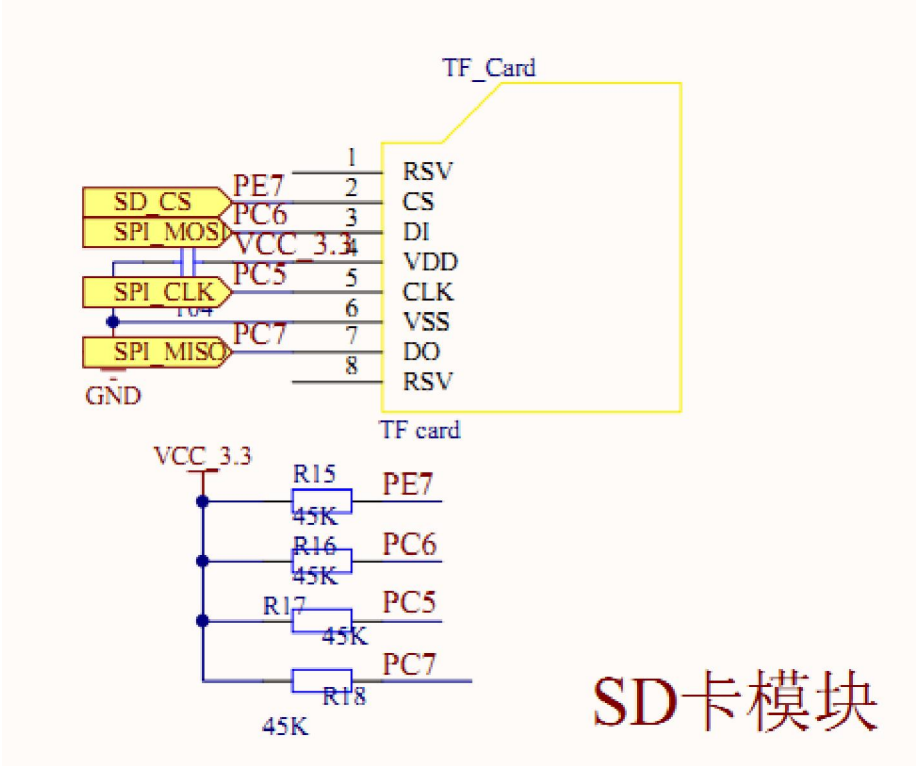
例程二十一 SDCard 基于 SPI 总线的读写

SDCard 现在可以说是非常流行的了，无论是手机，相机等等的电子产品中都能见到它的影子，所以对于 SDCard 的读写就变得十分之重要了，SDCard 有 2 中链接方法，一种是 SPI，另一种是 SDIO。2 种链接方式的读写速度相差很大的，基于 SPI 的读写只要 25Mbit/s，而基于 SDIO 方式便可以达到 100Mbit/s。在我的 STM8 开发板上没有 SDIO 的总线，那只能是基于 SPI 总线的读写。那我们先看看它的链接方式吧



这就 SPI 总线的接法。

SDCard 涉及的知识非常多，如果要具体到每一点去讲解的话，恐怕要将几天都讲不完，下面我主要讲一下简单实用，并且是在单片机的应用，其他的就留给读写再深入去了解。

SDCard 每个扇区大约有 10 万次的写寿命，读的话就没有限制，这应该够我们的吧。擦除操作可以加速写操作，因为在写之前会进行擦除操作。如果 SDCard 不在支持的电压范围内，他是忽略所有总线的传输。SD 卡总线采用的是单主多从结构，总线上所有卡公用时钟和电源线。主机可以通过不同的片选信号线依次分别访问每个卡，每个卡的 CID 寄存器中已预编程了一个唯一的卡标识号，来区分不同的卡，主机通过 READ_CID 的命令读取 CID 寄存器，CID 寄存器咋 SD 卡生产过程中的测试和格式化是被编程，主机只能读取改号。

卡状态分别存放在下面两个区域：

卡状态（Card Status），存放在一个 32 位状态寄存器，在卡响应主机命令时作为数据传送给主机。

SD 状态（SD_Status），当主机使用 SD_STATUS（ACMD13）命令时，512 位以一个数据块的方式发送给主机。SD_STATUS 还包括了和 BUS_WIDTH、安全相关位和扩展位等的扩展状态位。

数据读写的基本单元是一个字节，可以按要求组织成不同的块。

Block:块大小可以固定，也可以改变，**允许的块大小是实际大小等信息存储在 CSD 寄存器。**

Sector:和擦除命令相关，由几个块组成。Sector 的大小对每个设备是固定的，大小信息存储在 CSD 寄存器。

WP Group:写保护单位。大小包括几个 group，写保护由一位决定，对每个设备大小是固定的，存储在 CSD 寄存器。

SD 的读写操作有 2 种 Single Block Mode 和 Multiple Block Mode。
Single Block Mode ：主机根据事先定义的长度读写一个数据块。有发送模块生成一个 16 位的 CRC 校验码，接受端根据校验码进行检验。读操作的块的长度受设备 sector 大小（512byte）的限制，但是可以最小为一个字节。不对齐的访问时不允许的，每个数据块必须位于单个物理 sector 内。写操作的大小必须为 sector 大小，起始地址必须与 sector 边界对齐。
Multiple Block Mode：主机可以读写多个数据块（相同长度），根据命令中的地址读取或写入连续的内存地址。操作通过一个停止传输命令结束。写操作必须地址对齐。
数据保护 ：每个 sector 的数据通过 ECC 进行保护，在写 sector 时生成 ECC，在读 sector 是检验 ECC，如果发现错误，在传输前进行纠正。
数据擦除：SD 卡数据擦除的最小单位是 sector。为了加速擦除操作，多个 sector 可以同时擦除，为了方便选择，第一个指令包含起始地址，第二个指令包含结束地址，在地址范围内的所有 sector 将被擦除。

两种写保护方式可供选择，永久保护和临时保护，两种方式都可以通过 PROGRAM_CSD 指令进行设置。永久保护位一旦设置将无法清除。

所有 SD 卡的配置信息存储在 CSD 寄存器。通过 SEND_CSD 读取，PROGRAM_CSD 修改。

1.3 寄存器

名称	宽度	描述
CID	128	卡标识号
RCA	16	相对卡地址（Relative card address）:本地系统中卡的地址，动态变化，在主机初始化的时候确定 *SPI 模式中没有
CSD	128	卡描述数据:卡操作条件相关的信息数据
SCR	64	SD 配置寄存器:SD 卡特定信息数据
OCR	32	操作条件寄存器

主机通过重新上电来重置（reset）卡。卡有它自身检测上电的电路，当上电后卡状态切换到 idle 状态。也可以通过 GO_IDLE (CMD0)指令来重置。

上电后，包括热插入，卡进入 idle 状态。在该状态 SD 卡忽略所有总线操作直到接收到 ACMD41 命令。ACMD41 命令是一个特殊的同步命令，用来协商操作电压范围，并轮询所有的卡。除了操作电压信息，ACMD41 的响应还包括一个忙标志，表明卡还在 power-up 过程工作，还没有准备好识别操作，即告诉主机卡还没有就绪。主机等待(继续轮询)直到忙标志清除。单个卡的最大上电时间不能操作 1 秒。

上电后，主机开始时钟并在 CMD 线上发送初始化序列，初始化序列由连续的逻辑“1”组成。序列长度为最大 1 毫秒，74 个时钟或 supply-ramp-up 时间。额外的 10 个时钟(64 个时钟后卡已准备就绪)用来实现同步。

每个总线控制器必须能执行 ACMD41 和 CMD1。CMD1 要求 MMC 卡发送操作条件。在任何情况下，ACMD41 或 CMD1 必须通过各自的 CMD 线分别发送给每个卡。

下面我们来看看怎样初始话和读写 SDCard。还是从主函数看起

```
int main(void)
{
  /* Infinite loop */
  /*设置内部时钟16M为主时钟*/
  CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
  /*!<Set High speed internal clock  */
  Lcd_Configuration();
  Lcd_Initialize();
  SD_Init();
  Delay(100);
  Lcd_Clear(RED);
   while(SD_Idle_Sta())
  {
    Lcd_Write32X16String(0,0,0,0,GREEN,RED,"SD Card Failed!");
  }
  Lcd_Write32X16String(0,0,0,0,GREEN,RED,"SD Card Checked OK");
   Delay(0xfffff);

  SD_WriteSingleBlock(0x202, WriteT0SDcardData);
   Delay(0xfff);
  SD_ReadSingleBlock(0x202, WaitToRead);

  Lcd_WriteString(0,0,0,0,GREEN,RED,WaitToRead);
  while (1)
  {
    /*  添加你的代码  */
```

SD_Init();SD 的初始话就是 SPI 的初始话，跟在 SPI_Flash 的例程配置一样的，注意的一点是要把 W25X16 的片选信号线拉高。

```
void SD_Init(void)
{
  GPIO_Init(GPIOE,GPIO_PIN_6, GPIO_MODE_OUT_PP_HIGH_FAST);/*为防止与W25X16产生冲突*/
  SPI_FLASH_Init();
  GPIO_WriteHigh(GPIOE,GPIO_PIN_6);/*为防止与W25X16产生冲突*/
}
```

最主要就是这个函数 u8 SD_Idle_Sta(void)。下面就是函数原型

```
/*******************************************************************************
                        把SD卡设置到挂起模式
                        返回值:0,成功设置
                               1,设置失败
*******************************************************************************/
*******************************************************************************/
u8 SD_Idle_Sta(void)
{
  u16 i;
  u8 retry,r1;
  retry = 0;
  do
  {
    for(i=0;i<10;i++)
    (void)SPI_FLASH_SendByte(0xff);
    r1 = SD_SendCommand(0, 0,0x95);                    //发idle命令
    if(++retry>100) return 1;                  //超时退出
  }while(r1 != 0x01);
  retry = 0;
  do
  {
    r1 =SD_SendCommand(1, 0,0x95);              //发active命令
    if(++retry>100) return 1;           //超时退出
  } while(r1);

  r1 = SD_SendCommand(59, 0,0x95);               //关crc

  r1 = SD_SendCommand(16, 512,0x95);             //设扇区大小512

  return 0;                                 //正常返回

}
```

只有使 SD 卡设置为挂起的模式，才能向卡里面读写数据。函数里面主要是向 SD 卡命令，和等待 SD 卡的响应。
下面在看看发命令的函数

```
/*******************************************************************************
                    向SD卡发送一个命令
                    输入: u8 cmd   命令
                          u32 arg  命令参数
                          u8 crc   crc校验值
                    返回值:SD卡返回的响应
*******************************************************************************/
u8 SD_SendCommand(u8 cmd, u32 arg, u8 crc)
{
  u8 r1;
  u8 Retry=0;
  Set_SD_CS;
  SPI_FLASH_SendByte(0xff);//高速写命令延时
  SPI_FLASH_SendByte(0xff);
  SPI_FLASH_SendByte(0xff);
  //片选端置低，选中SD卡
  Clr_SD_CS;
  //发送
  SPI_FLASH_SendByte(cmd | 0x40);//分别写入命令
  SPI_FLASH_SendByte(arg >> 24);
  SPI_FLASH_SendByte(arg >> 16);
  SPI_FLASH_SendByte(arg >> 8);
  SPI_FLASH_SendByte(arg);
  SPI_FLASH_SendByte(crc);
      //等待响应，或超时退出
```

```
  r1=SPI_FLASH_SendByte(0xFF);
  while(r1==0xFF)
  {
    r1=SPI_FLASH_SendByte(0xFF);
    Retry++;
    if(Retry>200)break;
  }
        //关闭片选
    Set_SD_CS;
        //在总线上额外增加8个时钟，让SD卡完成剩下的工作
  //   SPI_FLASH_SendByte(0xFF);
        //返回状态值
    return r1;
}

/*******************************************************************************
写SD卡的一个block
输入:u32 sector 取地址（sector值，非物理地址）
     u8 *buffer 数据存储地址（大小至少512byte）
返回值:0:  成功
      other: 失败
*******************************************************************************/
u8 SD_WriteSingleBlock(u32 sector, u8* buffer)
{
  u8 r1;
  u16 i;
  //byte retry=0;
  if(sector<1) return 0xff;          //为了保护SD卡引导区，跳过该区
  r1 =SD_SendCommand(CMD24, sector<<9,0x95);   //写命令
  if(r1 != 0x00) return r1;

  Clr_SD_CS;

  (void)SPI_FLASH_SendByte(0xff);
  (void)SPI_FLASH_SendByte(0xff);
  (void)SPI_FLASH_SendByte(0xff);
  (void)SPI_FLASH_SendByte(0xfe);          //发开始符
  for(i=0; i<512; i++)                //送512字节数据
  {
    (void)SPI_FLASH_SendByte(*buffer++);
  }
```

```
  (void)SPI_FLASH_SendByte(0xff);
  (void)SPI_FLASH_SendByte(0xff);

  r1 = SPI_FLASH_SendByte(0xff);

  if((r1&0x1f) != 0x05)              //等待是否成功
  {
    Set_SD_CS;
    return r1;
  }
  //等待操作完
  while(!SPI_FLASH_SendByte(0xff));

    Set_SD_CS;

  return 0;
}
```

写一个 Block 要已边界对齐，上面已经介绍过

```
/*****************************************************************************
读SD卡的一个block
输入:u32 sector 取地址（sector值，非物理地址）
     u8 *buffer 数据存储地址（大小至少512byte）
返回值:0:  成功
      other: 失败
*****************************************************************************/
u8 SD_ReadSingleBlock(u32 sector, u8 *buffer)
{
 u8  r1;
 u16 i;
  r1 =SD_SendCommand(CMD17, sector<<9,0x95);//读命令
  if(r1 != 0x00)return r1;

  Clr_SD_CS;
  //等数据的开始
  while(SPI_FLASH_SendByte(0xff) != 0xfe);//if(++retry > 50){SD_deSel();return 1;}
  for(i=0; i<512; i++)//读512个数据
  {
    *buffer++ = SPI_FLASH_SendByte(0xff);
  }
  (void)SPI_FLASH_SendByte(0xff);//伪crc
  (void)SPI_FLASH_SendByte(0xff);
  Set_SD_CS;

  return 0;
}
```

读一个 Block，详解已在注释了

```
/*****************************************************************************
读SD卡的多个block
输入:u32 sector 取地址（sector值，非物理地址）
     u8 *buffer 数据存储地址（大小至少512byte）
     u8 count   N个block
返回值:0:  成功
      other: 失败
*****************************************************************************/
u8 SD_ReadMultiBlock(u32 sector, u8 *buffer, u8 count)
{
   u8  r1;
   u16 i;
  r1 =SD_SendCommand(CMD18, sector<<9,0x95);//读多个Block命令
  if(r1 != 0x00)return r1;
  Clr_SD_CS;
  while(SPI_FLASH_SendByte(0xff) != 0xfe);
  do
  {
    for(i=0; i<512; i++)//读512个数据
    {
      *buffer++ = SPI_FLASH_SendByte(0xff);
    }
    (void)SPI_FLASH_SendByte(0xff);//伪crc
    (void)SPI_FLASH_SendByte(0xff);
  }while(--count);

  //全部传输完毕，发送停止命令
  SD_SendCommand(CMD12, 0, 0);
  //释放总线
  Set_SD_CS;
 (void)SPI_FLASH_SendByte(0xff);
  if(count != 0)return count;    //如果没有传完，返回剩余个数
  else return 0;
}
```

```
/*****************************************************************************
写入SD卡的N个block
输入:u32 sector 扇区地址（sector值，非物理地址）
     u8 *buffer 数据存储地址（大小至少512byte）
     u8 count 写入的block数目
返回值:0:  成功
       other: 失败
*****************************************************************************/
u8 SD_WriteMultiBlock(u32 sector, const u8 *data, u8 count)
{
    u8 r1;
    u16 i;
    if(sector<1) return 0xff;          /*为了保护SD卡引导区，跳过该区*/
    r1 =SD_SendCommand(CMD25, sector<<9,0x95);  /*发多块写入指令命令*/
    if(r1 != 0x00)return r1;  /*应答不正确，直接返回     */
    Clr_SD_CS;/*开始准备数据传输    */
    (void)SPI_FLASH_SendByte(0xff);/*先放3个空数据，等待SD卡准备好*/
    (void)SPI_FLASH_SendByte(0xff);
        /*下面是N个sector写入的循环部分*/
    do
    {
          /*放起始令牌0xFC 表明是多块写入*/
          (void)SPI_FLASH_SendByte(0xFC);
            /*放一个sector的数据*/
          for(i=0;i<512;i++)
          {
```

```
            (void)SPI_FLASH_SendByte(*data++);
          }
          //发2个Byte的dummy CRC
          (void)SPI_FLASH_SendByte(0xff);
          (void)SPI_FLASH_SendByte(0xff);

          //等待SD卡应答
          r1 = SPI_FLASH_SendByte(0xff);
          if((r1&0x1F)!=0x05)
          {
            Set_SD_CS;     /*如果应答为报错，则带错误代码直接退出*/
            return r1;
          }

      }while(--count);//本sector数据传输完成
        /*发结束传输令牌0xFD*/
        r1 = SPI_FLASH_SendByte(0xFD);
        if(r1==0x00)
        {
          count =  0xfe;
        }

        Set_SD_CS;
        return count;    //返回count值，如果写完则count=0，否则count=1
}
/*****************************************************************
在指定扇区,从offset开始读出bytes个字节
输入:unsigned long address 扇区地址（sector值，非物理地址）
     unsigned char *buf     数据存储地址（大小<=512byte）
     unsigned int offsett  在扇区里面的偏移量
     unsigned int bytes  要读出的字节数
返回值:0:  成功
       other: 失败
*****************************************************************/
u8 SD_Read_Bytes(unsigned long address,unsigned char *buf,unsigned int offset,unsigned int bytes)
{
    u8 r1;u16 i=0;
    r1=SD_SendCommand(CMD17, address<<9,0x95);//发送读扇区命令
    if(r1)return r1;  //应答不正确，直接返回
    Clr_SD_CS;//选中SD卡
    if(SD_GetResponse(0xFE))//等待SD卡发回数据起始令牌0xFE
    {
      Set_SD_CS; //关闭SD卡
      return 1;//读取失败
    }
    for(i=0;i<offset;i++)SPI_FLASH_SendByte(0xff);//跳过offset位
    for(;i<offset+bytes;i++)*buf++=SPI_FLASH_SendByte(0xff);//读取有用数据
    for(;i<512;i++) SPI_FLASH_SendByte(0xff);     //读出剩余字节
    SPI_FLASH_SendByte(0xff);//发送伪CRC码
    SPI_FLASH_SendByte(0xff);
    Set_SD_CS;//关闭SD卡
    return 0;
```

本例程只是介绍 SDCard 的应用层，底层的结构暂时不讲，如果想了解更多的关于 SDCard 方面的知识，可以到网上下载更多的资料了解。我们搞单片机开发的只要能在应用层有所了解就行，只要能利用起来就行，对于 SDCard 底层的，我还有有待提高,大家一起学习交流吧。

把该例程下载进去，在风驰电子 STM8 开发板的 SDCard 套装上 SDCard，放上液晶，在液晶上就会看到从 SDCard 里面读出来的数据。

风驰电子祝您学习愉快