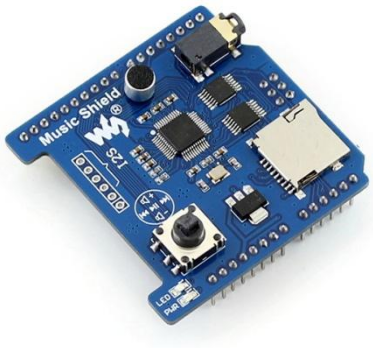
Music Shield User Manual

1. Overview

Music Shield is an Arduino expansion board for playing/recording audio, features onboard audio codec chip VS1053B and a TF card slot, supports common used audio formats.

Features:

- Arduino standard interfaces, compatible with Arduino boards like Arduino UNO, Leonardo, NUCLEO, XNUCLEO
- Supported formats : MP3/AAC/WMA/WAV/MIDI, etc.
- Onboard TF card slot, audio files in TF card can be played directly
- Onboard MIC for recording, with standard 3.5mm 4-segment headphone jack
- Onboard voltage level converter 74VHC125, compatible with 3.3V/5V MCUs
- One-key control, play music and adjust volume easily by the navigation key
- Features I2S and MIDI interface for functional expansion
- Note : Limited by the transmission speed and memory shortage, Arduino boards like UNO and Leonardo can neither play WAV files, nor use the recording function. NUCLEO or Due are recommended.



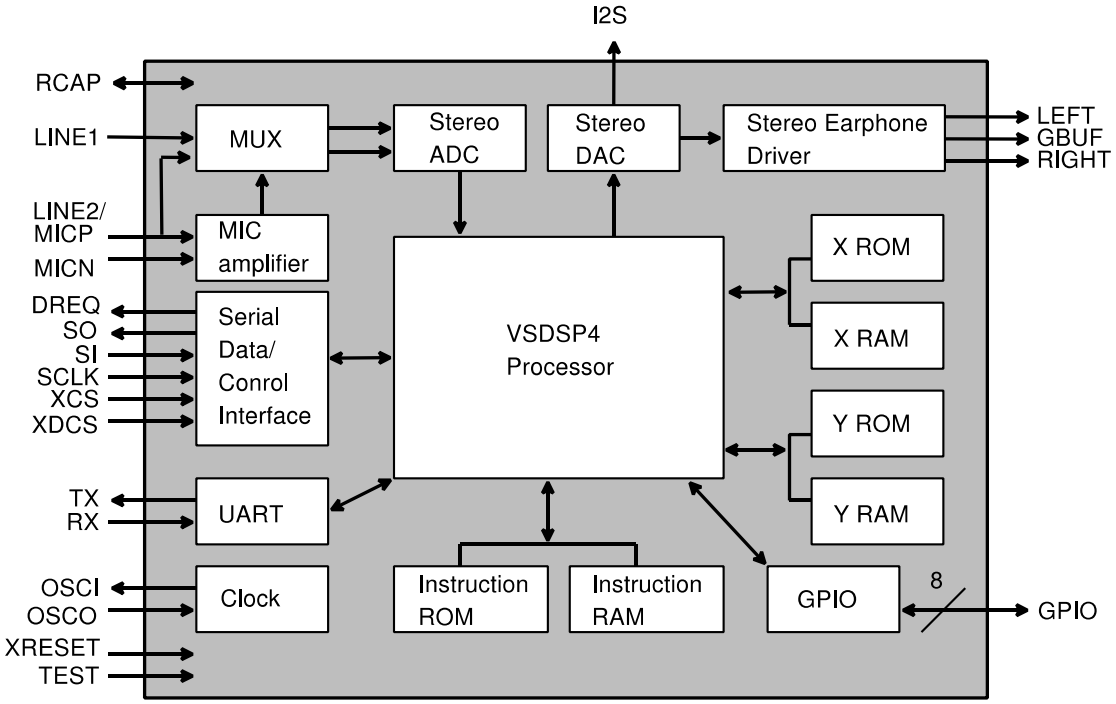
Note : Limited by the transmission speed and memory shortage, Arduino boards like UNO and Leonardo can neither play WAV files, nor use the recording function. NUCLEO or Due are recommended.

Revision 1.1. Date: December 7, 2015. Author: W. L. Yan. Editor: Felix.

2. Usage Descriptions

2.1. VS1053B

The Music Shield features a VS1053 audio decoder-and-encoder chip designed by VLSI from Finland, of which internal structure is shown in the following figure:



Control interface: the VS1053 communicates with a MCU through SPI, of which can be controlled by 7 data lines. They are:

XRSET	Active low asynchronous reset, schmitt-trigger input
XCS	Chip select input (active low)
XDCS	Data chip select (active low)
SI	Serial input
SO	Serial output
SCK	Clock for serial bus
DREQ	Data request, input bus. If DREQ is high, VS1053b can take at least 32 bytes of SDI data or one SCI command. DREQ is turned low when the stream buffer is too full and for the duration of a SCI command. At this point sending data or commands should be stopped.

Revision 1.1. Date: December 7, 2015. Author: W. L. Yan. Editor: Felix.

Audio output: The I2S pins and earphone are used for audio output from VS1053. The LEFT/RIGHT pads are used for Left/Right channel output. The GBUF pad is used for Common buffer for headphones. The I2S output pins (LOROUT, MCLK, SCLK, SDTA) are retained on the Music shield.

Recording: MICP/MICN, the differential mic input; LINE1/LINE2, the Line-in 1/2. The mic on Music Shield is connected to MICP/MICN and the earphone is connected to LINE2.

Note: please refer to the section ***Packages and Pin Descriptions*** of VS1053b datasheet.

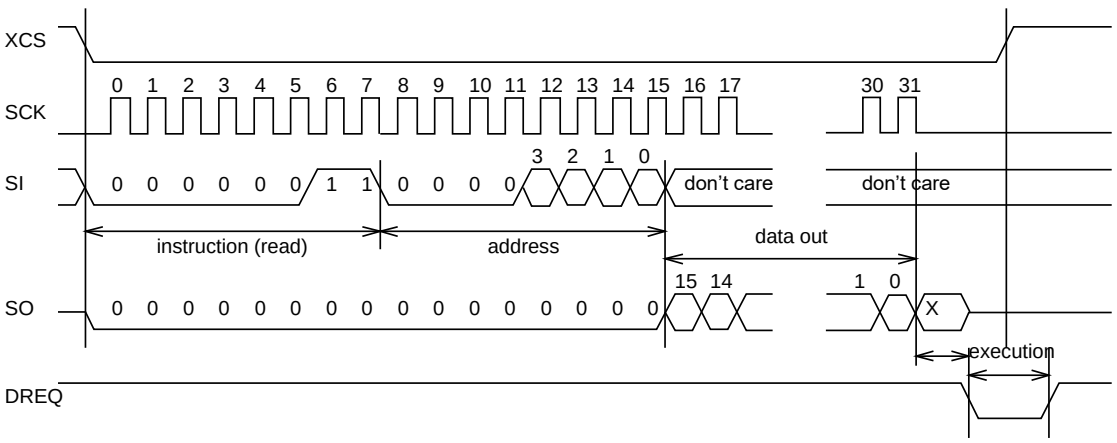
2.2. Serial Protocol for Serial Command Interface (SCI)

The serial bus protocol for the Serial Command Interface SCI (Chapter 8.6) consists of an instruction byte, address byte and one 16-bit data word. Each read or write operation can read or write a single register. Data bits are read at the rising edge, so the user should update data at the falling edge. Bytes are always send MSb first. XCS should be low for the full duration of the operation.

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

SCI Read



(See the Figure 6 of VS1053b datasheet)

VS1053b registers are read from using the following sequence. 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

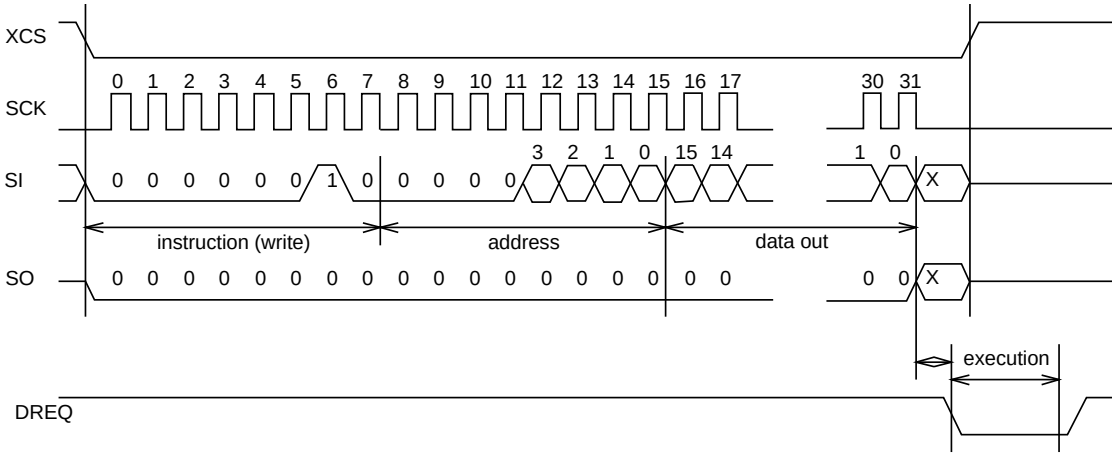
Revision 1.1. Date: December 7, 2015. Author: W. L. Yan. Editor: Felix.

address. After the address has been read in, any further data on SI is ignored by the chip. The 16-bit data corresponding to the received address will be shifted out onto the SO line.

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

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

SCI Write



(See the Figure 7 of VS1053b datasheet)

VS1053b registers are written from using the following sequence. 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.

2.3. SCI Register

VS1053 has 16 SCI registers in total. They are used to control the operation of VS1053b, as shown in the following figure:

SCI registers, prefix SCI_					
Reg	**Type**	**Reset**	**Time**	**Abbrev[bits]**	**Description**
0x0	rw	0x4800	80 CLKI	MODE	Mode control
0x1	rw	0x000C	80 CLKI	STATUS	Status of VS1053b
0x2	rw	0	80 CLKI	BASS	Built-in bass/treble control
0x3	rw	0	1200 XTALI	CLOCKF	Clock freq + multiplier

Revision 1.1. Date: December 7, 2015. Author: W. L. Yan. Editor: Felix.

0x4	rw	0	100 CLKI	DECODE_TIME	Decode time in seconds
0x5	rw	0	450 CLKI	AUDATA	Misc. audio data
0x6	rw	0	100 CLKI	WRAM	RAM write/read
0x7	rw	0	80 CLKI	WRAMADDR	Base address for RAM write/read
0x8	r	0	80 CLKI	HDAT0	Stream header data 0
0x9	r	0	210 CLKI	HDAT1	Stream header data 1
0xA	rw	0	80 CLKI	AIADDR	Start address of application
0xB	rw	0	80 CLKI	VOL	Volume control
0xC	rw	0	80 CLKI	AICTRL0	Application control register 0
0xD	rw	0	80 CLKI	AICTRL1	Application control register 1
0xE	rw	0	80 CLKI	AICTRL2	Application control register 2
0xF	rw	0	80 CLKI	AICTRL3	Application control register 3

Here are brief introductions about the main registers. (Please see the section ***SCI Registers*** of VS1053b datasheet for more details)

SCI_MODE Register: it is used to control the operation of VS1053b.

Bit	Name	Function	Value	Description
0	SM_DIFF	Differential	0 1	normal in-phase audio left channel inverted
1	SM_LAYER12	Allow MPEG layers I & II	0 1	no yes
2	SM_RESET	Soft reset	0 1	no reset reset
3	SM_CANCEL	Cancel decoding current file	0 1	no yes
4	SM_EARSPEAKER	EarSpeaker low setting	0 1	off active
5	SM_TESTS	Allow SDI tests	0 1	not allowed allowed
6	SM_STREAM	Stream mode	0 1	no yes
7	SM_EARSPEAKER	EarSpeaker high setting	0 1	off active
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	-	-	0 1	right wrong
14	SM_LINE1	MIC / LINE1 selector	0 1	MICP LINE1
15	SM_CLK	Input clock range	0 1	12..13 MHz 24..26 MHz

These bits are the primary focus of the present register:

2nd bit: SM_RESET Soft reset,

12th bit: SM_ADPCM ADPCM recording active,

14th bit: SM_LINE1 MIC / LINE1 selector.

SCI_BASS Register: it is used to control the sound effect of VS1053b.

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

The sound effect can be set by modifying the bit of the register.

SCI_CLOCKF Register: it is used to set the frequency, multiplier, etc, of the clock.

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

SCI_VOL Register: it is a volume control for the player hardware. The most significant byte of the volume register controls the left channel volume, the low part controls the right channel volume. The channel volume sets the attenuation from the maximum volume level in 0.5 dB steps. Thus, maximum volume is 0x0000 and total silence is 0xFEFE.

SCI_HDAT1/SCI_HDAT0 Register: After IMA ADPCM recording has been activated, registers SCI HDAT0 and SCI HDAT1 have new functions. The IMA ADPCM sample buffer is 1024 16-bit words. The fill status of the buffer can be read from SCI HDAT1. If SCI HDAT1 is greater than 0, you can read as many 16-bit words from SCI HDAT0. If the data is not read fast enough, the buffer overflows and returns to empty state.

2.4. Playing Audio Files

It is easy to use VS1053 as a hardware audio decoder to play audio files, only requires 3 steps to implement.

1) Reset VS1053

   The hardware and software reset at the boot up time and initialize the VS1053, preparing to decode the next music file.

2) Configure VS1053 registers

   The registers to be configured are Mode Register (SCI_MODE), Clock Register (SCI_CLOCKF) Bass Register (SCI_BASS), Vol Register (SCI_VOL Register), etc.

3) Transmit audio data

   If configuration complete, audio data can be transmitted to VS1053 constantly. VS1053 will decode the audio data, if decodable, until the audio data transmitting is finished. (Note: only if the DREQ is high can VS1053 receive data.)

2.5. Recording

The steps of Recording and playing are similar, first reset and then configure the related registers. The registers to be set mainly are:

Register	Bits	Description
SCI_MODE	2, 12, 14	Start ADPCM mode, select MIC/LINE1
SCI_AICTRL0	15..0	Sample rate 8000..48000 Hz (read at recording startup)
SCI_AICTRL1	15..0	Recording gain (1024 = 1×) or 0 for automatic gain control
SCI_AICTRL2	15..0	Maximum autogain amplification (1024 = 1×, 65535 = 64×)
SCI_AICTRL3	1..0 2 15..3	0 = joint stereo (common AGC), 1 = dual channel (separate AGC), 2 = left channel, 3 = right channel 0 = IMA ADPCM mode, 1 = LINEAR PCM mode reserved, set to 0

IMA ADPCM recording mode is activated by setting bits SM RESET and SM ADPCM in SCI MODE. Line input 1 is used instead of differential mic input if SM LINE1 is set.

Revision 1.1. Date: December 7, 2015. Author: W. L. Yan. Editor: Felix.

(The mic on Music Shield is connected to differential mic input while the earphone is connected to Line input 2.)

Before activating ADPCM recording, user must write the right values to SCI AICTRL0 and SCI AICTRL3. These values are only read at recording startup. SCI AICTRL1 and SCI AICTRL2 can be altered anytime, but it is preferable to write good init values before activation. See the section ***Activating PCM / ADPCM Recording Mode*** of VS1053b datasheet for more details.

If the registers have been configured correctly, the audio data can be read constantly. If SCI HDAT1 is greater than 0, you can read as many 16-bit words from SCI HDAT0. (See the section ***Reading PCM / IMA ADPCM Data*** of VS1053b datasheet)

Note: the recording data do not contain any RIFF header of the audio file. So to make your recording data a RIFF / WAV file, you have to add a header before the actual data. See the section ***Adding a PCM RIFF Header*** of VS1053b datasheet.

2.6. Real-Time MIDI

If GPIO0 is low and GPIO1 is high during boot, real-time MIDI mode is activated. In this mode the PLL is configured to 4.0×, the UART is configured to the MIDI data rate 31250 bps, and real-time MIDI data is then read from UART and SDI. Both input methods should not be used simultaneously. If you use SDI, first send 0xff and then send the MIDI data byte.

(The RX of UART on the Music Shield is connected to MIDI input.)

Note: If GPIO1 is not high, Real-Time MIDI can also be started with a small patch code using SCI.

3. Getting Started

3.1. Pin Descriptions

1) D11, D12 and D13 are the MISO, MOSI and SCLK buses of the SPI interface by default.
2) A0, A1, A2 and A3 are connected to the XRESET, DREQ, XDCS and XCS of VS1053.
3) D10 and D9 are the CS and SD_Detect pins of the SD card.
4) D7,D6,D5,D4 and D3 correspond to the button Down (Volume -), Left (Previous), Middle (Play/Stop/Record), Right (Next), Up(Volume +).
5) D8 is the LED indicator (it blinks when the module plays or records).

3.2. How to use

In this section, we will illustrate the applications of this shield by using 2 kinds of development boards.

3.2.1. XNUCLEO-F103RB（MCU STM32F103R）:

1. Playing Audio

1) Copy some audio files to the root directory of a TF card and insert it into the TF socket of a Music Shield.
2) Connect the development board to a PC.
3) Compile and program the Music Shield Player Demo.
4) Run the serial monitor software (PuTTY, SecureCRT, Arduino Serial Monitor, etc.) Set the software as: Baud: 9600, Data bits: 8, Stop bits 1; Parity: None, Flow control: None.
5) Plug in an earphone and you will hear the music. Press the Up/Down button to adjust the volume and Left/Right button to select the audio file. Press the Middle button to play or pause. Hold the Middle button to start recording (The Line In port is set as default recording input or you can edit the source code to set the onboard mic as a recording input) then press the Middle button to stop recording. The Serial Monitor software will print:

```
**********playing list**********
1 . test.wav
2 . test.aac
3 . test.flac
4 . test.mp3
5 . test.ogg
```


Revision 1.1. Date: December 7, 2015. Author: W. L. Yan. Editor: Felix.

```
6 . test.wma


Playing test.wav ...
Recording......
Press play button to stop.
Recording end
```

2. MIDI (SD card is not required under Real-Time MIDI)

1) Connect the development board to a PC.
2) Compile and program the Music Shield MIDI Demo.
3) Run the serial monitor software (PuTTY, SecureCRT, Arduino Serial Monitor, etc.) Set the software as: Baud: 9600, Data bits: 8, Stop bits 1; Parity: None, Flow control: None.
4) Plug in an earphone and you will hear the beautiful MIDI music. Meanwhile the Serial Monitor software will print:

```
Init vs10xx in MIDI format...
done
Fancy Midi Sounds
 Instrument: 30
N: 27
N: 28
N: 29
N: 30
N: 31
N: 32
N: 33
N: 34
N: 35
N: 36
N: 37
N: 38
N: 39
N: 40
```

3.2.2. Arduino UNO PLUS

1. Playing Audio (High quality or lossless music and recording function are not supported by UNO, due to the lack of speed and memory.)

1) Copy some audio files to the root directory of a TF card and insert it into the TF socket of a Music Shield.
2) Connect the development board to a PC.
3) Unzip the demo files to the directory of Arduino software: ..\arduino\libraries.
4) Run Arduino software and click File --> Examples --> MusicPlayer --> MusicShield to open the demo. Compile (click  , Verify) and program (click  , Upload) it.
5) Run the serial monitor software (PuTTY, SecureCRT, Arduino Serial Monitor, etc.) Set the software as: Baud: 115200, Data bits: 8, Stop bits 1; Parity: None, Flow control: None.
6) Plug in an earphone and you will hear the music. Press the Up/Down button to adjust the volume and Left/Right button to select the title. Press the Middle button to play or pause. Meanwhile the Serial Monitor software will print:

```
---- songs in TF card (root dir) ----
RECORD~1.WAV
TEST.ACC
TEST.MP3
TEST.OGG
TEST.WAV
TEST.WMA
Playing RECORD~1.WAV
[done!]
Playing TEST.WAV
```

2. MIDI (SD card is not required under Real-Time MIDI)

1) Connect the development board to a PC,
2) Run Arduino software and click File --> Examples --> MusicPlayer --> MusicShield to open the demo. Compile (click  , Verify) and program (click  , Upload) it.
3) Run the serial monitor software (PuTTY, SecureCRT, Arduino Serial Monitor, etc.) Set the software as: Baud: 115200, Data bits: 8, Stop bits 1; Parity: None, Flow control: None.
4) Plug in an earphone and you will hear the beautiful MIDI music. Meanwhile the Serial Monitor software will print:

```
Init vs10xx in MIDI format...done
Fancy Midi Sounds
 Instrument: 30
N: 27
N: 28
N: 29
N: 30
```

```
N: 31
N: 32
N: 33
N: 34
N: 35
N: 36
N: 37
N: 38
N: 39
N: 40
```

4. Revision history

Revision	Changes	Date
1.0	Initial release	June 26 2015