单片机实现对 CF 卡的读写

摘要：CF 卡是一种包含了控制和大容量 Flash 存储器的标准器件，具有容量大、体积小、高性能、携带方便等优点，已广泛应用在数据采集系统和许多消息类电子产品中。本文详细介绍 CF 卡在单片机系统中的硬件接口电路，以及单片机对 CF 卡进行标准文件读写的实现，且写入的文件能被 Windows 操作系统读写。

关键词：CF 卡 单片机 FAT 文件格式

引言

由于 CF 卡（Compact Flash Card）具有容量大、体积小、高性能、携带方便等优点，而且读写速度快，可与多种电脑操作系统平台兼容，因此在数据采集系统中的数据记录或与 PC 机之间的数据转存多采用 CF 卡。为了在 PC 机中能方便地进行数据处理，在下位机端必须采用一种标准的格式组织数据，即将数据按照 Windows 标准文件格式写入，在 PC 机端通过读卡器将写入 CF 的内容以标准文件形式读出。Windows 标准文件格式有 FAT、FAT32 和 NTFS。考虑到广泛使用的 Windows 98 系统的 CF 卡的容量等因素，通常采用 FAT（File Allocation Table）文件系统。单片机系统对 CF 卡的读写，就是从底层对它进行直接操作，包括寻址、创建文件和读写等。

1 CF 卡简介

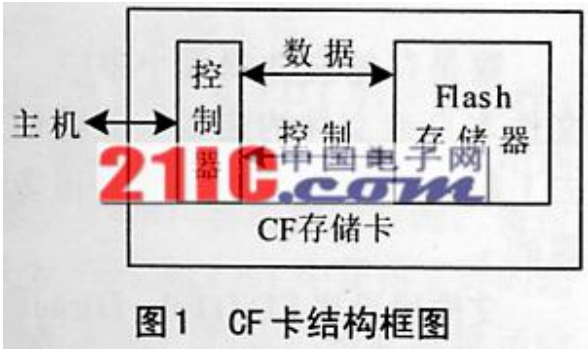

图1 CF卡结构框图

CF 卡内集成了控制器、Flash Memory 阵列和读写缓冲区，如图 1 所示。内置的智能控制器，使外围电路设计大大简化，而且完全符合 PC 机内存卡的国际联合会 PCMCIA（Personal Computer Memory Card International Association）和 ATA（Advanced Technology Attachment）接口规范。实际上，控制器起到了一种协议转换的作用，即将对 Flash Memory 的读写转化成了对控制器的访问，这样不同的 CF 卡都可以用单一的机构来读写，而不用担心兼容性问题。CF 卡的缓冲区结构，使得外部设备与 CF 卡通信的同时，CF 卡的片内控制器可以对 Flash 进行读写。这种设计可以增加 CF 卡数据读写的可靠性，同时提高数据传输速率。

CF 卡支持多种接口访问模式，有符合 PCMCIA 规范的 Memory Mapped 模式、I/O Card 模式和符合 ATA 规范的 True IDE 模式。上电时，OE（9 脚）为低电平，CF 卡进入 True IDE 模式，此时引脚 OE 也叫 ATA SEL；上电时，OE（9 脚）为高电平，CF 卡进入 PCMCIA 模式，即 Memory Mapped 模式或 I/O Card 模式，此时可通过修改配置选项寄存器进入相应的模式。

配置选项寄存器格式如下：

SRESET	LevelREQ	conf5	conf4	conf3	conf2	conf1	conf0

SRESET LevelREQ conf5 conf4 conf3 conf2 conf1 conf0

SRESET—软复位信号；

Level REQ—中断模式选择（电平或边沿触发）。

例如，要加入 Memory mapped 模式，只需要在上电时保证 OE 为高电平，因为配置选项寄存器的 conf5～conf0 位的初始化值为“00000”；而要进入 I/O Card 模式，除了上电时保证 OE 为高电平外，还要进一步设置 conf5～conf0，如表 1 所列。但是对于具体型号的 CF 卡而言，下面三种情况也是被 CFA（CF card Association）所允许的：①上电时进入 True IDE 模式，工作过程中，只要监测到 OE 变为高，就退出 True IDE 模式；②允许卡在复位时重新配置；③上电时进入 PCMCIA 模式，允许过程中，只要监测到 OE 变为低，就进入 True IDE 模式。

表 1 模式选择

conf5	conf4	conf3	conf2	conf1	conf0	模　式
0	0	0	0	0	0	Memory map
0	0	0	0	0	1	I/O Mapped,对应16位系统
0	0	0	0	1	0	I/O，对应1F0h-1F7h/3F6h-3F3h
0	0	0	0	1	1	I/O,对应170h-177h/376h-377h

2 CF 卡与 51 单片机的接口

CF 卡在 PC Memory 方式与 51 芯片的接口电路如图 2 所示。由于采用 CF 卡上电后自动进入的 Memory 模式，而且不存在对特性寄存器的读写，故可将 REG 接高电平。片选信号 CE1 和 CE2 组合可选择数据位宽度，如表 2 所列。图 2 中 CE2 接 VCC，选用的是 8 位（D7～D0）数据宽度。

表 2 数据宽度选择

	8位（D7～D0）	8位（D15～D8）	16位	高　阻
CE1	0	1	0	1
CE2	1	0	0	1

为了实现即插即用的功能，CE 卡上提供了两个用来检测卡是否存在的引脚（CD1、CD2），由卡内部接地。当主机检测到与其相连的 CD1 和 CD2 两个引脚同时为低电平时，可判断出卡与主机相连；否则，卡未与主机相连。

由于 I/O 口紧张，RDY/BSY 引脚悬空不用，通过查询状态寄存器能判断 CF 卡是否准备就绪。在实际应用中，由于一次至少要读写一个扇区 512 字节，所以要扩充一块 RAM。我们选用的是 62256，容量为 32KB，这样便可以支持大到 2GB 的 CF 卡（参见下文），增加了其扩展性。

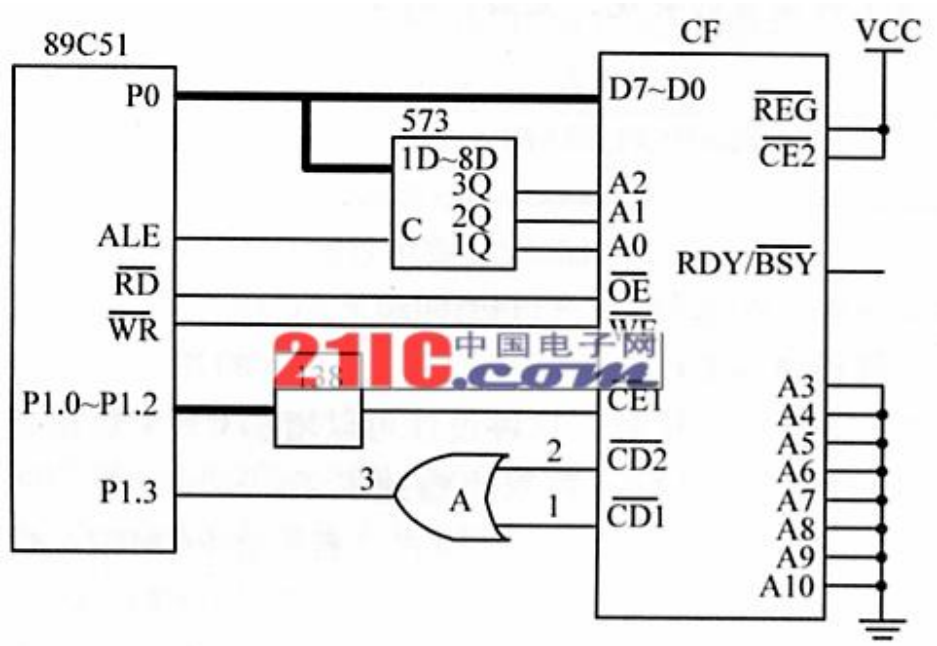

图 2　CF 卡与 51 单片机的接口电路

3 FAT 文件系统

FAT 文件系统是基于 DOS 的文件系统。常说的 FAT 有 12 位的 FAT12 和 16 位的 FAT16，另外就是 32 位的 FAT32。考虑到 CF 卡的容量有限，宜选用 FAT16。这里只对 FAT 文件系统作一简单介绍，更详细的内容请见参考文献。

磁盘的寻址方式有两种：物理寻址 C/H/S（柱面/磁头/扇区）方式和逻辑块 LBA（Logical Block Addressing）寻址方式。二者之间的转换关系为：

LBA 地址=（柱面号×磁头数+磁头号）×扇区数+扇区数-1

采用 LBA 寻址方式，没有磁头和磁道的转换操作，在访问连续的扇区时，操作速度比物理寻址方式要快，而且也简化了对磁盘的访问。

硬盘的结构布局分为 MBR（主引导扇区）和最多 4 个逻辑分区（含 DOS 分区或非 DOS 分区），而在 DOS 逻辑分区中的磁盘组织如下：

引导扇区	FAT1	FAT2	根目标区	数据区

引导扇区 DBR（DOS Boot Record）:位于 LBA 0 扇区，包含跳转指令、厂商标识和 DOS 版本号、BPB（BIOS Parameter Block,BIOS 参数块）、DOS 引导程序、结束标志字 AA55。其中 BPB 包含每扇区字节数、每簇扇区数、每个 FAT 扇区数、扇区总线、根目录项数等等参数。

FAT 是给每个文件分配磁盘物理空间的表格。FAT16 簇数的上限是 2 16，即 65536 个，每簇扇区数的上限是 64 个，因此其分区空间的上限为 2G。FAT1 位于逻辑 1 扇区。FAT 簇映射中，0000 表示空簇，FFF0～FFF6 备用，FFF8～FFFF 表示簇链结束，FFF7 表示坏簇，其余值表示其后续簇的簇号。图 3 所示的文件起始簇号为 2，结束簇号为 4，共占用 2、3、4 三个簇。

簇是存储文件的最小单位，可以包含多个扇区。当文件本身或文件的最后一簇哪怕只有 1 个字节，也要占去 1 簇。这样，当这种文件很多时，空间的浪费是很可观的。

文件目录表 FDT（File Directory Table）是操作系统寻找文件的入口，其内容是每一个文件的目录。FDT 中的每一个目录项由 32 个字节组成。前 8 个字节是文件名，不足时用空格填满。紧跟着的 3 个字节是文件扩展名，接下来是 10 个字节的系统保留字。然后是文件产生的时刻和日期占 8 个字节，再后的 2 个字节是文件首簇号，最后 4 个字节是文件大小。FDT 的起始扇区可由 FAT 的大小计算出，而 FAT 的大小可在 DBR 中读出。

图 3 FAT 表

4 软件实现

按照 FAT16 方式存储文件，是一个通用的解决方案。因为这样可以得到现有的 DOS 和 Windows 系统的支持，但是代价是浪费一部分空间，也就是说存储效率下降了。为了改善这一情况，采用了改进的存储方法。就是先创建一个空文件，并根据需要为其分配一个大的存储空间，写入动作只是从尾部追加数据。这样就避免了很多小文件的产生，既可以充分利用存储空间，又可以使地址连续。

CF 卡的读写是通过卡内的缓冲区进行的，不支持直接读写存储区域。缓冲区为一个 FIFO 结构，读写顺序进行，不支持随机存取，系统只能一次性地按顺序读完或写完所有一个或多个扇区。

设计时使用 LBA 方式访问 CF 卡比较方便，读写时只需要先在相应的寄存器写入 LBA 地址即可。要设定 LBA 方式，需访问驱动器/磁头寄存器。内存模式下部分寄存器译码如表 3 所列。

表 3 内存模式下部分寄存器译码

REG	A10	A9～A4	A3～A0	offset	OE=0	WE=0
1	0	X	0000	0	偶字节读	侧字节写
1	0	X	0001	1	错误寄存器	特性寄存器
1	0	X	0010	2	扇区数	扇区数
1	0	X	0011	3	扇区号（LBA7～0）	扇区号（LBA7～0）
1	0	X	0100	4	低柱面号（LBA15～8）	低柱面号（LBA15～8）
1	0	X	0101	5	高柱面号（LBA23～16	高柱面号（LBA23～16）
1	0	X	0110	6	驱动器/磁头（LBA27～24）	驱动器/磁头（LBA27～24）
1	0	X	0111	7	状态寄存器	命令寄存器

驱动器/磁头寄存器结构如下：

1	LBA	1	DRV	HS3	HS2	HS1	HS0

LBA—1 为 LBA 方式，0 为 C/H/S（柱面/磁头/扇区）方式；DRV—选择驱动器 0 或驱动器 1；HS3～HS0—LBA27～24，或为 C/H/S 方式的磁头号。

文件创建过程也就是针对 FAT 和 FDT 的读写过程。首先在 FDT 中申请表项，创建文件名称、属性、起始簇号、文件大小等，然后修改 FAT，分配数据空间，备份 FAT。文件存储就是要先从 FDT 和 FAT 中获得文件的起始簇号和簇号链，即 LBA 地址。然后，将此地址送给寄存器 3、4、5、6（表 3 中的 offset3、4、5、6），向扇区数寄存器填写读写数据所占的扇区个数，再向 CF 卡的命令寄存器写入操作的命令字，写操作 30H，读操作 20H。当写入命令或写入数据后要查询状态寄存器的状态，以判定 CF 卡是否准备就绪或写入成功。状态寄存器结构如下：

BUSY	RDY	DWF	DSC	DRQ	CORR	0	ERR

各位的值为 1 时含义如下：

BUSY—CF 卡记，此时不能接受其它命令；

RDY—卡可以接受命令；

DWF—写错误；

DSC—卡准备就绪；

DRQ—CF 卡请求数据传送；

CORR—数据错误但被修正，不会终止多扇区读操作；

ERR—在上一命令以某种错误结束，可以在错误寄存器中查看错误类型。

下面以向 CF 卡写一个扇区数据为例，给出图 4 所示流程和 C 程序代码。

```
bit flag_1,flag_2;
void cfwr()
{
unsigned char status;
cfwr_comm(0xe0,0x00,0x00,0x6c);
//写参数命令，指向逻辑 6c 扇区
do{status=PBYTE[0x07]; //读状态寄存器
if((status & 0x01)==0x01)
flag_1=1; //若 ERR=1，置出错标志，做相应处理
while(status!=0x58);
cfwr_dat(); //写入数据
do{status=PBYTE[0x07]; //读状态寄存器
if((status & 0x20)==0x20)
flag_2=1; //若 DWF=1 时，置出错标志，做相应处理
while(status!=0x50);
}
```

```
void cfwr_comm(unsigned char lba27,lba23,la15,lba7) //写参数命令函数
{PBYTE[0x02] 扇区数为 1
PBYTE[0x03]=lba7;
PBYTE[0x04]=la15;
PBYTE[0x05]=lba23;
PBYTE[0x06]=lba27; //设定 LBA 方式
PBYTE[0x07]=0x30; //送写入命令 30H
}
void cfwr_dat() //写数据函数
{unsigned int i,temp;
unsigned char xdata dat[512]; //dat[]存放一个扇区的数据
for (i=0;i<512;i++) //连续写 512 字节
{P1=P1 & 0xf8; //选中外部 RAM
temp=dat[i];
P1++; //根据实际电路选择中 CF 卡
PBYTE[0x00]=temp;}
}
```

图4　写扇区流程

5 结论

笔者在湿度检测仪中，根据本文所介绍的方法，用 CF 卡向计算机转存数据，可以非常方便地对数据进行维护。