SafeFLASH NOR Driver for CFI and SFDP SPI Flash User Guide

Version 2.10

For use with SafeFLASH NOR Driver for CFI and SFDP SPI Flash versions 1.12 and above

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.
- Fail-safety – defines fail-safety and describes the quality of service that SafeFLASH provides.
- 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.

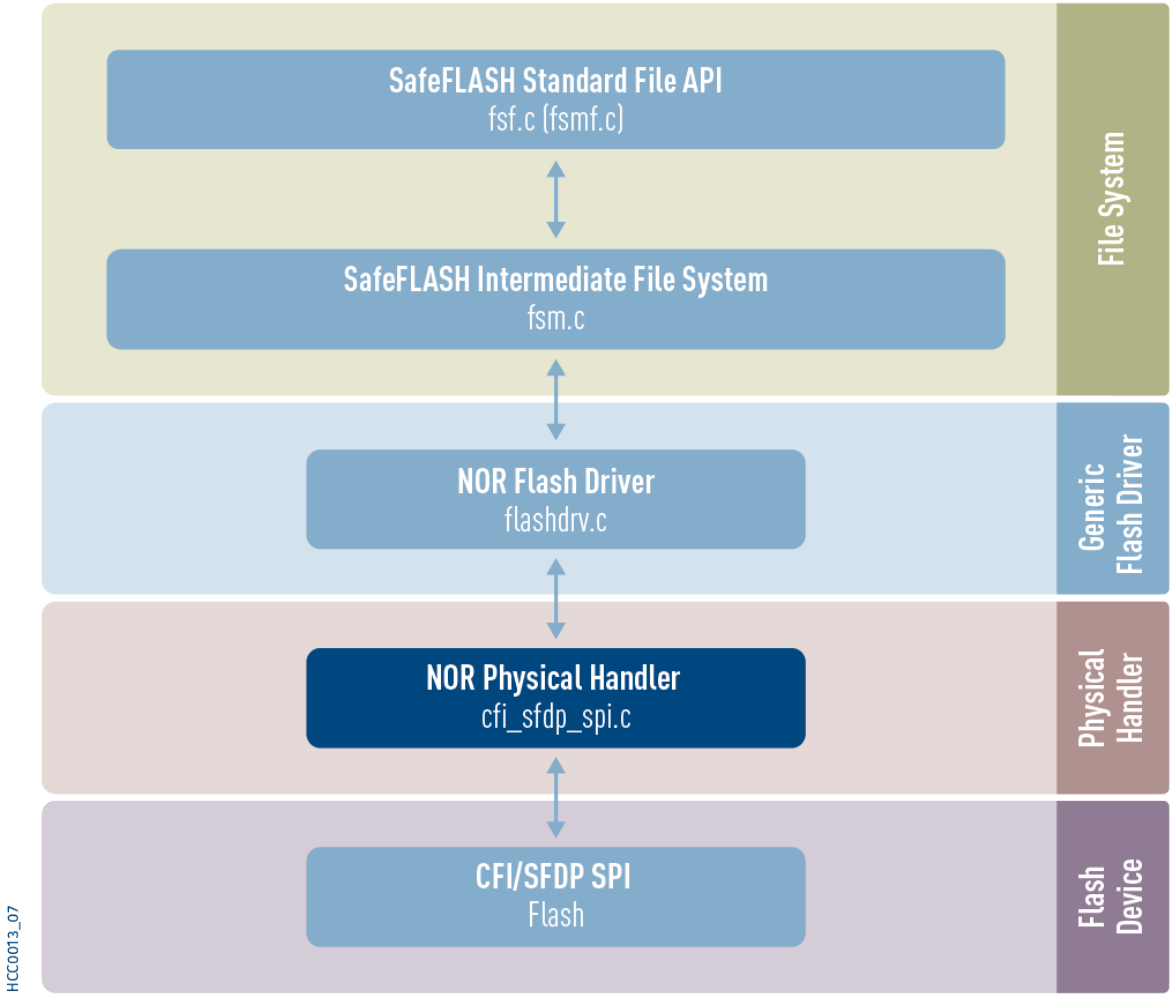
1.1. Introduction

This guide is for those who want to implement a SafeFLASH NOR driver for use with CFI/SFDP SPI flash devices. This is for use with HCC's SafeFLASH file system. These flash devices support the Common Flash Memory Interface (CFI) and Serial Flash Discoverable Parameters (SFDP) standards on a Serial Peripheral Interface (SPI) or Quad Serial Peripheral Interface (QSPI).

CFI and SFDP are standards widely used by flash manufacturers that allow software to obtain detailed information about the internal configuration of the flash device. This driver uses that information to configure the system automatically to use the attached flash type.

The SafeFLASH file system driver design is highly portable while still maintaining excellent performance. The basic device architecture includes a high level driver for each general media type that shares some common properties. This driver handles issues of FAT maintenance, wear leveling, and so on.

The following 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 *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 – the NOR device driver for the CFI/SFDP SPI flash. 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 CFI/SFDP SPI flash device.

Note: 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 flash connected by SPI or Quad SPI (QSPI).
- Supports all common NOR devices.
- Can handle any array of NOR flash.
- Supports the Common Flash Memory Interface (CFI) standard for NOR flash devices.
- Supports the Serial Flash Discoverable Parameters (SFDP) standard for serial NOR flash devices.

1.3. Fail-safety

This driver for serial 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_cfi_sfdp_spi	The NOR driver for CFI and SFDP package described in this document.
psp_template_spi	The SPI Platform Support Package (PSP).

Documents

For an overview of HCC file systems and guidance on choosing a file system, 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 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 driver that is used with the NOR CFI/SFDP SPI package.

HCC SafeFLASH NOR Driver for CFI and SFDP SPI Flash 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 File System PDFs.
- For the history of changes made to the package code, see History: fs_safe_nor_drv_cfi_sfdp_spi.

The current version of this manual is 2.10. The full list of versions is as follows:

Manual version	Date	Software version	Reason for change
2.10	2020-3-02	1.12	Added new files **config_safe_nor_cfi_sfdp.c** and **api_safe_nor_cfi_sfdp.h**. Replaced **psp_stm32f222ze_nucleo_qspi** PSP with **psp_stm32f7xx_nucleo_qspi**. Removed PSP config file.
2.00	2020-02-03	1.11 R2	New document template.
1.50	2019-11-19	1.11 R2	Added QSPI reference PSP for Nucleo-F722ZE board.
1.40	2019-10-31	1.11	Added QSPI references, mainly in *Configuration Options*.
1.30	2019-01-23	1.09	Corrected diagram in *Introduction*.
1.20	2017-08-31	1.09	Corrected *Packages* list.
1.10	2017-06-26	1.08	New *Change History* format.
1.00	2017-04-24	1.08	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 files and PSP files.

API Header File

The file **src/api/api_safe_nor_cfi_sfdp.h** is the only file that should be included by an application using this module. This defines the *nor_qspi_descriptor_t* device configuration structure.

Configuration Files

These files are in the directory **src/config**:

File	Description
config_safe_nor_cfi_sfdp.h	Configurable system parameters. Configure these as required.
config_safe_nor_cfi_sfdp.c	Defines the *nor_qspi_table* of settings for different devices.

System Files

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

File	Description
cfi_sfdp_spi.c	SPI driver source code.
cfi_sfdp_spi.h	SPI driver header file.

Version File

The file **src/version/ver_safe_nor_cfi_sfdp.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.

Platform Support Package (PSP) Files

There are two sets of files.

psp_template

These files in the directory **src/psp/target/safe_nor_cfi_sfdp** define the psp_safe_nor_cfi_sfdp_delay() function. Modify these files as required for your hardware; see PSP Porting.

Note:

- These are PSP implementations for the specific microcontroller and board; you may need to modify these to work with a different microcontroller and/or development board.
- In the package these files are offset to avoid overwriting an existing implementation. Copy them to the root **hcc** directory for use.

The files are as follows:

File	Description
psp_safe_cfi_sfdp.c	Source code defining the **psp_safe_nor_cfi_sfdp_delay()** function. This waits at least 100 ns, the chip-select deselect time.
psp_safe_cfi_sfdp.h	Header file.

This PSP also has a version file, **ver_psp_safe_nor_cfi_sfdp.h**.

Nucleo-F7xx QSPI Board

These files in the directory **psp_stm32f7xx_nucleo_qspi/src** define the PSP for this board.

File	Description
psp/include/psp_qspi.h	Transfer mode settings and **psp_qspi_xxx()** function definitions.
psp/target/safe_nor_cfi_sfdp/psp_ftl_nor_cfi_sfdp.c and **.h**	Source code and header file defining **psp_safe_nor_cfi_sfdp_delay()**.
psp/target/include/hcc_stm32f7xx_regs.h	Registers file.
psp/target/qspi/psp_qspi.c	Source code of **psp_qspi_xxx()** functions.

This PSP has the following version files:

File	Description
ver_psp_ftl_nor_cfi_sfdp.c	**psp_safe_nor_cfi_sfdp_delay()** files version.
ver_psp_proc_reg.h	Registers file version.
ver_psp_qspi.h	PSP version.

3. Configuration Options

There are two configuration files, described below.

3.1. config_safe_nor_cfi_sfdp.h

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

NOR_PAGE_SIZE

The maximum number of bytes for the page program command. The default is 256.

NOR_BLOCK_THRESHOLD

The block sector size threshold. Below this number of blocks, a 4K sector size is preferred. The default is 128.

> **Note:** For full instructions on how to set up the following options, see the following sections in the *HCC SafeFLASH File System NOR Drive User Guide:*
>
> - Sectors and File Storage
> - Physical Device Usage - this describes all the block types in detail and gives examples.

NOR_BLOCKSTART

Specify the space before the block start that is not to be used by the file system. The default is 0.

NOR_SECTORSIZE

The logical sector size. This must be less than or equal to the block size (*g_block_size*/NOR_BLOCK_SIZE or NOR_VIRTUAL_BLOCK_SIZE). The default is 4096.

NOR_DESCSIZE

The descriptor size. This must be less than or equal to the block size (*g_block_size*/NOR_BLOCK_SIZE or NOR_VIRTUAL_BLOCK_SIZE). The default is 16384.

NOR_CACHEDESCSIZE

The cache size. This must be less than the descriptor size (NOR_DESCSIZE). The default is 2048.

NOR_VIRTUAL_BLOCK_SIZE

Physical blocks can be joined to a larger virtual block (to allow use of a larger descriptor block). Keep the default of 0 to disable virtual blocks, otherwise it is must be a multiple of *g_block_size*/NOR_BLOCK_SIZE.

Note: The following three parameters are only used if flash identification is not possible using SFDP or CFI.

NOR_BLOCK_SIZE

The block size. The default is 4096.

NOR_NUM_BLOCKS

The number of blocks. The default is 128.

NOR_CMD_ERASE

The erase command for the block size specified by NOR_BLOCK_SIZE. The default is 0x20.

NOR_TBPARM_TOP

If this is non-zero, the driver checks whether the TBPARM bit is set (that is, whether the device is already configured to have parameter sectors on the top of the address space). If this OTP bit is not set, it sets it. The default is 1.

NOR_ENABLE_QSPI

Keep the default of 1 to use QSPI instead of SPI. Otherwise set this to 0. In QSPI mode the driver uses four lines to send/receive data, but only one line (MOSI) to send command and address information.

NOR_QSPI_TABLE_SIZE

The number of devices in the table. The default is 18.

NOR_SPI_UNIT

The ID of the SPI or QSPI to use. The default is 0.

NOR_SPI_BAUDRATE

The SPI/QSPI clock frequency in Hz . The default is 50000000.

NOR_TMO_ERASE_MS

The erase timeout in milliseconds. The default is 3000.

NOR_TMO_PROGRAM_MS

The program timeout in milliseconds. The default is 500.

NOR_TMO_WREN_MS

The write enable timeout in milliseconds. The default is 5.

NOR_TMO_WRITE_SR_MS

The write status register timeout in milliseconds. The default is 8.

NOR_CFI_SFDP_TMO_RESET_US

The reset timeout in microseconds. The default is 260.

3.2. config_safe_nor_cfi_sfdp.c

The file **config_safe_nor_cfi_sfdp.c** defines the *nor_qspi_table* of settings for different devices. See the dscription of the *nor_qspi_descriptor_t* structure for more details of each element.

Note:

- Processing is linear so put the more narrowly filtered flash devices uppermost in the list below each manufacturer.
- In the flash name xxx means replacable by any character - usually density.

The default settings are shown below:

```
const nor_qspi_descriptor_t  nor_qspi_table[NOR_QSPI_TABLE_SIZE] =
{
  /* MACRONIX */
  {                             /* MX25 and MX66L family */
    0xC2000000u, 0x00000000u    /* Flash identifier */
    , 0xFF000000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0xffu                     /* 1-1-4 write mode cmd */
    , 0x38u                     /* 1-4-4 write mode cmd */
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x40u                     /* Quad Enable bit mask */
  }

  /* ISSI */
  , {                           /* IS25LPxxx */
    0x9D600000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0x02                      /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x40u                     /* Quad Enable bit mask */
  }
  , {                           /* IS25WPxxx */
    0x9D700000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0x02                      /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
```

```
  , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
  , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
  , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
  , 0x40u                     /* Quad Enable bit mask */
}
, {                           /* IS25LQxxx */
  0x9D400000u, 0x00000000u    /* Flash identifier */
  , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
  , 0x02u                     /* 1-1-1 write mode cmd */
  , 0x32u                     /* 1-1-4 write mode cmd */
  , 0xffu                     /* 1-4-4 write mode cmd */
  , 0xffu                     /* 4-4-4 write mode cmd */
  , 0u                        /* 1-1-1 read mode cmd */
  , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
  , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
  , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
  , 0x40u                     /* Quad Enable bit mask */
}

/* MICRON */
, {                           /* MT25Q family */
  0x20000010u, 0x44000000u    /* Flash identifier */
  , 0xFF0000FFu, 0xFF000000u /* Flash identifier's mask */
  , 0x02u                     /* 1-1-1 write mode cmd */
  , 0x32u                     /* 1-1-4 write mode cmd */
  , 0x38u                     /* 1-4-4 write mode cmd */
  , 0x02u                     /* 4-4-4 write mode cmd */
  , 0u                        /* 1-1-1 read mode cmd */
  , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
  , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
  , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
  , 0u                        /* Quad Enable bit mask */
}
, {                           /* N25Q family */
  0x20000010u, 0x00000000u    /* Flash identifier */
  , 0xFF0000FFu, 0xFF000000u /* Flash identifier's mask */
  , 0x02u                     /* 1-1-1 write mode cmd */
  , 0x32u                     /* 1-1-4 write mode cmd */
  , 0xffu                     /* 1-4-4 write mode cmd */
  , 0x02u                     /* 4-4-4 write mode cmd */
  , 0x03u                     /* 1-1-1 read mode cmd */
  , 0x6bu, 8u                 /* 1-1-4 read mode cmd and dummy cycles */
  , 0xebu, 10u                /* 1-4-4 read mode cmd and dummy cycles */
  , 0xebu, 10u                /* 4-4-4 read mode cmd and dummy cycles */
  , 0x80u                     /* Quad Enable bit mask */
}

/* WINBOND */
, {                           /* W25QxxxFV */
  0xEF400000u, 0x00000000u    /* Flash identifier */
  , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
  , 0x02u                     /* 1-1-1 write mode cmd */
  , 0x32u                     /* 1-1-4 write mode cmd */
  , 0xffu                     /* 1-4-4 write mode cmd */
```

```
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* W25QxxxFW */
    0xEF600000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* W25QxxxJV */
    0xEF700000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0xffu                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* W25QxxxJW_DTR */
    0xEF800000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }

  /* SPANSION */
  , {                           /* S25FS512S */
    0x01022000u, 0x00810000u    /* Flash identifier */
    , 0xFFFFFF00u, 0x00FF0000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0xffu                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
```

```
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* S25FS064S */
    0x01021700u, 0x00810000u    /* Flash identifier */
    , 0xFFFFFF00u, 0x00FF0000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* S25FLxxxL */
    0x01600000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* S25FLxxxSA */
    0x01000000u, 0x00800000u    /* Flash identifier */
    , 0xFF000000u, 0x00FF0000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0xffu                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }

  /* ADESTO */
  , {                           /* AT25SFxxxA */
    0x1F890000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
```

```
    , 0xffu                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* AT25xLxxxA */
    0x1F420000u, 0x00000000u    /* Flash identifier */
    , 0xFFFF0000u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0xffu                     /* 1-1-4 write mode cmd */
    , 0033u                     /* 1-4-4 write mode cmd */
    , 0x02u                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* AT25SF041 */
    0x1F840100u, 0x00000000u    /* Flash identifier */
    , 0xFFFFFF00u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0xffu                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0xffu                     /* 4-4-4 write mode cmd */
    , 0u                        /* 1-1-1 read mode cmd */
    , 0u, 0u                    /* 1-1-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 1-4-4 read mode cmd and dummy cycles */
    , 0u, 0u                    /* 4-4-4 read mode cmd and dummy cycles */
    , 0x02u                     /* Quad Enable bit mask */
  }
  , {                           /* AT25XE321B */
    0x1F470200u, 0x00000000u    /* Flash identifier */
    , 0xFFFFFF00u, 0x00000000u /* Flash identifier's mask */
    , 0x02u                     /* 1-1-1 write mode cmd */
    , 0x32u                     /* 1-1-4 write mode cmd */
    , 0xffu                     /* 1-4-4 write mode cmd */
    , 0xffu                     /* 4-4-4 write mode cmd */
    , 0x03u                     /* 1-1-1 read mode cmd */
    , 0x6bu, 8u                 /* 1-1-4 read mode cmd and dummy cycles */
    , 0xebu, 2u                 /* 1-4-4 read mode cmd and dummy cycles */
    , 0xffu, 0u                 /* 4-4-4 read mode cmd and dummy cycles */
    , 0x01u                     /* Quad Enable bit mask */
  }
};
```

4. nor_qspi_descriptor_t

The *nor_qspi_descriptor_t* structure defines the device configuration.

Note the following:

- Zeroes in commands mean that the flash will automatically try to obtain information from the CFI/SFDP structure. This is only available with read commands. 0xff means that that particular command is not supported.
- Only provide read parameters if SFDP or CFI is not available, or the intention is to overwrite the defaults stored in flash.
- In the flash name xxx means replacable by any character - usually density.

Element	Type	Description
id_hi	uint32_t	Flash identifier. Because some flash manufacturers use a different approach to the flash ID, two 32 bit identifiers are stored here, this and the next item. The two are treated in software as one 64 bit ID. In the flash name xxx means replacable by any character, usually density.
id_lo	uint32_t	Flash identifier.
id_mask_hi	uint32_t	Mask for *id_hi*. This allows you to select the bits that are important in differentiating each flash device. A bit set in the mask means that the corresponding bit in ID is important in identifying the correct flash type. Zero means the bit is not used for identification - like density, vendor-specific length, etc.
id_mask_lo	uint32_t	Mask for *id_lo*.
wr_mode_1_1_1_cmd	uint8_t	1-1-1 write mode command.
wr_mode_1_1_4_cmd	uint8_t	1-1-4 write mode command.
wr_mode_1_4_4_cmd	uint8_t	1-4-4 write mode command.
wr_mode_4_4_4_cmd	uint8_t	4-4-4 write mode command.
rd_mode_1_1_1_cmd	uint8_t	1-1-1 read mode command.
rd_mode_1_1_4_cmd	uint8_t	1-1-4 read mode command.
rd_mode_1_1_4_dummy	uint8_t	1-1-4 read mode command dummy cycles.
rd_mode_1_4_4_cmd	uint8_t	1-4-4 read mode command.
rd_mode_1_4_4_dummy	uint8_t	1-4-4 read mode command dummy cycles.
rd_mode_4_4_4_cmd	uint8_t	4-4-4 read mode command.
rd_mode_4_4_4_dummy	uint8_t	4-4-4 read mode command dummy cycles.
qe_bit_mask;	uint8_t	Quad Enable bit mask. If this is 0, no register write occurs.

5. 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 file **config_ftl_nor_cfi_sfdp.h** defines configuration options for the Nucleo-F722ZE QSPI board.

The files **psp_safe_cfi_sfdp.c** and **psp_safe_cfi_sfdp.h** define the psp_safe_nor_cfi_sfdp_delay() function. Modify these files as required for your hardware.

QSPI Functions

The functions used by the Nucleo-F722ZE QSPI Board PSP are as follows. For details, see the detailed descriptions.

Function	Description
psp_qspi_init()	Initializes a QSPI port and allocates the required resources.
psp_qspi_start()	Starts a QSPI port.
psp_qspi_stop()	Stops a QSPI port.
psp_qspi_delete()	Deletes a QSPI port and releases the resources it used.
psp_qspi_get_baudrate()	Gets the baud rate.
psp_qspi_set_baudrate()	Sets the baud rate.
psp_qspi_lock()	Locks the QSPI for the specific unit.
psp_qspi_unlock()	Unlocks the QSPI for the specific unit.
psp_qspi_register_isr()	Registers the ISR ID for a QSPI unit. This only applies if ISRs are supported.
psp_qspi_transfer()	Transmits/receives data over the QSPI.
psp_qspi_wait_status()	Waits until the status register matches a value.

5.1. psp_safe_nor_cfi_sfdp_delay

Use this function to wait for at least 100 ns, which is the chip select/deselect time.

Format

```
void psp_safe_nor_cfi_sfdp_delay ( void )
```

Arguments

None.

Return Values

None.

5.2. QSPI Functions

This section contains the QSPI Functions.

psp_qspi_init

Use this function to initialize a QSPI port.

Note: You must call this before any other QSPI function.

Format

```
t_qspi_ret psp_qspi_init ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_delete

Use this function to delete a QSPI port and release the associated resources.

Format

```
t_qspi_ret psp_qspi_delete ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_stop

Use this function to stop a QSPI port.

Format

```
t_qspi_ret psp_qspi_stop ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_start

Use this function to start a QSPI port.

Note: You must call **psp_qspi_init()** before this to initialize the module.

Format

```
t_qspi_ret psp_qspi_start ( uint8_t uid )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_register_isr

Use this function to register the ISR ID for a QSPI unit.

Note: This only applies if ISRs are supported.

Format

```
t_qspi_ret psp_qspi_register_isr (
    uint8_t         uid,
    oal_isr_id_t    isr_id )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
isr_id	The ISR ID.	oal_isr_id_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_get_baudrate

Use this function to get the baud rate.

Format

```
t_qspi_ret psp_qspi_get_baudrate (
    uint8_t      uid,
    uint32_t *   p_br )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_br	The baud rate in Hz.	uint32_t *

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_set_baudrate

Use this function to set the baud rate.

Format

```
t_qspi_ret psp_qspi_set_baudrate (
    uint8_t    uid,
    uint32_t   p_br )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_br	The baud rate in Hz.	uint32_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_lock

Use this function to lock the QSPI for the specific unit.

This can be useful if multiple units are attached to the same QSPI bus.

Format

```
t_qspi_ret psp_qspi_lock (
    uint8_t   uid,
    uint8_t   from_isr )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
from_isr	The function is called from the interrupt flag.	uint8_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_unlock

Use this function to unlock the QSPI for the specific unit.

This can be useful if multiple units are attached to the same QSPI bus.

Format

```
t_qspi_ret psp_qspi_unlock (
    uint8_t   uid,
    uint8_t   from_isr )
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
from_isr	The function is called from the interrupt flag.	uint8_t

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_transfer

Use this function to transmit and receive a number of bytes.

Format

```
t_psp_qspi_ret psp_qspi_transfer (
    uint8_t             uid,
    t_qspi_transfer *   p_transfer)
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_transfer	A pointer to the transfer parameters.	t_qspi_transfer *

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution.
PSP_QSPI_ERROR	Operation failed.

psp_qspi_wait_status

Use this function to wait until the status register matches a value.

Format

```
t_psp_qspi_ret psp_qspi_transfer (
    uint8_t                 uid,
    t_qspi_wait_status *    p_wait_status)
```

Arguments

Parameter	Description	Type
uid	The unit ID.	uint8_t
p_wait_status	A pointer to the wait parameters.	t_qspi_wait_status *

Return Values

Return value	Description
PSP_QSPI_SUCCESS	Successful execution; the status register changed to the predefined value.
PSP_QSPI_ERROR	Operation failed.

t_qspi_transfer

The *t_qspi_transfer* structure is as follows:

Element	Type	Description
mode	uint8_t	The transfer mode, PSP_QSPI_TRANSFER_MODE_xxx.
cmd	uint8_t	The command to execute.
addr	uint32_t	The address to send.
addr_size	uint8_t	The number of address bytes, 3 or 4 bytes.
dummy_cycles	uint8_t	The number of dummy clocks between address and data bytes .
p_data	uint8_t*	A pointer to the data buffer.
data_size	uint32_t	The number of bytes in the data buffer.

t_qspi_wait_status

The content of the *t_qspi_wait_status* structure is as follows:

Element	Type	Description
mode	uint16_t	The transfer mode, PSP_QSPI_TRANSFER_MODE_xxx..
cmd	uint8_t	The command to send to get the register's value.
status_size	uint8_t	The number of bytes in the status register.
status_mask	uint32_t	The status register mask.
status_value	uint32_t	The masked status register compared to this value.
timeout_ms	uint32_t	The timeout in milliseconds.

Transfer Modes

The possible transfer modes are as folllows:

Mode	Value	Description
PSP_QSPI_TRANSFER_MODE_ADDR_1L	0x01	Send address on a single line.
PSP_QSPI_TRANSFER_MODE_ADDR_2L	0x02	Send address on two lines.
PSP_QSPI_TRANSFER_MODE_ADDR_4L	0x04	Send address on four lines.
PSP_QSPI_TRANSFER_MODE_DATA_WR	0x08	Send data.
PSP_QSPI_TRANSFER_MODE_DATA_RD	0x10	Receive data.
PSP_QSPI_TRANSFER_MODE_DATA_1L	0x20	Transfer data on a single line.
PSP_QSPI_TRANSFER_MODE_DATA_2L	0x40	Transfer data on two lines.
PSP_QSPI_TRANSFER_MODE_DATA_4L	0x80	Transfer data on four lines.

The address mask is defined as follows:

```
PSP_QSPI_TRANSFER_MODE_ADDR_MASK \
   ( (uint8_t)( PSP_QSPI_TRANSFER_MODE_ADDR_1L \
              | PSP_QSPI_TRANSFER_MODE_ADDR_2L \
              | PSP_QSPI_TRANSFER_MODE_ADDR_4L )
```

For sending data the mode is:

```
PSP_QSPI_TRANSFER_MODE_ADDR_1L | PSP_QSPI_TRANSFER_MODE_DATA_WR |
PSP_QSPI_TRANSFER_MODE_DATA_4L
```

For receiving data the mode is:

```
PSP_QSPI_TRANSFER_MODE_ADDR_1L | PSP_QSPI_TRANSFER_MODE_DATA_RD
| PSP_QSPI_TRANSFER_MODE_DATA_4
```

6. Version

Version 2.10

For use with SafeFLASH NOR Driver for CFI and SFDP SPI Flash versions 1.12 and above