# Application Note

## Application Note

**Document No.: AN1092**

**APM32F4xx_SMC_SRAM Application Note**

**Version: V1.0**

# 1 Introduction

This application note provides a guide on how to configure and apply SMC peripheral on APM32F4xx series to access external SDRAM memory.

# Contents

# 2 Introduction to APM32F4xx SMC

The full name of SMC is Static Memory Controller. This peripheral is used to drive static storage devices such as SRAM, PSRAM, NandFlash, NorFlash, and PCCard. The APM32F4xx SMC has four internal storage blocks, each of which controls different types of memory. Different memory types are selected by configuring SMC control register. Only one external device can be accessed at any time; each storage block can be configured separately, and the SMC control timing can be programmed for different external storage devices.

## 2.1 SMC Structure Block Diagram

SMC consists of five parts: AHB bus interface, configuration register, NORFlash controller, NANDFlash/PC card controller and external device interface signal, specifically as shown in the figure below:

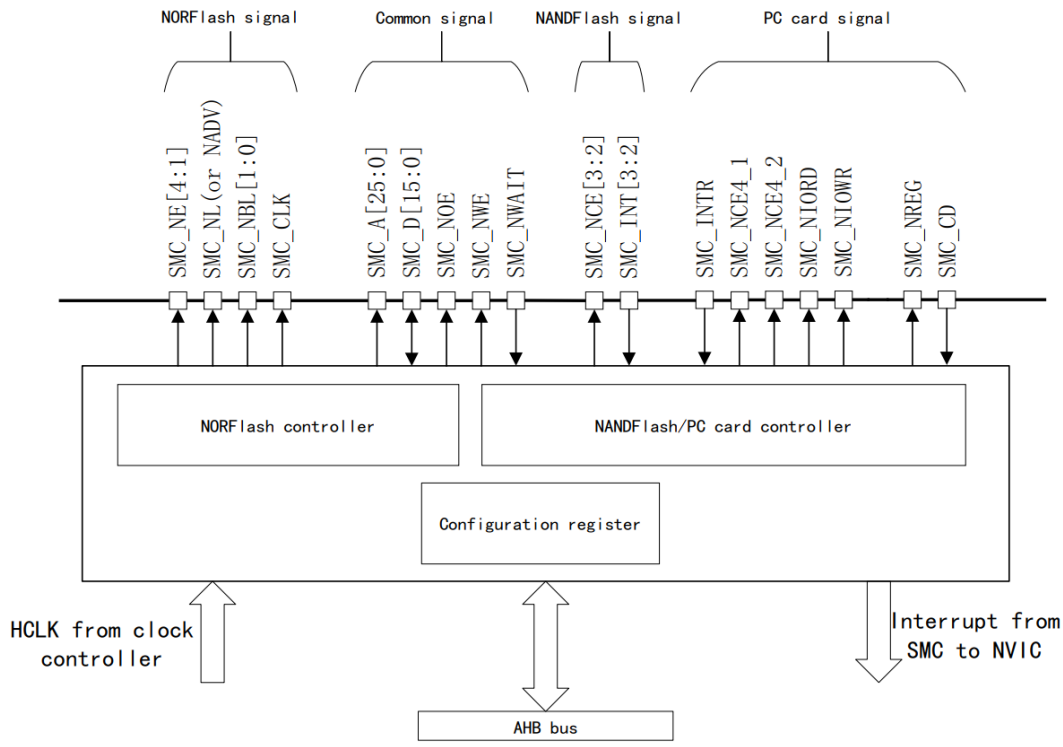


Figure1 SMC Structure Block Diagram

## 2.2 Address mapping of SMC

When APM32F4xx SMC peripheral is used to connect an external memory to expand the storage space, the storage space of the external memory will be mapped to the memory storage space of the MCU, and different external storage devices correspond to different address spaces, as shown in the

following table:

Table 1 External Storage Device Address Mapping Table

| Start address | End address | Storage block | Storage types supported |
|---|---|---|---|
| 0x60000000 | 0x6FFFFFFF | Storage block 1 (4*64MB) | NOR/PSRAM |
| 0x70000000 | 0x7FFFFFFF | Storage block 2 (4*64MB) | NAND |
| 0x80000000 | 0x8FFFFFFF | Storage block 3 (4*64MB) | NAND |
| 0x90000000 | 0x9FFFFFFF | Storage block 4 (4*64MB) | PC card |

It can be seen that, SMC divides its internal 1GB storage space into four 256MB storage blocks, each of which has its own address space and applicable memory type. The storage block 1 is used to control NOR/PSRAM memory, and the storage blocks 2, 3, and 4 are used to control NAND flash memory and PC card.

In addition, each storage block is internally divided into four 64MB areas with the same size, each of which has corresponding control pin to connect the chip selection signals of the external memory, so as to determine which 64MB storage area the external memory is specifically connected to. For example, the storage block 1 has the corresponding control signal line SMC_NE[4 : 1], which is used to select four 64MB address spaces within the storage block 1. The specific address allocation is as follows:

Table 2 Address Mapping Table of Storage Block 1

| Internal area block of storage block 1 | Chip selection signal | Start address | End address |
|---|---|---|---|
| Area block 1 | SMC_NE1 | 0x60000000 | 0x63FFFFFF |
| Area block 2 | SMC_NE2 | 0x64000000 | 0x67FFFFFF |
| Area block 3 | SMC_NE3 | 0x68000000 | 0x6BFFFFFF |
| Area block 4 | SMC_NE4 | 0x6C000000 | 0x6FFFFFFF |

# 3 Introduction to SRAM

## 3.1 SRAM storage structure

SRAM, the full name of which is Static Random Access Memory, is a kind of static random access memory, and is mainly used for the internal cache of CPU or is integrated in MCU as a data memory. The so-called "static" of SRAM means that as long as it remains powered on, the data stored in it can be saved all along, generally compared to DRAM, namely dynamic memory.

The storage unit structure of SRAM uses a latch to store data, and the storage unit circuit is as follows:

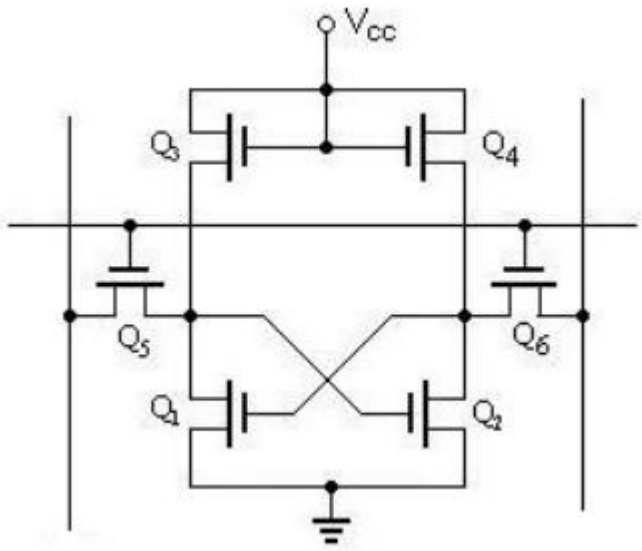


Figure 2 SRAM Storage Unit Circuit

## 3.2 Signal pin connection between APM32F4xx SMC and SRAM

Although there are different manufacturers and models of SRAM chips on the market, the pin signals of SRAM chips are basically the same, which all include three types i.e. address line, data line, and control line.

Shown below is the signal pins connection relationship between the common SRAM chip IS62WV12816 and the APM32 SMC peripheral:

Table 3 Signal Pins Correspondence between APM32F4xx SMC Peripheral and External SRAM Memory

| APM32 SMC signal pin | SRAM signal pin | APM32 port or pin | Signal description |
|---|---|---|---|
| A0 – A18 | A0 – A18 | GPIOD/GPIOF/GPIOG | Address signal |
| D0 – D15 | D0 – D15 | GPIOD/GPIOE | Data signal |
| SMC_NE[1 : 4] | CS# | PD7/PG9/PG10/PG12 | Chip selection signal |
| SMC_NOE | OE# | PD4 | Read enable signal |
| SMC_NWE | WE# | PD5 | Write enable signal |
| SMC_NLB0 | LB# | PE0 | Low-byte mask signal of data |
| SMC_NLB1 | UB# | PE1 | High-byte mask signal of data |

For SRAM models with a data bit width of 8 bits, the high-bit data line D8-D15 may not be connected.

# 4 Initialization Parameter Description for APM32F4xx SMC Peripheral

When driving an external SRAM chip through the APM32 SMC, the timing register and control register of the SMC need to be configured. However, in the firmware library of APM32F4xx, SMC related operations have been encapsulated. There are two important structures, namely the timing structure and the initialization configuration structure. Below are some explanations and configuration description of member variables of these structures.

## 4.1 Timing structure

The code of the timing structure is as follows:

```
/**
 * @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;
```

The members of this structure are all time-related parameters in the NorFlash/SRAM read and write process. These structure members are described as follows:

(1) addressSetupTime: This member is used to set the address setup time, which can be set to 0-15 HCLK clock cycles. The HCLK clock frequency of APM32F4xx is 168MHz, so a HCLK clock cycle is 1/168 microseconds.

(2) addressHodeTime: This member is used to set the address hold time, which can also be set to 0-15 HCLK clock cycles.

(3) dataSetupTime: This member is used to set the data setup time, which can be set to 0-15 HCLK clock cycles.

(4) busTurnaroundTime: This member is used to set the bus transition cycle and

is only applicable to the NorFlash memory in bus multiplexed mode.

(5) clockDivision: This member is used to configure the clock division factor, which can use the HCLK clock as the input source and be configured to 0-16 divisions for the memory that requires clock synchronization.

(6) dataLatency: This member is used to set the data hold time to configure the number of memory cycles to wait before reading the first data, which can be set to 0-15 clock cycles. This member is only applicable to the NorFlash operations in synchronous burst mode.

(7) accessMode: This member configures the mode of accessing memory, which can be configured to A/B/C/D mode. The timing of SMC output varies in different modes. For SRAM, the Mode A is generally used.

All time-related parameters of the timing structure are in HCLK clock cycles. If the HCLK clock of APM32F4xx is 168MHz, 1 HCLK clock cycle is 1/168 us, which is approximately equal to 6ns.

## 4.2 Initialization structure

The code of the initialization configuration structure is as follows:

```
/**
 * @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;
```

This structure is mainly used to configure the parameters required for SMC initialization. The underlying code is actually used to configure the chip selection control, chip selection timing, and write timing registers of SMC. These structure members are briefly introduced below:

(1) bank: This parameter is used to select the storage area of the SRAM type according to the different connections of chip selection pins. The storage areas corresponding to different chip selection signals have also been introduced above.

(2) dataAddressMux: Used to set whether to multiplex the address line and data line. For the NorFlash memory type, the address line and data line allow time division multiplexing, which can reduce the use of GPIO pins. This parameter is only valid for NorFlash.

(3) memoryType: It is used to configure the memory type, which can be configured as SRAM, PSRAM, and NorFlash.

(4) memoryDataWidth: It is used to configure the data bit width of the control memory, which can be configured as 8 or 16 bits.

(5) BurstAccessMode: This member is used to configure whether to use the burst access mode. The burst access mode refers to continuous access to multiple data after transmitting an address, while in non-burst mode, an address needs to be entered for access to each data. This member is only valid in synchronous mode.

(6) asynchronousWait: It is used to configure whether to enable synchronous wait signals. For synchronous type of NorFlash or PSRAM memory, SMC_NWAIT pin can be used to insert the wait state.

(7) waitSignalPolarity: This member is used to configure the polarity of the wait signal and can set the signal required to wait to high or low.

(8) wrapMode: It is used to configure whether to enable non-aligned burst mode, which is only valid in burst mode.

(9) waitSignalActive: It is used to configure that the wait signal is valid before or during the wait period, which is only valid in burst mode.

(10) writeOperation: It is used to configure whether to enable write enable. If write enable is disabled, the SMC peripheral can only read the memory, and write operation is disabled.

(11) waiteSignal: This member is used to set whether to enable the NWAIT signal to insert the wait state, which is only valid in burst mode.

(12) extendedMode: This member is used to set whether to enable extension mode. When the extension mode is enabled, the access mode of SMC can be selected from Mode A to Mode D; otherwise only Mode 1 or Mode 2 can be

selected.

(13) WriteBurst: It is used to configure whether to enable write burst operation, which is only valid in burst mode.

(14) readWriteTimingStructure: Read-write timing structure. When the extension mode is not enabled, the structure can configure both read and write timing, which means the read and write timing is the same.

(15) writeTimingStruct: Write timing structure, which can configure SMC_WRTTIM register when the extension mode is enabled, so as to realize use of different timing for read and write.

## 4.3 Timing calculation for APM32F4xx SMC controlling SRAM memory

To control the external SRAM memory, the APM32F4xx SMC must correctly configure two timing parameters in the timing structure: address setup time and data setup time. The other timing parameters are not used and can be set to 0.

Taking the SRAM chip model IS62WV12816 as an example, the timing calculation of the APM32F4xx SMC controlling the SRAM chip is introduced below. According to the IS62WV12816 chip manual, the key timing parameters are as follows:

Table 4 Requirements for Read and Write Timing Parameters of IS62WV12816 SRAM Chip

| Time parameter | Parameter meaning | Time requirements |
|---|---|---|
| tWC | Write Cycle Time | Not less than 55ns |
| tPWE | WE Pulse Width, namely the duration from the time when SMC sends a write enable signal to the time when the data input to SRAM is valid | Not less than 40ns |
| tRC | Read Cycle Time | Not less than 55ns |
| tSA | Address Setup Time | 0 ns, which means no requirements |

The APM32F4xx SMC can control the reading and writing timing of external SRAM in various modes. The read and write timing of Mode A is shown in the following figure:

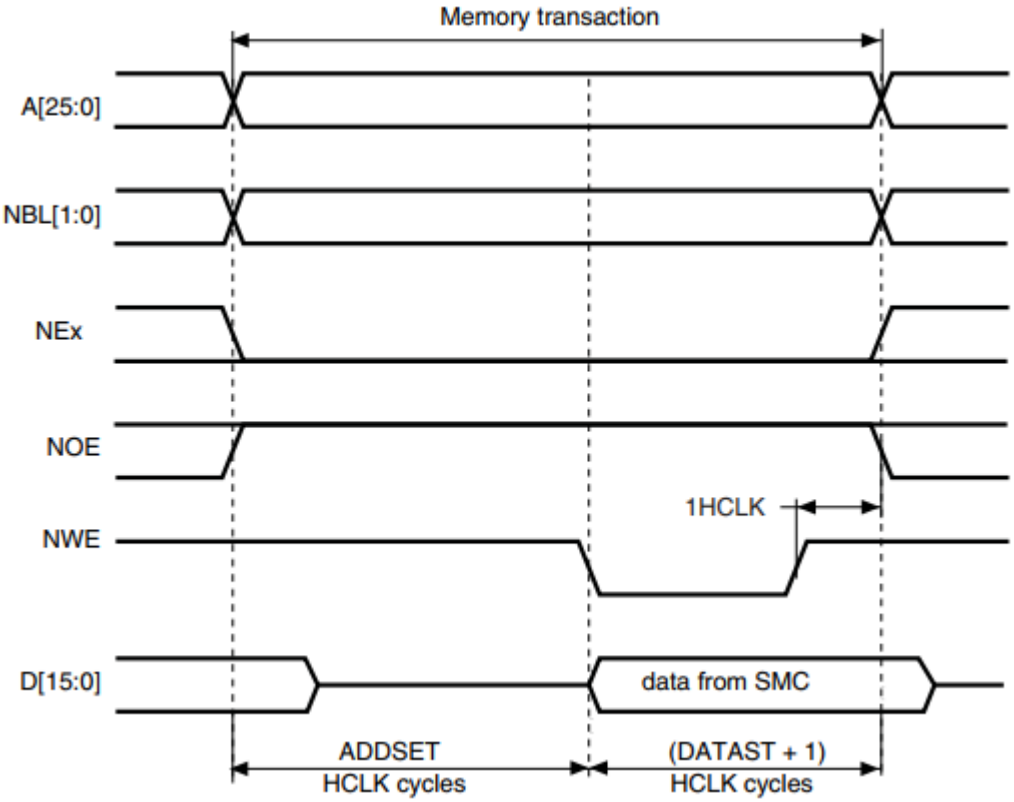


Figure 3 Write Timing of SMC Mode A

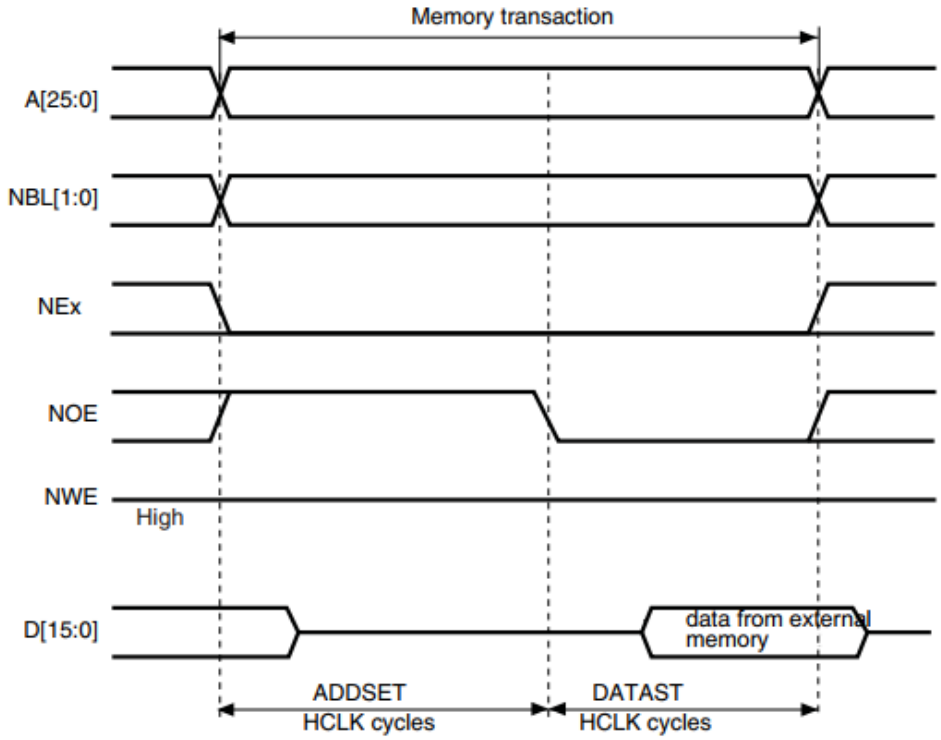


Figure 4 Read Timing of SMC Mode A

According to the time parameter requirements of the IS62WV12816 chip and the timing diagram of SMC reading and writing SRAM, the time parameter calculation formula of the IS62WV12816 chip can be obtained as follows:

tWC = ADDSET + DATASET + 1 >= 55ns

tPWE = DATASET >= 40ns

tRC = ADDSET + DATASET >= 55ns

The parameters ADDSET and DATASET that assign values to the timing structure are in 1 HCLK cycle, while the HCLK clock frequency of APM32F4xx is 168MHz, so 1 HCLK cycle is 1/168 us=6ns.

According to the key time parameter table of IS62WV12816 mentioned earlier, the address setup time is 0, so the parameter ADDSET of the timing structure can be assigned a value of 0. Then, according to the above three expressions, in order to meet the requirements of both read and write timing, it can be concluded that the parameter DATASET is assigned a value of 10, and the time requirements of SRAM read and write can be met.

# 5 Routine of APM32F4xx SMC reading and writing external SRAM

## 5.1 Hardware design

The external SRAM chip used is the IS62WV12816 chip, and the schematic diagram of the connection between the SRAM chip and APM32F4xx is as follows:

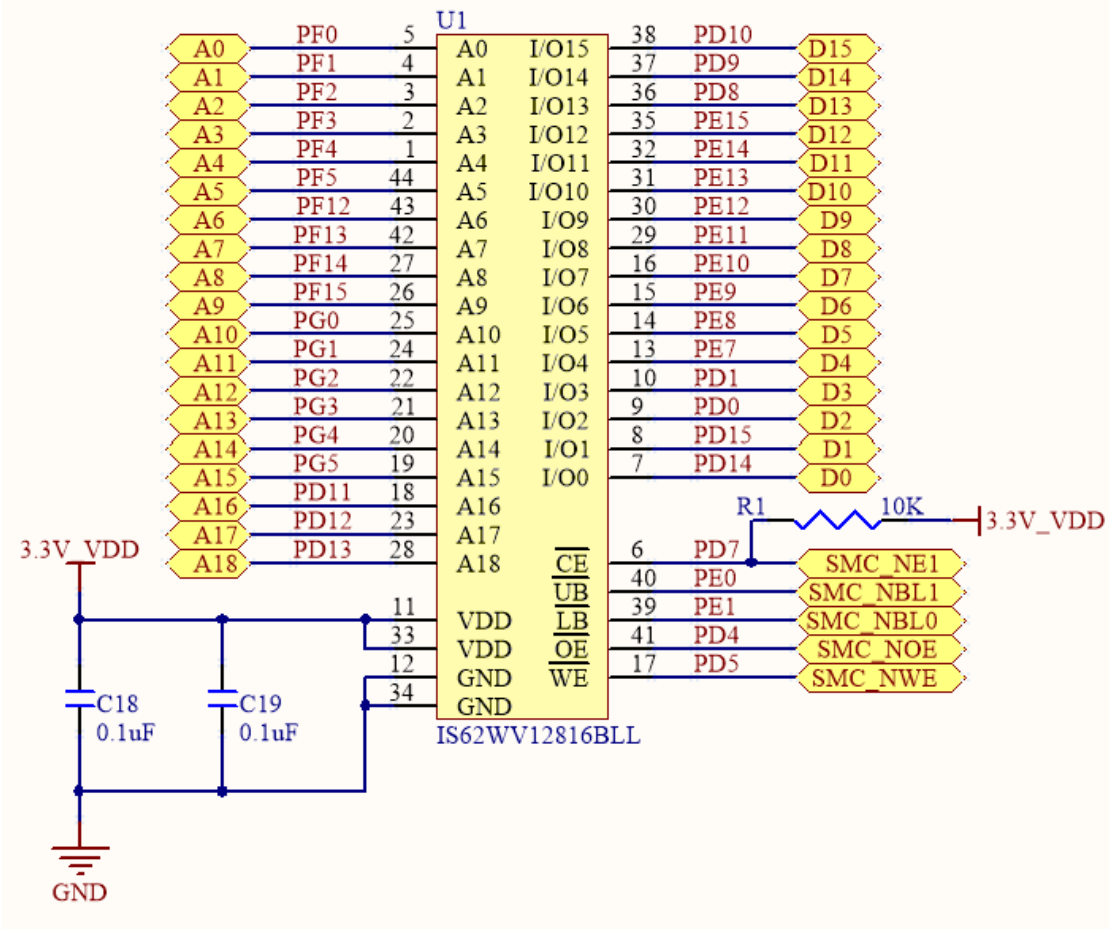


Figure 5 Schematic Diagram of Connection between External SRAM memory and APM32F4xx

It can be seen that many signal lines of the SRAM chip need to be connected to the APM32F4xx chip pins. Most signal lines of the SRAM chip are pins fixed and connected to the MCU, and we can only select the chip selection pin to connect to the SMC_NE1 - NE4. Different connections of chip selection pins will map to different address spaces in the MCU.

In the above schematic diagram, the chip selection pin connects to SMC_NE1, and the storage space of SRAM will be mapped to the address space of 0x60000000 ~ 0x63FFFFFF chip inside the MCU. For the IS62WV12816 model, its storage space is 256KB, and when the MCU accesses the 256KB memory space starting from 0x60000000, the SMC peripheral will generate corresponding access timing according to the initialization configuration, so as to realize external SRAM read and write operations.

## 5.2 Software design

To access the external SRAM memory through APM32F4xx SMC, it is only required to configure the used GPIO pins related to the SMC peripheral and

configure the timing structure and initialization structure of the SMC. The code implementation flowchart is as follows:

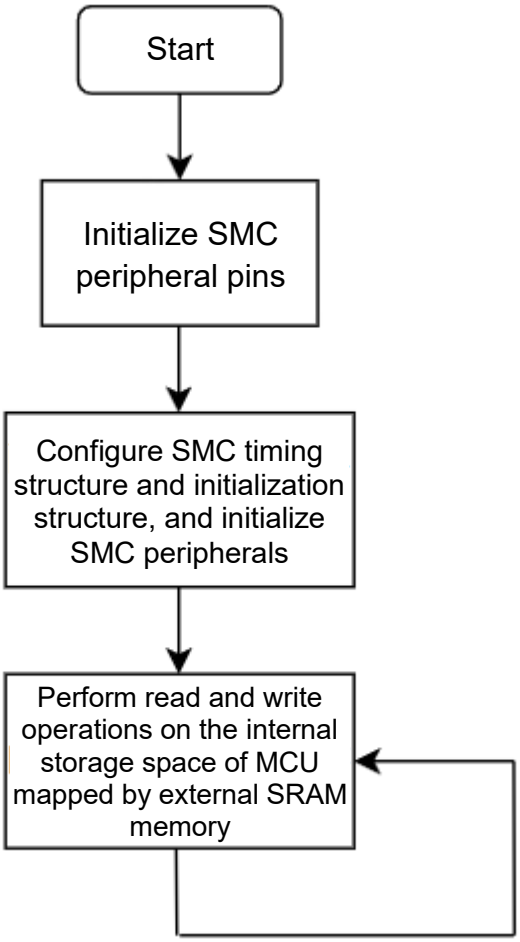


Figure 6 Program Flowchart of SMC Accessing External SRAM Memory

### 5.2.1 SMC GPIO pin configuration

All GPIO pins used in SMC peripheral are configured to the multiplexing function output mode. The PD0 pin is used below as an example for configuration reference, and the configuration of all other pins is similar. The reference code is as follows:

```
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 initialization configuration

The APM32F4xx firmware library has encapsulated the parameters related to timing control and initialization configuration of SMC peripheral in the timing structure and initialization configuration structure. The initialization configuration of SMC peripheral is mainly to assign values to the member variables of these two structures, and finally call the initialization configuration function provided by the firmware library function to implement the initialization configuration of SMC peripheral. The specific code is as follows:

```
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);

}
```

For the parameter configuration related to the timing structure SMC_NORSRAMTimingConfig_T, the values of the corresponding structural members needs to be calculated according to the time parameter requirements provided in the SRAM chip manual, in order to reasonably configure the parameters of each member. The model of the SRAM chip used in this routine is IS62WV12816.

### 5.2.3 SMC accessing external SRAM memory

From the schematic diagram, it can be concluded that the chip selection pin of the external SRAM chip is connected to SMC_NE1 pin, so the storage space of the external SRAM will be mapped to the address space within the range of 0x60000000 ~ 0x63FFFFFF inside the MCU. For the IS62WV12816 model used in this routine, its storage space is 256KB, so the start address of accessing the external SRAM is 0x60000000 and the end address is 0x60040000.

As long as the SMC peripheral is initialized, read and write access to the external SRAM is read and write operation on the memory. For C language, the specific memory addresses can be read and written through pointers, e.g.:

For the 0x6000000 address of SRAM, read 1 byte of data:

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

For the 0x6000000 address of SRAM, write 1 byte of data:

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

To define a variable on a specified memory address, the extension keywords provided by the corresponding compiler can be used to modify the variable. For example, the variable of the ARM-CC compiler is defined as follows:

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

# 6 Revision history

Table 5 Document Version History

| Date | Version | Revision History |
|---|---|---|
| June 5, 2023 | 1.0 | First draft |

## Statement

This manual is formulated and published by Zhuhai Geehy Semiconductor Co., Ltd. (hereinafter referred to as "Geehy"). The contents in this manual are protected by laws and regulations of trademark, copyright and software copyright. Geehy reserves the right to correct and modify this manual at any time. Please read this manual carefully before using the product. Once you use the product, it means that you (hereinafter referred to as the "users") have known and accepted all the contents of this manual. Users shall use the product in accordance with relevant laws and regulations and the requirements of this manual.

1. Ownership of rights

This manual can only be used in combination with chip products and software products of corresponding models provided by Geehy. Without the prior permission of Geehy, no unit or individual may copy, transcribe, modify, edit or disseminate all or part of the contents of this manual for any reason or in any form.

The "Geehy" or "Geehy" words or graphics with "®" or "TM" in this manual are trademarks of Geehy. Other product or service names displayed on Geehy products are the property of their respective owners.

2. No intellectual property license

Geehy owns all rights, ownership and intellectual property rights involved in this manual.

Geehy shall not be deemed to grant the license or right of any intellectual property to users explicitly or implicitly due to the sale and distribution of Geehy products and this manual.

If any third party’s products, services or intellectual property are involved in this manual, it shall not be deemed that Geehy authorizes users to use the aforesaid third party’s products, services or intellectual property, unless otherwise agreed in sales order or sales contract of Geehy.

3. Version update

Users can obtain the latest manual of the corresponding products when ordering Geehy products.

If the contents in this manual are inconsistent with Geehy products, the agreement in Geehy sales order or sales contract shall prevail.

4. Information reliability

The relevant data in this manual are obtained from batch test by Geehy Laboratory or cooperative third-party testing organization. However, clerical errors in correction or errors caused by differences in testing environment may occur inevitably. Therefore, users should understand that Geehy does not bear any responsibility for such errors that may occur in this manual. The relevant data in this manual are only used to guide users as performance parameter reference and do not constitute Geehy's guarantee for any product performance. Users shall select appropriate Geehy products according to their own needs, and effectively verify and test the applicability of Geehy products to confirm that Geehy products meet their own needs, corresponding standards, safety or other reliability requirements. If loses are caused to users due to the user's failure to fully verify and test Geehy products, Geehy will not bear any responsibility.

5. Compliance requirements

Users shall abide by all applicable local laws and regulations when using this manual and the matching Geehy products. Users shall understand that the products may be restricted by the export, re-export or other laws of the countries of the product suppliers, Geehy, Geehy distributors and users. Users (on behalf of itself, subsidiaries and affiliated enterprises) shall agree and promise to abide by all applicable laws and regulations on the export and re-export of Geehy products and/or technologies and direct products.

6. Disclaimer

This manual is provided by Geehy "as is". To the extent permitted by applicable laws, Geehy does not provide any form of express or implied warranty, including without limitation the warranty of product merchantability and applicability of specific purposes.

Geehy will bear no responsibility for any disputes arising from the subsequent design and use of Geehy products by users.

7. Limitation of liability

In any case, unless required by applicable laws or agreed in writing, Geehy and/or any third party providing this manual "as is" shall not be liable for damages, including any general damages, special direct, indirect or collateral damages arising from the use or no

use of the information in this manual (including without limitation data loss or inaccuracy, or losses suffered by users or third parties).

8. Scope of application

The information in this manual replaces the information provided in all previous versions of the manual.