软盘容量的扩充方法

马学文　（武汉大学）　戴耀东　（信阳师范学院）

摘要：本文从较低层次上对 5.25 英寸软盘的结构进行分析，说明增加磁道个数与每道扇区数的可行性和可靠性，并给出这种扩充的具体方法；从而使磁盘容量从 360KB 增加到 480KB。

市售 5.25 英寸倍密软盘上常常标有 500KB 字样，说明这样的软盘容纳 500KB 的信息是不成问题的。然而，实际中用户只能使用 360KB。这不禁令人产生一种疑问，剩余的 140KB 跑到哪里去了呢？能否将其发掘出来为我所用？笔者对磁盘低层结构进行了分析，发现这不是不可能的。而且实践表明，无需更换硬件设备，也不用修改系统软件，就能可靠地使用其中的 480KB。

一、磁盘结构

磁盘容量与磁盘结构密切相关。只有充分了解各级结构布局，才能发现和开发其中的荒芜之地。

1.磁盘的物理结构

众所周知，磁盘由磁道构成，磁道由扇区构成。但这仅仅是粗略的描述。实际上，一条磁道除了通常所说的扇区外，还包括前置区、扇区间隙和后置区。

前置区位于一条磁道的开端。在改进调频制 MFM 记录方式中，前置区由 32 字节的 16 进制 FF 信号组成。主要用于磁盘缓冲，以防止因索引孔位置、传感器位置误差而造成的软盘互换性的降低。

扇区间隙位于两扇区的交界处，将相邻扇区隔开。其作用也是用于缓冲，防止软盘转速误差对互换性的影响。其长度是可变的，在 MFM 制式，每道 9 扇区时，标准值是 84 字节。

磁道最后一部分是后置区，用于连接当前最后一个扇区与下邻磁道索引脉冲前沿的间隙，防止主轴转速变化对总区段占用磁道长度的影响而造成的信息丢失。它的长度也是可变的，主要由格式化时磁盘机械误差、瞬时电信号波动等随机因素引起。甚至同一磁道多次格式化，其长度也不尽相同，平均值是 296 字节。

2.扇区的结构

不熟悉磁盘结构的读者也许会认为，扇区就是磁道上一个长度为 512 字节的区段，全部用于存放用户数据。其实不然，正常情况下，扇区的长度是 574 字节，它又被分成若干小段，上面所说的 512 字节仅是其中的一段（称为数据区）。此外还包括同步校验字符序列、ID 地址标记、数据标记、扇区地址（ID）、循环冗余校验和地址区与数据区之间的空隙等。详见表 1。

表 1　扇区结构

段名	同步校验	地址标识		地址(ID)				冗余校验	间隙	同步校验	数据标识		数据区	冗余校验
				磁道号	磁头号	扇区号	记录长度							
长度	12	3	1	1	1	1	1	2	22	12	3	1	512	2
值	00H	AIH	FEH	0–48	0–1	1–9	2	?	4EH	00H	AIH	FeH	用户数据	?

3.磁盘的逻辑结构

磁盘上的扇区是用来存放信息的，从这个意义上讲，磁盘上所有扇区的地位是相同的。但磁盘上的信息是有组织有顺序的，因而不同扇区的作用也是有区别的。为便于磁盘上信息的存取与管理，磁盘从逻辑上划分成几个区域。表 2 是各区域的含义及地址。

表 2　磁盘逻辑结构

含义	绝对地址	逻辑扇区号
DOS 引导记录	0 磁道、0 磁头、1 扇区	0
文件分配表 FAT	0 道、1 头、1 扇区～0 道、0 头、2 扇区	1～2
FAT 副本	0 道、1 头、2 扇区～0 道、0 头、3 扇区	3～4
文件目录表 FDTO	0 道、1 头、3 扇区～0 道、0 头、6 扇区	5～11
用户数据区	0 道、0 头、7 扇区起余下部分	12～

二、磁盘容量扩充的可行性与可靠性分析

由磁盘结构可以看出，欲实现磁盘容量的扩充不外乎有两种方法：一是增加磁道个数，二是增加每道扇区数。

5.25 英寸双面双密软盘的标准格式是每面 40 个磁道，每道 9 个扇区。那么是否每面只能容纳 40 个磁道呢？磁盘规格说明中，磁道密度指标为 48TPI，即每英寸 48 个磁道。5.25 英寸磁盘的读写窗口宽度正好为 1 英寸，因而可以容纳 48 个磁道。只用 40 个磁道主要是考虑到磁盘机和磁盘的互换性。0～39 道较好，内圈的 8 个磁互换性差，因而被舍弃了。

不使用内圈的磁道并不是说内圈的磁道不能用，一般的磁盘机都可正常读写 46 个磁道，即 0～45 道。有些磁盘机，特别是日本原装磁盘机甚至可以完全读写。因此，可根据现有硬件情况适量增加磁道个数。

磁道格式表明，一个磁道的容量大约在 6224 字节左右，随磁盘机转速不同稍有偏差。磁道上除包含控制信息、寻道标记、用户数据区等必不可少的字段外，还包括许多间隙。在这些间隙中扇区间隙和后置区的长度是可变的。扇区间隙是由于驱动器主轴转速误差而引入的。有了它，即使转速发生微小变化，也不会出现在写入一个扇区的数据时覆盖别的一个扇区内容的情况。扇区间隙预留 84 字节，是以所允许主轴电机最大转速误差而设置的。一般来说，其长度不少于 5 个字节，就不会发生读写故障。

现在设想，如果将扇区间隙减少到 10 个字节，能否在一个磁道内容纳 10 个 512 字节扇区呢？现在来计算一下，一个标准扇区共有 12 +4 +4 +2 +22 +12 +4+ 512 +2= 574 字节，加上前置区与扇区间隙共有 32+10x (574+10) = 5872 个字节，小于磁道容量 6224。因而是没有问题的。

无论是增加磁道还是扩充扇区数，都没有改变磁盘的存储密度，故并不会降低磁盘的可靠性。当然，也带来一些麻烦。首先正常的格式化程序无法格式化出这样特殊的磁盘；其次即使有这样的盘，系统也无法正常访问。因而，必须编制新的格式化程序，并对磁盘参数作相应修改。

三、特殊格式化程序的设计

1.物理格式化

ROM BIOS 的 13H 片中断提供了磁道格式化功能。其入口参数是：

AH＝5(功能号)；

DL＝驱动器号(A:0，B:1，C:80H，D:81H)；

DH＝磁头号；

CH＝磁道号；

ES:BX＝ID 地址参数表首址。

需要加以说明的是 ID 地址参数表。该表就是前面提到的磁道各个扇区 ID 段要填写的内容，即磁道号、磁头号、扇区号、记录长度(0 代表 128、1 代表 256、2 代表 512、3 代表 1024)。欲使每道 10 个扇区，这张表就应该有 10 个这样的项。具体数值参考程序中的 IDMARK 变量。

细心的读者也许会发现，磁道格式化功能的入口参数并没有提到扇区间隙和每道扇区数。而这恰恰又是重要的，因为我们的目的之一就是要增加每道扇区数，而扇区数的增加又是以缩短扇区间隙为前提的。为了使格式化功能理解我们的意图，就必须把这些信息告诉给它。其实，13H 中断的操作是严格按照磁盘基数表进行的。这张表位于磁盘引导扇区偏移字节 21H～2BH 处，在系统起动时被读到内存绝对地址 0000:0522H 处。全表共 11 个子项，其中偏移 4 和 5 字节分别是每道扇区数和扇区间隙长度。只要在调用 INT13 之前，把它们改成相应的值就能达到目的。

2.逻辑格式化

一张软盘仅有物理格式化，DOS 仍无法使用，还必须进行适当的逻辑划分。这个工作包括传送引导记录、建立 FAT 和 FDT。由于我们的磁盘是特殊格式的，不能原样拷贝普通盘上的引导扇区，拷贝之前，需要修改该区的两个磁盘参数表。基数表的位置已在前面说明。磁盘 I／O 参数表位于记录扇区偏移字节①BH～ICH，也包含 11 个子项。其中相对于该表头第 8 和第 13 字节分别是总扇区和道扇区数，必须修改这两处的值。

FAT 的前三字节是介值述，这里可仍取 FD，FF，FF(十六进制)。其余项可均填为零。FOT 是目录表对于新盘来说它的所有项都为零。FAT 和 FAT 位置与长

度请参考表 2。

3.程序

见附录。

四、特殊格式化盘的使用

本文提供的程序已在 IBM PC／XT 机上通过。使用时只需按照提示输入最大磁道号，就可很方便地得到一张容量扩充了的软盘。如果提供的最大磁道号是 47，则该软盘的容量可达 492520 字节(即 480KB)。

但到目前为止，这张盘还不能方便地使用。欲使 DOS 承认这张盘，就必须把新盘上的基数表装入内存。由于这个过程通常是在系统启动时完成，因此，有必要制备一张以特殊格式化盘为介质的系统启动盘。具体方法如下：

1.进入 DEBUG

2.在 DEBUG 中修改内存参数表

```
-eoooo:0522↓    ;基数表首址
0000:0522  DF. 02. 25. 02. 09.0a 2A.0a↓
-q↓    ;退出 DEBUG。
```

3.因软盘基数表与硬盘基数表在内存中的位置不同，修改后，并不影响硬盘工作。现在假设硬盘上包含两个隐含文件(IBMBIO、10M、IBM DOS、COM)，再打入如下命令：

```
A>C↓;改省缺驱动器为 C
C>SYSA:↓;拷贝隐含文件到 A
C>COPY COMMAND、AOM A:R;拷贝命令处理文件于 A
C>
```

类似地也可把自己需要的应用程序拷到新盘，这样一张系统启动盘就作好了。以后只要用该盘启动，由新格式化程序制作的软盘就能象普通盘一样方便地使用了。

```
A>TYPE B:NFORM.ASM
;*******软盘特殊格式化程序*******
;————————————————————————————
stack  segment  para  stack'stack'
       db  128  dup (?)
stack  ends
;————————————————————————————
DATA   SEGMENT PARA PUBLIC 'DATA'
IDMARK DB  0,0, 1,2, 0, 0, 2, 2,0,0, 3,2,0, 0,4,2, 0,0,
       5,2
       DB  0, 0,6, 2,0, 0,7, 2,0, 0,8, 2,0, 0,9, 2,0,
       0,10, 2
STRAC  DB  0H          ;当前磁道号
ETRAC  DB  45          ;结束磁道号
GRP    DB  0           ;扇区间隙字节数
SNUMB  DB  10D         ;扇区数／道
BUFF   DB  3,0         ;键盘输入缓充区
       DB  3 DUP (0)
HEAD   DB  0           ;磁头号
BOOT   DB  521  DUP(?)
MESS1  DB  0AH,0DH,  5.25英寸软盘特殊格式化
       程序、版本 1.00'
       DB  0AH,0DH,  武汉大学数学系研究生，马
       学文.1993',0ah,0dh
       DB  0AH,0DH,  请输入下列数据，',',s'
MESS3  DB  0AH,0DH,'最大磁道号(0—47，省缺值
       45),',',s'
MESS5  DB  0AH,0DH,0AH,0DH,  请把新盘插入
       区动器 A:'
       DB  0AH,0BH,  准备就序按任意键开始
       ,0AH,0DH,'s'
MESS7  DB  0AH, 0DH,  正在进行格式化……s'
MESS8  DB  0AH,0DH,  格式化完毕
       0AH,0DH,'S'
MESS9  DB  0AH,0DH,  格式化失败，0AH ,0DH,
       07H, 07H, 'S'
MESS6  DB  0AH,0DH,  请把系统盘插入驱动器B:'
       DB  0AH,0DH,  按任意键BOOT扇区传送
       开始,0AH,0DH,'S'
MESS10 DB  0AH,0DH,  BOOT扇区不能读出
       ,0AH,0DH,07H,07H,'s'
MESS11 DB  0AH,0DH,  BOOT扇区传送失败
       ,0AH,0DH,07H,07H,'S'
DATA   ENDS
;————————————————————————————
CODE   SEGMENT  PARA  PUBLIC 'CODE'
       ASSUME  CS; CODE, DS: DATA, ES:DATA,
SS: STACK
```

```
        《主程序》
MAIN    PROC  FAR
        PUSH  DS
        XOR   AX,AX
        PUSH  AX
        MOV   AX,DATA
        MOV   DS,AX
        MOV   ES,AX
        MOV   DX,OFFSET MESS1
        MOV   AH,09H
        INT   21H
DISP2:  LEA   DX,MESS3
        MOV   AH,09H
        INT   21H
        LEA   DX,NUFF
        MOV   AH,0AH
        INT   21H
        MOV   AL,BUFF+1
        CMP   AL,1
        IZ DISP4
        CHLL  SOBPROC
        CMP   AL,47
        JG    DISPZ
DISP4:  MOV   ETRAC,AL
        LEA   DX,MESS5
        MOV   AH,09H
        INT   21H
        MOV   AH,0CH
        MOV   AL,08H
        INT   21H
        LEA   DX,MESS7
        MOV   AH,09H
        INT   21H
        CALL  SUBMOD
        MOV   HEAD,0
BECFOR:LEA BX,IDMARK
        MOV   CL,SNUMB
        MOV   CH,0H
        MOV   AL,STRAC
        MOV   AH,HEAD
MODID:  MOV   BYIE  PIR [BX],AL
        MOV   BYTE  PTR [BX+1],AH
        ADD   BX,4
        LOOP  MODID
        MOV   CX,5
NEXT:   PVSH  CX
        MOV   AH,05H
        MOV   AL,SNUMB
        MOV   CH,STRAC
        MOV   DH,HEAD
        MOV   DL,00H
        MOV   BX,OFFSET IDMARK
        INT   13H
        POP   CX
        JNZ   NEXT
        MOV   DX,OFFSET MESS9
        MOV   AH,09H
        INT   21H
        CLAA  SUBREC
        RET
CHAN:   MOV   AL,HEAD
        ADD   STRAC,AL
        INC   HEAD
        AND   HEAD,00000001B
        MOV   AL,STRAC
        CMP   AL,STRAC
        JLE   BEGFOR
        LEA   DX,MESS8
        MOV   AH,09H
        INT   21H
        CALL  SUBREC
        CALL  SUBRORT
        RET
MAIN    ENDP
;       《字符转数字子程序》
SUBPROC PROC NEAR
        MOV   AL ,BUYFF+2
        SUB   AL,30H
        CMP   AL,0
        JL    ERROR
        CMP   AL,9
        JG    ERROR
        MOV   BL,BUFF+3
        CMP   BL,0DH
        JE    RETUR
        SUB   BL,30H
        CMP   BL,0
        JL    ERROR
        CMP   BL,9
        JG    ERROR
        MOV   CL,10D
        MUL   CL
        ADD   AL,BL
        JMP   RETUR
ERROR:  MOV   AL,125D
RETUR:  RET
SUBPROC ENDP
;       《修改内存基数表子程序》
SHBMOD  PRLC NEAR
        MOV   AL,SNUMB
        PUSM  DS
        XOR   BX,BX
        MOV   DS,BX
        MOV   BX,0522H
        MOV   BYTE  PTR[BX+4],AL
        MOV   AL,BYIE  PTR[BX+5]
        MOV   RYTE  PTR[BX+5],10
        POP   DS
        MOV   GRP,AL
        RET
SUBMOD  ENDP
;       《恢复内存基数表子程序》
SUBREC  PROC  NEAR
```

```
        MOV  AL,GRP
        PUSH  DS
        XOR  BX,BX
        MOV  DS,BX
        MOV  BX,0522H
        MOV  BYTE  PTR[BX+4],9
        MOV  BYTE  PTR[BX+5],AL
        POP  DS
        RET
SUBREC ENDP
;     《建立 BOOT / FAT / FDT 子程序》
subrout proc  near
        MOV  DX,OFFSET  MESS6
        MOV  AH,09H
        INT  21H
        MOV  AX,0C08H
        INT  21H
        MOV  CX,05H
READBOOT:PUSH CX
        MOV  BX,OFFSET  BOOT
        MOV  AX,0201H
        MOV  CX,0001H
        MOV  CX,0001H
        INT  13H
        POP  CX
        JNC  MODBOOT1
        DEC  CX
        JNZ  READNOOT
        MOV  DX,OFFSET MESS10
        MOV  AH,09H
        INT  21H
        RET
MODBOOT1:MOV  BX,OFFSET  BOOT
        ADD  BX,0BH
        MOV  AL,ETRAC
        INC  AL
        SUB  AL,STRAC
        MUL  SNRMB
        ROL  AX,1
        MOV  WORD  PTR[BX+8],AX
        MOV  AL,SNUMB
        MOV  AH,0
        MOV  NORD  PTR[BX+13D],AX
MODBOOT2:MOV  BX,OFFSET  BOOT
        ADD  BX,21H
        MOV  AL,SNUMB
        MOV  BYTE  TR[BX+4],AL
        MOV  BYTE  PTR[BX+5],10
        CALL  SUBMOD
        MOV  CX,5
WRITBOOT:PUSH CX
        MOV  BX,OFFSET BOOT
        MOV  CX,0001H
        MOV  DX,0000H
        MOV  AX,0301H
        INT  13H
        POP  CX
        JNC  GREAT
        DEC  CX
        JNZ  WRITBOOT
        MOV  DX,OFFSET  MESS11
        MOV  AH,09H
        INT  21N
        CALL  SUBREC
        RET
GREAT: MOV  CX,512D
        MOV  SI,0
ZERO:  MOV  DOOT[SI],0
        INC  SI
        LOOP  ZERO
        MOV  SNUMB,1
        MOV  HFAD,1
        MOV  CX,11
GREATREP:PUSH  CX
        MOV  BX,OFFSET  BOOT
        MOV  CH,00H
        MOV  CL,SHUMB
        MOV  DH,HEAD
        MOV  DL,00H
        MOV  AX,0301H
        INT  13H
        MOV  AL,HEAD
        ADD  SHUMB,AL
        INC  HEAD
        AHD  HEAD,00000001B
        POP  CX
        LOOP  GREATREP
        MOV  BOOT,0FDH
        MOV  BOOT+1,0FFH
        MOV  BOOT+2,-FFH
        MOV  BX,OFFSET BOOT
        MOV  CX,0001H
        MOV  CX,0100H
        MOV  AX,0301H
        INT  40H
        MOV  BX,OFFSET BOOT
        MOV  CX,0002H
        MOV  DX,0100H
        MOV  SX,0301H
        INT  12H
        CALL  SUNREC
        RET
SUNROUT  ENDP
;.....
CODE    ENDS
;————————————————————————————
        END  MAIN
```