STM32Cube高效开发教程（基础篇）

第19章 FSMC连接外部SRAM

王维波
中国石油大学（华东）控制科学与工程学院

STM32Cube高效开发教程（基础篇）

作者：王维波，鄢志丹，王钊

人民邮电出版社

2021年9月出版

如果有读者需要本书课件的PPT版本用于备课，可以给作者发邮件免费获取，并可加入专门的教学和技术交流QQ群

邮箱：wangwb@upc. edu. cn

19.1 FSMC连接外部SRAM的原理

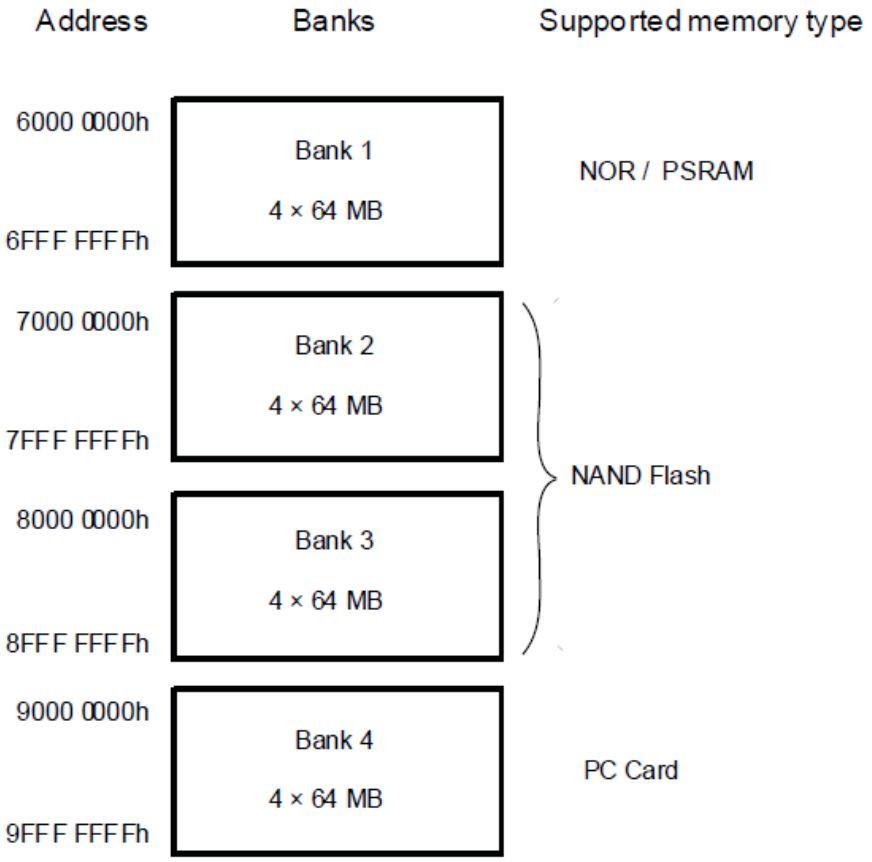
19.2 访问外部SRAM的HAL驱动程序

19.3 示例1：轮询方式读写外部SRAM

19.1.1 FSMC控制区域的划分

FSMC能够连接NOR/PSRAM、NAND Flash、PC Card、TFT LCD等。FSMC连接的所有外部存储器共享地址、数据和控制信号，但有各自的片选信号。

FSMC外部存储器被划分为4个固定大小的存储区域（memory bank），每个区域大小为256MB

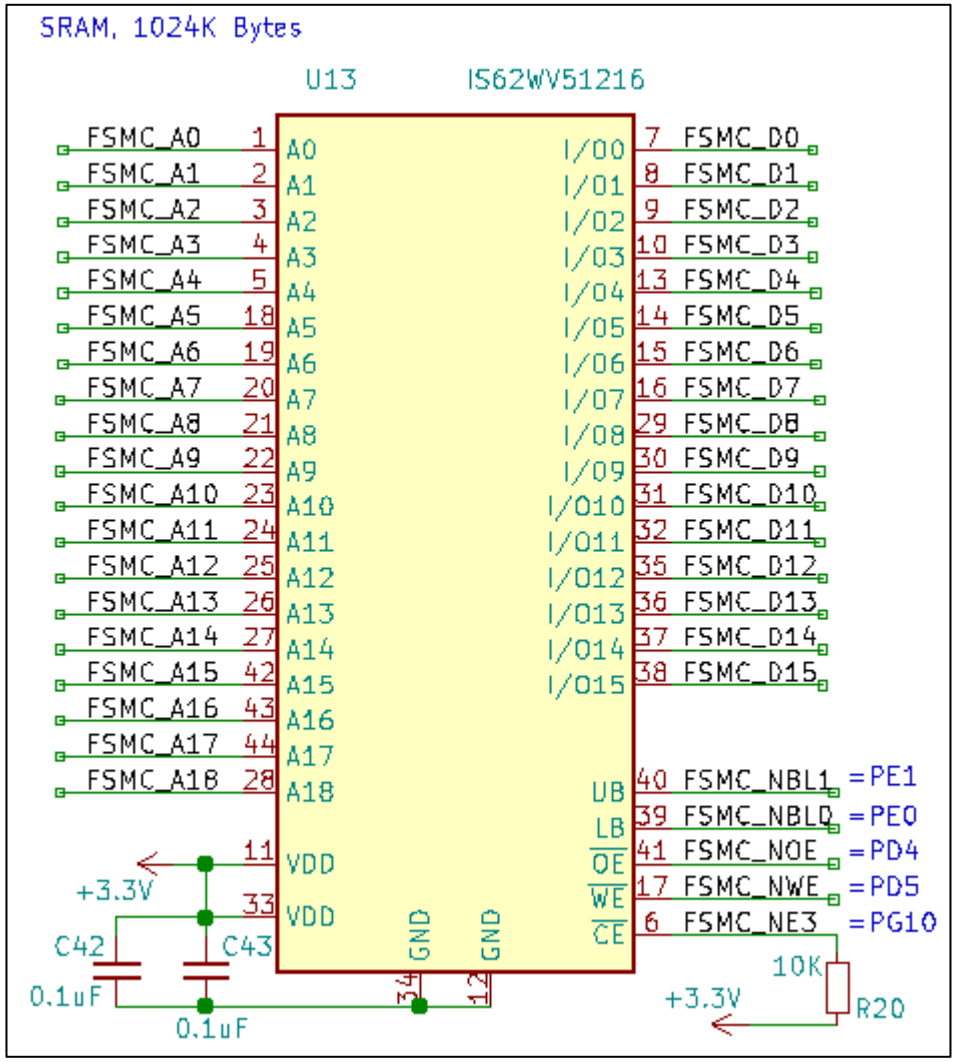


Bank 1被分为4个子区域，每个子区域容量64MB，有专用的片选信号。

- Bank 1- NOR/PSRAM 1，片选信号NE1
- Bank 1- NOR/PSRAM 2，片选信号NE2
- Bank 1- NOR/PSRAM 3，片选信号NE3（开发板上用于连接扩展SRAM）
- Bank 1- NOR/PSRAM 4，片选信号NE4（开发板上用于连接TFT LCD）

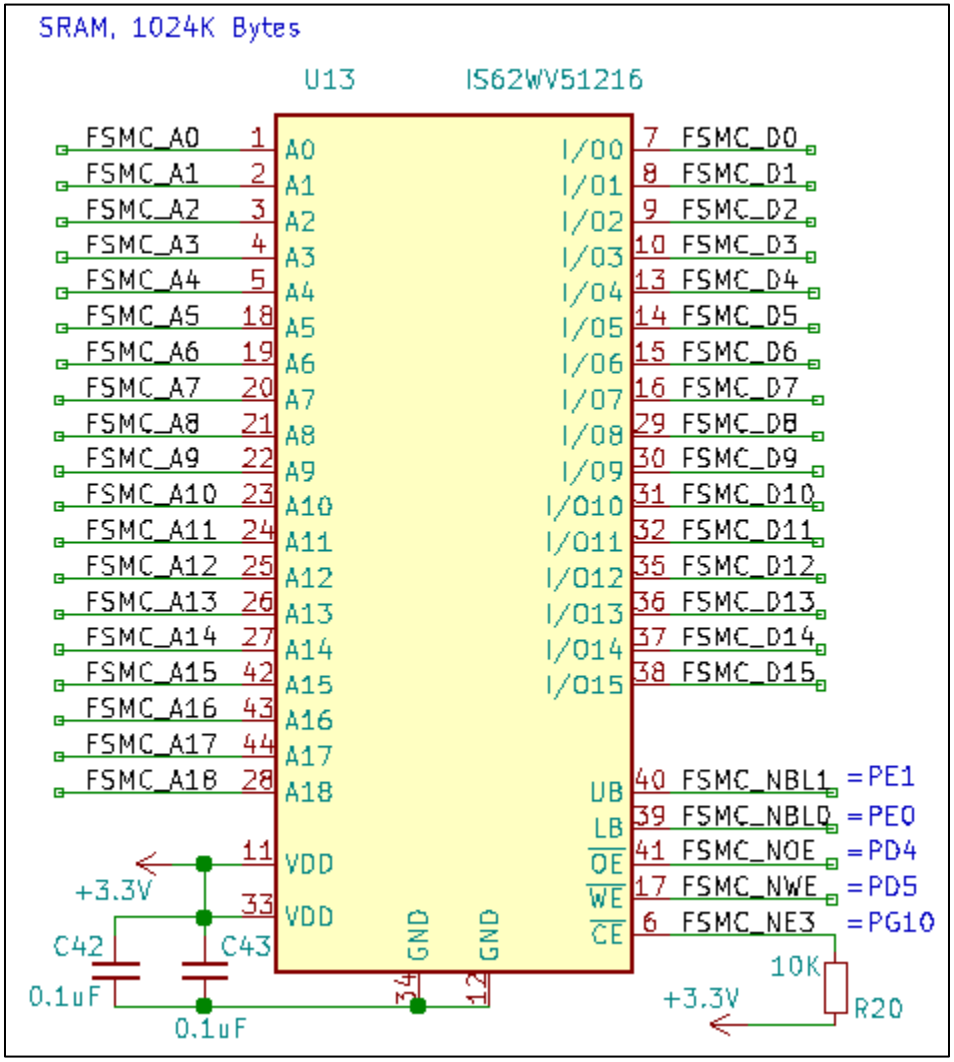
19.1.2 SRAM芯片与MCU的连接

开发板上有一个SRAM芯片IS62WV51216，是512 K*16位，即1024K字节。它与MCU的连接电路如图所示



- A0至A18是19根地址线，连接FSMC的19根地址线，即FSMC_A0至FSMC_A18
- I/O0至I/O15是16位数据线，连接FSMC的FSMC_D0至FSMC_D15数据线

- /CE是芯片的片选信号
- /OE是输出使能信号，是读数据时的使能信号
- /WE是写使能信号，是写数据使能信号
- UB是高字节使能信号；LB是低字节使能信号。通过UB和LB的控制可以只读取一个地址的高字节（I/O15-I/O8）或低字节（I/O7-I/O0）数据，或读取16位数据



19.1 FSMC连接外部SRAM的原理

19.2 访问外部SRAM的HAL驱动程序

19.3 示例1：轮询方式读写外部SRAM

19.2.1 外部SRAM初始化与控制

函数名	功能描述
HAL_SRAM_Init()	外部SRAM初始化函数，主要是FSMC访问接口的定义
HAL_SRAM_MspInit()	外部SRAM初始化MSP函数，需重新实现，主要是GPIO配置和中断设置
HAL_SRAM_WriteOperation_Enable()	使能SRAM存储器的写操作
HAL_SRAM_WriteOperation_Disable()	禁止SRAM存储器的写操作
HAL_SRAM_GetState()	返回SRAM存储器的当前状态，返回值是枚举类型HAL_SRAM_StateTypeDef

CubeMX生成的FSMC外设初始化函数，初始化程序文件里会定义一个FSMC子区外设对象变量，如：

```
SRAM_HandleTypeDef  hsram3;  //访问外部SRAM的FSMC子区外设对象变量
```

19.2.2 外部SRAM读写函数

函数名	功能描述
HAL_SRAM_Read_8b()	从指定地址读取指定长度的8位数据到一个缓存区
HAL_SRAM_Write_8b()	向指定地址写入一定长度的8位数据
HAL_SRAM_Read_16b()	从指定地址读取指定长度的16位数据到一个缓存区
HAL_SRAM_Write_16b()	向指定地址写入一定长度的16位数据
HAL_SRAM_Read_32b()	从指定地址读取指定长度的32位数据到一个缓存区
HAL_SRAM_Write_32b()	向指定地址写入一定长度的32位数据

这些函数的输入参数形式相似，例如，HAL_SRAM_Write_8b()的原型定义如下：

```
HAL_StatusTypeDef HAL_SRAM_Write_8b(SRAM_HandleTypeDef *hsram,
    uint32_t *pAddress, uint8_t *pSrcBuffer, uint32_t BufferSize)
```

其中，

- hsram是FSMC子区对象指针；
- pAddress是需要写入的SRAM目标地址指针；
- pSrcBuffer是源数据的缓存区地址指针；
- BufferSize是源数据缓存区长度，即数据点个数。

开发板上使用FSMC的Bank1子区3访问外部SRAM，Bank1子区3的起始地址是0x6800 0000，那么向这个起始地址的外部SRAM写入一个字符串的示意代码如下：

```
uint32_t *pAddr=(uint32_t *)(0x68000000);     //给指针赋值
uint8_t  strIn[]="Moment in UPC";             //准备写入的字符串
uint16_t dataLen=sizeof(strIn);               //数据长度，包括最后的结束符'\0'
HAL_SRAM_Write_8b(&hsram3, pAddr_8b, strIn, dataLen);
```

读取8位数据的函数HAL_SRAM_Read_8b()的原型定义如下：

```
HAL_StatusTypeDef HAL_SRAM_Read_8b(SRAM_HandleTypeDef *hsram,
      uint32_t *pAddress, uint8_t *pDstBuffer, uint32_t BufferSize);
```

其中，pAddress是需要读取的SRAM目标地址指针，pDstBuffer是读出数据的缓存区，BufferSize是缓存区长度，即数据点个数

例如，从SRAM起始地址偏移1024字节处读取一个uint8_t类型数组数据的示意代码如下：

```
uint32_t *pAddr=(uint32_t *)(0x68000000+1024);              //给指针赋值
uint8_t  strOut[30];
uint16_t dataLen=30;                 //数据点个数
HAL_SRAM_Read_8b(&hsram3, pAddr, strOut, dataLen);
```

19.2.3 直接通过指针访问外部SRAM

可以直接使用指针访问外部SRAM的数据。例如，向SRAM一个目标地址写入一批uint16_t类型数据的示意代码如下：

```
uint16_t num=1000;
uint16_t *pAddr_16b=(uint16_t *)(0x68000000);   //16位数据的指针
for(uint16_t i=0; i<10; i++)        //连续写入10个16位整数
{
    num += 5;
    *pAddr_16b =num;                //直接向指针所指的地址写入数据
    pAddr_16b++;                    //++一次，地址加2, 因为是16位数据
    LCD_ShowUint(50,LCD_CurY+20, num);
}
```

同样地，也可以通过指针读出数据，示意代码如下：

```
uint16_t num=0, data[10];
uint16_t  *pAddr_16b=(uint16_t *)(0x68000000);              //指针赋值
for(uint16_t i=0; i<10; i++)
{
    num=*pAddr_16b;                          //直接从指针所指的地址读数
    data[i]=num;
    pAddr_16b++;                  //++一次，地址加2, 因为是16位数据
}
```

19.2.4 DMA方式读写外部SRAM

外部SRAM还可以通过DMA方式读写，DMA2控制器具有存储器到存储器的DMA流。

函数名	功能
HAL_SRAM_Read_DMA()	以DMA方式从指定地址读取一定长度的32位数据
HAL_SRAM_Write_DMA()	以DMA方式向指定地址写入一定长度的32位数据
HAL_SRAM_DMA_XferCpltCallback()	DMA流传输完成事件中断的回调函数
HAL_SRAM_DMA_XferErrorCallback()	DMA流传输错误事件中断的回调函数

19.1 FSMC连接外部SRAM的原理

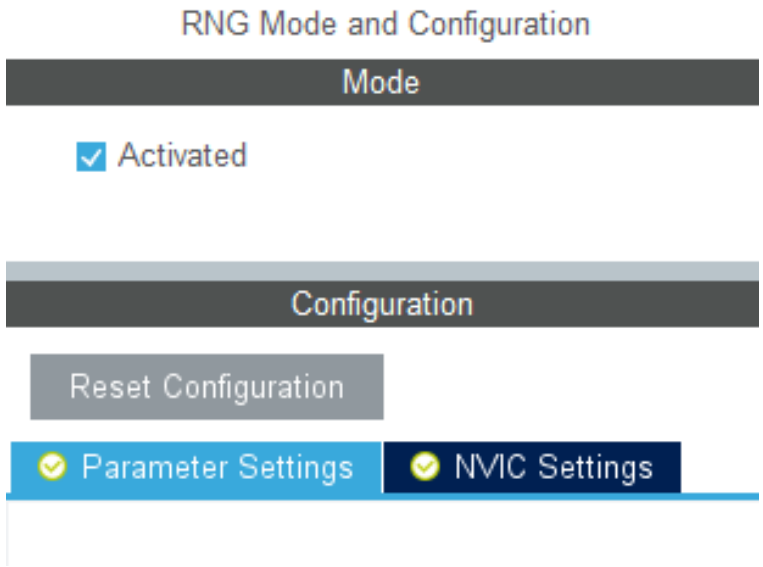
19.2 访问外部SRAM的HAL驱动程序

19.3 示例1：轮询方式读写外部SRAM

19.3.1 示例功能和CubeMX项目设置

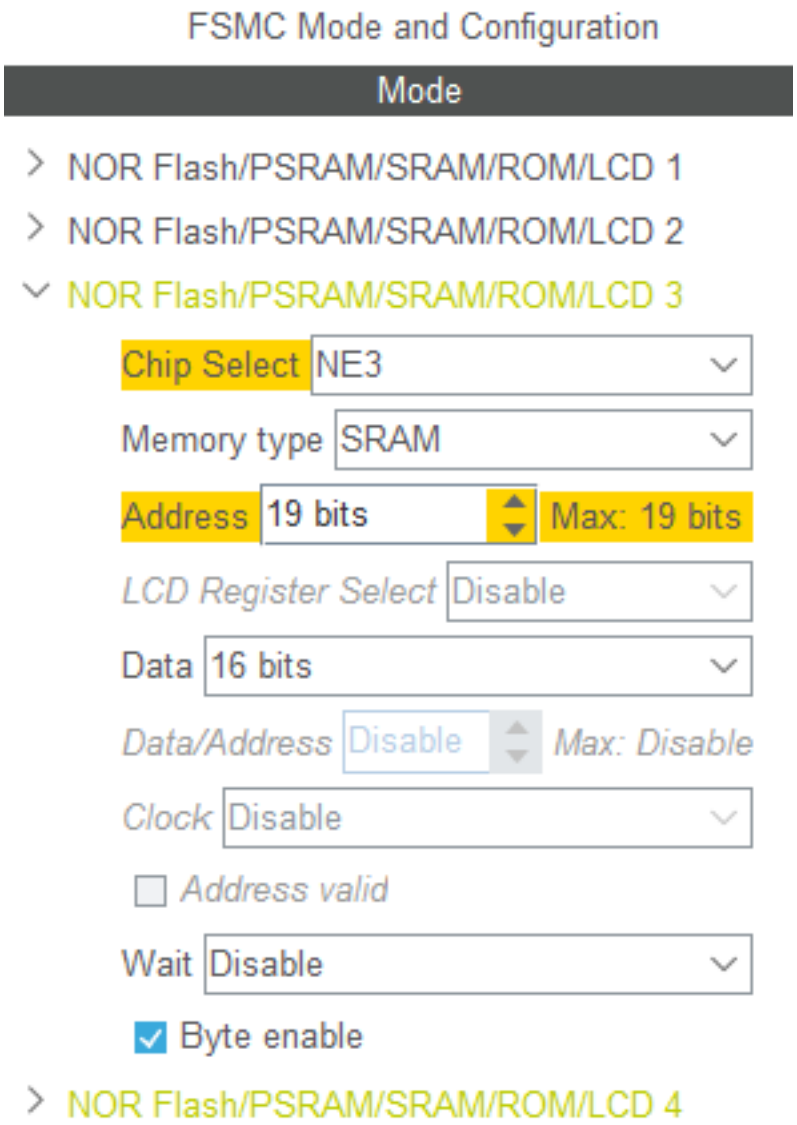
本节示例Demo19_1SRAM，演示连接外部SRAM的FSMC接口设置，以及轮询方式读写外部SRAM的方法。

本示例中还用到随机数生成器RNG，在组件面板Security分组里有RNG模块，启用RNG即可，没有任何参数设置。



1. FSMC的Bank 1子区3模式设置

- Chip Select设置为NE3
- Memory type设置为SRAM
- Address设置为19 bits，因为用到了FSMC_A0至FSMC_A18
- Data设置为16 bits
- Wait设置为Disable。Wait是PSRAM芯片发给FSMC的等待输入信号，SRAM芯片没有这个信号
- Byte enable需要勾选，表示允许字节访问。允许字节访问时，将通过芯片的UB和LB信号控制访问高位字节和低位字节

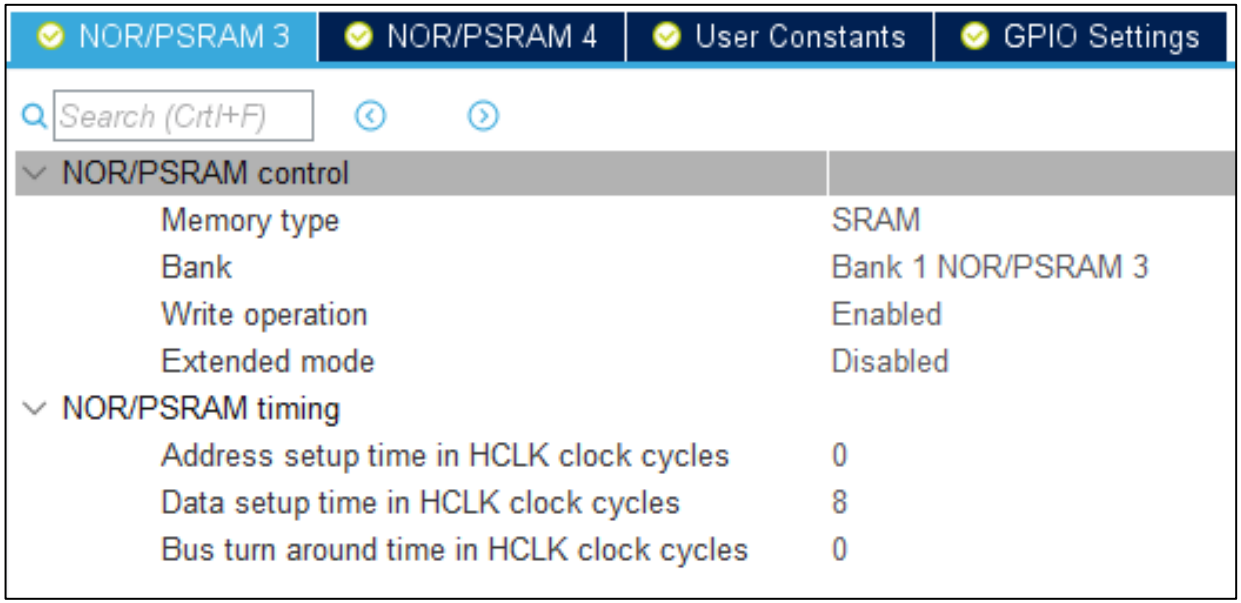


FSMC自动分配的GPIO引脚与实际电路的引脚是一致的，所以无需更改。

NOR/PSRAM 3 | NOR/PSRAM 4 | User Constants | GPIO Settings

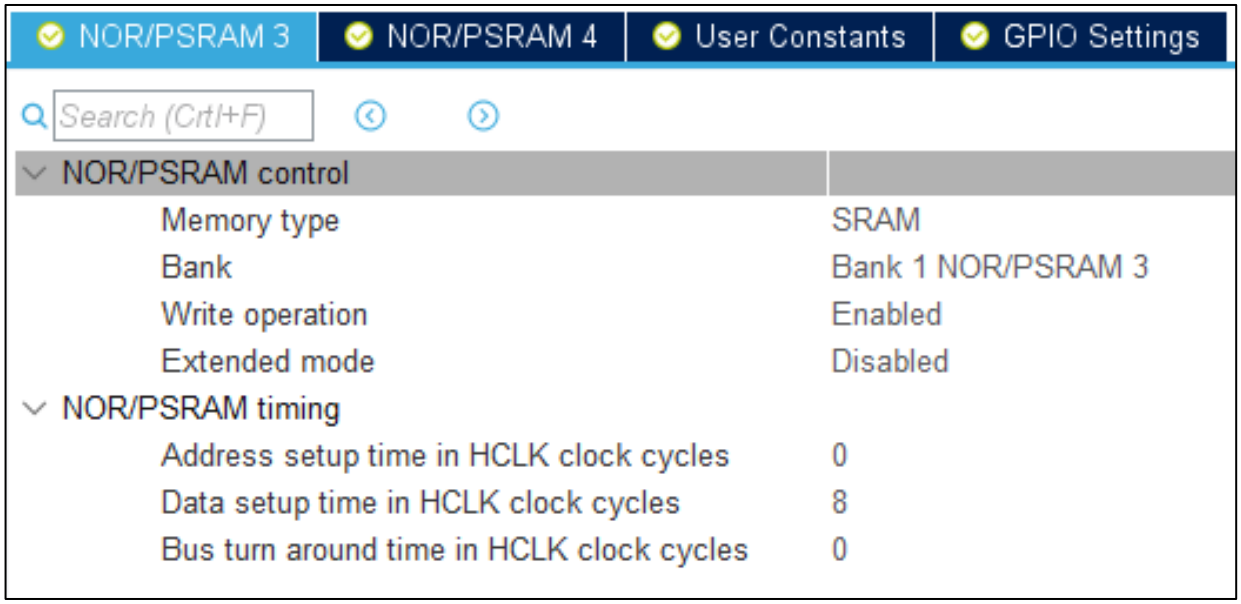
Pin Name	Signal on Pin	GPIO mode	GPIO Pull-up/Pull-down	Maximum output speed
PF0	FSMC_A0	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF1	FSMC_A1	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF2	FSMC_A2	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF3	FSMC_A3	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF4	FSMC_A4	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF5	FSMC_A5	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF12	FSMC_A6	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF13	FSMC_A7	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF14	FSMC_A8	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PF15	FSMC_A9	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PG0	FSMC_A10	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PG1	FSMC_A11	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PG2	FSMC_A12	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PG3	FSMC_A13	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PG4	FSMC_A14	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PG5	FSMC_A15	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PD11	FSMC_A16	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PD12	FSMC_A17	Alternate Function Push Pull	No pull-up and no pull-down	Very High
PD13	FSMC_A18	Alternate Function Push Pull	No pull-up and no pull-down	Very High

2. FSMC的Bank 1子区3参数设置



（1）NOR/PSRAM control组，子区控制参数

- Extended mode设置为Disabled。FSMC自动使用模式A对SRAM进行操作。SRAM的读操作和写操作的速度基本相同，所以读写操作可以使用相同的时序参数，无需使用扩展模式单独设置读时序和写时序。



（2）NOR/PSRAM timing组，读写操作时序参数

- Address setup time in HCLK clock cycles，即地址建立时间参数ADDSET，设置范围0到15
- Data setup time in HCLK clock cycles，即数据建立时间参数DATAST，设置范围1到255
- Bus turn around time in HCLK clock cycles，总线翻转时间，设置范围0到15

19.3.2 程序功能实现

1. 主程序

程序较长，看源程序

主要是显示了一个文字菜单，然后分别作出响应。

[1]KeyUp　= Write by HAL functions

[2]KeyDown = Read by HAL functions

[3]KeyLeft　= Write by pointer

[4]KeyRight= Read by pointer

2. FSMC初始化

进行FSMC外设初始化的函数MX_FSMC_Init()在文件fsmc.c中实现，这个函数是CubeMX自动生成的，对Bank1的子区3和子区4进行初始化。

```
/* 文件: fsmc.c  ------------------------------------------------------------*/
#include "fsmc.h"
SRAM_HandleTypeDef  hsram3;    //Bank1 子区3的外设对象变量，用于外部SRAM
SRAM_HandleTypeDef  hsram4;    //Bank1 子区4的外设对象变量，用于TFT LCD
```

程序较长，看源程序

3. 外部SRAM数据读写

main()函数中响应4个菜单项的4个函数

程序较长，看源程序

[1]KeyUp = Write by HAL functions

[2]KeyDown = Read by HAL functions

[3]KeyLeft = Write by pointer

[4]KeyRight= Read by pointer

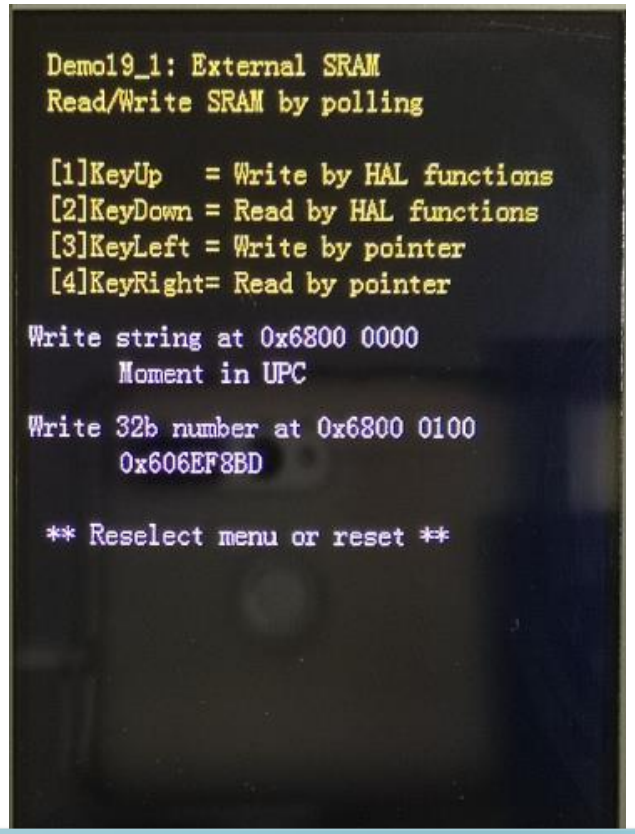
void SRAM_WriteByFunc()

void SRAM_ReadByFunc()

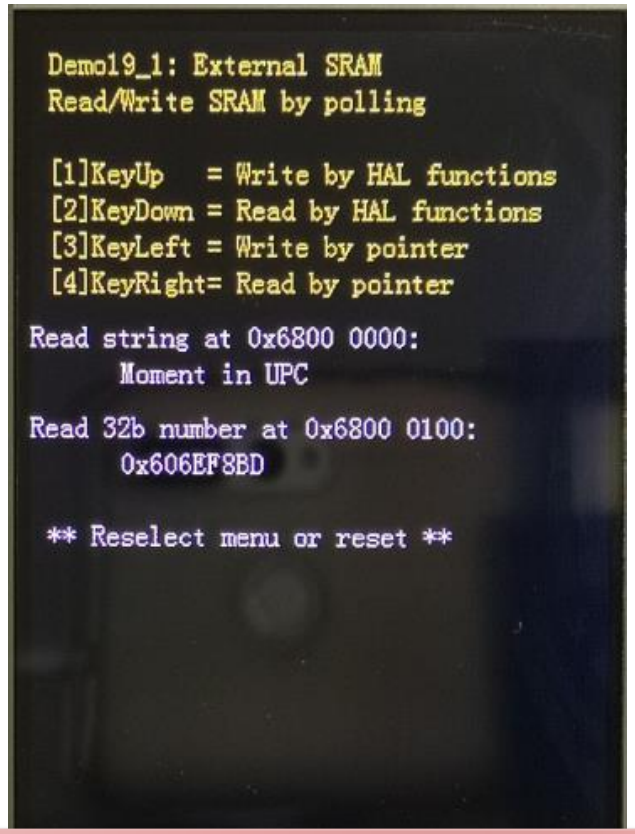
void SRAM_WriteByPointer()

void SRAM_ReadByPointer()

运行测试

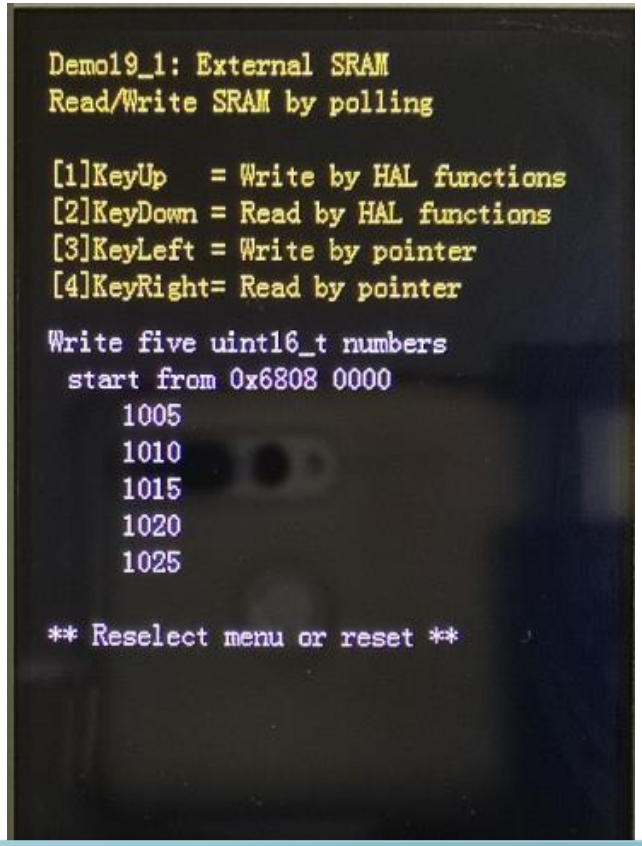


按KeyUp键，调用HAL函数在地址0x6800 0000处写入一个字符串，在地址0x6800 0100处写入一个32位随机数，随机数由硬件RNG产生

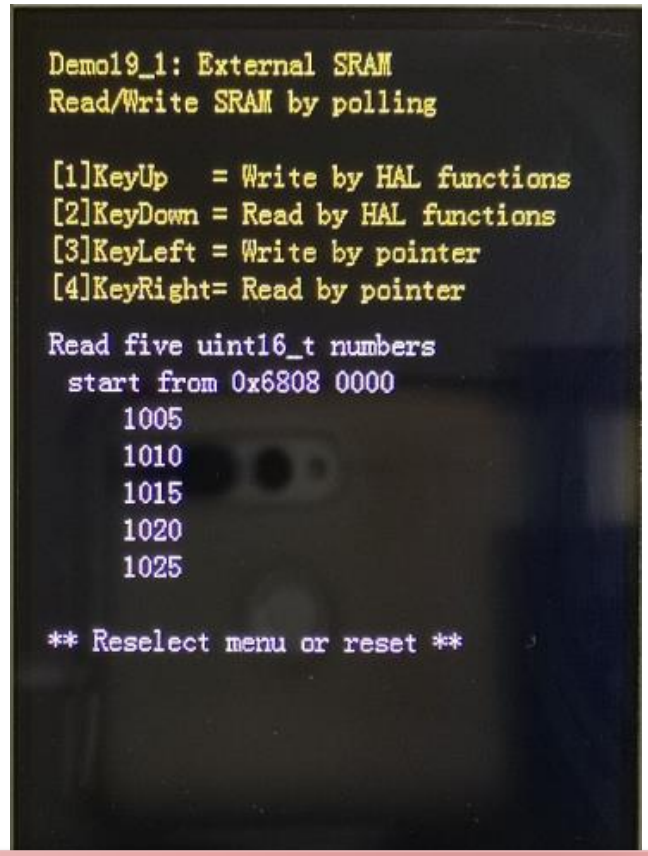


按KeyDown键，调用HAL函数从地址0x6800 0000读取字符串，从地址0x6800 0100读取一个32位的数。读出的与写入的一致，说明读写正确

运行测试



按KeyLeft键，直接使用指针，从地址0x6808 0000开始写入5个uint16_t类型的数



按KeyRight键，直接使用指针，从地址0x6808 0000开始读取5个uint16_t类型的数。读出的数与写入的一致，说明读写操作正确

练习任务

1. 看教材19.4节，实现用DMA方式读写外部SDRAM