SafeFLASH NOR Driver for SPIFI User Guide

Version 1.00

For use with SafeFLASH NOR Driver for SPIFI versions 1.01 and above

Exported on 01/23/2019

Table of Contents

1 System Overview

This chapter contains the fundamental information for this module.

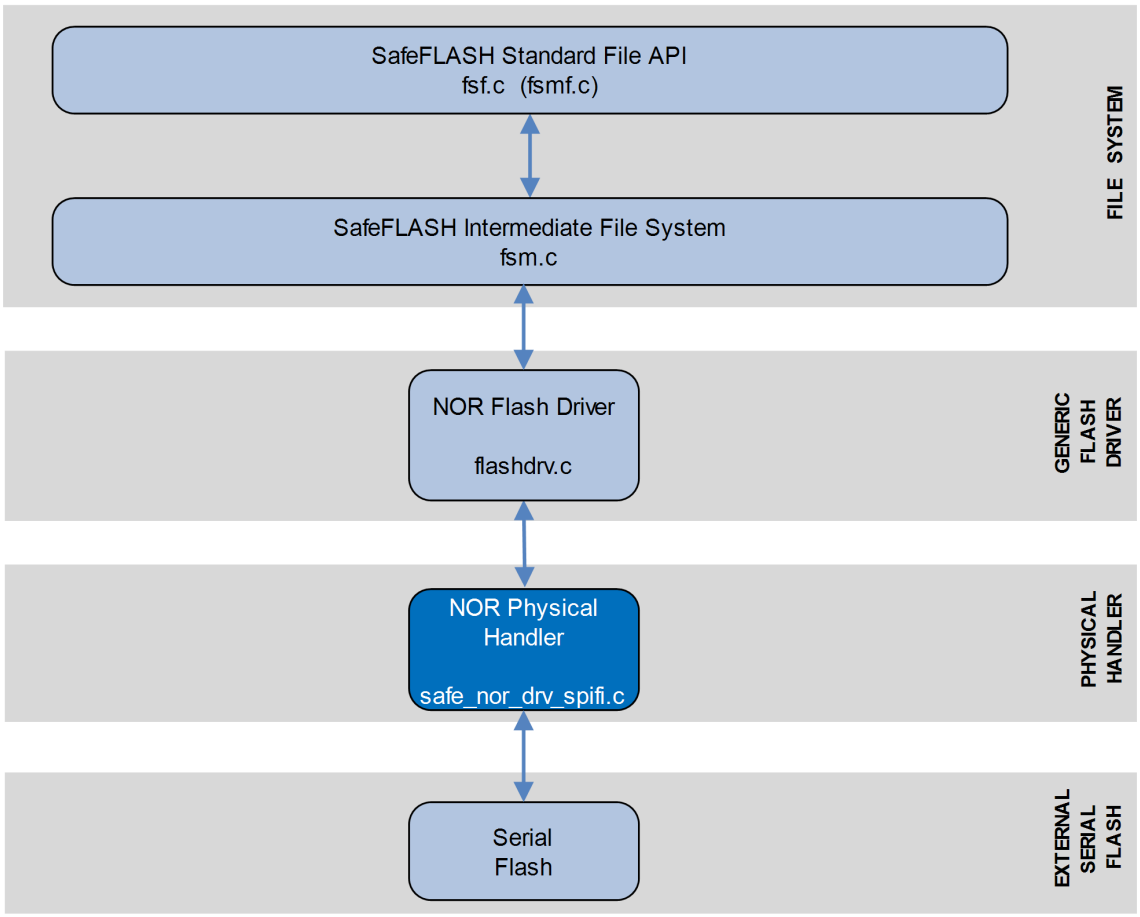
The component sections are as follows:

- Introduction – describes the main elements of the module.
- Feature Check – summarizes the main features of the module as bullet points.
- 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 implement a SafeFLASH NOR driver for Serial Peripheral Interface Flash Interface (SPIFI) devices. This is for use with HCC's SafeFLASH file system. SPIFI provides an interface to external serial flash memory. The serial flash appears in the microcontroller's address space and can be accessed in standard RAM fashion.

This diagram shows the structure of the file system software:



This diagram shows:

- The main SafeFLASH package – this provides the file API and intermediate file system. This is described in the the HCC SafeFLASH File System User Guide.
- The NOR flash driver – the generic device driver for NOR flash, provided by the base NOR package. This driver handles issues of FAT maintenance, wear leveling, and so on. It is described in the HCC SafeFLASH File System NOR Drive User Guide.
- The NOR physical handler – provided by this module, this performs the translation between the driver and the physical flash hardware. This guide shows how to add this to the build. Using the available sample drivers as a model, you can create a driver that meets your specific needs.
- The serial flash – this can be accessed in the standard random access way.

Note:

- In similar HCC products the physical driver is the specific flash chip driver which includes the physical interface handler. However, here this is separated out because the microcontroller manufacturer has provided a special software library for interfacing these devices.
- HCC Embedded offers hardware and firmware development consultancy to assist developers with the implementation of flash file systems.

1.2 Feature Check

The main features of the module are the following:

- Conforms to the HCC Advanced Embedded Framework.
- Designed for integration with both RTOS and non-RTOS based systems.
- Supports NOR Serial Peripheral Interface Flash Interface (SPIFI) devices.
- Supports static and dynamic wear leveling.
- Supports bad block management.
- A sample driver with a porting description is available.

1.3 Fail-safety

This driver for SPIFI NOR flash is designed as part of HCC's SafeFLASH file system. SafeFLASH guarantees a defined level of fail-safety (see the SafeFLASH File System User Guide). For the system to be able to guarantee fail-safety, each component must provide a defined quality of service.

For this driver the following must be guaranteed to ensure the system is fail-safe:

- All write operations must be committed to flash in the sequence in which they are provided to the driver.
- Any write operation that fails must return an error.
- Any erase operation that fails must return an error.
- The system must ensure that there is at most one partially complete write or erase operation. At this point the file system should be restarted so that it can be recovered.

To achieve this, the hardware should ensure that, in the event of a falling voltage approaching the specified minimum programming level of the flash, the system either resets or provides a signal to the software to block access to the flash.

An alternative solution is to add capacitance to the design. This must provide sufficient power that, after a hardware error or reset condition is detected, the active operation on the flash can be completed.

Only by using one of these techniques can the system guarantee correct operation even after an unexpected system reset.

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.
fs_safe_nor	The SafeFLASH NOR flash driver.
fs_safe_nor_drv_spifi	The low level driver package described in this document.

Documents

For an overview of HCC file systems and guidance on choosing a file system, refer to the Product Information section of 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 SafeFLASH File System User Guide

This document describes the base SafeFLASH System.

HCC SafeFLASH File System NOR Drive User Guide

This document describes the SafeFLASH NOR generic driver.

HCC SafeFLASH NOR Driver for SPIFI User Guide

This is this document.

1.5 Change History

To download manuals, see File System PDFs.

For the history of changes made to the package code itself, see History: fs_safe_nor_drv_spifi.

The current version of this manual is 1.00.

Manual version	Date	Software version	Reason for change
1.00	2019-01-23	1.00	First online version.

2 Source File List

The following sections describe 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 file.

2.1 Configuration File

The file **src/config/config_safe_nor_drv_spifi.h** contains the configurable parameters of the system. Configure these as required. For detailed explanation of these options, see Configuration Options.

2.2 System Files

These files are in the directory **src/safe-flash/nor/phy/spifi**. **These files should only be modified by HCC**.

File	Description
safe_nor_drv_spifi.c	SPIFI driver source code.
safe_nor_drv_spifi.h	SPIFI driver header file.

2.3 Platform Support Package (PSP) File

The file **src/psp/target/spifi/psp_spifi.c** provides functions and elements the core code needs to use, depending on the hardware.

Note:

- This is a PSP implementation for the specific microcontroller and board. You may need to modify this to work with a different microcontroller and/or development board; see PSP Porting for details.
- In the package this file is offset to avoid overwriting an existing implementation. Copy it to the root **hcc** directory for use.

2.4 Version File

The file **src/version/ver_safe_nor_drv_spifi.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 configuration options in the file **src/config/config_safe_nor_drv_spifi.h**. This section lists the available options and their default values.

NOR_FIRST_PHY_BLOCK

The first physical block index to be used by the file system. The default is 0.

NOR_PAGE_SIZE

This is purely logical on NOR flashes. The default is 256.

> **Note:** The following three parameters configure the blocks. The default usage is:
>
> - Blocks 0 - 1: two 64K blocks for descriptors.
> - Block 2 - 127: 126 64K blocks used for data storage.

NOR_DESC_SIZE

The descriptor size. The default is 8192.

NOR_DESC_BLOCKS

The number of descriptor blocks. The default is 2.

NOR_CACHE_DESC_SIZE

The descriptor cache size. The default is 2048.

> **Note:** Set the following two parameters according to the type of flash used.

NOR_SECTORSIZE

The logical sector size. The default is 4096.

NOR_BLOCK_SIZE

The block size. The default is (64 * 1024).

NOR_SPIFI_UNIT

The SPIFI ID to use. The default is 0.

4 Integration

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

4.1 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 functions:

Function	Package	Component	Description
psp_memcmp()	psp_base	psp_string	Compares two blocks of memory.
psp_memcpy()	psp_base	psp_string	Copies a block of memory. The result is a binary copy of the data.

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.

Function	Package	Element	Description
psp_spifi_init()	psp_base	psp_spi	Initializes the device.
psp_spifi_start()	psp_base	psp_spi	Starts the device.
psp_spifi_stop()	psp_base	psp_spi	Stops the device.
psp_spifi_delete()	psp_base	psp_spi	Deletes the device, releasing the asociated resources.
psp_spifi_write()	psp_base	psp_spi	Writes to the device.
psp_spifi_erase()	psp_base	psp_spi	Erases part of the device.
psp_spifi_get_id()	psp_base	psp_spi	Gets the device ID, type and manufacturer.

The package includes samples in the PSP file **src/psp/target/spifi/psp_spifi.c**. These functions are described in the following sections.

psp_spifi_init

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

Format

```
int psp_spifi_init (
    uint8_t      uid,
    uint32_t *   p_flash_base,
    uint32_t *   p_density )
```

Arguments

Argument	Description	Type
uid	The unit ID.	uint8_t
p_flash_base	A pointer to the flash base.	uint32_t *
p_density	A pointer to the device size.	uint32_t *

Return Values

Return value	Description
PSP_SPIFI_SUCCESS	Successful execution.
PSP_SPIFI_ERROR	Operation failed.

psp_spifi_start

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

Format

```
int psp_spifi_start ( uint8_t uid )
```

Arguments

Argument	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_SPIFI_SUCCESS	Successful execution.
PSP_SPIFI_ERROR	Operation failed.

psp_spifi_stop

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

Format

```
int psp_spifi_stop ( uint8_t uid )
```

Arguments

Argument	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_SPIFI_SUCCESS	Successful execution.
PSP_SPIFI_ERROR	Operation failed.

psp_spifi_delete

This function is provided by the PSP to delete the device, releasing the associated resources.

Format

```
int psp_spifi_delete( uint8_t uid )
```

Arguments

Argument	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_SPIFI_SUCCESS	Successful execution.
PSP_SPIFI_ERROR	Operation failed.

psp_spifi_write

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

Format

```
int psp_spifi_write (
    uint8_t     uid,
    uint32_t    addr,
    uint8_t *   src,
    uint32_t    len )
```

Arguments

Argument	Description	Type
uid	The unit ID.	uint8_t
addr	The address to write to.	uint32_t
src	The source of the data.	uint8_t *
len	The length of the data to write.	uint32_t

Return Values

Return value	Description
PSP_SPIFI_SUCCESS	Successful execution.
PSP_SPIFI_ERROR	Operation failed.

psp_spifi_erase

This function is provided by the PSP to erase part of the device.

Format

```
int psp_spifi_erase (
    uint8_t    uid,
    uint32_t   addr,
    uint32_t   len )
```

Arguments

Argument	Description	Type
uid	The unit ID.	uint8_t
addr	The address to erase from.	uint32_t
len	The length of the data to erase.	uint32_t

Return Values

Return value	Description
PSP_SPIFI_SUCCESS	Successful execution.
PSP_SPIFI_ERROR	Operation failed.

psp_spifi_get_id

This function is provided by the PSP to get the device ID, type, and manufacturer.

Format

```
int psp_spifi_get_id (
    uint8_t     uid,
    uint8_t *   mfg,
    uint8_t *   dev_type,
    uint8_t *   dev_id )
```

Arguments

Argument	Description	Type
uid	The unit ID.	uint8_t
mfg	On return, the device manufacturer.	uint8_t *
dev_type	On return, the device type.	uint8_t *
dev_id	On return, the device ID.	uint8_t *

Return Values

Return value	Description
PSP_SPIFI_SUCCESS	Successful execution.
PSP_SPIFI_ERROR	Operation failed.