Document Title: PM_FW file system module design description of Safety Control System

Document Number: 17-Q4-000513

Project Number: CT-RD-1601

Project Name: First phase of Safety Control System Development Project

Material Number: N/A

Document Version: A

Classification Level: Highly secret

Document Status: CFC

Controlled Status: Under control

Prepared by: Zhang Lei 2016-12-26

Checked by: Zhu Genghua 2016-12-30

Countersigned by: Liu Yue

Approved by: Wen Yiming 2016-12-30

Revision History

No.	Relevant Chapter	Change Description	Date	Version Before Change	Version After Change	Prepared by	Checked by	Approved by
1		Document created	2016-12-26	None	A	Zhang Lei	Zhu Genghua	Wen Yiming
2								
3								
4								
5								

Relationship between this version and old versions: None.

文件名称：安全控制系统 PM_FW 文件系统模块设计说明书

文件编号：17-Q4-000513

项目编号：CT-RD-1601

项目名称：安全控制系统开发项目一期

物料编号：

版本号/修改码：A

文件密级：机密

文件状态：CFC

受控标识：受控

拟制：张磊 2016 年 12 月 26 日

审核：朱耿华 2016 年 12 月 30 日

会签：刘跃

批准：温宜明 2016 年 12 月 30 日

修订页

编号	章节名称	修订内容简述	修订日期	订前版本	订后版本	拟制	审核	批准
1		创建	2016-12-26		A	张磊	朱耿华	温宜明
2								
3								
4								
5								
6								
7								
8								
9								
10								
11								
12								
13								
14								
15								
16								

本版本与旧文件（版本）的关系：

Content 目录

1 Document overview 文档概述

1.1 Introduction 综述

This document describes the design description of file system of PM of Safety Control System. The document describes the overall concept of the function of the module, and then the sub-function of the modules are described in detail.

This document is the output of module design phase of PM, and is the input for the follow-up coding phase.

本文档描述了安全控制系统中 PM 文件系统的设计方案。文档首先描述了模块功能的总体设计思路，然后将模块功能划分为若干子功能并进行详细说明。

本文档是 PM 文件系统模块设计的输出，也是后续编码的输入。

1.2 Reference 参考文档

1.2.1 Project documents 内部参考文档

[1] Embedded software safety concept of Safety Control System [505], 15-Q02-000059

[1] 安全控制系统嵌入式软件安全概念说明书 [505], 15-Q02-000059

[2] PM_FW software overall design description of safety control system [506], 15-Q02-000074

[2] 安全控制系统 PM_FW 总体设计说明书 [506], 15-Q02-000074

1.3 Terms and abbreviations 术语和缩略语

1.3.1 Terms 术语

Table 1-1 Terms

表 1-1 术语

No. 序号	Term 术语	Description 解释
1.	Erase unit 擦除单位	Erase should before write, erase every time to one block, this block is the erase unit, some Flash may support different erase unit. 写 Flash 之前需要先擦除，每次擦除都要擦除一片，这一片就是一个擦除单位，有的 Flash 会支持多个擦除单位。
2.	File Header 文件头	Each file has one file header to descript its information, such as file name, file length, used Flash space, it is usually stored in Flash, after mounting file system they will be read into memory. 每个文件都有一个文件头用来描述文件信息，如文件名、长度、占用的 Flash 空间等，一般保存在 Flash 上，挂载文件系统之后会读取到内存。
3.	Mount 挂载	Start file system. 启动文件系统。
4.	File descriptor. 文件描述符	The ID of the file. 文件的 ID。

1.3.2 Abbreviation 缩略语

Table 1-2 Abbreviations

表 1-2 缩略语

No. 序号	Abbreviation 缩略语	English description 英文	Chinese description 中文
1.	PM	Processor Module	主处理器模块
2.	SFFS	SPI Flash File System	SPI Flash 文件系统。
3.	SE	Flash Sector	Flash 扇区。
4.	SS	Flash subsector	Flash 子扇区。
5.	SB	File system’s Super Block	文件系统的超级块。
6.	WL	Wearing Level	磨损均衡。

2 Module overview 模块概述

The location of the file system (marked red) in the software hierarchy is shown below.

文件系统（标红）在软件层次中的位置如下图所示。

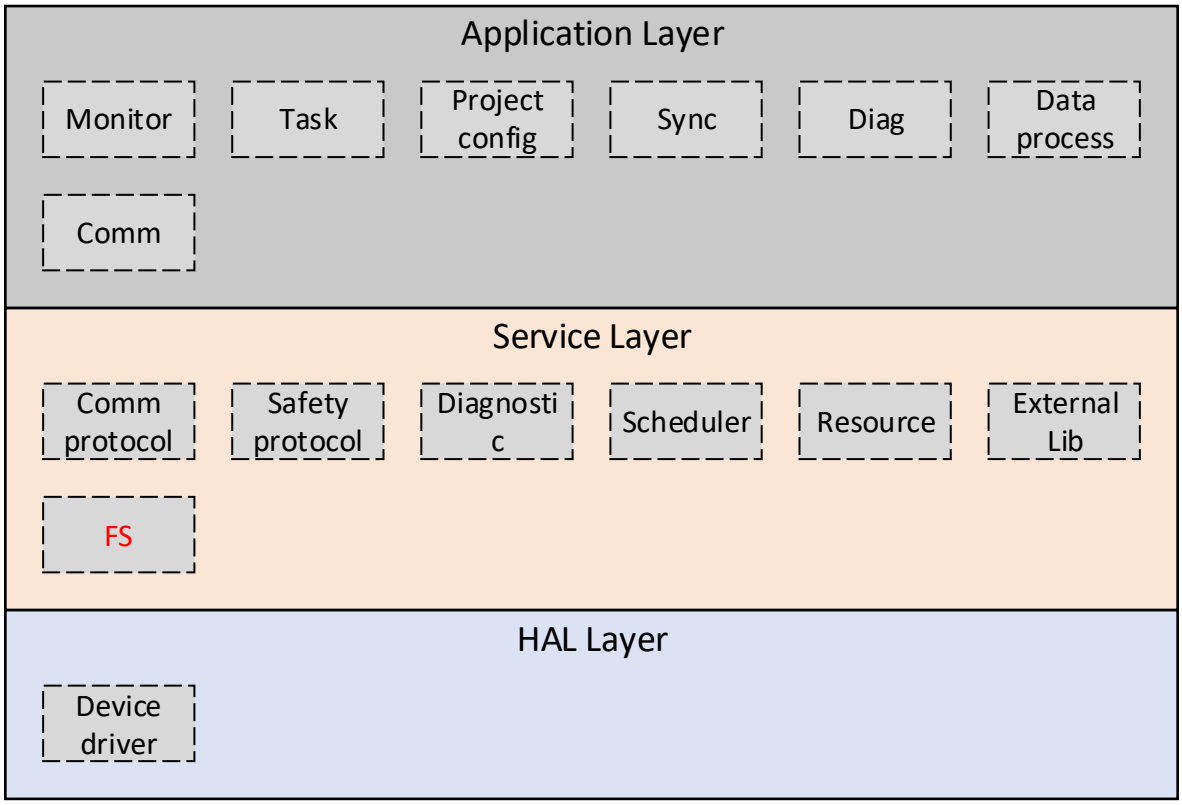

Figure 2-1 the location of the data processing module

图 2-1 模块位置

File system is used to store and manage project files on PM.

文件系统主要用于存储、管理 PM 上的工程文件。

3 Module design 模块设计

3.1 Function description 功能描述

SFFS is used to manage files on PM SPI Flash, and provides read, write, and delete operation interfaces. According to Flash's structure characteristics, SFFS implements basic WL and exception handling mechanism.

SFFS 用于管理 PM SPI Flash 上保存的文件，提供了读、写、删操作接口。根据 Flash 的结构特点，SFFS 实现了基本的擦写平衡和异常处理机制。

3.2 Design concept 设计思路

3.2.1 File access on Flash 文件在闪存上的存取

3.2.1.1 Write file 写文件

It is necessary to follow Flash's structure to store file into Flash. First step is erase Flash as its erase unit. Then write file data to Flash, at the same time, a file header is required to record how much Flash space the file occupies, file header while be stored in memory and Flash. The steps are as follows:

在 Flash 上保存文件需要按照 Flash 的结构特性操作 Flash。首先要按照 Flash 擦除单位擦除 Flash，然后再将文件数据写到 Flash，同时需要一个文件头记录下文件占用了多少 Flash，文件头会同时保存在内存和 Flash 之上。步骤如下：

1. Open a new file and create a file header in memory;
2. Finding out the usable SE in Flash;
3. Call Flash driver to write data;
4. Close file and store file header into Flash.

1. 打开新文件，在内存中创建文件头信息；
2. 在 Flash 上找出可用的 SE；
3. 调用驱动写入数据；
4. 关闭文件，将文件头信息写入 Flash。

3.2.1.2 Read file 读文件

Read file needs to use file header to determine data's length and position in Flash, and the order of every SE. Then file system can read the specified position and length of the file data under application's command. The steps are as follows:

读文件需要使用文件头来确定数据的长度和在 Flash 上的位置，以及各个 SE 的顺序。然后文件系统就可以根据应用程序的命令按读出指定位置、指定长度的文件数据。步骤如下：

1. Open file and read file header's information;
2. Get SE that stored file from file header;
3. Read data on Flash with the position information of file's read/write pointer;
4. Close file.

1. 打开文件，读取文件头信息；
2. 从文件头获取保存文件的 SE；
3. 按照文件读写指针的位置信息读取 Flash 上的数据；
4. 关闭文件。

3.3 Interface function 接口函数

The interface functions which is provided by this module is shown as follows:

1. INT32 fs_mount(INT32 type)

Input argument 输入参数	Output argument 输出参数	Description 描述
type - file system's mount method. type - 文件系统的挂载方式。	FS_OK - Operation was successful. FS_FAIL - Operation failure. FS_OK - 操作成功。	Mount file system. 挂载文件系统。

	FS_FAIL - 操作失败。	

2. INT32 fs_umount(INT32 fsid)

Input argument 输入参数	Output argument 输出参数	Description 描述
fsid - filesystem ID fsid -文件系统号。	FS_OK - Operation was successful. FS_FAIL - Operation failure. FS_OK - 操作成功。 FS_FAIL - 操作失败。	Unmount file system 卸载文件系统

3. INT32 fs_lseek(INT32 fd, INT32 pos, INT32 cfg)

Input argument 接口输入参数	Output argument 接口输出参数	Description 描述
fd - file descriptor. pos - file read/write pointer's offset. cfg - configuration option. fd -文件描述符 pos -文件读写指针偏移量 cfg -配置选项。	FS_FAIL - Operation failure. Other value is the position of file's read/write pointer. FS_FAIL - 操作失败。 其它值文件读写指针位置。	Update file's read/write pointer. 修改文件读写指针位置

4. INT32 fs_open(const INT8 *fname, INT32 flags, INT32 mode)

Input argument 接口输入参数	Output argument 接口输出参数	Description 描述
fname - file name. flags - file open flag. mode - file open mode. fname - 文件名 flags - 文件打开标志 mode - 文件打开模式	FS_FAIL - Operation failure. Other value is file's descriptor. FS_FAIL - 操作失败。 其它值为文件描述符。	Open file with specified format. 按照特定的格式打开指定文件。

5. INT32 fs_read(INT32 fd, void *buf, UINT32 count)

Input argument 接口输入参数	Output argument 接口输出参数	Description 描述

fd - file descriptor. buf - read buffer. count - number of read data. fd - 文件描述符 buf - 读缓冲 count - 读取字节数	FS_FAIL - Operation failure. Other value is the length of the data is actually read. FS_FAIL - 操作失败。 其它值为实际读出的数据字节数。	Read data from file. 从文件读数据。

6. INT32 fs_write(INT32 fd, const void *buf, UINT32 count)

Input argument 接口输入参数	Output argument 接口输出参数	Description 描述
fd - file descriptor. buf - read buffer. count - number of wrote data. fd - 文件描述符 buf - 写缓冲 count - 写入字节	FS_FAIL - Operation failure. Other value is the length of the data is actually written. FS_FAIL - 操作失败。 其它值为实际写入的数据字节数。	Write data to file. 向文件写数据。

7. INT32 fs_close(INT32 fd)

Input argument 接口输入参数	Output argument 接口输出参数	Description 描述
fd - file descriptor. fd - 文件描述符	FS_OK - Operation was successful. FS_FAIL - Operation failure. FS_OK - 操作成功。 FS_FAIL - 操作失败。	Close file. 关闭文件。

8. INT32 fs_format(INT32 type)

Input argument 接口输入参数	Output argument 接口输出参数	Description 描述
type - format method. type - 格式化方式	FS_OK - Operation was successful. FS_FAIL - Operation failure. FS_OK - 操作成功。 FS_FAIL - 操作失败。	Format file system 格式化文件系统

9. INT32 fs_remove(const INT8 *fname)

Input argument 接口输入参数	Output argument 接口输出参数	Description 描述
fname - file name. fname - 文件名	FS_OK - Operation was successful. FS_FAIL - Operation failure. FS_OK - 操作成功。 FS_FAIL - 操作失败。	Delete file. 删除文件。

3.4 Global variable 全局变量

Table 3-1 Global variable list

表 3-1 全局变量列表

No. 序号	Type 变量类型	Name 名称	Description 描述
1.	struct sffs	sffs_slot	Used for storing file system's super block. 保存文件系统的超级块。
2.	INT32	hdrsStat	Used for recording file headers' usage. 记录文件头使用情况。
3.	INT32	fsStat	Identifies the file system is used for the first time 标识文件系统是第一次使用。

3.5 Data structure 数据结构

1. File header structure

```
struct sffs_entry {
   UINT32          magic_num;
   INT8            name[FNAME_LEN+1];
   INT16           name_len;
   INT32           size;
   INT32           ofs;
   UINT32          fid;
   UINT32          seq;
   INT16           se_no[SFFS_BNO_LIM];
   INT16           hdr_ss;
   UINT8           comt;
   UINT8           unused;
   UINT8           deled;
   //file is in writing or reading
   UINT8           stat;
   UINT8           unused2[2];
};
```

2. File system's super block structure

```
struct sffs {
   INT32              device_size;
   struct sffs_entry     file[SFFS_FILE_NUMBER];
   UINT32             free_se[SE_NUM/FREE_SIZEOF];
   INT8               free_ss_hdr[SFFS_FHDR_NUM];
   INT16              free_se_cur;
   INT16              free_ss_hdr_cur;
   INT32              fileNum;
   INT32              mounted;
   INT32              errno;
   UINT32             fidcnt;
   UINT32             seqcnt;
   UINT32             FlashId;
   UINT32             fsid;
   INT8               lock;
   UINT8              pageErase;
   INT8               fsname[16];
   UINT32             magic_num;
   INT32              ss_size;
   INT32              se_size;
   INT32              erase_unit;
   erase_func         erase;
   write_func         write;
   read_func          read;
   seek_func          seek;
};
```

3.6 List of sub-function 子功能列表

The sub-functions list is shown as follows:

子功能列表如下。

Table 3-2 Sub function list

表 3-2 子功能列表

Sub function No. 子功能编号	Function description 功能描述
SWDD-PM-SFFS_NSafR_NSecR_A_001	Mount file system. 挂载文件系统。
SWDD-PM-SFFS_NSafR_NSecR_A_002	Unmount file system. 卸载文件系统
SWDD-PM-SFFS_NSafR_NSecR_A_003	Format file system. 格式化文件系统

SWDD-PM-SFFS_NSafR_NSecR_A_004	Open file. 打开文件。
SWDD-PM-SFFS_NSafR_NSecR_A_005	Write file. 写文件。
SWDD-PM-SFFS_NSafR_NSecR_A_006	Read file. 读文件。
SWDD-PM-SFFS_NSafR_NSecR_A_007	Close file. 关闭文件。
SWDD-PM-SFFS_NSafR_NSecR_A_008	Delete file. 删除文件。
SWDD-PM-SFFS_NSafR_NSecR_A_009	Update file read/write pointer. 更新文件读写指针。
SWDD-PM-SFFS_NSafR_NSecR_A_010	Rename a file. 重命名文件。

4 Design of sub-function 子功能设计

4.1 Mount file system interface

SWDD-PM-SFFS_NSafR_NSecR_A_001

4.1.1 Mount file system interface

4.1.1.1 Function Description 功能描述

This function is used to mount file system.

该函数用于挂载文件系统。

4.1.1.2 Argument Description 参数说明

- Definition 函数定义

  INT32 fs_mount(INT32 type)

- Input argument 输入参数

  type - mount method of file system.

  type - 文件系统的挂载方式。

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.1.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.1.2 Mount file system implementation

4.1.2.1 Function Description 功能描述

This function is used to implement the function of mount file system.

该函数用于实现挂载文件系统功能。

4.1.2.2 Argument Description 参数说明

- Definition 函数定义

  INT32 sffs_mount(struct sffs *fs, INT32 type)

- Input argument 输入参数

  fs - sffs file system's super block pointer.

  type - mount method of file system.

  fs - sffs 文件系统超级块指针。

  type - 文件系统的挂载方式。

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.1.2.3 Processing flow 处理流程

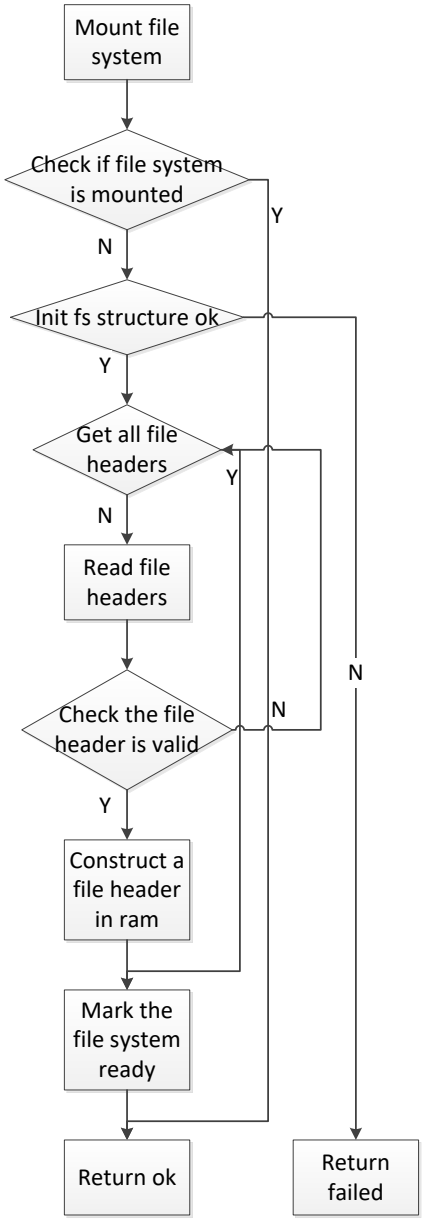

Figure 4-1 the process of mount file system

图 4-1 挂载文件系统的过程

4.2 Unmount file system

SWDD-PM-SFFS_NSafR_NSecR_A_002

4.2.1 Unmount file system interface

4.2.1.1 Function Description 功能描述

This function is used to unmount file system.

该函数用于卸载文件系统。

4.2.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_umount(INT32 fsid)

- Input argument 输入参数

  fsid - file system ID.

  fsid - 文件系统号。

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.2.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.2.2 Umount file system implementation

4.2.2.1 Function Description 功能描述

This function is used to implement the function of unmounts file system.

该函数用于实现卸载文件系统的功能。

4.2.2.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 sffs_umount(struct sffs *fs)

- Input argument 输入参数

  fs - pointer to sffs file system’s super block.

  fs - sffs 文件系统超级块指针。

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.2.2.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.3 Format file system

SWDD-PM-SFFS_NSafR_NSecR_A_003

4.3.1 Format file system interface

4.3.1.1 Function Description 功能描述

This function is used to format file system.

该函数用于格式化文件系统。

4.3.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_format(INT32 type)
- Input argument 输入参数

  type format method.

  type 格式化方式。
- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.3.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.3.2 Format file system implementation

4.3.2.1 Function Description 功能描述

This function is used to implement the function of format file system.

该函数用于实现格式化文件系统的功能。

4.3.2.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 sffs_format(struct sffs *fs, INT32 type)
- Input argument 输入参数

  fs - pointer to sffs file system’s super block.

  type - format method.

  fs - 指向 sffs 文件系统超级块的指针。

  type -格式化方式。
- Output argument 输出函数

FS_OK - Operation is successful.

FS_FAIL - Operation is failed.

FS_OK - 操作成功。

FS_FAIL - 操作失败。

4.3.2.3 Processing flow 处理流程

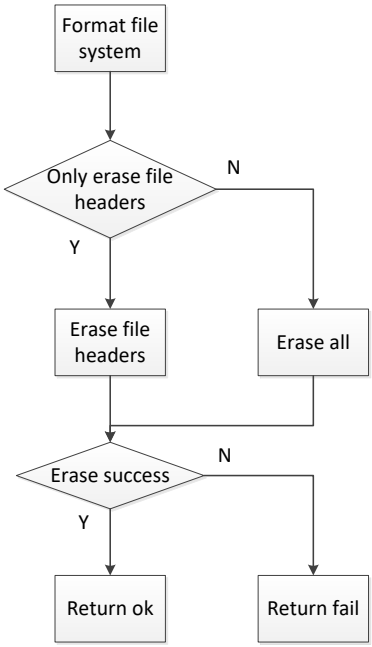

Figure 4-2 the process of format file system

图 4-2 格式化文件系统的过程

4.4 Open file

SWDD-PM-SFFS_NSafR_NSecR_A_004

4.4.1 Open file interface

4.4.1.1 Function Description 功能描述

This function is used to open or create a file.

本函数用于打开或创建一个文件。

4.4.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_open(const INT8 *fname, INT32 flags, INT32 mode)

- Input argument 输入参数

  fname - file name

  flags - file open flags

  mode - file open mode

fname - 文件名

flags - 文件打开标志

mode - 文件打开模式

- Output argument 输出函数

FS_OK - Operation is successful.

FS_FAIL - Operation is failed.

FS_OK - 操作成功。

FS_FAIL - 操作失败。

4.4.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.4.2 Open file implementation

4.4.2.1 Function Description 功能描述

This function is used to implement the function of open or create a file.

本函数用于实现打开或创建一个文件的功能。

4.4.2.2 Argument Description 参数说明

- Function Definition 函数定义

INT32 sffs_open(struct sffs *fs,const INT8 *fname,INT32 mode)

- Input argument 输入参数

fs - pointer to sffs file system’s super block.

fname file name

mode file open mode

fs - 指向 sffs 文件系统超级块的指针。

fname 文件名

mode 文件打开模式

- Output argument 输出函数

FS_OK - Operation is successful.

FS_FAIL - Operation is failed.

FS_OK - 操作成功。

FS_FAIL - 操作失败。

4.4.2.3 Processing flow 处理流程

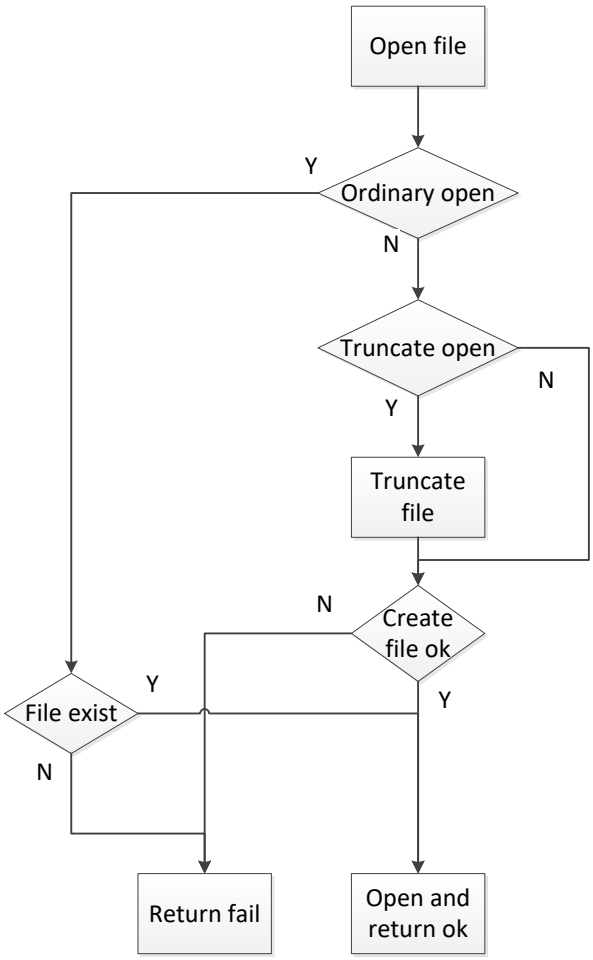

Figure 4-3 the process of open file

图 4-3 打开文件的过程

4.5 Write file

SWDD-PM-SFFS_NSafR_NSecR_A_005

4.5.1 Write file interface

4.5.1.1 Function Description 功能描述

This function is used to write file to Flash.

本函数用于将文件写入 Flash。

4.5.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_write(INT32 fd, const void *buf, UINT32 count)

- Input argument 输入参数

  fd file descriptor

  buf write buffer

count data size in byte

fd 文件描述符

buf 写缓冲

count 数据的字节数

- Output argument 输出函数

FS_FAIL Operation is failed.

Other value is actually written data bytes.

FS_FAIL 操作失败。

其它值为实际写入的数据字节数。

4.5.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.5.2 Write file implementation

4.5.2.1 Function Description 功能描述

This function is used to implement the function of write file to Flash.

本函数用于实现将文件写入 Flash 的功能。

4.5.2.2 Argument Description 参数说明

- Function Definition 函数定义

INT32 sffs_write(struct sffs *fs, INT32 fd, const void *data, INT32 size)

- Input argument 输入参数

fs - pointer to sffs file system’s super block.

fd - file descriptor

data - write buffer

size - data size in byte

fs - 指向 sffs 文件系统超级块的指针。

fd - 文件描述符。

data - 写缓冲。

size- 数据的字节数。

- Output argument 输出函数

FS_FAIL - Operation is failed.

Other value is actually written data bytes.

FS_FAIL - 操作失败。

其它值为实际写入的数据字节数。

4.5.2.3 Processing flow 处理流程

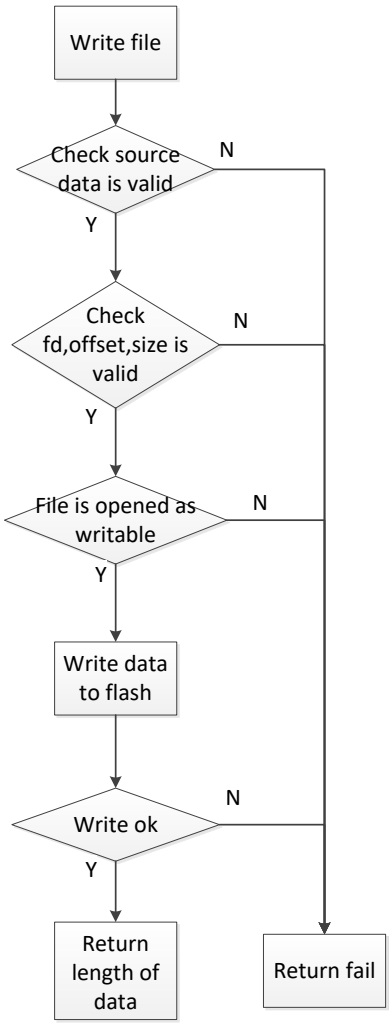

Figure 4-4 the process of write file

图 4-4 写文件的过程

4.6 Read file

SWDD-PM-SFFS_NSafR_NSecR_A_006

4.6.1 Read file interface

4.6.1.1 Function Description 功能描述

This function is used to read file from Flash.

本函数用于从 Flash 读文件。

4.6.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_read(INT32 fd, void *buf, UINT32 count)

- Input argument 输入参数

fd - file descriptor

buf - read buffer

count - The number of bytes of data to read

fd - 文件描述符

buf - 读缓冲

count - 读取数据的字节数

- Output argument 输出函数

FS_FAIL - Operation is failed.

Other value is actually read data bytes.

FS_FAIL - 操作失败。

其它值为实际读出的数据字节数。

4.6.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.6.2 Read file implementation

4.6.2.1 Function Description 功能描述

This function is used to read file from Flash.

本函数用于从 Flash 读文件。

4.6.2.2 Argument Description 参数说明

- Function Definition 函数定义

INT32 sffs_read(struct sffs *fs, INT32 fd, void *data, INT32 size)

- Input argument 输入参数

fs - pointer to sffs file system’s super block.

fd - file descriptor.

data - read buffer.

size - the bytes number of the read date

fs - 指向 sffs 文件系统超级块的指针。

fd - 文件描述符。

data - 读缓冲。

size - 读取字节数。

- Output argument 输出函数

FS_FAIL - Operation is failed.

Other value is actually read data bytes.

FS_FAIL - 操作失败。

其它值为实际读出的数据字节数。

4.6.2.3 Processing flow 处理流程

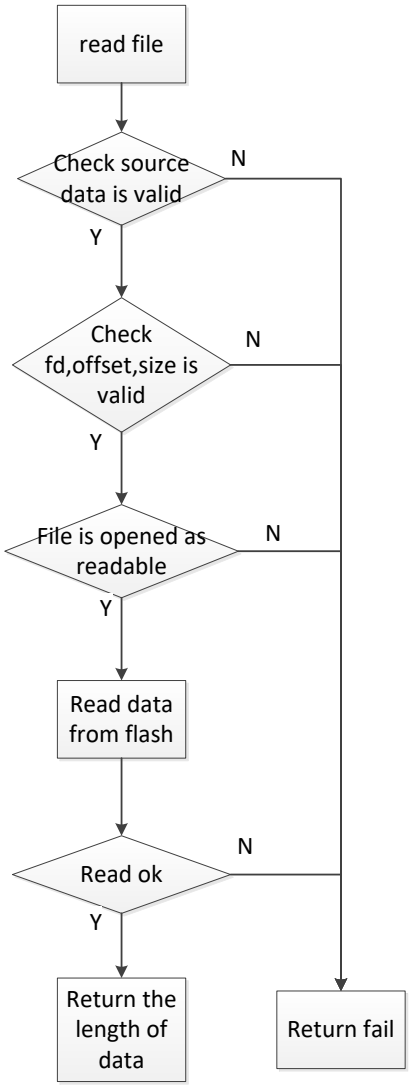

Figure 4-5 the process of read file

图 4-5 读文件的过程

4.7 Close file

SWDD-PM-SFFS_NSafR_NSecR_A_007

4.7.1 Close file interface

4.7.1.1 Function Description 功能描述

This function is used to close file.

本函数用于关闭文件。

4.7.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_close(INT32 fd)

- Input argument 输入参数

  fd - file descriptor。

  fd - 文件描述符

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.7.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.7.2 Close file implementation

4.7.2.1 Function Description 功能描述

This function is used to implement the function of close file.

本函数用于实现关闭文件的功能。

4.7.2.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 sffs_close(struct sffs *fs, INT32 fd)

- Input argument 输入参数

  fd - file descriptor.

  fd - 文件描述符。

- Output argument 输出函数

  FS_OK Operation is successful.

  FS_FAIL Operation is failed.

  FS_OK 操作成功。

  FS_FAIL 操作失败。

4.7.2.3 Processing flow 处理流程

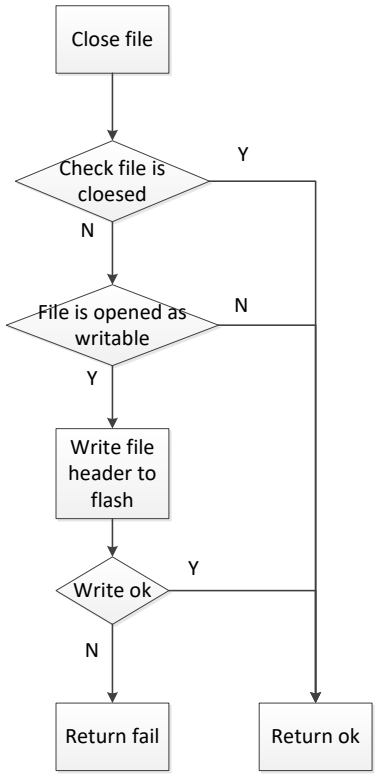

Figure 4-6 the process of close file

图 4-6 关闭文件的过程

4.8 Delete file

SWDD-PM-SFFS_NSafR_NSecR_A_008

4.8.1 Delete file interface

4.8.1.1 Function Description 功能描述

This function is used to delete a file.

本函数用于删除一个文件。

4.8.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_remove(const INT8 *fname)

- Input argument 输入参数

  fname - file name.

  fname - 文件名。

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

FS_OK - 操作成功。

FS_FAIL - 操作失败。

4.8.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.8.2 Delete file implementation

4.8.2.1 Function Description 功能描述

This function is used to implement the function of delete a file.

本函数用于删除一个文件的功能。

4.8.2.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 sffs_remove_byname(struct sffs *fs, const INT8 *fname)

- Input argument 输入参数

  fs - pointer to sffs file system's super block.

  fname - file name

  fs - 指向 sffs 文件系统超级块的指针。

  fname - 文件名

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.8.2.3 Processing flow 处理流程

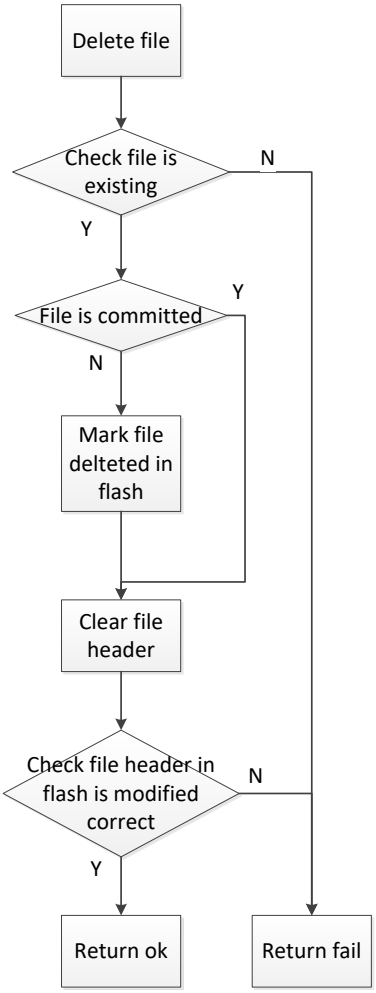

Figure 4-7 the process of delete file

图 4-7 删除文件的过程

4.9 Modify file read/write pointer

SWDD-PM-SFFS_NSafR_NSecR_A_009

4.9.1 Modify file read/write pointer interface

4.9.1.1 Function Description 功能描述

This function is used to reposition or acquire file read/write pointer's position.

本函数用来改变或获取文件读写指针的位置。

4.9.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_lseek(INT32 fd, INT32 pos, INT32 cfg)

- Input argument 输入参数

  fd - file descriptor.

pos - file read/write pointer's offset.

cfg - configuration options.

fd - 文件描述符

pos - 文件读写指针偏移量

cfg - 配置选项。

- Output argument 输出函数

FS_FAIL - Operation is failed.

Other value is the position of file read/write pointer.

FS_FAIL - 操作失败。

其它值为文件读写指针位置。

4.9.1.3 Processing flow 处理流程

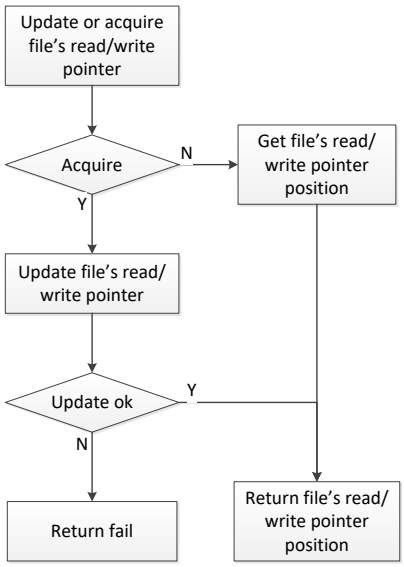

Figure 4-8 the process of fs_lseek interface

图 4-8 fs_lseek 接口的处理过程

4.9.2 Modify file read/write pointer implementation

4.9.2.1 Function Description 功能描述

This function is used to modify file read/write pointer's position.

本函数用来修改文件读写指针的位置。

4.9.2.2 Argument Description 参数说明

- Function Definition 函数定义

INT32 sffs_lseek(struct sffs *fs, INT32 fd, INT32 pos)

- Input argument 输入参数

fs - pointer to sffs file system's super block.

fd - file descriptor

pos - file read/write pointer's offset.

fs - 指向 sffs 文件系统超级块的指针。

fd - 文件描述符。

pos - 文件读写指针偏移量。

- Output argument 输出函数

FS_FAIL - Operation is failed.

Other value is the position of file read/write pointer.

FS_FAIL - 操作失败。

其它值为文件读写指针位置。

4.9.2.3 Processing flow 处理流程

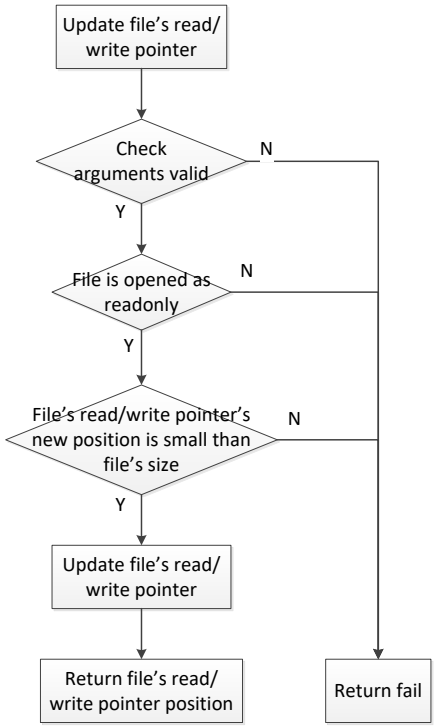

Figure 4-9 the process of modify file read/write pointer

图 4-9 修改文件读写指针的过程

4.10 Rename file

SWDD-PM-SFFS_NSafR_NSecR_A_010

4.10.1 Rename file interface

4.10.1.1 Function Description 功能描述

This function is used to rename file.

本函数用于重命名文件。

4.10.1.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 fs_rename(const INT8 *old_name, const INT8 *new_name)

- Input argument 输入参数

  old_name - file's current name

  new_name - file's new name.

  old_name - 文件当前的名字。

  new_name - 文件的新名字

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.10.1.3 Processing flow 处理流程

This function has no branch and the processing flow is omitted.

此函数无分支，流程图省略。

4.10.2 Rename file implementation

4.10.2.1 Function Description 功能描述

This function is used to implement the function of rename file.

本函数用于实现删除文件的功能。

4.10.2.2 Argument Description 参数说明

- Function Definition 函数定义

  INT32 sffs_rename(struct sffs *fs,const INT8 *old_name,const INT8 * new_name)

- Input argument 输入参数

  fs - pointer to sffs file system's super block.

  old_name - file's current name

  new_name - file's new name.

  fs - 指向 sffs 文件系统超级块的指针。

  old_name - 文件当前的名字。

  new_name - 文件的新名字

- Output argument 输出函数

  FS_OK - Operation is successful.

  FS_FAIL - Operation is failed.

  FS_OK - 操作成功。

  FS_FAIL - 操作失败。

4.10.2.3 Processing flow 处理流程

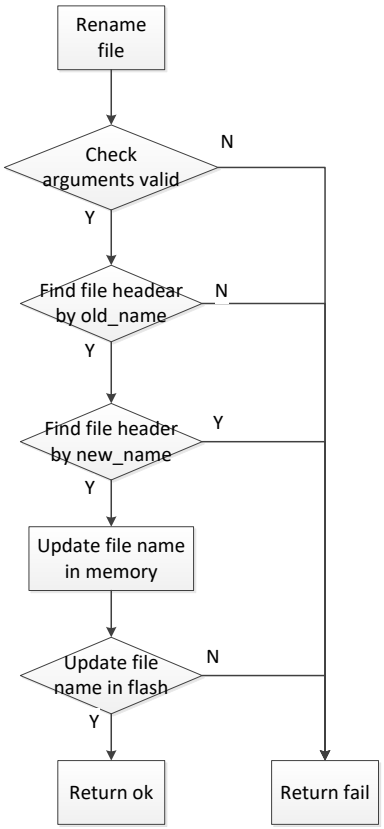

Figure 4-10 the process of rename file

图 4-10 重命名文件的过程

——以下无正文