SPIFFS文件系统移植–基于STM32F407

2018年6月3日 wujique 3 Comments

SPIFFS文件系统，有什么特别呢？
从名字就可知，这是一个用于SPI FLASH的file system。
现在好像在ESP8266上用的很多，感觉慢慢有的人气了。
从特性上说，这个文件系统用很少的RAM资源。
但是，我最后会吐槽吐槽这个特性的。
网络上资源不多，好在github上说明文档挺全。

github

地址：https://github.com/pellepl/spiffs
最新版本是0.3.7.
目录结构如下：

Search...

近期文章

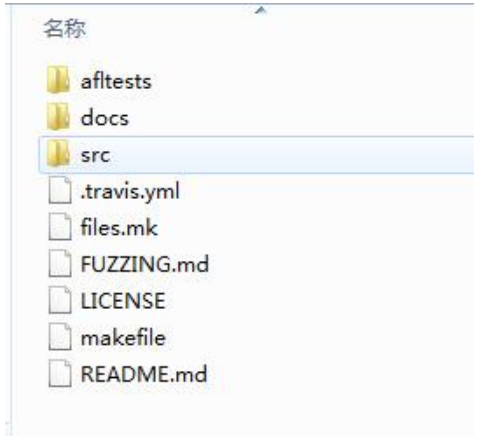
SI Sourceinsight 配色推荐
二维码解码库ZBAR移植到STM32
关于软件设计的一点思考
STM32CubeIDE+MSYS2+Openocd+DAPLink
调试STM32H750VB
1.4 基于标准库建立工程模板

分类目录

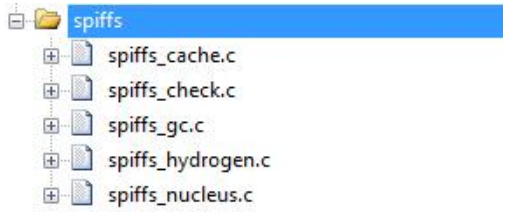
【提高】嵌入式软件开发实践
产品硬件资料
嵌入式开发经验分享
推荐

功能

登录
文章RSS
评论RSS
WordPress.org



添加到工程的文件如下：



在github上有wiki说明如何移植
https://github.com/pellepl/spiffs/wiki
认真读完这个wiki，基本就知道如何使用spiffs。
对于spiffs的设计，有一个网站可做参考, spiffs技术手册。
https://blog.csdn.net/zhangjinxing_2006/article/details/75050611

移植概述

下面我们说说移植的过程。
1. 配置
spiffs-0.3.7\src\default下有一个默认的配置文件，`spiffs_config.h`。
为了方便，拷贝一个到spiffs-0.3.7\src。
在工程中，头文件搜索路径，只包含src路径，不要包含\src\default。
spiffs_config.h头部包含的头文件修改如下。
在这里也定义了变量名。

```
//#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stddef.h>
#include "stm32f4xx.h"

typedef u32  u32_t;
typedef u16 u16_t;
typedef u8  u8_t;

typedef s32  s32_t;
typedef s16 s16_t;
typedef s8  s8_t;
```

修改几个宏定义

```
#ifndef SPIFFS_USE_MAGIC
#define SPIFFS_USE_MAGIC               (1)
#endif

#ifndef SPIFFS_USE_MAGIC_LENGTH
#define SPIFFS_USE_MAGIC_LENGTH        (1)
#endif
#endif
```

在spiffs-0.3.7\src\spiffs_nucleus.h文件中定义的联合用了一个gcc特性，在MDK编译会报错。
在联合前增加一句代码#pragma anon_unions 即可。

```
  #pragma anon_unions
  union {
    // type read cache
    struct {
      // read cache page index
      spiffs_page_ix pix;
    };
#if SPIFFS_CACHE_WR
    // type write cache
```

```
    struct {
      // write cache
      spiffs_obj_id obj_id;
      // offset in cache page
      u32_t offset;
      // size of cache page
      u16_t size;
    };
#endif
  };
} spiffs_cache_page;
```

2 应用
首先,
要实现SPI FLASH操作函数，SPIFFS需要的函数格式如下

```
s32_t core_spiflash_spiffs_read(u32_t addr, u32_t size, u8_t *dst);
static s32_t core_spiflash_spiffs_write(u32_t addr, u32_t size, u8_t *src);
static s32_t core_spiflash_spiffs_erase(u32_t addr, u32_t size);
```

第二，
定义文件系统相关参数

```
/*
    文件系统配置

    #define SPIFFS_SINGLETON (0) 这个宏配置为0，也就是支持多个spiffs
*/
spiffs_config cfg=
    {
        /* 分配给SPIFFS的空间 要是逻辑扇区的整数倍*/
        .phys_size = 1024*4*4*128,
        /* 起始地址 */
        .phys_addr = 0,
        /*
            物理扇区，也就是一次擦除的大小，要跟hal_erase_f函数擦除的一致
        */
```

```
        .phys_erase_block = 1024*4,
        /* 逻辑扇区，必须是物理扇区的整数倍
            最好是：log_block_size/log_page_size = （32-512）
        */
        .log_block_size = 1024*4*4,
        /*逻辑页，通常是256*/
        .log_page_size = LOG_PAGE_SIZE,

        .hal_read_f = core_spiflash_spiffs_read,
        .hal_write_f = core_spiflash_spiffs_write,
        .hal_erase_f = core_spiflash_spiffs_erase,

    };
```

第三，
挂载文件系统，如果是第一次挂载会失败，需要卸载文件系统再格式化文件系统，最后重新挂载即可。

```
/*文件系统结构体*/
static spiffs fs;

/*页定义*/
#define LOG_PAGE_SIZE      256

static u8_t spiffs_work_buf[LOG_PAGE_SIZE*2];
static u8_t spiffs_fds[32*4];
static u8_t spiffs_cache_buf[(LOG_PAGE_SIZE+32)*4];

/**
 *@brief:
 *@details: 格式化文件系统
 *@param[in]
 *@param[out]
 *@retval:
 */
void sys_spiffs_format(void)
{

```

```
    wjq_log(LOG_INFO, ">---format spiffs coreflash\r\n");

    /* 格式化之前要先unmount */
    SPIFFS_unmount(&fs);

    SPIFFS_format(&fs);

    int res = SPIFFS_mount(&fs,
      &cfg,
      spiffs_work_buf,
      spiffs_fds,
      sizeof(spiffs_fds),
      spiffs_cache_buf,
      sizeof(spiffs_cache_buf),
      0);
    wjq_log(LOG_INFO, "mount res: %i\r\n", res);

    wjq_log(LOG_INFO, ">---format spiffs coreflash finish\r\n");

}
/**
 *@brief:
 *@details: 挂载spiffs文件系统
 *@param[in]
 *@param[out]
 *@retval:
 */
void sys_spiffs_mount_coreflash(void)
{
    int res = SPIFFS_mount(&fs,
                           &cfg,
                           spiffs_work_buf,
                           spiffs_fds,
                           sizeof(spiffs_fds),
                           spiffs_cache_buf,
                           sizeof(spiffs_cache_buf),
                           0);
```

```
    wjq_log(LOG_INFO, "mount res: %i\r\n", res);

    if(SPIFFS_ERR_NOT_A_FS == res )
    {
        sys_spiffs_format();
    }
}
```

然后，就可以进行基本读写测试啦。

吐槽

1
如果要好好用这个文件系统，需要多次测试找一个设置平衡点。
什么意思呢？也就是你这个文件系统如何配置。
也就是spiffs_config中的：逻辑块多大，页设置多大。不同的设置会严重影响性能。
因为这个文件系统为了省内存，没有任何索引。
具体细节大家自己分析，我就说一个事实：

> 打开一个文件，很可能要轮询所有BLOCK的第一页。
> 当你BLOCK跟页设置小，数量就多，操作文件就会很慢很慢，，，，。
> 为什么这么慢？作者在WIKI上有说明。

2
目前还没有认真使用这个文件系统，但是总感觉它有点上不成下不就的感觉。
是的，嵌入式RAM紧张，但是嵌入式对速度也敏感啊！
想起以前公司用的文件系统，速度不慢，RAM用得也不算多。
怎么做到呢？
限制其他性能，例如，最多只能创建100个文件，文件系统最大只能到1M。
这样的限制对于单片机系统来说，其实是能用的。
（带这个文件系统的产品估计出货也1000万台了吧）

移植源码

在开源STM32代码上有这个文件系统的移植，需要请参观：
https://github.com/wujique/stm32f407/tree/sw_arch

> 这个仓库是个人写的一些代码，主要是按照经验进行了一些程序设计，自认为比大部分教程的例程代码要好。
> 后续会慢慢添加各种外设驱动。大家拿来用即可。
> 欢迎商用，后果自负

嵌入式开发经验分享

← Previous
STM32F407 DMA配置分析

Next →
Littlefs文件系统移植

3 thoughts on “SPIFFS文件系统移植–基于STM32F407”

罗拉 says:
2018年9月29日 at 下午5:22

呵呵。学习了。感触良多！

回复

daxi says:
2018年10月25日 at 上午8:53

学无止境，认真拜读！

回复

xing says:
2018年11月12日 at 下午4:36

来看看，因为，总能学到东西！

回复

发表评论

电子邮件地址不会被公开。 必填项已用*标注

评论

姓名 *

电子邮件 *

发表评论