MLC Player 8k

Audio Player for 8kB-page MLC

Description

The VS1000 ROM firmware natively supports 512-byte page and 2048-byte page single-level cell (SLC) NAND FLASH memories. The newest NAND FLASH memories use 8192-byte pages with multi-level cell (MLC) technology, and these memories can not be directly used with VS1000.

The MLC Player 8k example shows how these memories can be supported with VS1000 Developer Board using custom user code. Both low-level page read / page write routines and higher-level logical-to-physical-sector mapper routines must be replaced.

A small SPI EEPROM is required if the NAND FLASH requires RESET command to be sent after power-on (see Data Protection and Power-Up Sequence chapter of the FLASH datasheet for details).

MLCPlayer has been tested with these NAND FLASHes with 8 kB page size:

- Samsung **K9GAG08U0E**: 2 GB total size, 1 MB erase block (make program2g)
- Hynix **HY27UAG8T2BTR**: 2 GB total size, 2 MB erase block (make program2g2)

Both of these require the SPI EEPROM to give the NAND-FLASH RESET (0xff) command after power-on.

- Toshiba **TC58NVG4D2FTA00**: 1 GB total size, 1 MB erase block (make program1g)

This memory does not require RESET after power-on, if pin 39 (PSL) is grounded or NC.

You are welcome to send samples of other MLC FLASH chips, so we can test them in the application.

Revision History

Rev	Date	Author	Description
1.02	2011-05-24	POj	Listed some FLASHes the player has been tested with.
1.01	2011-05-18	POj	Picture added.
1.00	2011-05-09	POj	Initial revision.

Contents

1 General

Multi-level-cell (MLC) technology restricts how page programming can be performed on a NAND FLASH chip. Each page can only be programmed once between erase operations, including the spare area. Also, pages in each erase block must be programmed in sequential order, otherwise the data will be corrupted.

Because data can only be written 8 kB at a time, and sequentially into each erase block, the default ROM mapper can not be used. The new MLC mapper uses different data structures to map out bad erase blocks, so it is easy and fast for the logical read functions to locate data blocks. The player needs to keep as little information in memory as possible.

Because VS1000 instruction memory size is limited and we need to replace a lot of code, the player is divided into several parts that are loaded separately.

- SPI Boot – loaded from SPI EEPROM
- MLC Boot – loaded from NAND FLASH using the ROM functions
- MLC Init – loaded from NAND FLASH using replaced functions
- MLC 8k – loaded from NAND FLASH using replaced functions.
- MLC Player – loaded from NAND FLASH using replaced functions.

1.1 SPI Boot

As a data integrity measure the new MLC FLASHes with 8 kB page size start in power-down state after power-on. They require that a RESET command (0xff) is sent after power-on to bring the memory into operation mode. VS1000D does not send this RESET command after reset, so boot from NAND FLASH is not possible, if the chip requires RESET to be sent.

A small program in SPI EEPROM sends the RESET command to NAND FLASH, then proceeds with normal boot code.

Some NAND FLASHes allow the behavior to be configured using an I/O pin. With these memories the SPI EEPROM is not required.

1.2 MLC Boot

Because NAND FLASH boot is performed using the VS1000 ROM firmware, which does not support the 8 kB page size, the size of the boot program is limited to 2048 bytes. For faster boot (and lack of use of ECC in the ROM boot routine) the size of the MLC Boot program is kept as small as possible, currenly below 1024 bytes.

MLC Boot program loads the next program (MLC Init) with the updated low-level routines with error correction code (ECC).

1.3 MLC Init

Because the USB Mass Storage driver needs to both read and write sectors, it is the largest program. Initialization that can be performed beforehand, have have been moved into this program.

MLC Init scans erase blocks to locate the first data block, bad blocks, and possible replaced blocks. It also creates a mapping table which the MLC mapper read function uses in MLC Player.

After initialization has been performed, MLC Init loads either the USB Mass Storage program or MLC Player depending on whether USB is connected or not.

1.4 MLC 8k – Mass Storage

The MLC 8k program performs page and erase block updates when new sectors are written through USB Mass Storage. The use of multi-level-cell technology causes two major restrictions. 1) Each page can only be written once, thus 8 kB of data and the corresponding meta information (into the spare area) must be written in one go. 2) Each page in an erase block must be written in sequence.

These restrictions mean that the original logical-to-physical mapper that is in the ROM can not be used, a new storage structure is required. The MLC Mapper uses a structure very much like the “mapperless” system described in the nandfirmware package. A very simple mapping from logical to physical sector is performed. The lowest bits in each logical sector number directly determines in which page the sector resides inside an erase block. The correct erase block is determined by the higher bits of the logical sector number, by the starting block of the data area (one erase block is reserved for boot, another for settings, and a third one for player-mode logical-to-physical mapping table), and by the bad block information (bad erase blocks are skipped).

When writing data, first the 512-byte sectors that the MLC mapper receives through USB are cached into data RAM to compose data into 8 kB pages. These are written to a journaling erase block in the FLASH. When the journal becomes full or the last page in an erase block is written, the current erase block is updated by copying unchanged data to temporary storage, erasing the data block, then copying all valid data back to the block from journal and temporary storage.

When done (USB detached), MLC 8k reloads MLC Boot (with ECC).

1.5 MLC Player

MLC Player is the actual audio player program. The player provides normal play controls: volume and prev / next, EarSpeaker spatial processing control from KEY5 (Feature key, GPIO0_4), and shuffle function toggle with a long press of the same key.

The player uses OLED display for status information. The OLED backlight is dimmed after 10 seconds of inactivity (no key presses). The player is automatically turned off after 5 minutes of pause mode.

If USB is attached, MLC Player reloads MLC Boot (with ECC).

Note: some of the features can be enabled/disabled and configured using pre-processor defines in `player.c` .

2 Player User Interface

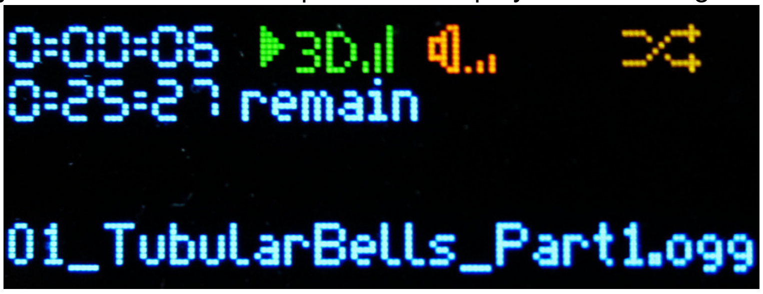
The MLC Player uses the default key configuration.

Key Mapping

Key	**Short Press**	**Long Press**
PWRBTN	pause / play	power off
KEY 1	volume down	volume down
KEY 2	volume up	volume up
KEY 3	previous	rewind
KEY 4	next	fast forward
KEY 5	EarSpeaker select	shuffle mode toggle

EarSpeaker select rotates between four settings: off, low, mid, and high. When active, the selected setting is displayed on the OLED with a "3D" symbol and one, two, or three bars.

The OLED display of the VS1000 Developer Board displays the following information:



- Top line
  - Current playing time in HH:MM:SS format
  - Play or pause symbol
  - EarSpeaker level, if EarSpeaker is active
  - Volume setting
  - Shuffle symbol, if shuffle mode is selected
- Second line: remaining playing time (configurable)
- Fourth/fifth line: file name

3 Unit Programming

The MLC Player 8k package source code contains a Makefile for automatic building of image files.

If you work with Linux, first compile the image generator tools by:

```
> make tools
```

To program the NAND FLASH reset code into SPI EEPROM, say:

```
> make resetfromspi.eep
```

This will create `resetfromspi.bin` executable, `resetfromspi.spi` boot image and program it into SPI EEPROM (16-bit address) with `spiprom.bin` . This is not needed if the MLC NAND FLASH does not require reset after power-on.

To create and program the combined MLC Player 8k boot image into NAND FLASH, you need to check the total size and the size of the erase block of the MLC NAND FLASH you are using.

For 1 GB MLC FLASH with 8 kB page and 1 MB erase block size use:

```
> make program1g
```

For 2 GB MLC FLASH with 8 kB page and 1 MB erase block size use:

```
> make program2g
```

For 2 GB MLC FLASH with 8 kB page and 2 MB erase block size use:

```
> make program2g2
```

The image will be programmed with ECC (one bit corrected for each 512 bytes), so it can be later used when loading programs. The programmed image is started automatically after a successful verify.

For other size combinations you can modify `Makefile`.

4 Quick Guide to Source Code

Makefile handles everything. It compiles and links the source code of each separate program, generates NAND FLASH boot images from each program, then combines them into a single image. Makefile also contains targets to program images into SPI EEPROM and NAND FLASH.

You should not need to change any of the other programs than `player.c`. The following defines are present in the file to easily enable/disable or otherwise customize the main features.

USE_DISPLAY

- When defined use the default OLED display of the VS1000 developer board. The LCD routines from vs1000 developer libarary are used (dev1000.h /libdev1000.a). If you use another SPI-connected display, you need to build your own initialization string.
- During playback the display is updated with various information:
  - play position
  - play/pause icon
  - EarSpeaker level, if enabled
  - Volume level
  - shuffle icon, if enabled
  - remaining playtime (if enabled with GET_PLAYTIME)
  - file name

BACKLIGHT_DIM

- The display is dimmed after BACKLIGHT_DIM ticks have passed from the last key press. If BACKLIGHT_DIM is not defined, this feature is removed.

QUIET_CANCEL

- Pause mode can be enabled at any time, and the decoder can have samples waiting to be played. If you change songs directly from pause mode, these waiting samples will be played before the decoding of the current file is cancelled. This may be distracting. With this define the output volume is lowered while the remaining samples are playing and volume is restored for the next song.

GET_PLAYTIME

- Enables and uses a routine to calculate the remaining play time of an Ogg Vorbis file. Or actually the number of samples in the Ogg Vorbis file. The number of samples is converted to seconds by dividing it by `cs.sampleRate`, which is valid when the file is being played. (You can also read the samplerate from the right offset at the beginning of the Ogg Vorbis file.)

5 Latest Document Version Changes

This chapter describes the most important changes to this document.

Version 1.02, 2011-05-24

- Listed some FLASHes the player has been tested with.
- Some source code information.

Version 1.01, 2011-05-18

- Added a picture of the user interface.

Version 1.00, 2011-05-09

- Initial revision.

6 Contact Information

VLSI Solution Oy
Entrance G, 2nd floor
Hermiankatu 8
FI-33720 Tampere
FINLAND

Fax: +358-3-3140-8288
Phone: +358-3-3140-8200
Email: sales@vlsi.fi
URL: http://www.vlsi.fi/