# Lesson 7 SPI_Flash Read and Write

## 1. Project Purpose

Learn the principle FLASH memory and SPI bus communication, realize reading and writing of SPI FLASH on the controller, and display the written characters in the serial port assistant.

## 2. Project Principle

Flash memory is a type of nonvolatile memory that can retain data for an extended period of time even without current supply, and its storage characteristic is equivalent to the hard disk drive. This feature is the basis for flash memory to become the storage medium for various portable digital devices.

The Serial Peripheral Interface (SPI), developed by Motorola, is a high-speed full duplex interface. It is widely used in ADCs, LCDs and MCUs and suitable for occasions with higher communication speed requirements.

```
/*****************************************************
SPI Initialization
Entrance parameter: None
Exit parameter: None
*****************************************************/
void InitSpi(void)
{
    pinMode(SS,OUTPUT);

    SPI.begin ();// Initialize SPI bus, set SCK(Pin13), MOSI(Pin11) and SS(Pin10) pin to output mode
    SPI.setDataMode(SPI_MODE0);/*SPI_MODE0 (Rising edge sampling, falling edge position, 0 when SCK is idle)
                               SPI_MODE1 (Rising edge position, falling edge sampling, 0 when SCK is idle)
                               SPI_MODE2 (Falling edge sampling, Rising edge position, 1 when SCK is idle)
                               SPI_MODE3 (Falling edge position, Rising edge sampling, 1 when SCK is idle)
    SPI.setBitOrder(MSBFIRST);//When setting the serial data transmission, whether to transmit the high bit or the status first, there are two types of options: LSBFIRST (lowest bit first) and MSDFIRST (highest bit first)
}
```

SPI FLASH is a type of flash memory that reads and writes through SPI interface.

The general SPI FLASH has two characteristics for reading and writing:

1) When writing, only 1 can be written, not 0.
2) When erasing, it is erased by sector (that is, all data becomes 0), and the sector size varies according to different chips (the chip we chose has 4096 bytes per sector).

Based on the above two points, we can know that the data of a byte is to change the corresponding data bit in the chip from 0 to 1 or from 1 to zero.

Because FLASH does not support writing 0 when writing so we are required to erase the corresponding sector to 0. However, the original data will be lost after erasing, so it is generally read first, and then the sector is erased. At the end, rewrite the modified data into Flash.

## 3. Program Analyst

1) The InitFlash function is called in the setup section and its function body is in Flash.cpp. It calls the InitSpi function that used to initialize SPI.
2) When reading and writing SPI Flash,corresponding commands are required to send to SPI Flash. This commands can usually be found in the chip manual. The following figure is the command of table SPI Flash:

| INSTRUCTION NAME | BYTE 1 (CODE) | BYTE 2 | BYTE 3 | BYTE 4 | BYTE 5 | BYTE 6 |
|---|---|---|---|---|---|---|
| Write Enable | 06h | | | | | |
| Write Disable | 04h | | | | | |
| Read Status Register-1 | 05h | (S7–S0) (2) | | | | |
| Read Status Register-2 | 35h | (S15-S8) (2) | | | | |
| Write Status Register | 01h | (S7–S0) | (S15-S8) | | | |
| Page Program | 02h | A23–A16 | A15–A8 | A7–A0 | (D7–D0) | |
| Quad Page Program | 32h | A23–A16 | A15–A8 | A7–A0 | (D7–D0, ...) (3) | |
| Block Erase (64KB) | D8h | A23–A16 | A15–A8 | A7–A0 | | |
| Block Erase (32KB) | 52h | A23–A16 | A15–A8 | A7–A0 | | |
| Sector Erase (4KB) | 20h | A23–A16 | A15–A8 | A7–A0 | | |
| Chip Erase | C7h/60h | | | | | |
| Erase Suspend | 75h | | | | | |
| Erase Resume | 7Ah | | | | | |
| Power-down | B9h | | | | | |
| High Performance Mode | A3h | dummy | dummy | dummy | | |
| Mode Bit Reset (4) | FFh | FFh | | | | |
| Release Power down or HPM / Device ID | ABh | dummy | dummy | dummy | (ID7-ID0) (5) | |
| Manufacturer/ Device ID (6) | 90h | dummy | dummy | 00h | (M7-M0) | (ID7-ID0) |
| Read Unique ID (7) | 4Bh | dummy | dummy | dummy | Dummy | (ID63-ID0) |
| JEDEC ID | 9Fh | (M7-M0) Manufacturer | (ID15-ID8) Memory Type | (ID7-ID0) Capacity | | |

3) We define the commands that may be used in the header file with macro definitions according to this table, which is convenient to use.

```
#define SFC_WREN            0x06                //Serial Flash command set
#define SFC_WRDI            0x04
#define SFC_RDSR            0x05
#define SFC_WRSR            0x01
#define SFC_READ            0x03
#define SFC_FASTREAD        0x0B
#define SFC_RDID            0xAB
#define SFC_PAGEPROG        0x02
#define SFC_RDCR            0xA1
#define SFC_WRCR            0xF1
#define SFC_SECTORER        0xD7
#define SFC_BLOCKER         0xD8
#define SFC_SECTOR_ERASE    0x20
#define SFC_CHIPER          0xC7

```

4) The following figure is to check whether Flash is busy. Because Flash will not receive read and write commands when it is busy. Therefore, the Flash status must be checked before reading and writing, and the reading and writing operations can be initiated until the Flash is idle.

```
void CheckBusy()
{
    digitalWrite(SS, HIGH);
    digitalWrite(SS, LOW);
    SPI.transfer(SFC_RDSR);//transfer: used to transmit a data on the SPI bus including send and receive.
    while(SPI.transfer(0) & 0x01); SPI. transfer (0) reads ststus. Returning 1 means busy and 0 means idle.
    digitalWrite(SS, HIGH);
}

```

5) After the status detection, pull down the selection. Then use the fast read command and send the address to be read. Finally, use a for loop to store the read data into the buffer. The writing operation is similar to the reading operation and the difference is that the status detection is done at the end.

Finally, the command to erase Flash sector is as follow:

```
74
75  void FlashEraseSector(DWORD addr)
76  {
77      digitalWrite(SS, HIGH);
78      digitalWrite(SS, LOW);
79      SPI.transfer(SFC_WREN);//enable Flash write command
80      digitalWrite(SS, HIGH);
81      digitalWrite(SS, LOW);
82      SPI.transfer(SFC_SECTOR_ERASE);//Erase command
83      SPI.transfer((BYTE)(addr>>16));
84      SPI.transfer((BYTE)(addr>>8));
85      SPI.transfer((BYTE)addr);
86      digitalWrite(SS, HIGH);
87      CheckBusy();
```

```
39  void FlashRead(DWORD addr, DWORD size, BYTE *buffer)
40  {
41      digitalWrite(SS, HIGH);
42      digitalWrite(SS, LOW);//Pull down selection
43      SPI.transfer(SFC_READ);//send quick read command
44      SPI.transfer((BYTE)(addr>>16));
45      SPI.transfer((BYTE)(addr>>8));
46      SPI.transfer((BYTE)addr);
47      for(int i = 0; i < size; i++)
48      {
49          buffer[i] = SPI.transfer(0);The fifth byte starts with the internal storage data returned by Flash. The 0 in the brackets can be any value, but it is only used to trigger sck.
50      }
51      digitalWrite(SS, HIGH);
52      CheckBusy();
53  }
54
55  void FlashWrite(DWORD addr, DWORD size, BYTE *buffer)
56  {
57      digitalWrite(SS, HIGH);
58      digitalWrite(SS, LOW);
59      SPI.transfer(SFC_WREN);//Send write enable command
60      digitalWrite(SS, HIGH);
61      digitalWrite(SS, LOW);
62      SPI.transfer(SFC_PAGEPROG);//Send page editing commands
63      SPI.transfer((BYTE)(addr>>16));//Set start command
64      SPI.transfer((BYTE)(addr>>8));
65      SPI.transfer((BYTE)addr);
66      for(int i = 0; i < size; i++)
67      {
```