VS1053 USB Hi-Fi Player

VSMPG "VLSI Solution Audio Decoder"

Project Code:
Project Name: VSMPG

Revision History			
Rev.	**Date**	**Author**	**Description**
1.0	2009-11-09	PkP	Initial version

1 Description

The VS1053 USB Hi-Fi Player is a cost-efficient low component count MP3 + WMA + AAC + HEAAC + Ogg Vorbis + MIDI + WAV music player. It uses a single VS1053 DSP chip for audio decoding and a user interface.

Music files are stored in an SD card and a high speed USB interface (480 Mbit/s) is provided for file transfers. SDHC cards larger than 2 GB are supported.

Functionality of the player can be customized using a boot EEPROM.

Figure 1.1: VS1053 USB Hi-Fi Player Prototype

2 Usage

When the device is not connected to USB, it works like an MP3 player. When connected to USB, it works as an SD-card reader/writer and charges the internal Li-Ion battery.

2.1 Playing a File

When powered up, the player starts to play automatically all songs in supported file formats from all directories in the SD-card.

2.1.1 Default Key Mappings

The figure below describes the key mappings used in the default software. Top line describes operation of a short press and bottom line describes operation of a long press.

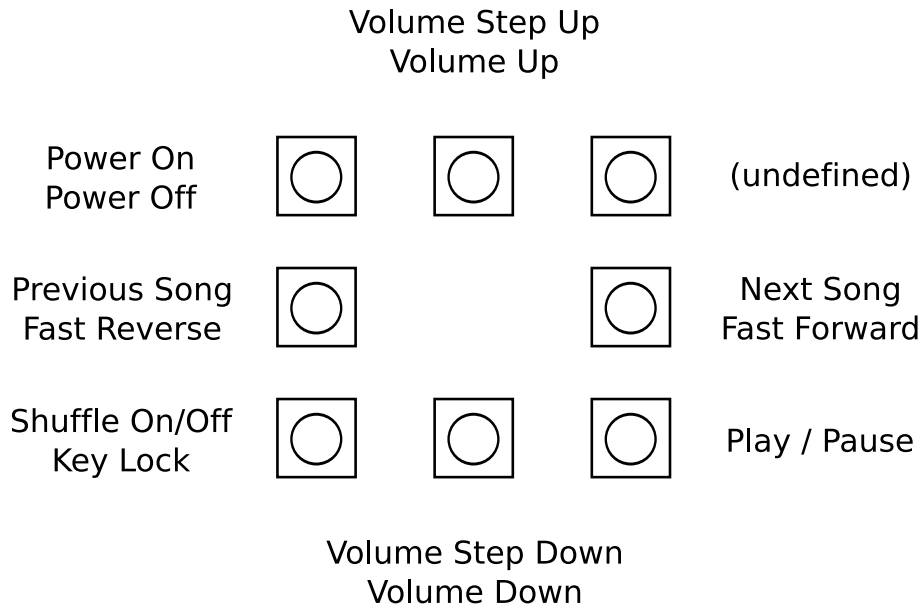

Figure 2.1: Default software keys

The key functions can be changed by modifying the software.

2.1.2 Supported File Formats

Format	Extension	Status
MP1	.mp1	Supported (selectable by software)
MP2	.mp2, .mp3	Supported (selectable by software)
MP3	.mp3	Supported
WMA	.wma	Supported
AAC, LC-AAC	.aac, .m4a	Supported
HE-AAC	.aac, .m4a	Supported
Ogg Vorbis	.ogg	Supported
PCM	.wav	Supported
IMA ADPCM	.wav, .ima	Supported
General Midi 1	.mid	Supported (fmt0:streamable)
FLAC	.flac	Support can be added with software

2.2 Display Options

The device can be made 1) without display, 2) with a low cost LCD display or 3) with an OLED display, as depicted in the picture.

2.3 Memory Options

The device usually has an SD card slot for storing music. The SD card can be replaced with internal NAND flash + an SD card controller chip (CBM3082).

2.4 Recording

The card contains an electret microphone connected to the mic input of VS1053. The default software does not support recording. Recording to WAV file can be added by modifying the software.

2.5 Battery Voltage Measurement

The battery voltage can be measured by reading ADC input 2 ("Line In R"). For example of battery measurement see function `ScreenShowVolts()` in the default software.

The default software shuts down the player when battery voltage drops below 3.6 volts for 10 consecutive measurements.

2.6 USB Operation

The card has an external SD card reader IC (CBM4082) for data interface between the PC and the SD card.

A Li-Ion battery charger chip (LTC4054) is used for charging the on-chip battery when connected to USB.

IMPORTANT! Battery charging is performed by a chip not designed by VLSI Solution Oy. While VLSI Solution has tested and measured battery charging and have no reason to believe there are problems with it, VLSI Solution takes absolutely no responsibility over the suitability or safety of it.

2.6.1 Connecting USB

Signal `VBUSDET` on the PCB tells the VS1053 that USB is connected. This forces signal `SD_#ON` to automatically be pulled high. A 3-state buffer (74AC125) then disconnects the SD card from VS1053's SPI bus.

USB voltage power us the SD card reader chip, which now controls the data interface between the PC and the SD card.

2.6.2 Disconnecting USB

When USB is disconnected, the software uses signal `SD_#ON` to toggle power of the SD card to reset and restart the SD card. Player operation then continues normally.

3 Firmware Update

The firmware can be modified and recompiled using the `vskit 1.34` version of the windows command line tools that are available from `http://www.vlsi.fi` .

First unzip vskit 1.34 to folder c:\vskit134.
Then unzip the firmware to folder c:\vskit134\vs1053usb.

3.1 Compiling the Code

The code can be compiled in a Windows command shell by using the `make` utility. This can be done by entering the commands:

```
C:\> cd vskit134
C:\vskit134> cd vs1053usb
C:\vskit134\vs1053usb> c:\vskit\bin\make
```

The make utility runs the compiler (vcc), assembler (vsa), linker (vslink) and image writer programs and makes an eeprommable image `boot1053b.img`. You should see a lot of text output, including lines like:

```
vcc -P130 -O -fsmall-code -D RECORD_FS=8000 -Ilibc16 -Ih1053 -I.  -D VS1053
-D ENABLE_UI -o main1053but.o standalone.c
...
Successfully compiled 4860 lines (3044 in source file)
...
player1053bbut.bin:
C 3255 CF 0 X 777 Y 128 F 0
total  code memory size 0x0cb7  3255 words
total     X memory size 0x0309   777 words
total     Y memory size 0x0080   128 words
...
```

If you don't get this kind of output, you should add the folder c:\vskit134\bin in your computer's PATH environment.

3.2 Writing the Boot Image to the SPI Flash

The file `boot1053b.img` can be written to an SPI flash using an external eeprommer or an RS-232 adapter.

3.2.1 The RS-232 adapter

To update the firmware using RS-232, connect the RS-232 adapter to pins marked "- T R +" on the bottom of the device. Connect the RS-232 cable to port COM1 in the PC.

If you use a USB-RS232 adapter in the PC, configure the adapter to use port COM1. If COM1 cannot be used, you need to change the "-P n" parameter in each call to `vs3emu` in `Makefile` and the `*.bat` files. By default, vs3emu is called with parameter "`-P 1`". If your COM port number is, for example, COM5, you need to change the parameter to "`-P 5`".

If you don't have an RS-232 adapter from VLSI, you can build it yourself using a chip such as MAX3232.

The pin descriptions are:

Pin	Direction	Description
-	Ground	Ground
T	Output	TX Data from device (to pin RX in PC)
R	Input	RX Data to device (from pin TX in PC)
+	Supply output	Supply 3V for the RS-232 adapter

3.2.2 Writing the SPI flash using RS-232

The file `prom24.bat` contains a script to write the flash. It can be run with the command "`prom24 boot1053b.img`". To update the firmware, use the following procedure:

1. Detach USB and remove Battery
2. Connect Programming Jumper
3. Connect Battery
4. Press Power button and keep it pressed
5. Enter command "PROM24 BOOT1503B.IMG"
6. Wait until programming is finished
7. Stop pressing Power Button
8. Remove Programming Jumper

When the update is successful, you see the following output:

```
c:\vskit134\vs1053usb> prom24 boot1053b.img

VSEMU 2.1 Oct  1 2008 13:34:08(c)1995-2007 VLSI Solution Oy
Using serial port 1, COM speed 9600
Waiting for a connection to the board...
Chip version "1053"
Stack pointer 0x1920, bpTable 0x4f56
User program entry address 0x50
Speed changed to 38400
prom24.bin: includes optional header, 22 sections, 760 symbols
Section 1: abs_x       page:1 start:6272 size:192 relocs:0 fixed
  ...other sections...
Section 22: VS_stdiolib0  page:0 start:993 size:134 relocs:37

Erasing Boot Block...
ffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
(should be ffff)... Done. Programming...
.............................
Finished!!

  A2  : 0x10  A1  : 0x1010  A0  : 0x1010
  ...other register dump values...
  PC  : 0x00000000
Next Exec: 0x0000 LDC 0x0,A0
```

3.3 Modifying the Firmware

The default firmware of the player is provided as "C" language files. The main code of the application is in file `standalone.c`. Other files provide support functions. Some of the functions are written in assembly language for speed and size optimization.

The decoders for MP3, WMA, AAC, WAV, IMA, Ogg Vorbis and MIDI are in the chip ROM. Source code for the decoders is NOT available. They can be called only.

3.3.1 Modifying the Key Functions

The key functionality is located in file `standalone.c`. First part of function `UserHook` reads the GPIO pins and sets variable `ui.Data`.

Second part of the function examines the current key state `ui.Data` and the previous

key state `ui.Key` and implements the functionality accordingly. Please see the comments inside `standalone.c` file.

Timing Source

The flag variable `ui.Strobe` is set around 16 times each second and can be used for user interface timing.

3.3.2 Modifying the Display Contents

In the beginning of `standalone.c`, there are a number of functions for interfacing to a graphical LCD or OLED display. For example, there is a font table, which is used by the default firmware screens.

The function `ScreenUpdate()` draws a new secreen when there is free CPU time. Please see its implemention in `standalone.c`.

3.3.3 Debugging the Code with RS-232

When RS-232 is used, you can set the define USE_DEBUG to enable console debugging. You can use functions like `puts(''Hello'')` and `fputs(''Hello'',stdout)` for writing to the PC screen.

It is also possible to use `fopen()` to open files inside the PC for reading and writing.

Be sure to remove the debugging code before trying to run the code without RS-232!