DRIVER AND LIBRARY GUIDE

SPI Flash

Prepared for:

Sharif University of Technology Branch – Optic Center

Prepared by:

Majid Derhambakhsh

September 23, 2021

Content

1.1 SPI Flash library structures

1.1.1 SPI_Flash_TypeDef

Data Fields

- **GPIO_Type CS_GPIO_Port**
- **uint32_t CS_GPIO_Pin**
- **uint32_t JedecID**
- **SPI_Flash_IDTypeDef ID**
- **uint8_t *UniqID**
- **uint16_t PageSize**
- **uint32_t NumberOfPage**
- **uint32_t SectorSize**
- **uint32_t NumberOfSector**
- **uint32_t BlockSize**
- **uint32_t NumberOfBlocks**
- **uint32_t Capacity**
- **uint8_t StatusRegister1**
- **uint8_t StatusRegister2**
- **uint8_t StatusRegister3**
- **uint8_t WaitToComplete**

Field Documentation

- ***GPIO_Type SPI_Flash_TypeDef:: CS_GPIO_Port***

  Specifies the GPIO Port of CS pin
- ***uint32_t SPI_Flash_TypeDef:: CS_GPIO_Pin***

  Specifies the GPIO Pin of CS pin
- ***uint32_t SPI_Flash_TypeDef:: JedecID***

  The JedecID of flash memory
- ***SPI_Flash_IDTypeDef SPI_Flash_TypeDef::ID***

  The specific device ID of flash memory
- ***uint8_t SPI_Flash_TypeDef::*UniqID***

  The specific UniqID of flash memory
- ***uint16_t SPI_Flash_TypeDef::PageSize***

  The page size of flash memory (B)
- ***uint32_t SPI_Flash_TypeDef::NumberOfPage***

  The number of flash page
- ***uint32_t SPI_Flash_TypeDef::SectorSize***

  The sector size of Flash

- ***uint32_t SPI_Flash_TypeDef::NumberOfSector***

  The number of flash sector
- ***uint32_t SPI_Flash_TypeDef::BlockSize***

  The block size of Flash
- ***uint32_t SPI_Flash_TypeDef::NumberOfBlocks***

  The number of flash block
- ***uint32_t SPI_Flash_TypeDef::Capacity***

  The capacity of flash (KB)
- ***uint8_t SPI_Flash_TypeDef::StatusRegister1***

  Specifies the register value of flash
- ***uint8_t SPI_Flash_TypeDef::StatusRegister2***

  Specifies the register value of flash
- ***uint8_t SPI_Flash_TypeDef::StatusRegister3***

  Specifies the register value of flash
- ***uint8_t SPI_Flash_TypeDef::WaitToComplete***

  The write status of flash for use in ISR and RTOS

1.2 SPI Flash library API description

1.2.1 How to use this library

This library can be used as follows:

1. Config MCU SPI and initialize it
2. Add library Header and Source file in your project
3. Config the flash memory in ***"spi_flash_conf.h"***
4. Create flash memory object with ***SPI_Flash_TypeDef*** type and set specific GPIO
5. Initialize flash memory with ***SPI_Flash_Init***
6. Use flash memory operation functions

1.2.2 Initialization and de-initialization functions

This section provides functions allowing to:

- Initialize and configure the flash memory

This section contains the following APIs:

- *SPI_Flash_Init()*

1.2.3 Operation functions

This section contains the following APIs:

- *SPI_Flash_GetID()*

- *SPI_Flash_GetUniqID()*
- *SPI_Flash_WriteEnable()*
- *SPI_Flash_WriteDisable()*
- *SPI_Flash_ReadStatusRegister()*
- *SPI_Flash_WriteStatusRegister()*
- *SPI_Flash_WaitForWriteEnd()*
- *SPI_Flash_ChipErase()*
- *SPI_Flash_SectorErase()*
- *SPI_Flash_BlockErase()*
- *SPI_Flash_Write()*
- *SPI_Flash_PageWrite()*
- *SPI_Flash_SectorWrite()*
- *SPI_Flash_BlockWrite()*
- *SPI_Flash_BurstWrite()*
- *SPI_Flash_Read()*
- *SPI_Flash_PageRead()*
- *SPI_Flash_SectorRead()*
- *SPI_Flash_BlockRead()*
- *SPI_Flash_BurstRead()*

1.2.4 Detailed description of functions

SPI_Flash_Init

Function name **uint8_t SPI_Flash_Init (SPI_Flash_TypeDef *_flash)**

Function description This function is used to initialize flash memory

Parameters

- **_flash:** pointer to flash struct

Return values

- **SPI:** status of SPI peripheral

SPI_Flash_GetID

Function name **uint8_t SPI_Flash_GetID (SPI_Flash_TypeDef *_flash)**

Function description This function is used to get flash memory ID

Parameters

- **_flash:** pointer to flash struct

Return values

- **SPI:** status of SPI peripheral

Notes

- The flash ID is stored in the flash struct

SPI_Flash_GetUinqID

Function name **uint8_t SPI_Flash_GetUniqID (SPI_Flash_TypeDef *_flash)**

Function description This function is used to get Unique ID of flash memory

Parameters

- **_flash:** pointer to flash struct

Return values

- **SPI:** status of SPI peripheral

Notes

- The flash Unique ID is stored in the flash struct

SPI_Flash_WriteEnable

Function name **uint8_t SPI_Flash_WriteEnable (SPI_Flash_TypeDef *_flash)**

Function description This function is used to enable write mode of flash memory

Parameters

- **_flash:** pointer to flash struct

Return values

- **SPI:** status of SPI peripheral

SPI_Flash_WriteDisable

Function name **uint8_t SPI_Flash_WriteDisable (SPI_Flash_TypeDef *_flash)**

Function description This function is used to disable write mode of flash memory

Parameters

- **_flash:** pointer to flash struct

Return values

- **SPI:** status of SPI peripheral

SPI_Flash_ReadStatusRegister

Function name **uint8_t SPI_Flash_ReadStatusRegister (SPI_Flash_TypeDef *_flash, SPI_Flash_RegTypeDef _register)**

Function description This function is used to read status registers of flash memory

Parameters

- **_flash:** pointer to flash struct
- **_register:** register number of flash memory

Return values

- **SPI:** status of SPI peripheral

Notes

- The registers value is stored in the flash struct

SPI_Flash_WriteStatusRegister

Function name	**uint8_t SPI_Flash_WriteStatusRegister (SPI_Flash_TypeDef *_flash, SPI_Flash_RegTypeDef _register)**
Function description	This function is used to write status registers of flash memory

Parameters

- **_flash:** pointer to flash struct
- **_register:** register number of flash memory

Return values

- **SPI:** status of SPI peripheral

Notes

- The registers value is stored in the flash struct and flash memory

SPI_Flash_WaitForWriteEnd

Function name	**uint8_t SPI_Flash_WaitForWriteEnd (SPI_Flash_TypeDef *_flash)**
Function description	This function is used to check write status and wait for end of it

Parameters

- **_flash:** pointer to flash struct

Return values

- **SPI:** status of SPI peripheral

SPI_Flash_ChipErase

Function name	**uint8_t SPI_Flash_ChipErase (SPI_Flash_TypeDef *_flash)**
Function description	This function is used to erase flash memory

Parameters

- **_flash:** pointer to flash struct

Return values

- **SPI:** status of SPI peripheral

Notes

- This function clears the memory completely

SPI_Flash_SectorErase

Function name **uint8_t SPI_Flash_SectorErase (SPI_Flash_TypeDef *_flash, uint32_t _sector)**

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

Parameters

- **_flash:** pointer to flash struct
- **_sector:** sector number to erase

Return values

- **SPI:** status of SPI peripheral

SPI_Flash_BlockErase

Function name **uint8_t SPI_Flash_BlockErase (SPI_Flash_TypeDef *_flash, uint32_t _block)**

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

Parameters

- **_flash:** pointer to flash struct
- **_block:** block number to erase

Return values

- **SPI:** status of SPI peripheral

SPI_Flash_Write

Function name **uint8_t SPI_Flash_Write (SPI_Flash_TypeDef *_flash, uint8_t _data, uint32_t _address)**

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

Parameters

- **_flash:** pointer to flash 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 memory, you can use other functions to write many bytes in memory

SPI_Flash_PageWrite

Function name **uint8_t SPI_Flash_PageWrite (SPI_Flash_TypeDef *_flash, 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 flash memory

Parameters

- **_flash:** pointer to flash 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_Flash_SectorWrite

Function name **uint8_t SPI_Flash_SectorWrite (SPI_Flash_TypeDef *_flash, 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 flash memory

Parameters

- **_flash:** pointer to flash 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_Flash_BlockWrite

Function name **uint8_t SPI_Flash_BlockWrite (SPI_Flash_TypeDef *_flash, 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 flash memory

Parameters

- **_flash:** pointer to flash 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_Flash_BurstWrite

Function name **uint8_t SPI_Flash_BurstWrite (SPI_Flash_TypeDef *_flash, uint8_t *_pData, uint32_t _address, uint32_t _size)**

Function description This function is used to write many bytes of data in flash memory

Parameters

- **_flash:** pointer to flash struct
- **_pData:** pointer to data to write
- **_address:** start of address to write data
- **_size:** size of data to write (Range: 1B ~ Flash 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: Flash Capacity - 1B

SPI_Flash_Read

Function name **uint8_t SPI_Flash_Read (SPI_Flash_TypeDef *_flash, uint8_t *_data, uint32_t _address)**

Function description This function is used to read a byte of data from flash memory

Parameters

- **_flash:** pointer to flash 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 memory; you can use other functions to read many bytes from memory

SPI_Flash_PageRead

Function name **uint8_t SPI_Flash_PageRead (SPI_Flash_TypeDef *_flash, 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 flash memory

Parameters

- **_flash:** pointer to flash 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_Flash_SectorRead

Function name **uint8_t SPI_Flash_SectorRead (SPI_Flash_TypeDef *_flash, 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 flash memory

Parameters

- **_flash:** pointer to flash 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_Flash_BlockRead

Function name **uint8_t SPI_Flash_BlockRead (SPI_Flash_TypeDef *_flash, 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 flash memory

Parameters

- **_flash:** pointer to flash 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_Flash_BurstRead

Function name **uint8_t SPI_Flash_BurstRead (SPI_Flash_TypeDef *_flash, uint8_t *_pData, uint32_t _address, uint32_t _size)**

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

Parameters

- **_flash:** pointer to flash struct

- **_pData:** pointer to data to read
- **_address:** start of address to read data
- **_size:** size of data to read (Range: 1B ~ Flash 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: Flash Capacity - 1B

1.3 SPI Flash library defines

1.3.1 SPI Flash

SPI Flash Page Size

_SPI_FLASH_PAGE_SIZE

SPI Flash Sector Size

_SPI_FLASH_SECTOR_SIZE

SPI Flash Sector Per Block

_SPI_FLASH_SECTOR_PER_BLOCK

SPI Flash Status Registers

_SPI_FLASH_STATUS_REGISTER_1

_SPI_FLASH_STATUS_REGISTER_2

_SPI_FLASH_STATUS_REGISTER_3

SPI Flash ID

_SPI_FLASH_25Q10

_SPI_FLASH_25Q20

_SPI_FLASH_25Q40

_SPI_FLASH_25Q80

_SPI_FLASH_25Q16

_SPI_FLASH_25Q32

_SPI_FLASH_25Q64

_SPI_FLASH_25Q128

_SPI_FLASH_25Q256

_SPI_FLASH_25Q512

_SPI_FLASH_25Q01

SPI Flash Exported Macros

__CONVERT_BYTE_TO_KBYTE

__CONVERT_KBYTE_TO_BYTE

__CONVERT_PAGE_TO_SECTOR

__CONVERT_PAGE_TO_BLOCK

__CONVERT_SECTOR_TO_PAGE

__CONVERT_SECTOR_TO_BLOCK

__CONVERT_BLOCK_TO_PAGE

__CONVERT_BLOCK_TO_SECTOR

__CONVERT_ADDRESS_TO_PAGE

1.4 SPI Flash Library Configuration

Open "spi_flash_conf.h" to configure library

- Set MCU in controller series section, for example:

  + in this section, you should define the microcontroller of your project

```
/* ---------- Controller Series --------- */
#define _MCU_SERIES
```

  + _MCU_SERIES: microcontroller series

    • _LPC17XX for LPC17 series

- Add requirement driver / libraries, for example:

  + in this section, you should include the required libraries of the spi_flash library

```
/* ------ Required Driver.Library ------- */
#include "spi_library.h"
#include "gpio_library.h"
```

  + lpc_spi_ex.h and lpc_gpio_ex.h for LPC17 series

- Set MUC SPI in configuration section, for example:

  + in this section, you should set the SPI peripheral and its 'time out'

```
/* ............ SPI ............ */
#define _SPI_FLASH_PERIPHERAL SPIx
#define _SPI_FLASH_TIMEOUT    Time
```

  + SPIx: the SPI struct or its identifier

    - LPC_SPI for LPC17 series

  + Time: the SPI time out (Range: 1 ~ x)

- Set RTOS mode in configuration section, for example:

  + in this section, you should set the ROTS mode

```
/* ............ RTOS ........... */
#define _SPI_FLASH_USE_RTOS    Mode
```

  + Mode: RTOS mode

    - 0: Disabled

    - 1: Enabled

- Set flash manufacturer, for example:

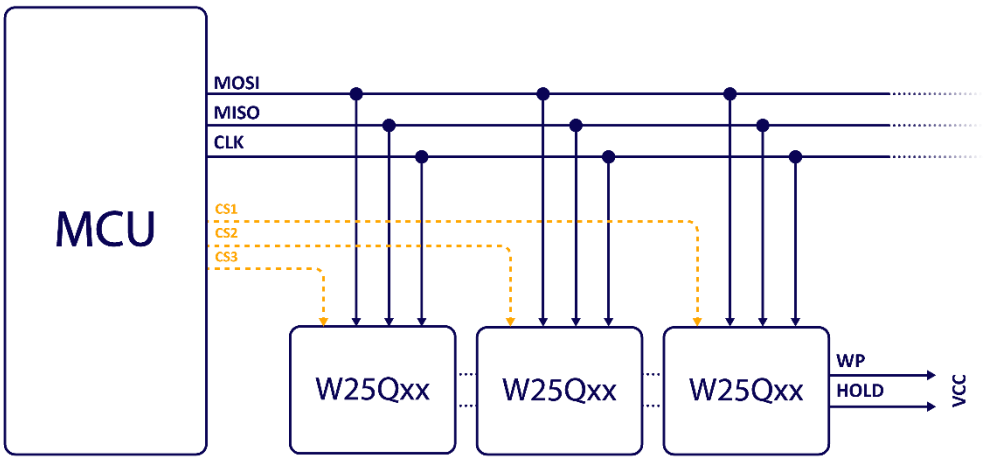
  + in this section, you should define the microcontroller of your project

```
/* ..... Flash manufacturer .... */
#define _SPI_FLASH_x
```

  + _SPI_FLASH_x: flash manufacturer

    - _SPI_FLASH_WINBOND for Winbond company

2. Circuit Design



3. Examples

- Example 1: Initialize and use external flash memory 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.h"

SPI_CFG_Type SPI_ConfigStruct;

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);

	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 Example ~~~~~~~~~~~~~~~~~~ */
    /* --------- Setup Flash ---------- */
    SPI_Flash_TypeDef     Flash1;
    SPI_Flash_TypeDef     Flash2;
    SPI_Flash_TypeDef     Flash3;

    Flash1.CS_GPIO_Port = 0;
    Flash1.CS_GPIO_Pin  = (1 << 16);

    Flash2.CS_GPIO_Port = 0;
    Flash2.CS_GPIO_Pin  = (1 << 19);

    Flash3.CS_GPIO_Port = 0;
    Flash3.CS_GPIO_Pin  = (1 << 7);

    /* ----------- Commands ----------- */
    SPI_Flash_Init(&Flash1);
    SPI_Flash_Init(&Flash2);
    SPI_Flash_Init(&Flash3);

    SPI_Flash_ChipErase(&Flash1);
    SPI_Flash_ChipErase(&Flash2);
    SPI_Flash_ChipErase(&Flash3);

    SPI_Flash_SectorWrite(&Flash2, textData, 0, 10, 44);
    SPI_Flash_SectorRead(&Flash2, Rx_Buf, 0, 10, 44);

    while(1)
    {

    }
    /* Loop forever */

}
```

4. Requirements

1. CMSIS driver in LPCxx series
2. lpc_gpio_ex driver in LPCxx series
3. lpc_spi_ex driver in LPCxx series

5. Important tips

1. All defines beginning with _ (underline)
2. 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

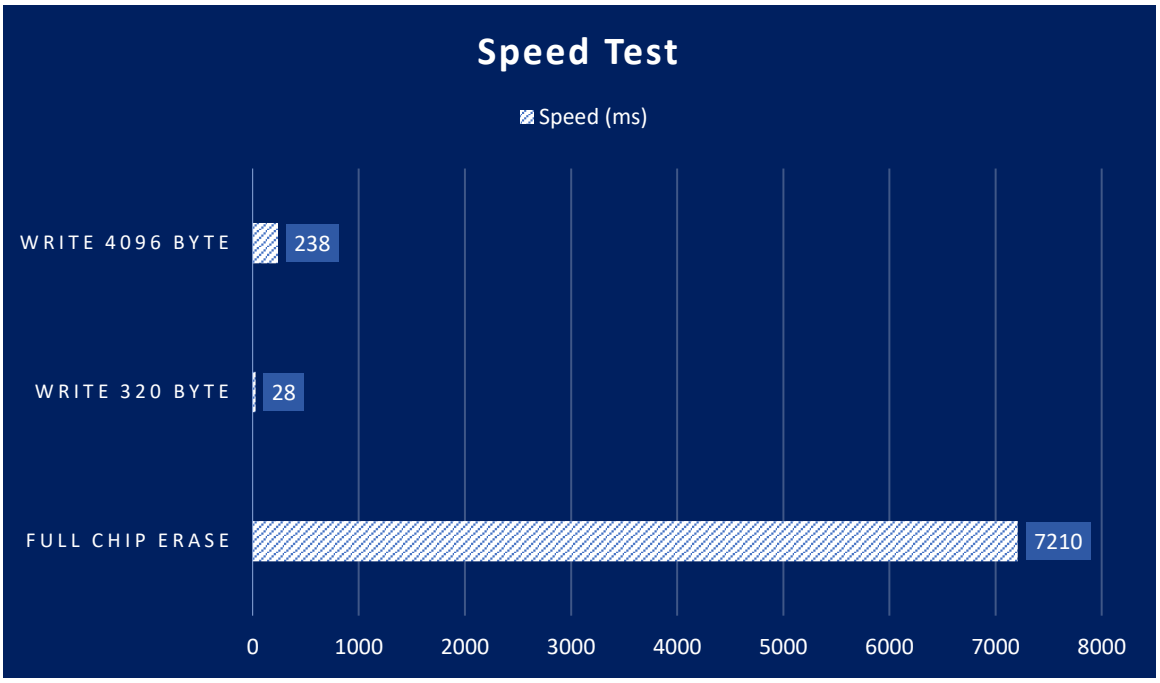
7. Supported memories

- **Winbond**

W25Q10	W25Q20	W25Q40	W25Q80	W25Q16	W25Q32	W25Q64	W25Q128	W25Q256	W25Q512
W25Q01									

8. Tests performed

- **Speed test in 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
- ⓘ License and copyright notice
- ⓘ State changes

This is not legal advice. Learn more about repository licenses.