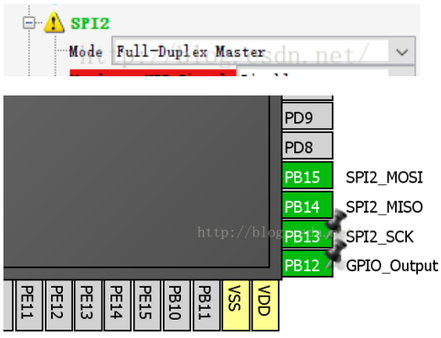
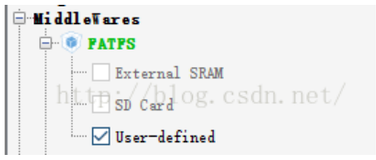
使用STM32CUBEMX生成FatFS代码，操作SPI FLASH

2016-03-10 21:50 280人阅读 评论(0) 收藏 举报

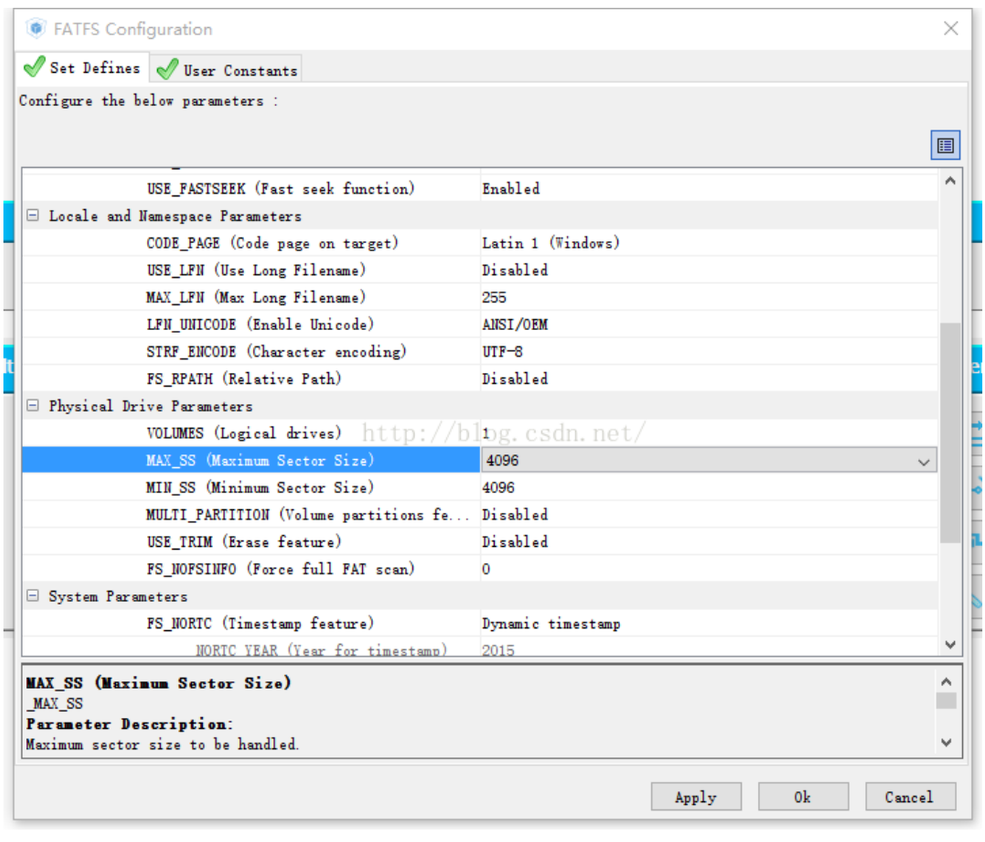
首先配置SPI，我的板子是SPI2连接到SPI FLASH 上，我的flash是W25Q64， PB12用来当CSN。



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：

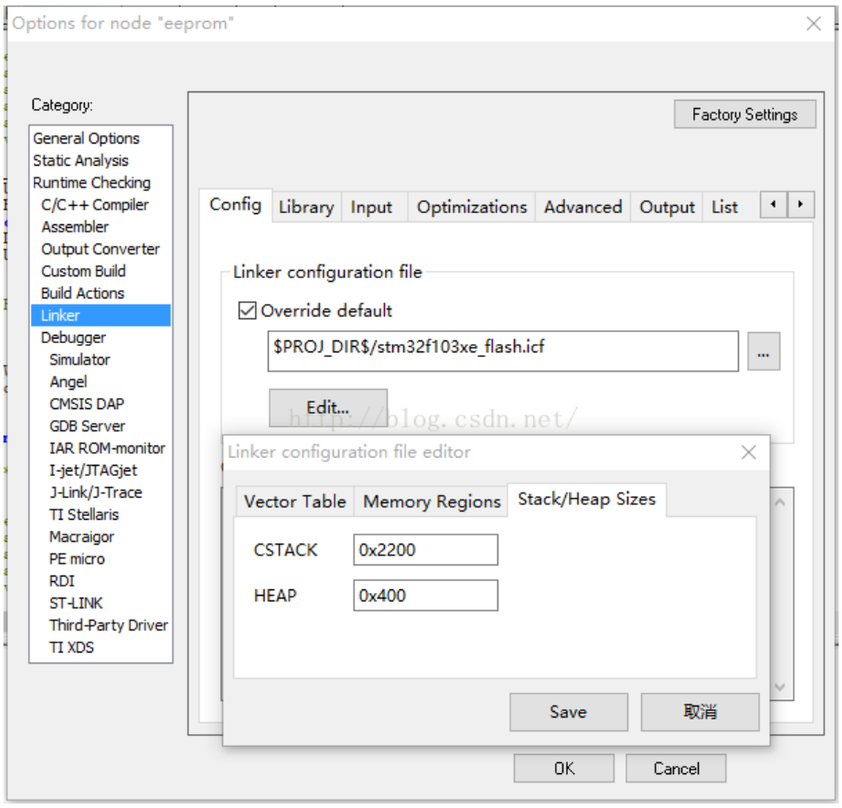
mengfei314

+ 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

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
#define PAGE_SIZE       256
#define SECTOR_SIZE     4096

#define SECTOR_COUNT    200
#define BLOCK_SIZE  65536
#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
/**
  * @brief  Reads Sector(s)
  * @param  pdrv: Physical drive number (0..)
  * @param  *buff: Data buffer to store read data
  * @param  sector: Sector address (LBA)
  * @param  count: Number of sectors to read (1..128)
  * @retval DRESULT: Operation result
  */
DRESULT USER_read (
    BYTE pdrv,      /* Physical drive nmuber to identify the drive */
    BYTE *buff,     /* Data buffer to store read data */
    DWORD sector,   /* Sector address in LBA */
    UINT count      /* Number of sectors to read */
)
{
  /* USER CODE HERE */
  UINT i = 0;
  for(i = 0; i < count; i ++)
  {
    W25_ReadSector(sector, buff);
    sector ++;
    buff += SECTOR_SIZE;
  }
  return RES_OK;
}
```

```cpp
/**
  * @brief  Writes Sector(s)
  * @param  pdrv: Physical drive number (0..)
  * @param  *buff: Data to be written
  * @param  sector: Sector address (LBA)
  * @param  count: Number of sectors to write (1..128)
  * @retval DRESULT: Operation result
  */
#if _USE_WRITE == 1
DRESULT USER_write (
    BYTE pdrv,          /* Physical drive nmuber to identify the drive */
    const BYTE *buff,   /* Data to be written */
    DWORD sector,       /* Sector address in LBA */
```

```
    UINT count          /* Number of sectors to write */
)
{
  /* USER CODE HERE */
  UINT i = 0;
  for(i = 0; i < count; i ++)
  {
    W25_WriteSector(sector, buff);
    sector ++;
    buff += SECTOR_SIZE;
  }
  return RES_OK;
}
#endif /* _USE_WRITE == 1 */
```

```cpp
/**
  * @brief  I/O control operation
  * @param  pdrv: Physical drive number (0..)
  * @param  cmd: Control code
  * @param  *buff: Buffer to send/receive control data
  * @retval DRESULT: Operation result
  */
#if _USE_IOCTL == 1
DRESULT USER_ioctl (
    BYTE pdrv,      /* Physical drive nmuber (0..) */
    BYTE cmd,       /* Control code */
    void *buff      /* Buffer to send/receive control data */
)
{
  DRESULT res = RES_OK;

  switch(cmd)
  {
    case CTRL_SYNC :
        break;

    case CTRL_TRIM:
        break;

    case GET_BLOCK_SIZE:
    *(DWORD*)buff = BLOCK_SIZE;
    break;

    case GET_SECTOR_SIZE:
    *(DWORD*)buff = SECTOR_SIZE;
        break;

    case GET_SECTOR_COUNT:
    *(DWORD*)buff = SECTOR_COUNT;
    break;

    default:
    res = RES_PARERR;
    break;
    }

  return res;
}
#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
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
void mount_disk(void)
{
  uint8_t res = f_mount(&fs, USER_Path, 0);
  if (res != FR_OK)
  {
    printf("FAILED: %d\n",res);
   return;
 }
  printf("MOUNT OK\n");
```

```
}

void format_disk(void)
{
  uint8_t res = 0;

  printf("PROCESSING...\n");
  res = f_mkfs(USER_Path, 1, 4096);
  if (res == FR_OK)
  {
    printf("OK!\n");
  }
  else
  {
    printf("failed with: %d\n",res);
  }
}

void create_file(void)
{
  FIL file;
  FIL *pf = &file;
  uint8_t res;

  res = f_open(pf, "0:/test.txt", FA_OPEN_ALWAYS | FA_WRITE);
  if (res == FR_OK)
  {
    printf("creat ok\n");
  }
  else
  {
    printf("creat failed\n");
    printf("error code: %d\n",res);
  }

  f_printf(pf, "hello fatfs!\n");

  res = f_close(pf);
    if (res != FR_OK)
    {
      printf("close file error\n");
      printf("error code: %d\n",res);
    }
}

void get_disk_info(void)
{
    FATFS fs;
    FATFS *fls = &fs;
    FRESULT res;
    DWORD fre_clust;

    res = f_getfree("/",&fre_clust,&fls);         /* Get Number of Free Clusters */
    if (res == FR_OK)
    {
                                                 /* Print free space in unit of MB (assuming 4096
        printf("%d KB Total Drive Space.\n"
               "%d KB Available Space.\n",
               ((fls->n_fatent-2)*fls->csize)*4,(fre_clust*fls->csize)*4);
    }
    else
    {
        printf("get disk info error\n");
        printf("error code: %d\n",res);
    }
}

void read_file(void)
{
  FIL file;
  FRESULT res;
  UINT bw;
  uint8_t rbuf[100] = {0};

  res = f_open(&file, "0:/test.txt", FA_READ);
  if (res != FR_OK)
  {
    printf("open error: %d\n",res);
    return;
  }
  f_read(&file, rbuf, 20, &bw);
  printf("%s\n", rbuf);

  res = f_close(&file);
  if (res != FR_OK)
  {
     printf("close file error\n");
     printf("error code: %d\n",res);
  }
}
```

顶 0 踩 0

快速回复

上一篇 使用STM32C 返回顶部 SB Mass Storage代码，通过SDIO读写TF卡

猜你在找

- 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

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 Unity
Splashtop 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