

个人资料



mengfei314

+ 加关注

✉ 发私信

访问： 719次

积分： 26

等级： BLOG > 1

排名： 千里之外

原创： 2篇    转载： 0篇

译文： 0篇    评论： 3条

文章搜索

Q

文章存档

2016年03月 (1)

2016年02月 (1)

阅读排行

使用STM32CUBEMX生 (418)

使用STM32CUBEMX生 (274)

评论排行

使用STM32CUBEMX生 (3)

使用STM32CUBEMX生 (0)

推荐文章

- \*Android RoccoFix 热修复框架
- \* android6.0源码分析之Camera API2.0下的初始化流程分析
- \*Android\_GestureDetector手势滑动使用
- \*Android MaterialList源码解析
- \*Android开源框架Universal-Image-Loader基本介绍及使用
- \*Android官方开发文档Training系列课程中文版：创建自定义View之View的创建

最新评论

- 使用STM32CUBEMX生成USB M  
mengfei314: @lizq3531:建议检查一下SD卡读写sector的函数，可以先裸测一下
- 使用STM32CUBEMX生成USB M  
lizq3531: 可否请教一个问题，为什么我的SD卡的读和写都不能返回OK，但是获取状态信息和SD卡的信息都没有问题， ...
- 使用STM32CUBEMX生成USB M  
lizq3531: 可否请教一个问题，为什么我的SD卡的读和写都不能返回OK，但是获取状态信息和SD

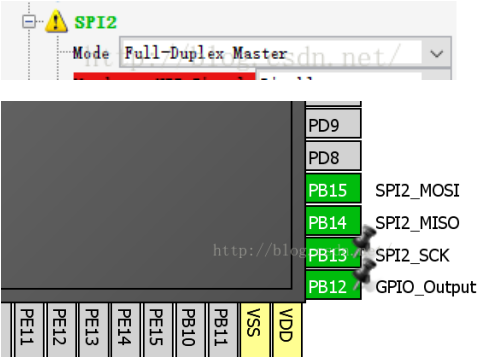
💡 【公告】博客系统优化升级    📁 【收藏】Html5 精品资源汇集    我们为什么选择Java

📄 使用STM32CUBEMX生成FatFS代码，操作SPI FLASH

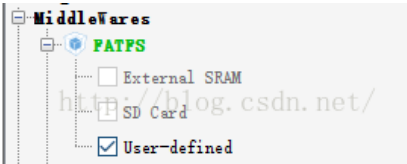
2016-03-10 21:50    👁 280人阅读    💬 评论(0)    收藏    举报

📄 版权声明：本文为博主原创文章，未经博主允许不得转载。

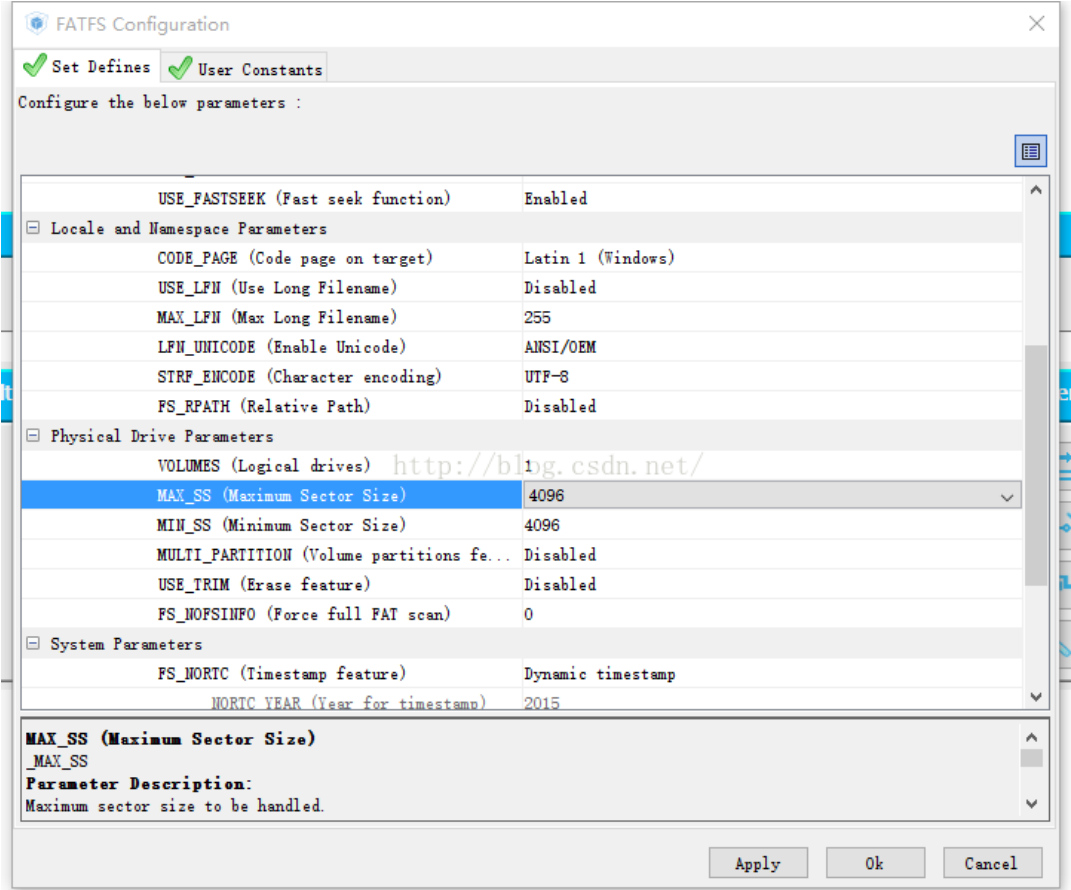
首先配置SPI，我的板子是SPI2连接到SPI FLASH 上，我的flash是W25Q64， PB12用来当CSN。



接下来配置FATFS，这里选择用户定义的。

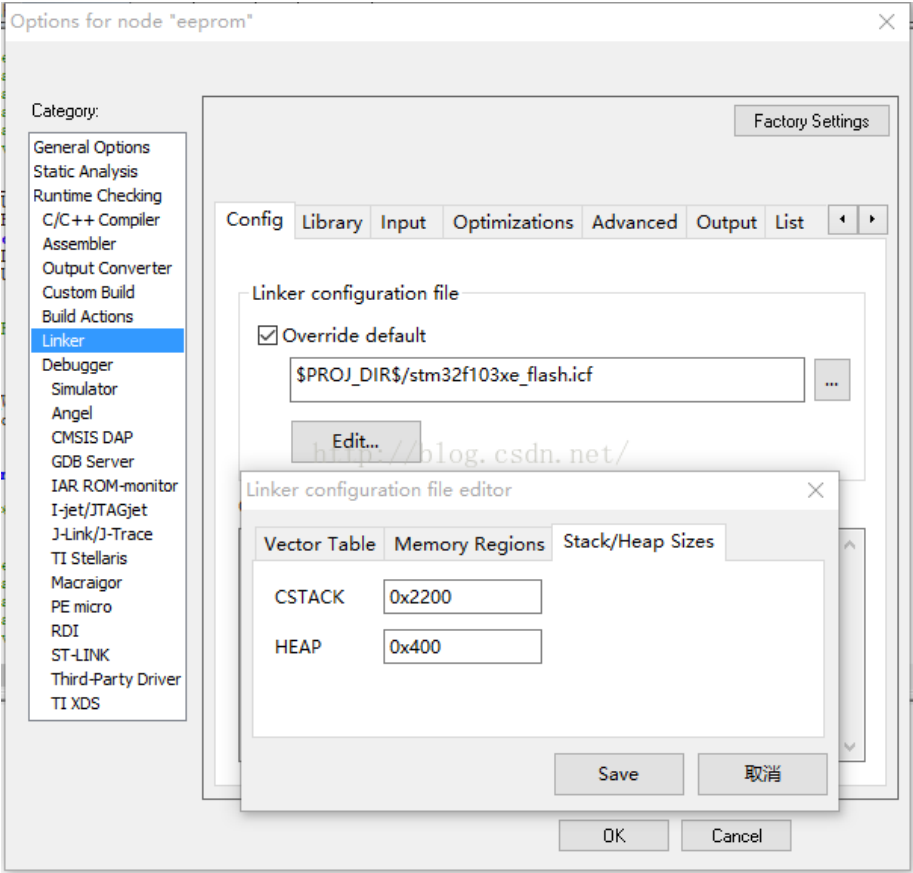


配置FATFS的时候要注意，由于SPI FLASH 的sector是4096字节的，故需要设置sector的大小为4096，其余选项根据自己情况配置。



配置好了生成代码和工程。记得把堆栈尺寸调大一些。我用IAR这样配置：

卡的信息都没有问题，...



接着把SPI FLASH的读写操作实现：读一个sector和写一个sector。我写好的函数叫做W25\_WriteSector和W25\_ReadSector. 然后定义好常量：

```
[cpp]
01. #define PAGE_SIZE      256
02. #define SECTOR_SIZE    4096
03.
04. #define SECTOR_COUNT    200
05. #define BLOCK_SIZE      65536
06. #define FLASH_PAGES_PER_SECTOR SECTOR_SIZE/PAGE_SIZE
```

下一步打开user\_diskio.c 文件，填充几个函数。

```
[cpp]
01. /**
02.  * @brief Reads Sector(s)
03.  * @param pdrv: Physical drive number (0..)
04.  * @param *buff: Data buffer to store read data
05.  * @param sector: Sector address (LBA)
06.  * @param count: Number of sectors to read (1..128)
07.  * @retval DRESULT: Operation result
08.  */
09. DRESULT USER_read (
10.     BYTE pdrv,      /* Physical drive nmuber to identify the drive */
11.     BYTE *buff,      /* Data buffer to store read data */
12.     DWORD sector,    /* Sector address in LBA */
13.     UINT count       /* Number of sectors to read */
14. )
15. {
16.     /* USER CODE HERE */
17.     UINT i = 0;
18.     for(i = 0; i < count; i++)
19.     {
20.         W25_ReadSector(sector, buff);
21.         sector++;
22.         buff += SECTOR_SIZE;
23.     }
24.     return RES_OK;
25. }
```

```
[cpp]
01. /**
02.  * @brief Writes Sector(s)
03.  * @param pdrv: Physical drive number (0..)
04.  * @param *buff: Data to be written
05.  * @param sector: Sector address (LBA)
06.  * @param count: Number of sectors to write (1..128)
07.  * @retval DRESULT: Operation result
08.  */
09. #if _USE_WRITE == 1
10. DRESULT USER_write (
11.     BYTE pdrv,      /* Physical drive nmuber to identify the drive */
12.     const BYTE *buff, /* Data to be written */
13.     DWORD sector,    /* Sector address in LBA */
```

```
14.     UINT count          /* Number of sectors to write */
15. )
16. {
17.     /* USER CODE HERE */
18.     UINT i = 0;
19.     for(i = 0; i < count; i++)
20.     {
21.         W25_WriteSector(sector, buff);
22.         sector++;
23.         buff += SECTOR_SIZE;
24.     }
25.     return RES_OK;
26. }
27. #endif /* _USE_WRITE == 1 */
```

```
[cpp]
01. /**
02.  * @brief I/O control operation
03.  * @param pdrv: Physical drive number (0..)
04.  * @param cmd: Control code
05.  * @param *buff: Buffer to send/receive control data
06.  * @retval DRESULT: Operation result
07.  */
08. #if _USE_IOCTL == 1
09. DRESULT USER_ioctl (
10.     BYTE pdrv,          /* Physical drive nmuber (0..) */
11.     BYTE cmd,           /* Control code */
12.     void *buff          /* Buffer to send/receive control data */
13. )
14. {
15.     DRESULT res = RES_OK;
16.
17.     switch(cmd)
18.     {
19.         case CTRL_SYNC :
20.             break;
21.
22.         case CTRL_TRIM:
23.             break;
24.
25.         case GET_BLOCK_SIZE:
26.             *(DWORD*)buff = BLOCK_SIZE;
27.             break;
28.
29.         case GET_SECTOR_SIZE:
30.             *(DWORD*)buff = SECTOR_SIZE;
31.             break;
32.
33.         case GET_SECTOR_COUNT:
34.             *(DWORD*)buff = SECTOR_COUNT;
35.             break;
36.
37.         default:
38.             res = RES_PARERR;
39.             break;
40.     }
41.
42.     return res;
43. }
44. #endif /* _USE_IOCTL == 1 */
```

这些函数填充好，就可以用FATFS了。对于一个新的SPI FLASH，先挂载f\_mount，再格式化文件系统f\_mkfs，之后就可以做各种新建文件、读写操作了。

补充一个，在fatfs.c文件中，定义了这样一个：

```
[cpp]
01. char USER_Path[4]; /* USER logical drive path */
```

之后我们可以在main或者其他文件里用extern声明它，mount和mkfs时的USER\_Path都是它。

附上我的测试代码

```
[cpp]
01. void mount_disk(void)
02. {
03.     uint8_t res = f_mount(&fs, USER_Path, 0);
04.     if (res != FR_OK)
05.     {
06.         printf("FAILED: %d\n",res);
07.         return;
08.     }
09.     printf("MOUNT OK\n");
```

```
10. }
11.
12. void format_disk(void)
13. {
14.     uint8_t res = 0;
15.
16.     printf("PROCESSING...\n");
17.     res = f_mkfs(USER_Path, 1, 4096);
18.     if (res == FR_OK)
19.     {
20.         printf("OK!\n");
21.     }
22.     else
23.     {
24.         printf("failed with: %d\n",res);
25.     }
26. }
27.
28. void create_file(void)
29. {
30.     FIL file;
31.     FIL *pf = &file;
32.     uint8_t res;
33.
34.     res = f_open(pf, "0:/test.txt", FA_OPEN_ALWAYS | FA_WRITE);
35.     if (res == FR_OK)
36.     {
37.         printf("creat ok\n");
38.     }
39.     else
40.     {
41.         printf("creat failed\n");
42.         printf("error code: %d\n",res);
43.     }
44.
45.     f_printf(pf, "hello fatfs!\n");
46.
47.     res = f_close(pf);
48.     if (res != FR_OK)
49.     {
50.         printf("close file error\n");
51.         printf("error code: %d\n",res);
52.     }
53. }
54.
55. void get_disk_info(void)
56. {
57.     FATFS fs;
58.     FATFS *fls = &fs;
59.     FRESULT res;
60.     DWORD fre_clust;
61.
62.     res = f_getfree("/",&fre_clust,&fls);          /* Get Number of Free Clusters */
63.     if (res == FR_OK)
64.     {
65.                                     /* Print free space in unit of MB (assuming 4096
66.         printf("%d KB Total Drive Space.\n"
67.               "%d KB Available Space.\n",
68.               ((fls->n_fatent-2)*fls->csize)*4,(fre_clust*fls->csize)*4);
69.     }
70.     else
71.     {
72.         printf("get disk info error\n");
73.         printf("error code: %d\n",res);
74.     }
75. }
76.
77. void read_file(void)
78. {
79.     FIL file;
80.     FRESULT res;
81.     UINT bw;
82.     uint8_t rbuf[100] = {0};
83.
84.     res = f_open(&file, "0:/test.txt", FA_READ);
85.     if (res != FR_OK)
86.     {
87.         printf("open error: %d\n",res);
88.         return;
89.     }
90.     f_read(&file, rbuf, 20, &bw);
91.     printf("%s\n", rbuf);
92.
93.     res = f_close(&file);
94.     if (res != FR_OK)
95.     {
96.         printf("close file error\n");
97.         printf("error code: %d\n",res);
98.     }
99. }
```

[▲ 上一篇](#) [使用STM32C](#) [▲ 返回顶部](#) [SB Mass Storage代码，通过SDIO读写TF卡](#)

- Web前端从零基础到高手之路
- iOS进阶开发-调试程序
- iOS8-Swift开发教程
- Flash实战技能应用从入门到精通
- 微信公众平台开发入门
- stm32f4基于spi用fatfs读写SD卡的实现
- stm32使用SPI对W25Q64—8M字节FLASH的读写
- stm32神州开发板SPI Flash使用分析
- 使用stm32cubemx开发四串口标准化输出
- 使用STM32CubeMX开发三按键中断实验



集装箱房屋 呼叫中心系统 语音验证码 大专文凭 短信接口 erp管理系统 oa系统 推广

[查看评论](#)

暂无评论

您还没有登录,请[登录](#)或[注册](#)

\* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

## 核心技术类目

全部主题		Hadoop	AWS	移动游戏	Java	Android	iOS	Swift	智能硬件	Docker	OpenStack
VPN	Spark	ERP	IE10	Eclipse	CRM	JavaScript	数据库	Ubuntu	NFC	WAP	jQuery
BI	HTML5	Spring	Apache	.NET	API	HTML	SDK	IIS	Fedora	XML	LBS
Splashstop		UML	components	Windows	Mobile	Rails	QEMU	KDE	Cassandra	CloudStack	
FTC	coremail	OPhone	CouchBase	云计算	iOS6	Rackspace	Web App	SpringSide	Maemo		
Compuware		大数据	aptech	Perl	Tornado	Ruby	Hibernate	ThinkPHP	HBase	Pure	Solr
Angular	Cloud Foundry	Redis	Scala	Django	Bootstrap						