Vector FBL SPI Driver

Technical Reference

Version 1.01.00

Authors	Johannes Krimmel
Status	Released

Document Information

History

Author	Date	Version	Remarks
Johannes Krimmel	2015-07-07	1.00.00	Initial version
Johannes Krimmel	2016-03-04	1.01.00	Added API description of SetTransferMode

Reference Documents

No.	Source	Title	Version

Caution
We have configured the programs in accordance with your specifications in the questionnaire. Whereas the programs do support other configurations than the one specified in your questionnaire, Vector´s release of the programs delivered to your company is expressly restricted to the configuration you have specified in the questionnaire.

Contents

1 Introduction 5
1.1 Scope of this document 5
1.2 Functionality of the SPI module 5
1.3 Architectural Overview 6

2 Integration 7
2.1 Files 7
2.2 Include Structure 8

3 Configuration 9
3.1 Configuration of the FBL SPI driver 9
3.2 Configuration of the SPI Interface Module FblDrvSpi_If 10

4 API Description 11
4.1 Type definitions 11
4.1.1 tFblSpi<Man><IF>ConfigParam 11
4.1.2 tFblSpiTransferParam 11
4.2 Services provided by the Vector FBL SPI driver 12
4.2.1 FblSpi<Man><IF>Init 12
4.2.2 FblSpi<Man><IF>Deinit 13
4.2.3 FblSpi<Man><IF>TransferSync 13
4.2.4 FblSpi<Man><IF>TransferAsync 14
4.2.5 FblSpi<Man><IF>GetTransferStatus 14
4.2.6 FblSpi<Man><IF>Cancel 15
4.2.7 FblSpi<Man><IF>ChangeConfiguration 15
4.2.8 FblSpi<Man><IF>SetTransferMode 15

5 Glossary and Abbreviations 17
5.1 Abbreviations 17

6 Contact 18

Illustrations

Figure 1-1 Architectural overview 6

Tables

Table 2-1 File location and description 7
Table 3-1 SPI driver configuration 9
Table 3-2 Configuration of SPI Interface Module 10
Table 4-1 Elements of tFblSpiTransferParam 12
Table 4-2 FblSpi<Man><IF>Init 12
Table 4-3 FblSpi<Man><IF>Deinit 13
Table 4-4 FblSpi<Man><IF>TransferSync 13
Table 4-5 FblSpi<Man><IF>TransferAsync 14
Table 4-6 FblSpi<Man><IF>GetTransferStatus 15
Table 4-7 FblSpi<Man><IF>Cancel 15
Table 4-8 FblSpi<Man><IF>ChangeConfiguration 15
Table 4-9 FblSpi<Man><IF>SetTransferMode 16

1 Introduction

1.1 Scope of this document

This document describes the functionality, configuration and API of the Vector FBL SPI module. The information in this document is solely about the hardware independent aspects of the Vector FBL SPI driver and not about the hardware specific settings of the individual implementations for a specific hardware platform.

Hardware dependent parts of file and function names have been replaced throughout this document in the following way:

- **<Man>** - The manufacturer of the microcontroller family the SPI driver is intended for or the microcontroller platform.
- **<IF>** - The Interface (regular spi, quad spi) or manufacturer‘s name for the SPI cell, for example spi, dspi or qspi.

Example
The file name fbl_spi_<Man>_<IF>.c of the documentation could stand for:
fbl_spi_freescale_qspi.c
fbl_spi_renesas_sci.c
fbl_spi_panasonic_spi.c

Note
For the hardware specific configuration options of the delivered SPI driver, please refer to the comments in the *_cfg.* files of your SPI driver.
Drivers with advanced features come with an additional Technical Reference which describes the hardware dependent configuration options. Check the /Doc folder of your delivery.

1.2 Functionality of the SPI module

The SPI driver provides services for basic communication with external components. It offers a hardware independent interface to the components of upper layers.

The main services of this module are:

> Initialization and deinitialization of the SPI hardware.

> Providing a synchronous and asynchronous interface for sending and receiving data over the SPI bus.

Note
The delivered SPI driver supports multiple configurations. It can be configured to use multiple SPI channels or communicate with multiple devices which require a different set of configuration parameters.

1.3 Architectural Overview

The following figure gives an overview of the architecture. The SPI driver is an abstraction layer between a device driver and the physical bus.

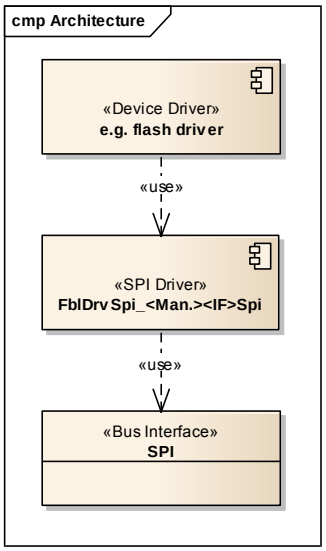

Figure 1-1 Architectural overview

2 Integration

This chapter describes the integration of the SPI driver.

2.1 Files

The Vector FBL SPI driver consists of the following files.

File	Description
fbl_spi_<Man>_<IF>.c	SPI driver implementation
fbl_spi_<Man>_<IF>.h	SPI driver interface
_fbl_spi_<Man>_<IF>_cfg.c	SPI driver configuration file template
_fbl_spi_<Man>_<IF>_cfg.h	SPI driver configuration file template
fbl_spi_<Man>_<IF>_inc.h	SPI driver include file

Table 2-1 File location and description

The following steps have to be executed to use the SPI driver in your project.

Practical Procedure

1. Add fbl_spi_<Man>_<IF>.c, fbl_spi_<Man>_<IF>.h and fbl_spi_<Man>_<IF>_inc.h to your project.
2. Remove the underscores from the configuration templates and add them to your project.
3. Adapt the configuration files. Refer to chapter 3 for step by step instructions.

2.2 Include Structure

In general, the Vector FBL SPI modules have the following dependencies:

- Generated files with hardware dependent information and general definitions of the Bootloader are included, for example v_cfg.h, v_def.h or fbl_def.h.
- fbl_spi_if_cfg.h and fbl_spi_if.h: These files contain definitions that are common to all SPI drivers in the project.
- To use the SPI module, the upper layer has to include the include file fbl_spi_<Man>_<IF>_inc.h of the SPI driver. Please adapt the configuration files of the module(s) that shall use the SPI driver accordingly.

3 Configuration

3.1 Configuration of the FBL SPI driver

This chapter describes the configuration of the SPI driver module. The following configuration options are available in fbl_spi_<Man>_<IF>_cfg.c and fbl_spi_<Man>_<IF>_cfg.h.

Option	Description
g_Spi<Man><IF>ConfigParam[]	Initialization structure for the used SPI configuration sets. An individual configuration set of type tFblSpi<Man><IF>ConfigParam should be configured for every channel/configuration used. It's recommended to use one configuration per SPI device. See the comments in the*_cfg.* source files and/or the hardware specific documentation of the SPI driver for a more detailed description.
FBL_SPI_NUMBER_OF_HANDLES	The number of the defined configuration sets for this driver. This value must be equal or less than FBL_SPI_MAX_NUMBER_OF_HANDLES which is defined in fbl_spi_if_cfg.h.
FBL_SPI_ENABLE_EXTERNAL_CS_HANDLING FBL_SPI_ENABLE_INTERNAL_CS_HANDLING	Define EXTERNAL_CS_HANDLING if the caller(s) handle(s) the chip select pin(s). Otherwise define INTERNAL_CS_HANDLING and configure the macros described below.
FBL_SPI_INIT_CS(spiHandle)	Defines a function/macro that initializes SPI pins that are associated with the value of spiHandle. This macro is used if internal chip select handling is activated by the integrator.
FBL_SPI_DEINIT_CS(spiHandle)	Defines a function/macro that deinitializes SPI pins that are associated with the value of spiHandle. This macro is used if internal chip select handling is activated.
FBL_SPI_SET_CS(spiHandle)	Defines a function/macro that sets CS pin of SPI device that is associated with the value of spiHandle. This macro is used if internal chip select handling is activated.
FBL_SPI_CLR_CS(spiHandle)	Defines a function/macro that deselects SPI device that is associated with the value of spiHandle . This macro is used if internal chip select handling is activated.
FBL_SPI_DUMMY_DATA	Defines a dummy value that is transferred when a transfer function of the driver is called with pTransmitBuffer set to NULL.

Table 3-1 SPI driver configuration

3.2 Configuration of the SPI Interface Module FblDrvSpi_If

This chapter describes the configuration of the FblDrvSpi_If module. This module is delivered in addition to the SPI driver module itself and contains definitions that are common to all SPI drivers in the project.

Following configuration options must be set in file fbl_spi_if_cfg.h:

Option	Description
FBL_SPI_MAX_NUMBER_OF_HANDLES	Set this value to the highest number of handles that is used in any SPI driver in the project. The number of handles for an individual driver is set with the define FBL_SPI_NUMBER_OF_HANDLES in file fbl_spi_<Man>_<IF>_cfg.h.

Table 3-2 Configuration of SPI Interface Module

4 API Description

4.1 Type definitions

The following table describes the structure definitions of `tFblSpi<Man><IF>ConfigParam` and `tFblSpiTransferParam`.

4.1.1 tFblSpi<Man><IF>ConfigParam

The `tFblSpi<Man><IF>ConfigParam` structure contains information needed for configuring an SPI channel. Usually settings like the baudrate and configuration options like clock phase and –polarity for the corresponding device are set here.

`g_Spi<Man><IF>ConfigParam[]` is an array of structs of type `tFblSpi<Man><IF>ConfigParam`. Each entry in the ConfigParam structure represents one set of configuration parameters. The array index to `g_Spi<Man><IF>ConfigParam[]` is used as “handle” for one individual configuation/device.

Since the settings contained in `tFblSpi<Man><IF>ConfigParam` are hardware dependent, please check the comments in the *_cfg.* source files and/or the hardware specific documentation of the SPI driver for a more detailed description.

4.1.2 tFblSpiTransferParam

Struct Element Name	C-Type	Description	Value Range
pTransmitBuffer	vuint8*	Pointer to data that shall be transmitted.	(0-0xFFFFFFFF) If set to NULL, FBL_SPI_DUMMY_DATA is sent.
pReceiveBuffer	vuint8*	Pointer to buffer for incoming data.	(0-0xFFFFFFFF) Can be set to NULL if incoming data is irrelevant.
transferLength	vuint16	Number of bytes to transmit/receive.	(0-0xFFFF)
chipSelectMode	vuint8 *	Specifies handling of CS for transfer.	> FBL_SPI_CHIP_SELECT_MODE_SET: CS is set before transfer. > FBL_SPI_CHIP_SELECT_MODE_CLR: CS is cleared after transfer. > FBL_SPI_CHIP_SELECT_MODE_BOTH: CS is set before and cleared after transfer. > FBL_SPI_CHIP_SELECT_MODE_NONE: State of CS will not be changed during transfer.

Struct Element Name	C-Type	Description	Value Range
pollingFct	tFblSpiPollingFct	Pointer to function that will be called during long lasting operations.	NULL or address of void(void) function.

Table 4-1 Elements of tFblSpiTransferParam

4.2 Services provided by the Vector FBL SPI driver

Note
The functions of the module that handle the initialization/deinitialization and the transfer of data over the SPI bus are not public. However preprocessor defines in fbl_spi_<Man>_<IF>.h wrap all the functions needed and form the API of the driver.

These defines must be used by the modules of the upper layer to call the API functions. In the function definitions below, the names of the defines are used instead of the underlying function prototypes.

Please also note that the prototypes of the API functions vary depending on the number of configured SPI handles (1 handle only: static configuration, more than 1 handle: dynamic configuration).

4.2.1 FblSpi<Man><IF>Init

Prototype	
Static configuration	`tFblResult FblSpi<Man><IF>Init ( void )`
Dynamic configuration	`tFblResult FblSpi<Man><IF>Init ( vuint8 spiHandle )`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`. Please refer to the hardware specific documentation of the SPI driver for more details about how to configure the init structure.
Return code	
`tFblResult`	Success of initialization: > `kFblSpiOk`: initialization was successful > `kFblSpiFailed`: initialization failed

Table 4-2 FblSpi<Man><IF>Init

4.2.2 FblSpi<Man><IF>Deinit

Prototype	
Static configuration	`tFblResult FblSpi<Man><IF>Deinit ( void )`
Dynamic configuration	`tFblResult FblSpi<Man><IF>Deinit ( vuint8 spiHandle )`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`. Please refer to the hardware specific documentation of the SPI driver for more details about how to configure the initialization structure.
Return code	
`tFblResult`	Success of deinitialization: > `kFblSpiOk`: deinitialization was successful > `kFblSpiFailed`: deinitialization failed

Table 4-3 FblSpi<Man><IF>Deinit

4.2.3 FblSpi<Man><IF>TransferSync

Prototype	
Static configuration	`tFblResult FblSpi<Man><IF>TransferSync ( tFblSpiTransferParam * transferParam )`
Dynamic configuration	`tFblResult FblSpi<Man><IF>TransferSync ( vuint8 spiHandle, tFblSpiTransferParam * transferParam )`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`.
`transferParam`	Pointer to the transfer parameter structure.
Return code	
`tFblResult`	Success of transmission: > `kFblOk`: transmission was successful > `kFblFailed`: transmission failed
Functional Description	
This is the Synchronous transfer function of the driver. It receives and transmits the number of bytes specified by the `tFblSpiTransferParam` structure. If the pointer to the transmit buffer or receive buffer is NULL, the incoming data is discarded/the value of `FBL_SPI_DUMMY_DATA` is sent. The function blocks until the transfer is completed and calls the specified polling function in `transferParam` cyclically.	

Table 4-4 FblSpi<Man><IF>TransferSync

4.2.4 FblSpi<Man><IF>TransferAsync

Prototype	
Static configuration	`tFblResult FblSpi<Man><IF>TransferAsync ( tFblSpiTransferParam * transferParam )`
Dynamic configuration	`tFblResult FblSpi<Man><IF>TransferAsync ( vuint8 spiHandle, tFblSpiTransferParam * transferParam )`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`.
`transferParam`	Pointer to the transfer parameter structure.
Return code	
`tFblResult`	Success of transmission: > `kFblOk`: transmission started successfully > `kFblFailed`: starting transmission failed
Functional Description	
This function starts an asynchronous transfer. The functionality is the same as a call to FblSpi<Man><IF>TransferSync, except it returns immediately and doesn't call the specified polling function in parameter `transferParam`.	

Table 4-5 FblSpi<Man><IF>TransferAsync

4.2.5 FblSpi<Man><IF>GetTransferStatus

Prototype	
Static configuration	`tFblSpiTransferStatus FblSpi<Man><IF>GetTransferStatus ( void )`
Dynamic configuration	`tFblSpiTransferStatus FblSpi<Man><IF>GetTransferStatus ( vuint8 spiHandle )`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`.
Return code	
`tFblSpiTransferStatus`	State of transmission: > `FBL_SPI_TRANSFER_STATUS_IDLE`: If state of transfer was idle before call, this function will do nothing and the transfer state will stay idle. Call FblSpi<Man><IF>TransferAsync to start the transfer! > `FBL_SPI_TRANSFER_STATUS_BUSY`: Transfer in progress > `FBL_SPI_TRANSFER_STATUS_COMPLETED`: Transfer successfully executed > `FBL_SPI_TRANSFER_STATUS_FAILED`: Transfer has failed.
Functional Description	
This function must be called repeatedly after an asynchronous transfer has been started until the status of the transfer is returned as FBL_SPI_TRANSFER_STATUS_COMPLETED (or `FBL_SPI_TRANSFER_STATUS_FAILED` in case an error has occurred).	

Table 4-6 FblSpi<Man><IF>GetTransferStatus

4.2.6 FblSpi<Man><IF>Cancel

Prototype	
Static configuration	`tFblResult FblSpi<Man><IF>Cancel ( void )`
Dynamic configuration	`tFblResult FblSpi<Man><IF>Cancel ( vuint8 spiHandle )`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`.
Return code	
`tFblResult`	Success of cancellation: > `kFblOk`: transmission cancelled successfully > `kFblFailed`: cancellation of transmission failed
Functional Description	
This function cancels any ongoing transfer associated with the specified handle and re-initializes the SPI interface for that handle.	

Table 4-7 FblSpi<Man><IF>Cancel

4.2.7 FblSpi<Man><IF>ChangeConfiguration

Prototype	
Dynamic configuration	`tFblResult FblSpi<Man><IF>ChangeConfiguration ( vuint8 spiHandle, vuint8 newSpiHandle )`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`. This configuration will be deinitialized.
`newSpiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`. This configuration will be initialized.
Return code	
`tFblResult`	Success of configuration change: > `kFblOk`: change successfully > `kFblFailed`: change failed
Functional Description	
This function deinitializes the configuration specified with `spiHandle` and initializes the configuration specified with `newSpiHandle`, which essentially changes the configuration.	

Table 4-8 FblSpi<Man><IF>ChangeConfiguration

4.2.8 FblSpi<Man><IF>SetTransferMode

Prototype	
Static configuration	`tFblResult FblSpi<Man><IF>SetTransferMode (tFblSpiTransferMode transferMode )`

Dynamic configuration	`tFblResult FblSpi<Man><IF>SetTransferMode ( vuint8 spiHandle, tFblSpiTransferMode transferMode)`
Parameter	
`spiHandle`	Handle to an entry of the initialization structure `g_Spi<Man><IF>ConfigParam`.
`transferMode`	The transfer mode that shall be used.
Return code	
`tFblResult`	Success of configuration change: > `kFblOk`: mode set successfully > `kFblFailed`: mode switch failed or mode not supported
Functional Description	
Call to change the transfer mode of the driver. Typical SPI slave devices use single mode (full-duplex), but there are some memory devices that use dual or quad mode to improve performance (half duplex, while multiple pins are being used for transfers in the respective direction).	

Table 4-9 FblSpi<Man><IF>SetTransferMode

5 Glossary and Abbreviations

5.1 Abbreviations

Abbreviation	Description
SPI	Serial Peripheral Interface

6 Contact

Visit our website for more information on

- News
- Products
- Demo software
- Support
- Training data
- Addresses

www.vector.com