北京大学
PEKING UNIVERSITY

实验3.4：简单读写Flash

房宇轩

目 录

CONTENTS

Flash与SPI

北京大学 PEKING UNIVERSITY

- **FLSAH**

  掉电后数据不丢失的存储器。

  在存储控制上，FLASH芯片只能一大片地擦写（扇区擦除和整片擦除），不能单个字节擦写。

- **SPI通讯串口方式**

  SPI的SI及SO都连接到数据**移位寄存器**上。

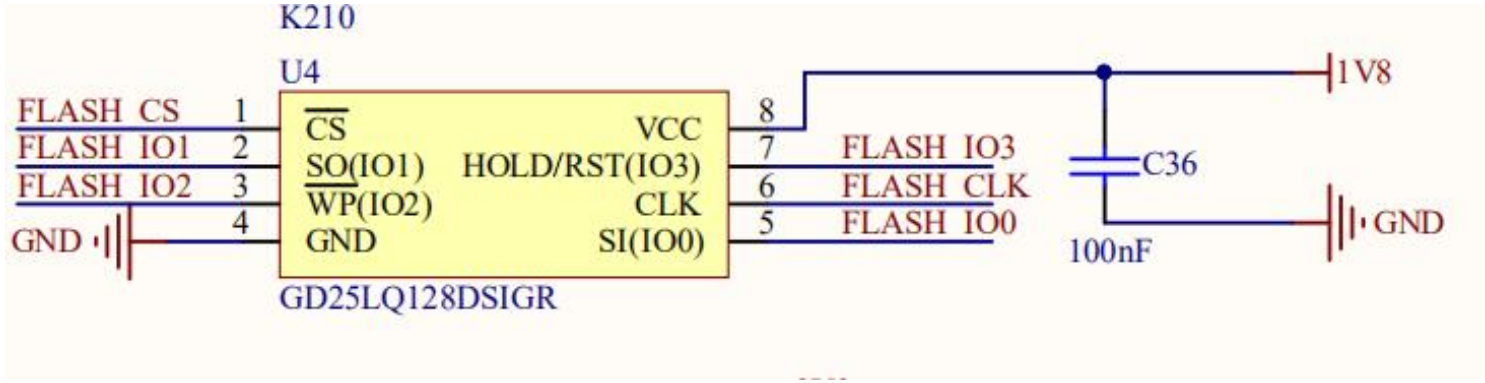
  数据移位寄存器的数据来源来源于接收缓冲区及发送缓冲区。

  通过写SPI的“数据寄存器DR”把数据填充到发送缓冲区中。

  通过读“数据寄存器DR”，可以获取接收缓冲区中的内容。

硬件连接

- 由于SPI速率较高，如在下面IO加RC，务必注意控制RC时间常数不能太大，否则引起信号质量问题，导致读写问题，必须是1.8V SPI FLASH



- SPI 通信需要四根线： CS_n、CLK、SI和 SO。
- CS_n、CLK、SI是由主机输出给从机，而 SO由从机输出给主机。
- CS_n：控制从机是否被选中，也就是说只有片选信号有效时，对从机的操作才有效；
- CLK:由主机产生的同步时钟，用于同步数据；
- SI和SO ： 串行数据输入引脚和发送的数据脚。传输数据时。
- WP_n：低电平可读可写，高电平仅可读

实验例程

- 初始化通讯使用的目标引脚及端口时钟；

w25qxx_init:

- 配置SPI的模式、地址、速率等参数并使能SPI初始化；
- 使能SPI的时钟；
- 初始化并使W25QXX在quad模式进行读操作

main.c

```
sysctl_pll_set_freq(SYSCTL_PLL0, 800000000);
uarths_init();
uint8_t manuf_id, device_id;
uint32_t index, spi_index;
spi_index = 3;
printf("spi%d master test\n", spi_index);

w25qxx_init(spi_index, 0, 60000000);
```

w25qxx.c

```
w25qxx_status_t w25qxx_init(uint8_t spi_index, uint8_t spi_ss, uint32_t rate)
{
    spi_bus_no = spi_index;
    spi_chip_select = spi_ss;
    spi_init(spi_bus_no, SPI_WORK_MODE_0, SPI_FF_STANDARD, DATALENGTH, 0);
    spi_set_clk_rate(spi_bus_no, rate);
    w25qxx_enable_quad_mode();
    return W25QXX_OK;
}
```

实验例程

北京大学 PEKING UNIVERSITY

- 读取w25qxx厂商和设备id
- 判断是否是目标设备
  - 如果不是则打印出厂商和设备id之后结束程序

main.c

```
w25qxx_read_id(&manuf_id, &device_id);
printf("manuf_id:0x%02x, device_id:0x%02x\n", manuf_id, device_id);
if ((manuf_id != 0xEF && manuf_id != 0xC8) || (device_id != 0x18 && device_id
!= 0x17 && device_id != 0x16))
{
    printf("manuf_id:0x%02x, device_id:0x%02x\n", manuf_id, device_id);
    return 0;
}
```

w25qxx.c

```
w25qxx_status_t w25qxx_read_id(uint8_t *manuf_id, uint8_t *device_id)
{
    uint8_t cmd[4] = {READ_ID, 0x00, 0x00, 0x00};
    uint8_t data[2] = {0};

    w25qxx_receive_data(cmd, 4, data, 2);
    *manuf_id = data[0];
    *device_id = data[1];
    return W25QXX_OK;
}
```

实验例程

```c
for (index = 0; index < TEST_NUMBER; index++)
    data_buf[index] = (uint8_t)(index);

/*write data*/
uint64_t start = sysctl_get_time_us();
w25qxx_write_data(DATA_ADDRESS, data_buf, TEST_NUMBER);
uint64_t stop = sysctl_get_time_us();
printf("%ld us\n", (stop - start));

for(index = 0; index < TEST_NUMBER; index++)
    data_buf[index] = 0;
```

- 在index到数据缓存指定位置
- FLASH写入数据
- 清除数据缓存中数据

- 读出整个扇区，并计算在扇区内的偏移
- 擦除这个扇区
- 在偏移处开始写入指定长度的数据。

```c
while (length)
{
    uint32_t sector_remain
    sector_remain = w25qxx_FLASH_SECTOR_SIZE - sector_offset;
    write_len = ((length < sector_remain) ? length : sector_remain);
    w25qxx_read_data(sector_addr, swap_buf, w25qxx_FLASH_SECTOR_SIZE);
    pread = swap_buf + sector_offset;
    pwrite = data_buf;
    for (index = 0; index < write_len; index++)
    {
        if ((*pwrite) != ((*pwrite) & (*pread)))
        {
            w25qxx_sector_erase(sector_addr);
            while (w25qxx_is_busy() == W25QXX_BUSY)
                ;
            break;
        }
        pwrite++;
        pread++;
    }
    if (write_len == w25qxx_FLASH_SECTOR_SIZE)
    {
        w25qxx_sector_program(sector_addr, data_buf);
    }
    else
    {
        pread = swap_buf + sector_offset;
        pwrite = data_buf;
        for (index = 0; index < write_len; index++)
            *pread++ = *pwrite++;
        w25qxx_sector_program(sector_addr, swap_buf);
    }
    length -= write_len;
    addr += write_len;
    data_buf += write_len;
}
return W25QXX_OK;
```

实验例程

```
w25qxx_read_data(DATA_ADDRESS, data_buf, TEST_NUMBER);
uint32_t v_remain = length % 4;
if(v_remain != 0)
{
    length = length / 4 * 4;
}

uint32_t len = 0;

while (length)
{
    len = ((length >= 0x010000) ? 0x010000 : length);
    _w25qxx_read_data(addr, data_buf, len);
    addr += len;
    data_buf += len;
    length -= len;
}

if(v_remain)
{
    uint8_t v_recv_buf[4];
    _w25qxx_read_data(addr, v_recv_buf, 4);
    memcpy(data_buf, v_recv_buf, v_remain);
}
return W25QXX_OK;
```

读数据

- 从数据起始地址开始逐步递增地址
- 读数据最长0x010000
- 将数据缓存区读出的数据送到接口

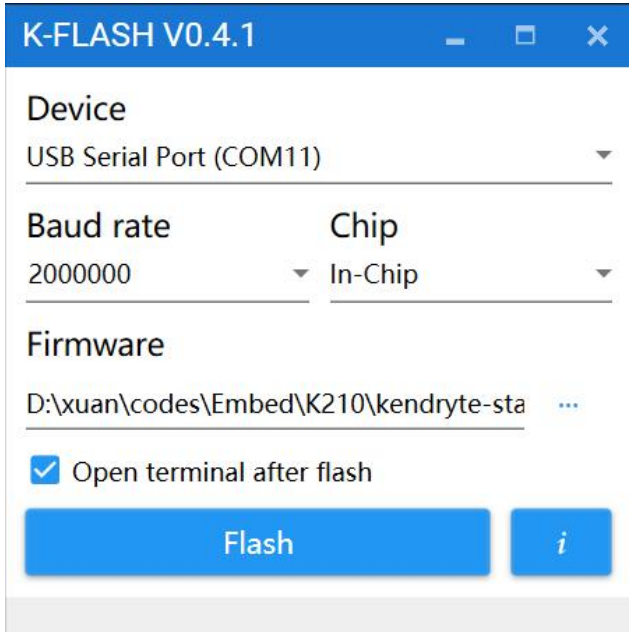
```
for (index = 0; index < TEST_NUMBER; index++)
{
    if (data_buf[index] != (uint8_t)(index))
    {
        printf("quad fast read test error\n");
        return 0;
    }
}
printf("spi%d master test ok\n", spi_index);
```

测试，对读写数据进行校验。

操作流程

```
PS D:\xuan\codes\Embed\K210\kendryte-standalone-sdk-develop> cd build
PS D:\xuan\codes\Embed\K210\kendryte-standalone-sdk-develop\build> cmake .. -DPROJ=flash_w25qxx -G "Unix Makefiles"
PROJ = flash_w25qxx
-- Check for RISCV toolchain ...
-- Using D:/software/kendryte-toolchain/bin RISCV toolchain
-- The C compiler identification is GNU 8.2.0
-- The CXX compiler identification is GNU 8.2.0
```

```
PS D:\xuan\codes\Embed\K210\kendryte-standalone-sdk-develop\build> make
Scanning dependencies of target nncase-v0
```




```
spi3 master test
manuf_id:0xc8, device_id:0x18
write data
302319 us
read 1708 us
spi3 master test ok
```