



DRIVER AND LIBRARY GUIDE

SPI Flash Extension

Prepared for:

Sharif University of Technology Branch – Optic Center

Prepared by:

Majid Derhambakhsh

September 23, 2021



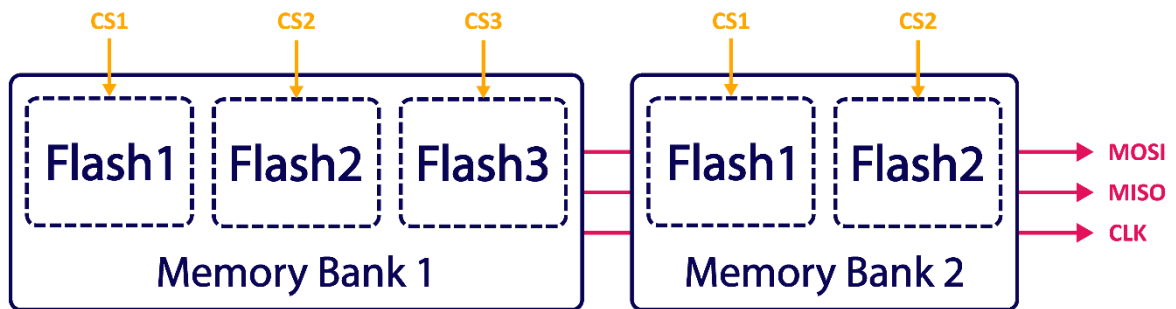
Content

1. What does this library do?	3
2. Guide	3
2.1 SPI Flash Extension library structures	4
2.1.1 SPI_Flash_ListTypeDef	4
2.2 SPI Flash Extension library API description	4
2.2.1 How to use this library	4
2.2.2 Initialization and de-initialization functions	5
2.2.3 Operation functions	5
2.2.4 Detailed description of functions	5
2.3 SPI Flash Extension library Configuration	10
3. Examples	10
4. Requirements	12
5. Important tips	12
6. Error and Warning's	12
7. Supported memories	13
8. Tests performed	13
9. License	13



1 What does this library do?

This library merges a collection of flash memories as a memory bank:



Notes

- All the flash memories use the common MOSI, MISO, and CLK pin.
- Each flash memory uses a separately cs pin.
- The control of each collection is managed by the library.
- Each memory bank capacity consists of the sum of the flash memory capacity of that set



2.1 SPI Flash Extension library structures

2.1.1 SPI_Flash_ListTypeDef

Data Fields

- `SPI_Flash_TypeDef *FlashList`
- `uint32_t TotalNumberOfPages`
- `uint32_t TotalNumberOfSectors`
- `uint32_t TotalNumberOfBlocks`
- `uint32_t TotalCapacity`
- `uint8_t NumberOfChip`
- `uint32_t LastWriteAddress`

Field Documentation

- ***`SPI_Flash_TypeDef SPI_Flash_ListTypeDef::FlashList`***
The list of flash for merge
- ***`uint32_t SPI_Flash_ListTypeDef::TotalNumberOfPages`***
The sum of merged flash pages
- ***`uint32_t SPI_Flash_ListTypeDef::TotalNumberOfSectors`***
The sum of merged flash sectors
- ***`uint32_t SPI_Flash_ListTypeDef::TotalNumberOfBlocks`***
The sum of merged flash blocks
- ***`uint32_t SPI_Flash_ListTypeDef::TotalCapacity`***
The total capacity of merged flash's (KB)
- ***`uint8_t SPI_Flash_ListTypeDef::NumberOfChip`***
The number of merged flash chips
- ***`uint32_t SPI_Flash_ListTypeDef::LastWriteAddress`***
The Last address written (B)
This value is equal to: last address + 1

2.2 SPI Flash Extension library API description

2.2.1 How to use this library

This library can be used as follows:

1. Add SPI Flash library as a base library and config it
2. Add SPI Flash Extension library Header and Source file in your project
3. Config the library in "*[spi_flash_ex_conf.h](#)*"
4. Create array of flash memory object with *[SPI_Flash_TypeDef](#)* type and set specific GPIO
5. Create flash memory list object with *[SPI_Flash_ListTypeDef](#)* type and set FlashList, NumberOfChip



6. Initialize memory bank with *SPI_FlashEx_Init*
7. Use flash memory operation functions

2.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize and configure the flash memory

This section contains the following APIs:

- *SPI_FlashEx_Init()*

2.2.3 Operation functions

This section contains the following APIs:

- *SPI_FlashEx_ChipErase()*
- *SPI_FlashEx_SectorErase()*
- *SPI_FlashEx_BlockErase()*
- *SPI_FlashEx_Write()*
- *SPI_FlashEx_PageWrite()*
- *SPI_FlashEx_SectorWrite()*
- *SPI_FlashEx_BlockWrite()*
- *SPI_FlashEx_BurstWrite()*
- *SPI_FlashEx_Read()*
- *SPI_FlashEx_PageRead()*
- *SPI_FlashEx_SectorRead()*
- *SPI_FlashEx_BlockRead()*
- *SPI_FlashEx_BurstRead()*

2.2.4 Detailed description of functions

SPI_FlashEx_Init

Function name **uint8_t SPI_FlashEx_Init (SPI_Flash_ListTypeDef *_flash_list)**

Function description This function is used to initialize memory bank

Parameters

- **_flash_list:** pointer to flash list struct

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_ChipErase



Function name **uint8_t SPI_FlashEx_ChipErase (SPI_Flash_ListTypeDef *_flash_list)**

Function description This function is used to erase memory bank

Parameters

- **_flash_list:** pointer to flash list struct

Return values

- **SPI:** status of SPI peripheral

Notes

- This function clears the memory bank completely

SPI_FlashEx_SectorErase

Function name **uint8_t SPI_FlashEx_SectorErase (SPI_Flash_ListTypeDef *_flash_list, uint32_t _sector)**

Function description This function is used to erase special sector of memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_sector:** sector number to erase

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_BlockErase

Function name **uint8_t SPI_FlashEx_BlockErase (SPI_Flash_ListTypeDef *_flash_list, uint32_t _block)**

Function description This function is used to erase special block of memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_block:** block number to erase

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_Write

Function name **uint8_t SPI_FlashEx_Write (SPI_Flash_ListTypeDef *_flash_list, uint8_t _data, uint32_t _address)**

Function description This function is used to write a byte of data in memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_data:** data to write
- **_address:** address to write data

Return values



- **SPI:** status of SPI peripheral

Notes

- This function writes a single byte in memories, you can use other functions to write many bytes in memories

SPI_FlashEx_PageWrite

Function name `uint8_t SPI_FlashEx_PageWrite (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _page, uint32_t _offset, uint32_t _size)`

Function description This function is used to write many bytes of data in a special page of memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_pData:** pointer to data to write
- **_page:** page number to write data
- **_offset:** offset of data in page (Range: 0B ~ 255B)
- **_size:** size of data to write (Range: 1B ~ 256B)

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_SectorWrite

Function name `uint8_t SPI_FlashEx_SectorWrite (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _sector, uint32_t _offset, uint32_t _size)`

Function description This function is used to write many bytes of data in a special sector of memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_pData:** pointer to data to write
- **_sector:** sector number to write data
- **_offset:** offset of data in sector (Range: 0B ~ 4095B)
- **_size:** size of data to write (Range: 1B ~ 4096B)

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_BlockWrite

Function name `uint8_t SPI_FlashEx_BlockWrite (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _block, uint32_t _offset, uint32_t _size)`

Function description This function is used to write many bytes of data in a special block of memory bank

Parameters

- **_flash_list:** pointer to flash list struct



- **_pData:** pointer to data to write
- **_block:** block number to write data
- **_offset:** offset of data in block (Range: 0B ~ 65535B)
- **_size:** size of data to write (Range: 1B ~ 65536B)

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_BurstWrite

Function name

uint8_t SPI_FlashEx_BurstWrite (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _address, uint32_t _size)

Function description

This function is used to write many bytes of data in memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_pData:** pointer to data to write
- **_address:** start of address to write data
- **_size:** size of data to write (Range: 1B ~ Memory Bank Capacity)

Return values

- **SPI:** status of SPI peripheral

Notes

- This function has don't any limitations for size of data
- Maximum address to write is: Memory Bank Capacity - 1B

SPI_FlashEx_Read

Function name

uint8_t SPI_FlashEx_Read (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_data, uint32_t _address)

Function description

This function is used to read a byte of data from memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_data:** pointer to data to read
- **_address:** address to read data

Return values

- **SPI:** status of SPI peripheral

Notes

- This function read a single byte from memories; you can use other functions to read many bytes from memories

SPI_FlashEx_PageRead



Function name **uint8_t SPI_FlashEx_PageRead (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _page, uint32_t _offset, uint32_t _size)**

Function description This function is used to read many bytes of data from a special page of memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_pData:** pointer to data to read
- **_page:** page number to read data
- **_offset:** offset of data in page (Range: 0B ~ 255B)
- **_size:** size of data to read (Range: 1B ~ 256B)

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_SectorRead

Function name **uint8_t SPI_FlashEx_SectorRead (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _sector, uint32_t _offset, uint32_t _size)**

Function description This function is used to read many bytes of data from a special sector of memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_pData:** pointer to data to read
- **_sector:** sector number to read data
- **_offset:** offset of data in sector (Range: 0B ~ 4095B)
- **_size:** size of data to read (Range: 1B ~ 4096B)

Return values

- **SPI:** status of SPI peripheral

SPI_FlashEx_BlockRead

Function name **uint8_t SPI_FlashEx_BlockRead (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _block, uint32_t _offset, uint32_t _size)**

Function description This function is used to read many bytes of data from a special block of memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_pData:** pointer to data to read
- **_block:** block number to read data
- **_offset:** offset of data in block (Range: 0B ~ 65535B)
- **_size:** size of data to read (Range: 1B ~ 65536B)

Return values

- **SPI:** status of SPI peripheral



SPI_FlashEx_BurstRead

Function name `uint8_t SPI_FlashEx_BurstRead (SPI_Flash_ListTypeDef *_flash_list, uint8_t *_pData, uint32_t _address, uint32_t _size)`

Function description This function is used to read many bytes of data from memory bank

Parameters

- **_flash_list:** pointer to flash list struct
- **_pData:** pointer to data to read
- **_address:** start of address to read data
- **_size:** size of data to read (Range: 1B ~ Memory Bank Capacity)

Return values

- **SPI:** status of SPI peripheral

Notes

- This function has don't any limitations for size of data
- Maximum address to read is: Memory Bank Capacity - 1B

2.3 SPI Flash Extension Library Configuration

Open "[spi_flash_ex_conf.h](#)" to configure library

- Set the 'last write address' ability in the configuration section, for example:

+ in this section, you can define the "`_SPI_FLASH_EX_USE_LAST_WRITE_ADD`" to use this ability

```
/* ----- Configuration ----- */
#define _SPI_FLASH_EX_USE_LAST_WRITE_ADD
```

+ Comment this define to disable the ability

3. Examples

- Example 1: Initialize and use external flash memories as a memory bank with LPC1768

```
#include "lpc17xx.h"
#include "lpc17xx_gpio.h"
#include "lpc17xx_spi.h"
#include "lpc17xx_libcfg.h"
#include "lpc17xx_pinsel.h"

#include "lpc_spi_ex.h"
#include "spi_flash_ex.h"

SPI_CFG_Type SPI_ConfigStruct;

uint8_t flashData[4096];
uint8_t sampleData[320];
uint8_t textData[44] = "Hello from master!, this is a test program!\n";
uint8_t Rx_Buf[50];
```



```

int main()
{
    /* ----- Setup GPIO ----- */
    PINSEL_CFG_Type PinCfg;

    /*
     * Initialize SPI pin connect
     * P0.15 - SCK;
     * P0.0 / P0.1 - SSEL - used as GPIO
     * P0.17 - MISO
     * P0.18 - MOSI
     */

    PinCfg.Funcnum    = 3;
    PinCfg.OpenDrain  = 0;
    PinCfg.Pinmode    = 0;
    PinCfg.Portnum    = 0;
    PinCfg.Pinnum     = 15;
    PINSEL_ConfigPin(&PinCfg);

    PinCfg.Pinnum     = 17;
    PINSEL_ConfigPin(&PinCfg);

    PinCfg.Pinnum     = 18;
    PINSEL_ConfigPin(&PinCfg);

    /* Set GPIO Direction */
    GPIO_SetDir(0, (1 << 16), 1);
    GPIO_SetDir(0, (1 << 19), 1);
    GPIO_SetDir(0, (1 << 7), 1);

    GPIO_SetValue(0, (1 << 16));
    GPIO_SetValue(0, (1 << 19));
    GPIO_SetValue(0, (1 << 7));

    /* ----- Setup SPI ----- */
    SPI_ConfigStruct.CPHA      = SPI_CPHA_FIRST;
    SPI_ConfigStruct.CPOL      = SPI_CPOL_HI;
    SPI_ConfigStruct.ClockRate = 300000;
    SPI_ConfigStruct.DataOrder = SPI_DATA_MSB_FIRST;
    SPI_ConfigStruct.Databit   = SPI_DATABIT_8;
    SPI_ConfigStruct.Mode      = SPI_MASTER_MODE;

    SPI_Init(LPC_SPI, &SPI_ConfigStruct);

    /* ----- Wait to init ----- */
    SPI_Delay(1);

    /* ~~~~~~ Flash Bank Example ~~~~~~ */
    SPI_Flash_TypeDef      MainFlashList[3];
    SPI_Flash_ListTypeDef  FlashBank;

    MainFlashList[0].CS_GPIO_Port = 0;
    MainFlashList[0].CS_GPIO_Pin  = (1 << 16);

    MainFlashList[1].CS_GPIO_Port = 0;
    MainFlashList[1].CS_GPIO_Pin  = (1 << 19);

    MainFlashList[2].CS_GPIO_Port = 0;
    MainFlashList[2].CS_GPIO_Pin  = (1 << 7);

    FlashBank.FlashList  = MainFlashList;
    FlashBank.NumberOfChip = 3;

    /* ----- Commands ----- */
    SPI_FlashEx_Init(&FlashBank);
    SPI_FlashEx_ChipErase(&FlashBank);

```



```

SPI_FlashEx_BurstWrite(&FlashBank, flashData, FlashBank.LastWriteAddress, 10);
SPI_FlashEx_BurstWrite(&FlashBank, sampleData, 50000, 320);
SPI_FlashEx_BurstWrite(&FlashBank, textData, 8388586, 44);
SPI_FlashEx_BurstWrite(&FlashBank, flashData, 12580864, 4096);

SPI_FlashEx_PageWrite(&FlashBank, textData, 150, 0, 44);

SPI_FlashEx_BurstRead(&FlashBank, Rx_Buf, 8388586, 44);

while(1)
{

}
/* Loop forever */
}

```

4. Requirements

1. CMSIS and SPL driver in LPCxx series
2. lpc_gpio_ex driver in LPCxx series
3. lpc_spi_ex driver in LPCxx series
4. SPI Flash library as a base library

5. Important tips

1. This library use the SPI Flash library as a base library
2. All defines beginning with (underline)
3. All functions are written as CamelCase

6. Error and Warning's

- **Error's**

- **[FLASH ERROR01]Controller is not selected Or not supported:** This error occurs when the MCU or its library not supported.

- **Warning's**

- None

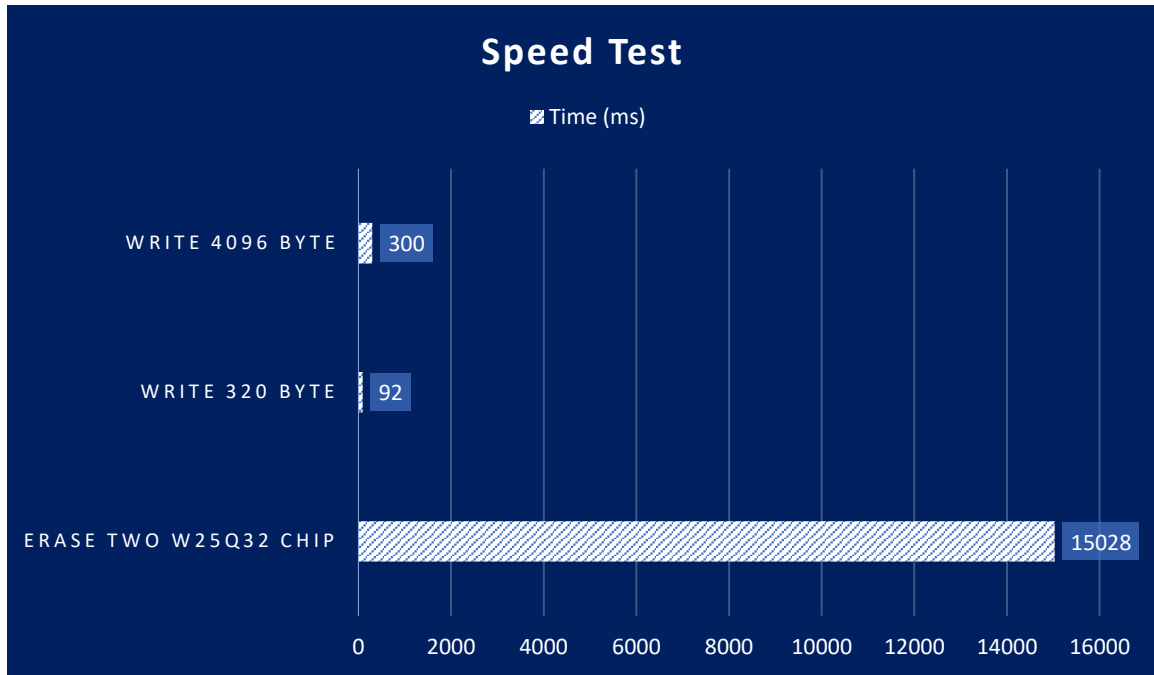


7. Supported memories

- All supported memories by SPI Flash library

8. Tests performed

- Speed test in two W25Q32



Test Parameter :

- **Core:** STM32F469NI
- **Core Speed:** 180MHz
- **SPI Speed:** 45Mbits

9. License



Majid-Derhambakhsh/SPI-Flash is licensed under the
Apache License 2.0

A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Permissions

- ✓ Commercial use
- ✓ Modification
- ✓ Distribution
- ✓ Patent use
- ✓ Private use

Limitations

- ✗ Trademark use
- ✗ Liability
- ✗ Warranty

Conditions

- ℹ License and copyright notice
- ℹ State changes

This is not legal advice. [Learn more about repository licenses.](#)



