FTL Media Driver for Adesto Dataflash User Guide

Version 1.20

For use with FTL Media Driver for Adesto® DataFlash versions 1.02 and above

Date: 18-Aug-2017 11:23

Table of Contents

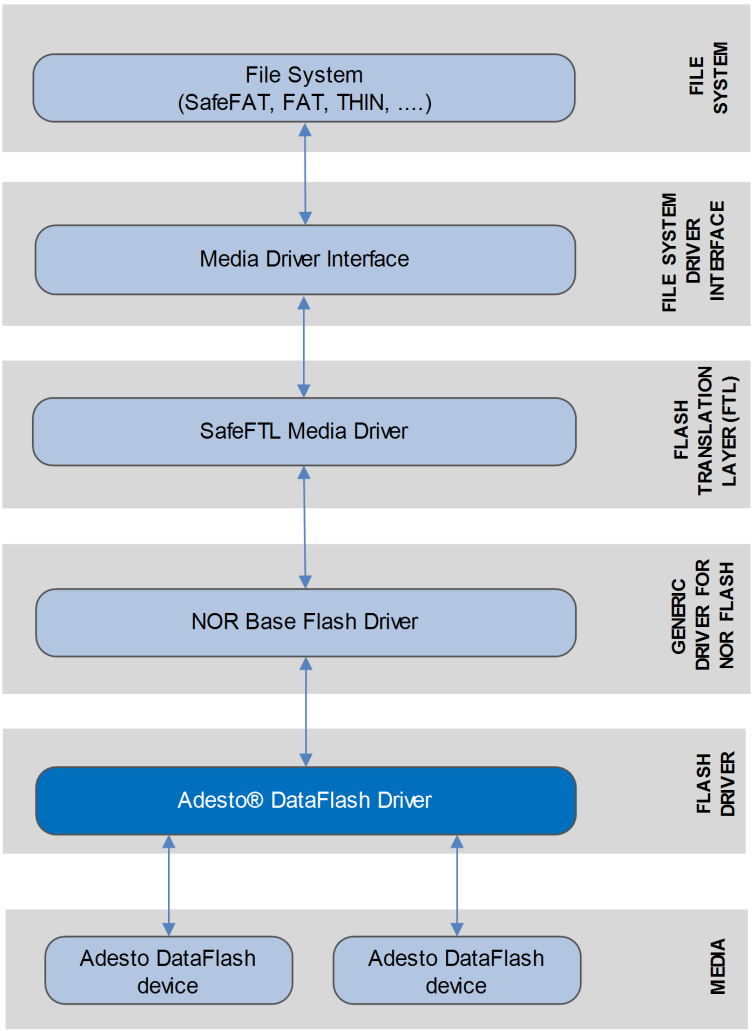
1 System Overview

1.1 Introduction

This guide is for those who want to use HCC Embedded's FTL Media Driver for Adesto® Technologies DataFlash drives in their system. These DataFlash devices have a page size of 512+16 bytes. This guide covers all aspects of configuration and use; read it thoroughly before implementing a driver.

Note: This DataFlash technology was formerly owned by Atmel but is now owned by Adesto® Technologies.

This diagram shows how an Adesto® DataFlash driver fits into the system.



This media driver conforms to the *HCC Media Driver Interface Specification*.

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), a system for attaching arrays of flash to a media driver, is the SafeFTL media driver. This has its own manual.
- The NOR base flash driver is a generic driver which handles the NOR flash drivers. Many different NOR flash drives can be attached simultaneously. The NOR base flash driver is designed for use with most standard types of NOR flash, in simple or complex configurations. The NOR base flash driver has its own manual.
- The Adesto® DataFlash driver controls one flash drive connected to the system. It interfaces to the NOR base flash driver.
- An entry for the Adesto® DataFlash drive must be added to the SafeFTL drive list.

The following layout is used:

- The data area of pages is stored contiguously, starting at address 0 of each block.
- The 16 byte spare area of pages is stored contiguously, starting at address SPARE_REGION_OFFSET of the block.
- We calculate and store a four byte checksum for all the data and spare bytes at the end of each page's spare area.
- Data integrity on NOR devices can be verified by simple checksums.

The NOR layer is assumed to support partial page writes.

1.2 Feature Check

For a full list of SafeFTL features, see the *HCC SafeFTL User Guide*.

The features especially relevant to Adesto® Technologies DataFlash are as follows:

- The system can handle up to 4TB in a single wear-leveled array.
- The system can handle arrays of flash greater than 4TB.

The system supports the following Adesto® DataFlash devices:

- AT45DB021E
- AT45DB041E
- AT45DB081E
- AT45DB161E
- AT45DB321E
- AT45DB641E
- AT45DQ081
- AT45DB161
- AT45DB321

It also supports the following legacy devices:

- AT45DB021B/D
- AT45DB041B/D
- AT45DB081B/D
- AT45DB161B/D
- AT45DB321B/D
- AT45DB641B/D

1.3 Packages and Documents

Packages

All HCC software is delivered as a set of modular packages. The table below lists the packages which 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_dflash	The package described in this document.
psp_template_spi	The package of SPI functions that this package uses.
psp_template_base	The base 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 specification for the upper layer interface that SafeFTL uses. This means that SafeFTL can be used as a set of drives by any file system using this Media Driver Interface standard.

HCC SafeFTL User Guide

The user's guide for HCC's Safe Flash Translation Layer, SafeFTL.

HCC FTL NOR Base Flash Driver User Guide

This describes the NOR base flash driver which handles the Adesto® DataFlash driver.

HCC FTL Media Driver for Adesto DataFlash User Guide

This is this document.

1.4 Change History

This section describes past changes to this manual.

- To download earlier manuals, see Archive: FTL Media Driver for Adesto Dataflash User Guide.
- For the history of changes made to the package code itself, see History: media_drv_ftl_dflash.

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	2017-08-18	1.02	Updated *Packages* list.
1.10	2017-06-26	1.02	New *Change History* format.
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 file.

2.1 API Header File

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

2.2 Configuration File

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

2.3 Source File

The file **src/media-drv/ftl/drivers/dflash/ftl_dflash.c** is the main source code file. **This file should only be modified by HCC.**

2.4 Version File

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

3 Configuration Options

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

FTL_DFLASH_SPI_UNIT

The total size of the page. The default is 0.

FTL_DFLASH_SPI_BAUDRATE

The SPI baud rate. The default is 18000000.

FTL_DFLASH_MIN_PAGE_DATA_SIZE

The minimum page size for supported devices in bytes. Set this depending on the target device. The default is 512.

FTL_DFLASH_FREE_BLOCK_AVAILABLE

The number of free blocks. The default is 24. These blocks absorb created bad blocks so should be increased proportionally to the size of the flash drive. The HCC reference drivers contain tested settings for this value.

FTL_DFLASH_LOG_BLOCK_AVAILABLE

The number of free log blocks available. The default is 7; do not set it below this. To ensure the efficiency of the system this number should be increased proportionally to the size of the flash drive. The HCC reference drivers contain tested settings for this value.

FTL_DFLASH_NUM_OF_DIF_MAPBLOCK

The number of blocks used for mapping in the system. The default is 2. More map blocks will improve system performance. The maximum useful number of map blocks that can be set is given by ((Number_blocks*8/z_pagedata)+1). The HCC reference drivers contain tested settings for this value.

FTL_DFLASH_MAPBLOCK_SHADOW

The number of map shadow blocks. The default is 4. 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. The HCC reference drivers contain tested settings for this value.

FTL_DFLASH_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.

FTL_DFLASH_WEAR_STATIC_LIMIT

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

FTL_DFLASH_WEAR_STATIC_COUNT

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

FTL_DFLASH_FRAG_PER_MAP

The default is 2.

FTL_DFLASH_MAX_FRAGNUM

The default is (FTL_DFLASH_FRAG_PER_MAP * FTL_DFLASH_NUM_OF_DIF_MAPBLOCK).

3.1 Restrictions

The settings must satisfy the following expressions:

- `FTL_DFLASH_FREE_BLOCK_AVAILABLE < 254`
- `FTL_DFLASH_PAGE_PER_BLOCK < 254`
- `FTL_DFLASH_LOG_BLOCK_AVAILABLE < MDRIVER_FTL_MAX_LOG_BLOCK_AVAIL`
- `FTL_DFLASH_NUM_OF_DIF_MAPBLOCK < 1 || FTL_DFLASH_NUM_OF_DIF_MAPBLOCK > 16`
- `FTL_DFLASH_MAPBLOCK_SHADOW >= 1`
- `FTL_DFLASH_FREE_BLOCK_AVAILABLE - ( FTL_DFLASH_NUM_OF_DIF_MAPBLOCK * FTL_DFLASH_MAPBLOCK_SHADOW + 1 ) - FTL_DFLASH_LOG_BLOCK_AVAILABLE >= 8`
- `FTL_DFLASH_FREE_BLOCK_AVAILABLE * 2 <= FTL_DFLASH_PAGE_DATA_SIZE / 2`

4 Application Programming Interface

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

4.1 ftl_dflash_init

Use this function to initialize the driver interface.

Format

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

Arguments

Argument	Description	Type
drvnum	The number of the drive to initialize. This is not used as multiple driver instances are not supported.	uint32_t
pps_ftl_driver	A pointer to a *t_ftl_driver* structure.	t_ftl_driver * *

Return values

Return value	Description
0	Successful execution.
Non-zero	Operation failed.

4.2 t_ftl_driver

The **ftl_dflash_init()** function returns a pointer to a *t_ftl_driver* structure. This structure is passed to the flash driver by the call.

The flash driver fills in all the function pointers for the FTL to use when using the flash drive.

Parameter	Description	Type
user_data	User-defined data. For example, this may be used to identify a specific flash drive.	uint32_t
pf_getphy	A pointer to the **ll_getphy()** function, used to get the physical characteristics of the flash drive.	(* pf_getphy)
pf_read	A pointer to the **ll_read()** function, used to read a page on the flash drive.	(* pf_read)
pf_readpart	A pointer to the **ll_readpart()** function, used to read part of a page on the flash drive.	(* pf_readpart)
pf_write	A pointer to the **ll_write()** function, used to write a page to the flash drive.	(* pf_write)
pf_writedouble	A pointer to the **ll_writedouble()** function, used to write a page from two buffers to the flash drive.	(* pf_writedouble)
pf_erase	A pointer to the **ll_erase()** function, used to erase a block on the flash drive.	(* pf_erase)
pf_isbadblock	A pointer to the **ll_isbadblock()** function, used to check whether a specific block is bad.	(* pf_isbadblock)
pf_readonebyte	A pointer to the **ll_readonebyte()** function, used to read a single byte from the flash drive.	(* pf_readonebyte)

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 Serial Physical Interface (SPI) PSP functions. For details, see 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_set_baudrate()	psp_base	psp_spi	Sets the baud rate.
psp_spi_rx()	psp_base	psp_spi	Receives a number of bytes.
psp_spi_tx()	psp_base	psp_spi	Transmits a number of bytes.
psp_spi_tx1()	psp_base	psp_spi	Transmits one byte.

The module makes use of the following standard PSP macros:

Macro	Package	Element	Description
PSP_RD_LE16	psp_base	psp_endianness	Reads a 16 bit value stored as little-endian from a memory location.
PSP_RD_LE32	psp_base	psp_endianness	Reads a 32 bit value stored as little-endian from a memory location.