

VS10XX STANDALONE PLAYER

VSMPG “VLSI Solution Audio Decoder”

Project Code:
Project Name: VSMPG

Revision History			
Rev.	Date	Author	Description
1.20	2017-11-13	PO	Various fixes.
1.19	2009-10-50	PO	Volume and current song saving for vs1053.
1.18	2009-08-14	PO	Open file by name, loop mode fixes, I2S for VS1053.
1.17	2009-01-15	PO	Fix for VS1053.
1.16	2008-12-22	PO	Faster file open with lot of directories. Shuffle play.
1.15	2008-02-25	PO	SCI-Controlled Recorder, SCI+UART control, VS1103b and VS1053b player versions.
1.14	2007-07-09	PO	High-Capacity SD support added (not to recorder).
1.13	2007-01-23	PO	Drivers kept inactive until MMC found. SCI-version fixes. VS1033c version added.
1.12	2006-07-13	PO	Recorder updates. MMC communication changed.
1.11	2006-05-08	PO	More fragments allowed. SCI-version fixes.
1.10	2006-03-10	PO	Recorder bug fix, samplerate changed to 8000 Hz.
1.10pre	2006-01-13	PO	First Recorder Implementation.
1.02	2005-11-17	PO	File type detection from filename suffix. Pulldown resistor recommended for GPIO3.
1.01	2005-11-07	PO	Minor changes, see section 6 for details.
1.00	2005-09-30	PO	SCHEMATICS CHANGED! 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)
0.5	2005-05-09	PO	Initial version

Contents

VS10xx Standalone Player Front Page	1
Table of Contents	2
1 VS10xx Standalone Player	3
2 Boot EEPROM and MMC	4
2.1 Fixing Old VS10xx Prototyping Board 1.5	5
3 Player with Three-Button UI	5
3.1 Boot Images	7
3.2 Power-on Defaults	7
4 SCI-Controlled Player	8
4.1 UART Control	12
4.2 Reading the 8.3-character Filename	13
4.3 Bypass Mode	14
5 Example Implementation	15
6 Document Version Changes	18
7 Playing Order	22

List of Figures

1	SPI-Boot and MMC connection	4
2	VS10xx Prototyping Board 1.5 Fix	5
3	Three-button interface connection	6
4	SCI connection	8
5	Example of shared access	14
6	Standalone Player in Prototyping Board	15
7	Play Order with subdirectories	22
8	Play Order with nested subdirectories	23

1 VS10xx Standalone Player

All information in this document is provided as-is without warranty. Features are subject to change without notice.

The SPI bootloader that is available in VS1011E, VS1002D, VS1003B, VS1033C, VS1053B, and VS1103B 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 application uses MMC/SD 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/SD/SDHC for storage. Hot-removal and insertion of card is supported.
- Supports FAT and FAT32 filesystems, **including subdirectories** (upto 16 levels). FAT12 is partially supported: subdirectories or fragmented files are not allowed.
- Automatically starts playing from the first file after power-on.
- Power-on defaults are configurable.
- VS1011E/VS1002D transfer speed 4.1 Mbit/s (24.576 or 2×12.288 MHz clock¹).
- VS1003B/VS1033C/VS1053B transfer speed 4.8 Mbit/s (3.5×12.288 MHz clock).
- High transfer speed supports even 48 kHz 16-bit stereo WAV files.
- Optional three-button interface allows pause/play, shuffle play and loudness toggle, song selection, and volume control.
- Optional LED for user interface feedback

¹ Because MMC communication takes some CPU time, 320 kbit/sec MP3 files need higher than 12.288 MHz clock in VS1011E/VS1002D if both bass enhancer and treble control are active.

With Optional Microcontroller:

- External microcontroller can control the player through SCI or UART.
- 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.

2 Boot EEPROM and MMC

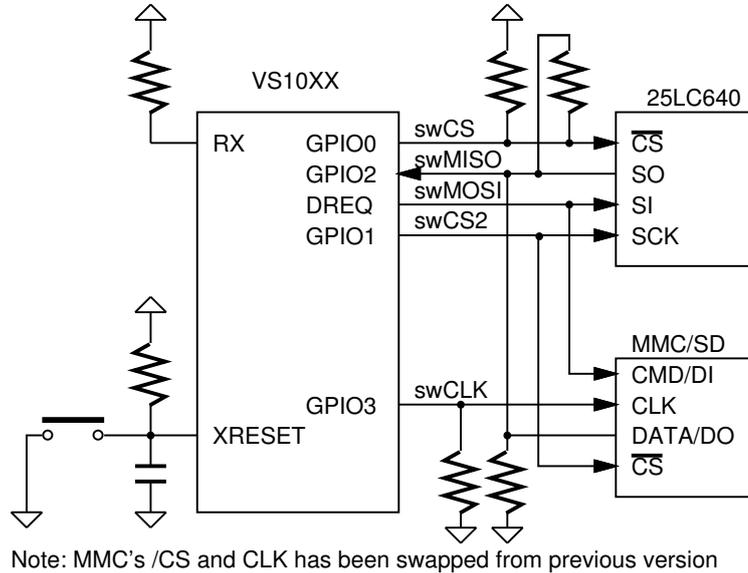


Figure 1: SPI-Boot and MMC connection

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 and MMC/SD usage redefines the following pins:

Pin	SPI Boot	Other
GPIO0	swCS (EEPROM XCS)	100 kΩ pull-up resistor
GPIO1	swCS2 (MMC XCS)	Also used as SPI clock during boot
DREQ	swMOSI	
GPIO2	swMISO	100 kΩ between xSPI & swMISO, 680 kΩ to GND
GPIO3	swCLK (MMC CLK)	Data clock for MMC, 10 MΩ to GND

Pull-down resistors on GPIO2 and GPIO3 keep the MMC CLK and DATA in valid states on powerup.

The SPI EEPROM boot is used for the button-controlled standalone player. The code for the SCI-controlled player can be 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 something like a 330 Ω resistor between swMOSI (DREQ) and MMC's CMD/DI pin.** Normally this resistor is not required.

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 was to change how the MMC is connected. The minimum changes were 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. Because the swap only occurred on the MMC pins, the SPI EEPROM connection is unchanged!

2.1 Fixing Old 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.

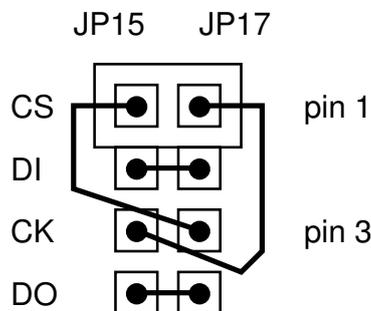


Figure 2: VS10xx Prototyping Board 1.5 Fix

3 Player with Three-Button UI

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.

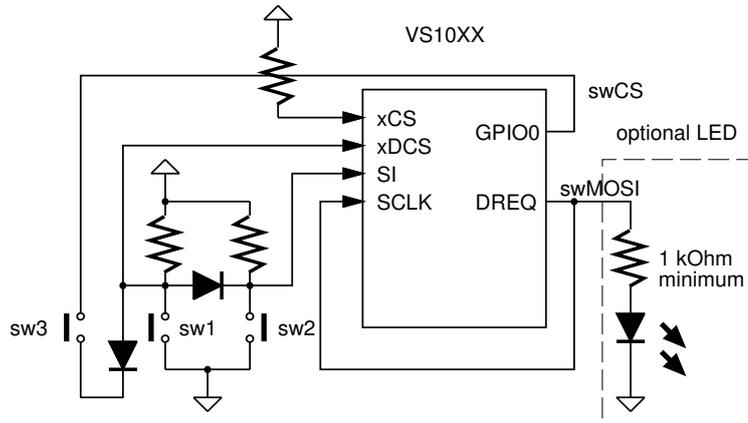


Figure 3: Three-button interface connection

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 shuffle 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 shuffle 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
VS1033C	player1033cbut.bin	Three-button interface, watchdog
VS1053B	player1053bbut.bin	Three-button interface, watchdog
VS1103B	player1103bbut.bin	Three-button interface, watchdog

3.2 Power-on Defaults

Default values are loaded from SPI EEPROM at power-on reset.

The input clock is assumed to be 12.288 MHz. If you want to use a different crystal, the `SCI_CLOCKF` value can be found from byte offsets 10 and 11 in the boot image. The default values are 0x9800 (2×12.288 MHz) for VS1011e and VS1002d, and 0xa000 (3.5×12.288 MHz) for VS1003b, VS1033c, and VS1103b, and 0x8000 (3.5×12.288 MHz) for VS1053b. You can reduce the power consumption a bit by lowering the default clock and allowing the clock add (see chip datasheet for details).

Volume (`SCI_VOL`) 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. `SCI_BASS` 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	<code>SCI_BASS</code>	0x0000	Bass enhancer control at power-up
10, 11	<code>SCI_CLOCKF</code>	0x9800	Clock control (for VS1003B/33C 0xa000)
26, 27	<code>SCI_VOL</code>	0x2020	Power-up volume, left and right channel
28, 29	<code>SCI_AICTRL0</code>	0	Song number to play at power-up
32, 33	<code>SCI_AICTRL2</code>	0x33d9	Treble and bass control for loudness
34, 35	<code>SCI_AICTRL3</code>	0	Play mode & Miscellaneous configuration

In VS1103b `SCI_AICTRL2` is used for the number of songs, so loudness setting is not available.

4 SCI-Controlled Player

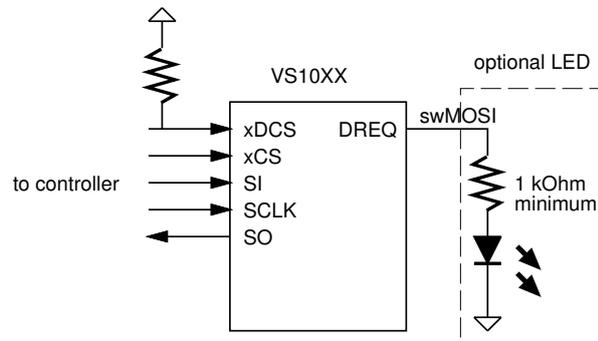


Figure 4: SCI connection

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. xDCS should have a pull-up resistor. The connection from SCLK to DREQ is not used, that connection is only used for reading the buttons in the 3-button user interface.

Normally the code is loaded through SCI by the microcontroller. In this case the boot EEPROM can be eliminated, and the pull-up resistor in GPIO0 can be changed into a pull-down resistor. Because the SCI/SDI connection is available, the VS10XX chip can be used also normally in slave mode. When standalone playing from MMC/SD is wanted, the code is loaded and started through SCI. Software or hardware reset returns the chip to slave mode.

The application loading tables for the microcontroller are available in the `code/` subdirectory. To start the application after uploading the code, write 0x30 (0x50 for VS1053b) to SCI_AIADDR (SCI register 10). Before starting the code, you should initialize SCI_CLOCKF and SCI_VOL.

Chip	File	Features
VS1011E	player1011esci.c	SCI control
VS1002D	player1002sci.c	SCI+UART control, watchdog
VS1003B	player1003sci.c	SCI+UART control, watchdog
VS1003B	player1003nwsci.c	SCI+UART control, watchdog, No WMA
VS1033C	player1033csci.c	SCI+UART control, watchdog
VS1053B	player1053bsci.c	SCI+UART control, watchdog
VS1103B	player1103bsci.c	SCI+UART control, watchdog

All non-application SCI registers can be used normally, except that SM_SDINew must be kept at '1' to enable GPIO2 and GPIO3 (default for other chips than VS1011e). 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	HDATA0	Stream header data 0
0x9	HDATA1	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	Number of songs on MMC (VS1103b)
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_AICTRL1 (SCI_AICTRL2 for VS1103b) 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 AICTRL1 will contain 0x7fff after MMC/SD has been successfully initialized.

In other chips than VS1002d you can use SCI_WRAMADDR and SCI_WRAM to both write and read memory. With VS1002d you can use SCI_WRAMADDR and SCI_WRAM to write memory and SCI_AICTRL2 to read memory.

Write the address you want to read (0..0x3fff for X memory, 0x4000..0x7fff for Y memory) to SCI_WRAMADDR, then read from SCI_AICTRL2. The first read will return garbage, 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.

SCI_AICTRL3 bits		
Name	Bit	Description
CTRL3_UPDATE_VOL	15	'1' = update volume (for UART control)
CTRL3_I2S_ENABLE	9	Enable I2S output, VS1053 only
CTRL3_BY_NAME	8	'1' = locate file by name
CTRL3_AT_END	6	if PLAY_MODE=3, 1=pause at end of file
CTRL3_NO_NUMFILES	5	0=normal, 1=do not count the number of files
CTRL3_PAUSE_ON	4	0=normal, 1=pause ON
CTRL3_FILE_READY	3	1=file found
CTRL3_PLAY_MODE_MASK	2:1	0=normal, 1=loop song, 2=pause before play, 3=pause after play
CTRL3_RANDOM_PLAY	0	0=normal, 1=shuffle play

AICTRL3 should be set to the desired play mode by the user before starting the code. If it is changed during play, care must be taken.

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.

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. CTRL3_FILE_READY will be set to indicate a file was found and playing has started, but it will not be automatically cleared.

Pause before play mode will first locate the file, then go 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 will play 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 $0x8000 + \text{song number}$ 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 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 AICTRL1 (SCI_AICTRL2 for VS1103b) will contain $0x7fff$ 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.

Since the 1.18 version, you can open specific files by using the CTRL3_BY_NAME bit.

You should first set pause mode bit CTRL3_PAUSE_ON and the open-by-name bit CTRL3_BY_NAME in AICTRL3 (and CTRL3_NO_NUMFILES), 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 check 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 0x4780 (0x5800 for VS1003B/VS1033C/VS1053B) 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. Only uppercase characters are used.

With VS1053 you can use CTRL3_I2S_ENABLE to activate the I2S output. GPIO4 to 7 are then configured as I2S output pins, MCLK output is enabled, and 48 kHz output rate is selected (with 12.288 MHz XTAL).

SCI-Controlled Player with SPI Boot

If your microcontroller does not have enough memory for the code loading tables, the SCI-controlled version can also be loaded from SPI-EEPROM. Then the SCI register default values are also loaded from EEPROM. You can change the power-on defaults in the same way by editing the boot image than in the standalone player version.

Chip	File	Features
VS1011E	player1011esci.bin	SCI control
VS1002D	player1002sci.bin	SCI+UART control, watchdog
VS1003B	player1003sci.bin	SCI+UART control, watchdog
VS1003B	player1003nwsci.bin	SCI+UART control, watchdog, No WMA
VS1033C	player1033csci.bin	SCI+UART control, watchdog
VS1053B	player1053bsci.bin	SCI+UART control, watchdog
VS1103B	player1103bsci.bin	SCI+UART control, watchdog

If you want to use the chip in normal slave mode also with the SPI EEPROM, change the GPIO0 pull-up resistor into a pull-down resistor. This prevents automatic boot after reset, and the chip stays in normal slave mode.

To start the SCI-controlled standalone player, write 0xC017 to SCI_WRAMADDR, then 0x0001, 0x0000, and 0x0001 to SCI_WRAM. This sets GPIO0 to output a '1'. Then give a software reset. The chip now detects GPIO0 high, and performs boot from SPI EEPROM.

To return to slave mode either give a hardware reset, or write 0xC017 to SCI_WRAMADDR, then 0x0000 to SCI_WRAM, and give a software reset.

4.1 UART Control

The SCI-Controlled Player (and recorder) also supports 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 from VS10xx. When UART control is used, the code is loaded from SPI EEPROM and SCI connection is not needed. Loading the code through UART is possible, but complicated, so that will not be available unless there is serious demand.

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.

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 have the most significant bit cleared. The last byte has the most significant bit set, and the register number is 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);
    putch((value>>7) & 127);
    putch(value & 127);
    putch(reg | 0x80);
}

```

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 in all VS10xx chips 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). However, VS1053B uses the new SCI_VOL value automatically.

To force volume calculation in chips that need it, 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.

Byte/TX	Status
0x65 'e'	song/recording ended
0x70 'p'	song paused
0x63 'c'	play continued
0x6c 'l'	song looped
0x72 'r'	recording started

Some status information is also returned. 'e' is returned after a song or recording 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.

4.2 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, VS1033C). 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/VS1033C */
        ReadMp3Reg(SCI_AICTRL2); /* dummy read - required for VS1002 only */
        for (i=0; i < 6; i++) { /* read filename */
            ch[i] = ReadMp3Reg(SCI_WRAM); /* first 2 chars, AICTRL2 for VS1002 */
            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));
```

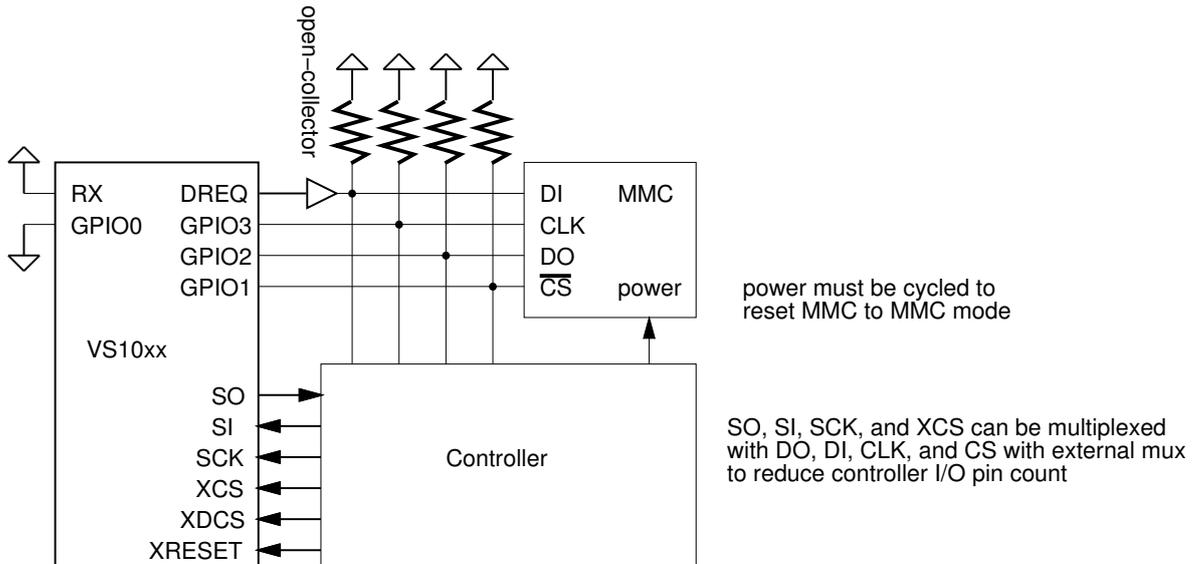
4.3 Bypass Mode

VS10xx can be disconnected from MMC to allow direct microcontroller access. A good way to disconnect VS10xx from MMC is keeping GPIO0 low when reset is deasserted (software reset can also be used). This bypasses the SPI-boot, leaving GPIO pins as inputs. SM_SDINew must be '1', this is the default in VS1002/VS1003/VS1033. DREQ rises when normal firmware is ready. In this case an open-collector driver is used to connect DREQ and the controller's I/O pin to MMC's DI-pin.

Because this 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. The controller can upload the SCI-controlled standalone player through SCI and start it whenever it wants.

Because the MMC can not be returned to MMC mode without power cycling, the controller needs a way to power off the MMC.

Concept connection diagram for SCI-controlled standalone player when code is loaded through SCI.



To start playing:

- 1) Cycle MMC power to reset it to default state
- 2) Reset VS10xx – DREQ will rise when boot complete
- 3) Upload the code from controller to VS10xx through SCI
- 4) Start the code, VS10xx accesses the MMC
- 5) The player can be controlled through SCI commands

Note: controller pins connected to MMC must be high-impedance state

To access MMC from controller:

- 1) hardware (XRESET) or software-reset (through SCI) VS10xx
- 2) DREQ rises when boot complete, GPIO's remain high-impedance
- 3) Cycle MMC power to reset it to default state
- 4) Access MMC with controller in either MMC or SPI mode

Figure 5: Example of shared access

5 Example Implementation

The standalone player was implemented using the VS10xx prototyping board.

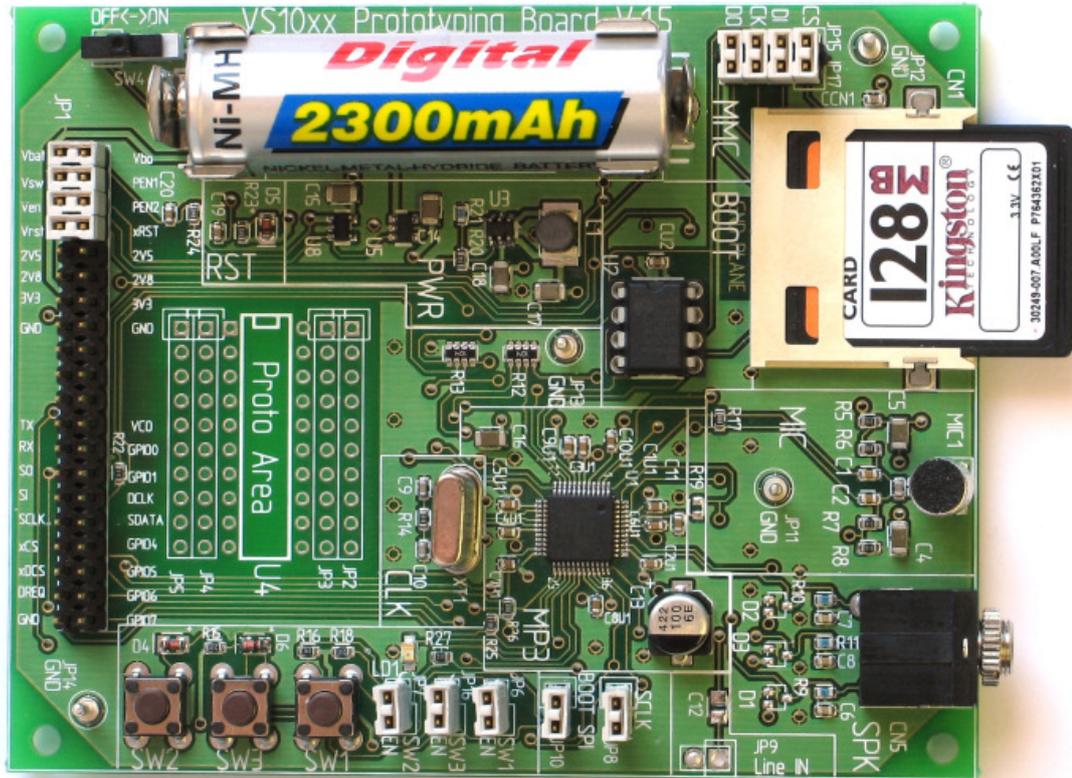
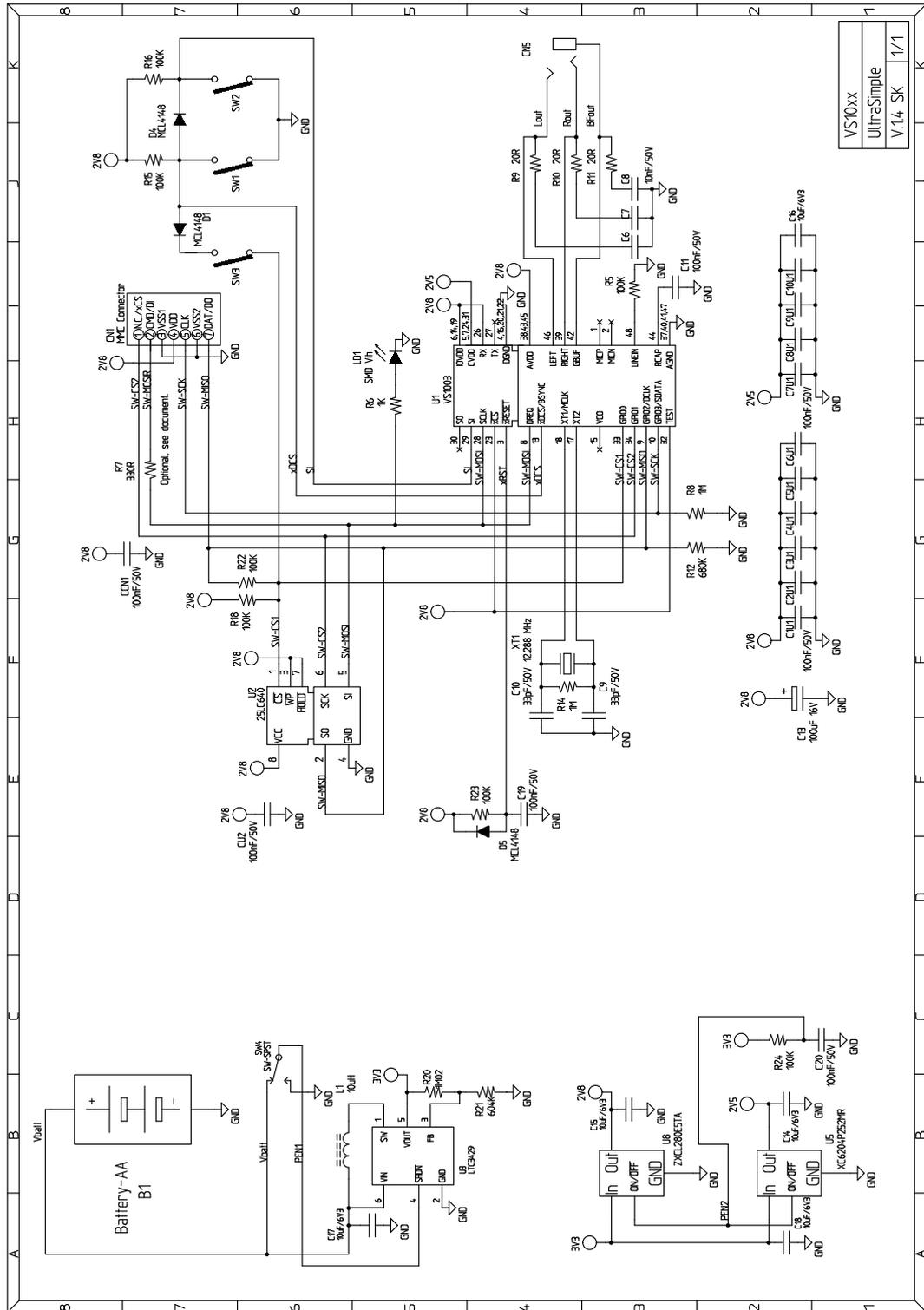


Figure 6: Standalone Player in Prototyping Board

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: **MMC's /CS and CLK swapped.** Optional resistor fixes problems with some MMC's (chapter2). See also Figure 3.

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

Version 1.20, 2017-11-13

- Fixed open-by-name with vs1003b.
- Removed recorder. See vs1053b or vs1063a standalone players.

Version 1.19, 2009-10-05

- VS1053 saves volume setting and current song to EEPROM.
- Mention of startup delay was removed.

Version 1.18, 2009-08-14

- Filename read example uses SCI_WRAM (SCI_AICTRL2 with VS1002 only).
- UART control echoes the received characters. Also more robust when song change happens in the middle of a UART command.
- Open file by name possible in SCI-controller player.
- Fixes to loop mode in SCI-controlled player.
- SDHC support was not enabled with VS1053.
- CTRL3_I2S_ENABLE bit added to AICTRL3 (VS1053 only).

Version 1.17, 2009-01-15

- VS1053: IROM switching did not restore the default state after playing midi.

Version 1.16, 2008-12-22

- Going through all files was slow with lot of directories and a large cluster size. The new version detects end of directory immediately instead of at the end of a cluster.
- Random play changed to shuffle play (not in VS1002).

Version 1.15, 2008-02-25

- SCI-controlled version:
 - *pause after play* mode was added.
 - *file ready* flag is set regardless of play mode when file is found.
 - *no number of files* flag (bit 5) disables the file count, AICTRL1 will have 0x7fff after MMC/SD has been initialized correctly. This speeds up the play of a file.

- When new file has been selected by writing 0x8000+song to AICTRL0, *pause on* and *file ready* bits are automatically cleared.
- UART control added.
- Clarifications to the SCI-controlled version. A picture added.
- SCI-controlled recorder added. It has also UART control.
- VS1053b and VS1103b player versions added. Recorder not available yet.
- Matched suffixes for VS1103b: .MID .WAV
- Matched suffixes for VS1053b: .MP3 .WAV .MID .WMA .WMV .ASF .AAC .MP4 .M4A .3GP .3G2 .OGG

Version 1.14, 2007-07-09

- High-Capacity SD card support added to 3-button and SCI-controlled standalone players, but NOT in the standalone recorder!
- SCI-controlled version: jumps to next track immediately when AICTRL0 selects a new track.

Version 1.13, 2007-01-23

- Analog drivers are kept inactive until a MMC/SD card is found.
- SCI-controlled version now skips ID3V2 tags.
- SCI-controlled version increments decode time between repeats when in looped play mode. Thus the decode time increases even for short (less than one second) WAV files.
- VS1033c version added.

Version 1.12, 2006-07-13

- MMC communication changed. Now supports some problem MMC's.
- mkrecord.c changed to handle large WAV's better (unsigned size calculation).
- Sample rate and mic/line selection can now be configured from the boot image for the Standalone Recorder.
- Standalone Recorder now has pause in VS1003B version.
- The maximum playable/recordable filesize increased from around 544 MB to over 1 GB. (Filesize was limited in 1.11 when more fragments were supported.)

Version 1.11, 2006-05-08

- Standalone Recorder auto gain maximum set to 64× (0xffff). It can be changed from the EEPROM image.
- SCI-controlled VS1002D version song change fixed (bug introduced in 1.02).
- SCI_AICTRL1 is zero while the directory is being processed.
- Large FAT32 directories tend to be very fragmented. Now handles maximum of 35 fragments (was 16).
- Pull-down resistors added to GPIO2 and GPIO3.

Version 1.10, 2006-03-10

- Standalone Recorder has less player features to make recording possible.
- Standalone Recorder sample rate set to 8000 Hz.
- VS1011B and VS1023 versions removed
- Added SPI-EEPROM boot images for SCI-controlled version.
- A bug that could trash the directory fixed in Standalone Recorder.

Version 1.02, 2005-11-17

- Now detects the file type from 8.3-character filename suffix. Only matching files are processed.
 - VS1011E and VS1002D: .MP3 .WAV
 - VS1003B: .MP3 .WAV .MID .WMA .WMV .ASF
 - VS1033C: .MP3 .WAV .MID .WMA .WMV .ASF .AAC .MP4 .M4A

If the file name suffix matches but still nothing decodable is found from the first 400 kB, the file is skipped.

Version 1.01, 2005-11-07

- 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.

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.

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/VS1033 versions

Version 0.94, 2005-08-24

- FAT 8.3-character filename readable in SCI-controlled version (see section 4.2)
- 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.

Since the 1.02 version, if the filename suffix does not match any of the valid ones for the specific chip, the file is ignored.

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.

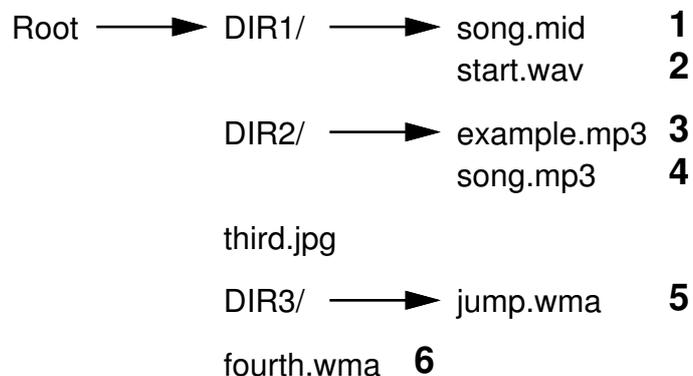


Figure 7: Play Order with subdirectories

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 but ignored because the suffix does not match a supported file type, 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.

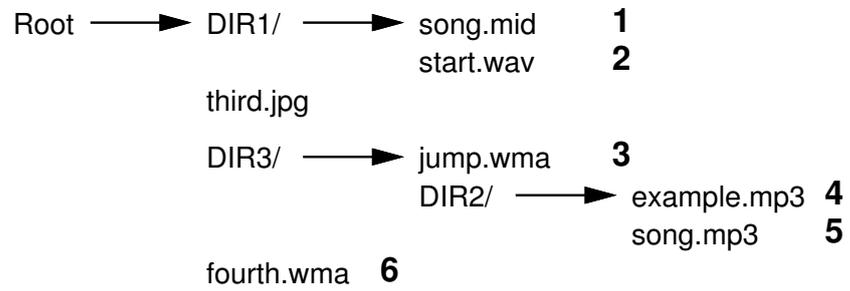


Figure 8: Play Order with nested subdirectories