FTL NAND Media Driver for Spansion S34MLxxG1 User Guide

Version 1.30

For use with FTL NAND Media Driver for Spansion® S34MLxxG1 versions 1.03 and above

Date: 11-Aug-2017 16:19

Table of Contents

1 System Overview

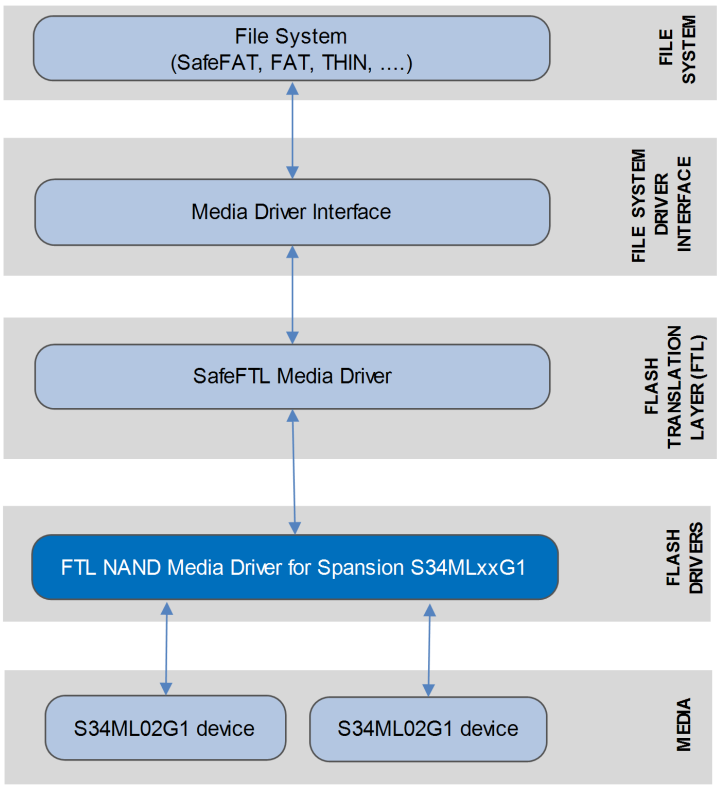
1.1 Introduction

This guide is for those who want to use HCC's FTL NAND Media Driver for Spansion® S34MLxxG1 flash devices in their system. The xx represents the size in Gbits and can be 01, 02, or 04. The driver supports devices with bus widths of 8 and 16 bits.

This guide covers all aspects of configuration and use. This media driver conforms to the *HCC Media Driver Interface Specification*.

The media driver provides an interface for a file system to read from and write to NAND Flash storage devices. A single media driver can support one or more physical media, each of these being represented as a different drive at the media driver interface. The file system handles all drives identically, regardless of their internal design features.

The diagram below shows a typical system architecture including a file system, media driver, and media. As an example, this shows two S34ML02G1 MCUs.



Note the following:

- The file system can be any HCC file system that addresses logical sector arrays (including SafeFAT, FAT, and THIN).
- The Flash Translation Layer (FTL) is the SafeFTL media driver. This has its own manual.

- The NAND flash driver is written specifically for the NAND flash controller (integrated with the S34MLxxG1 microcontroller) and the specific NAND flash array used.

1.2 About S34MLxxG1 NAND Flash

This table summarizes the properties of the various device types:

	S34ML01G1	S34ML02G1	S34ML04G1
Size (Gb)	1	2	4
Bytes per page	2112	2112	2112
Pages per block	64	64	64
Blocks	1k	2k	4k
Planes	1	2	2

Error-Correcting Code (ECC) Requirement

1 bit per 528 bytes. The sample driver includes a software ECC algorithm. This can be modified to use hardware ECC if this is provided by the host microcontroller.

1.3 Feature Check

The main features of the media driver are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Conforms to the low level NAND flash interface specification defined by HCC's SafeFTL.
- Supports multiple S34ML01G1, S34ML02G1, or S34ML04G1 flash drives.

1.4 Packages and Documents

Packages

The table below lists the packages that you need in order to use this module:

Package	Description
hcc_base_doc	This contains the two guides that will help you get started.
media_drv_ftl_base	The base SafeFTL package.
media_drv_ftl_nand_s34mlxg1_8b_ecc	The media driver package described in this document.

Documents

For an overview of HCC file systems and flash management technologies, see Product Information on the main HCC website.

Readers should note the points in the HCC Documentation Guidelines on the HCC documentation website.

HCC Firmware Quick Start Guide

This document describes how to install packages provided by HCC in the target development environment. Also follow the *Quick Start Guide* when HCC provides package updates.

HCC Source Tree Guide

This document describes the HCC source tree. It gives an overview of the system to make clear the logic behind its organization.

HCC Media Driver Interface Guide

This document describes the HCC Media Driver Interface Specification.

HCC SafeFTL User Guide

The user guide for SafeFTL.

HCC FTL NAND Media Driver for Spansion® S34MLxxG1 User Guide

This is this document.

1.5 Change history

This section describes past changes to this manual.

- To download earlier manuals, see Archive: FTL NAND Media Driver for Spansion S34MLxxG1 User Guide.
- For the history of changes made to the package code itself, see History: media_drv_ftl_nand_s34mlxg1_8b_ecc.

The current version of this manual is 1.30. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.30	2017-08-11	1.03	Corrected *Packages* list.
1.20	2017-06-21	1.03	New *Change History* format.
1.10	2016-02-11	1.03	Added **psp_nand_s34mlxg1_ecc_init()** function.
1.00	2015-11-23	1.01	First online version.

2 Source File List

This section describes all the source code files included in the system. These files follow the HCC Embedded standard source tree system, described in the *HCC Source Tree Guide*. All references to file pathnames refer to locations within this standard source tree, not within the package you initially receive.

> **Note:** Do not modify any files except the configuration files and PSP files.

2.1 API Header File

The file **src/api/api_ftl_nand_s34mlxg1_8b_ecc.h** is the only file that should be included by an application using this module. For details of the single API function, see Application Programming Interface.

2.2 Configuration File

The file **src/config/config_ftl_nand_s34mlxg1_8b_ecc.h** contains all the configurable parameters of the system. Configure these as required.

2.3 Source Code File

The file **src/media-drv/ftl/drivers/nand/spansion/nand_s34mlxg1_8b_ecc.c** holds the source code for the media driver. **This file should only be modified by HCC**.

2.4 Platform Support Package (PSP) Files

These files provide functions the core code needs to call, depending on the hardware. They are in the directory **src/psp/target/nand_s34mlxg1_8b_ecc**.

> **Note:** You must modify these PSP implementations for your specific microcontroller and development board; see PSP Porting for details.

File	Description
psp_nand_s34mlxg1_8b_ecc.c	Source code.
psp_nand_s34mlxg1_8b_ecc.h	Header file.

2.5 Version File

The file **src/version/ver_ftl_nand_s34mlxg1_8b_ecc.h** contains the version number of this module. This version number is checked by all modules that use a module to ensure system consistency over upgrades.

3 Configuration Options

Set the system configuration options in the file **src/config/config_ftl_nand_s34mlxg1_8b_ecc.h**. This section lists the available configuration options and their default values.

S34MLXG1_ECC_ID

The NAND ID. The default is ((0x01 << 0) | (0xD3 << 8) | (0xD1 << 16) | (0x95 << 24)).

S34MLXG1_ECC_BLOCK_NUM

The number of erasable blocks in the target flash array. The default is 8192.

S34MLXG1_ECC_PAGE_DATA_SIZE

The data area in bytes available on one page. The default is 2048.

S34MLXG1_ECC_PAGE_TOTAL_SIZE

The total size of the page, including the data and spare areas. The default is 2112.

S34MLXG1_ECC_PAGE_PER_BLOCK

The number of pages per erasable block. The default is 64. This must not be greater than MDRIVER_FTL_MAX_PAGE_PER_BLOCK.

S34MLXG1_ECC_FREE_BLOCK_AVAILABLE

The number of free blocks. The default is 99. This must not be greater than MDRIVER_FTL_MAX_FREE_BLOCKS.

S34MLXG1_ECC_LOG_BLOCK_AVAILABLE

The number of log blocks. The default is 6; do not set it below this. This must not be greater than MDRIVER_FTL_MAX_LOG_BLOCK_AVAIL.

S34MLXG1_ECC_NUM_OF_DIF_MAPBLOCK

The number of blocks used for mapping in the system. The default is 2. The valid range is 1 to 16.

S34MLXG1_ECC_MAPBLOCK_SHADOW

The number of map shadow blocks. This cannot be 0. The default is 2.

The system may be more efficient if more map shadow blocks are used, but each additional block reduces the number of free blocks in the system.

S34MLXG1_ECC_RESERVED_BLOCKS

The number of reserved blocks, the blocks at the start of the flash area that driver should not use. The default is 0.

S34MLXG1_ECC_WEAR_STATIC_LIMIT

The maximum value that the difference between the maximum and minimum wear count can be. The default is 1024.

S34MLXG1_ECC_WEAR_STATIC_COUNT

The number of merge operations after which static wear checking must be run. The default is 128.

3.1 Restrictions

The following value must be 8 or greater:

```
S34MLXG1_ECC_FREE_BLOCK_AVAILABLE - ( S34MLXG1_ECC_NUM_OF_DIF_MAPBLOCK *

S34MLXG1_ECC_MAPBLOCK_SHADOW + 1 ) - S34MLXG1_ECC_LOG_BLOCK_AVAILABLE
```

`S34MLXG1_ECC_FREE_BLOCK_AVAILABLE * 2` must not be greater than `S34MLXG1_ECC_PAGE_DATA_SIZE / 2`

4 Application Programming Interface

This section describes the single function and the structure it uses.

When the media driver is used:

1. The file system calls the media driver's **nand_s34mlxg1_ecc_init()** function.
2. **nand_s34mlxg1_ecc_init()** returns a pointer to a *t_ftl_driver* structure containing a set of functions for accessing the media driver.

4.1 nand_s34mlxg1_ecc_init

This function initializes the driver for this device.

This is called automatically from SafeFTL, using its table of flash drives. Refer to the *HCC SafeFTL User Guide* for details.

Format

```
t_ftl_ret nand_s34mlxg1_ecc_init (
    uint32_t           drvnum,
    t_ftl_driver * *   pps_ftl_driver )
```

Arguments

Argument	Description	Type
drv_num	The number of the drive to initialize. The first drive is 0.	uint32_t
pps_ftl_driver	On return, a pointer to a *t_ftl_driver* structure defining the interface to that driver.	t_ftl_driver * *

Return values

Return value	Description
0	Successful execution.
1	Operation failed.

4.2 SafeFTL Flash Drive Structure Example

SafeFTL uses a flash drive structure containing all the available flash drives. Each available flash driver must have an entry in this table, specifying its initialization function and the parameter to be passed to it in that function. The flash drives are numbered from 0 to (MDRIVER_FTL_MAX_DRIVE-1). The index to this table is used to reference the flash drive.

This structure is held in the main SafeFTL package's **src/config/config_mdriver_ftl.c** file.

The following example shows how a Spansion® drive would appear in this structure. In this case it is the only NAND drive, followed by two NOR drives.

```
t_ftldrive_init  as_ftldrive_init[MDRIVER_FTL_MAX_DRIVE] =
{
  { nand_s34mlxg1_ecc_init, 0U }
  , { ftl_nor_init, 0U }
  , { ftl_nor_init, 1U }
};
```

4.3 Error Codes

The possible return codes are shown in the table below:

Code	Value	Description
LL_OK	0	Successful execution.
LL_ERASED	1	Page is empty.
LL_ERROR	2	Other error.

4.4 t_ftl_driver

The **nand_s34mlxg1_ecc_init()** function returns a pointer to this *t_ftl_driver* structure.

Element	Type	Description
user_data	uint32_t	User-defined data.
pf_getphy	(* pf_getphy)	Pointer to **getphy()** function.
pf_read	(* pf_read)	Pointer to **read()** function.
pf_readpart	(* pf_readpart)	Pointer to **readpart()** function.
pf_write	(* pf_write)	Pointer to **write()** function.
pf_writedouble	(* pf_writedouble)	Pointer to **writedouble()** function.
pf_erase	(* pf_erase)	Pointer to **erase()** function.
pf_isbadblock	(* pf_isbadblock)	Pointer to **isbadblock()** function.
pf_readonebyte	(* pf_readonebyte)	Pointer to **readonebyte()** function.

5 Integration

This section describes all aspects of the module that require integration with your target project. This includes porting and configuration of external resources.

5.1 PSP Porting

The Platform Support Package (PSP) is designed to hold all platform-specific functionality, either because it relies on specific features of a target system, or because this provides the most efficient or flexible solution for the developer.

The module makes use of the following standard PSP function:

Function	Package	Element	Description
psp_memset()	psp_base	psp_string	Sets the specified area of memory to the defined value.

The module makes use of the following PSP function from the file **src/psp/target/nand_s34mlxg1_8b_ecc/psp_nand_s34mlxg1_8b_ecc.h**:

Function	Description
psp_nand_s34mlxg1_ecc_init()	Enables peripheral clocks and initializes the flash pins for NAND access.

This function is described in the following section.

psp_nand_s34mlxg1_ecc_init

This function is provided by the PSP to initialize the device.

This initializes the clocks and flash pins for NAND access.

Format

```
uint32_t psp_nand_s34mlxg1_ecc_init ( void )
```

Arguments

None.

Return Values

Return value	Description
0	Successful execution.
Else	Operation failed.