VS1103 Module

Features

- Plays audio from μSD card
- Plays audio from SPI Flash (separate firmware)
- Plays MIDI and WAV (PCM, IMA ADPCM)
- Firmware and content can be customized
- Controlled via SPI (μSD) or UART (SPI FLASH)
- High-performance CD-quality analog out
- Supports various sample rates
- Operates from a single supply (VCC=4.0 to 6.0 V)
  (Note: IO pins are not 5 V-tolerant. Do not drive them over 3.6 V!)
- Line output
- Headphone output
- Line input
- Microphone input
- SPI FLASH for code and/or audio data storage
- μSD connector for large audio data storage
- The most useful signals are routed to pin headers for easy access and customization

Applications

- Elevators
- Ticket machines
- Audio user's manuals
- Vending machines
- Car accessories
- Alarm systems
- PC accessories
- Speaking hi-tech toys
- Development board for VS1103 chip

Description

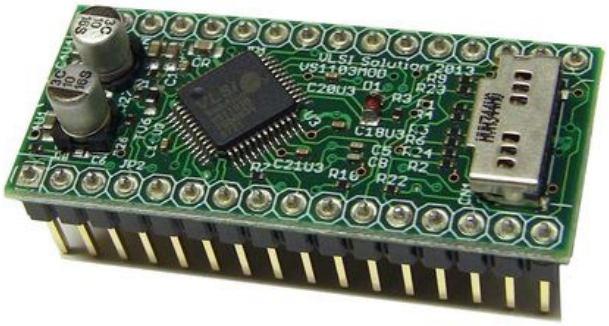
The VS1103 Module is a small, low-cost, high-performance, easy-to-use audio player that is controlled from serial control interface (when playing from μSD card) or an UART (when playing from SPI FLASH). It can be used as a "plug-in" audio board in electronic systems or as a standalone small audio player.

The product is supported by design services, audio content preparation and pre-programming. It is also fully configurable by the user by using free Integrated Development Tools (VSIDE) for the VS1103 IC. The use of the module does not require any advanced information from the user.

Pin headers of the PCB are compatible with DIL32 footprint 2.54 pitch 15.24 mm wide (100 mils pitch, 600 mils wide). This makes it possibe to use standard DIL32 ZIF sockets or solder it by using DIL32 footprint.

The left row of the PCB has digital connections for SPI control or other circuitry. The right row has power connections, analog connections and a serial port. These can be used to interface the board with a PC and VSIDE or a host microcontroller.

The VS1103 Module operates from a single power supply. The board has either 2MBytes (VSMD301) or 16MBytes (VSMD321) of on-board FLASH and a μSD card connector for audio content. The module boots from the on-board FLASH memory.



Contents

List of Figures

1 Disclaimer

All properties and figures are subject to change.

2 Definitions

B Byte, 8 bits.

b Bit.

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.

3 Placement and Pinout

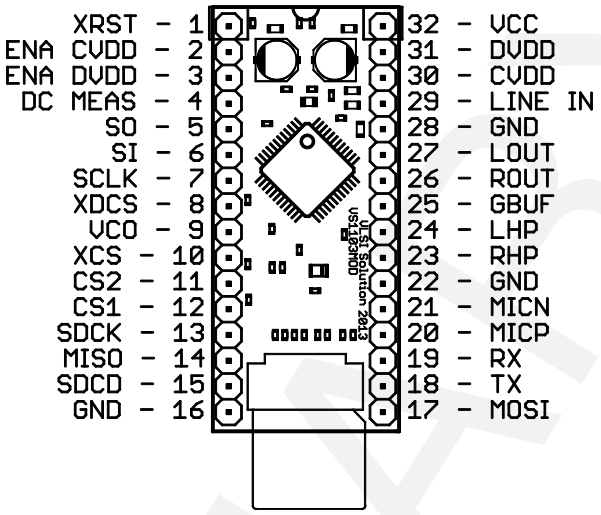


Figure 1: Pinout.

Pin	Name	Description
1	XRESET	Active Low reset, 100KΩ pull-up
2	ENA_CVDD	Core voltage regulator enable, 100KΩ pull-up
3	ENA_DVDD	IO voltage regulator enable, 100KΩ pull-up
4	DC_MEAS [1]	DC input to ADC
5	SO	SCI data output
6	SI	SCI / SDI data input
7	SCLK	SCI / SDI data clock
8	XDCS	SDI chip select, 100KΩ pull-up
9	VCO	Optional clock output
10	XCS	SCI chip select, 100KΩ pull-up
11	CS2	μSD chip select, GPIO1, 100KΩ pull-up
12	CS1	SPI FLASH chip select, GPIO0, 10KΩ pull-up
13	SDCK	CLK for μSD, GPIO3, 100KΩ pull-down
14	MISO	Data out from SPI FLASH and μSD, GPIO2, 100KΩ pull-down, 10KΩ to CS1 [2]
15	SDCD	SD card detect, 100KΩ pull-up
16	GND	Ground, connected to the ground plane

Pin	Name	Description
32	VCC	Power supply voltage 4.0V - 6.0V
31	DVDD	DVDD output of 3.3V, for external circuitry
30	CVDD	CVDD output of 2.7V, for external circuitry
29	LINE_IN [1]	LINE input with DC block
28	GND	Ground, connected to the ground plane
27	LOUT	Left channel line out (AC-coupled)
26	ROUT	Right channel line out (AC-coupled)
25	GBUF	Headphone 1.2 V common output (note DC-bias)
24	LHP	Headphone left channel output (note DC-bias)
23	RHP	Headphone right channel output (note DC-bias)
22	GND	Ground, connected to the ground plane
21	MICN	MIC input negative
20	MICP	MIC input positive
19	RX	Serial port receive, diode, 10KΩ pullup, 5V tolerant
18	TX	Serial port transmit
17	MOSI	Data in for μSD and SPI FLASH, DREQ

[1] DC_MEAS and LINE_IN use the same ADC input, so connect only one.
[2] Due to the resistor between CS1 and MISO, MISO gets a pull-up during μSD access and a pull-down during boot from SPI FLASH.
Note: The maximum voltage for IO pins (except RX) is 3.6V ! Only VCC can be 4.0 V - 6.0 V.

4 Dimensions

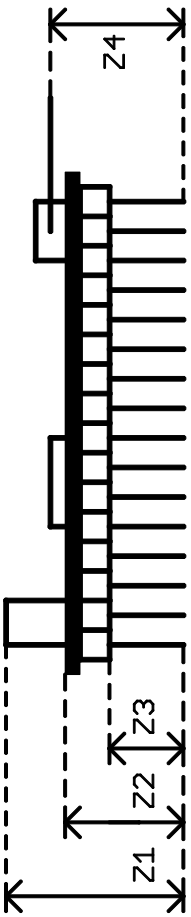


Figure 2: Side Dimensions.

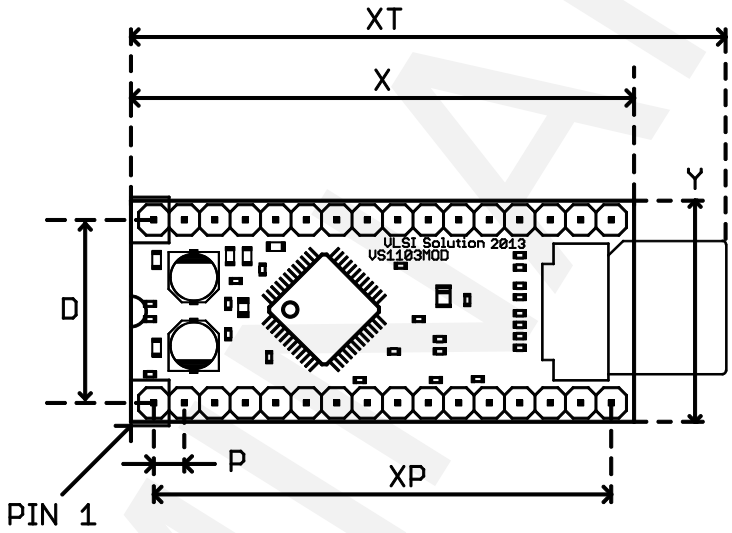


Figure 3: Top Dimensions.

Parameter	Symbol	Min	Typ	Max	Unit
Width of PCB	Y		18.40		mm
Length of PCB	X		41.91		mm
Total length (includes USB connector)	XT		40.65		mm
Length of pin connector	XP		38.10		mm
Maximum height of module	Z1		16.60		mm
Minimum height of module	Z2		11.60		mm
Pitch of pins	P		2.54		mm
Pin height	Z3		6.80		mm
Distance from lower pin row to upper pin row	D		15.24		mm
Distance to microSD card centerline	Z4		14.40		mm
Pin type and size	PT		0.65 square		mm

5 Characteristics & Specifications

5.1 Absolute Maximum Ratings

Parameter	Symbol	Min	Max	Unit
Supply voltage	VCC	-0.3	6.0	V
Voltage at Any Digital Input		-0.3	IOVDD+0.3[1]	V
Total Injected Current on Pins			±200 [2]	mA
Operating Temperature		-40	+85	°C
Storage Temperature		-65	+150	°C

[1] IOVDD is 3.3 V in this module. Voltage on any digital IO pin should not exceed 3.6 V (except the RX pin)
[2] Latch-up limit

5.2 Recommended Operating Conditions

Parameter	Symbol	Min	Typ	Max	Unit
Operating temperature		-40		+85	°C
Ground [1]	GND		0.0		V
Supply voltage	VCC	4.0[2]	5.0	6.0	V
Digital positive supply[3]	CVDD		2.7		V
Analog positive supply[3]	AVDD		2.7		V
I/O positive supply	DVDD		3.3		V
Input clock frequency	XTALI		12.288		MHz

[1] Do not float ground for latch-up immunity.
[2] See the regulator datasheet. To provide 3.3 V at maximum power consumption, the supply voltage needs to be at least 3.9 V.
[3] CVDD and AVDD share a regulator in this design.

5.3 Analog Characteristics of Audio Outputs

Unless otherwise noted: AVDD=2.7V, CVDD=2.7V, IOVDD=3.3V, TA=-40..+85°C, XTALI=12.288 MHz, DAC tested with full-scale output sinewave, measurement bandwidth 20..20000 Hz, analog output load: LHP to GBUF 30 Ω, RHP to GBUF 30 Ω, LOUT: 10k Ω, LOUT: 10k Ω.

Parameter	Symbol	Min	Typ	Max	Unit
DAC Resolution			18		bits
Dynamic range (DAC unmuted, A-weighted, min gain)	IDR		90		dB
S/N ratio (full scale signal, no load[1])	SNR		86		dB
S/N ratio (full scale signal, 30 ohm load[2])	SNRL	75	84		dB
Total harmonic distortion, max level, no load[1]	THD		0.01		%
Total harmonic distortion, max level, 30 ohm load[2]	THDL		0.1	0.3	%
Crosstalk (LOUT/ROUT to ROUT/LOUT), no load[1]	XTALK0		75		dB
Crosstalk (LHP/RHP to RHP/LHP), 30 ohm load, without GBUF [3]	XTALK1		75		dB
Crosstalk (LHP/RHP to RHP/LHP), 30 ohm load, with GBUF	XTALK2		40		dB
Gain mismatch (LOUT/ROUT to ROUT/LOUT)	GERR	-0.5		0.5	dB
Frequency response	AERR	-0.1		0.1	dB
Full scale output voltage	LEVEL	450	530	600	mVrms
Deviation from linear phase	PH		0	5	°
Analog output load resistance	AOLR		30[4]		Ω
Analog output load capacitance	AOLC			100	pF
DC level (GBUF, LHP, RHP)		1.1		1.3	V

[1] Characteristics with no load are measured from LOUT/ROUT outputs towards GND such that LHP/RHP outputs are not loaded.
[2] Characteristics with 30 Ω load are measured from LHP/RHP outputs towards GBUF such that LOUT/ROUT outputs are not loaded.
[3] Loaded from LHP/RHP pin to analog ground via 100 µF capacitors.
[4] AOLR may be lower than *Typical*, but distortion performance may be compromised.

5.4 Power Consumption

The power consumption is audio-dependant as well as load-dependant.

Parameter	Min	Typ	Max	Unit
Current Consumption in Reset (XRESET=0V) @ 25 °C		90		µA
SPI FLASH player application VCC=4.0V				
Total Power, play mode, no load		100		mW
Total Power, play mode, LHP and RHP with 30Ω load to GBUF		120	200	mW
Total Power, play mode, LOUT and ROUT with 10kΩ load				mW
uSD player application VCC=4.0V				
Total Power, play mode, LHP and RHP with 30Ω load to GBUF		220		mW
Total Power, play mode, LOUT and ROUT with 10kΩ load				mW

5.5 Digital Characteristics

Parameter	Min	Typ	Max	Unit
High-Level Input Voltage	$0.7\times$IOVDD		IOVDD+0.3	V
Low-Level Input Voltage	-0.2		$0.3\times$IOVDD	V
High-Level Output Voltage, -1.0 mA load	$0.7\times$IOVDD			V
Low-Level Output Voltage, 1.0 mA load			$0.3\times$IOVDD	V
XTALO high-level output voltage, -0.1 mA load	$0.7\times$IOVDD			V
XTALO low-level output voltage, 0.1 mA load			$0.3\times$IOVDD	V
Input leakage current	-1.0		1.0	μA
Rise time of all output pins, load = 30 pF			50	ns

6 PCB and Component Layout

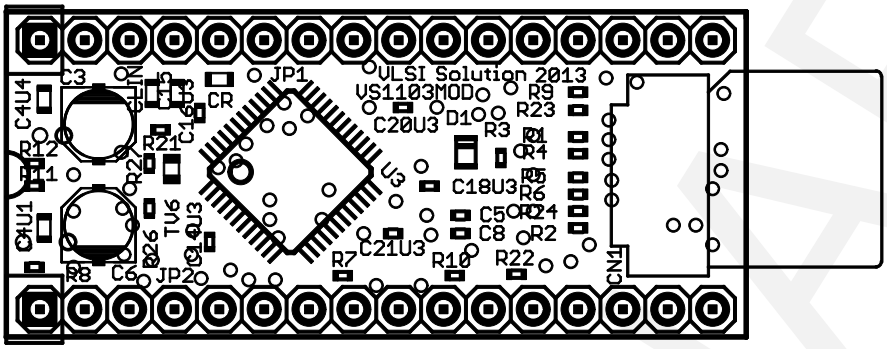


Figure 4: Top layer and silkscreen of the PCB (Top view)

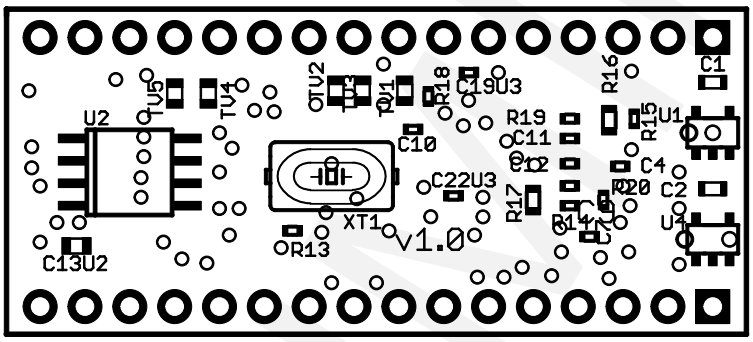


Figure 5: Bottom layer and silkscreen of the PCB (Bottom view)

7 PCB Schematics

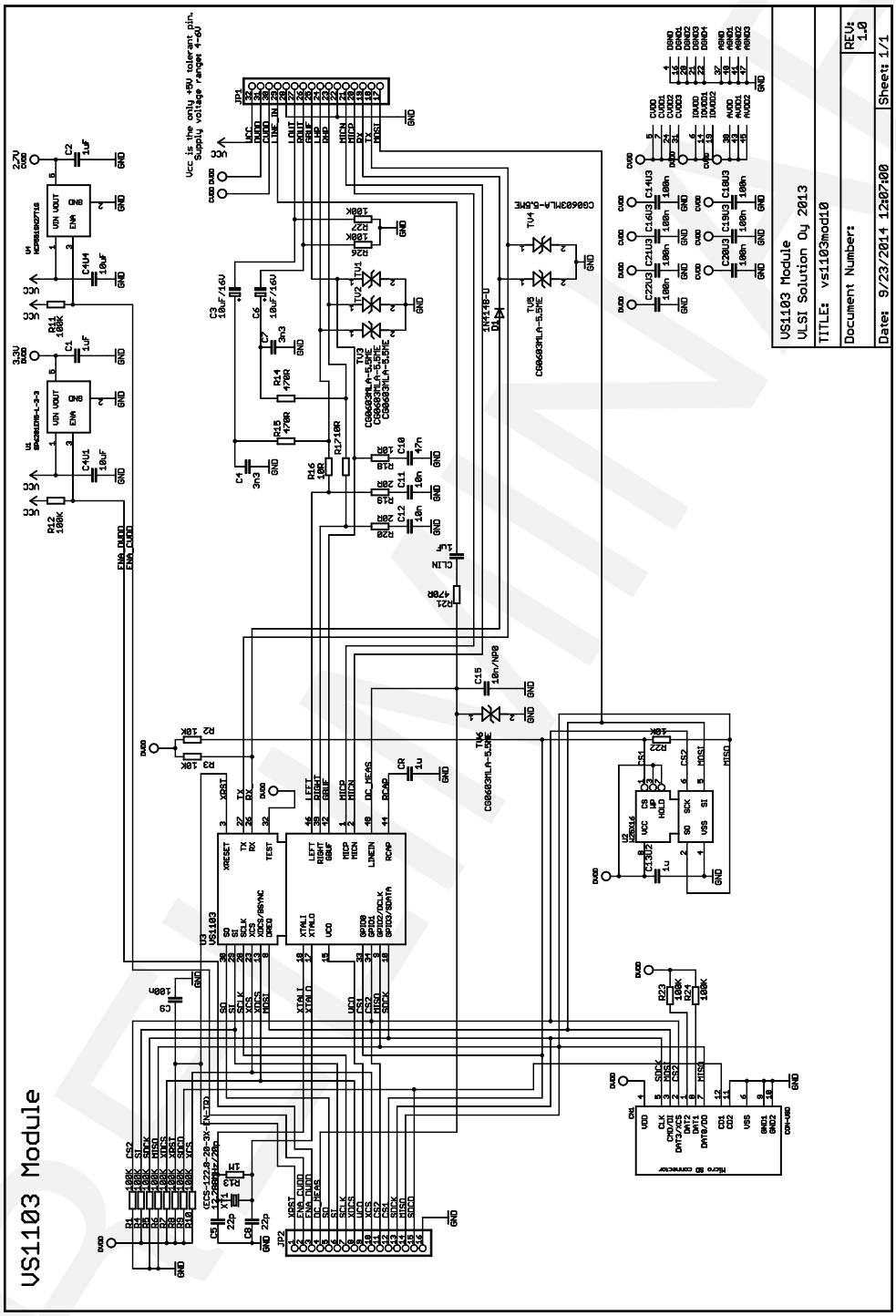


Figure 6: Schematic of the VS1103 Module.

7.1 Notes about the Schematic

- Line outputs have a first order RC low-pass filter that removes some of the DA converter quantization noise. Useful when connecting to a limited bandwidth amplifier.
- Line outputs are AC-coupled, so they can be connected directly to an amplifier.
- Headphone outputs (including common buffer) are DC-biased to 1.2V. Do not connect the common buffer output to ground! Do not connect Left/Right headphone outputs directly to an amplifier!
- Power supply voltage (from 4.0V to 6.0V) is connected to pin 1 of JP1, pin 32 of the module.
- DC_MEAS and LINE_IN use the same ADC input, so it is not a good idea to connect them both.
- Note: The maximum voltage for IO pins except RX is 3.6V ! The supply voltage (VCC) can be from 4.0 V to 6.0 V.
- The pinout is partly compatible with the VS1000 Audio Module.

8 Available Firmware Versions

- sdmodule - play audio files from μSD card. This is the default version.
- spimodule - play audio files from the internal SPI FLASH.
- rtmidimodule - starts the module in real-time MIDI mode.
- recmodule - plays files to and records files from the SPI FLASH.

The main firmware variants for the VS1103 Module are sdmodule and spimodule. The firmware resides in the SPI FLASH and is loaded from there during power-on / reset. Sdmodule plays files from a μSD card. Spimodule plays pre-programmed audio from the same SPI FLASH.

`rtmidimodule` plays audio using real-time MIDI commands. `recmodule` can save audio to SPI FLASH and play back the results using UART control.

Playback Formats

Only the formats supported by VS1103 are played. MIDI (.mid, format 0) and WAV (.wav, mono and stereo linear 8-bit and 16-bit and IMA ADPCM) are supported. You may need to convert your midi file from format 1 to format 0. See http://www.vlsi.fi/en/support/software/vs10xxtools.html for a converter source code.

8.1 sdmodule - uSD Player

- Plays files from an inserted μSD card.
- Controlled through Serial Control Interface (SCI, see VS1103 datasheet)
- Some status information is sent through UART
- Limited control through UART (protocol converts UART bytes to SCI writes)

In practice this is the same firmware than the VS1103 Standalone Player application.

The source code is available as `vs1103module.zip`. It uses the `make` tool from the VSIDE or generic VSDSP tools package and a `Makefile` file to build the different versions. Start `make` from the command line and it performs the build actions for you. You need `VSIDE\bin` in your command line search path.

Starting from version 0.60 the source code is available as a VSIDE solution. One VSIDE solution contains the single-image `sdmodule`, `spimodule`, and `rtmidimodule` firmwares. Another VSIDE solution contains the multi-load `recmodule` firmware.

`pcmodflash.exe` is included in both packages.

8.1.1 Power-on Defaults

Default values are loaded from SPI EEPROM after power-on or reset. The values can be found and edited from the source file `c1103.s` .

The input clock is assumed to be 12.288 MHz. If you want to use a different crystal, you can change the SCI_CLOCKF value from `c1103.s` . The default value is 0xc000 for 4.0× 12.288 MHz.

Volume (SCI_VOL) default value is 0x0000, which corresponds to full volume, i.e. no attenuation.

SCI_BASS default value is 0, i.e. both bass and treble controls are off. If you set bass control, the bass control value should be odd to make the loudness indicator LED blink work.

8.1.2 SCI Control

xCS, SI, SO, and SCLK should be connected to the host controller's SPI bus. Serial Control Interface (SCI) reads and writes are used to communicate with the player. See vs1103 datasheet for more information about SCI.

All non-application SCI registers can be used normally, except that SM_SDINEW must be kept at '1' to enable GPIO2 and GPIO3. SCI_AIADDR, SCI_AICTRL0, SCI_AICTRL1, SCI_AICTRL2, and SCI_AICTRL3 are used by the player.

SCI registers		
Reg	**Abbrev**	**Description**
0x0	MODE	Mode control, SM_SDINEW=1
0x1	STATUS	Status of VS10xx
0x2	BASS	Built-in bass/treble control
0x3	CLOCKF	Clock freq + multiplier
0x4	DECODE_TIME	Decode time in seconds
0x5	AUDATA	Misc. audio data
0x6	WRAM	RAM write/read
0x7	WRAMADDR	Base address for RAM write/read
0x8	HDAT0	Stream header data 0
0x9	HDAT1	Stream header data 1
0xA	**AIADDR**	**Player private, do not change**
0xB	VOL	Volume control
0xC	**AICTRL0**	**Current song number / Song change**
0xD	**AICTRL1**	**Player internal use, do not change**
0xE	**AICTRL2**	**Number of songs on MMC**
0xF	**AICTRL3**	**Play mode**

The currently playing song can be read from SCI_AICTRL0. In normal play mode the value is incremented when a file ends, and the next file is played. When the last file has been played, SCI_AICTRL0 becomes zero and playing restarts from the first file.

Write 0x8000 + song number to SCI_AICTRL0 to jump to another song. The high bit will be cleared when the song change is detected. The pause mode (CTRL3_PAUSE_ON), file ready (CTRL3_FILE_READY), and paused at end (CTRL3_AT_END) bits are automatically cleared. If the song number is too large, playing restarts from the first file. If you write to SCI_AICTRL0 before starting the code, you can directly write the song number of the first song to play.

SCI_AICTRL2 contains the number of songs (files) found from the MMC card. You can disable this feature (CTRL3_NO_NUMFILES) to speed up the start of playback. In this case AICTRL2 will contain 0x7fff after MMC/SD has been successfully initialized.

SCI_AICTRL3 bits		
Name	**Bit**	**Description**
CTRL3_UPDATE_VOL	15	'1' = update volume (for UART control)
CTRL3_BY_NAME	8	'1' = locate file by name
CTLR3_AT_END	6	if PLAY_MODE=3, 1=paused at end of file
CTLR3_NO_NUMFILES	5	0=normal, 1=do not count the number of files
CTLR3_PAUSE_ON	4	0=normal, 1=pause ON
CTLR3_FILE_READY	3	1=file found
CTLR3_PLAY_MODE_MASK	2:1	0=normal, 1=loop song, 2=pause before play, 3=pause after play
CTLR3_RANDOM_PLAY	0	0=normal, 1=shuffle play

If the lowest bit of SCI_AICTRL3 is 1, a random song is selected each time a new song starts. The shuffle play goes through all files in random order, then plays them in a different order. It can play a file twice in a row when a new random order is initiated.

Play Modes

The play mode mask bits are used to change the default play behaviour. Set AICTRL3 to contain the default play mode from `c1103.s`. If the mode is changed during play through the serial control interface (SCI), care must be taken.

In *normal* mode the files are played one after another.

In *loop song* mode the playing file is repeated until a new file is selected. CTRL3_FILE_READY is set to indicate a file was found and playing has started, but it will not be automatically cleared.

Pause before play mode first locates the file, then goes to pause mode. CTRL3_PAUSE_ON will get set to indicate pause mode, CTRL3_FILE_READY will be set to indicate a file was found. When the user has read the file ready indicator, he should reset the file ready bit. The user must also reset the CTRL3_PAUSE_ON bit to start playing. One use for the *pause before play* mode is scanning the file names.

Pause after play mode plays files normally, but will go to pause mode and set the CTRL3_AT_END bit right after finishing a file. AICTRL0 will be increased to point to the next file (or the number of files if the song played was the last file), but this file is not yet ready to play. CTRL3_PAUSE_ON will get set to indicate pause mode, The user must reset the CTRL3_PAUSE_ON bit to move on to locate the next file, or select a new file by writing 0x8000 + song number to AICTRL0. CTRL3_PAUSE_ON, CTRL3_FILE_READY, and CTRL3_AT_END bits are automatically cleared when new file is selected through AICTRL0.

Pause after play and *loop mode* are only checked when the file data has been fully read. *Pause before play* is checked after the file has been located, but before the actual playing starts. Take this into account, if you want to change playing mode while files are playing.

You can speed up the start of playback by setting CTRL3_NO_NUMFILES. In this case the number of files on the card is not calculated. In this mode SCI_AICTRL2 contains 0x7fff after MMC/SD has been successfully initialized. This affects the working of the shuffle mode, but the bit is useful if you implement random or shuffle play on the microcontroller. You probably want to determine the number of files on the card once to make it possible to jump from the first file to the last.

CTRL3_BY_NAME

You can also open specific files by using the CTRL3_BY_NAME bit.

First set pause mode bit CTRL3_PAUSE_ON and the open-by-name bit CTRL3_BY_NAME in AICTRL3, then write the 8.3-character filename into memory, then write 0xffff to AICTRL0 to select the song. After a file has been located you can read back the file name to see if the file was located or not. You can also check SCI_AICTRL0: if it is non-zero, the file has been located, otherwise you have to check the file name to be certain.

To write the file name, first write 0x5800 to SCI_WRAMADDR, then the 6 words of the file name to SCI_WRAM. The MSDOS 8.3-character filename does not include the point, so instead of sending "00000002.MP3" you need to send "00000002MP3\0", i.e. without the . and pad with a zero byte.

8.1.3 UART Control

The sdmodule player supports only limited control through UART at 9600bps data rate (8 data bits, no parity, 1 stop bit). You should use SCI control if you can, the UART control is not very stable and you can't read values back.

There are short periods during MMC/SD initialization when VS10xx does not receive UART bytes. The received bytes are echoed to TX so you can resend bytes if needed.

In the `sdmodule` firmware UART is just an alternative way to write SCI registers. You send 4-byte commands to write to SCI registers, or any register in the range of 0xc000..0xc07f. The first three bytes send 2, 7, and 7 bits of a 16-bit data value and each byte has the most significant bit cleared. The last byte has the most significant bit set, and carries the register number to write to in the low 7 bits.

```
/* putch() sends a 8-bit value through UART */
void WriteRegThroughUart(unsigned short value, unsigned short reg) {
    putch((value>>14) & 127); /* Send top 2 bits */
    putch((value>>7) & 127);  /* Send 7 more bits */
    putch(value & 127);       /* Send the low 7 bits */
    putch(reg | 0x80);        /* Send the register address with high bit set */
}
```

Example: to select song #10, use WriteRegThroughUart(0x800a, 0x0c) i.e. send bytes 0x02,

0x00, 0x0a, 0x8c. The value to write is (0x02<<14) | (0x00<<7) | 0x0a = 0x800a, and it will be written to 0xc00c, i.e. to SCI_AICTRL0. As can be seen from the SCI control documentation, this will prompt the player to end the playing of current song and start playing song #10.

To set volume: send 0x00, 0x30, 0x10, 0x8b to set SCI_VOL to 0x1010. This sets the volume register to the new value, but does not yet calculate new volume, because the write did not cause an SCI interrupt (this is also why writing to SCI_WRAMADDR and SCI_WRAM with UART control does not write to memory).

To force the new volume to be applied, send 0x02, 0x00, 0x00, 0x8f to set the volume update flag (CTRL3_UPDATE_VOL) in SCI_AICTRL3. If you are using some other play mode than normal play mode, or pause mode is on, you have to adapt the writes accordingly.

Status Through UART

Some status information is also returned. 'e' is returned after a song ends. Loop mode sends 'l' for every restart of the song. 'p' is sent when pause mode is entered, and 'c' when playing continues. This information is sent also when you use SCI control.

Byte/TX	Status
0x65 'e'	song
0x70 'p'	song paused
0x63 'c'	play continued
0x6c 'l'	song looped

8.2 spimodule - SPI FLASH Player

- Controlled through Serial Control Interface (SCI, see VS1103 datasheet)
- Plays files from the SPI FLASH
- Status information is sent through UART
- Good control through UART

The Serial Control Interface can be used to control the spimodule firmware in the same way as sdmodule. Also power-on defaults are configured in the same way. See the sdmodule documentation for details (sections 8.1.1 and 8.1.2). In addition, spimodule can be controlled extensively using UART commands.

UART speed 9600 bps is used by default. The module boots up in the *continuous* playing mode, which plays all available files sequentially and continues playing until the module is powered off.

The source code is available as `vs1103module.zip`. It uses the `make` tool from the VSIDE or generic VSDSP tools package and a `Makefile` file to build the different versions. Start `make` from the command line and it performs the build actions for you. You need `VSIDE\bin` in your command line search path.

Starting from version 0.60 the source code is available as a VSIDE solution. One VSIDE solution contains the single-image `sdmodule`, `spimodule`, and `rtmidimodule` firmwares. Another VSIDE solution contains the multi-load `recmodule` firmware.

`pcmodflash.exe` is included in the packages.

8.2.1 UART Commands

- `'+'` volume up, responds with a byte indicating the new volume value
- `'-'` volume down, responds with a byte indicating the new volume value
- "XV\n" sets volume attenuation to X (decimal) in 0.5dB steps, e.g. "24V\n" sets -12dB. Volume commands respond with a byte indicating the new volume value.
- `'C'` cancel play, in effect replay the same file from the beginning (unless pause-before-play mode is active)
- `'n'` next file
- `'p'` previous file, or if played 5 seconds or more, the same file from the beginning
- "XP\n" play file X (decimal), e.g. "11P\n" plays the 12th file.
  Responds with "done\n" at the end of the current file, "files" + two bytes + "\n" for the total number of files available, then "play FILENAME\n".
  Note: do not send the next byte immediately after commands that change the file (n, p, and P). Reinitialization of the UART FIFO may lose bytes.
- `'c'` selects continuous play mode
- `'f'` selects file play mode (pause before play)
- `'l'` selects loop play mode (the same file is played in an infinite loop)
- `'='` pause play
- `'>'` continue play (set 1x play speed)

- '≫' (0xbb) play faster (only works for MIDI)
- '?' request information, will respond with two-byte decode time (high byte first) and a byte indicating file read position 0..255 (start..end). You can calculate a percentage by dividing the value by 2.56 .

8.3 rtmidimodule - Real-Time MIDI player

- Plays MIDI from real-time MIDI commands
- MIDI commands received from both UART (31250bps) or from SDI (expand each MIDI byte to a word before sending)

The source code is available as `vs1103module.zip`. It uses the `make` tool from the VSIDE or generic VSDSP tools package and a `Makefile` file to build the different versions. Start `make` from the command line and it performs the build actions for you. You need `VSIDE\bin` in your command line search path.

Starting from version 0.60 the source code is available as a VSIDE solution. One VSIDE solution contains the single-image `sdmodule`, `spimodule`, and `rtmidimodule` firmwares. Another VSIDE solution contains the multi-load `recmodule` firmware.

`pcmodflash.exe` is included in the packages.

8.4 recmodule - SPI FLASH Player / Recorder

- Controlled through Serial Control Interface (SCI, see VS1103 datasheet)
- Plays files from the SPI FLASH
- Status information is sent through UART
- Records files into SPI FLASH from line input or microphone
- Good control through UART
- This ia a multi-load application (has separate programs for the operations)

The Serial Control Interface can be used to control the recmodule firmware in the same way as spimodule. Also power-on defaults are configured in the same way. See the sdmodule documentation for details (sections 8.1.1 and 8.1.2). In addition, recmodule can be controlled extensively using UART commands.

UART speed 9600 bps is used by default. The module boots up in the *continuous* playing mode, which plays all available files sequentially and continues playing until the module is powered off.

8.4.1 UART Commands

The UART commands are almost the same as in the **spimodule** firmware.

Play mode

- `'+'` volume up, responds with a byte indicating the new volume value
- `'-'` volume down, responds with a byte indicating the new volume value
- "XV\n" sets volume attenuation to X (decimal) in 0.5dB steps, e.g. "24V\n" sets -12dB. Volume commands respond with a byte indicating the new volume value.
- `'C'` cancel play, in effect replay the same file from the beginning (unless pause-before-play mode is active)
- `'n'` next file
- `'p'` previous file, or if played 5 seconds or more, the same file from the beginning
- "XP\n" play file X (decimal), e.g. "11P\n" plays the 12th file.
  Responds with "done\n" at the end of the current file, "files" + two bytes + "\n" for the total number of files available, then "play FILENAME\n".
  Note: do not send the next byte immediately after commands that change the file (n, p, and P). Reinitialization of the UART FIFO may lose bytes.
- `'c'` selects continuous play mode
- `'f'` selects file play mode (pause before play)
- `'l'` selects loop play mode (the same file is played in an infinite loop)
- `'='` pause play
- `'>'` continue play (set 1x play speed)
- `'≫' (0xbb)` play faster (only works for MIDI)
- `'?'` request information, will respond with two-byte decode time (high byte first) and a byte indicating file read position 0..255 (start..end). You can calculate a percentage by dividing the value by 2.56 .
- `'R'` record a file from line input
- `'M'` record a file from microphone input
- `"239F"` format the disk - removes all files

Record mode

- `'+'` volume up, responds with a byte indicating the new volume value
- `'-'` volume down, responds with a byte indicating the new volume value
- `"`X`V\n"` set volume attenuation to X (decimal) in 0.5dB steps, e.g. "24V\n" sets -12dB. Volume commands respond with a byte indicating the new volume value.
- `'C'` cancel recording, does not create a new file
- `'E'` end recording, creates a file entry (named "RECORDXX", where XX is an ascending number).
- `'?'` request information, currently responds with a two-byte decode time (high byte first)

8.4.2 Source Code

A VSIDE solution of the recmodule is available as `vs1103-spirecorder.zip`. The solution consists of 5 projects, post-build instructions to create separate boot images and combine them into one boot image (spiall.spi), and the pcmodflash.exe external tool.

Starting from version 0.60 the source code is available as a VSIDE solution. One VSIDE solution contains the single-image `sdmodule`, `spimodule`, and `rtmidimodule` firmwares. Another VSIDE solution contains the multi-load `recmodule` firmware.

vs1103-spirecorder
This is the initialization program (`recinit.c`), which sets up the hardware and performs firmware initializations, determines the SPI FLASH size from the RDID sequence, sets up UART receive interrupt, and then loads and executes the player program (`recmodule.c`).

recplayer
This is the player program (`recmodule.c`), which performs the player functionality. See the UART commands from the previous section. There are separate UART commands to start recording from the microphone input and from the line input.

When required, the player loads either the recording program (`recmodule2.c`), which handles recording, or the command program (`reccmd.c`), which handles external commands that are too large to fit into the player program.

recrecorder
This program handles the analog to digital conversion and saving of the data into the SPI FLASH. UART commands are listed in the previous section. You are able to cancel the recording without creating a file, or you can end the recording normally, which creates a file.

After the recording is ended, the player program is loaded automatically.

reccmd
This program (`reccmd.c`) handles commands that do not fit the player program, for example the formatting of the SPI FLASH from audio data.

libmodule
This project creates a static link library which the programs in the other projects can use. The link library `libmodule.a` contains SPI, MMC, UART, and filesystem routines.

Post-Build
The post-build steps of the init, player, recorder, and cmd programs create boot images from each of the executable programs and then create a combined boot image `spiall.spi` . This is also copied to `eeprom.img` .

`eeprom.img` can be programmed with audio files into the SPI FLASH using the `pcmodflash.exe` tool as described later in this document.

Tools
You can configure an external tool entry `Program vs1103 module` for the Tools menu of VSIDE to perform the programming using `pcmodflash.exe` . You will probably need to change the serial port setting or at least the audio files that are included. Use the `Manage External Tools` option, add the `Program vs1103 module` entry using the "New..." button (or select it if you already have the entry), and change the command line to match the serial port of your setup and the audio files you want to include.

The default `Program vs1103 module` entry would be something like:

pcmodflash -p 5 -l spiall.spi -f Ben2.wav

You can then select the entry from the Tools menu to easily program the module. However, note that the output of the program will appear in the debug window only after the program has run completely, so running pcmodflash from commandline is also a good option to see the progress in real time.

Note that you need the newest version of pcmodflash, so that any unused areas in the directory are set to 0xff instead of 0x00. Otherwise the data area needs to be formatted before recmodule can save any files.

9 Programming/Reprogramming the Module

The VS1103 Module can be programmed using pcmodflash.exe through the RX and TX UART pins.

An already programmed vs1103b module with the spimodule firmware can be reprogrammed in the same way, because the spimodule firmware detects the jump-to-monitor command sent through UART. However, for sdmodule or other firmware without this feature, you need to prevent the normal boot by grounding CS1 during power-up, then remove the grounding before running pcmodflash.

Pcmodflash connects to the VS1103 chip through UART, uploads a SPI FLASH programmer using the ROM monitor routines, then communicates with the programmer to send both firmware file and the audio content. Block transfer is handled using interrupts, so the programming of the previous block can be performed simultaneously.

9.1 spimodule or other firmware with content files

The spimodule firmware does not use FAT filesystem, because that would require unnecessary code space. A custom structure is used to list the file, so that they can still be named and referred by their names.

When programming, the audio content can be given as an image file prepared by mkspifs.exe (included as source code), or each audio file (.wav or .mid) can be specified as a parameter to pcmodflash using the -f option for each file to add. pcmodflash will then create the required structure automatically.

Example:

```
> pcmodflash -l spimodule.spi -f content-for-2M.wav -p 1 -sm 4
```

The above command specifies COM1 (-p 1), the firmware file to program (-l spimodule.img), and the files to program as audio contents (this time only one -f option to program one audio file).

The UART speed can be increased with the -sm option. Normally -sm 4 is the maximum that can be used (for 460800bps). Fortunately, above this the SPI FLASH write speed starts to be the limiting factor instead of the UART speed.

The output of the command looks like the following.

```
Adding File 'content-for-2M.wav' 'content-for-2M.wav' 1946684 bytes
1 files (1)
Header 1024 bytes (2 sectors)
Data   1947136 bytes (3803 sectors)
Trying to connect..
a5 VS10XX
```

```
Loading code...
Executing VS1103 code (0x0030)
Changed UART speed to 460800bps.
Programming tmp.img at sector 64
VS1103 Ready, performing chip erase.
Chip Erase complete.
1948160 bytes block 3868
Programming custom/spiplayer-sci-uart-button.img
1953242 bytes block 91
Programming Finished Successfully
3817 blocks programmed in 93 seconds,  20.51kB/sec
```

If the connection can not be established (pcmodflash.exe retries a number of times), break the execution by pressing ctrl-C and try again. If needed, disable boot by grounding GPIO0 and remove the grounding before running the programmer.

9.2 sdmodule, rtmidimodule, or other firmware without content

Example:

```
> pcmodflash -l sdmodule.spi -p 1 -sm 4
```

The above command specifies COM1 (-p 1) and the firmware file to program (-l sdmodule.spi).

The UART speed can be increased with the -sm option. Normally -sm 4 is the maximum that can be used (for 460800bps). Fortunately, above this speed the SPI FLASH write speed starts to be the limiting factor instead of the UART speed. When programming an image without content, speed is not generally an issue, because the boot image is small.

If the connection can not be established (pcmodflash.exe retries a number of times), break the execution by pressing ctrl-C and try again. Remember to disable boot by grounding GPIO0 and remove the grounding before running the programmer.

10 Application Examples

Note: these are compatible with the VS1000 Audio Module.

10.1 Headphone Connection

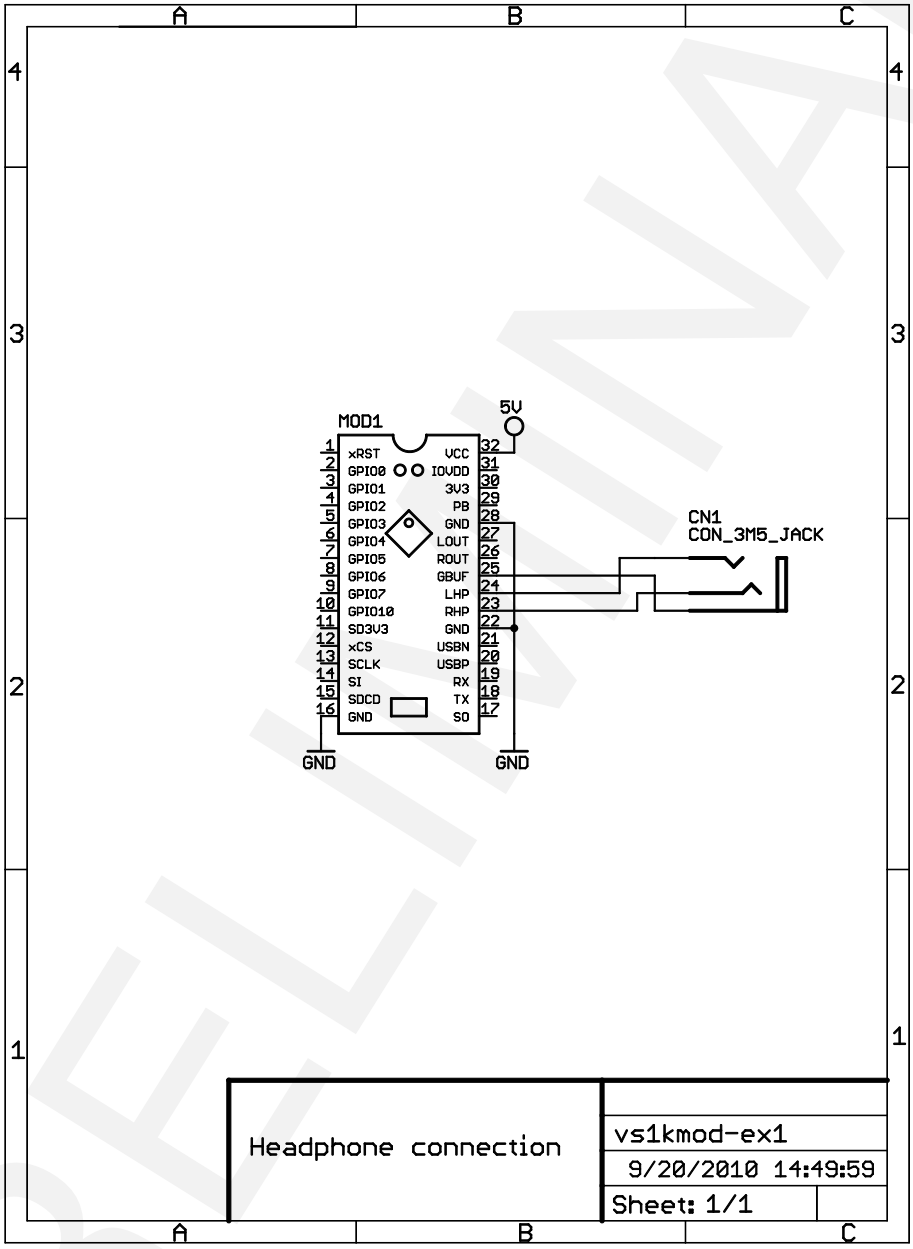


Figure 7: Headphone Connection.

The minimum connections required to get sound to headphones is shown above. (In addition you need ground and supply voltage.)

The firmware starts playing the first audio file from the SPI FLASH or μSD card (depending on the firmware) automatically after power-on.

10.2 Line Out Connection

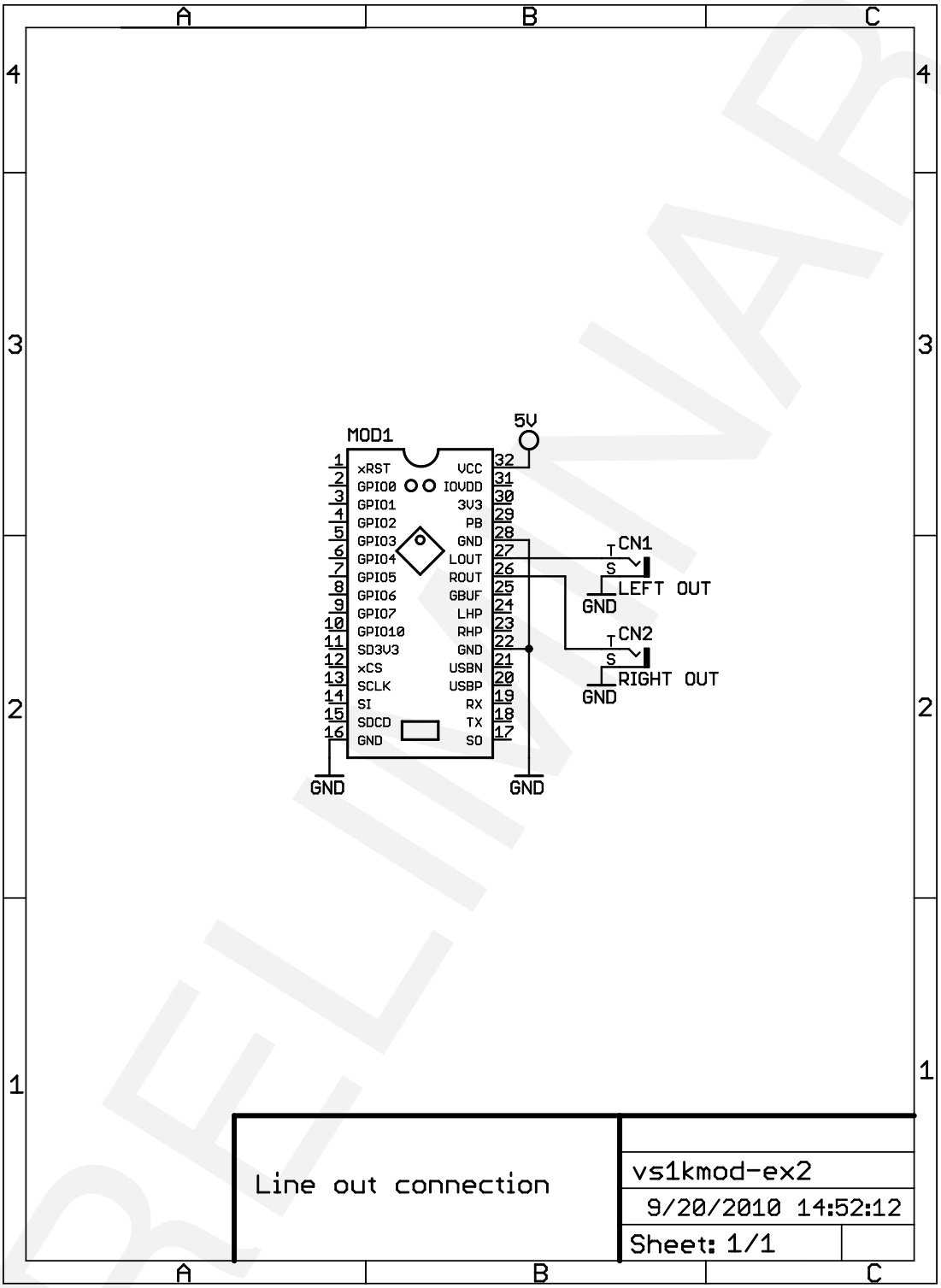


Figure 8: Line Out Connection.

The line outputs have suitable filtering and DC block, so that you can connect the pins to external devices.

10.3 UART Control

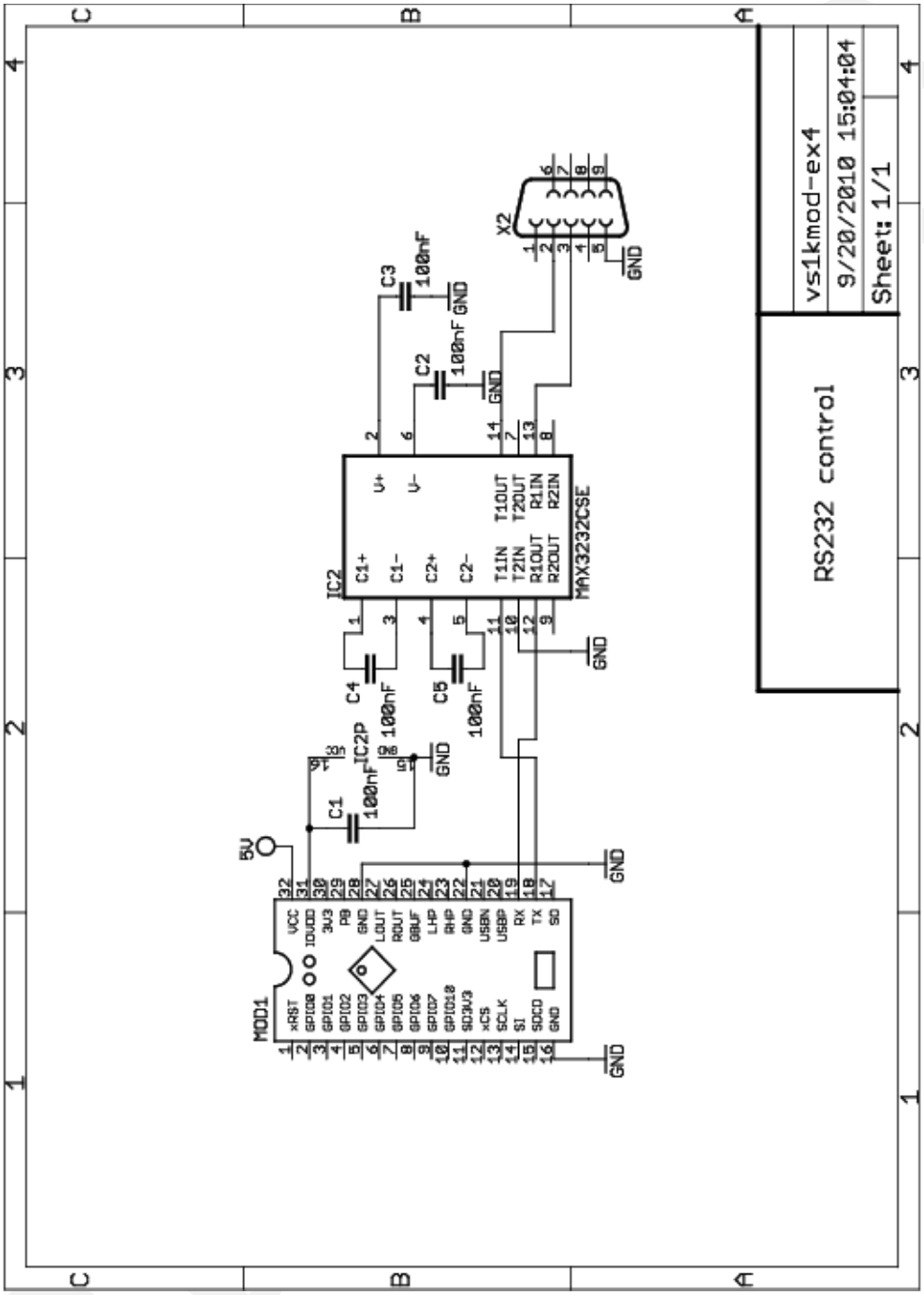


Figure 9: RS232 Control.

The TX and RX pins of the module are logic-level UART pins, so the use of the module with a RS232 interface requires an external RS232 adapter.

Alternatively you can use an USB-UART adapter which uses logic-level signals.

11 Document Version Changes

This chapter describes the most important changes to this document.

Version 0.52, 2015-05-13

- Small modification to pcmodflash.c and modprog.c to be able to use SPI FLASHes, which do not implement the REMS (0x90) command. (Use RDID 0x9f instead.) Also, pcmodflash.c now aligns the data sectors to 4kB so that recmodule is able to save (add to existing content) without first formatting the data area.

Version 0.51, 2014-11-13

- Preliminary datasheet.

12 Contact Information

VLSI Solution Oy
Hermiankatu 8 G
FIN-33720 Tampere
FINLAND

Phone: +358-3-3140 8222
Email: support@vlsi.fi
URL: http://www.vlsi.fi/