# Part 5 SD Card

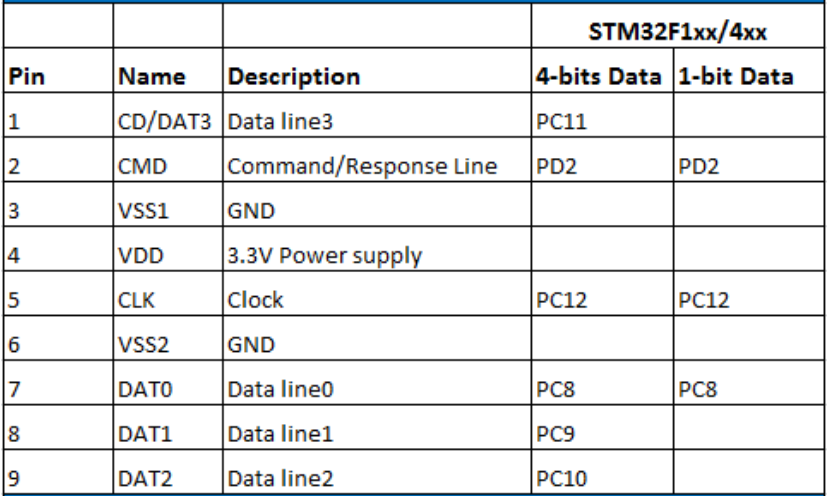
首先介绍SD Card的2个模式和对应的9个引脚的功能

9

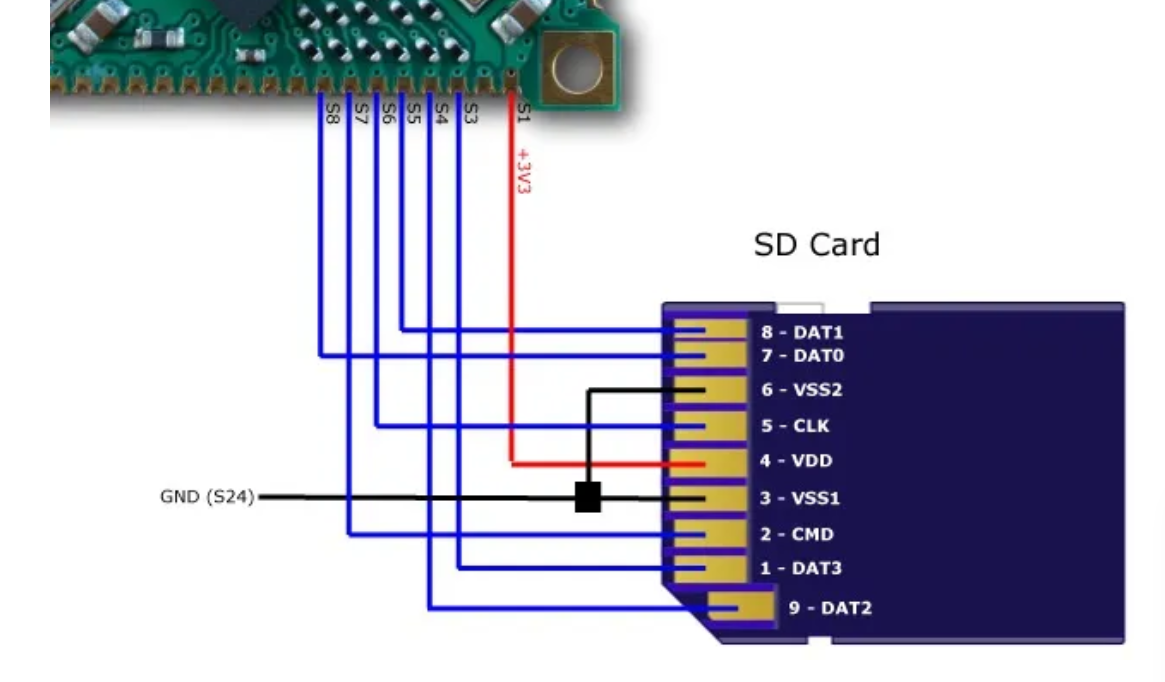
1 2 3 4 5 6 7 8

### 模式1 SDIO（Secure Digital Input and Output安全数字输入输出接口）模式

其中数据传输模式也存在两种为4bits和1bit，若使用4bits方式传输数据，则DAT0-3分别使用7，8，9，1接口，若使用1bits方式传输数据，则DAT0使用7接口

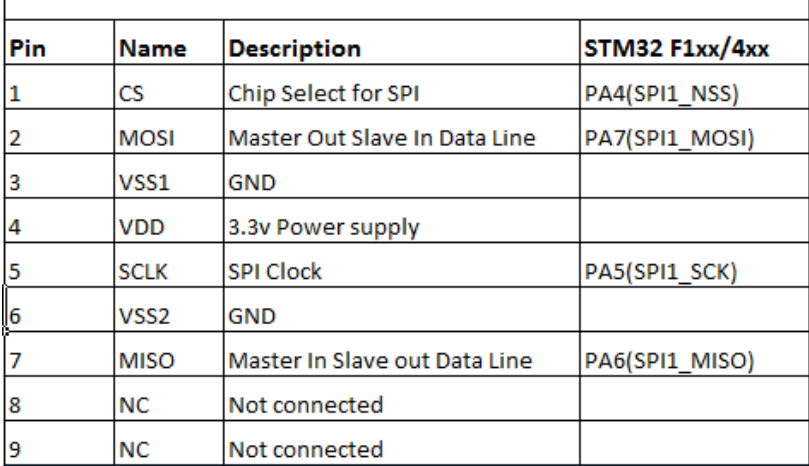


下图为SD卡与单片机的连接方式

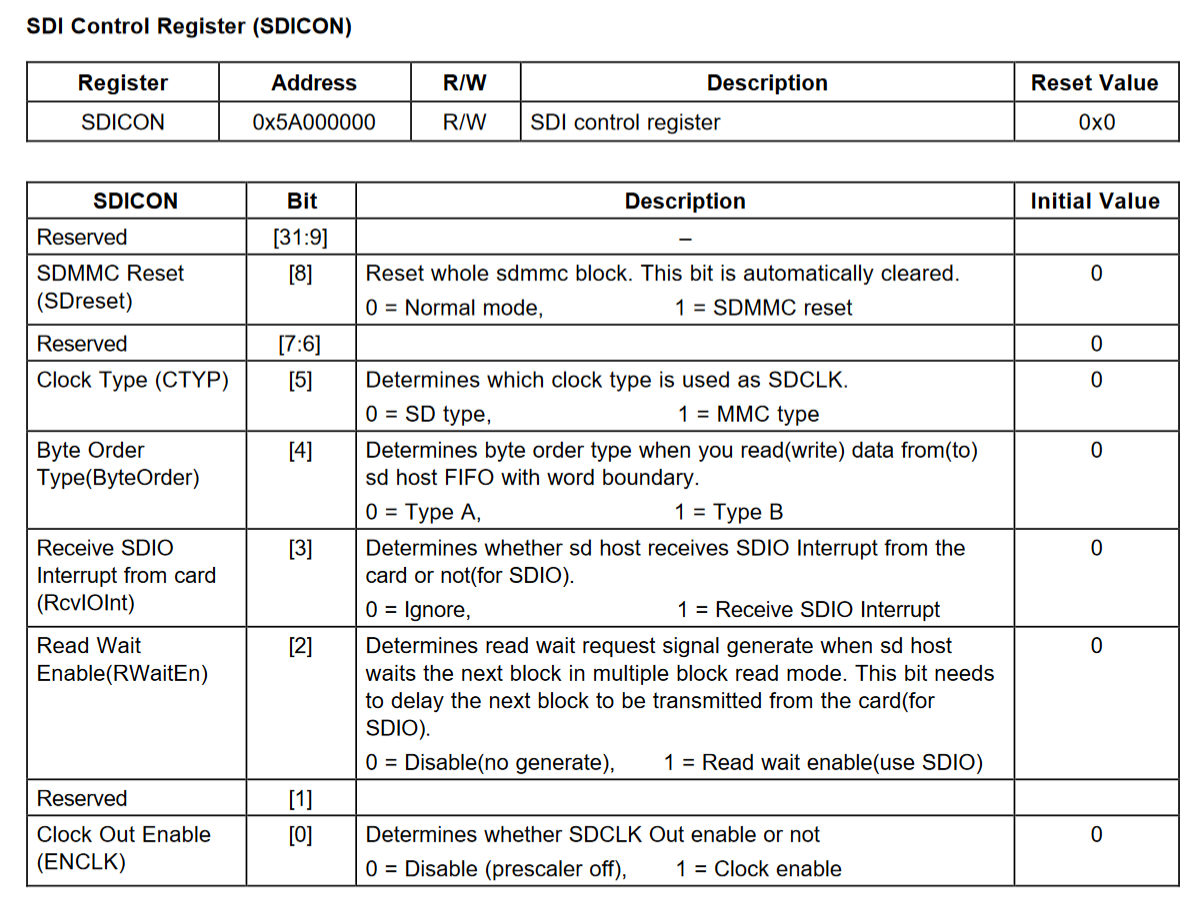


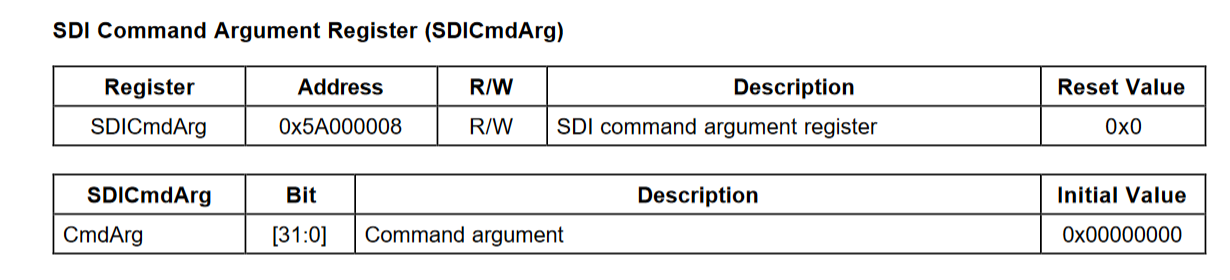
### 模式2 SPI（Serial Peripheral Interface串行外围接口）模式

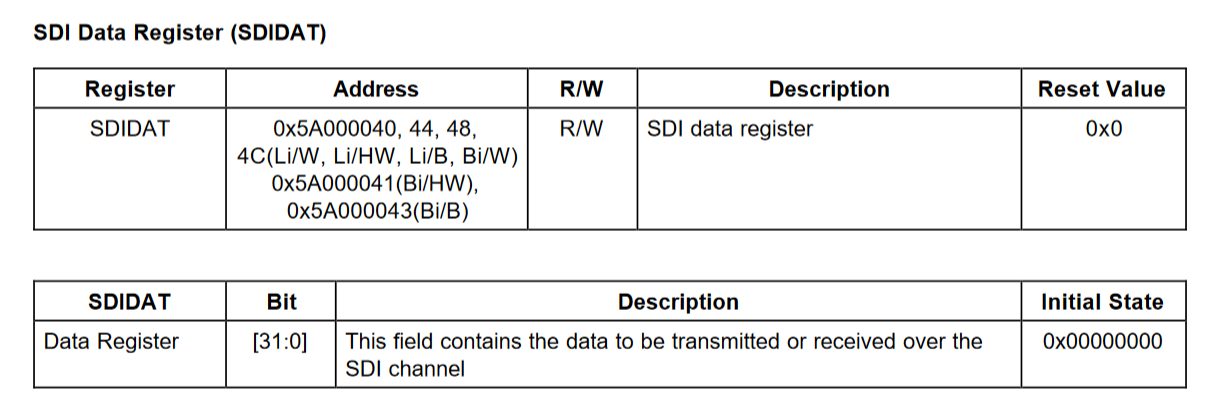
这里只用到单片机的4根信号线



SDI中的特殊寄存器







## 代码部分解读：

程序主要使用SPI模式进行初始化，读写SD卡，有关报警信号的精确信息。

这里程序流程比较简单：

1）配置串口，用作程序的调试输出

2）填充将要给SD卡写入数据的数组send\_data（利用Part7的RTC日期时间子程序）。

3）初始化SD卡，根据返回SD\_Init()返回值确定SD卡初始化是否完成。

4）进行单块读写

主函数的代码如下：

int main(void)

{

u16 i;

USART1\_Config();

send\_data[1536] = RTC.get\_time()

switch (SD\_Init())

{

case 0:

USART1\_Puts("\r\nSD Card Init Success!\r\n");

break;

case 1:

USART1\_Puts("Time Out!\n");

break;

case 99:

USART1\_Puts("No Card!\n");

break;

default: USART1\_Puts("unknown err\n");

break;

}

SD\_WriteSingleBlock(30, send\_data);

SD\_ReadSingleBlock(30, receive\_data);

if (Buffercmp(send\_data, receive\_data, 512))

{

USART1\_Puts("\r\n single read and write success \r\n");

//USART1\_Puts(receive\_data);

}

while (1);

}

初始化SD卡的函数

u8 SD\_Init(void)

{

u16 i;

u8 r1;

u16 retry;

u8 buff[6];

SPI\_ControlLine();

//SD卡初始化时时钟不能超过400KHz

SPI\_SetSpeed(SPI\_SPEED\_LOW);

//CS为低电平，片选置低，选中SD卡

SD\_CS\_ENABLE();

//纯延时，等待SD卡上电稳定

for (i = 0; i < 0xf00; i++);

//先产生至少74个脉冲，让SD卡初始化完成

for (i = 0; i < 10; i++)

{

//参数可随便写，经过10次循环，产生80个脉冲

SPI\_ReadWriteByte(0xff);

}

//-----------------SD卡复位到idle状态----------------

//循环发送CMD0，直到SD卡返回0x01,进入idle状态

//超时则直接退出

retry = 0;

do

{

//发送CMD0，CRC为0x95

r1 = SD\_SendCommand(CMD0, 0, 0x95);

retry++;

} while ((r1 != 0x01) && (retry < 200));

//跳出循环后，检查跳出原因，

if (retry == 200) //说明已超时

{

return 1;

}

//如果未超时，说明SD卡复位到idle结束

//发送CMD8命令，获取SD卡的版本信息

r1 = SD\_SendCommand(CMD8, 0x1aa, 0x87);

//下面是SD2.0卡的初始化

if (r1 == 0x01)

{

// V2.0的卡，CMD8命令后会传回4字节的数据，要跳过再结束本命令

buff[0] = SPI\_ReadWriteByte(0xFF);

buff[1] = SPI\_ReadWriteByte(0xFF);

buff[2] = SPI\_ReadWriteByte(0xFF);

buff[3] = SPI\_ReadWriteByte(0xFF);

SD\_CS\_DISABLE();

//多发8个时钟

SPI\_ReadWriteByte(0xFF);

retry = 0;

//发卡初始化指令CMD55+ACMD41

do

{

r1 = SD\_SendCommand(CMD55, 0, 0);

//应返回0x01

if (r1 != 0x01)

return r1;

r1 = SD\_SendCommand(ACMD41, 0x40000000, 1);

retry++;

if (retry > 200)

return r1;

} while (r1 != 0);

//初始化指令发送完成，接下来获取OCR信息

//----------鉴别SD2.0卡版本开始-----------

//读OCR指令

r1 = SD\_SendCommand\_NoDeassert(CMD58, 0, 0);

//如果命令没有返回正确应答，直接退出，返回应答

if (r1 != 0x00)

return r1;

//应答正确后，会回传4字节OCR信息

buff[0] = SPI\_ReadWriteByte(0xFF);

buff[1] = SPI\_ReadWriteByte(0xFF);

buff[2] = SPI\_ReadWriteByte(0xFF);

buff[3] = SPI\_ReadWriteByte(0xFF);

//OCR接收完成，片选置高

SD\_CS\_DISABLE();

SPI\_ReadWriteByte(0xFF);

//检查接收到的OCR中的bit30位（CSS），确定其为SD2.0还是SDHC

//CCS=1：SDHC CCS=0：SD2.0

if (buff[0] & 0x40)

{

SD\_Type = SD\_TYPE\_V2HC;

}

else

{

SD\_Type = SD\_TYPE\_V2;

}

//-----------鉴别SD2.0卡版本结束-----------

SPI\_SetSpeed(1); //设置SPI为高速模式

}

}

读写数据的函数

u8 SPI\_ReadWriteByte(u8 TxData)

{

while (SPI\_I2S\_GetFlagStatus(SPI1, SPI\_I2S\_FLAG\_TXE) == RESET);

SPI\_I2S\_SendData(SPI1, TxData);

while (SPI\_I2S\_GetFlagStatus(SPI1, SPI\_I2S\_FLAG\_RXNE) == RESET);

return SPI\_I2S\_ReceiveData(SPI1);

}