, Supported Audio Codecs.

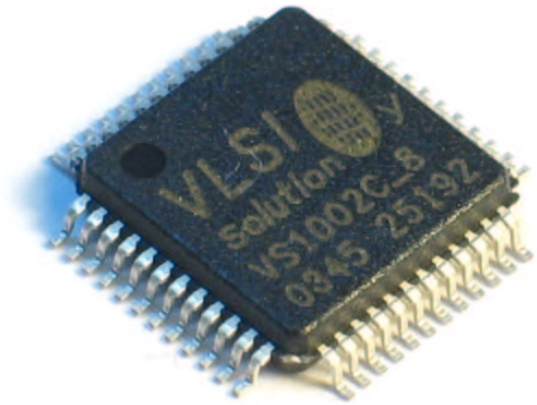
12.6 Version 0.5 for VS1002c, 2004-01-15

- Modified Analog Characteristics (Chapter 4.1).
- Bit SM_ADPCM was marked as broken. Because SM_ADPCM actually can be made work if implemented according to application notes, the notice has been removed.

12.7 Version 0.4 for VS1002c, 2003-12-23

- Added SDI/SCI Examples (Chapter 7.6).

13 Contact Information



VLSI Solution Oy
Hermiankatu 6-8 C
FIN-33720 Tampere
FINLAND

Fax: +358-3-316 5220
Phone: +358-3-316 5230
Email: sales@vlsi.fi
URL: http://www.vlsi.fi/

Note: If you have questions, first see http://www.vlsi.fi/vs1002/faq/ .