

## VS10xx STANDALONE PLAYER

### VSMPPG “VLSI Solution Audio Decoder”

Project Code:  
Project Name: VSMPPG

Revision History			
Rev.	Date	Author	Description
1.01	2005-11-07	PO	Minor changes, see section 6 for details.
1.00	2005-09-30	PO	<b>SCHEMATICS CHANGED!</b> See also chapter 2 and section 2.1
0.95	2005-09-16	PO	SCI-version fixes, transfer routine fixes, skips non-audio files, startup delay
0.94	2005-08-24	PO	8.3-char filename in SCI version, loudness defaults, subdirectory support, partial FAT12 support fix for MMC-related powerup problem (chapter 2)
0.93	2005-07-25	PO	Version for VS1023 added
0.92	2005-07-13	PO	Version for VS1011E added, schematics fixed
0.91	2005-06-22	PO	SCI-controlled version upgrades (more to come)
0.9	2005-06-07	PO	Fixed user hook init for SCI-version and some MMC powerup problems
0.8	2005-05-27	PO	Fixed third button connection in the concept pic
0.7	2005-05-18	PO	Third button and optional LED added
0.6	2005-05-12	PO	Prototyping board picture added
0.5	2005-05-09	PO	Initial version

# 1 VS10xx Standalone Player

The SPI bootloader that is available in VS1011E, VS1002D, VS1003B, and VS1023 can be used to add new features to the system. Patch codes and new codecs can be automatically loaded from SPI EEPROM at startup. One interesting application is a single-chip standalone player.

The standalone player system uses MMC directly connected to VS10xx using the same GPIO pins that are used to download the player software from the boot EEPROM.

The instruction RAM of 1280 words (5 kilobytes) is used for MMC communication routines, read-only handling of the FAT and FAT32 filesystems and a simple three-button user interface.

Standalone Features:

- **No microcontroller is required**, boots from SPI EEPROM (25LC640).
- Low-power operation
- Uses MMC for storage. Hot-removal and insertion of MMC is supported.
- Supports FAT and FAT32 filesystems, **including subdirectories** (upto 16 levels). FAT12 is partially supported: only unfragmented files are allowed for FAT12.
- Automatically plays the MMC contents from the first file after power-on.
- Power-on defaults configurable.
- VS1011E/VS1002D transfer speed 4.1 Mbit/s (24.576 or  $2 \times 12.288$  MHz clock).
- VS1003B/VS1023 transfer speed 4.8 Mbit/s ( $3.5 \times 12.288$  MHz clock).
- High transfer speed supports even 48 kHz 16-bit stereo WAV files.
- Optional three-button interface allows pause/play, random play and loudness toggle, song selection, and volume control.
- Optional LED for user interface feedback

With Optional Microcontroller:

- Optional external microcontroller can control the player through SCI.
- Bypass mode allows MMC to be accessed also directly by the microcontroller.
- Code can be loaded through SCI by a microcontroller to eliminate SPI EEPROM.
- If code is loaded by a microcontroller, also VS1011B can be used.

## 2 Boot EEPROM and MMC

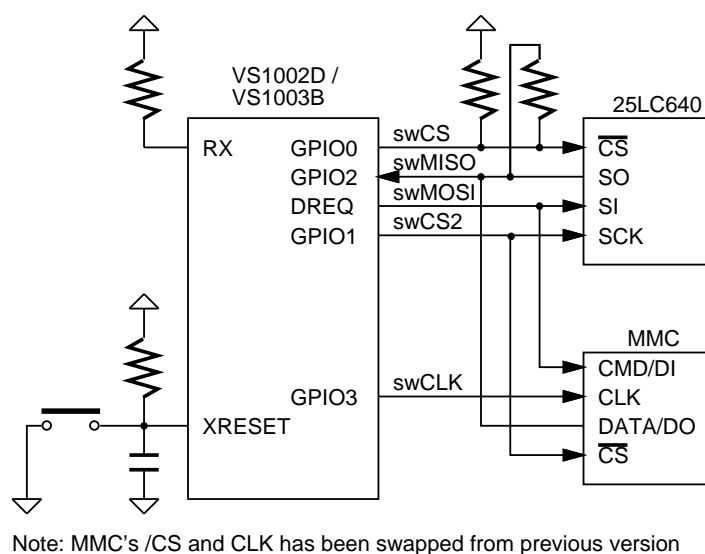


Figure 2.1: SPI-Boot and MMC connection

The standalone player software is loaded from SPI eeprom at power-up or reset when GPIO0 is tied high with a pull-up resistor. The memory has to be an SPI Bus Serial EEPROM with 16-bit addresses. The player code currently requires almost 5 kB, thus at least 8 kB SPI EEPROM is recommended.

SPI boot redefines the following pins (**note: MMC pin assignment has changed**):

Pin	SPI Boot	Other
GPIO0	swCS (CS for SPI EEPROM)	100 k $\Omega$ pull-up resistor
GPIO1	<b>swCS2 (CS for MMC)</b>	Also used as SPI clock during boot
DREQ	swMOSI	
GPIO2	swMISO	100 k $\Omega$ between xSPI and swMISO
GPIO3	<b>swCLK (CLK for MMC)</b>	Data clock for MMC

The SPI EEPROM boot is used for the button-controlled standalone player. The code for the SCI-controlled player is uploaded through the SCI instead of using an SPI EEPROM.

Defective or partially defective MMC cards can drive the CMD (DI) pin until they get the first clock. This interferes with the SPI boot if MMC's drive capability is higher than VS10xx's. So, **if you have powerup problems when MMC is inserted, you need a 330  $\Omega$  resistor between swMOSI (DREQ) and MMC's CMD/DI pin.**

Because the SPI EEPROM and MMC share pins, it is crucial that MMC does not drive the pins while VS10xx is booting. MMC boots up in mmc-mode, which does not care about the chip select input, but listens to the CMD/DI pin. Mmc-mode commands are protected with cyclic redundancy check codes (CRC's). Previously it was assumed that when no valid command appears in the CMD pin, the MMC does not do anything. However, it seems that some MMC's react even to commands with invalid CRC's, which messes up the SPI boot.

The only way to cure this problem is to change how the MMC is connected. The minimum changes are achieved by swapping MMC's chip select and clock inputs. This way MMC does not get clocked during the SPI boot and the system should work with all MMC's. But notice that the swap only occurs on the MMC pins, do not change the SPI EEPROM connection!

## 2.1 Fixing Existing VS10xx Prototyping Board 1.5

Since the 1.00 version of the standalone player MMC's /CS and CLK have been swapped. This change has no effect elsewhere in the design. However, to be able to use 1.00 or later player version, you need to use the new pin assignments.

Fixing the VS10xx Prototyping Board 1.5 is easy (if it has not been done for you already). Remove the CS and CK jumpers from JP15 / JP17 (pins 1 and 3). Then connect pin 1 of JP15 to pin 3 of JP17, and pin 1 of JP17 to pin3 of JP15. And that's all.

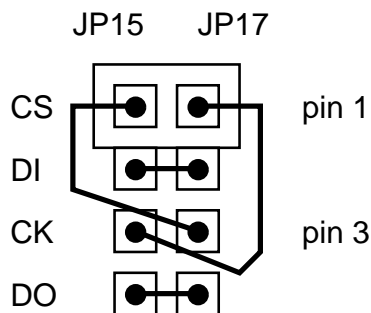


Figure 2.2: VS10xx Prototyping Board 1.5 Fix

## 3 Player with Three-Button UI

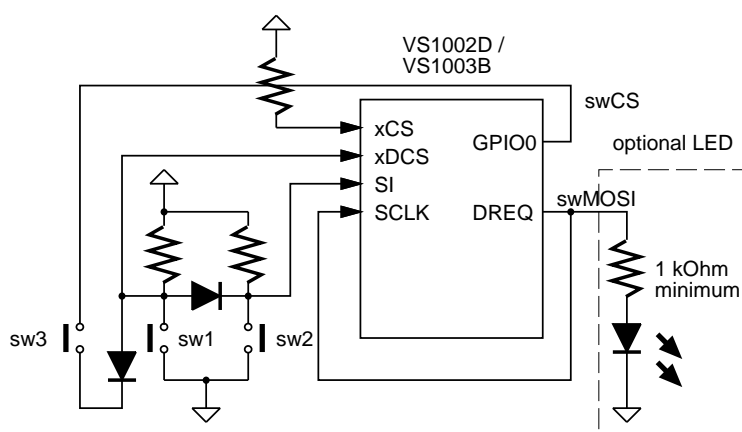


Figure 3.1: Three-button interface connection

A three-button interface is implemented with two diodes and two resistors. Only one button is detected simultaneously. If both SW1 and SW2 are pressed, only the other one (SW2) is detected. SW3 is only read if neither SW1 nor SW2 are pressed.

The three-button interface provides the most needed controls.

Button	Short Keypress	Long Keypress
SW1	Next song	Volume up
SW2	Previous song	Volume down
SW3	Pause/Play	Play mode: Toggle loudness Pause mode: Toggle random play

Very little changes to the user interface are possible, because of the very restricted instruction RAM availability.

An optional LED can be used for indicating system activity. In play mode a long blink of the LED indicates loudness ON, in pause mode a long blink indicates random play ON. Otherwise the LED shows MMC activity. In pause mode the LED lights up dimly.

Notice that SCI and SDI can not be used simultaneously with the three-button interface.

### 3.1 Boot Images

The SPI EEPROM boot images can be found from the `code/` subdirectory. **Note that this application is highly chip-specific. It only works on the exact firmware versions mentioned.** Note that to be able to use 1.00 or later player version, you need to use the new MMC pin assignments (see chapter 2 and section 2.1).

For VS1003B you can also select a version that does not play WMA files. If you use that version in your product, a WMA license should not be required.

Chip	File	Features
VS1011E	player1011ebut.bin	Three-button interface
VS1002D	player1002but.bin	Three-button interface, watchdog
VS1003B	player1003but.bin	Three-button interface, watchdog
VS1003B	player1003nwbut.bin	Three-button interface, watchdog, No WMA
VS1023	player1023but.bin	Three-button interface, watchdog

### 3.2 Power-on Defaults

Default values are loaded from SPI EEPROM at power-on reset. Before the MMC/SD card is first accessed after power-on, approximately 22 ms delay is executed. The startup delay time can be changed from the boot image. The middle bytes in the string 0x00 0x12 0x34 0x0e contain the default value 0x1234 (22 ms). This value can be changed between 0x0000 (0 ms) and 0x3fff (80 ms). Do not change the 0x00 and 0x0e bytes.

The input clock is assumed to be 12.288 MHz. If you want to use a different crystal, the SCLCLOCKF value can be found from byte offsets 10 and 11 in the boot image. The default values are 0x9800 ( $2 \times 12.288$  MHz) for VS1011E and VS1002D, and 0xa000 ( $3.5 \times 12.288$  MHz) for VS1003B and VS1023. You can reduce the VS1003B/VS1023 power consumption a bit by using 0x8800 ( $3.0 \times .. 3.5 \times 12.288$  MHz).

Volume (SCLVOL) default value is in byte offsets 26 and 27. Loudness default is in byte offsets 32 and 33 (treble and bass controls, respectively). The bass control value should be odd to make the loudness indicator LED blink work. SCLBASS default value is in byte offsets 8 and 9.

If you want the loudness ON by default, replace bytes 8 and 9 in the image with the same values you use as the loudness default in offsets 32 and 33.

Offset	Register	Default	Meaning
8, 9	SCLBASS	0x0000	Bass enhancer control at power-up
10, 11	SCLCLOCKF	0x9800	Clock control (for VS1003B 0xa000)
26, 27	SCLVOL	0x2020	Power-up volume, left and right channel
28, 29	SCLAICTRL0	0	Song number to play at power-up
32, 33	SCLAICTRL2	0x33d9	Treble and bass control for loudness
34, 35	SCLAICTRL3	0	Play mode & Miscellaneous configuration

## 4 SCI-Controlled Version

If the button-interface is not used, the player can be controlled through the serial control interface (SCI). In this mode xCS, SI, SO, and SCLK are connected to the host controller's SPI bus and xDCS should have a pull-up resistor.

The code is loaded through SCI by the microcontroller. The application loading tables for the microcontroller are available in the `code/` subdirectory. To start the application after uploading the code, write 0x30 to SCI\_AIADDR (SCI register 10).

Chip	File	Features
VS1011B	player1011bsci.c	SCI control
VS1011E	player1011esci.c	SCI control
VS1002D	player1002sci.c	SCI control, watchdog
VS1003B	player1003sci.c	SCI control, watchdog
VS1003B	player1003nwsci.c	SCI control, watchdog, No WMA
VS1023	player1023sci.c	SCI control, watchdog

All non-application SCI registers can be used normally, except that SM\_SDINEW must be set to '1' to enable GPIO2 and GPIO3 (default for VS1002 / VS1003 / VS1023). SCI\_CLOCKF must be set by the user, preferably before starting the code.

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	Number of songs on MMC
0xE	AICTRL2	Memory read in SCI version
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  $0x8000 + \text{song number}$  to SCI\_AICTRL0 to jump to another song. 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\_AICTRL1 contains the number of songs (files) found from the MMC card.

You can use SCI\_AICTRL2 to read VS10XX memory. First write the address ( $0..0x3fff$  for X memory,  $0x4000..0x7fff$  for Y memory) to SCI\_WRAMADDR, then read from SCI\_AICTRL2. The first read will return carbage, the second one returns the value from the address you specified. Each read increments the internal address so you can get data from consecutive addresses with consecutive reads from SCI\_AICTRL2.

In VS1003/VS1023 and VS1011E you can use SCI\_WRAMADDR and SCI\_WRAM to both write and read memory and you don't need to use SCI\_AICTRL2.

SCI_AICTRL3 bits		
Name	Bit	Description
CLTR3_PAUSE_ON	4	0=normal, 1=pause ON
CLTR3_FILE_READY	3	if PLAY_MODE=2, 1=file found
CLTR3_PLAY_MODE_MASK	2:1	0=normal, 1=loop song, 2=pause before play
CLTR3_RANDOM_PLAY	0	0=normal, 1=random play

If the lowest bit of SCI\_AICTRL3 is 1, a random song is selected each time a new song starts. Note that this is random play, not shuffle play. Shuffle plays all songs in random order, while random play can select the same song multiple times.

The play mode mask bits can be used to change the default play behaviour. 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.

*Pause before play* mode will first locate the file, then go to pause mode. Both CTRL3\_PAUSE\_ON and CTRL3\_FILE\_READY will be set. When the user has read the file ready indicator, he should reset the 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.

AICTRL3 should be set to the desired play mode by the user before starting the code.



## 4.1 Reading the 8.3-character Filename

When a file has been selected, the MSDOS short filename (8+3 characters) can be read from VS10xx memory. The filename is in Y memory at addresses 0x780..0x785 (VS1011B, VS1011E, VS1002D) or 0x1800..0x1805 (VS1003B, VS1023). The first character is in the most-significant bits of the first word.

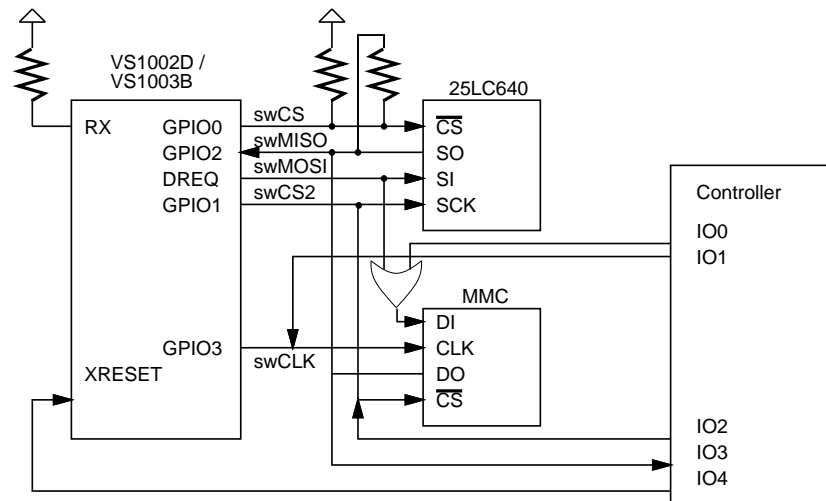
The following pseudocode tries to locate a file named "SONG.MP3". If it is found, it is played continuously in a loop.

```
#define MKWORD(a,b) (((int)(unsigned char)(a)<<8)|(unsigned char)(b))
int song = 0;
WriteMp3Reg(SCI_AICTRL3, (2<<1)); /* pause before play mode */
WriteMp3Reg(SCI_AICTRL0, 0x8000+song); /* select song */
while (1) {
    if (ReadMp3Reg(SCI_AICTRL3) & (1<<3)) { /* file ready */
        unsigned short ch[6], name[6] = {
            MKWORD('S','0'), MKWORD('N','G'), MKWORD(' ',' '),
            MKWORD(' ',' '), MKWORD('M','P'), MKWORD('3','\0')};
        int i;

        WriteMp3Reg(SCI_WRAMADDR, 0x4780); /* 0x5800 for VS1003B/VS1023 */
        ReadMp3Reg(SCI_AICTRL2); /* dummy read */
        for (i=0; i < 6; i++) { /* read filename */
            ch[i] = ReadMp3Reg(SCI_AICTRL2); /* first 2 characters */
            printf("%c%c", ch[i]>>8, ch[i]);
        }
        ch[5] &= 0xff00; /* mask away unused bits */
        printf("\n");
        if (!memcmp(ch, name)) { /* compare filenames */
            break; /* filename matched, leave loop */
        } else {
            /* the right file not found!! */
            if (++song == ReadMp3Reg(SCI_AICTRL1)) {
                /* The requested file was not on the card! */
            } else {
                /* clear file ready, keep pause on, pause before play mode */
                WriteMp3Reg(SCI_AICTRL3, (1<<4)|(2<<1));
                WriteMp3Reg(SCI_AICTRL0, 0x8000+song); /* select next song */
            }
        }
    }
}
/* SONG.MP3 file number is now in the variable 'song' */
/* clear file ready and pause, select loop song mode */
WriteMp3Reg(SCI_AICTRL3, (1<<1));
```

## 4.2 Bypass Mode

VS10xx can be disconnected from MMC to allow direct microcontroller access by keeping it in reset. This leaves GPIO pins as inputs, but connecting microcontroller to MMC's DI needs an OR between DREQ and controller's I/O pin. This is because DREQ is always an output pin.



Note: MMC's /CS and CLK has been swapped from previous version  
The connection of MMC's DI depends on the state of DREQ:  
if VS10xx is kept in reset during access, use an OR port (DREQ=0)

Figure 4.1: Example of shared access

Another way to disconnect VS10xx from MMC is keeping GPIO0 low when reset is deasserted. This bypasses the SPI-boot, leaving GPIO pins as inputs. SM\_SDINEW must be '1' (the default in VS1002/VS1003). DREQ rises when normal firmware is ready. In this case an AND between DREQ and the controller's I/O pin is required.

Because the latter bypass mode is actually the normal firmware operation mode, the controller can use VS10xx through SCI and SDI normally, for example for audio cues while accessing the MMC.

## 5 Example Implementation

The standalone player was implemented using a brand-new VS10xx prototyping board.

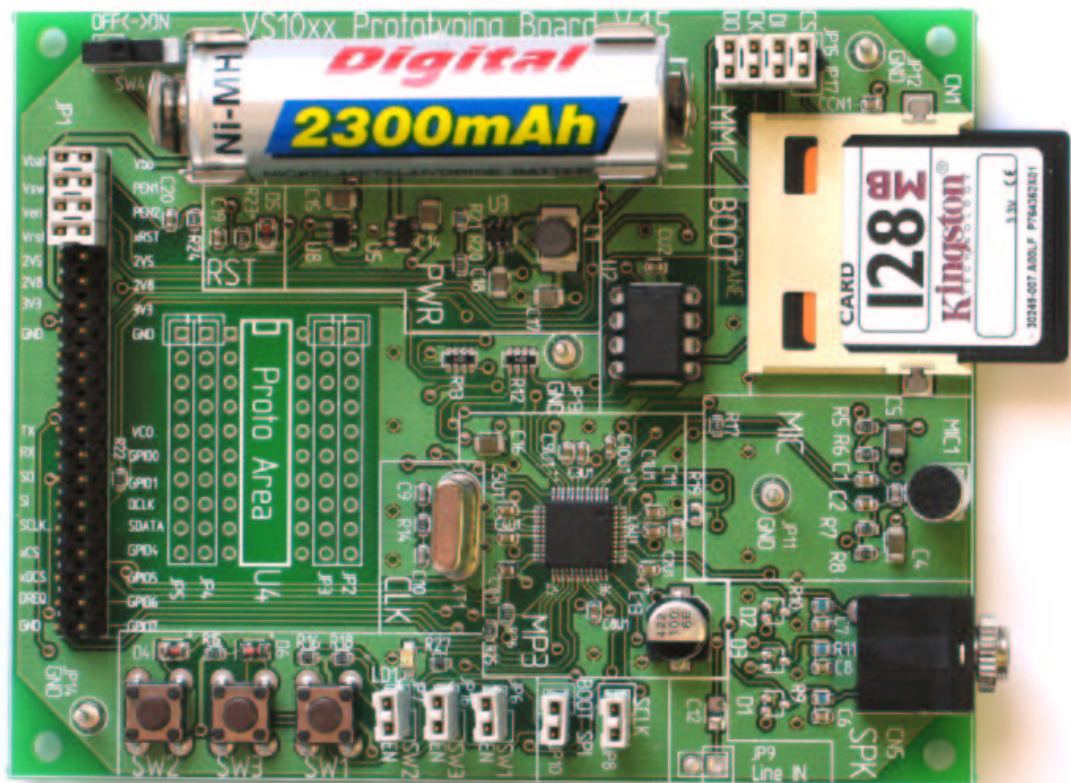
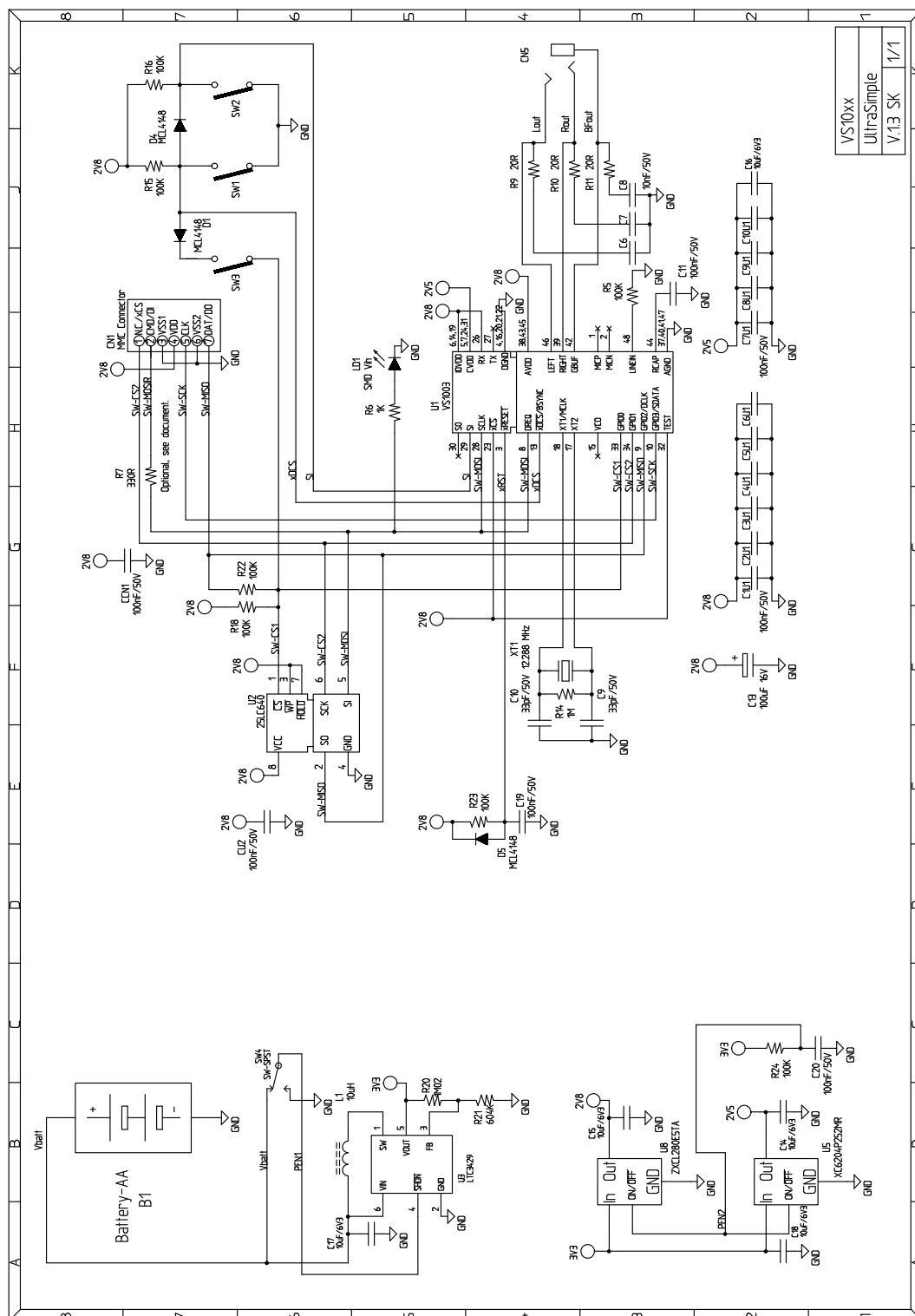


Figure 5.1: Standalone Player in Prototyping Board

The following example schematics contains a simple implementation for VS1003B. Power generation and player logic are separated. **Note: the schematics is a stripped-down version of the Prototyping Board. Use the attached schematics only as a basis for your own designs and refer to the Prototyping Board schematics when you work with the Prototyping Board.**

VS1002D version differs by having no separate IOVDD and CVDD, and no line input. VS1011E has no microphone input either. Note that the current implementation does not support microphone or line input. When using VS1003B, the line input should have a pull-down resistor. The microphone input is self-biased and does not need to be connected.



Note: MMC's /CS and CLK swapped. Optional resistor fixes problems with some MMC's (chapter2). See also Figure 3.1.

Battery life was tested with the prototyping board:

Test conditions:

- VS1002D
- Kingston 128 MB MultiMediaCard
- Beyerdynamic DT 131 headphones connected all of the time
- a single 2300 mAh AA-sized NiMH 1.2V Rechargeable Battery
- 128 kbps MP3 song (`utopia-free-sample.mp3`) on autorepeat
- default volume (-16dB)
- no LED

Test result: 25 hours 27 minutes of play time.

## 6 Document Version Changes

### 6.1 Version 1.01, 2005-11-07

- MMC communication timing changed slightly to support SanDisk MMC's.
- Song numbering with IMA-ADPCM playback fixed - only VS1002D was affected.
- Watchdog was not activated in the SCI-controlled version 1.00. Note that for SCI-controlled version you need to reactivate the player after watchdog reset.
- Added chapter 7 to explain the playing order of files.

### 6.2 Version 1.00, 2005-09-30

- **SCHEMATICS CHANGED!**
  - MMC's CLK and /CS have been swapped. Everything else is the same.
  - MMC no longer interferes with SPI boot on powerup.
  - The swap can be done in existing boards with two wires in JP15 and JP17. See section 2.1.
- VS1011E/VS1002D transfer speed increased to 4.1 Mbit/s
- Support for FAT12/FAT16 and FAT32 disks that have no partition block.
- Random play can now select all of the supported 32768 files.

### 6.3 Version 0.95, 2005-09-16

- Power-on startup delay (default 22 ms) added to reduce power consumption spike.
- If nothing playable is found in the first 300 kB of a file, the rest is skipped. This takes approximately 3 seconds per file.
- SCI-controlled version fixes:
  - Initialization of SCI registers left to user (BASS,CLOCKF,VOL)
  - SCI control fixed: SCI interrupts were only enabled in vs1011 version
- Transfer routine fixes:
  - Internal timing changed to support more MMC/SD cards
  - Transfer speed reduced to 4.78 Mbit/s in VS1003B/VS1023 versions

## 6.4 Version 0.94, 2005-08-24

- FAT 8.3-character filename readable in SCI-controlled version (see section 4.1)
- Subdirectories are now supported for FAT32 and FAT16 filesystems
  - Upto 16 levels of subdirectories can be used
  - Files and subdirectories are played in the order they appear in the filesystem structures
- Partial FAT12 support:
  - Unfragmented files in FAT12 root directory are supported

## 7 Playing Order

The playing order of files is not the same order as how they appear in Windows' file browser. The file browser sorts the entries by name and puts directories before files. It can also sort the entries by type, size or date. The standalone player does not have the resources to do that. Instead, the player handles the files and directories in the order they appear in the card's filesystem structures.

Normally the order of files and directories in a FAT filesystem is the order they were created. If files are deleted and new files added, this is no longer true. Also, if you copy multiple files at once, the order of those files can be anything. So, if you want a specific play order: 1) only copy files into an empty card, 2) copy files one at a time in the order you like them played.

There are also programs like LFNSORT that can reorder FAT16/FAT32 entries by different criteria. See "<http://www8.pair.com/dmurdoch/programs/lfnsort.htm>" .

The following picture shows the order in which the player processes files. First DIR1 and then DIR2 has been created into an empty card, then **third.jpg** is copied, DIR3 is created and the rest of the files have been copied. **song.mid** was copied before **start.wav**, and **example.mp3** was copied before **song.mp3** because they appear in their directories first.

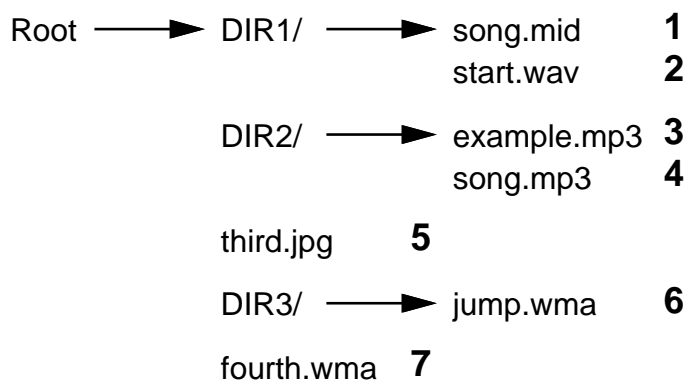


Figure 7.1: Play Order with subdirectories

Because DIR1 appears first, all files in it are processed first, in the order they are located inside DIR1, then files in DIR2. Because **third.jpg** appears in the root directory before DIR3, it is next, then files in DIR3, and finally the last root directory file **fourth.wma**.



If DIR2 is now moved inside DIR3, the playing order changes as follows.

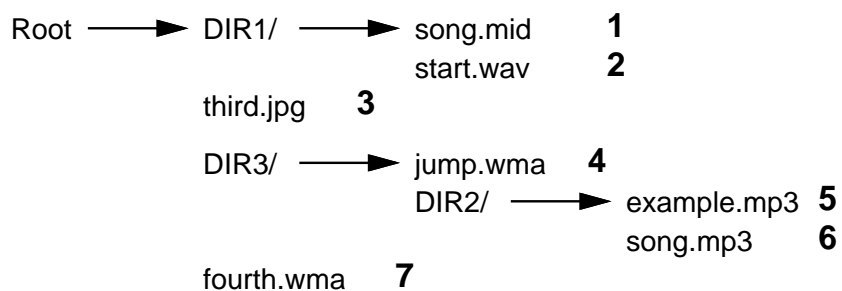


Figure 7.2: Play Order with nested subdirectories

Unsupported filetypes like Word documents and PDF files are also processed, but as they contain no decodable data, the decoder shortly gives up and moves to the next entry.