基于 MCF51QE128 的 SD 卡接口设计

1 SD 卡标准

SD 卡标准是 SD 卡协会针对可移动存储设备设计专利并授权的一种标准，主要用于制定卡的外形尺寸、电气接口和通信协议。

1．1 SD 卡引脚功能

SD 卡的外形如图 1 所示，引脚功能如表 1 所列。SD 卡的引脚具有双重功能，既可工作在 SD 模式，也可工作在 SPI 模式。不同的模式下，引脚的功能不同。

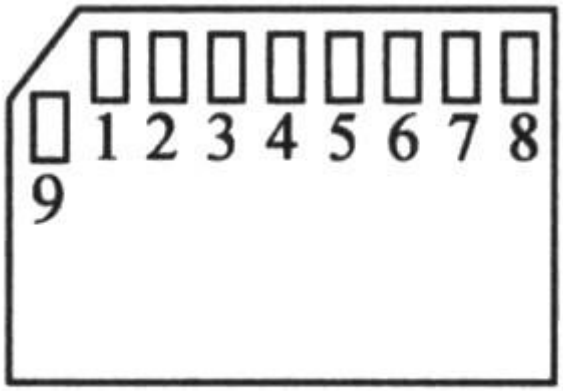

图 1　SD 卡外形

表 1　SD 卡引脚功能

引脚号	名　称	功能(SD 模式)	功能(SPI 模式)
1	DAT3/CS	数据线 3	片选/从选(SS)
2	CMD/DI	命令线	主出从入(MOSI)
3	VSS1	电源地	电源地
4	VDD	电源	电源
5	CLK	时钟	时钟(SCK)
6	VSS2	电源地	电源地
7	DAT0/DO	数据线 0	主入从出(MISO)
8	DAT1/IRQ	数据线 1	保留
9	DAT2/NC	数据线 2	保留

SD 模式多用于对 SD 卡读写速度要求较高的场合，SPI 模式则是以牺牲读写速度换取更好的硬件接口兼容性。由于 SPI 协议是目前广泛流行的通信协议，大多数高性能单片机都配备了 SPI 硬件接口，硬件连接相对简单，因此，在对 SD 卡读写速度要求不高的情况下，采用 SPI 模式无疑是一个不错的选择。

1．2 SPI 模式

SPI 模式是一种简单的命令响应协议，主控制器发出命令后，SD 卡针对不 S 同的命令返回对应的响应。

SD 卡的命令列表都是以 CMD 和 ACMD 开头，分别指通用命令和专用命令，后面接命令的编号。例如，CMD17 就是一个通用命令，用来读单块数据。

在 SPI 模式中，命令都是以如下的 6 字节形式发送的：

第 1 字节			第 2～5 字节	第 6 字节	
0	1	命令号	参数	CRC 校验	1

每帧命令都以“01”开头，然后是 6 位命令号和 4 字节的参数(高位在前，低位在后)，最后是 7 位 CRC 校验和 1 位停止位“1”。

SD 卡的每条命令都会返回对应的响应类型。在 SPI 模式下，共有 3 种响应类型：R1、R2 和 R3，分别占 1、2 和 3 个字节。这里仅列出了 R1 响应的格式，如表 2 所列。当出现表中所描述的状态时，相应的位置 1。R2 和 R3 的第 1 个字节格式与 R1 完全一样，详细内容请参考 SD 卡标准。

表 2　R1 响应格式

位	描　述
7	起始位,0
6	参数错误
5	地址错误
4	擦除顺序错误
3	CRC 错误
2	非法命令
1	擦除复位
0	空闲状态

2 硬件设计

本设计选用 Freescale 公司的 32 位低功耗微控制器 MCF51QE128，采用 SPI 模式实现与 SD 卡的接口。

由于 MCF51QE128 是一款低功耗的微控制器，工作电压的典型值为 3．6 V，与 SD 卡的工作电压兼容，因而可以直接与 SD 卡连接，无需电平转换电路。这里选用的是 MCF51 QE128 的第 2 个 SPI 口，硬件连接如图 2 所示。

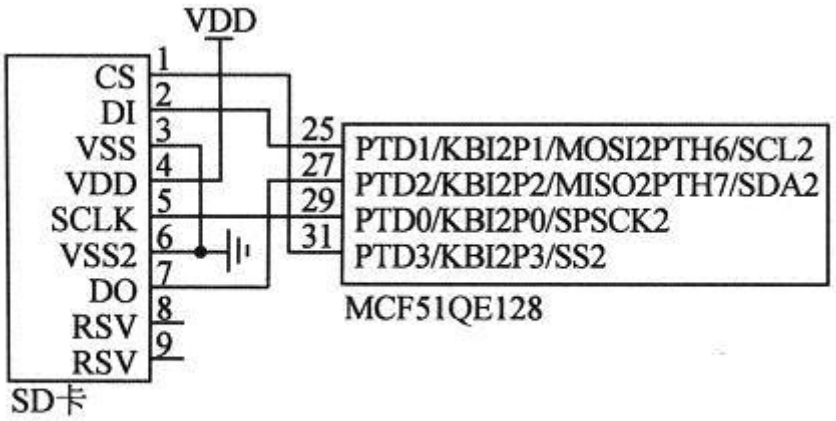

图 2　SD 卡与 MCF51QE128 的硬件连接

3 软件实现

软件部分主要实现 MCF51QE128 的初始化、底层 SPI 通信，以及 SD 卡的通用写命令、初始化和单块数据的读写等功能。

3．1 MCF51QE128 的初始化

在与 SD 卡通信之前，首先需要配置 MCF51QE128，并初始化 SPI 端口。代码如下：

```
//定义片选信号
#define select_card()     PTDD_PTDD3 = 0
#define unselect_card()   PTDD_PTDD3 = 1
void MCU_Init(void) {
    SOPT1 = 0x23;      //关看门狗
    SCGC1 = 0x00;      //禁用其他外设的总线时钟
    SCGC2 = 0x02;      //开 SPI2 模块的总线时钟
    PTDDD = 0x08;      //SPI 片选信号由软件设置
}
void SPI_Init (void) {
    SPI2BR = 0x44;     //设初始 SPI 时钟为 400 kHz
    SPI2C1 = 0xD0;
                  //SPI 中断允许,系统中断允许,主模式选择
    SPI2C2 = 0x00;
}
```

3．2 底层 SPI 通信

底层的 SPI 通信是实现最终读写的关键。由于 MCF51QE128 自带 SPI 硬件接口，因此只需要读写 SPI 数据寄存器的值。这里自定了 byte、word 和 dword 三种数据类型，分别对应于 8 位、16 位和 32 位数据。代码如下：

```
byte SPI_ReadByte(void) {          //SPI 读字节函数
    while (!SPI2S_SPTEF);          //等待,直到发送寄存器为空
    SPI2D = 0xff;                  //接收 1 字节数据
    while (!SPI2S_SPRF);
    return SPI2D;
```

```
}
byte SPI_WriteByte(byte val) {    //SPI 写字节函数
    while ((! SPI2S_SPTEF)&&(! PTDD_PTDD3));
                                  //等待,直到发送寄存器为空
    SPI2D = val;                  //发送数据
}
```

3．3 SD 卡的通用写命令

由于 SD 卡的命令具有统一的格式，因此可以用一个通用的写命令函数来实现所有命令的发送。另外，考虑到多数命令的响应类型都是 R1，这里的通用写命令函数所接收的响应类型默认为 R1。函数代码如下：

```
byte SD_SendCommand_R1(byte cmd, dword arg) {
    byte i,r1;
    SPI_WriteByte(0xff);                    //等待几个时钟周期
    SPI_WriteByte((byte)(cmd|0x40));        //写入命令号
    SPI_WriteByte((byte)(arg>>24));         //4 字节命令参数
    SPI_WriteByte((byte)(arg>>16));
    SPI_WriteByte((byte)(arg>>8));
    SPI_WriteByte((byte)(arg));
    SPI_WriteByte((byte)(cmd == 0x00? 0x95 : 0xff));
                                            //CRC 校验和
    for(i = 0; i < 10; ++i) {               //接收响应
      r1 = SPI_ReadByte();
      if(r1 != 0xff) break;
    }
    return r1;
}
```

3．4 SD 卡的初始化

SD 卡的初始化要遵循一定的步骤。首先将 SPI 时钟降低到 400 kHz，等待至少 74 个时钟周期。接着拉低片选信号，并发送 CMD0 命令，对 SD 卡进行复位并使其进入 SPI 模式，这里需要正确的 CRC 校验，校验字节为 0x95。若 SD 卡进入空闲状态(即接收响应为 0x01 时)，则发送 CMD1 命令，激活卡的初始化过程，此时响应为 0x00。然后设置块的长度，一般为 512 字节。最后将片选拉高并将 SPI 时钟设为最大值，以保证最大的读写速度。SD 卡初始化过程如图 3 所示。

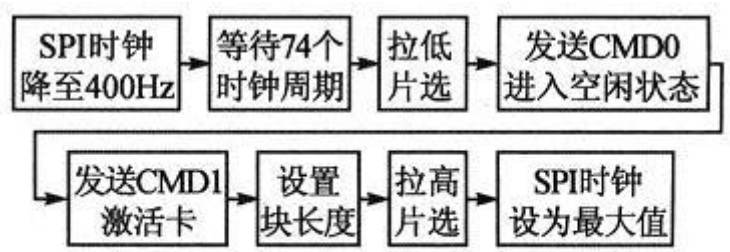

图 3　SD 卡初始化过程

SD 卡初始化代码如下：

```
byte SD_Init(void) {
    word i;
    byte response;
    for(i=0;i<10;i++)  SPI_ReadByte();
                                //等待至少 74 个时钟周期
    select_card();              //片选拉低
    for(i = 0; ; i++) {
      response = SD_SendCommand_R1(0x00,0);
                                //发送 CMD0
      if(response == 0x01) break;  //进入空闲状态
      if(i == 0x1ff) {
        unselect_card();
        return 0;
      }
    }
    for(i = 0; ; i++) {
      response = SD_SendCommand_R1(0x01,0);
                                //发送 CMD1,激活卡的初始化
      if(response == 0x00) break;
      if(i == 0x1ff) {
        unselect_card();
        return 0;
      }
    }
    if(SD_SendCommand_R1(0x10, 512)) {
                                //设置块长度为 512 字节
      unselect_card();
      return 0;
    }
    unselect_card();            //片选拉高
    //SPI2BR = 0x40;            //选择最高 SPI 时钟
    return 1;
}
```

3．5 SD卡单块数据读写

SPI模式支持单块和多块数据的读写操作，可通过发送相应的命令来实现。读单块数据的操作过程如图4所示。拉低片选后，首先由主控制器 MCF51QE128发送读单块数据命令CMD17，然后等待SD卡的响应。当收到数据块开始标志0xfe后，开始从SD卡读取512字节的数据，最后读取2字节的CRC校验位。

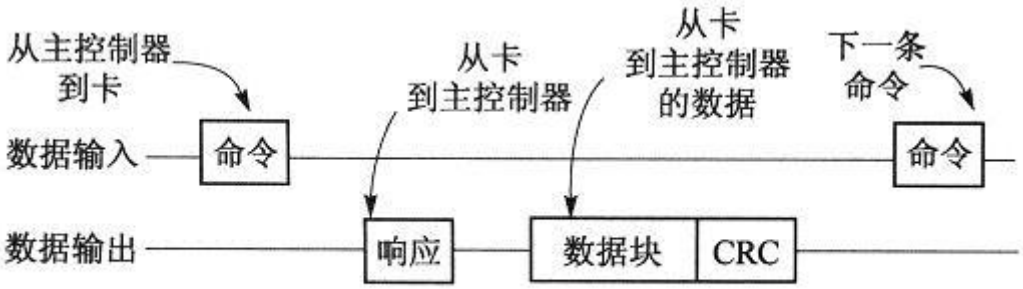

图4　读单块数据操作

读单块数据的函数代码如下：

```
byte SD_ReadSingleBlock(byte data[], dword sector) {
    word i;
    select_card();
    if(SD_SendCommand_R1(0x11, sector)) {
                    //发送读单块数据命令 CMD17
      unselect_card();
      return 0;
    }
    while(SPI_ReadByte() != 0xfe) ;
                    //等待,直到收到数据块开始标志 0xfe
    for(i = 0; i < 512; i++) data[i] = SPI_ReadByte();
                    //读 512 字节数据块
    SPI_ReadByte();
                    //读 16 位 CRC 校验
    SPI_ReadByte();
    unselect_card();
    SPI_ReadByte();
    return 1;
}
```

写单块数据的操作过程与读操作类似，如图 5 所示。拉低片选后同样由主控制器 MCF51QE128 发送写单块数据命令 CMD24，SD 卡正确响应后发送数据块开始标志 0xfe，接着发送 512 字节数据块和 2 字节 CRC 校验。

写入数据后，SD 卡会发送 1 字节的数据响应来反馈数据写入的情况，其格式如图 6 所示。当数据正确写入 SD 卡后，数据响应为 0x05。最后读数据总线，写数据忙时等待，直到总线为高电平。

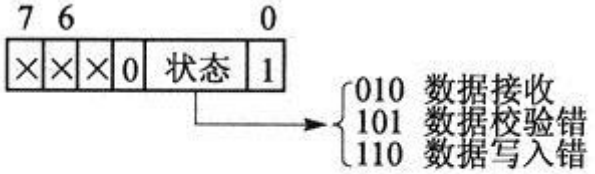

图 6　数据响应格式

写单块数据的函数代码如下：

```
byte SD_WriteSingleBlock(byte data[], dword sector) {
    word i;
    byte response;
    select_card();
    if(SD_SendCommand_R1(0x18, sector)) {
                                    //发送写单块数据命令 CMD24
  unselect_card();
  return 0;
}
SPI_WriteByte(0xfe);        //发送数据块开始标志 0xfe
for(i = 0; i < 512; i++)  SPI_WriteByte(data[i]);
                            //写 512 字节数据块
SPI_WriteByte(0xff);        //写 16 位 CRC 校验
SPI_WriteByte(0xff);
for(i = 0; ; i++) {         //读数据响应,判断数据是否
                            //正确写入
  response = SPI_ReadByte();
  if((response & 0x0f) == 0x05) break;
  if(i == 0x1ff) {
    unselect_card();
    return 0;
  }
}
while(SPI_ReadByte() != 0xff) ; //写数据忙时,等待
SPI_ReadByte();
unselect_card();
return 1;
}
```

结 语

SD 卡是目前广泛应用的可擦除的大容量存储设备，其接口设计可作为各类嵌入式系统中存储单元的一般解决方案。本文结合 SD 卡标准的相关技术，基于 MCF51QE128 微控制器完成了硬件接口和底层通信软件的设计。在此基础上，可进一步构建文件系统，实现对存储数据更有效的管理。