FTL NAND Media Driver for Hynix HY27UF08 User Guide

Version 1.20

For use with FTL NAND Media Driver for Hynix HY27UF08 versions 1.02 and above

Date: 11-Aug-2017 15:50

Table of Contents

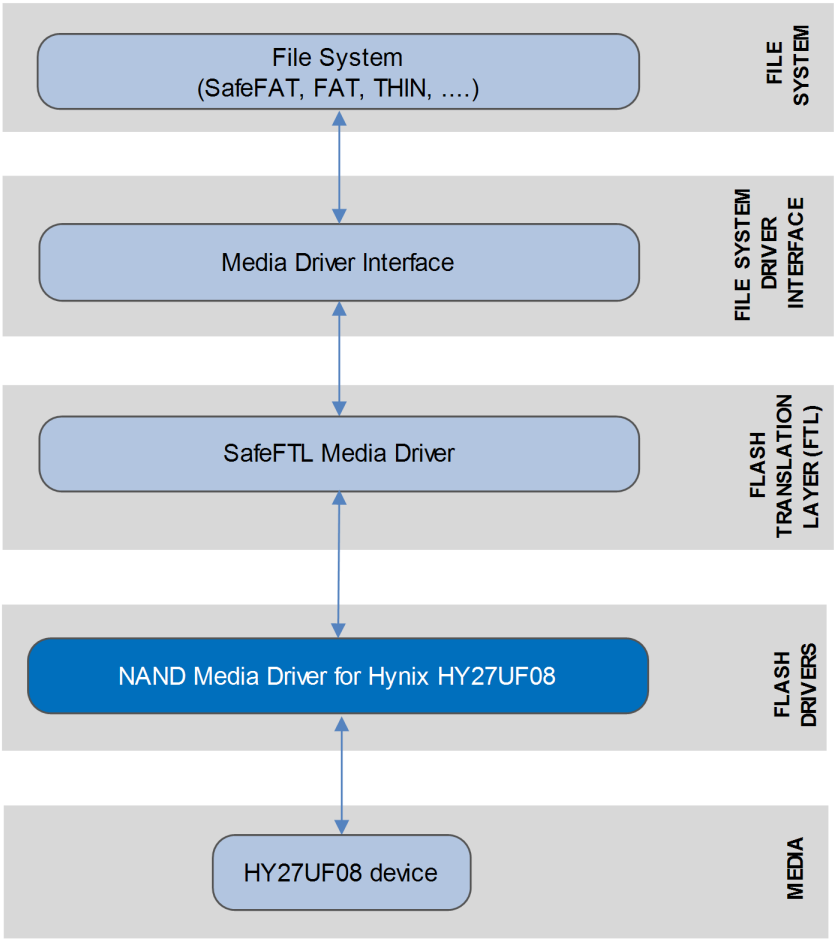
1 System Overview

1.1 Introduction

This guide is for those who want to use HCC's FTL NAND Media Driver for Hynix HY27UF08 flash devices in their system. Adding support for the HY27UF16 is a simple task.

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 a storage device. The diagram below shows a typical system architecture including a file system, media driver, and flash device.



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 HY27UF08 microcontroller) and the specific NAND flash array used.

1.2 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 defined by HCC's SafeFTL.
- With minor modifications, it can operate with other Hynix NAND flash devices. Adding support for the HY27UF16 is simple.

1.3 Flash Page Layout

To support this system, to allow efficient page access, and to preserve manufacturer bad block information, HCC uses each page of flash as shown in the table below.

Error Correction Code (ECC) Implementation

For this device each 512 bytes of data must be protected by an ECC capable of 1 bit correction and 2 bit detection. The correction is implemented in software in the driver provided but may be modified to use hardware if your microcontroller has hardware support for it.

Bytes	Content
0 - 511	Data block 0 [0 - 511].
512 - 515	ECC for data block 0.
516 - 1027	Data block 1 [0 - 511].
1028 - 1031	ECC for data block 1.
1032 - 1543	Data block 2 [0 - 511].
1544 - 1547	ECC for data block 2
1548 - 2047	Data block 3 [0 - 499].
2048 - 2051	Bad block information
2052 - 2063	Data block 3 [500 - 511].
2064 - 2067	ECC for data block 3.
2068 - 2083	HCC spare data area.
2084 - 2087	ECC for HCC spare data area.
2088 - 2111	Not used.

1.4 Device Description

This table summarizes the properties of the relevant Hynix device types:

	HY27UF08	HY27UF16
Size (Gb)	1	2
Page size	(2K + 64 spare) bytes	(IK + 32 spare) words
Pages per block	64	64
Block size	(128K + 4K spare) bytes	(64K + 2K spare) words
Planes	2	2
Bus width	x8	x16

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.

1.5 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_hy27uf08	The FTL media driver package described in this document.

Documents

For an overview of HCC file systems and data storage, 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 Hynix HY27UF08 User Guide

This is this document.

1.6 Change History

This section describes past changes to this manual.

- To download earlier manuals, see Archive: FTL NAND Media Driver for Hynix HY27UF08 User Guide .
- For the history of changes made to the package code itself, see History: media_drv_ftl_nand_hy27uf08.

The current version of this manual is 1.20. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.20	2017-08-11	1.02	Corrected *Packages* list.
1.10	2017-06-27	1.02	New *Change History* format.
1.00	2015-11-02	1.02	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 file and PSP files.

2.1 API Header File

The file **src/api/api_ftl_nand_hy27uf08_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_hy27uf08_ecc.h** contains all the configurable parameters of the system. Configure these as required. For details of these options, see Configuration Options.

2.3 Source Code Files

The file **src/media-drv/ftl/drivers/nand/hynix/nand_hy27uf08_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 and other elements the core code needs to call, depending on the hardware. They are in the directory **src/psp/target/nand_hy27uf08**.

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

File	Description
psp_nand_hy27uf08.c	Source code.
psp_nand_hy27uf08.h	Header file.

2.5 Version Files

The file **src/version/ver_ftl_nand_hy27uf08_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_hy27uf08_ecc.h**. This section lists the available configuration options and their default values.

NAND_HY27UF08_ECC_ID

The NAND ID. The default is ((0xAD << 0) | (0xDA << 8) | (0x10 << 16) | (0x95 << 24)).

NAND_HY27UF08_ECC_BLOCK_NUM

The number of erasable blocks in the target flash array. The default is 2048.

NAND_HY27UF08_ECC_PAGE_DATA_SIZE

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

NAND_HY27UF08_ECC_PAGE_TOTAL_SIZE

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

NAND_HY27UF08_ECC_PAGE_PER_BLOCK

The number of pages per erasable block. The value must not exceed MDRIVER_FTL_MAX_PAGE_PER_BLOCK. The default is 64.

NAND_HY27UF08_ECC_FREE_BLOCK_AVAILABLE

The number of free blocks. The value must not exceed MDRIVER_FTL_MAX_FREE_BLOCKS. The default is 40.

NAND_HY27UF08_ECC_LOG_BLOCK_AVAILABLE

The number of log blocks. The value must not exceed MDRIVER_FTL_MAX_LOG_BLOCK_AVAIL. The default is 4; do not set it below this.

NAND_HY27UF08_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, inclusive.

NAND_HY27UF08_ECC_MAPBLOCK_SHADOW

The number of map shadow blocks. The default is 1; do not set a value below this.

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.

NAND_HY27UF08_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.

NAND_HY27UF08_ECC_WEAR_STATIC_LIMIT

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

NAND_HY27UF08_ECC_WEAR_STATIC_COUNT

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

LL_TEST

Set this to 1 for a low level driver test. The default is 0.

3.1 Restrictions

`NAND_HY27UF08_ECC_FREE_BLOCK_AVAILABLE * 2` must not exceed
`NAND_HY27UF08_ECC_PAGE_DATA_SIZE / 2`

The following must not be less than 8:

```
NAND_HY27UF08_ECC_FREE_BLOCK_AVAILABLE - (NAND_HY27UF08_ECC_NUM_OF_DIF_MAPBLOCK
* NAND_HY27UF08_ECC_MAPBLOCK_SHADOW + 1) - NAND_HY27UF08_ECC_LOG_BLOCK_AVAILABLE
```

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_hy27uf08_ecc_init()** function.
2. **nand_hy27uf08_ecc_init()** returns a pointer to a *t_ftl_driver* structure containing a set of functions for accessing the media driver.

4.1 nand_hy27uf08_ecc_init

This function initializes the driver for this device.

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

Format

```
t_ftl_ret nand_hy27uf08_ecc_init (
    uint32_t           drvnum,
    t_ftl_driver * *   pps_ftl_driver )
```

Arguments

Argument	Description	Type
drvnum	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
LL_OK	Successful execution.
LL_ERROR	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 Hynix 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_hy27uf08_ecc_init, 0U }
  , { ftl_nor_init, 0U }
  , { ftl_nor_init, 1U }
};
```

4.3 Error Codes

If a function executes successfully, it returns with LL_OK, a value of 0. This table shows the meaning of the error codes:

Return Value	Value	Description
LL_OK	0	Successful execution.
LL_ERROR	1	Operation failed.

4.4 t_ftl_driver

The **nand_hy27uf08_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 macros:

Macro	Package	Element	Description
PSP_RD_LE32	psp_base	psp_endianness	Reads a 32 bit value stored as little-endian from a memory location.
PSP_WR_LE32	psp_base	psp_endianness	Writes a 32 bit value to be stored as little-endian to a memory location.

The module makes use of the following PSP functions. These functions are provided by the PSP to perform various tasks. Their design makes it easy for you to port them to work with your hardware solution. The package includes samples in the **src/psp/target/nand_hy27uf08/psp_nand_hy27uf08.c** file.

Macro	Description
psp_nand_hy27uf08_init()	Initializes the device.
psp_nand_hy27uf08_wait_ready()	Waits until the flash becomes ready or the timeout expires.

These functions are described in the following sections.

psp_nand_hy27uf08_init

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

This enables peripheral clocks and initializes the flash controller and flash pins for NAND access.

Format

```
uint32_t psp_nand_hy27uf08_init ( void )
```

Arguments

None.

Return Values

Return value	Description
LL_OK	Successful execution.
LL_ERROR	Operation failed.

psp_nand_hy27uf08_wait_ready

This function is provided by the PSP to wait until the flash becomes ready or the timeout expires.

Format

```
uint32_t psp_nand_hy27uf08_wait_ready ( uint32_t ms )
```

Arguments

Argument	Description	Type
ms	The timeout value in milliseconds.	uint32_t

Return Values

Return value	Description
0	The flash is ready.
1	The timeout expired.