应用笔记

Application Note

文档编号：**AN1092**

APM32F4xx_SMC_SRAM 应用笔记

版本：**V 1.0**

1 引言

本应用笔记提供如何在 APM32F4xx 系列上配置和应用 SMC 外设，从而访问外部 SRAM 存储器。

目录

2 APM32F4xx SMC 简介

SMC，全称是 Static Memory Controller ，即静态存储控制器。该外设用于驱动静态存储设备，比如 SRAM、PSRAM、NandFlash、NorFlash、PCCard 。APM32F4xx SMC 内部有四个存储块，每个存储块都对应控制不同类型的存储器，通过配置 SMC 控制寄存器选择不同的存储器类型；任一时刻只能访问一个外部设备；每个存储块都可以单独配置，SMC 控制时序可编程以适用不同的外部存储设备。

2.1 SMC 结构框图

SMC 主要有五个部分组成：AHB 总线接口、配置寄存器、NORFlash 控制器、NANDFlash/PC 卡控制器和外部设备接口信号，具体情况如下图：

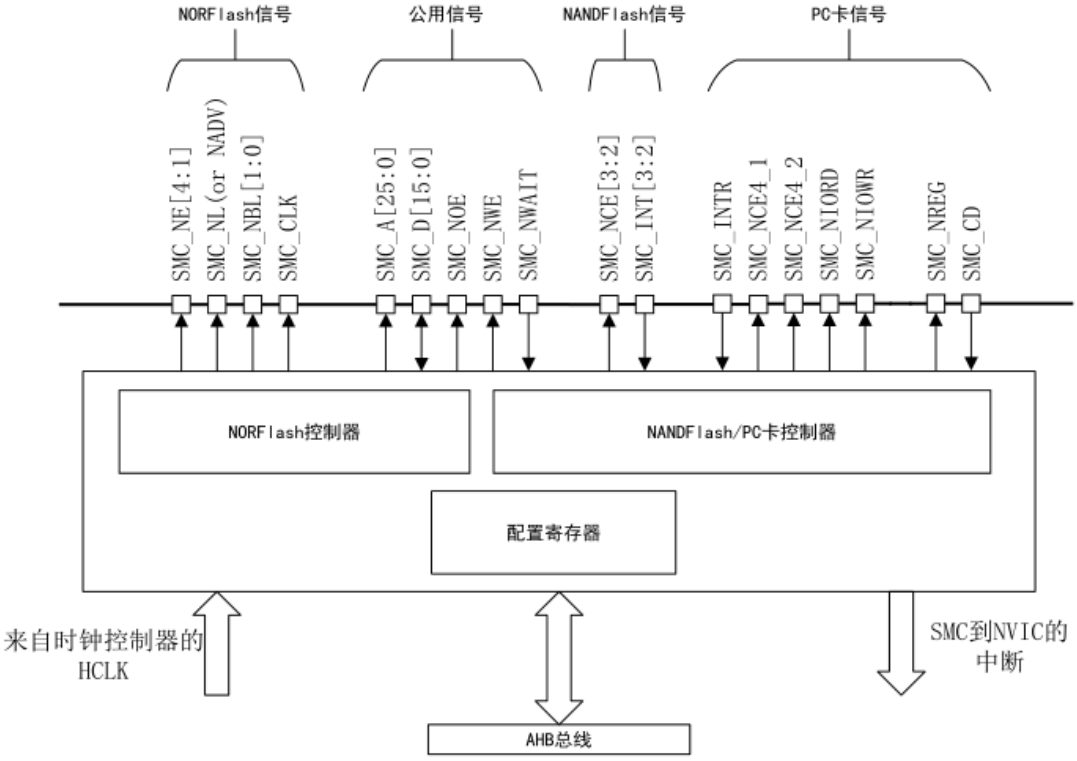

图 1 SMC 结构框图

2.2 SMC 的地址映射

使用 APM32F4xx SMC 外设外接存储器扩展存储空间时，外部的存储器存储空间会映射到 MCU 的内存存储空间，而且不同的外部存储设备对应不同的地址空间，如下表所示：

表格 1 外部存储设备地址映射表

起始地址	结束地址	存储块	支持存储类型
0x60000000	0x6FFFFFFF	存储块 1（4*64M）	NOR/PSRAM
0x70000000	0x7FFFFFFF	存储块 2（4*64M）	NAND
0x80000000	0x8FFFFFFF	存储块 3（4*64M）	NAND
0x90000000	0x9FFFFFFF	存储块 4（4*64M）	PC 卡

可以看到，SMC 把内部 1GB 的存储空间划分为了 4 个 256MB 的存储块，每个存储块有各自的地址空间以及适用的存储器类型，存储块 1 用于控制 NOR/PSRAM 存储器，而存储块 2、3、4 用于控制 NAND 闪存和 PC 卡。

此外，对于每个存储块的内部，又划分为 4 块 64MB 同等大小的区域，每个小块都有对应的控制引脚用于连接外部存储器的片选信号，从而决定外部存储器具体连接到的是哪个 64MB 的存储区域。比如对于存储块 1 ，有对应的控制信号线 SMC_NE[4 : 1] ，用于选择存储块 1 内部的 4 个 64MB 的地址空间，具体地址分配如下：

表格 2 存储块 1 地址映射表

存储块 1 内部区域块	片选信号	起始地址	结束地址
区域块 1	SMC_NE1	0x60000000	0x63FFFFFF
区域块 2	SMC_NE2	0x64000000	0x67FFFFFF
区域块 3	SMC_NE3	0x68000000	0x6BFFFFFF
区域块 4	SMC_NE4	0x6C000000	0x6FFFFFFF

3 SRAM 简介

3.1 SRAM 存储结构

SRAM，全称是 Static Random Access Memory，它是一种静态随机存取存储器，主要应用于 CPU 内部的高速缓存（Cache）或者集成在 MCU 内部作为数据存储器。SRAM 的所谓的“静态”，指的就是只要保持在通电状态，存储在其内部的数据就可以一直保存，一般是和 DRAM ，即动态存储器相比较而言的。

SRAM 的存储单元结构以锁存器来存储数据，存储单元电路如下：

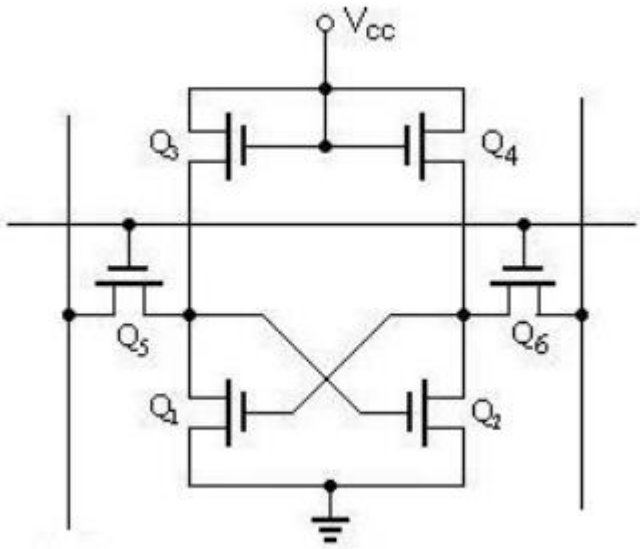

图 2 SRAM 存储单元电路

3.2 APM32F4xx SMC 与 SRAM 的信号引脚连接

尽管市面上有各种不同厂商、不同型号的 SRAM 芯片，但是 SRAM 芯片的引脚信号基本都是一致的，有地址线、数据线、控制线三种类型。

下面以常见的 SRAM 芯片型号 IS62WV12816 ，给出该 SRAM 芯片与 APM32 SMC 外设的信号引脚连接关系：

表 3 APM32F4xx SMC 外设与外部 SRAM 存储器信号引脚对应关系

APM32 SMC 信号引脚	SRAM 信号引脚	APM32 端口或引脚	信号说明
A0 – A18	A0 – A18	GPIOD/GPIOF/GPIOG	地址信号
D0 – D15	D0 – D15	GPIOD/GPIOE	数据信号
SMC_NE[1 : 4]	CS#	PD7/PG9/PG10/PG12	片选信号
SMC_NOE	OE#	PD4	读使能信号
SMC_NWE	WE#	PD5	写使能信号
SMC_NLB0	LB#	PE0	数据低字节掩码信号
SMC_NLB1	UB#	PE1	数据高字节掩码信号

对于数据位宽为 8bit 的 SRAM 型号，高位数据线 D8-D15 可以不用连接。

4 APM32F4xx SMC 外设初始化参数说明

通过 APM32 SMC 驱动外部 SRAM 芯片时，需要配置 SMC 的时序寄存器以及控制寄存器，但是 APM32F4xx 的固件库中，对 SMC 相关的操作已经封装好了，有两个结构体是比较重要的，分别是时序结构体和初始化配置结构体，下面对这些结构体成员变量作一些解释及配置说明。

4.1 时序结构体

时序结构体的代码如下：

```
/**
 * @brief Timing parameters for NOR/SRAM Banks
 */
typedef struct
{
    uint8_t          addressSetupTime;
    uint8_t          addressHodeTime;
    uint8_t          dataSetupTime;
    uint8_t          busTurnaroundTime;
    uint8_t          clockDivision;
    uint8_t          dataLatency;
    SMC_ACCESS_MODE_T accessMode;
} SMC_NORSRAMTimingConfig_T;
```

该结构体的成员都是 NorFlash/SRAM 读写过程中与时间相关的各项参数，对这些结构体成员说明如下：

(1) addressSetupTime ：本成员用于设置地址建立时间，可设置为 0-15 个 HCLK 时钟周期数。APM32F4xx 的 HCLK 时钟频率是 168MHz，那么一个 HCLK 时钟周期就是 1/168 微秒。

(2) addressHodeTime ：本成员用于设置地址保持时间，同样也可以设置为 0-15 个 HCLK 时钟周期数。

(3) dataSetupTime ：本成员用于设置数据建立时间，可设置为 0-15 个 HCLK 时钟周期数。

(4) busTurnaroundTime ：本成员用于设置总线转换周期，只适用于总线复用模式的 NorFlash 存储器。

(5) clockDivision ：本成员用于配置时钟分频系数，可以把 HCLK 时钟作为输入源，配置为 0-16 分频，用于需要时钟同步的存储器。

(6) dataLatency ：本成员设置数据保持时间，用于配置在读取第一个数据前

等待的存储器周期数目，可设置为 0-15 个时钟周期。该成员仅适用于同步突发模式的 NorFlash 操作。

(7) accessMode ：本成员配置访问存储器的模式，可配置为 A/B/C/D 模式，不同的模式下 SMC 输出的时序不同。对应 SRAM 来说，一般使用模式 A 即可。

对于时序结构体所有与时间相关的参数，单位都是 HCLK 个时钟周期，APM32F4xx 的 HCLK 时钟为 168MHz ，那么 1 个 HCLK 时钟周期就是 1/168 us ，即约等于 6ns.

4.2 初始化结构体

初始化配置结构体的代码如下：

```
/**
 * @brief SMC NOR/SRAM Config structure
 */
typedef struct
{
    SMC_BANK1_NORSRAM_T         bank;
    SMC_DATA_ADDRESS_MUX_T      dataAddressMux;
    SMC_MEMORY_TYPE_T           memoryType;
    SMC_MEMORY_DATA_WIDTH_T     memoryDataWidth;
    SMC_BURST_ACCESS_MODE_T     burstAcceesMode;
    SMC_ASYNCHRONOUS_WAIT_T     asynchronousWait;
    SMC_WAIT_SIGNAL_POLARITY_T waitSignalPolarity;
    SMC_WRAP_MODE_T             wrapMode;
    SMC_WAIT_SIGNAL_ACTIVE_T    waitSignalActive;
    SMC_WRITE_OPERATION_T       writeOperation;
    SMC_WAITE_SIGNAL_T          waiteSignal;
    SMC_EXTENDEN_MODE_T         extendedMode;
    SMC_WRITE_BURST_T           writeBurst;
    SMC_NORSRAMTimingConfig_T* readWriteTimingStruct;
    SMC_NORSRAMTimingConfig_T* writeTimingStruct;
} SMC_NORSRAMConfig_T;
```

该结构体主要是在配置 SMC 初始化时所需要的参数，最底层的代码其实就是配置 SMC 的片选控制、片选时序、写时序这 3 个寄存器，下面简单介绍下这些结构体成员：

(1) bank：选择 SRAM 类型的哪个存储区域，该参数需要根据片选引脚的连接不同进行选择，前面也有介绍不同的片选信号对应的是哪个存储区域。

(2) dataAddressMux ：用于设置地址线与数据线是否进行复用。对于 NorFlash 存储器类型，地址线与数据线是可以分时复用的，这样可以减少 GPIO 引脚的使用，该参数仅对 NorFlash 有效。

(3) memoryType ：配置存储器类型，可以配置的存储器类型有：SRAM、PSRAM、NorFlash。

(4) memoryDataWidth ：配置控制存储器的数据位宽，可配置为 8 位或者 16 位。

(5) burstAcceesMode ：本成员用于配置是否使用突发访问模式。突发访问模式是指发送一个地址后连续访问多个数据，非突发模式下每访问一个数据都需要输入一个地址。该成员只对同步模式下才有效。

(6) asynchronousWait ：用于配置是否使能同步等待信号，对于同步类型的 NorFlash 或者 PSRAM 存储器，可以使用 SMC_NWAIT 引脚插入等待状态。

(7) waitSignalPolarity ：该成员用于配置等待信号极性，可设置要求等待的信号是高电平还是低电平。

(8) wrapMode ：配置是否使能非对齐突发模式，仅在突发模式下才有效。

(9) waitSignalActive ：用于配置等待信号是在等待之前有效还是等待期间有效，只在突发模式下有效。

(10) writeOperation ：用于配置是否开启写使能，如果禁止写使能，那么 SMC 外设对存储器只能进行读操作，写操作此时是禁止的。

(11) waiteSignal ：该成员用于设置是否使能 NWAIT 信号插入等待状态，只对突发模式下有效。

(12) extendedMode ：该成员用于设置是否使能扩展模式。当使能扩展模式后，SMC 的访问模式可以选择模式 A 至模式 D，否则只能选择模式 1 或者模式 2。

(13) writeBurst ：用于配置是否使能写突发操作，只在突发模式下有效。

(14) readWriteTimingStruct ：读写时序结构体。在不使能扩展模式时，该结构体可以同时配置读写时序，也就是说读写时序是一样的。

(15) writeTimingStruct：写时序结构体，在使能扩展模式时，可以配置 SMC_WRTTIM 寄存器，从而实现读写使用不同的时序。

4.3 APM32F4xx SMC 控制 SRAM 存储器时序计算

APM32F4xx SMC 要控制外部 SRAM 存储器，必须要正确配置时序结构体中的两个时序参数：地址建立时间和数据建立时间，其他的时序参数没有用到，设置为 0 即可。

下面以 SRAM 芯片型号 IS62WV12816 为例，介绍 APM32F4xx SMC 控制 SRAM 芯片的时序计算。根据 IS62WV12816 芯片手册，其关键的时序参数

如下：

表 4 IS62WV12816 SRAM 芯片读写时序参数要求

时间参数	参数含义	时间要求
tWC	Write Cycle Time，写周期时间	不小于 55ns
tPWE	WE Pulse Width，写使能脉冲宽度，就是 SMC 发出写使能信号后，到输入 SRAM 的数据有效的时间	不小于 40ns
tRC	Read Cycle Time，读周期时间	不小于 55ns
tSA	Address Setup Time，地址建立时间	0 ns，即不作要求

APM32F4xx SMC 控制外部 SRAM 的读写时序有多种不同的模式，下面展示的是模式 A 的读写时序，如下图所示：

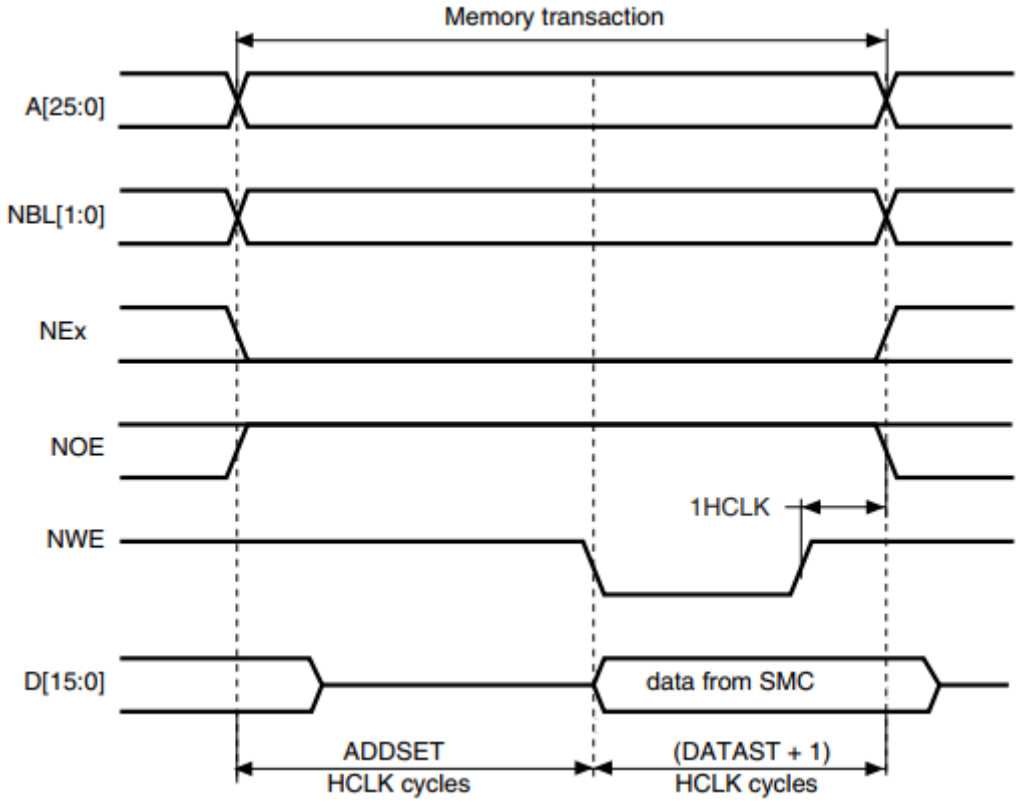

图 3 SMC 模式 A 写时序

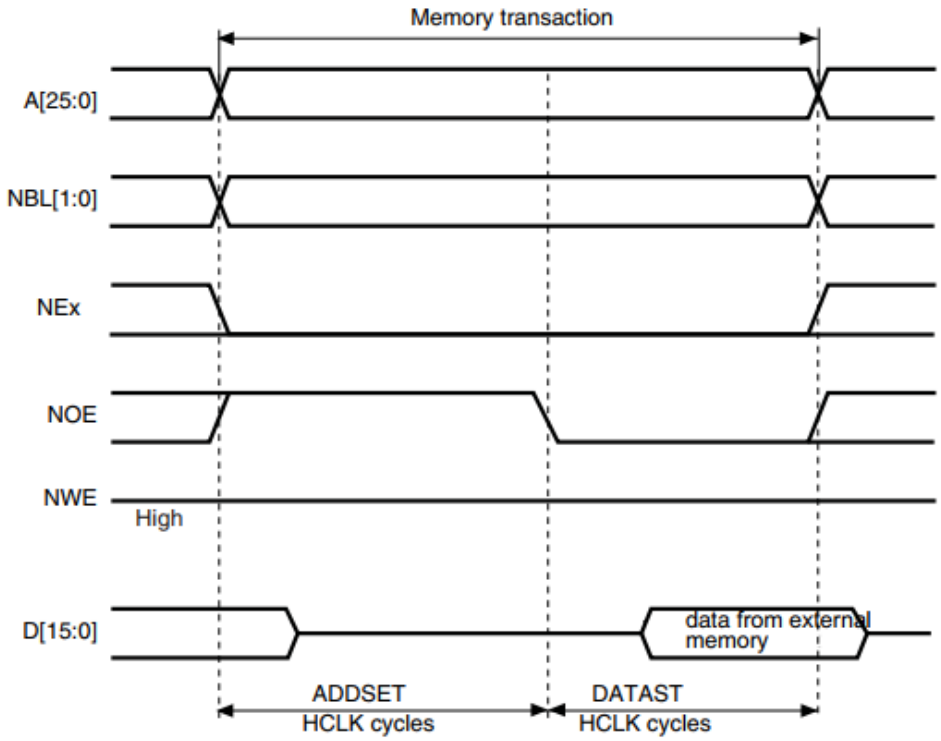

图 4 SMC 模式 A 读时序

根据 IS62WV12816 芯片的时间参数要求，以及 SMC 读写 SRAM 的时序图，可以得到 IS62WV12816 芯片的时间参数计算公式如下：

tWC = ADDSET + DATASET + 1 >= 55ns

tPWE = DATASET >= 40ns

tRC = ADDSET + DATASET >= 55ns

对时序结构体进行赋值的 ADDSET 和 DATASET 参数，它的单位都是 1 个 HCLK 周期，而 APM32F4xx 的 HCLK 时钟频率是 168MHz ，那么 1 个 HCLK 周期就是 1/168 us = 6ns .

根据前面的 IS62WV12816 关键的时间参数表格可知，地址建立时间是 0 ，所以时序结构体的 ADDSET 参数赋值为 0 即可；然后根据上面的三条表达式，为了同时满足读写时序的要求，可得出 DATASET 参数赋值为 10 ，这样就可以满足 SRAM 读写的时间要求了。

5 APM32F4xx SMC 读写外部 SRAM 例程

5.1 硬件设计

外部 SRAM 芯片使用的是 IS62WV12816 型号，SRAM 芯片和 APM32F4xx 连接的原理图如下：

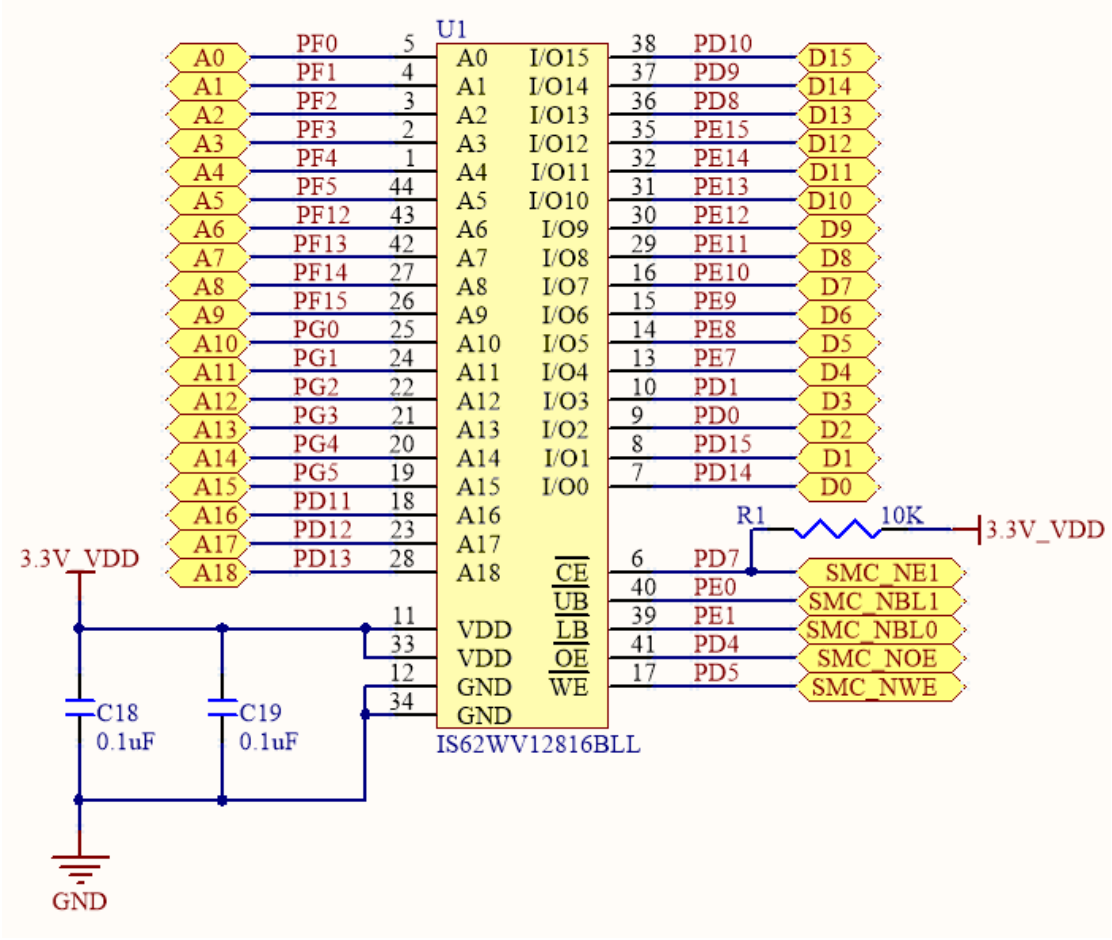

图 5 外部 SRAM 存储器与 APM32F4xx 连接原理图

可以看到，SRAM 芯片有很多的信号线需要与 APM32F4xx 芯片引脚相连，其中 SRAM 芯片的大部分信号线都是与 MCU 连接固定的引脚的，只有片选引脚可以让我们选择连接 SMC_NE1 - NE4，而片选引脚的连接不同，会映射到 MCU 不同的地址空间。

上面的原理图片选引脚连接的是 SMC_NE1，那么 SRAM 的存储空间会被映射到 MCU 内部的 0x60000000 ~ 0x63FFFFFF 这一片地址空间上。对于 IS62WV12816 型号，其存储空间大小是 256KB，那么当 MCU 访问 0x60000000 开始的 256KB 的内存空间时，SMC 外设会根据初始化的配置，产生相应的访问时序，从而实现外部 SRAM 的读写操作。

5.2 软件设计

要通过 APM32F4xx SMC 实现外部 SRAM 存储器的访问，只要配置使用到的 SMC 外设相关的 GPIO 引脚以及配置好 SMC 的时序结构体和初始化结构体即可。代码实现流程图如下：

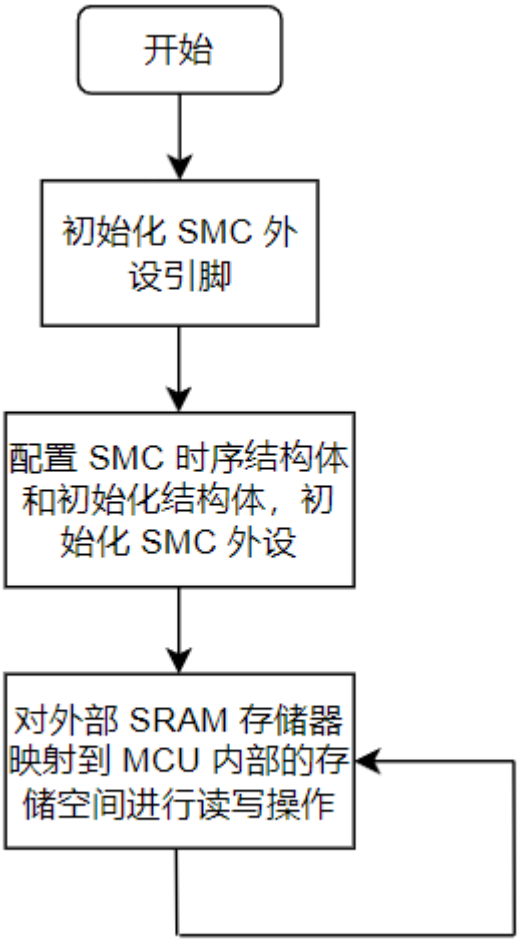

图 6 SMC 访问外部 SRAM 存储器程序流程图

5.2.1 SMC GPIO 引脚配置

对 SMC 外设所使用到的 GPIO 引脚，全部都配置为复用功能输出模式，下面以 PD0 引脚为例作为配置参考，其他所有引脚的配置都是类似的，参考代码如下：

```
GPIO_Config_T gpioConfig;

/* Enable GPIO Clock */
RCM_EnableAHB1PeriphClock(RCM_AHB1_PERIPH_GPIOD);

/* SMC SRAM GPIO Config */
gpioConfig.speed = GPIO_SPEED_100MHz;
gpioConfig.mode = GPIO_MODE_AF;
gpioConfig.otype = GPIO_OTYPE_PP;
gpioConfig.pupd = GPIO_PUPD_UP;
gpioConfig.pin = GPIO_PIN_0;
GPIO_Config(GPIOD, &gpioConfig);

GPIO_ConfigPinAF(GPIOD, GPIO_PIN_SOURCE_0, GPIO_AF_FSMC);
```

5.2.2 SMC 初始化配置

APM32F4xx 固件库已经对 SMC 外设对时序控制以及初始化配置相关的参数封装在时序结构体和初始化配置结构体了，SMC 外设的初始化配置，主要就是对这两个结构体成员变量进行赋值，最后调用固件库函数提供的初始化配置函数实现 SMC 外设的初始化配置，具体代码如下：

```
void SMC_SRAM_Init(void)
{
    SMC_NORSRAMConfig_T SMC_NORSRAM_ConfigStruct;
    SMC_NORSRAMTimingConfig_T readWriteTimingStruct;

    /* SMC SRAM GPIO Config */
    SMC_SRAM_GPIOConfig();
    /* Enable SMC Clock */
    RCM_EnableAHB3PeriphClock(RCM_AHB3_PERIPH_EMMC);

    /* SMC SRAM Timing Config */
    readWriteTimingStruct.addressSetupTime  = 0x00;
    readWriteTimingStruct.addressHodeTime   = 0x00;
    readWriteTimingStruct.dataSetupTime     = 0x0A;
    readWriteTimingStruct.busTurnaroundTime = 0x00;
    readWriteTimingStruct.clockDivision     = 0x00;
    readWriteTimingStruct.dataLatency       = 0x00;
    readWriteTimingStruct.accessMode        = SMC_ACCESS_MODE_A;

    /* SMC SRAM Init Config */
    SMC_NORSRAM_ConfigStruct.bank = SMC_BANK1_NORSRAM_1;
    SMC_NORSRAM_ConfigStruct.dataAddressMux = SMC_DATA_ADDRESS_MUX_DISABLE;
    SMC_NORSRAM_ConfigStruct.memoryType = SMC_MEMORY_TYPE_SRAM;
    SMC_NORSRAM_ConfigStruct.memoryDataWidth = SMC_MEMORY_DATA_WIDTH_16BIT;
    SMC_NORSRAM_ConfigStruct.burstAcceesMode = \
    SMC_BURST_ACCESS_MODE_DISABLE;
    SMC_NORSRAM_ConfigStruct.asynchronousWait = \
    SMC_ASYNCHRONOUS_WAIT_DISABLE;
    SMC_NORSRAM_ConfigStruct.waitSignalPolarity = \
    SMC_WAIT_SIGNAL_POLARITY_LOW;
    SMC_NORSRAM_ConfigStruct.wrapMode = SMC_WRAP_MODE_DISABLE;
    SMC_NORSRAM_ConfigStruct.waitSignalActive = \
    SMC_WAIT_SIGNAL_ACTIVE_BEFORE_WAIT_STATE;
    SMC_NORSRAM_ConfigStruct.writeOperation = SMC_WRITE_OPERATION_ENABLE;
    SMC_NORSRAM_ConfigStruct.waiteSignal = SMC_WAITE_SIGNAL_DISABLE;
    SMC_NORSRAM_ConfigStruct.extendedMode = SMC_EXTENDEN_MODE_ENABLE;
    SMC_NORSRAM_ConfigStruct.writeBurst = SMC_WRITE_BURST_DISABLE;
    SMC_NORSRAM_ConfigStruct.readWriteTimingStruct = \
    &readWriteTimingStruct;
    SMC_NORSRAM_ConfigStruct.writeTimingStruct = &readWriteTimingStruct;

    SMC_ConfigNORSRAM(&SMC_NORSRAM_ConfigStruct);
}
```

对于时序结构体 SMC_NORSRAMTimingConfig_T 相关的参数配置，需要根据 SRAM 的芯片手册提供的时间参数要求，计算出对应结构体成员的值，从而对各成员参数进行合理配置，本例程所使用的 SRAM 芯片型号是 IS62WV12816。

5.2.3 SMC 访问外部 SRAM 存储器

通过原理图可以确定，外部 SRAM 芯片的片选引脚连接到的是 SMC_NE1 引脚，那么外部 SRAM 的存储空间会映射到 MCU 内部的 0x60000000 ~ 0x63FFFFFF 范围内的地址空间。对于本例程所使用的 IS62WV12816 型号，其存储空间大小是 256KB ，所以我们访问外部 SRAM 的起始地址是 0x60000000 ，结束地址是 0x60040000 。

只要初始化了 SMC 外设之后，对外部 SRAM 的读写访问就是对内存的读写操作。对于 C 语言的来说，通过指针就可以实现对特定内存地址进行读写操作，例如：

对 SRAM 的 0x6000000 地址，读取 1 个字节数据：

```
uint8_t data = *(uint8_t *)0x6000000;
```

对 SRAM 的 0x6000000 地址，写入 1 个字节数据：

```
*(uint8_t *)0x6000000 = (uint8_t)0x55;
```

如果我们想要定义变量在指定的内存地址上，可以使用对应编译器提供的扩展关键字来修饰变量，比如对于 ARM-CC 编译器变量定义如下：

```
uint8_t data[256 * 1024] __attribute__((at(0x60000000)));
```

6 版本历史

表 5 文档版本历史记录

日期	版本	变更历史
2023.06.05	1.0	初稿

声明

本手册由珠海极海半导体有限公司（以下简称“极海”）制订并发布，所列内容均受商标、著作权、软件著作权相关法律法规保护，极海保留随时更正、修改本手册的权利。使用极海产品前请仔细阅读本手册，一旦使用产品则表明您（以下称“用户”）已知悉并接受本手册的所有内容。用户必须按照相关法律法规和本手册的要求使用极海产品。

1、权利所有

本手册仅应当被用于与极海所提供的对应型号的芯片产品、软件产品搭配使用，未经极海许可，任何单位或个人均不得以任何理由或方式对本手册的全部或部分内容进行复制、抄录、修改、编辑或传播。

本手册中所列带有“®”或“TM”的“极海”或“Geehy”字样或图形均为极海的商标，其他在极海产品上显示的产品或服务名称均为其各自所有者的财产。

2、无知识产权许可

极海拥有本手册所涉及的全部权利、所有权及知识产权。

极海不应因销售、分发极海产品及本手册而被视为将任何知识产权的许可或权利明示或默示地授予用户。

如果本手册中涉及任何第三方的产品、服务或知识产权，不应被视为极海授权用户使用前述第三方产品、服务或知识产权，除非在极海销售订单或销售合同中另有约定。

3、版本更新

用户在下单购买极海产品时可获取相应产品的最新版的手册。

如果本手册中所述的内容与极海产品不一致的，应以极海销售订单或销售合同中的约定为准。

4、信息可靠性

本手册相关数据经极海实验室或合作的第三方测试机构批量测试获得，但本手册相关数据难免会出现校正笔误或因测试环境差异所导致的误差，因此用户应当理解，极海对本手册中可能出现的该等错误无需承担任何责任。本手册相关数据仅用于指导用户作为性能参数参照，不构成极海对任何产品性能方面的保证。

用户应根据自身需求选择合适的极海产品，并对极海产品的应用适用性进行有效验证和测试，以确认极海产品满足用户自身的需求、相应标准、安全或其它可靠性要求；若因用户未充分对极海产品进行有效验证和测试而致使用户损失的，极海不承担任何责任。

5、合规要求

用户在使用本手册及所搭配的极海产品时，应遵守当地所适用的所有法律法规。用户应了解产品可能受到产品供应商、极海、极海经销商及用户所在地等各国有关出口、再出口或其它法律的限制，用户（代表其本身、子公司及关联企业）应同意并保证遵守所有关于取得极海产品及／或技术与直接产品的出口和再出口适用法律与法规。

6、免责声明

本手册由极海“按原样”（as is）提供，在适用法律所允许的范围内，极海不提供任何形式的明示或暗示担保，包括但不限于对产品适销性和特定用途适用性的担保。

对于用户后续在针对极海产品进行设计、使用的过程中所引起的任何纠纷，极海概不承担责任。

7、责任限制

在任何情况下，除非适用法律要求或书面同意，否则极海和/或以“按原样”形式提供本手册的任何第三方均不承担损害赔偿责任，包括任何一般、特殊因使用或无法使用本手册相关信息而产生的直接、间接或附带损害（包括但不限于数据丢失或数据不准确，或用户或第三方遭受的损失）。

8、适用范围

本手册的信息用以取代本手册所有早期版本所提供的信息。