VS1000B/C Code Examples

VS1000 "VLSI Solution Ogg Vorbis Player"

Project Code:
Project Name: VS1000

Revision History			
Rev.	**Date**	**Author**	**Description**
1.11	2008-09-12	PO	Matrix keyboard and RC5 remote added.
1.10	2007-11-01	PO	mmc.c added.
1.00	2007-10-08	PO	Initial version.

Table of Contents

1 VS1000 Developer Library **3**

2 nandfirmware.c **4**
2.1 MAPPERLESS . . . 5
2.2 SAVE_SETTINGS . . . 7
2.3 AUDIO_TESTS . . . 8
2.4 MAX_VOLUME . . . 8
2.5 DEFAULT_VOLUME . . . 8
2.6 NEW_OFF_LIMIT . . . 8
2.7 LOWER_VOLTAGES . . . 8
2.8 DOUBLE_VOLUME_SPEED . . . 8
2.9 EGG_FILES_KEYS . . . 8
2.10 POWERON_KEYWAIT . . . 9
2.11 POWEROFF_TIMEOUT . . . 9
2.12 LOOP_FILES . . . 9
2.13 POWER_MONITOR . . . 9
2.14 USE_CHAPTERS_CHARACTER . . . 9
2.15 PATCH_LBAB . . . 9

3 mmc.c **10**
3.1 Hardware . . . 10
3.2 Software . . . 11
3.3 USE_HC . . . 11
3.4 PATCH_LBAB . . . 11
3.5 USE_WAV . . . 12
3.6 USE_MATRIX_KEYBOARD . . . 12

4 rc5.c **14**

5 Document Version Changes **16**

1 VS1000 Developer Library

All information in this document is provided as-is without warranty.

To aid in software development, we have started to collect useful routines into a link library that any programmer can use easily. To use these routines in `libdev1000.a`, just `#include <dev1000.h>` in your program and have `-ldev1000` in the linker command line. If you are using vskit version 1.34, this has already been done for you.

Because new examples often use routines that are fresh additions to the developer library, you should have the latest library installed. The library in this package always corresponds to the examples, so if you are having problems with the examples, copy the contents of the `vs1000b-lib/` directory to the `lib/` directory on your vskit development setup.

There is a separate automatically generated document and hyperlinked HTML version about both the common VSDSP link libraries and the VS1000 developer library functions. (See http://www.vlsi.fi/en/support/software/vs1000tools.html .)

2 nandfirmware.c

The C-file nandfirmware.c contains a collection of helpful example codes for VS1000B/C that can be selected with a single preprocessor define. Comment all defines that you do not need and uncomment the features you want. Check out mmc.c too, it has example code that can be used with NAND FLASH also.

- MAPPERLESS - for factory-programmed content that the user can not change. The FLASH is programmed with a FLASH programmer.
- SAVE_SETTINGS - saves play position, volume, and other settings to flash when the unit is turned off.
- SAVE_NO_POSITION - ignores saved play position when SAVE_SETTINGS selected.
- AUDIO_TESTS - miscellaneous audio tests that are activated when KEY_5 is pressed at power-on.
- MAX_VOLUME - limits the maximum volume that can be selected.
- DEFAULT_VOLUME - changes the default volume after power-on (the first power-on with SAVE_SETTINGS).
- NEW_OFF_LIMIT - changes the default 2-second power-off press time.
- LOWER_VOLTAGES - uses different voltages than the defaults.
- DOUBLE_VOLUME_SPEED - doubles the volume adjustment speed.
- EGG_FILES_KEYS - play .egg files after a key combination is pressed.
- POWERON_KEYWAIT - requires a longer press to turn the unit on.
- POWEROFF_TIMEOUT - changes the default 5-minute timeout.
- LOOP_FILES - seamlessly plays a single file in a loop. Song change from the keys is still possible.
- POWER_MONITOR - flashes power LED when low power detected. In practise this only works with direct cell power supply, not with a switching power transformer.
- USE_CHAPTERS_CHARACTER - combines several files into groups.
- PATCH_LBAB - removes 4GB restriction from USB (SCSI).

2.1 MAPPERLESS

When NAND Flash storage is used, logical-to-physical mapping, error correction, bad block detection and management, and block usage balancing are handled by the Mapper module that is in the VS1000 firmware. Both the USB Mass Storage implementation and the player access the Flash disk through the Mapper.

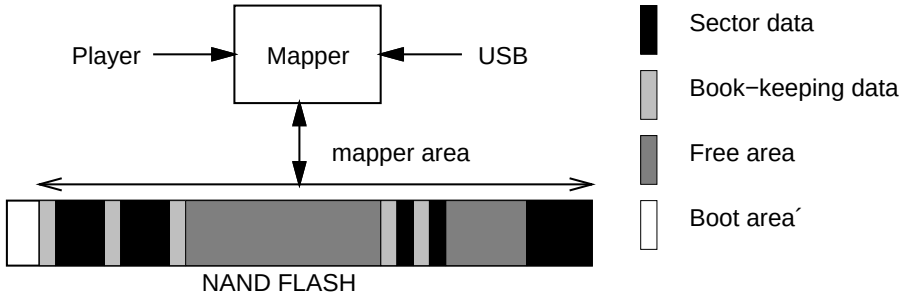

Figure 2.1: FLASH Usage with Mapper

The first erase block, or at least 128 kB is reserved for boot code, the rest of the FLASH is used by the Mapper. When sectors are written through USB to the disk, the Mapper allocates a new version of the block, writes the sector and error correction data, and updates the book-keeping data accordingly. Because of this any logical sector can be almost anywhere on the physical NAND FLASH chip and the Mapper performs the conversion.

In player mode the Mapper is in read-only mode, and performs the logical-to-physical mapping and error correction. Another module named MiniFAT interprets the logical disk contents and provides file services to the player.

The book-keeping data makes it possible to handle error correction, bad block remapping, and also otherwise dynamically recycle the NAND FLASH blocks, but it makes it hard to pre-program the FLASH contents.

Things become simpler if we only want to have content that is programmed during production, and the user is not allowed to alter it, i.e. no USB connection.

With more reliable NOR FLASH chips most of the Mapper's responsibilies can be greatly simplified. First, error correction is no longer needed. Also, the factory FLASH programmers provide a method for skipping bad erase blocks, so the memories can be preprogrammed with ease, when the player implements the same skipping scheme.

Because the contents of the disk are fixed, no book-keeping information is needed on the Flash. The player scans the erase blocks at startup, and remembers the location of the bad erase blocks. The number of erase blocks is fairly small, so this scanning is fast, and the number of bad erase blocks allowed by the manufacturer on the Flash is limited, so a short array is enough to remember them. It is then easy to map the logical sectors into physical sectors.

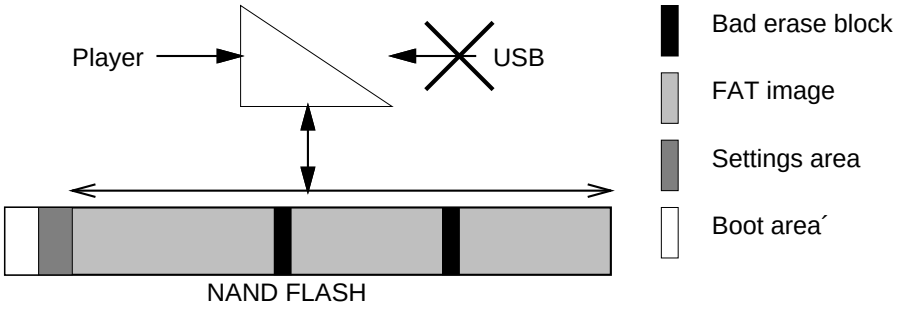

Figure 2.2: FLASH Usage with Mapperless

An image creator software is used to generate the FAT image from a list of files, to combine it with the appropriate boot code, and even select the FLASH parameters (small-page or large-page, erase block size, FLASH size). The resulting FLASH image can then be programmed into the FLASH chips with a FLASH programmer.

What is different in MAPPERLESS.

- FLASH can be factory-programmed / programmed with a Flash programmer.
- Content can not be changed. USB is not used. Connecting USB with MAPPERLESS may trash the FAT image unless USB detection is disabled with sofware.
- No space is lost for book-keeping.
- The player creates a `bads[]` array at startup.
- ReadDiskSector() is replaced with a version that uses the `bads[]` array and bypasses the Mapper.
- The player does not create Tiny Mapper (read-only mode of the Mapper).
- The Flash image is generated with image creator.

2.2 SAVE_SETTINGS

The SAVE_SETTINGS feature saves the current song and position, volume setting, and other settings to FLASH when the unit is turned off, and will restore the settings at power-on. Each save uses 16 bytes. One erase block is used for the settings. When the settings fill the whole erase block, it is erased and saving starts again from the start of the erase block. If unused space is not found after erase, as is the case with one-time-programmable (OTP) memory, the saving is skipped.

With MAPPERLESS the image creator leaves two erase blocks free. The first one is used for boot code, and the second is used for settings.

Without MAPPERLESS there are two possible locations for settings. If the erase block size is 128 kB or larger, we use the last erase block of the FLASH for settings and reduce the FLASH size that is reported to the Mapper. Because the Mapper only uses the size information when it low-level formats the FLASH, you should not have previous content on the FLASH when erase block size is 128 kB or larger when a firmware with this option set is first run. Otherwise the system seems to work, but at some point either settings are lost when new files are written, or the settings overwrite some file data. So, it is best to program it to a fresh board, or make certain the VS1000_B.RUN erases the filesystem while programming the firmware.

If the erase block size is smaller than 128 kB, the Mapper leaves more than one erase block unused for boot area. In this case we use the first erase block for the boot code, and the second erase block for the settings. In this case the previous contents of the FLASH does not matter, because the Mapper area stays the same size as before.

If SAVE_NO_POSITION is also set, all settings are saved, but the play position is ignored when settings are read.

2.3 AUDIO_TESTS

This includes miscellaneous audio tests that are activated when KEY_5 (feature key) is being pressed at power-on. The power key jumps from one test to another. The tests are: frequency sweep, amplitude sweep, crosstalk test, and intermodulation test.

2.4 MAX_VOLUME

MAX_VOLUME limits the maximum volume that can be selected.

2.5 DEFAULT_VOLUME

DEFAULT_VOLUME sets the default volume that is used after power-on. This also sets the initial volume at the first power-on with SAVE_SETTINGS.

2.6 NEW_OFF_LIMIT

Normally you have to press the power button for 2 seconds to turn the unit off. NEW_OFF_LIMIT can be used to reduce the time.

2.7 LOWER_VOLTAGES

This example shows how to change the default voltages. Do not drop the core voltage too low or you will activate the voltage monitor.

2.8 DOUBLE_VOLUME_SPEED

The default volume step size is 0.5 dB. This doubles the volume adjustment speed.

2.9 EGG_FILES_KEYS

EGG_FILES_KEYS shows how to dynamically change the supported suffixes. The example plays Ogg Vorbis files with the suffix .egg after a key combination is pressed. The desired key combination is set in the definition of EGG_FILES_KEYS.

2.10 POWERON_KEYWAIT

By default the unit turns on when the power button is pressed and starts to play the first song. POWERON_KEYWAIT requires a longer press to turn the unit on. If the press is not long enough, the unit turns itself off automatically.

2.11 POWEROFF_TIMEOUT

The low-power pause mode has a 5-minute timeout. After this time the unit turns itself off. The length of the timeout can be changed with POWEROFF_TIMEOUT.

2.12 LOOP_FILES

LOOP_FILES allows files to be looped seamlessly. Song change and other key controls are still possible.

2.13 POWER_MONITOR

POWER_MONITOR quickly flashes the power LED when low power situation is detected. In practise this only works with direct power cell supply. This does not work with a switching power transformer, because a voltage drop is only seen at VHIGH after the power cell is already virtually empty.

2.14 USE_CHAPTERS_CHARACTER

This option combines files into groups by checking the first character of file names of consequtive directory entries. Fast forward and rewind (long presses of next and previous) are replaced by chapter jump functions. For example long press of next will skip files forward until a file name starting with a different character is found.

Because the order of files in the directory is important, this option is mainly intended for pre-recorded content. A more advanced version would use subdirectories to group files.

2.15 PATCH_LBAB

This removes 4 GB restriction from the SCSI protocol implementation of VS1000B/C. The patch also calls ScsiTestUnitRead() function whenever SCSI receives the SCSI TEST UNIT READY command.

Currently nandfirmware.c provides an empty `ScsiTestUnitReady()` function (SCSI_OK always returned).

3 mmc.c

This example shows an Ogg Vorbis player that uses MMC/SD for storage. The player code is loaded from an SPI EEPROM.

3.1 Hardware

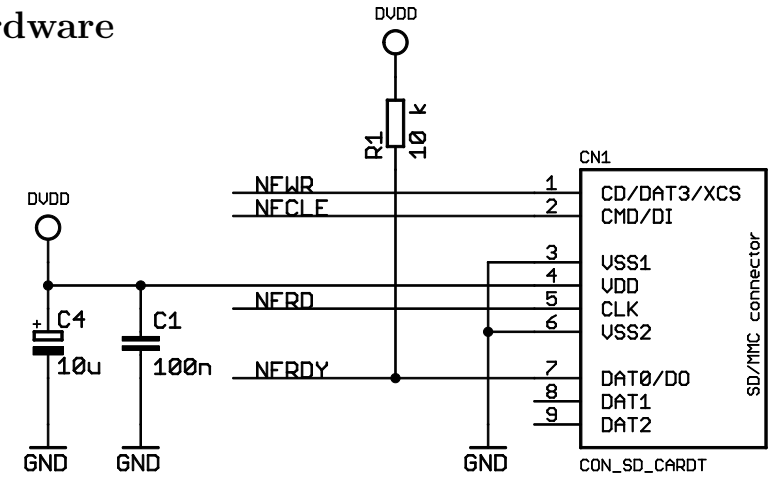

Figure 3.1: MMC/SD Connection

The MMC/SD pins are connected in the following way:

MMC Pin	VS1000 Pin	Other
SO	NFRDY (GPIO0_8)	10 kΩ pull-up resistor (shared with FLASH)
SI	NFCLE (GPIO0_12)	
SCLK	NFRD (GPIO0_9)	
XCS	NFWR (GPIO0_11)	

The MMC/SD is accessed using the SPI protocol, but GPIO pins are used instead of the SPI hardware. This leaves the VS1000 SPI hardware for SPI EEPROM and you can also easily connect a display. Also, you do not need to remove the NAND Flash from the developer board to be able to develop code for the MMC/SD as long as you keep NFCE (GPIO0_10) high.

Note that the 10 kΩ pull-up resistor already exists in the VS1000 developer and demo player boards, because NAND FLASH requires it on NFRDY.

These SD/MMC Add-On cards for the VS1000 Developer Board will be available from VLSI Solution.

3.2 Software

The mmc.c source implements the default player user interface, handles MMC/SD hot insertion and removal, and also allows the USB Mass Storage function to be used, so the MMC/SD can be accessed with PC. High-Capacity SD cards are also supported.

To make the MMC/SD communication faster, and to patch some USB (SCSI) functions with minimal amount of code, mmc.c uses some assembler functions. These functions are collected into a link library called `libdev1000.a` and C definitions and prototypes can be found from `dev1000.h`. Vskit134 uses this developer library by default.

To use these with vskit133, copy `dev1000.h` to the `include/` directory and `libdev1000.a` to the `lib/` directory and add `-ldev1000` to the linker options in `BUILD.BAT` (`vslink -k -m mem_user -L lib -lc -ldev1000 ..`).

Most of the features in nandfirmare.c can be directly copied to mmc.c. The exceptions to this are MAPPERLESS, which is not applicable with MMC/SD, because this code already replaces the FLASH mapper, and SAVE_SETTINGS, which needs to be adapted to save the settings to SPI EEPROM.

3.3 USE_HC

With this define SDHC (high-capacity) cards are supported. This is the default, but you can disable the feature to make some more code space available for your own additions.

The code also raises the AVDD voltage in player mode to 3.6 V. This increases the IOVDD drive capability slightly and allows some high-speed (i.e. high-current) cards to work. Otherwise IOVDD drops too much during card initialization, which triggers brownout detection in the card, thus disabling it.

```
voltages[voltAnaPlayer] = 30; /*3.6V for high-current MMC/SD!*/
PowerSetVoltages(&voltages[voltCorePlayer]);
```

3.4 PATCH_LBAB

This removes 4 GB restriction from the SCSI protocol implementation of VS1000B/C. The patch also uses the SCSI TEST UNIT READY command to check the presence of MMC/SD each time the command is received. This allows automatic removal and re-attachment of the mass storage device when MMC/SD card is changed.

If you disable the patch you should make it physically impossible to change the MMC/SD card while the unit is attached to USB. Also, without the patch the maximum size of the MMC/SD that is accessible through USB is 8GB.

This patch can also be ported to nandfirmware.c. You just need to make an empty `ScsiTestUnitReady()` function.

3.5 USE_WAV

When USE_WAV is defined, mmc.c uses the CodMiniWav codec from the developer library to decode linear wav files. The supported file suffixes are set to include both .WAV and .OGG, and the PlayCurrentFile hook is updated to check for a wav file before trying to decode a file as an Ogg Vorbis file.

You can use the wav support code in nandfirmware.c and in your own code by copying the relevant parts.

3.6 USE_MATRIX_KEYBOARD

Some applications require more keys than the five keys and the power button that are available in the default firmware. The VS1000 developer library provides routines (KeyScan7() and Suspend7()) to read an extra key connected to GPIO0_5. But sometimes even this is not enough. Upto 16 keys can be added with little effort using a matrix keyboard.

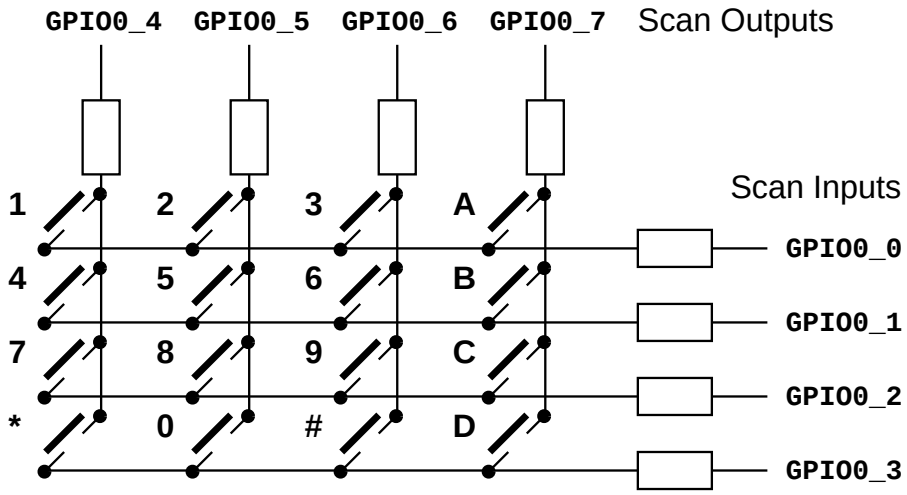

Figure 3.2: Key Matrix Connection

GPIO0[4:7] are used as scan outputs (columns), GPIO0[0:3] are scan inputs (rows). You can also use fewer than four scan outputs. When scanning keys, only one of the scan outputs will be high at a time, the others are kept low. The results are read from scan inputs. Series resistors of 47 kΩ prevent pressed keys from interfering with NAND FLASH accesses and causing short-circuits between scan outputs.

The current code only generates one keycode at a time. If more than one key is pressed at a time, a lower column and lower row gets priority. If three or more keys on one row are pressed simultaneously, keys are not detected (can be fixed by using diodes instead of resistors on the scan outputs).

To call the matrix scanning routine instead of the normal key scan, the idle hook needs to

be replaced. KeyScanMatrix() parameter defines what keycodes to generate. KeyScanMatrix() calls MatrixScan() function to scan the keys, performs short/long press logic, and calls the key event handler according to the current active keymap.

If you want to have new key events, the KeyEventHandler() hook needs to be replaced to implement them. The example code implements direct selection for songs 1-9.

The USBSuspend() hook must also be replaced so that pressing the keys of the keyboard can wake up the unit from low-power pause mode.

See mmc.c for code examples.

You can use the wav support code in nandfirmware.c and in your own code by copying the relevant parts.

4 rc5.c

The C file rc5.c shows how to use an infra-red RC5 remote to control the player. The required hardware is just a RC5 receiver chip connected to GND, IOVDD and GPIO0[14]. The polarity of the receiver does not matter.

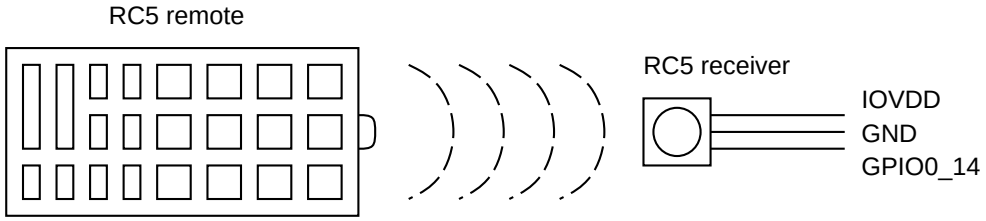

Figure 4.1: RC5 Remote

The RC5 code consist of an interrupt handler that is installed with Rc5Init(), a short FIFO buffer to hold the received RC5 codes, and Rc5GetFIFO() which returns received RC5 codes in the receiving order, or 0 if there are no codes waiting.

In addition to calling Rc5Init() the user has to enable both rising and falling edge interrupts and the relevant GPIO interrupt. In the example code GPIO0[14] is used. Any other free interruptable GPIO pin can also be used by changing the initialization routine. GPIO1 pins can also be used.

```
/* Initialize RC5 reception */
Rc5Init(INTV_GPIO0);
PERIP(GPIO0_INT_FALL) |= (1<<14); /*DISPLAY_XCS*/
PERIP(GPIO0_INT_RISE) |= (1<<14);
PERIP(INT_ENABLEL) |= INTF_GPIO0;
```

The example code replaces the idle hook to process the received RC5 codes. The new code first calls Rc5GetFIFO(). If there is a RC5 code available, a key event is generated according to a rc5 key mapping array. The default idle hook function UserInterfaceIdleHook() is then called to read the keys on the unit and to update the LEDs.

Check out rc5.c to see how to implement direct access of songs from 0 to 9.

To allow RC5 codes to be received during the pause mode, low-power pause mode can not be entered. This can be prevented by replacing the USBSuspend hook.

```
void MyUSBSuspend(u_int16 timeOut) {
    if (timeOut) {
        /* Do not enter low-power pause mode,
           stay in 1.0x in to receive the RC5 codes in pause mode. */
    } else {
        RealUSBSuspend(0); /* During USB suspend RC5 does not work. */
    }
}
...
    SetHookFunction((u_int16)USBSuspend, MyUSBSuspend);
```

Or if you do not care about USB Suspend, simply
`SetHookFunction((u_int16)USBSuspend, NullHook);`.

5 Document Version Changes

This chapter describes the most important changes to this document.

Version 1.11, 2008-09-12

- LOOP_FILES improved.
- USE_WAV and USE_MATRIX_KEYBOARD added.
- rc5.c added
- VS1000 developer library mentioned.