Application Note

SPC11X8/SPD11X8 Flash 使用指南

2019 年 7 月 – 版本 1

目录

1 Flash 存储器概述 5

1.1 主要特性 5

1.2 访问接口 5

1.3 写保护 6

2 Flash 驱动函数 7

3 Flash 操作实例 8

3.1 设定时序参数 8

3.2 XIP 读操作 8

3.3 Flash 控制器操作 8

4 修订历史 11

表格列表

表 1-1: Flash 存储器结构 6
表 2-1: Flash 相关宏定义 7
表 2-2: Flash 驱动函数 7
表 4-1: 文档修订历史 11

图片列表

图 1-1: Flash 存储器访问接口 5
图 1-2: Flash 存储器逻辑结构 6

1 Flash 存储器概述

1.1 主要特性

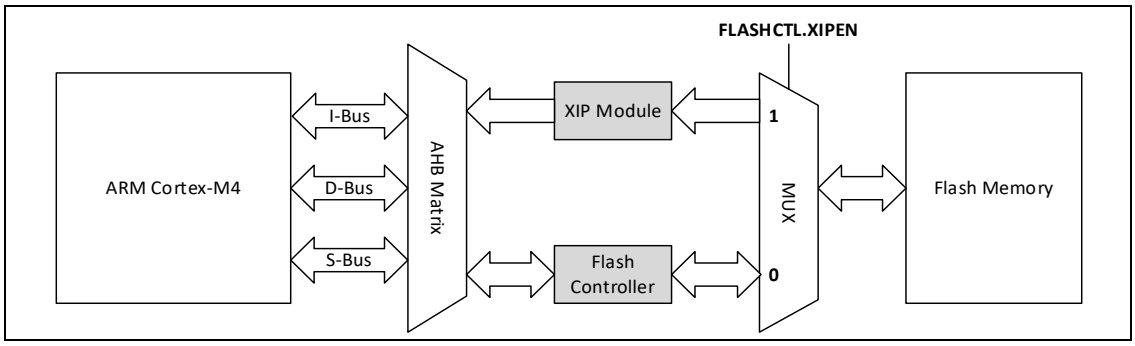
SPC11X8/SPD11X8 片内 Flash 存储器用于存储用户程序、数据等内容，其主要特点如下：

- 容量可达 64KB/128KB
- 存储单元组成
  - 主存储单元：包含 128/256 个扇区（Sector），每个扇区由 128 个 Word 组成
  - NVR 存储单元：包含 2 个扇区，每个扇区由 128 个 Word 组成
- 最小编程单元为一个 Word（32-bit）
- 支持大小为 512 字节的区块擦除
- 支持至少 100000 次擦写（T_J = 85 °C）
- Flash 内数据至少保存 10 年（T_J = 85 °C）
- 支持区块保护模式

1.2 访问接口

SPC11X8/SPD11X8 Flash 存储器的访问接口有两个：XIP 模块和 FLASH 控制器，如图 1-1 所示。XIP 模块只能实现从 Flash 存储器中读取数据，用于 CPU 从 Flash 存储器中取代码和数据；FLASH 控制器是接在 AHB 总线上的一个外设，可以实现 Flash 存储器的读、写以及擦除操作。但是，这两个接口不能同时工作，同一时刻只能有一个接口处于工作状态。Flash 存储器当前的访问接口由寄存器 FLASHCTL.XIPEN 进行控制。

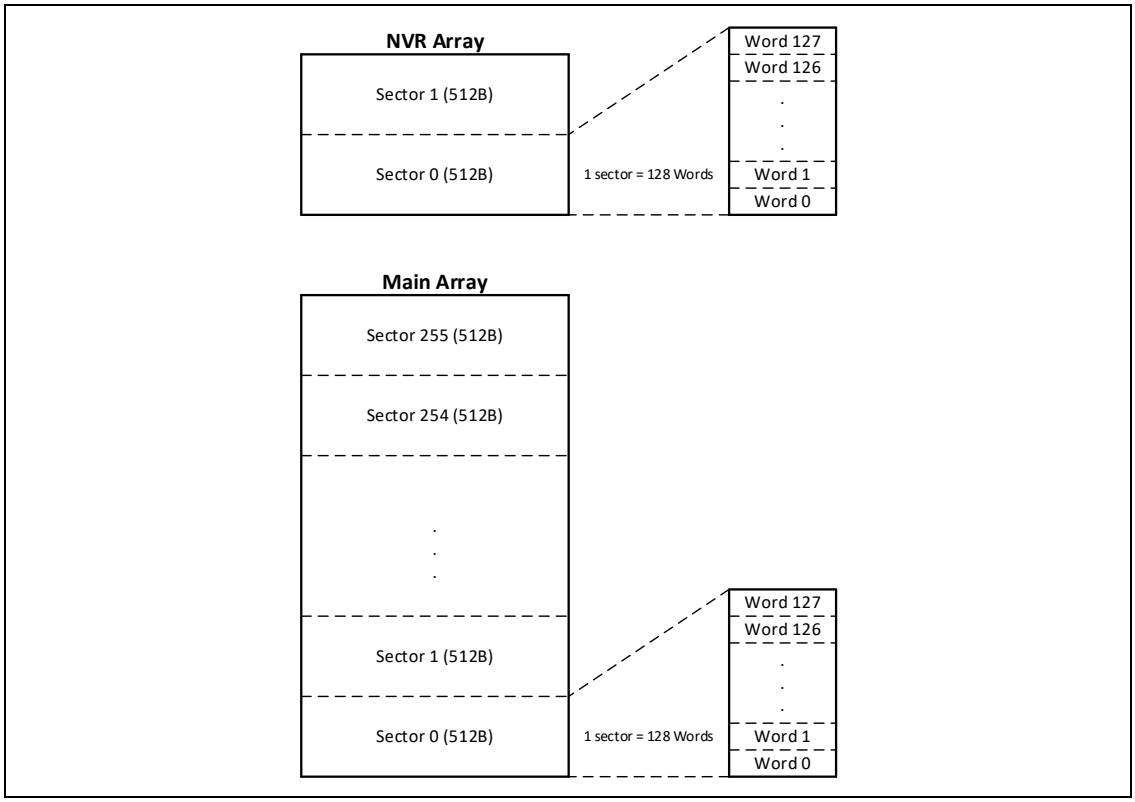
图 1-1: Flash 存储器访问接口



SPC11X8/SPD11X8 Flash 存储器的构成如图 1-2 和表 1-1 所示，包含两个部分：

- 主存储单元：这部分由 128/256 个扇区组成，每个扇区的大小为 512 字节，共计 64KB/128KB；
- NVR 存储单元：这部分由 2 个扇区组成，每个扇区的大小为 512 字节。这两个扇区中，一个扇区可以被配置为 OTP Flash 使用，通过在该扇区开始处写入 0x4C4F434B 实现；另一个扇区被用作 Configuration Words，用于配置 Multi-Zone 保护功能以及 WDT 启动使能等。需要特别说明的是，擦除 Configuration Words 扇区，同时会擦除整个主存储单元。

图 1-2:　Flash 存储器逻辑结构



（1）　对于容量为 64KB 的 Flash 存储器，主存储单元只有 128 个扇区（Sector 0 ~ Sector 127）。

表 1-1:　Flash 存储器结构

存储区块	名称	起始地址	大小
主存储单元	Sector 0	0x1000 0000	512 字节
	Sector 1	0x1000 0200	512 字节
	Sector 2	0x1000 0400	512 字节
	Sector 3	0x1000 0600	512 字节
	…	…	…
	Sector 255	0x1001 FE00	512 字节
NVR 存储单元	OTP	0x1100 0400	512 字节
	Configuration Words	0x1100 0600	512 字节

1.3　写保护

SPC11X8/SPD11X8 支持 Flash 存储器的写保护，用户可以通过设置寄存器 FLASHWPx（x = 0，1，2，3）来保护某些 Flash 存储器区域。一旦使能了写保护，该区域内的 Flash 内容就不能够被擦除或者改写。

2 Flash 驱动函数

SPC11X8/SPD11X8 提供了操作 Flash 存储器的驱动函数，如表 2-1 和表 2-2 所示。

表 2-1: Flash 相关宏定义

宏名	功能及说明
FLASH_EnableXIP()	使能 XIP 接口
FLASH_DisableXIP()	禁能 XIP 接口
FLASH_WALLOW()	使能 Flash 寄存器写操作
FLASH_WDIS()	禁能 Flash 寄存器写操作

表 2-2: Flash 驱动函数

函数名	功能及说明
void FLASH_Read(uint32_t *pu32Buf, uint32_t u32Addr, uint32_t u32NumWords)	从 Flash 读取数据 1） pu32Buf：数据存放地址 2） u32Addr：Flash 开始地址 3） u32NumWords：读取数据数量，单位 Word
ErrorStatus FLASH_ProgramWord(uint32_t u32Data, uint32_t u32Addr)	向 Flash 写入一个 Word 数据 1） u32Data：数据值 2） u32Addr：Flash 地址
ErrorStatus FLASH_Program(uint32_t *pu32Buf, uint32_t u32Addr, uint32_t u32NumWords)	向 Flash 写入数据 1） pu32Buf：数据存放地址 2） u32Addr：Flash 开始地址 3） u32NumWords：写入数据数量，单位 Word
void FLASH_EraseSector(uint32_t u32SectorAddr)	擦除一个扇区 1）u32SectorAddr：扇区地址
void FLASH_EraseChip(void)	擦除整个 Flash 主存储单元
ErrorStatus FLASH_SetTiming(uint32_t u32ClkFreq)	设置 Flash 时序参数 1）u32ClkFreq：系统始终工作频率，单位 Hz

3 Flash 操作实例

3.1 设定时序参数

只有设定合适的 Flash 时序参数，才可以对 Flash 进行正确的操作。Flash 时序参数的默认值对应于系统工作在 32MHz 下。因此，如果系统的工作频率发生变化，Flash 的时序参数也需要做相应的改动，用户可以直接调用驱动函数 FLASH_SetTiming 进行设置：

- 如果系统的工作频率要升高，用户需要先调用 FLASH_SetTiming 设置时序参数，再将系统工作频率升高；
- 如果系统的工作频率要降低，用户需要先将系统工作频率降低，再调用 FLASH_SetTiming 设置时序参数。

3.2 XIP 读操作

通过 XIP 模块，可以实现对 Flash 存储器的读操作。用户可以直接对 Flash 进行字节、半字或者字形式的读取。下面是一个示例代码，从 Flash 中地址 0x10000100 处读去一个字节数据。

```
uint8_t *pu8Data = (uint8_t *)0x10000100;
uint8_t u8Byte;

u8Byte = *pu8Data;
```

3.3 Flash 控制器操作

用户的代码一般是在 Flash 存储器中运行的，此时 XIP 模块是使能的。此时，通过 Flash 控制器对 Flash 存储器进行操作，涉及到 Flash 访问接口切换的问题，需要特别注意。为了能够使 Flash 访问接口切换时，程序能够正常工作，需要将 Flash 控制器的操作代码放在 SRAM 中运行。因此，通过 Flash 控制器对 Flash 存储器进行操作的过程如下：

- 程序从 Flash 中跳转到 SRAM 中，关闭 XIP 模块，使能 Flash 控制器；
- 通过 Flash 控制器执行相应的 Flash 操作；
- 关闭 Flash 控制器，使能 XIP 模块，程序重新跳转到 Flash 执行。

上述过程，在 SPC11X8/SPD11X8 提供的 Flash 驱动函数中都帮用户实现了。用户在实际使用中只需要将 Flash 驱动函数编译到 SRAM 中运行即可，这一步可以通过 Keil 软件提供的 Scatter 文件实现。下面是一个 Scatter 文件示例，将 Flash 驱动函数编译到开始地址为 0x1FFF8000 的 SRAM 中。

Scatter 文件示例

```
LR_IROM1 0x10000000 0x00020000  {    ; load region size_region
  ER_IROM1 0x10000000 0x00020000  {  ; load address = execution address
   *.o (RESET, +First)
   *(InRoot$$Sections)
   .ANY (+RO)
  }
  RW_IRAM1 0x20000000 0x00004000  {  ; RW data
   .ANY (+RW +ZI)
  }
}


LR_IROM2 0x10008000 0x00004000  {    ; load region size_region
  ER_IROM2 0x1FFF8000 0x00004000  {  ; load address = execution address
   flash.o (+RO +RW +ZI)
  }
}
```

下面是通过 Flash 控制器对 Flash 存储器进行读、写以及擦除操作的示例代码。

示例代码

```
#include "SPC11x8/SPD11x8.h"
#include <stdio.h>

uint32_t au32Data_W[128];
uint32_t au32Data_R[128];

int main(void)
{
  ErrorStatus error = SUCCESS;
  uint32_t i , j;

  FLASH_WALLOW();
  FLASH_SetTiming(200000000);
  /* Disable flash write access after flash operation had done */
  FLASH_WDIS();

  CLOCK_InitWithRCO(CLOCK_HCLK_200MHZ);

  Delay_Init();

  /*
   * Init the UART
   *
```

```
   * 1.Set the GPIO34/35 as UART FUNC
   *
   * 2.Enable the UART CLK
   *
   * 3.Set the rest para
   */
  GPIO_SetPinChannel(GPIO_34,GPIO34_UART_TXD);
  GPIO_SetPinChannel(GPIO_35,GPIO35_UART_RXD);
  CLOCK_EnableModule(UART_MODULE);
  UART_Init(UART,38400);


  for(i = 0; i < 128; i++)
    au32Data_W[i] = i;

  /* Erase Sector */
  printf("Erase Sector...\n");
  FLASH_EraseSector(0x10010000);

  printf("Program Sector...\n");
  error = FLASH_Program(au32Data_W, 0x10010000, 128);
  if(error == ERROR)
  {
    printf("Program Sector Error...\n");
    return 1;
  }

  printf("Read and Check Sector...\n");
  FLASH_Read(au32Data_R, 0x10010000, 128);
  for(j = 0; j < 128; j++)
  {
    if(au32Data_R[j] != j)
    {
      printf("READ[0x%08X] = 0x%08X\n", FLASH_START_ADDR + i + j * 4,
au32Data_R[j]);
    }
  }

  printf("Flash Test End!\n\n");

  while(1){}
}
```

4 修订历史

表 4-1: 文档修订历史

日期	版本	修改内容
2019-07-25	1	初始版本