利用 SD 存储介质扩展 MAXQ2000 的非易失性数据存储空间

摘要：本文讨论如何使用安全数字(SD)媒体格式扩展 MAXQ2000 的非易失数据存储器。

低功耗、低噪声的 MAXQ2000 微控制器适合于多种应用。MAXQ2000 在闪存中存储非易失性数据，并和用户代码空间共享 32k 字(64kB)的闪存容量。但如果你的应用需要更多的非易失性存储器怎么办呢？这篇文章说明了如何使用安全数码(SD™)存储介质来扩展 MAXQ2000 的非易失性数据存储器。

外部存储器的设计考虑

对于你的应用设计来说，首要应该考虑电源电压和电流的要求。在典型的 MAXQ2000 应用中，可采用一个双路线性稳压器，从而在所选的时钟频率下使处理器工作于尽可能低的核心电压(V_{DD})下。MAXQ2000 的 V_{DD} 电压可低至 1.8V。V_{DDIO} 为 MAXQ2000 的 I/O 引脚供电，允许的电压范围最低至 V_{DD}，最高可达 3.6V。可接受的外部存储器电流消耗受限于电源的额定电流，对于电池供电设备，则电流消耗受限于电池系统的容量。

其次，在保证能为目标应用提供足够带宽的前提下，应将连接外部存储器的 MAXQ2000 I/O 数限制到最少。例如，Atmel 的 AT29LV512 闪存芯片与一个主机微控制器连接时，需要 15 条地址线、8 条数据线和 3 条控制线。由于 MAXQ2000 没有外部地址/数据总线，在上述例子中，就需要由软件来控制总线事务。对于某些应用来说，这种方法不能很有效地利用 MAXQ2000 的 I/O 引脚。

然而，基于 SPI™和 I^2C 的外部闪存器件只需要 3 到 4 个接口引脚。MAXQ2000 具有一个硬件 SPI 模块，而 I^2C 在 MAXQ2000 上必须由用户通过软件实现(即"位模拟")。这种集成功能使得 SPI 接口成为访问外部非易失性存储器的主要途径。

SD 存储卡

SD 存储介质是一种非易失性外部存储器，可满足许多应用的上述要求。SD 格式是"多媒体卡" (MMC)格式的继任产品。SD 卡存储器一般工作于 3.3V 电压，具有适度的电流要求。SD 卡的容量从几兆字节到最高 4GB 不等。容量范围如此之宽，可为众多应用提供充足的外部存储空间。

由于 SD 卡采用专有的共享总线，乍一看，似乎很难与 MAXQ2000 接口。然而，SD 继承了 MMC 的第二总线格式－SPI。由于 MAXQ2000 提供 SPI 硬件支持，连接非常容易。

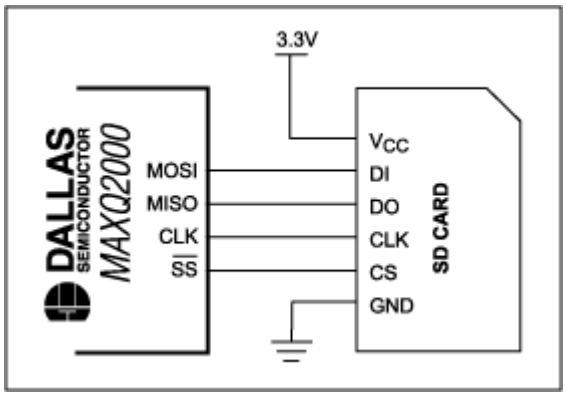

图 1. MAXQ2000 与 SD 存储卡的接口非常简单。

图 1 的电路图给出了一个典型应用电路。SD 卡要求全双工、8 位 SPI 操作。数据从 MAXQ2000 的 MOSI 引脚同步输入 SD 卡的 DI 引脚，并由 SD 卡的 DO 线同步输入 MAXQ2000 的 MISO 引脚。数据在 CLK 信号的上升沿同步输入和输出。在每次数据传输的末尾还必须提供 8 个额外的时钟，以允许 SD 完成任何未完结的操作。对应这些额外时钟的输入数据必须全为 1。识别阶段的时钟频率必须限定在 400kHz 以内，但 SD 卡一经识别后，时钟频率便可提高到 25MHz。

MAXQ2000 的 SPI 模块

MAXQ2000 包含一个硬件 SPI 模块，可以方便地针对 SD 卡接口进行配置。为了配置时钟极性和数据长度，需将 SPICF 寄存器置为全零。这种 SPI 模块配置在时钟的上升沿锁存数据，并将数据长度设置为 8 位。对于本应用，MAXQ2000 的系统时钟频率为 16MHz。在这种情况下，需要将 SPICK 寄存器置为 0x28，从而使 SPI 时钟频率接近 380kHz。必须将 SPICN 寄存器的低 2 位置位，以使能 SPI 主机模式。

SD SPI 数据格式

SD 卡的 SPI 协议与 SD 总线协议相似。如果一片 SD 卡没有数据要发送，则将 DO 引脚保持在全 1 的空闲状态，因此不是在每个时钟沿都从 SD 卡的 DO 引脚接收有效数据。当 SD 卡有数据要回送给主机时，要在数据之前先发送一些以 0 为起始位的特定令牌。当这些令牌发送完毕之后，SD 卡要发送的所有定长数据立即被发送出去。由于接收器事先已经知道要接收的字节数，因而响应中不包含表征长度的字节。此外，由于在起始令牌和数据都发送完毕后才会进入空闲状态，所以全部数据字节都以不带前缀的原始形式发送。和总线上其它所有通信过程一样，令牌大小也要和 SPI 传输的 8 位边界对齐。主机发送给 SD 卡的指令和数据都遵循类似的格式，以全 1 指示总线空闲。除了状态令牌以外，所有传输都由附加在数据末尾的循环冗余校验(CRC)码进行保护。系统提供两种 CRC 算法：CRC-7 用于小数据块，CRC-16 则用于大数据块。CRC 是 SD SPI 接口的可选部分，但除非应用系统限制它的使用，否则应该使用 CRC 来确保数据的完整性。

循环冗余校验

CRC 算法通常用于检测由不可靠的通信通道引起的误差。特定 CRC 类型的选择根据需要保护的数据长度来决定。对于基于 SD 存储介质的数据，采用 CRC-7 和 CRC-16 编码方式。

CRC 算法将被保护的数据用选中的除数进行除法运算，产生一个余数。因为该算法中用得的是多项式，所以该除法运算不含进位逻辑。无需考虑进位时，除法运算可通过逻辑 XOR 操作来实现。所选中的除数通常用 CRC 的多项式来表示。接着，计算出的余数和数据一起传输，接收器用此余数来检查确认数据在传输过程中是否正常。

对于 CRC-7，余数可通过一个 7 位移位寄存器在软件中计算。计算开始时，将该移位寄存器初始化置为全零。当受保护数据的每一位(MSB 在先)被移入移位寄存器的 LSB 时，移出移位寄存器的 MSB，并进行检查。如果移出的位为 1，则用 CRC-7 多项式系数 0x09 进行异或运算，以此来修正移位寄存器的内容。如果从移位寄存器中移出的位为 0，则无需 XOR 操作。受保护数据的最后一位被移入移位寄存器且完成了条件 XOR 操作后，必须按此类似方式移入 6 个或更多 0。这一过程称作扩张并完成多项式除法运算。此时，CRC-7 值可从移位寄存器直接读出。

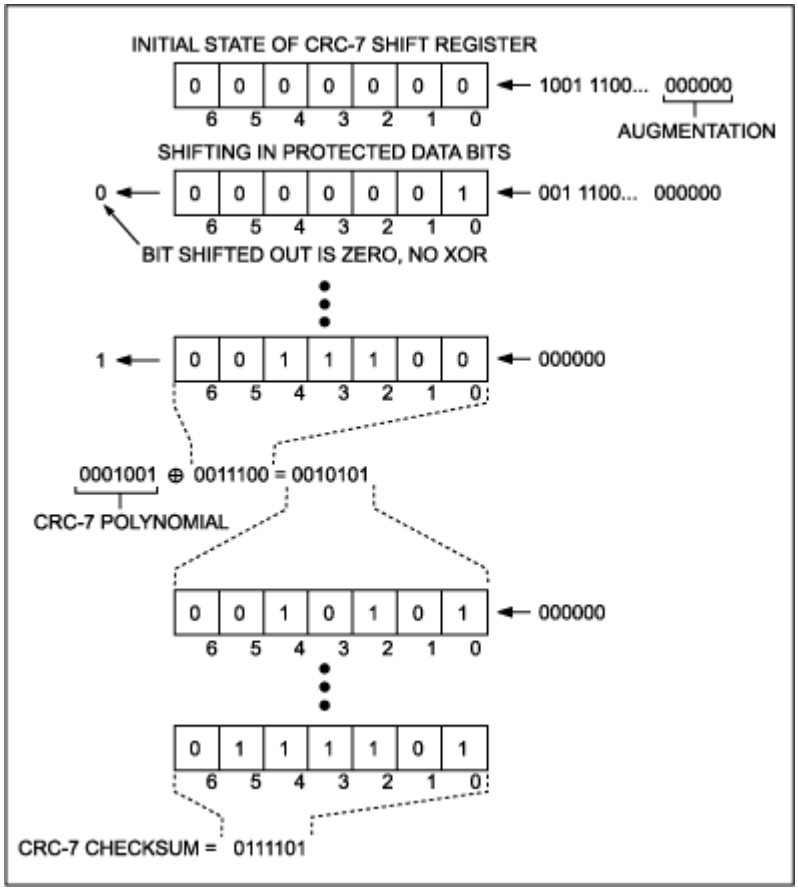

图 2. 通过移位寄存器架构计算 CRC-7。

当接收器收到所有受保护的数据后，接收器可计算基于受保护数据的 CRC-7 值并将改值与接收到的 CRC-7 值进行比较。如果这两个值不同，接收器就能判断出受保护数据在传输过程中出现错误。如果这两个值相同，则接收器可完全判定通信通道上的数据是完整的。

CRC-16 算法可用同样的方式来构建。在这种情况下，移位寄存器长度为 16 位而不是 7 位；多项式系数改为 0x1021，且输入数据通过 16 个 0 位来扩张。

SD 命令格式

发送给 SD 卡的命令采用 6 字节的格式(**图 3**)。命令的第 1 个字节可通过将 6 位命令码与 16 进制码 0x40 进行或运算得到。如果命令需要，则在接下来的 4 个字节中提供一个 32 位的参数；最后 1 个字节包含了从第 1 个字节到第 5 个字节的 CRC-7 校验和。**表 1** 列出了一些重要的 SD 命令。

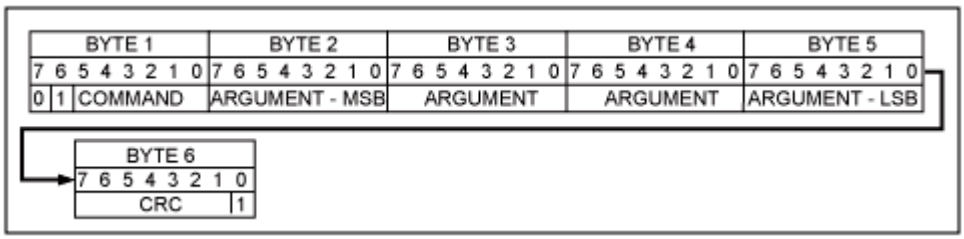

图 3. 发送给存储卡的 SPI 模式 SD 命令采用 6 字节格式。

表 **1.** 部分 **SD** 存储卡命令

Command	Mnemonic	Argument	Reply	Description
0 (0x00)	GO_IDLE_STATE	none	R1	Resets the SD card.
9 (0x09)	SEND_CSD	none	R1	Sends card-specific data.
10 (0x0a)	SEND_CID	none	R1	Sends card identification.
17 (0x11)	READ_SINGLE_BLOCK	address	R1	Reads a block at byte address.
24 (0x18)	WRITE_BLOCK	address	R1	Writes a block at byte address.
55 (0x37)	APP_CMD	none	R1	Prefix for application command.
59 (0x3b)	CRC_ON_OFF	Only Bit 0	R1	Argument sets CRC on (1) or off (0).
41 (0x29)	SEND_OP_COND	none	R1	Starts card initialization.

将 **SD** 卡初始化为 **SPI** 模式

刚上电时，SD 卡缺省使用专有的 SD 总线协议。为了将 SD 卡切换到 SPI 模式，主机应发出命令 0 (GO_IDLE_STATE)。SD 卡会检测到 SPI 模式选择信息，因为卡选择(CS)引脚在该命令和其它所有 SPI 命令传送过程中都保持为低电平。SD 卡以 R1 响应(**图 4**)作为回应。空闲状态位被置为高电平，表明 SD 卡已进入空闲状态。为了保持与 MMC 卡的兼容性，此阶段的 SPI 时钟频率一定不能超过 400kHz。

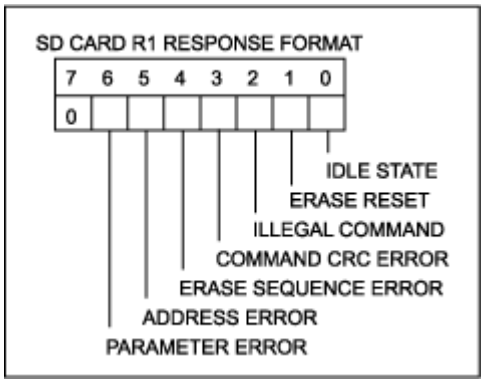

图 4. R1 响应指示发出的命令是否成功执行。

SD 卡进入 SPI 模式后，SD 规范要求主机在进行其它任何请求之前先发送一条初始化命令。为了能区分 MMC 卡和 SD 卡，SD 卡采用了一种不同的初始化命令，MMC 卡对该命令是不响应的。先向卡发送命令 55 (APP_CMD)，紧接着再发送应用命令 41 (SEND_OP_COND)，这样即完成了这个重要的步骤。MMC 卡对命令 55 不做回应，通过这一点可鉴别出 MMC 卡，并将其视作无效介质而拒绝访问。这个命令序列要一直重复执行，直到来自存储卡的 R1 响应中所有位均为 0 (也就是说，IDLE 位变为低)才停止。

```
while(status && (errors < retries)) {
  printf("-> Send CMD55_APP_CMD\r\n");
  xmitcmd(CMD55_APP_CMD, arg);
  if (waitForR1(&rxdata, 0) < 0) {
    /* If this is a MultiMediaCard (not SD), it will not respond here */
    printf("ERROR: Timeout! Perhaps this is a MMC card?\r\n");
    return TR_TIMEOUT;
  }
  check_r1(rxdata, R1_IDLE);

  printf("-> Send ACMD41_SEND_OP_COND\r\n");
  xmitcmd(ACMD41_SEND_OP_COND, arg);
  if (waitForR1(&rxdata, 0) < 0) {
    printf("ERROR: Timeout on ACMD41_SEND_OP_COND\r\n");
    return TR_TIMEOUT;
  }
  status = rxdata & R1_IDLE;
  if (status) {
    /* Pause here for a bit to let the card start up */
    for (i = 0; i < 10000; i++); /* busy loop */
  }
}
```

清单 1. 代码必须用 SEND_OP_COND 来识别 SD 和 MMC 卡。

SD 卡包含了一些重要的寄存器，用来提供 SD 卡的相关信息。最重要的寄存器是存储卡特定数据寄存器(CSD)。对于我们的应用示例而言，我们感兴趣的是存储器的数据块大小和总容量。我们还必须对存储卡标识数据寄存器(CID)加以注意，因为它包含了存储卡的制造商详细信息和序列号。**图 5** 显示了 CSD 寄存器和 CID 寄存器的配置结构。

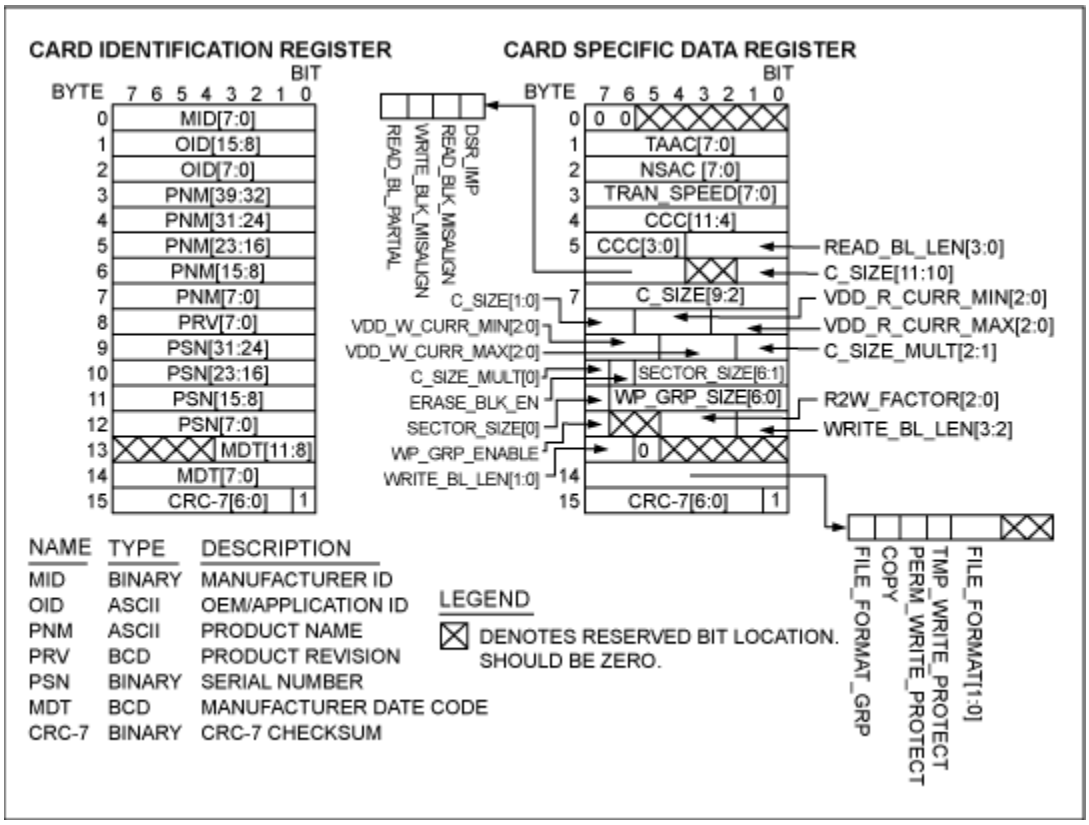

图 5. CSD 寄存器和 CID 寄存器提供 SD 卡的相关信息

检查 SD 卡的响应

要读取存储卡的寄存器或数据块，我们首先必须理解存储卡如何响应我们的请求。在 SPI 模式下，SD 卡以 R1 应答 SEND_CSD (9)、SEND_CID (10)和 READ_SINGLE_BLOCK (17)命令。接着则是一个起始令牌，然后是所请求的数据，最后是数据的 CRC-16 校验和。我们不能想当然地认为数据起始令牌是紧接着 R1 响应即刻发出的，因为总线在这两个事件之间会进入空闲状态一段时间。**图 6** 显示了数据响应的细节。

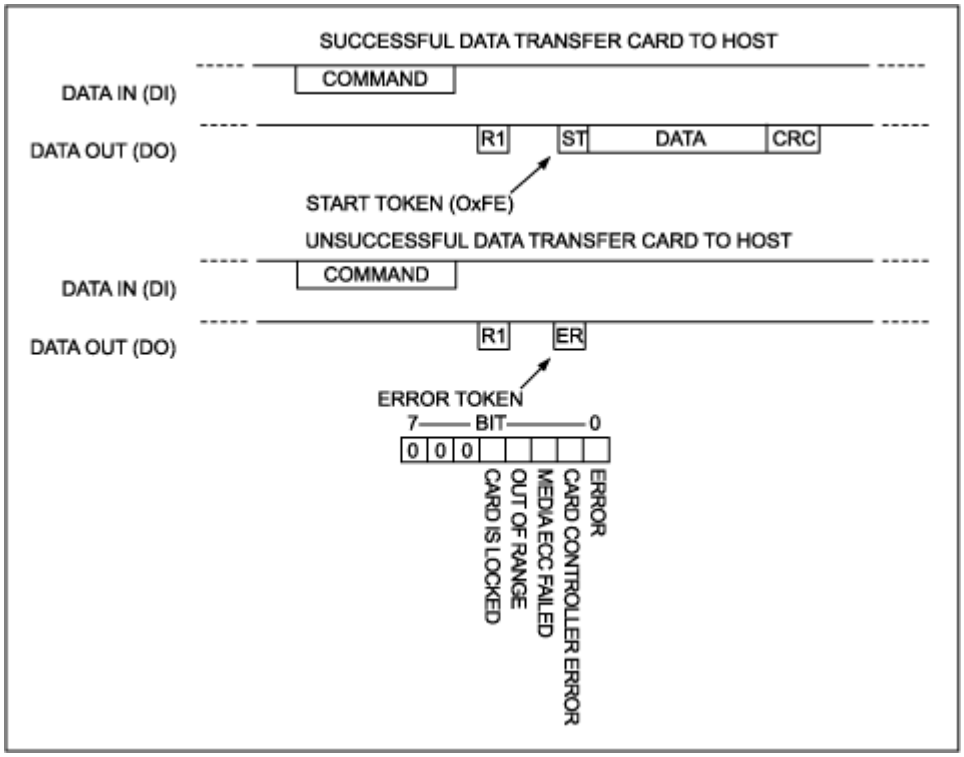

图 6. 从 SD 卡到主机的数据传输要加入一个起始令牌作为前缀。

读取 CSD 寄存器和 CID 寄存器的元数据

使用 SEND_CSD 和 SEND_CID 命令可返回寄存器的内容，以便确定 SD 卡的参数。这些命令分别返回与 CSD 或 CID 寄存器容量大小一致的固定数量的字节。SEND 命令字节中包含的参数被 SD 卡忽略。

从 SD 卡中读取一个数据块

从 SD 卡中读取一个数据块是相当简单的。主机发出 READ_SINGLE_BLOCK 命令，并将起始字节地址作为参数。这个地址必须和介质上一个块的起始位置对齐。然后 SD 卡会验证这个字节地址，并以一个 R1 命令作为响应。如果命令中的地址越界，则会在命令响应中指示这种情况。

如果完成了 SD 介质的读取操作并且没有错误发生，则先发送一个起始数据令牌，接着是固定数量的数据，最后是两个字节的 CRC-16 校验和。如果 SD 卡碰到硬件故障或介质读取错误，则不会发送起始数据令牌。而是发送一个错误令牌，数据传输随之中止。

向 SD 卡中写入一个数据块

写入一个数据块和读取数据块类似，即主机必须提供一个与 SD 卡数据块边界对齐的字节地址。写入数据块的大小必须与 READ_BL_LEN 相同，一般为 512 字节。通过发出 WRITE_BLOCK (24)命令启动写操作过程，SD 卡将以 R1 命令响应格式进行应答。如果命令响应表明写操作可以进行，则主机发送数据起始令牌，接着是固定数量的数据字节，最后以发送数据的 CRC-16 校验和结束。SD 卡返回一个数据响应令牌以指示待写入的数据是否被接受。

如果数据被接受，SD 卡会在存储卡忙时始终将 DO 线保持为低电平。存储卡忙期间，主机不必始终将卡选择线保持为低电平，如果解除 CS 选择状态，SD 卡将释放 DO 线。当多于一个的设备与 SPI 总线连接时，上述处理方式非常有用。主机可以一直等待 SD 卡释放忙指示标志，也可以定期触发片选信号以检查存储卡的工作状态。如果卡仍然处于忙状态，它会将 DO 线拉低以指示该状态。否则，存储卡会使 DO 线返回至空闲状态(见**图 7**)。

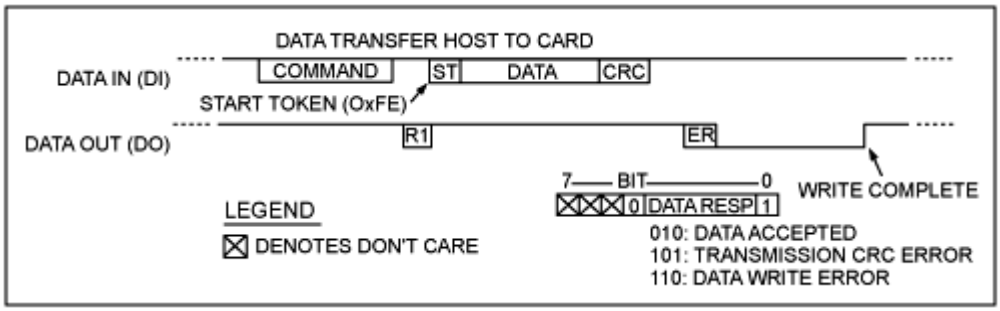

图 7. 从主机到 SD 卡的数据传输使用一套更为复杂的握手机制。

SPI 命令与数据错误检测

可利用 CRC-7 和 CRC-16 校验来检测主机与 SD 卡间的通信错误。如果因物理因素而导致错误发生，如在插入、移除时的触点抖动，或是可拆卸介质固有的接触不良状况，错误检测机制可实现坚固的错误恢复功能。我们强烈建议使用校验和机制，这可以通过发出 CRC_ON_OFF (59)命令并将参数的最低位置为高来启动该功能。

```
/* Enable CRC to protect against error */

printf("-> Send CMD59_CRC_ON_OFF\r\n");
arg[3] = 0x01; /* LSB set to 1 enables CRC verification */
xmitcmd(CMD59_CRC_ON_OFF, arg);
CLEAR_ARGS(arg);
if (waitForR1(&rxdata, 0) < 0) {
  printf("ERROR: Timeout on CMD59_CRC_ON_OFF\r\n");
  return -1;
}
if (rxdata != 0x00) {
  printf("WARNING: R1 status 0x%02x, expecting 0x00\r\n", rxdata);
}
```

清单 2. 强烈推荐使能 CRC 校验和。

结论

SD 存储卡为嵌入式系统提供了一种紧凑和低功耗的非易失性存储器方案。利用 MAXQ2000 微控制器提供的硬件 SPI 模块，能以极少的开销访问 SD 介质卡。Maxim 公司提供的参考软件(可由 www.maxim-ic.com.cn/MAXQ2000_SD

获取)演示了一个最小的实现方案，包括从 SD 卡读取数据块和向 SD 卡写入数据块所需的基本操作。

SD 卡 SPI 模式驱动程序!! --转自 Solomon
2007-09-26 23:58
本文关键词：SD SPI CMD0 CMD55 ACMD41 初始化

这些天没有出门，一直在家研究 SD 卡的 SPI 模式的初始化顺序，这里为大家总结了一下编写该程序所需要的知识：

1.SD 卡的官方资料(我承认这个资料很垃圾，比起民间的技术总结它的内容可谓又臭又长，但是作为基础也要了解一下，SD 协议不用看)
2.清晰明了的 MMC 卡时序图(虽然这个是 MMC 卡的，但是在初始化的时候 CMD0 的时序是一样的)

电路：我用的 SD 卡的电路其实很简单，参考 SD 卡的官方资料中的电路链接就可以的。
供电问题：由于 SD 卡的电压是 3.3V，所以你的 CPU 必须支持 3.3V 的 IO 端口输出。

再来说一说鸡毛蒜皮的细节：
1.为了使 SD 卡初始化进入 SPI 模式，我们需要使用的命令有 3 个：CMD0,ACMD41,CMD55（使用 ACMD 类的指令前应先发 CMD55，CMD55 起到一个切换到 ACMD 类命令的作用）。

2.为什么在使用 CMD0 以后不使用 CMD1？CMD1 是 MMC 卡使用的指令，虽然本文并不想讨论 MMC 卡的问题，但是我还是要说：为了实现兼容性，上电或者发送 CMD0 后，应该首先发送 CMD55+ACMD41 确认是否有回应，如果有回应则为 SD 卡，如果等回应超时，则可能是 MMC 卡，再发 CMD1 确认。

3.正确的回应内容应该是：
CMD0——0x01（SD 卡处于 in-idle state）
CMD55——0x01（SD 卡处于 in-idle-state）
ACMD41——0x00（SD 卡跳出 in-idle state，完成初始化准备接受下一条指令）

这里要说的是如果最后的回应内容还是 0x01 的话，可以循环发送 CMD55+ACMD41，直到回应的内容 0x00。

4.在所有的指令中，唯独 CMD0 特殊，在向 SD 卡发送以前需要向 SD 卡发送 74+个时钟。那么为什么要 74 个 CLK 呢？因为在上电初期，电压的上升过程据 SD 卡组织的计算约合 64 个 CLK 周期才能到达 SD 卡的正常工作电压他们管这个叫做 Supply ramp up time，其后的 10 个 CLK 是为了与 SD 卡同步，之后开始 CMD0 的操作，严格按照此项操作，一定没有问题。

5.关于 SD 卡的 SPI 总线，在读入数据时 SD 卡的 SPI 是 CLK 的上升沿输入锁存，输出数据也是在上升沿。

6.向 SD 卡写入一个 CMD 或者 ACMD 指令的过程是这样的：
首先使 CS 为低电平，SD 卡使能；其次在 SD 卡的 Din 写入指令；写入指令后还要附加 8 个填充时钟，是 SD 卡完成内部操作；之后在 SD 卡的 Dout 上接受回应；回应接受完毕使 CS 为低电平，再附加 8 个填充时钟。

7.在 SD 卡的 Din 没有数据写入时，应使 Din 保持高电平。

最近读者

1 网友:blueray 2008-04-02 15:39

你好！你说的步骤我了解了。我想请教一个问题：我是直接使用 SPI 接口去和 SD 卡连接的，开头的 74 个 CLK 我是通过发送 10 个 0xff 产生的 80 个 CLK 来提供的，不知道这个数字是不是要非常严格的在 74？但是我目前调试的效果太不理想了，连初始化都过不了……
另，请问 SD 卡和 MMC 卡的所有命令的实际内容在哪里可以找到啊？我在很多资料上看到都是用编号来代替的，却找不到该命令所包含的 6 个字节的实际内容……
可以的话能否将程序发到我的邮箱呀！我邮箱是：blueray_026@163.com。
谢谢了！

2  2008-04-03 13:25

不是严格的 74 个，只需要超过 74 个就行了，一般而言取 100 个左右比较好点

hao01222 那命令的最高为对应相应的命令，其数值=命令值+0X40，如命令 0 所对应的就是 0X40
而其他的数值为该命令的参数，结尾一般以 0XFF 结束，你再仔细看看 SD 的官方说明，那里面有介绍

初始化：
1、初始化读写 SD 卡的硬件条件（spi 接口和其他有用的管腿，如写保护）
2、上电延时过程
3、复位卡 CMD0
4、激活卡，内部初始化并获取存储卡的类型
CMD1,CMD55,ACMD41
5、查询 OCR，获取卡供电情况 CMD58
6、是否使用 CRC CMD59
7、设置读、写块数据长度，512B， CMD16
8、读取 CSD，获取存储卡的其他参数信息。CMD9
9、 8clock 后，禁止片选。

读取单块数据
1、主机发送 CMD17

2、接收卡响应 R1
3、接收读数据起始令牌 0xFE
4、接收数据
5、接收 2B CRC
6 、8clock 后，禁止片选。

写入单块数据
1、主机发送 CMD24
2、接收卡响应 R1
3、接收读数据起始令牌 0xFE
4、接收数据
5、接收 2B CRC
6 、8clock 后，禁止片选。

ＳＤ卡的读写和加解密

一、概述

SD 卡全称为 Secrue Digital Memory Card，具有轻巧、可加密、传输速度高、适用于手持设备使用等优点。

二、总线接口

SD 需要高速读写，同时也要使手持等嵌入式设备能方便使用，特设有两个访问接口中：SD 模式接口和 SPI 接口。SD 卡在上电初期，卡主控通过检测引脚 1 （DAT3）来决定使用 SD 模式还是 SPI 模式。当此脚接 50KOhm 上拉电阻时，卡进入 SD 模式；当此脚为低电平，卡则工作于 SPI 模式。

SD 引脚	SD 模式	SPI 模式
1	DAT3	CS
2	CMD	DI
3	VSS	VSS
4	VDD	VDD
5	CLK	SCLK
6	VSS	VSS
7	DAT0	DO
8	DAT1	Resvered
9	DAT2	Resvered

表 1：SD 卡接口定义

1 、SPI 接口

SPI 接口是为嵌入式和手持设备准备的，只使用普通的三线制 SPI 总线，即可对卡进行一般的慢速的读写等操作。

<!--[if !vml]-->图略<!--[endif]-->

图一：SPI 总线

如上图，由读卡器到卡的数据，在每个时种的上升沿把 DO 的数据锁存到卡主控，而卡的数据则在每个 CLK 的上升沿把 DI 的数据读入读卡器。

2 、SD 接口

SD 接口是为高速专有设备而设计，使设备能对卡进行高速可靠的传输而设计，因 SD 模式在每个命令及数据转输时，都必须具有正确的 CRC 校验。因此，此模式下主机一般需要专门设计的硬件模块以产生 CRC 校验。

在此模式下，SD 卡具有四根数据线，且时种速度最大可达 50MHz，所以此模式下数据传输速率比 SPI 模式快得多。

三、总线协议

SD 卡命令共分为 12 类，分别为 class0 到 class11，不同的卡主控根据其功能，支持不同的命令集。主要如下：

class0：卡的识别、初始化命令集。

class2：读卡命令集

class4：写卡命令集

class7：卡的锁定，解锁功能命令集

SD 卡只有唯一的主机，所有命令是由主机发出。总线上可传输三种类型数据，分别是命令帧、响应、数据。

命令帧：由总线上唯一的主机发出，由设备响应并执行。

响应：指设备响应主机的命令。

数据：指由主机传给设备，或由设备传给主机的数据实体。

<!--[if !vml]-->图略<!--[endif]-->

如上图，命令帧一共由 48bit 组成，其中起始位固定为 0，结束位固定为 1。每个命令最后 1 字节包含 7bit 的 CRC 校验，第一字节为的另外 7bit 为命令码，中间的 32bit 为命令参数。

一个完整的命令由命令帧和响应，或命令帧、响应和数据组合而成。

<!--[if !vml]-->图略<!--[endif]-->

图二：Mult Block Read

如上图，主机发出 command 请求，然后卡返回该命令的 respond，如果 respond 响应为正确，则 Host 通过读取 cmd 线状态，等待卡准备好数据；如果 cmd 为低电平，表示卡 busy，busy 结束后卡会把 cmd 线拉高，并且发出 start token，随即进入数据传输阶段；数据传输结束后，主机发送结束命令停止命令，传输结束。

四、卡的初始化

上电后，卡处于 IDLE 状态，主机发送 CMD0 复位 SD Card，然后通过 CMD55 和 ACMD41 判断当前电压是否在卡的工作范围内。但如果是 MMC 卡，则不能识别 CMD55，所以可以通过此命令的响应正确与否，判断插入的是 MMC 卡还是 SD 卡。

得到正确的响应后，主机可以继续通过 CMD10 读取 SD 卡的 CID 寄存器，通过 CMD16 设置数据 Block 长度，通过 CMD9 读取卡的 CSD 寄存器。从 CSD 寄存器中，主机可获知卡容量，支持的命令集等重要参数。

此时，卡已进入了传输状态，主机就可通过 CMD17/18 和 CMD24/25 对卡进行读写。

五、加密

SD 卡具有安全加密功能，内置 128bit 加密位，在加密状态下，用户需提供密码才可以访问卡内的数据。

在卡上电时，若卡包含密码，卡自动进入锁定状态，读写命令均返回错误，以保护卡内容不被读出及修改。

密码设置功能由 CMD42 实现，其数据包中包括该命令中所有的信息。

<table>
<tr><td>Byte</td><td>Bit7</td><td>Bit6</td><td>Bit5</td><td>Bit4</td><td>Bit3</td><td>Bit2</td><td>Bit1</td><td>Bit0</td></tr>
<tr><td>0</td><td colspan="4"></td><td>Erase Card</td><td>Lock/UnLock</td><td>ClrPwd</td><td>SetPwd</td></tr>
<tr><td>1</td><td colspan="8">Psw_Length</td></tr>
<tr><td>2</td><td colspan="8" rowspan="5">Pwd Data</td></tr>
<tr><td></td></tr>
<tr><td></td></tr>
<tr><td></td></tr>
<tr><td>PwdLen+1</td></tr>
</table>

擦除：此位置 1 时，卡的密码和内容会被强制擦除，在遗忘密码时可使用此功能。

锁定/解锁：此位置 1 时，表示命令结束后状态为锁定，为 0，表示卡解锁。

清除密码：此位置 1，表示清除卡的旧密码，此时数据中必须包含旧密码的正确内容。

加密：此位置 1，表示设置卡的新密码，数据中必须包含新密码内容；更改密码时，新密码紧跟随旧密码内容。

注：在 CMD42 命令之前，首先要使卡工作在传输状态，在 SD 模式下可使用 CMD7 进行状态转换，在 SPI 模式下，可使用初始化序列进行状态切换。

在任意刻，主机可以通过 CMD13 命令读取卡的内部状态，判断其锁定状态。定义如下：

<!--[if !vml]-->图略<!--[endif]-->

bit0 置 1 表示卡处于锁定状态。

1、设置密码

使用 CMD9 设置 Block 长度为密码长度为 PWD_LEN+2；

发送 CMD42 命令：0x6A,0x00,0x00,0x00,0x00,0x95；

发送命令数据：0x01,LEN,CMD_DATA0,CMD_DATA1……,0xFF,0xFF；

使用 CMD9 恢复原 Block 长度。

2、清除密码

使用 CMD9 设置 Block 长度为 PWD_LEN+2；

发送 CMD42 命令：0x6A,0x00,0x00,0x00,0x00,0x95；

发送命令数据：0x02,LEN,CMD_DATA0,CMD_DATA1……,0xFF,0xFF；

使用 CMD9 恢复原 Block 长度。

3、卡的锁定、解锁

使用 CMD9 设置 Block 长度为 PWD_LEN+2；

发送 CMD42 命令：0x6A,0x00,0x00,0x00,0x00,0x95；

发送命令数据：0x04,LEN,密码[LEN]……,0xFF,0xFF；

使用 CMD9 恢复原 Block 长度。

4、修改密码

使用 CMD9 设置 Block 长度为 OLD_PWD_LEN+NEW_PWD_LEN+2；

发送 CMD42 命令：0x6A,0x00,0x00,0x00,0x00,0x95；

发送命令数据：0x05,LEN,旧密码[n],新密码[m]……,0xFF,0xFF；

使用 CMD9 恢复原 Block 长度。

5、卡擦除

使用 CMD9 设置 Block 长度为 1；

发送 CMD42 命令：0x6A,0x00,0x00,0x00,0x00,0x95；

发送命令数据：0x08,0xFF,0xFF；

使用 CMD9 恢复原 Block 长度。