

Contents

1、介绍.....1

2、移植过程.....4

3、程序效果.....7

4、参考.....9

1、 介绍

[SFUD](#) 是一款开源的串行 SPI Flash 通用驱动库。由于现有市面的串行 Flash 种类居多，各个 Flash 的规格及命令存在差异， SFUD 就是为了解决这些 Flash 的差异现状而设计，让我们的产品能够支持不同品牌及规格的 Flash, 提高了涉及到 Flash 功能的软件的可重用性及可扩展性，同时也可以规避 Flash 缺货或停产给产品所带来的风险。

暂时支持的芯片

型号	制造商	容量	最高速度	SFDP 标准	QSP I 模式	备注
型号	制造商	容量	最高速度	SFDP 标准	Qspi 模式	备注
<a href="#">W25Q40BV</a>	Winbond	4Mb	50Mhz	不支持	双线	已停产
<a href="#">W25Q80DV</a>	Winbond	8Mb	104Mhz	支持	双线	
<a href="#">W25Q16BV</a>	Winbond	16Mb	104Mhz	不支持	双线	by <a href="#">slipperstree</a>
<a href="#">W25Q16CV</a>	Winbond	16Mb	104Mhz	支持	未测试	
<a href="#">W25Q16DV</a>	Winbond	16Mb	104Mhz	支持	未测试	by <a href="#">slipperstree</a>
<a href="#">W25Q32BV</a>	Winbond	32Mb	104Mhz	支持	双线	
<a href="#">W25Q64CV</a>	Winbond	64Mb	80Mhz	支持	四线	
<a href="#">W25Q128BV</a>	Winbond	128Mb	104Mhz	支持	四线	
<a href="#">W25Q256FV</a>	Winbond	256Mb	104Mhz	支持	四线	
<a href="#">MX25L3206E</a>	Macronix	32Mb	86MHz	支持	双线	
<a href="#">KH25L4006E</a>	Macronix	4Mb	86Mhz	支持	未测试	by <a href="#">JiapengLi</a>
<a href="#">KH25L3206E</a>	Macronix	32Mb	86Mhz	支持	双线	
<a href="#">SST25VF016</a>	Microchip	16Mb	50MHz	不支	不支	SST 已被 Microchip

<a href="#">B</a>				持	持	收购
<a href="#">M25P40</a>	Micron	4Mb	75Mhz	不支持	未测试	by <a href="#">redocCheng</a>
<a href="#">M25P80</a>	Micron	8Mb	75Mhz	不支持	未测试	by <a href="#">redocCheng</a>
<a href="#">M25P32</a>	Micron	32Mb	75Mhz	不支持	不支持	
<a href="#">EN25Q32B</a>	EON	32Mb	104MHz	不支持	未测试	
<a href="#">GD25Q16B</a>	GigaDevice	16Mb	120MHz	不支持	未测试	by <a href="#">TanekLiang</a>
<a href="#">GD25Q64B</a>	GigaDevice	64Mb	120MHz	不支持	双线	
<a href="#">S25FL216K</a>	Cypress	16Mb	65Mhz	不支持	双线	
<a href="#">S25FL032P</a>	Cypress	32Mb	104MHz	不支持	未测试	by <a href="#">yc 911</a>
<a href="#">S25FL164K</a>	Cypress	64Mb	108MHz	支持	未测试	
<a href="#">A25L080</a>	AMIC	8Mb	100MHz	不支持	双线	
<a href="#">A25LQ64</a>	AMIC	64Mb	104MHz	支持	支持	
<a href="#">F25L004</a>	ESMT	4Mb	100MHz	不支持	不支持	

<a href="#">PCT25VF016</a> B	PCT	16Mb	80Mhz	不支持	不支持	SST 授权许可，会被识别为 SST25VF016B
<a href="#">AT45DB161</a> E	ADESTO	16Mb	85MHz	不支持	不支持	ADESTO 收购 Atmel 串行闪存产品线

## 2、 移植过程

我们使用 FRDM-K64 板子，w25q32，将 sfud 移植到 k64 上。

硬件连接

FRDM-k64 引脚	w25q32 引脚
PTD0 (复用为 GPIO)	CS
PTD1 (SCK)	SCK
PTD2 (MISO)	SOUT
PTD3 (MOSI)	SIN

- 1、下载 sfud 源码 <https://github.com/armink/SFUD>
- 2、我们要使用的就是 sfud 这个文件夹。我们使用 k64 库函数的 dspi 的 master 的 polling 例程来进行移植。将他添加到工程中，添加头文件路径

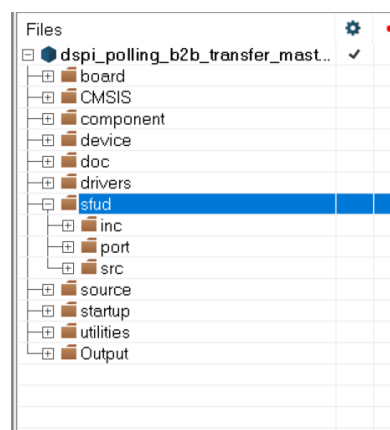


图 1.添加文件

- 3、在 sfud\_cfg.h 添加 SFUD\_USING\_FLASH\_INFO\_TABLE 的索引设备，SFUD\_FLASH\_DEVICE\_TABLE 设置索引名称等基本信息

```

enum {
    SFUD_W25Q32_DEVICE_INDEX = 0,
};

#define SFUD_FLASH_DEVICE_TABLE
{
    [SFUD_W25Q32_DEVICE_INDEX] = { .name = "w25q32", .spi.name = "SPI0"},
}

```

图 2.定义设备信息

- 4、sfud\_port.c 中，sfud\_spi\_port\_init 添加设备 case，然后对应初始化其 IO 引脚，时钟，GPIO，片选引脚配置为 GPIO，不要配置为 SPI 的 cs，初始化 spi。我们在 main 函数里一开始调用 sfud\_init 函数。这个函数实现就是获取 SFUD\_FLASH\_DEVICE\_TABLE 中设备，然后根据索引值去初始化对应的设备。

sfud\_spi\_port\_init 需要自己去实现的函数有，spi 时钟配置函数，引脚配置函数，spi 配置函数，最重要的是实现 spi\_write\_read 函数，他决定了 mcu 是否能正确读取数据。spi\_lock，和 spi\_unlock 实现就是开关中断。还要定义一个用户数据配置，他保存了一些 spi 基本信息，使用的是哪个 spi，片选 gpio 的基地址，片选信号是第几脚

```

static spi_user_data spi0 = { .spix = SPI0, .cs_gpiox = GPIOD, .cs_gpio_pin = PIN0_IDX };

sfud_err sfud_spi_port_init(sfud_flash *flash) {
    sfud_err result = SFUD_SUCCESS;

    /*
    switch (flash->index)
    {
    case SFUD_W25Q32_DEVICE_INDEX:
        {
            spi_clock_configuration(&spi0);
            port_configuration(&spi0);
            gpio_pin_config_t cs_config = {kGPIO_DigitalOutput, 1,};
            GPIO_PinInit (GPIOD,0,&cs_config);
            spi_configuration(&spi0);

            flash->spi.wr = spi_write_read;
            flash->spi.lock = spi_lock;
            flash->spi.unlock = spi_unlock;
            flash->spi.user_data = &spi0;
            /* about 100 microsecond delay */
            flash->retry.delay = retry_delay_100us;
            /* about 60 seconds timeout */
            flash->retry.times = 60 * 10000;
        }
        break;
    }
    return result;
}

```

图 3. sfud\_spi\_port\_init 实现以及用户数据的定义

```

static void spi_clock_configuration(spi_user_data_t spi)
{
    if(spi->spix == SPI0)
    {
        CLOCK_EnableClock(kCLOCK_PortD);
    }
    else if(spi->spix == SPI1)
    {
    }
}

static void port_configuration(spi_user_data_t spi)
{
    if(spi->spix == SPI0)
    {
        PORT_SetPinMux(PORTD, PIN0_IDX, kPORT_MuxAsGpio);
        PORT_SetPinMux(PORTD, PIN1_IDX, kPORT_MuxAlt2);
        PORT_SetPinMux(PORTD, PIN2_IDX, kPORT_MuxAlt2);
        PORT_SetPinMux(PORTD, PIN3_IDX, kPORT_MuxAlt2);
    }
    else if(spi->spix == SPI1)
    {
    }
}

```

*/\* PORTD0 (pin 93) is configured as SPI0\_PCS0 \*/  
/\* PORTD1 (pin 94) is configured as SPI0\_SCK \*/  
/\* PORTD2 (pin 95) is configured as SPI0\_SOUT \*/  
/\* PORTD3 (pin 96) is configured as SPI0\_SIN \*/*

图 4.spi 以及 port 初始化

```

static void spi_configuration(spi_user_data_t spi)
{
    if(spi->spix == SPI0)
    {
        dspi_master_config_t masterConfig;

        uint32_t srcClock_Hz;
        /* Master config */
        masterConfig.whichCtar = kDSPI_Ctar0;
        masterConfig.ctarConfig.baudRate = TRANSFER_BAUDRATE;
        masterConfig.ctarConfig.bitsPerFrame = 8U;
        masterConfig.ctarConfig.cpol = kDSPI_ClockPolarityActiveHigh;
        masterConfig.ctarConfig.cpha = kDSPI_ClockPhaseFirstEdge;
        masterConfig.ctarConfig.direction = kDSPI_MsbFirst;
        masterConfig.ctarConfig.pcsToSckDelayInNanoSec = 1000000000U / TRANSFER_BAUDRATE;
        masterConfig.ctarConfig.lastSckToPcsDelayInNanoSec = 1000000000U / TRANSFER_BAUDRATE;
        masterConfig.ctarConfig.betweenTransferDelayInNanoSec = 1000000000U / TRANSFER_BAUDRATE;

        masterConfig.whichPcs = EXAMPLE_DSPI_MASTER_PCS_FOR_INIT;
        masterConfig.pcsActiveHighOrLow = kDSPI_PcsActiveLow;

        masterConfig.enableContinuousSCK = false;
        masterConfig.enableRxFifoOverWrite = false;
        masterConfig.enableModifiedTimingFormat = false;
        masterConfig.samplePoint = kDSPI_SckToSin0Clock;

        srcClock_Hz = DSPI_MASTER_CLK_FREQ;
        DSPI_MasterInit(EXAMPLE_DSPI_MASTER_BASEADDR, &masterConfig, srcClock_Hz);
    }
    else if(spi->spix == SPI1)
    {
    }
}

```

图 5.spi 配置

```

static void spi_lock(const sfud_spi *spi)
{
    __disable_irq();
}

static void spi_unlock(const sfud_spi *spi)
{
    __enable_irq();
}

```

图 6.spi 锁配置

在实现 spi\_write\_read 函数之前，先要实现 spi 传输配置函数。

```

static void SPI_Transfer(SPI_Type* SPIx, uint8_t* masterTxData, uint8_t* masterRxData, uint8_t size)
{
    dspi_transfer_t masterXfer;
    masterXfer.txData = masterTxData;
    masterXfer.rxData = masterRxData;
    masterXfer.dataSize = size;
    masterXfer.configFlags = kDSPI_MasterPcsContinuous; //kDSPI_MasterCtar0 / EXAMPLE_DSPI_MASTER_PCS_FOR_TRANSFER /
    DSPI_MasterTransferBlocking(SPIx, &masterXfer);
}

static uint8_t SPI_ReadWriteByte(SPI_Type* SPIx, uint8_t inData)
{
    uint8_t a = inData, b=0x00;
    SPI_Transfer(SPIx, &a, &b, 1);
    return b;
}

```

图 7.spi 传输配置函数

```

static sfud_err spi_write_read(const sfud_spi *spi, const uint8_t *write_buf, size_t write_size, uint8_t *read_buf,
size_t read_size)
{
    #if 1
    sfud_err result = SFUD_SUCCESS;
    uint8_t send_data;
    spi_user_data_t spi_dev = (spi_user_data_t) spi->user_data;

    if (write_size)
    {
        SFUD_ASSERT(write_buf);
    }
    if (read_size)
    {
        SFUD_ASSERT(read_buf);
    }

    GPIO_PinWrite(spi_dev->cs_gpioux, spi_dev->cs_gpio_pin, 0);
    /* 开始读写数据 */
    for (size_t i = 0; i < write_size + read_size; i++)
    {
        /* 先写缓冲区中的数据到 SPI 总线, 数据写完后, 再写 dummy(0xFF) 到 SPI 总线 */
        if (i < write_size)
        {
            send_data = *write_buf++;
            SPI_ReadWriteByte(spi_dev->spix, send_data);
        }
        else
        {
            send_data = SFUD_DUMMY_DATA;
            *read_buf++ = SPI_ReadWriteByte(spi_dev->spix, send_data);
        }
    }
    GPIO_PinWrite(spi_dev->cs_gpioux, spi_dev->cs_gpio_pin, 1);
    #endif
    return result;
}

```

图 8.spi\_write\_read 实现

5、sfud\_log\_debug, sfud\_log\_info 里的串口打印函数要修改。  
修改完以后, 在 main 函数调用 sfud\_init。

### 3、 程序效果

如果成功的话, 串口会打印以下数据。只要红线数值能正常被读出来, 那么基本就移植成功了。红线就是工厂 id, 内存类型, 容量 id, 每个 flash 存储都不一样, 如果读出来都是 0xff 那就有问题, 得检查 io 配置是否正确, 读写函数是否正常。在下载的文件 demo 里还有 sfud\_demo 来测试 flash, 直接复制进来用, 不需要修改。完整效果如下

```

Connecting to COM30...
Connected.

DSPI board to board polling example.
[SFUD](..\sfud\src\sfuld.c:ld) Start initialize Serial Flash Universal Driver(SFUD) V1.1.0. [SFUD](..\sfud\src\sfuld.c:ld) You can get the latest version o
n https://github.com/armink/SFUD . [SFUD](..\sfud\src\sfuld.c:ld) The flash device manufacturer ID is 0xEF, memory type ID is 0x40, capacity ID is 0x16. [S
FUD](..\sfud\src\sfuld_sfdp.c:ld) Check SFDP header is OK. The reversion is V1.5, NPN is 0. [SFUD](..\sfud\src\sfuld_sfdp.c:ld) Check JEDEC basic flash par
ameter header is OK. The table id is 0, reversion is V1.5, length is 16, parameter table pointer is 0x000080. [SFUD](..\sfud\src\sfuld_sfdp.c:ld) JEDEC ba
sic flash parameter table info:
[SFUD](..\sfud\src\sfuld_sfdp.c:ld) MSB-LSB 3 2 1 0 [SFUD](..\sfud\src\sfuld_sfdp.c:ld) [0001] 0xFF 0xF9 0x20 0xE5 [
SFUD](..\sfud\src\sfuld_sfdp.c:ld) [0002] 0x01 0xFF 0xFF 0xFF [SFUD](..\sfud\src\sfuld_sfdp.c:ld) [0003] 0x6B 0x08 0xEB 0x44 [SFUD](..\sfud\src\sfuld_sfdp.c:
ld) [0004] 0xBB 0x42 0x3B 0x08 [SFUD](..\sfud\src\sfuld_sfdp.c:ld) [0005] 0xFF 0xFF 0xFF 0xFE [SFUD](..\sfud\src\sfuld_sfdp.c:ld) [0006] 0x00 0x00 0xFF 0xFF [
SFUD](..\sfud\src\sfuld_sfdp.c:ld) [0007] 0xEB 0x40 0xFF 0xFF [SFUD](..\sfud\src\sfuld_sfdp.c:ld) [0008] 0x52 0x0F 0x20 0x0C [SFUD](..\sfud\src\sfuld_sfdp.c:
ld) [0009] 0x00 0x00 0xD8 0x10 [SFUD](..\sfud\src\sfuld_sfdp.c:ld) 4 KB Erase is supported throughout the device. Command is 0x20. [SFUD](..\sfud\src\sfuld_
sfdp.c:ld) Write granularity is 64 bytes or larger. [SFUD](..\sfud\src\sfuld_sfdp.c:ld) Target flash status register is non-volatile. [SFUD](..\sfud\src\sfuld_
sfdp.c:ld) 3-Byte only addressing. [SFUD](..\sfud\src\sfuld_sfdp.c:ld) Capacity is 4194304 Bytes. [SFUD](..\sfud\src\sfuld_sfdp.c:ld) Flash device suppor
ts 4KB block erase. Command is 0x20. [SFUD](..\sfud\src\sfuld_sfdp.c:ld) Flash device supports 32KB block erase. Command is 0x52. [SFUD](..\sfud\src\sfuld_s
fdp.c:ld) Flash device supports 64KB block erase. Command is 0x08. [SFUD](..\sfud\src\sfuld_sfdp.c:ld) Find a Winbond flash chip. Size is 4194304 bytes. [SFUD](..\sfud\src\sfuld.c:ld) F
lash device reset success. [SFUD]w25q32 flash device is initialize success.
[SFUD]w25q32 flash device is initialize success.
Write the w25q32 flash data finish. Start from 0x00000000, size is 1024. Erase the w25q32 flash data finish. Start from 0x00000000, size is 1024.
Read the w25q32 flash data success. Start from 0x00000000, size is 1024. The data is:
Offset (h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
[00000000] 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
[00000010] 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
[00000020] 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
[00000030] 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F

```

图 8.运行效果 1

```

[00000170] 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
[00000180] 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
[00000190] 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
[000001A0] A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
[000001B0] B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
[000001C0] C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
[000001D0] D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
[000001E0] E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
[000001F0] F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
[00000200] 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
[00000210] 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
[00000220] 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
[00000230] 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
[00000240] 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
[00000250] 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
[00000260] 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
[00000270] 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
[00000280] 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
[00000290] 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
[000002A0] A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
[000002B0] B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
[000002C0] C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
[000002D0] D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
[000002E0] E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
[000002F0] F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF
[00000300] 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
[00000310] 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
[00000320] 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
[00000330] 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
[00000340] 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
[00000350] 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
[00000360] 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
[00000370] 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F
[00000380] 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
[00000390] 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
[000003A0] A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
[000003B0] B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
[000003C0] C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF
[000003D0] D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF
[000003E0] E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF
[000003F0] F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF

The w25q32 flash test is success.

```

图 9.运行效果 2



## 4、参考

- 1) [SFUD](#)