FTL NAND Media Driver for Micron MT29FxG01 SPI User Guide

Version 1.40

For use with FTL NAND Media Driver for Micron® MT29FxG01 SPI Devices versions 1.06 and above

Exported on 04/02/2019

HCC Embedded reserves the right to make changes to this document and to the related software at any time and without notice. The information in this document has been carefully checked for its accuracy; however, HCC Embedded makes no warranty relating to the correctness of this document.

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

The component sections are as follows:

- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- Device Description - summarizes the features of the supported devices.
- Packages and Documents – the *Packages* section lists the packages that you need in order to use this module. The *Documents* section lists the relevant user guides.
- Change History – lists the earlier versions of this manual, giving the software version that each manual describes.

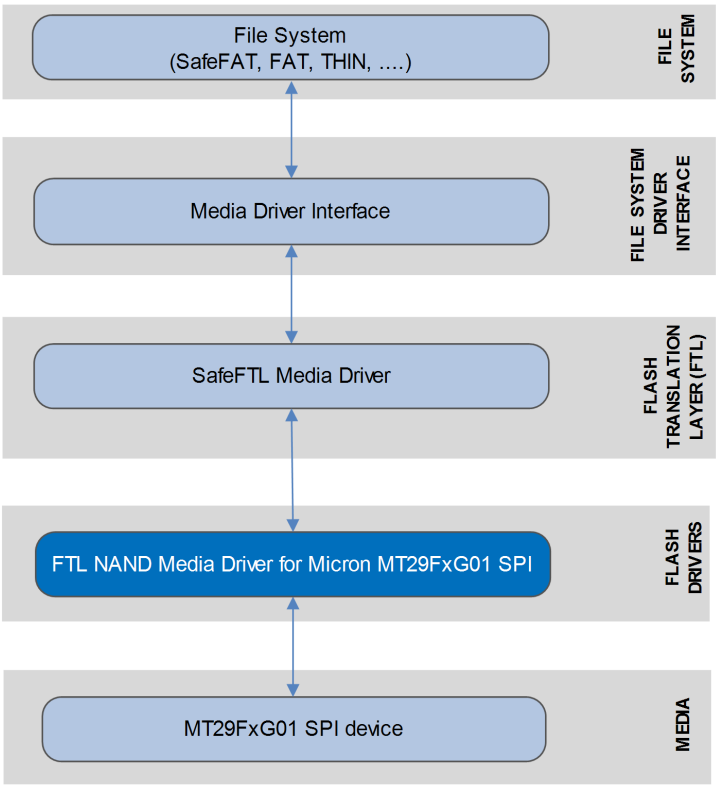
1.1 Introduction

This guide is for those who want to use HCC's FTL NAND Media Driver for Micron® MT29FxG01 Serial Peripheral Interface (SPI) flash devices in their system. This driver supports MT29F1G01 (1Gb), MT29F2G01 (2Gb), MT29F4G01 (4Gb), and MT29F8G01 (8Gb) devices.

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 NAND flash storage device. 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.



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 MT29FxG01 SPI 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 Flash Translation Layer.
- Supports Micron® MT29F1G01 (1Gb), MT29F2G01 (2Gb), MT29F4G01 (4Gb), and MT29F8G01 (8Gb) SPI flash devices.
- 8-bit ECC handling for M7xA variants.

1.3 Device Description

This table summarizes the features of the supported devices:

	MT29F1G01	MT29F2G01	MT29F4G01	MT29F8G01
Size (Gb)	1	2	4	8
Bus width	1, 2 or 4	1, 2 or 4	1, 2 or 4	1, 2 or 4
Plane size	2 planes x 512 blocks per plane	2 planes x 1024 blocks per plane	2 planes x 2048 blocks per plane	2 planes x 4096 blocks per plane
Page size	2048 bytes + 64 spare bytes	2048 bytes + 64 spare bytes	2048 bytes + 64 spare bytes	2048 bytes + 64 spare bytes
Blocks	1024	2048	4096	8192
Block size	64 pages (128K + 4K bytes)	64 pages (128K + 4K bytes)	64 pages (128K + 4K bytes)	64 pages (128K + 4K bytes)
Internal ECC	4 bit	4 bit	4 bit	4 bit
SLC	Yes	Yes	Yes	Yes

HCC has tested this driver with the following devices:

1.8V devices	3.3V devices
• MT29F1G01ABBFD12 • MT29F4G01ABBFD12 • MT29F8G01ADBFD12	• MT29F1G01ABAFD12 • MT29F2G01ABAGD12 • MT29F4G01ABAFD12 • MT29F8G01ADAFD12

To test the memory devices, HCC mounted them on a Secure Digital (SD) form factor board, providing a convenient interface to development boards with SD card slot support for SPI communication.

HCC created its own boards for testing. To request an HCC Embedded SD form factor board to validate this solution with Micron® SPI NAND Flash, contact an HCC Embedded sales office or local distributor.

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_mt29fxg01_spi	The media driver package described in this document.
psp_template_spi	The SPI Platform Support Package (PSP).

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 Micron MT29FxG01 SPI User Guide

This is this document.

1.5 Change History

This section describes past changes to this manual.

- To download this manual or a PDF describing an earlier software version, see FTL Media Driver PDFs.
- For the history of changes made to the package code itself, see History: media_drv_ftl_nand_mt29fxg01_spi.

The current version of this manual is 1.40. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
1.40	2019-04-02	1.07	Added 8Gb devices to *System Overview* sections.
1.30	2018-01-24	1.06	Added support for MT29F4G01Ax devices - new configuration option. Removed ref. to **psp_memcpy()** from *PSP Porting*.
1.20	2017-08-11	1.04	Corrected *Packages* list.
1.10	2017-06-27	1.04	New *Change History* format.
1.00	2016-02-15	1.03	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.

2.1 API Header File

The file **src/api/api_ftl_nand_mt29fxg01.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_mt29fxg01.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/micron/mt29fxg01.c** holds the source code for the media driver. **This file should only be modified by HCC**.

2.4 Version File

The file **src/version/ver_ftldrv_mt29fxg01.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_ftldrv_mt29fxg01.h**. This section lists the available options and their default values.

MT29FXG01_VARIANT

The device variant. Possible values are MT29FXG01_VARIANT_M6XA (the default) and MT29FXG01_VARIANT_M7XA.

MT29F1G01_SUPPORT

Keep the default of 1 to provide support for MT29F1G01 flash memory. Otherwise, set this to 0.

MT29F2G01_SUPPORT

Keep the default of 1 to provide support for MT29F2G01 flash memory. Otherwise, set this to 0.

MT29F4G01_SUPPORT

Keep the default of 1 to provide support for MT29F4G01 flash memory. Otherwise, set this to 0.

MT29FXG01_SPI_UID

The SPI unit ID. The default is 0.

MT29FXG01_SPI_BAUD_RATE

The frequency for SPI. The default is 5000000.

MT29FXG01_FREE_BLOCK_MIN

The number of free blocks. The value must not exceed MDRIVER_FTL_MAX_FREE_BLOCKS. The default is 16.

The total number of management blocks is (MT29FXG01_FREE_BLOCK_MIN + maximum number of bad blocks). This depends on the device detected.

MT29FXG01_LOG_BLOCK_AVAILABLE

The number of log blocks. The value must not exceed MDRIVER_FTL_MAX_LOG_BLOCK_AVAIL. The default is 5; do not set it below this.

MT29FXG01_NUM_OF_DIF_MAPBLOCK

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

MT29FXG01_MAPBLOCK_SHADOW

The number of map shadow blocks. The default is 2. The minimum value is 1.

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.

MT29FXG01_RESERVED_BLOCKS

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

MT29FXG01_WEAR_STATIC_LIMIT

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

MT29FXG01_WEAR_STATIC_COUNT

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

MT29FXG01_LL_TEST

Set this to 1 when testing this driver with *ftldrv_test*. The default is 0.

3.1 Restriction

The following must not be less than 8:

```
MT29FXG01_FREE_BLOCK_MIN – ( MT29FXG01_NUM_OF_DIF_MAPBLOCK *
MT29FXG01_MAPBLOCK_SHADOW + 1 ) – MT29FXG01_LOG_BLOCK_AVAILABLE
```

4 Application Programming Interface

This section describes the single Application Programming Interface (API) function and the structure it uses.

When the media driver is used:

1. The file system calls the media driver's **ftldrv_mt29fxg01_init()** function.
2. **ftldrv_mt29fxg01_init()** returns a pointer to a t_ftl_driver structure containing a set of functions for accessing the media driver.

4.1 ftldrv_mt29fxg01_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 ftldrv_mt29fxg01_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 an MT29FxG01 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] =
{
  { ftldrv_mt29fxg01_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 **ftldrv_mt29fxg01_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 SPI functions. These are described in the *HCC SPI Driver PSP User Guide*.

Function	Package	Element	Description
psp_spi_init()	psp_base	psp_spi	Initializes the SPI port.
psp_spi_start()	psp_base	psp_spi	Starts the SPI port.
psp_spi_cs_hi()	psp_base	psp_spi	Sets chip select high.
psp_spi_cs_lo()	psp_base	psp_spi	Sets chip select low.
psp_spi_get_baudrate()	psp_base	psp_spi	Gets the baud rate.
psp_spi_set_baudrate()	psp_base	psp_spi	Sets the baud rate.
psp_spi_rx()	psp_base	psp_spi	Receives a number of bytes.
psp_spi_tx1()	psp_base	psp_spi	Transmits one byte.
psp_spi_lock()	psp_base	psp_spi	Locks the SPI for the specific unit. This can be useful if multiple units are attached to the same SPI bus.
psp_spi_unlock()	psp_base	psp_spi	Unlocks the SPI for the specific unit. This can be useful if multiple units are attached to the same SPI bus.